

МИНИСТЕРСТВО ОБРАЗОВАНИЯ
РЕСПУБЛИКИ БЕЛАРУСЬ

В.М.НОСОВ

**ПРОГРАММИРОВАНИЕ
НА ПЕРСОНАЛЬНЫХ ЭВМ
ЗАДАЧ ТЕОРЕТИЧЕСКОЙ
МЕХАНИКИ**

*Допущено Министерством образования
Республики Беларусь в качестве учебного пособия
для студентов инженерно-технических специальностей
высших учебных заведений*

**«ТЕХНОПРИНТ»
Минск 1997**

© Электронный учебник, В.М.Носов, 2003

Все права защищены. Никакая часть данной книги не может быть воспроизведена в какой бы то ни было форме без письменного разрешения владельца авторских прав.

Зарегистрирован в Государственном реестре информационных ресурсов Республики Беларусь.

Регистрационное свидетельство № 1180300285.

ББК 32.973–01 я73
Н 84
УДК 531:681.322–181.4(075.8)

Рецензенты:

заслуженный деятель науки и техники Российской Федерации
доктор технических наук, профессор **Яблонский А.А.**;
заведующий кафедрой технической механики Белорусского
государственного университета информатики и радиоэлектроники,
доктор технических наук, профессор **Сурин В.М.**;
кафедра теоретической и прикладной механики
Санкт-Петербургского университета,
доктор физико-математических наук, профессор **Товстик П.Е.**

Носов В.М.

Н 84 Программирование на персональных ЭВМ задач теоретической механики:
Учеб. пособие. – Мн.: ТЕХНОПРИНТ, 1997. – 367 с.: ил.

ISBN 985–6373–11–5 ТЕХНОПРИНТ

Пособие посвящено вопросам практической работы на персональных ЭВМ. Его можно рассматривать со специализированной и с универсальной точек зрения. В первом случае рассматривается использование готовых программ для решения задач по теоретической механике и оно является необходимым дополнением к широко используемому в ВУЗах и имеющемуся в их библиотеках сборнику заданий для курсовых работ под редакцией А.А. Яблонского (вышедшему в 4-х изданиях тиражом свыше 1 млн. экземпляров), из которого взяты все типовые примеры. С универсальной точки зрения пособие можно рассматривать как практикум по программированию, в котором для удобства пользования приводятся краткие необходимые сведения по алгоритмическому языку Фортран и по практической работе на ПЭВМ. Изложение ведется на рассматриваемых типовых примерах, чем достигается целостность восприятия. Описано самостоятельное составление программ из типовых блоков и примеров. Разнообразные дополнения позволяют самостоятельно составить более пятисот их модификаций. Все это делает пособие самоучителем работы на персональных ЭВМ. Пособие предназначено для студентов инженерно-технических специальностей вузов, инженерно-технических работников, а также для массового пользователя.

Н $\frac{5310000000-001}{97}$ Гос.заказ № 25 - 8/99 – 96. ББК 32.973–01 + 22.21 я73

ISBN 985–6373–11–5 ТЕХНОПРИНТ

© В.М.Носов, 1997

© Электронный учебник, В.М.Носов, 2003

**МОЕЙ
ЗАМЕЧАТЕЛЬНОЙ
ГЕРОИЧЕСКОЙ МАМЕ**

ЕЛЕНЕ МАТВЕЕВНЕ НОСОВОЙ

ПОСВЯЩАЕТСЯ

ПРЕДИСЛОВИЕ

Пособие посвящено вопросам практической работы на любых IBM-совместимых персональных ЭВМ на примерах решения задач и выполнения индивидуальных заданий по теоретической механике. Оно является продолжением серии из шести пособий [14], где весь материал рассмотрен для работы на “больших” ЭВМ. Поэтому отбор проводился так, чтобы все программы с минимальными типовыми переделками (или совсем без них) могли работать на любых IBM-совместимых ЭВМ (как персональных IBM PC, так и “больших” IBM-360/370, отечественные аналоги которых соответственно ЕС-1841–1863 и ЕС-1033–1066). Такой одинаковый подход к IBM-совместимым ЭВМ является отличительной особенностью пособия и представляется важным как с практической, так и с дидактической точек зрения. Одновременно с этим в пособии, кроме основ операционной системы MS DOS и удобных сервисных программ, рассматриваются и самые современные программные средства, такие, как MS Windows.

Пособие состоит из семи частей и 22-х глав. В первой части приводятся необходимые сведения для практической работы на IBM-совместимых персональных компьютерах. В чётных частях пособия (2, 4 и 6) рассматривается использование готовых программ для решения задач и выполнения индивидуальных заданий соответственно по разделам “Статика”, “Кинематика” и “Динамика” курса теоретической механики. Следующие нечётные части пособия (3, 5 и 7) посвящены составлению программ на Фортране для IBM-совместимых ЭВМ с использованием блочного подхода соответственно для тех же типовых примеров решения задач статики, кинематики и динамики.

При изучении курса теоретической механики с применением ЭВМ пособие является необходимым дополнением к широко используемому в ВУЗах и имеющемуся в их библиотеках сборнику заданий для курсовых работ [22], вышедшему в 4-х изданиях тиражом свыше 1 млн. экземпляров. Из него взяты все типовые примеры рассматриваемых задач, аналитические решения которых методами теоретической механики рассмотрены там же и в настоящем пособии не повторяются.

Вместе с этим пособие можно рассматривать и как практикум по программированию, в котором для удобства пользования приводятся краткие необходимые сведения по алгоритмическому языку Фортран (главы 7 и 12) и по прак-

тической работе на ПЭВМ (главы 1–3). С этой точки зрения все приводимые примеры носят обычный искусственный характер. При таком подходе изучаются только нечётные части пособия, любая из которых может рассматриваться независимо от других. Его может использовать любой, желающий овладеть алгоритмическим языком Фортран или уже использующий его для своей инженерной деятельности и имеющий выход на любую IBM-совместимую ЭВМ. С этой точки зрения 3-я часть посвящена решению систем линейных алгебраических уравнений с применением ЭВМ, 5-я — работе с подпрограммами на примерах численного дифференцирования, а 7-я часть — численному интегрированию дифференциальных уравнений. Все это делает пособие самоучителем работы на персональных ЭВМ для решения основных вычислительных задач.

Описываемые в пособии готовые программы для целей организации учебного процесса можно получить у автора вместе с необходимыми консультациями на кафедре теоретической механики БГПА, тел. (8-017) 2327425. Дополнительную помощь для преподавателей кафедр теоретической механики призвана оказывать организуемая Республиканским образовательным центром при БГПА специализация “Применение программирования и вычислительной математики в курсе теоретической механики” (По всем вопросам обращаться по тел. (8-017) 2399160, директор РОЦ БГПА Сапелкин Е.П.).

В заключение считаю приятным долгом выразить сердечную благодарность рецензентам, особенно заслуженному деятелю науки и техники РФ, д.т.н., профессору А.А. Яблонскому, к огромному сожалению ныне покойному. Его критические замечания и пожелания помогли автору в течение пяти лет, начавшись с двух длинных бесед и рецензии ко 2-й части методического пособия [14].

Также особую благодарность автор выражает рецензирующей кафедре теоретической и прикладной механики Санкт-Петербургского университета и ее заведующему д.ф.-м.н., профессору Товстику П.Е. за критические замечания и пожелания, высказанные при обсуждении работы, а также за помощь, советы и консультации, начавшиеся при прохождении ФПК на этой кафедре в 1989-м году.

В апробации настоящей работы в учебном процессе и проверке разработанных программ и дополнений в рамках УИРС принимали участие в течение десяти лет студенты машиностроительного и факультета дорожного строительства БГПА, а также слушатели семинара ФПКП БГПА.

Текст пособия подготовлен автором на машинном носителе. Он также принимал участие и заканчивал изготовление оригинал-макета с использованием программы Page Maker 6.0. Все программы и файлы данных помещались в оригинал-макет машинным образом и дополнительно тщательно проверялись.

Автор открыт для сотрудничества с издательствами, использующими цивилизованные формы работы.

Автор

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

В пособии ставится задача дать возможность студенту при изучении курса теоретической механики самому выбирать форму общения с ЭВМ от использования готовых программ, до их самостоятельного составления из приведенных готовых блоков. На каждом уровне возможен выбор нескольких вариантов использования ЭВМ для решения задачи:

1. Для реализации поставленной задачи для самого низкого уровня разработан пакет универсальных и специализированных программ по всем разделам курса:

- по статике (часть 2-я) студент должен подготовить исходные данные для выбранной им программы STAT, STATN, STAT9 или STAT9N (различающихся по степени универсальности и специализации для работы с разреженными СЛАУ с большим количеством нулевых элементов, к которым относятся задачи статики);
- по кинематике (часть 4-я) студентом подготавливаются исходные данные и одна (для работы программ KINMP и DKINMP) или две подпрограммы (для K9MP и DK9MP);
- по динамике (часть 6-я) студентом подготавливаются как исходные данные для работы специализированных программ DINSP или DDINSP, так и соответствующие подпрограммы для универсальных программ IDUSP и DIDUSP, представляющие интегрируемое уравнение в виде системы дифференциальных уравнений первого порядка. Это является следующим необходимым этапом перед самостоятельным написанием программы для интегрирования дифференциального уравнения.

Уже на этом уровне возможны элементы исследования, связанные с возможностью выбора используемой подпрограммы, реализующей различные численные методы, точности вычислений (простой или удвоенной), возможности варьирования условий задачи или различных параметров.

Специализированные программы по кинематике и динамике содержат правильные решения заданий из сборника А.А.Яблонского, для выполнения которых они предназначены. С ними сравнивается решение студента, что освобождает преподавателя от механической работы по проверке задания. Это позволяет организовать автономную самостоятельную работу студентов при выполнении заданий и акцентировать внимание на последующем ана-

лизе результатов расчета и выводах, а также на доведении первоначально полученного решения до правильного.

2. Написание простой программы по приведенным готовым блокам (с прохождением всех этапов обработки задания на ЭВМ) после повторения или изучения основных сведений из глав 7 и 12 по алгоритмическому языку Фортран.

3. Усложнение пункта 2 путем выбора способа задания ввода данных, выбора стандартной подпрограммы, сравнения результатов работы нескольких самостоятельно написанных программ после дальнейшего изучения материала пособия.

4. Углубленное изучение механики с применением ЭВМ после практического изучения всего приведенного материала: выполнение большего числа заданий и задач, приобретение дополнительных навыков программирования.

Для удобства пользования в пособии приведены краткие необходимые сведения для практической работы на ПЭВМ и по алгоритмическому языку Фортран, причем для каждого раздела механики все изложение производится с использованием одного типового примера задания из сборника [21]. Это означает, что все необходимые сведения по алгоритмическому языку Фортран, его практической реализации и конкретному использованию при работе на ПЭВМ приводятся применительно к этому примеру, чем достигается целостность восприятия.

Отличительной особенностью пособия является то, что кроме разнообразных основных программ, последовательно возрастающих по степени сложности для решения одних и тех же заданий по каждому разделу механики (10 программ по статике, 14 по кинематике и 12 по динамике), приводится также большое количество разнообразных дополнений (соответственно 10, 17 и 16), которые можно применять практически ко всем основным описанным программам. Это помогает восприятию материала, разбивает его на ряд ступеней по возрастающей сложности, что облегчает возможность самостоятельной проверки модификаций этих программ на ЭВМ. Студент конструирует применительно к собственному варианту задания конкретные модификации этих программ с использованием готовых блоков (около сотни возможных вариантов по статике и более двухсот вариантов и по кинематике и по динамике), наблюдая и осознавая влияние вносимых изменений на результаты работы программ. Это приучает студента к динамичной форме использования имеющихся у него знаний и навыков по видоизменению и приспособливанию программ для своих целей и задач.

Отметим, что выбор любой из программ и самостоятельное выполнение ее на ЭВМ с каким-либо дополнением даст некоторые навыки программирования, но полнота понимания достигается лишь разнообразием изученных

возможностей.

Все рассматриваемые типовые примеры взяты из двух последних изданий [21] и [22] широко и активно используемого в курсе теоретической механики сборника заданий под ред. проф. А.А.Яблонского. Это дает возможность не дублировать приведенные там аналитические выкладки для получения уравнений равновесия, дифференциальных уравнений движения и т.п., а сконцентрировать все внимание на различных возможностях их численного решения на ПЭВМ. Поэтому пособие может быть использовано как для организации учебного процесса с применением ПЭВМ при изучении курса теоретической механики, так и в качестве универсального практикума по программированию на Фортране. В последнем случае на источник получения рассматриваемых примеров обращать внимание не нужно.

Следует отметить, что реальность рассматриваемых задач, а также обычное параллельное их решение другими аналитическими способами при изучении курса теоретической механики, дает хорошее понимание задачи и создает возможность для анализа и проверки получаемых результатов. Естественное разделение курса теоретической механики на статику, кинематику и динамику обеспечивает возможность последовательного использования сначала материала, связанного с решением СЛАУ и матрицами, затем с подпрограммами и численным дифференцированием, затем с численным интегрированием систем дифференциальных уравнений.

Таким образом, с этой точки зрения пособие представляет собой профилирование курса программирования и вычислительной математики для нужд теоретической механики. Оно может служить основой для координации междисциплинарных связей и организации непрерывной подготовки по активному и согласованному использованию вычислительной техники при формировании будущих специалистов в ВУЗе.

Все рассматриваемые программы настоящего пособия, описанные для “больших” ЭВМ в работе [14], переделаны для ПЭВМ так, чтобы подготовка данных и основная последовательность действий сеанса работы оставались одинаковыми.

Соответствие между формальными параметрами, использующимися при описании подпрограмм, и фактическими, указывающимися при обращении к ним, происходит по их порядку следования, типу и количеству в списке аргументов. Однако для удобства пользования и простоты описания везде в пособии для фактических параметров используются обозначения, которые аналогичны или даже одинаковы с обозначениями формальных параметров. Условимся, что в этом случае они имеют тот же смысл для конкретных переменных или массивов. Поэтому те фактические параметры, обозначения которых совпадают с соответствующими формальными параметрами, исполь-

зованными при описании подпрограмм, повторно не рассматриваются.

Ссылки на главы, параграфы, подпараграфы и пункты обозначаются буквой “п” (например, п. 1, п. 1.2, п. 1.2.1, п. 1.2.1.1). Если материал понимается, то на них обращать внимание не надо.

В пособии используется также два вида ссылок при описании операторов и программ: на номер в круглых скобках и на номер без скобок.

Ссылка на номер в круглых скобках идентифицирует:

- весь приведенный сегмент фортран-программы, если в следующих строках данного программного модуля нет номеров со скобками;
- или только записи, находящиеся в этой же строке, если следующие операторы идентифицированы также номерами с круглыми скобками.

Нумерация с круглыми скобками сквозная в порядке возрастания по всему тексту пособия.

Ссылка на номер без скобок идентифицирует только сам оператор, находящийся в этой строке. Номера длинных операторов, расположенных в нескольких строках, указываются в первой или в последней строке.

Номера без скобок относятся к описываемым программам и проставлены только для удобства пояснений. В программе на бланке или на экране дисплея операторы не нумеруются.

При описании вариантов и дополнений к основным программам вставляемые операторы должны располагаться в основной программе по порядку возрастания номеров (без скобок), при этом:

- если номер (без скобок) вставляемого оператора совпадает с имеющимся в основной программе, то последний заменяется;
- если номер не совпадает — то оператор вставляется между операторами с соответствующими меньшим и большим номерами в основной программе.

При описании блоков программ (в главах 8, 13 и 14) они обозначаются номером со скобкой. Ссылка на них производится также с указанием цифры и скобки, например, блок 3).

Запись формул в пособии производится в форме соответствующих операторов присваивания фортран-программ (например для возведения в степень используются символы $**$). Переменные и углы в формулах обозначаются в форме их идентификаторов. При этом греческие буквы записываются в их латинской транскрипции, а для двухсимвольных переменных цифра или буквы I, J, K и N выполняют роль индекса (например X0, X1, TN, TK).

В главах 4-6 индекс также может обозначаться второй буквой (строчной)

за обозначением индексированной величины: реакция R в точке E будет обозначаться Re, в точке F — Rf; горизонтальные и вертикальные составляющие реакции в точке A будут указываться соответственно X_a, Y_a и т.д.; 1-й и 9-й элемент одномерного массива X — X₁ и X₉ соответственно, также используется обозначение X(1) и X(9). Там же значения углов тригонометрических функций при записи уравнений равновесия указаны в градусах.

ОСНОВНЫЕ ОБОЗНАЧЕНИЯ

ДУ	— дифференциальное уравнение,
ДУ2П	— дифференциальное уравнение второго порядка,
ИДФ	— идентификатор файла (его полное имя с указанием пути доступа к нему),
ИФ	— имя файла,
ОС	— операционная система,
ПЭВМ	— персональная ЭВМ,
СДУ	— система дифференциальных уравнений,
СДУ1П	— система дифференциальных уравнений первого порядка,
С2ДУ1П	— система двух дифференциальных уравнений первого порядка,
С2ДУ2П	— система двух дифференциальных уравнений второго порядка,
С3ДУ2П	— система трех дифференциальных уравнений второго порядка,
С4ДУ1П	— система четырех дифференциальных уравнений первого порядка,
С6ДУ1П	— система шести дифференциальных уравнений первого порядка,
СЛАУ	— система линейных алгебраических уравнений,
ССС	— система сходящихся сил,
ТФ	— тип файла,
ЭВМ	— электронная вычислительная машина.

ЧАСТЬ 1. КРАТКИЕ СВЕДЕНИЯ ДЛЯ ПРАКТИЧЕСКОЙ РАБОТЫ НА ПЕРСОНАЛЬНЫХ ЭВМ (ПЭВМ)

ГЛАВА 1. ОБЩЕЕ ОПИСАНИЕ ПЭВМ

Приведем необходимые сведения, достаточные для самостоятельной работы на IBM-совместимых ПЭВМ.

ПЭВМ конструктивно включает следующие основные устройства: системный блок, дисплей (монитор) и клавиатуру. К ним могут дополнительно подключаться: печатающее устройство (принтер) для выдачи текстовой и графической информации из компьютера на бумажный носитель, манипулятор типа “мышь”, облегчающий редактирование текстовой или графической информации, графопостроитель (плоттер), позволяющий выдавать на бумажный лист технические чертежи или любую графическую информацию и другие устройства.

Системный блок ПЭВМ обычно состоит из процессора, выполняющего вычисления и управление компьютером, блока дисководов (накопителей) для гибких магнитных дисков (НГМД), накопителя на жестком магнитном диске (НЖМД или винчестер). Визуально пользователь видит только вход накопителей на гибких магнитных дисках и индикатор накопителя на жестком магнитном диске. На передней панели системного блока компьютеров IBM PC также обычно находятся индикатор питания (POWER), кнопка TURBO (увеличение тактовой частоты процессора) и индикатор включения этого режима, индикатор обращения к жесткому диску (H.D.D. или H.Disk) и кнопка RESET для перезагрузки ОС. С тыльной стороны системного блока имеются разъемы, позволяющие подключать принтер, дисплей, клавиатуру и т.д.

Дисплей (монитор) предназначен для вывода на экран текстовой и графической информации. Может работать в одном из двух режимов: текстовом, устанавливаемом при включении ПЭВМ, или графическом (для вывода на экран рисунков и графиков). При работе в алфавитно-цифровом (текстовом) режиме все мониторы имеют идентичные характеристики: на экране размещается 25 строк по 80 символов в каждой строке. В качестве этих символов (общим числом 256)

входят не только большие и малые буквы латинского и русского алфавитов, цифры и символы, изображенные на клавиатуре, но и псевдографические символы, используемые для вывода на экран таблиц, рамок и диаграмм.

Клавиатура ПЭВМ является устройством ввода команд и данных в компьютер. Стандартная клавиатура IBM PC-совместимого компьютера содержит 101 (102) клавишу, расположение и число которых на различных моделях ПЭВМ может несколько отличаться друг от друга, но назначение соответствующих клавиш, конечно, совпадает. Их условно можно разбить на 4 группы. Для обозначения клавиш будем использовать соответствующие надписи на клавиатуре, заключенные в угловые скобки: <Надпись>.

1.1. Сравнительное описание клавиатур зарубежных и отечественных ПЭВМ

1. В центральном поле клавиатуры находятся алфавитно-цифровые и знаковые клавиши, расположенные примерно как на пишущей машинке.

2. Верхний ряд (над центральным полем) содержит функциональные клавиши <F1>–<F12>, обычно разделенные на три группы по 4 клавиши в каждой (в некоторых ПЭВМ они находятся в левой части и содержат 10 клавиш <F1>–<F10>). На компьютерах отечественного производства они обозначаются <Ф1>–<Ф12> или <Ф1>–<Ф10> соответственно. Поскольку каждая функциональная клавиша при нажатии передает уникальный (непечатаемый) код, прикладные программы могут присваивать специальные значения этим клавишам. После этого можно одним нажатием клавиши ввести в ЭВМ часто используемую сложную командную последовательность (дисковой операционной системы — DOS) или последовательность обычно используемых операций. Это сокращает объем и время работы, поскольку нажатием одной клавиши можно заменить необходимость громоздкого набора этих команд с клавиатуры. Отметим, что функциональные клавиши могут выполнять разные задачи для различных прикладных программ, поскольку не существует стандартов, определяющих присвоение программных функций каждой клавише.

3. Малое поле в правой части клавиатуры, где расположены цифровые и управляющие клавиши, образуют двухрежимную малую цифровую клавиатуру. При включенном режиме NumLock (Цифровая блокировка) эта самая правая группа клавиш дублирует цифровые клавиши, а при выключенном — клавиши управления перемещения курсора.

4. Слева и справа центральное светлое поле обрамляют стандартные управляющие клавиши. Рассмотрим их функциональное назначение и

соответствие для клавиатур ПЭВМ зарубежного и отечественного производства, которые будем разделять тире “—”. Основные модификации названий для каждой из клавиатур указываются через запятую.

<Enter>, <CR>, <Return> – <Ввод>, <↵> — ввод текущей строки и перевод курсора в начало следующей строки. Ввод каждой команды также должен оканчиваться нажатием клавиши <Enter> – <Ввод>.

<Esc> – <КЛЮЧ>, <Спец>, <ВЫХ> — сокращенное написание слова “Escape” (“Переход”). Отменяет текущие символы командной строки, которая в некоторых случаях при этом стирается. Используется для выхода из текущего режима в прикладных программах. В ряде других ситуаций выводит символ косой черты “\” в конце текущей строки и курсор перебрасывается в начальную позицию следующей строки.

<Shift> – <Верх>, <РЕГ>, <↑> — смена регистра (на время нажатия клавиши). Обычно используются для ввода прописных (заглавных) букв и специальных символов верхнего регистра клавиатуры, для чего нужно нажать клавишу <Shift> и, не отпуская ее, нажать клавишу с обозначением соответствующей буквы или символа.


<Caps Lock> – <Фикс Верх>, <ФПБ>, <ЗГЛ> — нажатие этой клавиши фиксирует клавиатуру в режиме прописных (заглавных) букв, что удобно при вводе заголовков или текста, состоящего из таких букв, так как теперь для их набора не требуется одновременного нажатия клавиши <Shift>. Отметим, что при нажатии клавиши <Shift> в режиме заглавных букв одновременное нажатие любой буквенной клавиши приводит к появлению на экране соответствующей строчной буквы нижнего регистра. При повторном нажатии клавиши <Caps Lock> происходит возврат к режиму строчных букв. Действие клавиши <Caps Lock> охватывает только клавиши с изображением букв. Для ввода же специальных символов верхнего регистра всегда нужно нажимать клавишу <Shift> совместно с соответствующей клавишей.

На компьютерах отечественного производства фиксация регистра русских или латинских букв осуществляется с помощью нажатия клавиш <РУС> или <ЛАТ>, сопровождаемое загоранием соответствующего светового индикатора. Если на клавиатуре ПЭВМ присутствует только одна клавиша <РУС>, то переключение алфавитов осуществляется после каждого ее нажатия. Для одноразового ввода буквы другого алфавита используется клавиша <Р/Л>, которая действует только при нажатии буквенной клавиши. На ряде клавиатур ПЭВМ клавиша <Р/Л> также используется для ввода специальных символов, указанных третьими справа по середине клавиш верхнего цифрового ряда.

На компьютерах зарубежного производства это переключение шрифтов определяется видом использованной специальной программы — драйвера

клавиатуры, который после нажатия соответствующей комбинации клавиш начинает передавать в компьютер символы другого алфавита. Обычно в качестве переключателя регистра используются клавиши, без которых в принципе можно обойтись. В зависимости от вида используемого драйвера для переключения алфавитов может быть использована клавиша <Caps Lock>, клавиша <F11>, правые клавиши <Ctrl>, <Shift> или одновременное нажатие обеих клавиш <Shift>+<Shift> (правой и левой), одновременное нажатие клавиш <Ctrl>+<Alt>, <Ctrl>+правый<Shift> (в этом случае <Ctrl>+левый<Shift> переключает клавиатуру в режим латинских букв) и т.п.

<CTRL>, <Control> – <УПР> и <ALT> – <ДОП>, <АЛТ> — (“Control” – “Управление” и “Alternate” – “Изменение”) используются только совместно с другими клавишами, изменяя их действие.

<TAB> – <ТАБ>,  > — (табуляция) перемещение курсора по строке на 8 позиций вправо на нижнем регистре и на 8 позиций влево на верхнем регистре (<Shift>+<Tab>).

<Back Space>, <BS>, <Back sp>, <BKSP> – <←> (стрелка влево над клавишей <Enter>), <ВШ> — удаление символа слева от курсора с одновременным смещением курсора на одну позицию влево.

<Ins>, <Insert> – <ВСТ> — переключение между двумя режимами ввода символов: вставка или замещение (замена). В режиме вставки курсор принимает форму мигающего квадрата, а не черточки, на месте которого появляется набираемый символ, сдвигающий всю остальную часть строки вправо.

, <Delete> – <УДП>, <УД>, <Удал> — (“Вычеркивание”) удаление символа в той позиции, в которой находится курсор (при этом все символы справа от курсора сдвигаются на одну позицию влево для заполнения освободившегося места).

<HOME> – <НАЧ>, <Начало> — Перевод курсора в начало строки.

<END> – <КОН>, <Кнц>, <Конец> — Перевод курсора в конец строки.

<PageUp>, <PgUp> – <СтрВв>, <СтрВверх>, <СТР↑> — Переход на предыдущую страницу изображаемого на экране текста.

<PageDown>, <PgDn> – <СтрВнз>, <СтрВниз>, <СТР↓> — Переход на следующую страницу изображаемого на экране текста.

Клавиши, обозначенные стрелками, используются для перемещения курсора по тексту влево, вверх, вниз и вправо.

<NUM LOCK> – <ЦИФ>, <БлкЦифр> — (“Цифровая блокировка”) — Включение/Выключение правой малой цифровой клавиатуры (вышеописанной в нецифровом режиме, начиная с клавиши <Ins> – <ВСТ>). В цифровом режиме каждая из клавиш рассматриваемой группы генерирует одну из цифр от 0 до 9-ти или десятичную точку соответственно нанесенному на ней обозначению.

1.2. Действия при “зависании” компьютера

Если в процессе работы ПЭВМ возникнет ситуация, что компьютер не реагирует на нажатия клавиш или реагирует на них неадекватно, то нужно выйти из состояния “зависания” одним из следующих возрастающих по силе действий:

1. Нажать <Esc> – <КЛЮЧ>.

2. Воспользоваться комбинацией <Ctrl>+<ScrollLock> – <УПР>+<ФД/СТОП>, следствием чего должно быть прекращение работы выполняемой команды или программы и перевод ПЭВМ в режим ожидания ввода команды с клавиатуры. Для набора комбинаций клавиш следует нажать одну из них и, не отпуская ее, нажать другую (или другие). Подобные совместные нажатия клавиш будут обозначаться с помощью знака “+” между ними.

3. Произвести перезагрузку, нажав одновременно <Ctrl>+<Alt>+ – <УПР>+<ДОП>+<УДП>.

4. Нажать клавишу <Reset> или <СБРОС> на корпусе системного блока (соответственно на ПЭВМ зарубежного или отечественного производства).

5. Выключить компьютер, а затем включить снова. При этом нужно помнить одно общее правило относительно системного блока: выключаться он должен первым перед выключением остальных внешних устройств, а включаться последним после включения всех внешних устройств ПЭВМ (т.о. при совмещении дисководов с системным блоком выключение ПЭВМ происходит в последовательности: системный блок — монитор, а включение в обратном порядке: монитор — системный блок, а при отдельном конструктивном выполнении дисководов выключение ПЭВМ происходит в последовательности: системный блок — дисководы, а включение в обратном порядке: дисководы — системный блок).

При включении и выключении ПЭВМ следует также убедиться в отсутствии в дисководах дискет, которые могут быть при этом испорчены. Отметим, что перед выключением ПЭВМ желательно ввести команду park, что приводит магнитные головки чтения-записи на жестком диске в нерабочее состояние. Для этого также можно воспользоваться сервисной программой PCSTOOLS (см. п. 3.2).

1.3. Диски

Диски предназначены для долговременного хранения информации. По своей конструкции они подразделяются на два класса: жесткие (винчестер) и гибкие. Диск представляет собой круглую пластину, покрытую слоем из магнитного материала (винчестер состоит из нескольких соосных

пластин), вращающуюся в процессе чтения или записи информации с большой скоростью. Перемещающийся по радиусу в непосредственной близости от рабочей поверхности блок магнитных головок осуществляет чтение или запись информации в нужную область диска, на котором предварительно создается определенная структура секторов и дорожек (см. описание команды FORMAT в п. 2.3). При форматировании диска (т.е. создании такой структуры) вся информация на нем будет утеряна.

Дорожки представляют собой концентрические окружности, имеющие сквозную нумерацию от внешнего края, разделенные на равные части емкостью по 512 байт, называемые секторами. Единица емкости запоминающих устройств 1 байт характеризует элемент памяти компьютера, в котором можно хранить 1 символ — букву, цифру или специальный знак (+, -, ;, %, и т.п.). 1 килобайт = 1 Кб = $2^{10} = 1\,024$ байт (примерно половина страницы текста), а 1 мегабайт = 1 Мб = $2^{20} = 1\,048\,576$ байт (примерно 500 страниц текста). Данные запоминаются на магнитной поверхности диска на дорожках. Количество данных, которое может храниться на диске, зависит от количества его рабочих поверхностей и плотности записи, определяемой количеством дорожек на поверхности диска и емкостью каждой дорожки (т.е. количеством на ней секторов, которое не фиксировано и задается при выполнении форматирования).

Жесткий диск или винчестер обычно выполняется несъемным и представляет вместе с блоком магнитных головок единую конструкцию в герметичном корпусе, заполненном инертным газом, называемую накопителем на жестком магнитном диске (НЖМД).

Гибкие диски, называемые также “флоппи-диски” или “дискеты”, позволяют хранить информацию отдельно от ПЭВМ и переносить ее с одного компьютера на другой. Они могут иметь различные физические параметры, отражаемые маркировкой, и размеры. Наиболее распространены дискеты размером 5,25 дюйма (133 мм) емкостью до 1,2 Мбайт и 3,5 дюйма (89 мм) емкостью до 1,44 Мбайт. Они могут быть односторонними (SS — Single Side) и двусторонними (DS — Double Side), одинарной (SD — Single Density), двойной (DD — Double Density), учетверенной (QD — Quadre Density), и высокой (HD — High Density) плотности записи.

Информационная емкость гибкого диска определяется не только его физическими параметрами, но и типом дисководов (накопителя на гибких магнитных дисках — НГМД) конкретного компьютера, а также используемыми драйверами (вспомогательными программами) и параметрами команды FORMAT (см. п. 2.3).

Емкость дискет, зависящая от их физических параметров, связана с маркировкой следующим образом:

- 5,25 дюйма, DS/DD — 360 Кб, (двусторонняя, двойная плотность),
- 5,25 дюйма, DS/HD — 1,2 Мб, (двусторонняя, высокая плотность),
- 3,5 дюйма, SS/HD — 720 Кб, (односторонняя, высокая плотность),
- 3,5 дюйма, DS/HD — 1,44 Мб, (двусторонняя, высокая плотность).

В настоящее время чаще всего используются дискеты размером 3,5 дюйма емкостью 1,44 Мбайта (DS/HD).

Дисководы для дискет емкостью 1,2 Мбайта используют магнитные головки чтения записи, обеспечивающие более узкую дорожку для записи информации, что позволяет делать соответствующее специальное магнитное покрытие этих дискет, которое труднее намагнитить и размагнитить. Это приводит к тому, что дискеты емкостью 1,2 Мбайта, как правило, не могут использоваться в дисководах емкостью 360 Кбайт, хотя дисководы для дискет емкостью 1,2 Мбайта обычно могут работать также и с дисками емкостью 360 Кбайт.

Дискеты диаметром 5,25 дюйма имеют небольшую толщину, являются самой чувствительной частью компьютерной системы и требуют бережного обращения: при постоянной неправильной эксплуатации выходят из строя, что приводит к порче имеющейся на них информации, а также могут явиться причиной повреждения самого дисковода. Поэтому для защиты от внешних воздействий каждый диск помещают в защитную оболочку — несъемный квадратный пластиковый конверт, на верхней поверхности которого отсутствуют швы и в углу имеется фирменная наклейка изготовителя с соответствующей маркировкой, которая должна быть сверху при вставлении гибкого диска в дисковод.

Дискеты диаметром 3,5 дюйма (89 мм) заключены в жесткий пластмассовый конверт, что значительно повышает их надежность и долговечность.

Имена накопителей на дисках (дисководов)

Отметим, что НГМД в MS DOS имеют обычно имена А: и В:, причем А: соответствует первому (главному) накопителю (нижний или левый дисковод). Чтобы облегчить управление огромной памятью НЖМД (винчестера), ее для удобства работы подразделяют на несколько томов или разделов — условных дисков, каждый из которых имеет свою собственную метку наподобие дисковода, которым присваивают имена С:, D: и т.д.. При этом машина не делает различия между разделом накопителя на жестком диске и накопителем на дискетах.

ГЛАВА 2. ОСНОВЫ ОПЕРАЦИОННОЙ СИСТЕМЫ DOS

Операционная система (ОС) служит для создания среды, в которой происходит диалог пользователя с ЭВМ. Она представляет собой совокупность программ, осуществляющих перевод вводимых команд на язык, понятный для ЭВМ. Существует три разновидности операционной системы DOS, под управлением которой работает IBM-совместимый персональный компьютер (MS DOS, PC DOS и DR DOS). Из-за несущественности различий при первоначальном практическом использовании этих систем они обычно [26] обозначаются одинаково (просто DOS) и описываются одновременно.

2.1. Основные термины и понятия

Файл (от слова “File” — “Массив”) — это именованный упорядоченный массив однородной информации, размещенной на внешнем носителе (жестком или гибком диске, магнитной ленте), имеющей определенное функциональное назначение и воспринимаемой операционной системой как единое целое. В виде файлов на диске хранится вся информация. Файлом может быть текст программы на алгоритмическом языке, например, Фортран; набор исходных данных; объектный модуль программы, пригодный для ее понимания ЭВМ и т.п.

Имя файла состоит обычно из собственно имени (filename) и расширения (extension, часто называемого типом файла), разделенных между собой точкой. Имя ИФ и тип файла ТФ могут содержать от 1 до 8 и от 0 до 3 символов соответственно, в качестве которых могут использоваться алфавитно-цифровые и некоторые специальные символы.

Наличие расширения в полном имени файла не является обязательным, однако оно обычно используется для описания типа информации, записанной в файле, т.е. для описания типа файла. Обычно файлы, относящиеся к одной основной программе, имеют одинаковые имена, но разные типы. Для некоторых типов информации обычно применяют стандартные или типовые расширения:

- .bat — командный файл DOS для пакетной обработки, представляющий собой последовательность команд, выполняемых автоматически;
- .com — команда или программа в машинном коде, пригодные для непосредственного выполнения под управлением DOS (командный файл);
- .dat — файл данных;

- .exe — перемещаемая программа, готовая к выполнению под управлением DOS;
- .for — исходная программа на языке Фортран;
- .obj — скомпилированная объектная программа на машинном языке (т.е. во внутренних кодах ПЭВМ);
- .sys — системные файлы, являющиеся частью операционной системы и используемые для расширения ее возможностей (например, драйверы), и др.

Каталог (Directory) — это таблица содержимого диска или его части, в которой находится список имен файлов, объединенных по некоторому критерию, с указанием их атрибутов (размера, даты и времени последней зарегистрированной модификации каждого файла). Каталоги файлов создаются для удобства организации файловых систем, так как каждый каталог можно рассматривать как раздел внешней памяти, с содержимым которого можно работать достаточно независимо. При создании каждого файла его имя, тип и другие атрибуты записываются в каком-то каталоге. При этом говорят, что файл находится в данном каталоге. Для создания нового каталога на диске в DOS используется команда *MKDIR* (или *MD*).

Имена каталогов подчиняются тем же требованиям, что и имена файлов, однако возможность расширения имени для каталогов обычно не используется. Имена каталогов указываются в оглавлении большими буквами, тогда как имена файлов маленькими.

Корневой каталог, не имеющий имени, всегда имеется на каждом магнитном диске. В нем регистрируются файлы и каталоги (называемые в этом случае обычно подкаталогами). Корневой каталог обозначается с помощью указателя дисководов, за которым следует обратная наклонная черта “\” (в некоторых версиях DOS вместо знака “\” используется знак “/”). Например, A : \ — корневой каталог гибкого диска на дисковом диске A, C : \ — корневой каталог жесткого диска (или его раздела — условного диска), расположенного на дисковом диске C.

Многоуровневые каталоги. В каждом подкаталоге (или каталоге 1-го уровня) корневого каталога, кроме находящихся в нем файлов, также можно создать произвольное количество других подкаталогов или каталогов 2-го уровня, для которых каталог 1-го уровня будет надкаталогом или родительским каталогом. В каждом созданном подкаталоге (или каталоге 2-го уровня) также можно создать произвольное количество других подкаталогов или каталогов 3-го уровня и т.д. Каждый подкаталог представляет собой обычный файл, содержащий список имен относящихся к нему других файлов, часть которых (или все) также может быть в свою очередь подкаталогами, содержащими соответственно свои файлы и другие подкаталоги.

Цепочки включенных друг в друга каталогов обозначаются их именами, разделяемыми знаком обратной наклонной черты “\” (backslash). Если знак “\” стоит перед первым именем каталога, значит выше “по иерархии” находится только корневой каталог данного диска. В этом случае знак “\” как бы отделяет несуществующее имя корневого каталога от имени каталога первого уровня, подчиненного корневному. Примеры цепочек подчиненных каталогов: \ТЕОР_МЕС — каталог ТЕОР_МЕС находится на 1-м уровне; \ТЕОР_МЕС\UPF — каталог UPF находится на 2-м уровне; \ТЕОР_МЕС\UPF\СТАТИКА — каталог СТАТИКА находится на 3-м уровне.

Так образуется иерархическая древовидная файловая структура, называемая многоуровневой системой каталогов. Слово “иерархическая” ни в каком случае не следует понимать буквально. Ни одному каталогу не присваивается приоритет по отношению к другому. Каждый каталог любого уровня также можно рассматривать как раздел внешней памяти, с содержанием которого можно работать абсолютно независимо, поэтому употребляемые часто обозначения “подкаталог”, “надкаталог” или “родительский каталог” имеют только значение для указания их относительного расположения.

Текущий или рабочий каталог. Каталог (или подкаталог), в котором вы находитесь, называется текущим или рабочим каталогом. В каждый данный момент во внимание принимается только рабочий или текущий каталог, который и становится главным на время работы в нем. Когда вы создадите файл или подкаталог, он помещается в текущий каталог. Можно представить, что каждый диск разделен на столько секций, сколько в нем подкаталогов. Командам доступны только файлы, содержащиеся в текущем подкаталоге, чем достигается возможность независимой работы разных пользователей. В текущем подкаталоге также должны быть расположены все файлы и подпрограммы, необходимые для работы имеющихся там же программ (или в команде *PATH* должны быть указаны пути доступа к соответствующим каталогам, в которых содержатся необходимые файлы).

Пути и полные имена файлов. Принцип многоуровневой организации памяти на диске приводит к тому, что для указания файла уже недостаточно указать только его имя с соответствующим расширением: одноименные (и притом различные по содержанию) файлы могут находиться в нескольких каталогах. Для точной идентификации файла необходимо указать механизм определения местоположения файла в многоуровневой системе каталогов, в качестве чего указывается путь доступа к файлу. Он представляет собой маршрут по именам каталогов, содержащий последовательный указатель перечня имен каталогов на пути к файлу, разделенных между собой символом “\”.

Путь может начинаться или от корневого каталога, или от текущего. В зависимости от этого маршрут поиска называется соответственно абсолютным или относительным. Если путь доступа начинается с обратной косой черты “\”, то DOS осуществляет поиск файла, начиная с корневого каталога. В противном случае она просматривает указанный путь в поисках файла, начиная от текущего (рабочего) каталога.

Путь доступа к файлу stat9n.for, находящемуся в каталоге 3-го уровня СТАТИКА, из корневого каталога можно указать так:

```
\TEOR_MEC\UPF\STATIKA\stat9n.for (2.1)
```

Если файл расположен на жестком диске (диск C), а текущим является диск A, то полное описание файла, требуемое для доступа к файлу stat9n.for, выглядит следующим образом:

```
C:\TEOR_MEC\UPF\STATIKA\stat9n.for (2.2)
```

Путь с включенным в него именем файла полностью идентифицирует расположение файла на диске и образует полное описание файла или идентификатор файла (ИДФ). Отметим, что имя файла также должно быть отделено обратной косой чертой от последнего имени подкаталога. Полное описание файла имеет следующий формат:

```
[диск:] [путь\]имя_файла[.расширение] (2.3)
```

Здесь элементы, заключенные в квадратные скобки, являются необязательными. В этом случае, если в описании не указывается диск C, то подразумевается текущий диск C или текущий каталог. Так в примере (2.1) описатель дисковода не указан, так как текущим является диск C и маршрут поиска указывается от корневого каталога, а при работе в текущем каталоге СТАТИКА идентификатором находящегося в нем файла ИДФ будет только имя файла с его расширением (если оно имеется, в нашем примере stat9n.for). Последняя возможность очень удобна. Поэтому всегда в дальнейшем будем подразумевать, что работаем с файлами, находящимися в текущем каталоге.

Подстановочные символы “*” и “?” разрешается использовать в обозначениях имен файлов и их расширений для обозначения сразу нескольких файлов, что называется шаблоном имени файла (“групповым”, “родовым” именем или обозначением имени по маске). Знак “*”, встречаемый в имени или типе файла, обозначает любое число произвольных символов, которые допускаются в именах и типах файлов, количество которых ограничено диапазоном от нуля (т.е. отсутствует вовсе) до значения, дополняющего число указанных в шаблоне символов до их максимально допустимого значения (8 для имени и 3 для типа). Вопросительный знак, встречающийся в имени или типе файла, соответствует любому одному произвольному допустимому сим-

волу или его отсутствию. Примеры шаблонов: `ABC*.D*` — определяет подмножество файлов из существующих на диске в текущем каталоге, имена которых начинаются с `ABC` и имеют длину от 3-х до 8-ми символов, и расширениями, начинающимися с `D` и имеющими длину от одного до 3-х символов; `*.*` — все файлы текущего каталога, у которых отсутствует тип; `A??B.*` — все файлы текущего каталога, начинающиеся с `A`, заканчивающиеся `B` и имеющие длину от 2-х до 4-х символов.

Обратим внимание, что DOS игнорирует все символы в имени и типе файла после подстановочного символа звездочки, поэтому обозначения `*abc.*d` и `*.*` трактуются одинаково, как все файлы текущего каталога.

Дополнение 2.1. Внешние и внутренние команды DOS, утилиты и драйверы устройств.

Командный процессор `COMMAND.COM` обрабатывает и исполняет команды, вводимые пользователем с клавиатуры или полученные при выполнении командного файла, а также осуществляет исполнение файла автозапуска `AUTOEXEC.BAT` для начальной настройки ОС при ее запуске. Он является обычным файлом, который может занимать любое место на системном диске, с которого загружается ОС. Простейшие, наиболее часто используемые команды, являются внутренней частью файла `COMMAND.COM` и не отражаются в каталоге системного диска (поэтому их называют внутренними командами). К ним относятся `CHDIR`, `COPY`, `DEL (ERASE)`, `DIR`, `MKDIR`, `PATH`, `REN`, `RMDIR`.

Любое имя файла с расширением `.com`, `.exe` или `.bat` рассматривается как внешняя команда. Для их выполнения процессор ищет на дисках программу с соответствующим именем (сначала в текущем каталоге, а затем в каталогах, перечисленных в команде `PATH`, обычно включаемой в файл `autoexec.bat`), и, если находит ее, то загружает в память и передает ей управление. Например, такие файлы, как `format.com` и `diskcopy.com` являются стандартными внешними командами. Их часто также называют утилитами DOS и размещают на системном диске в отдельном каталоге `\DOS`. Так как любой файл с расширением `.com`, `.exe` или `.bat` может выполнять роль внешней команды наряду со стандартными утилитами, то это дает возможность пользователю самостоятельно создавать необходимые для него новые команды. При вводе любой внешней команды достаточно указать только имя файла (расширение указывать нет необходимости). Однако, если у Вас на диске имеются несколько внешних команд с одинаковыми именами, то DOS будет выполнять только одну из них в соответствии со следующим порядком приоритетов: `.com`, `.exe`, `.bat`.

Для подключения новых или нестандартного использования имеющихся устройств применяются специальные программы, являющиеся также внешними командами и называемые в этом случае драйверами устройств.

Они обрабатываются и загружаются в память ПЭВМ при загрузке ОС (если их имена указаны в файле `config.sys`), или по мере необходимости перед их непосредственным использованием (см., например, расширение возможностей команды *FORMAT* (2.17) после обработки соответствующих драйверов).

Дополнение 2.2. Принимаемый по умолчанию дисковод. Приглашение DOS или системная подсказка.

После успешного окончания загрузки на экране появляется приглашение DOS к вводу команд, стандартный вид которого содержит только идентификатор текущего дисковода, обозначаемого одной из букв латинского алфавита, с последующим знаком “>” (“больше”), например: `A>` или `C>`, указывающие соответственно на текущий дисковод `A:` или `C:` (иногда также называемый принимаемым по умолчанию дисководом). Появление приглашения на экране означает, что DOS готова к приему команд пользователя. Отсутствие приглашения говорит о том, что пользователь общается не с DOS, а с какой-либо прикладной программой или транслятором.

С помощью команды *PROMPT* [6, с. 65–67], включаемой в состав файла `AUTOEXEC.BAT`, обычно изменяют приглашение, включив в него, например, указание на текущий каталог, откуда задаются команды: `C:\TEOR_MEC\UPF\СТАТИКА>`. При содержании приглашения в виде указания на текущий (рабочий) каталог, она определяет для ПЭВМ и пользователя указание на текущий диск или путь к текущему каталогу.

Дополнение 2.3. Командные файлы.

Для ввода часто повторяющейся последовательности команд используются специальные файлы, имеющие расширение `.BAT` и называемые командными файлами. Нужная последовательность команд помещается в образываемый командный файл с помощью любого текстового редактора либо путем непосредственного копирования с клавиатуры. В последнем случае вводятся следующие команды.

1. Набрать команду `COPY CONимя_файла.bat`.
2. Набрать последовательно строки команд, которые вы хотите включить в файл `имя_файла.bat` (после каждой строки `<Enter>` (`<ВВОД>`)), в результате чего очередная строка с командой помещается в файл.
3. Нажать одновременно две клавиши `<Ctrl>+<Z>` (`<УПР>+<Z>`), являющиеся признаком конца файла, потом `<Enter>` (`<ВВОД>`), чтобы сохранить этот командный файл.

Указанная последовательность команд, записанная в файле с расширением `.bat`, будет выполнена после ввода из командной строки имени этого файла (при этом его расширение можно не указывать). Эта процедура на-

зывается пакетной обработкой и позволяет использовать командные файлы для создания собственных команд, что широко используется при трансляции программ, их сборке и т.п. (см. п. 8.4).

Особую роль для создания удобной рабочей обстановки для пользователя ПЭВМ играют два специальных командных файла `config.sys` и `autoexec.bat`. Данные файлы должны находиться в корневом каталоге системного диска. Они отрабатываются каждый раз при любом включении или перезагрузке системы и производят конфигурирование и начальную настройку системы.

С помощью файла конфигурации `config.sys` можно расширять операционную систему, добавлять новые внешние устройства или нестандартно использовать имеющиеся. Эти программы, называемые драйверами внешних устройств, можно включить в систему при загрузке ОС, инициализировав их из файла `config.sys`.

Файл автозапуска `autoexec.bat` содержит набор последовательных команд и является обычным пакетным командным файлом, выполняющим необходимую настройку системы. Например, можно включить в приглашение DOS указание на текущий каталог, задавать путь поиска внешних команд с помощью команды *PATH* и т.д. В списке каталогов, задаваемых командой *PATH*, обычно перечисляются через точку с запятой те каталоги, в которых находятся исполняемые программы общего назначения. Рекомендуется [26] имена каталогов в команде *PATH* указывать полностью от корневого каталога соответствующего диска, что позволит командному процессору DOS правильно их находить из любого текущего каталога и диска, например:

```
PATH C:\NC;C:\F5\INCLUDE;C:\F5\LIB;C:\F5\BIN;C:\F5\TMP (2.4)
```

2.2. Основные сведения о командах DOS

2.2.1. Соглашения об обозначениях форматов команд

Здесь и далее по тексту при записи форматов команд приняты следующие правила:

- ключевые слова, записанные латинскими буквами, должны использоваться в команде без изменений и сокращений;
- параметры, записанные русскими буквами, должны быть заменены необходимыми конкретными значениями в соответствии с описанием команды;
- атрибуты команды или часть ее имени, заключенные в квадратные скобки “[...]”, являются необязательными элементами и могут быть опущены при использовании команды;

- фигурные скобки “{...}”, ограничивающие собой какое-либо используемое обозначение или его часть, даже при использовании латинского алфавита указывают на необходимость его замены во всех случаях соответствующими конкретными значениями (обычно числами). Сами символы фигурных скобок указывают только на необходимость замены заключенных в них выражений в формате команды при ее описании их конкретными значениями, поэтому в реальной команде они не указываются. Например, по описанию ИФ.D{NW}, где {NW} — номер варианта, следует в данной команде подставить конкретное имя файла, а в обозначении типа после латинской буквы D поставить номер решаемого варианта, что для имени файла STAT9N и {NW}=22 будет иметь вид: STAT9N.D22.
- сокращения используются для указания в форматах команд часто встречающихся переменных значений, например: ИДФ — идентификатор файла, ИФ — имя файла, ТФ — тип файла.
- группа слов, определяющих одну переменную, связывается символом “_”.

2.2.2. Структура команд DOS

Будем считать все командные файлы находящимися в каталоге DOS диска C, путь к которому указан в команде *PATH*, включенной в состав файла *autoexec.bat*. Тогда для запуска любой команды достаточно будет указать только имя соответствующего файла, обозначаемого нами далее как ИМЯ_КОМАНДЫ. Тогда формат любой команды DOS в общем случае можно представить в виде:

ИМЯ_КОМАНДЫ [ОПЕРАНДЫ] [/РЕЖИМЫ] (2.5)

где ОПЕРАНДЫ и РЕЖИМЫ (или ОПЦИИ) могут включать в себя следующее: идентификатор файла, над которым данной командой совершается определенное действие, аргумент и переключатель.

Идентификатор файла ИДФ (2.3) следует в команде сразу за указанием ее имени и относится к числу позиционных операндов, которые должны задаваться в определенной последовательности.

Будем считать все обрабатываемые файлы (т.е. те, с которыми нам придется работать) находящимися в текущем каталоге текущего диска. Тогда идентификатором файла {ИДФ}, однозначно определяющим его местоположение, будет только имя файла (при необходимости с указанием соответствующего типа), а диск и путь можно будет не указывать.

Опция “аргумент” предоставляет команде некоторую дополнительную информацию. Обычно приходится выбирать один из нескольких вариантов значений аргумента, например, ON (ВКЛ) или OFF (ВЫКЛ), который также указывается после имени команды и относится к числу позиционных операндов.

Режимы выполнения команды, называемые также переключателем, начинаются с символа косой черты “/”, за которой сразу без пробела указывается соответствующий режим. Они представляют собой ключевые операнды и могут указываться в произвольной последовательности, например, /f/g/q/s. Часть их принимается по умолчанию, поэтому указание режимов при обычном действии команды необязательно.

Все вводимые команды печатаются в командной строке после соответствующего приглашения DOS и после нажатия клавиши <Enter> – (<ВВОД>) вместе с ним вводятся в ПЭВМ.

2.2.3. Перенаправление потоков ввода и вывода

Устройствами, принимаемыми по умолчанию, в DOS для ввода является клавиатура, а для вывода — экран. Однако для практической работы на ПЭВМ при подготовке исходных данных удобнее помещать их в отдельный файл, возможность записи которого на диске позволит облегчить последующий поиск ошибок и доведение первоначального решения до правильного. Это можно сделать посредством использования в команде знака “меньше” (<). Также довольно часто удобнее направлять свой вывод в файл, из которого его можно будет распечатать в удобное время. Для этой цели используется знак “больше” (>). Например, по команде `stat9n_.exe <wm412c9.d22> wm412c9.I22` исходные данные для работы программы `stat9n_.exe` будут взяты из предварительно подготовленного файла `wm412c9.d22`, а результаты расчета помещены в файл `wm412c9.I22`.

2.3. Основные команды DOS

Использующему ПЭВМ специалисту для практической работы на ПЭВМ с использованием сервисных программ желательны понятия о следующих наиболее часто встречающихся командах:

- 1) создание нового каталога на диске (*MKDIR* или *MD*);
- 2) переход в другой каталог (*CHDIR* или *CD*);
- 3) удаление каталога с диска (*RMDIR* или *RD*);
- 4) просмотр дерева подкаталогов на диске (*TREE*);
- 5) просмотр содержимого файла (*TYPE*);
- 6) копирование файлов (*COPY*);
- 7) удаление файлов из каталога (*DEL* или *DELETE*);
- 8) изменение имени файла (*REN* или *RENAME*);

9) подготовка дискеты к использованию (*FORMAT*);

10) копирование с дискеты на дискету (*DISKCOPY*).

Команды 1) – 4) предназначены для работы с каталогами, 5) – 8) для работы с файлами, а 9) – 10) представляют собой команды для работы с дисками:

Команда MKDIR или **MD** производит создание нового каталога на диске. Формат команды *MD*:

```
MD [диск:] [путь\]имя_каталога (2.6)
```

Создание нового каталога может быть произведено в любом уже существующем каталоге на указанном [диск:] или текущем диске в указанном [путь\] или текущем каталоге. Если вы в команде *MD* не указываете [диск:], то каталог создается на текущем диске по указанному пути. Если не указан и путь: *MD имя_каталога*, то каталог с указанным именем создается в текущем каталоге (в котором вы находитесь) и будет являться его подкаталогом.

Команда CHDIR (CD) выводит на экран имя текущего каталога или осуществляет переход в другой каталог (т.е. изменяет текущий каталог). Формат команды *CD*:

```
CD [диск:] [путь\]имя_каталога (2.7)
```

При переходе из текущего каталога в подчиненный в команде *CD* достаточно указать его имя. Для перехода в каталог предыдущего уровня [родительский каталог] используются символы “..” [две точки]: *CD ..*, а для перехода в корневой каталог текущего диска достаточно ввести команду *CD *.

Команда RMDIR (RD) удаляет указанный каталог или подкаталог. Формат использования команды:

```
RD [диск:] [путь\]имя_каталога (2.8)
```

Перед удалением каталога (или подкаталога) нужно удалить в нем все файлы и подкаталоги (он должен быть пуст). Нельзя использовать *RMDIR* для удаления текущего каталога (нужно перейти сначала в другой каталог).

Команда TREE графически показывает на экране структуру каталога. Формат использования команды:

```
TREE [диск:] [путь\] [/F] (2.9)
```

Параметр [диск:][путь\] задает диск и маршрут, для которого вы хотите вывести структуру каталога. Параметр /F выводит имена файлов в каждом каталоге. Для вывода имен всех подкаталогов на текущем диске дайте команду *TREE *.

Команда TYPE используется для вывода на экран содержимого указанного текстового файла. Вывод можно приостановить нажатием клавиш <Ctrl>+<S> и продолжить их повторным нажатием. Формат использования команды:

```
TYPE [диск:] [путь\]имя_файла (2.10)
```

Команда *COPY* осуществляет копирование файлов. Формат использования команды:

```
COPY ИДФ1 [ИДФ2] [/ключ] (2.11)
```

Здесь ИДФ1 задает источник копирования: имя файла (или файлов — при этом допускается использование шаблона) или имя каталога, если необходимо скопировать все файлы каталога. Если ИДФ2 — назначение — не задано, то файлы копируются в текущий каталог с сохранением имен. Если ИДФ2 — имя каталога, то файлы копируются в указанный каталог также с сохранением имен. Если ИДФ2 — имя файла, то источник копируется в файл с указанным именем. Например, команда *COPY stat9n.dat stat9n.d22* позволяет создать копию существующего файла *stat9n.dat* под идентификатором *stat9n.d22* в том же текущем каталоге.

Обратим внимание, что команда *COPY* всегда выполняется в режиме замены существующего файла (если он есть на диске) выводным файлом, если у них одинаковые идентификаторы.

В командах DOS значения неуказанных операндов или их частей выбираются системой принимаемыми по умолчанию. Поэтому для копирования всех файлов с жесткого диска C на A без изменения имен и типов файлов достаточно ввести *COPY C: A:*.

При копировании допускается использование логических устройств в качестве источника или назначения копирования, например:

CON — консоль (клавиатура — при вводе, дисплей — при выводе), что позволяет непосредственно вводить в нужный файл необходимую информацию с клавиатуры (*COPY CON имя_файла*) или получать вывод файла на экран (*COPY имя_файла CON*). Отметим, что ввод информации в файл осуществляется построчно после ее набора в командной строке и нажатии клавиши <ВВОД>, а символ признака конца создаваемого файла (код 26) записывается в файл при одновременном нажатии клавиш <Ctrl>+<Z> (на экране этот символ отображается в виде ^Z);

PRN — принтер (только как выходной файл):

```
COPY имя_файла PRN (2.12)
```

Последняя возможность позволяет удобно распечатывать файлы с использованием оболочки Norton Commander (см. п. 3.1), что обычно удобнее применения команды *PRINT*.

Команду *COPY* можно также использовать для комбинирования файлов. В этом случае формат использования команды имеет вид:

```
COPY ИДФ1 [+ИДФ2+...] [ИДФ] (2.13)
```

Источник представляет собой перечень файлов ИДФ1 [+ИДФ2+. . .], соединенных знаком “+” (плюс), а приемником (или файлом назначения) является идентификатор выводного файла [ИДФ]. Чтобы скопировать несколько файлов в один, перечислите в источнике любое число файлов (разделив их плюсом) и задайте имя выводного файла: *COPY stat.d1+stat.d2+stat.d3 stat.dsu*. При этом файлы текущего диска и каталога *stat.d1*, *stat.d2* и *stat.d3* объединяются и помещаются в файл *stat.dsu* (также в текущем каталоге).

Допускается использование подстановочных символов. Команда *COPY stat.d* stat.dsu* комбинирует все файлы в текущем каталоге с именем *stat* и расширениями, начинающимися с *d* и имеющими длину от одного до 3-х символов, в один файл *stat.dsu*.

Команда *DEL* или *DELETE* удаляет файл имя_файла из текущего или указанного каталога. Формат использования команды:

```
DEL [диск:] [путь\]имя_файла [/P] (2.14)
```

Ключ */P* перед удалением файла выводит запрос на подтверждение. Например, чтобы удалить файл *stat9n.dat* из текущего каталога, вы можете воспользоваться командой: *DEL stat9n.dat*.

Команда *REN* или *RENAME* изменяет имена заданных файлов (файла). Не допускается применять команду *REN* для переименования файлов с указанием другого диска или для перемещения файлов в другой каталог. Формат использования команды:

```
REN [диск:] [путь\]имя_файла1 имя_файла2 (2.15)
```

Параметр *[диск:] [путь\]имя_файла1* задает расположение файла или набора файлов, которые нужно переименовать. Параметр “*имя_файла2*” задает новое имя файла (или новые имена файлов при использовании шаблона). Новый диск и маршрут вы указать не можете. Например, команда *REN stat9n.dat stat9n.d22* позволяет изменить имя существующего файла *stat9n.dat* на *stat9n.d22* без изменения содержания в том же текущем каталоге.

Если файл “*имя_файла2*” уже существует, *REN* работать не будет, и выводится аварийное сообщение: “*Duplicate file name or file not found*” (“Имя файла дублируется, или файл не найден”).

Команда *FORMAT* производит подготовку (специальную разметку) гибких дисков [диск:] к дальнейшему использованию в DOS. Все новые диски необходимо предварительно форматировать. Если на диске была записана какая-либо информация, то после форматирования она будет безвозвратно потеряна. Основной формат команды:

```
FORMAT диск: [/ключ] (2.16)
```

Команда *FORMAT* (например, *FORMAT A:*) по типу дисководов определяет нужные параметры и на какую емкость будет отформатирован диск на данном дисководе (для дискет 5.25-дюйма на 360 Кбайт или 1.2 Мбайта, что также удобно выполнить с помощью оболочки PCSTOOLS — см. п. 3.2).

В команде имеются многочисленные ключи, позволяющие изменять параметры форматирования по умолчанию (обычно в сторону их увеличения) после отработки соответствующих драйверов (например, 800.COM или 900.COM). Так дискета, формируемая по вышеприведенной команде (*FORMAT A:*) на емкость 360 Кбайт, будет отформатирована на 720 Кбайт после указания нужного количества секторов (/t:сект) и дорожек (/n:дор) в команде

```
FORMAT A: /t:80 /n:9 (2.17)
```

Другие значения параметров команды *FORMAT* можно узнать после запуска драйвера 900.com в режиме запроса: 900.com /?, причем перед использованием таких нестандартно отформатированных дискет также должен отработаться соответствующий драйвер (если его запуск не включен в файл autoexec.bat).

Команда *DISKCOPY* [диск1:] [диск2:] производит копирование (по дорожкам) всей информации с указанного [диск1:] на [диск2:]. Если копирование производится на одном и том же устройстве (например, *DISK COPY A: A:*), то на экран выводятся сообщения о необходимости неоднократной замены гибких дисков. Оба диска должны быть одного и того же формата, т.е. содержать одинаковое количество дорожек и секторов. Если диск, на который производится копирование, не форматирован, то он автоматически форматировается с параметрами диска источника.

ГЛАВА 3. ПРАКТИЧЕСКАЯ РАБОТА НА ПЭВМ С ИСПОЛЬЗОВАНИЕМ СЕРВИСНЫХ ПРОГРАММ (NORTON COMMANDER, PCTOOLS) И MICROSOFT WINDOWS

Для удобства практической работы в DOS разработаны специальные операционные оболочки, позволяющие за счет некоторого сокращения широких возможностей DOS существенно увеличить простоту и наглядность выполнения большинства наиболее употребительных ее команд. Вся предыдущая глава нужна в основном для понимания того, что будут делать эти программы-оболочки, а также для оценки достоинств работы с ними. К таким наиболее популярным сервисным программам относятся Norton Commander и PCTOOLS (Central Point Software).

3.1. Программа Norton Commander (NC)

Запуск Norton Commander осуществляется набором в командной строке NC. Эта команда обычно включается в файл автозапуска autoexec.bat, в результате чего после загрузки DOS пользователь сразу оказывается в среде Norton Commander. Экран при этом оказывается разделенным на два окна, с каждым из которых пользователь может работать независимо. Переход курсора из одного окна в другое осуществляется с помощью клавиши табуляции. Эти окна обычно называют панелями и в них содержатся оглавления каталогов, имена которых (и пути доступа к ним), изображаются сверху каждой панели.

Одна строка в каждом окне содержит необходимую информацию о каком-либо файле или каталоге. При этом первая колонка символов соответствует имени и типу файла, вторая — размеру файла. У подкаталога во второй колонке находится указатель SUB-DIR. Имена файлов в оглавлении каталога выводятся строчными буквами, а подкаталоги для их отличия — заглавными. Обычно сначала выводятся сведения о подкаталогах, а затем — о файлах. В оглавлении подкаталога самую верхнюю строку занимает ссылка на родительский каталог в виде многоточия “..”, справа от которых изображается UP-DIR.

Выделенный файл или каталог. Будем называть выделенным такой файл или каталог, который на экране выделен серым цветом (на монохромном дисплее — инверсным изображением). С помощью клавиш перемещения курсора можно передвигать выделенный участок по панели в пределах текущего каталога, имя которого выведено сверху панели. Нажав клавишу табуляции, можно перевести выделенный участок в другую панель Norton Commander, т.е. сделать текущим находящийся там каталог.

Если нажать клавишу <Alt> (<ДОП>) и, не отпуская ее, начать набирать первые буквы имени нужного файла, то Norton Commander выделит нужный файл в текущем каталоге, как только будет введено достаточное для его отличия количество букв.

Если выполнить набор имени файла после нажатия клавиш <Alt>+<F7> (<ДОП>+<Ф7>), то поиск файла будет проведен во всех каталогах текущего диска. В имени файла можно использовать шаблоны “*” и “?”. Если будет найдено несколько файлов, то клавишами вертикального перемещения курсора среди найденных необходимо выделить нужный файл. Затем следует в меню выделить “ChDir” и нажать <Enter> (<ВВОД>) для перехода в тот каталог, где находится нужный файл.

Если выделить файл с расширением .com, .exe или .bat и нажать <ВВОД>, то начнется выполнение этого файла.

Выделенная группа файлов. Постоянное выделение файла (т.е. помещение его в группу выделенных) осуществляется нажатием клавиши <Ins> (<ВСТ>). Для снятия выделения файла следует также нажать <Ins>.

Для выделения группы файлов можно также нажать <+> (плюс на функциональной клавиатуре) и задать шаблон для выбора имен файлов. В нем можно использовать подстановочные символы “*” и “?”, смысл которых тот же, что и в командах DOS. После нажатия клавиши <Enter> выделится группа файлов, имена которых соответствуют указанному шаблону (при вводе шаблона, установленного по умолчанию *.*, выделится весь подкаталог). Для снятия выделения у этой группы файлов нужно нажать <-> (минус на функциональной клавиатуре) и ввести тот же шаблон.

Работа с дисками и каталогами. Для перехода на другой диск следует нажать <Alt>+<F1> (<ДОП>+<Ф1>) для левой панели или <Alt>+<F2> (<ДОП>+<Ф2>) для правой панели. Затем клавишами горизонтального перемещения курсора следует выбрать из появившегося списка доступных дисков нужный и нажать <Enter>.

Для перехода в содержащийся в оглавлении подкаталог нужно совместить курсор с его именем и нажать клавишу <Enter> (<ВВОД>). Norton Commander “войдет” в этот каталог и выведет его оглавление. Для перехода в родительский каталог надо выделить <..> и нажать <ВВОД> или при любом положении курсора нажать одновременно <Ctrl>+<PageUp> (<УПР>+<СтрВв>).

Для перехода в другой каталог на том же диске следует нажать комбинацию клавиш <Alt>+<F10> (<ДОП>+<Ф10>), выделить клавишами перемещения курсора среди появившихся на экране нужный каталог и нажать <Enter> (<ВВОД>).

Командная строка. Ниже панелей располагается командная строка с приглашением DOS, в которой можно вводить обычные команды или запускать программы. После их выполнения происходит возвращение в сре-

ду Norton Commander к ее панелям. Снятие или возвращение панелей, осуществляемое для просмотра выводимых на экран результатов выполнения команд, достигается совместным нажатием клавиш <Ctrl>+<O> (<УПР>+<О>).

Для вывода в командную строку имени выделенного файла на панели следует нажать <Ctrl>+<Enter> (<УПР>+<ВВОД>), а для очистки командной строки — <Esc> (<КЛЮЧ>).

Введенные команды запоминаются в памяти в виде последовательного списка. Нажатием клавиш <Ctrl>+<E> (<УПР>+<E>) или <Ctrl>+<X> (<УПР>+<X>) в командную строку каждый раз выводится команда, которая была введена соответственно перед или после ранее выполненной команды в списке. Выбранная нужная команда после необходимых изменений может быть отправлена на выполнение.

Для выполнения одной из ранее введенных команд без всяких изменений следует нажать <Alt>+<F8> (<ДОП>+<F8>), выделить с помощью клавиш перемещения курсора в появившемся на экране списке нужную команду и нажать <Enter> (<ВВОД>).

Назначение функциональных клавиш

За командной в самом низу экрана располагается строка, напоминающая о назначении в Norton Commander функциональных клавиш <F1> – <F10> (<Ф1> – <Ф10>).

<F1> — Help (Помощь). Выдается развитая система подсказок и пояснений о назначении функциональных клавиш при работе с NC.

<F2> — User (User Menu — Меню пользователя). Выводит на экран дисплея функциональное меню, задаваемое пользователем в файле NC.MNU. В него включаются наиболее часто встречающиеся в повседневной работе команды и функции. Например, если пользователь включил в это меню программу PCTOOLS, то для ее вызова уже не потребуется выходить в DOS: достаточно вызвать User Menu с помощью клавиши <F2> и после этого войти в PCTOOLS, нажав соответствующую клавишу.

<F3> — View (Просмотр). Служит для просмотра выделенного с помощью курсора файла. После нажатия клавиши <F3> содержимое файла можно читать с помощью клавиш со стрелками вверх и вниз, <PageUp> (<СтрВверх>) и <PageDn> (<СтрВниз>), <HOME> (<НАЧ>) и <END> (<КОН>). Для выхода из режима просмотра следует нажать <F10>.

<F4> — Edit (Редактирование). Предназначена для редактирования выделенного курсором файла. Для создания нового файла необходимо нажать клавиши <Shift>+<F4>, в появившемся на экране приглашении набрать имя файла и нажать клавишу <Enter>. Если файл с таким именем не найден, то в появившемся на экране сообщении об этом при выделении курсором варианта New-file (Новый файл) следует еще раз нажать <Enter>.

Редактирование файла производится в режиме вставки символов, поэтому клавиша <Ins> (<ВСТ>) не работает. Клавиша <Enter> используется для разбиения строки на две, для чего надо поместить курсор в том месте, где надо разделить строку, и нажать клавишу <Enter> (<ВВОД>). Если при этом курсор находился в начале или в конце строки, то впереди нее или за ней вставится пустая строка. Для соединения двух строк, надо поместить курсор правее последнего символа первой строки и нажать (<УДЛ>). Отметим, что при нахождении в позиции курсора или справа от него любого символа, действие клавиши по удалению символа происходит обычным образом. Также описанным в п. 1.1 обычным образом используются остальные стандартные управляющие клавиши. Клавиша <F2> используется для записи файла на диск с тем же именем. С помощью клавиш <Shift>+<F2> можно записать отредактированный файл под другим именем, которое запрашивается. Клавиша <Esc> (<ВЫХ>), как и <F10>, предназначена для выхода из режима редактирования без записи файла на диск.

Встроенный редактор NC версии 4.0 позволяет выделять блоки из строк текста. Для этого надо установить курсор последовательно в первую и последнюю строку блока, каждый раз нажимая при этом <F3>. Затем выделенный блок можно скопировать или переместить в позицию перед курсором, нажав соответственно <F5> или <F6>, а при нажатии <F8> удалить. Одновременное нажатие <Shift>+<F3> отменяет выделение блока текста.

<F5> — Copy (Копирование). Применяется для копирования выделенных файла или группы файлов. В середине экрана NC выдает в окне сообщение, что и куда он намерен копировать. По умолчанию файл или группа файлов копируется в каталог, показанный на другой панели, без изменения имен. Если напечатать в окне только новое имя файла, то копирование будет выполнено в текущем каталоге под указанным именем. При необходимости можно набрать и другое имя каталога, в который надо производить копирование. Затем для копирования надо нажать клавишу <Enter>. Если файл с таким именем в каталоге назначения уже существует, то выдается предупреждение с предложением выбрать один из вариантов: *Overwrite* — скопировать файл на место существующего, *Skip* — не копировать данный файл, *All* — копировать все файлы (для группы выделенных) на место существующих файлов без дополнительного предупреждения. Для отмены команды надо нажать клавишу <Esc>.

Если после нажатия клавиши <F5> в появившемся окне в качестве назначения копирования указать логическое устройство *PRN*, то после подключения принтера к ПЭВМ и нажатия клавиши <Enter>, будет распечатан выделенный файл.

<F6> — *Renmov* (*Rename/Move* — Переименовать/Переместить). Предназначена для переименования или переноса выделенного курсором файла

из одного каталога в другой. Для переименования файла или подкаталога после нажатия клавиши <F6> в выведенной строке, куда автоматически перемещается курсор, необходимо набрать его новое имя и нажать <ENTER>. Команда переноса файла выполняется так же, как и команда копирования. Отличие состоит в том, что команда переноса стирает исходный файл.

<F7> — Mkdir (Make Direktory — создать каталог). Служит для создания нового подкаталога. Для этого в выводимой на экран строке следует указать имя нового подкаталога. После нажатия клавиши <Enter> в оглавлении текущего каталога появится новый пустой подкаталог с указанным именем.

<F8> — Delete (Уничтожить). Предназначена для стирания файла, выделенного курсором, помеченной группы файлов или пустого подкаталога.

<F9> — Menu. В верхней строке экрана выводится меню, содержащее режимы работы Norton Commander.

<F10> — Quit (Покинуть). Выход из Norton Commander.

Приобретение нужных навыков работы с программой Norton Commander быстро происходит при ее практическом использовании, когда в полной мере и оцениваются ее достоинства.

3.2. Программа PC Tools

Рассмотрим некоторые операции, удобные для практической работы, которые выполняет программа PC Tools. Она состоит из двух независимых частей: меню файлов и меню дисков, интерфейса с пользователем. Первое из них реализует операции, касающиеся одного или группы файлов (копирование, перемещение, удаление и т. п.) и которые обычно удобнее выполнять с использованием программы Norton Commander. Второе реализует операции, касающиеся всего дискового пространства (копирование дисков, инициализация (форматирование) дискет, восстановление файлов, парковка НЖМД). В меню файлов существуют удобные для практической работы и дополнительные к Norton Commander сервисные функции, которые мы и рассмотрим.

Загрузка PC Tools осуществляется набором в командной строке имени программы pctools (причем текущим должен быть подкаталог с файлом pctools.exe или в команде PATH должен быть указан путь доступа к нему). Если пользователь включил в User Menu программу PC Tools, то для ее вызова будет достаточно вызвать User Menu с помощью клавиши <F2> и после этого войти в PC Tools, нажав соответствующую клавишу.

После загрузки программы, на экране появляется приглашение к работе, из которого вам предлагается перейти в меню файлов (при нажатии любой кла-

виши) или в меню дисков (при нажатии клавиши <F3>). Для выхода из программы следует нажать <Esc>.

После нажатия клавиши <F3> в нижней части экрана в рамке появляется меню режима работы с магнитными дисками, в котором отображается перечень имен выполняемых команд. Вызов той или иной команды осуществляется набором заглавной буквы из приведенных английских названий команд, например (в скобках указано полное имя команды):

N (iNitalize) — инициализация (форматирование) гибкого диска. После нажатия клавиши <N> в ответ на появляющийся запрос следует ввести имя формируемого диска (для А: просто нажать <ВВОД>). Для ответа на следующий запрос с помощью клавиш вертикального перемещения курсора нужно выбрать, на какой объем форматировать дискету (на 360, 320, 180, 160 Кбайт или 1.2 Мбайт), после чего нажать <ВВОД>. Затем на экране появляется предупреждение “Дискета X будет отформатирована”, где X — имя выбранного диска. Далее при нажатии любой клавиши происходит форматизация (F) и проверка (V) дорожек дискеты, сопровождающаяся визуальным показом на экране. Затем на экран выходит запрос на ввод метки диска (label). В ответ можно просто нажать <ВВОД>. Далее на экран выходит запрос — надо ли резервировать место на диске под системные файлы. Если вы хотите отформатировать дискету как рабочую, то ответ “N”. После форматизации указывается общий объем диска, плохих секторов и свободного места на диске в байтах. Нажмите любую клавишу для форматирования следующей дискеты. Для отказа нажмите <Esc> (<КЛЮЧ>) и вы вернетесь в основное меню (с любого этапа форматирования).

C (Copy) — копирование содержимого одной дискеты на другую. После нажатия клавиши <C> в ответ на появляющиеся запросы следует последовательно ввести имя диска-оригинала, имя диска-копии и вставить соответствующие дискеты в указанные дисководы. При нажатии любой клавиши осуществляется процесс копирования, после чего происходит выход в основное меню.

Если ПЭВМ имеет один дисковод для гибких магнитных дисков, то на запрос о диске-оригинале, затем диске-копии указывается одно имя А, куда попеременно вставляются соответствующие дискеты. При вставлении в дисковод А диска-оригинала в оперативную память ПЭВМ считывается максимально допустимый объем информации, который потом записывается на диск-копию при его помещении на тот же дисковод А. Процесс сопровождается визуальным показом на экране и попеременная замена дискет повторяется столько раз, пока вся информация не будет скопирована. В обоих случаях при копировании на неотформатированную дискету

данная команда автоматически отформатирует ее по образцу исходной дискеты.

U (Undelete system) — восстановление удаленных файлов. В DOS после уничтожения файла его имя удаляется из каталога, однако сам файл остается записанным на диске на том же месте. Поэтому восстановление файлов осуществимо в том случае, если на место расположения файла не была записана новая информация.

После нажатия клавиши <U> в ответ на запросы следует ввести имя рабочего диска, клавишами вертикального перемещения курсора выбрать команду восстановления файлов (File) и нажать <Enter> (<Ввод>). Затем клавишами перемещения курсора на появившемся на экране дереве каталогов указанного вами диска выбирается подкаталог, файлы которого вы хотите восстановить, и нажимается <Enter>. После этого на экране высвечивается список всех ранее удаленных файлов этого подкаталога (в именах которых вместо 1-го символа стоит знак ?). Затем с помощью нажатия клавиши <Enter> (и клавиш перемещения курсора) выделяется список файлов, подлежащих восстановлению, и нажимается клавиша <G>.

Далее следует исправить 1-й символ в появившемся на экране имени 1-го восстанавливаемого файла и нажать <Enter>. Затем после появления на экране подсказки (о клавишах <F1> и <F2>) нажать <F1> и после сообщения об успешном восстановлении нажать любую клавишу. Далее следует ввести первый символ в имени второго восстанавливаемого файла, нажать <F1>, потом любую клавишу и т.д. до конца выделенного списка, после чего следует автоматическое возвращение в основное меню (для перехода туда на любом этапе следует нажать <Esc>).

P (Park) — фиксация магнитных головок жесткого диска в нерабочем положении перед выключением ПЭВМ, о чем появляется сообщение после нажатия <P>.

I (Info) и **H (Help)** — обеспечивают отображение полезной информации о компьютере и помощь в использовании программы PC Tools.

3.3. Microsoft Windows

Система Windows — это графическое окружение, которое предлагает новые, более удобные способы работы на ПЭВМ (на базе мощных процессоров Intel 80386, i486 или Pentium). Windows поставляется с большим набором полезных программ, которые помогают выполнять различную работу. Каждая программа в Windows имеет хотя бы одну прямоугольную область, называемую окном, предназначенную для связи пользователя с данной

программой. Экран монитора представляется в Windows как рабочий стол, на котором располагаются окна работающих в данный момент программ. Их в любой момент можно сжать до небольшого графического изображения — пиктограммы (icon) или восстановить в реальных размерах, перемещать в различные места экрана, а также удалять на время с него те элементы, которые в настоящий момент не нужны.

Microsoft Windows обычно устанавливается в каталог с именем windows или win и для его запуска необходимо отправить на выполнение программу win.com (достаточно набрать win и нажать Enter, если запуск не выполняется автоматически).

Windows Help (Справка) позволяет быстро получить справочную информацию о работе с Windows нажатием кнопки мыши или клавиши <F1>. Следующий раздел поможет вам лучше понять смысл и описание приводимых там процедур.

3.3.1. Основные элементы Windows

Пиктограмма (Icon) — графическое представление различных элементов в Windows (программы или окна документа, которые были минимизированы).

Указать (Point) — переместить мышь (mouse), использование которой обеспечивает максимальные удобства при работе с Windows, чтобы ее указатель показывал на нужный элемент. С ее помощью выполняются три следующие стандартные операции.

Операция Click — установив курсор на нужный элемент изображения, быстро нажать и отпустить (обычно левую) кнопку мыши (сокращенно обозначают “нажмите мышкой”). Используется для выделения различных элементов (имя файла и т.п.) или выбора команд (позиций меню), выполняющих какие-то действия.

Операция Double-click — установив курсор на нужный элемент изображения, быстро дважды нажать и отпустить кнопку мыши (сокращенно обозначают “дважды нажмите мышкой”). Используется обычно для запуска программ на выполнение. Если нажатия происходят недостаточно быстро, то вместо одной операции Double-click получается две операции Click.

Операция Drag (Переместить) — установив курсор на нужный объект или элемент изображения (граница окна экрана и т.п.), нажать и удерживать кнопку мыши при ее перемещении. При этом объект станет передвигаться по экрану синхронно с перемещением курсора и после отпускания кнопки мыши зафиксируется на новом месте.

Окно (Window) — прямоугольная область экрана, содержащая программу или файл документа. Ее можно открывать, закрывать, перемещать, изменять ее размеры, сжимать в пиктограмму или увеличивать до размеров всего экрана. Каждое окно имеет ряд общих элементов.

Системное меню (Control-menu box) расположено в верхнем левом углу каждого окна в минимизированном виде и представляет из себя небольшой квадратик с прямоугольником. Для его вызова достаточно нажать кнопкой мыши по этому квадратику (или воспользоваться клавишами <Alt>+<Пробел>). Команды этого меню позволяют изменять размеры окна, перемещать, увеличивать, минимизировать и закрывать окна, а также переключаться на список активных в настоящий момент программ.

Заголовок (Title bar), расположенный посередине вверху каждого окна, содержит название программы или документа. Если документ еще не был сохранен, то на его месте появляется надпись *untitled* (безымянный).

Строка меню (Menu bar) расположена ниже заголовка и в ней перечислены имеющиеся в наличии меню. Большинство программ имеют меню File (Файл), Edit (Редактировать), Help (Справка), а также специфические для данной программы меню. Для их открытия следует подвести курсор на имя меню в строке меню и нажать кнопку мыши. После открытия меню делается выбор из него нужной команды, производящей выполнение соответствующего действия, для чего также следует нажать кнопкой мыши имя нужного элемента. Для закрытия меню следует нажать кнопкой мыши его имя или любое место вне меню.

Записи элементов меню подчиняются следующим правилам. Если имя команды выглядит нечетко, то в данный момент вы не можете ею воспользоваться. Многоточие после выбора элемента меню означает, что после его выбора появится диалоговое окно, запрашивающее дополнительную информацию для выполнения команды. Галочка рядом с именем элемента меню определяет, что команда является активной (используется для команд, переключающихся между двумя состояниями). Комбинация клавиш после имени элемента меню является сокращением команды и позволяет ею воспользоваться без предварительного открытия меню. Треугольник, расположенный справа от команды меню означает, что она приводит к открытию каскадного меню, в котором содержатся дополнительные команды.

Две кнопки в правом верхнем углу окна, будучи нажатыми с помощью клавиши мыши, позволяют увеличить до размеров всего экрана (правая кнопка с треугольником острием вверх) или минимизировать до пиктограммы (левая с треугольником острием вниз) активное окно. После увеличения окна на

правой кнопке появляется два треугольника, расположенных один над другим. Нажав на эту кнопку, можно вернуть окну его первоначальные размеры.

В правой (и нижней) части окна расположен часто используемый в Windows орган управления, называемый полосой просмотра или слайдером. В верхней и нижней части полосы (или правой и левой) расположены кнопки в виде стрелок. После нажатия на них мышкой в соответствующую сторону происходит движение расположенного между ними движка слайдера (в виде выделенного прямоугольника), а с ним и перемещение информации, не помещающейся целиком в выделенном месте. Его положение между стрелками соответствует положению нужного места в документе (начало, середина или конец файла или списка), а перемещение в любую позицию осуществляется мышью (операция Drag). Нажатие кнопки мышки в любом месте слайдера выше или ниже движка (или слева или справа для горизонтальной полосы) приведет к перемещению (скроллингу) информации на одно окно.

3.3.2. Работа с диалоговыми окнами

Windows использует диалоговые окна (dialog boxes) для запроса нужной или выдачи дополнительной информации. Они появляются как при выборе команды, после имени которой в меню стоит многоточие (...), так и при использовании опций, запрашивающих дополнительную информацию. Для выбора кнопки команды, опции, элемента из поля списка или переключателя нужно нажать на кнопку с соответствующим именем мышкой. Если в нем содержится подчеркнутая буква, то эту команду или опцию можно выбрать также из любого места диалогового окна, нажав <Alt> (<Доп>) и эту подчеркнутую букву. Выделенная кнопка имеет более темный контур по сравнению с остальными или черную точку (в списках взаимно исключающих вариантов выбора), помеченный переключатель содержит X и их может быть несколько, а недоступные в данный момент выглядят нечетко. Чтобы изменить выбор, пометьте другую кнопку или для выделенного элемента списка вновь нажмите его мышкой.

Если требуется ввести дополнительную информацию, то для этого в окне существует прямоугольная область, называемая полем текста (text box). Если оно пусто, то при перемещении указателя мыши в нем появляется курсор ввода (мерцающая вертикальная черта), после чего можно печатать. Если поле уже содержит текст, то при перемещении в него он автоматически выделяется и любой набираемый текст заменит его. Перед этим следует нажать кнопку мышки в том месте, где нужно поместить начало текста, для появления в нем курсора ввода.

После ввода всех необходимых данных (имени и месте расположения файла и т.п.) необходимо отправить команду на выполнение. Для этого следует нажать мышкой кнопку с именем ОК (операция Click) или дважды нажать кнопку с именем нужной команды (операция Double-click). После этого диалоговое окно закрывается и команда выполняется. Закрытие диалогового окна без выполнения команды достигается нажатием клавиши <Esc>, или мышкой кнопки Cancel, либо двойным нажатием кнопки Системного Меню в верхнем левом углу экрана (последнее вызовет также закрытие окна программы или документа).

3.3.3. Работа с программами и файлами

Программы, написанные только для использования с системой Windows, использующие графический интерфейс, меню со стандартными элементами и диалоговые окна, подчиняются одним и тем же правилам и называются Windows-программами. Они не могут работать в MS-DOS.

Одновременно с этим накоплено огромное программное обеспечение еще до выпуска системы Windows, разработчики которой были вынуждены предусмотреть его использование. Они рассчитаны на работу с MS-DOS, называются DOS-программами и могут работать в графическом окружении Windows, хотя и не подчиняются его правилам (не используют меню и диалоговые окна, мышку и т.п.). К последним относятся и описываемые в настоящем пособии программы, которые являются универсальными и могут работать на любых IBM-совместимых ЭВМ (не только на персональных с MS-DOS, но и на “больших” с различными операционными системами, в частности СВМ).

В стандартном или реальном режиме все DOS-программы работают во всем экране. Это означает, что программа занимает весь экран и выглядит точно так, как при выполнении в MS-DOS, без Windows.

Для открытия файла следует выбрать команду Open (Открыть) из меню File программы, затем переместиться в поле списка Directories (Каталоги) и дважды нажать кнопкой мышки имя каталога, содержащего нужный файл. Потом в появившемся после этого поле списка Files (Файлы) следует указать нужный файл и дважды нажать мышкой его имя и кнопку ОК. Для создания нового файла в меню File следует выбрать команду New.

Положение курсора (мерцающая вертикальная черта) помечает место, где будут появляться набираемые символы. При работе с новым файлом можно сразу начать печатать (при необходимости установив курсор клавишами <Enter> или <Пробел> в нужное место). Перед внесением исправлений в существующем файле следует нажать кнопку мышки в нужном месте, после чего там появится указатель курсора.

Для сохранения изменений в уже существующем файле используется команда Save (Сохранить) из меню File (Файл). Чтобы сохранить новый или уже существующий файл под новым именем из меню File следует выбрать команду Save As, указать нужное имя и путь в поле текста Filename (Имя файла), после чего нажать кнопку ОК. Вместо указания полного пути предварительно в поле списка Directories (Каталоги) можно выбрать каталог, в котором нужно сохранить файл.

Запустить программу на выполнение в Windows можно тремя способами.

1. Открыв окно Program Manager, следует открыть окно группы, которая содержит запускаемую программу (см. п. 3.3.5). После этого нужно дважды нажать кнопкой мышки пиктограмму программы.

2. Запустив File Manager (см. п. 3.3.6), следует открыть окно каталога, содержащего файл программы (с расширениями .com, .exe, .bat или .pif). Теперь также нужно дважды нажать кнопкой мышки пиктограмму программы.

3. Выбрать команду Run (Запустить) из меню File в Program Manager или File Manager. В появившемся диалоговом окне Run следует набрать полное имя запускаемого файла (с указанием пути к нему, если он находится вне текущего каталога). После этого следует нажать клавишу <Enter> или мышкой кнопку ОК. Способ удобен для запуска редко используемых программ, которые не добавлены в группу в Program Manager.

На экране часто открыто много окон программ. Активное окно, в котором вы работаете, всегда появляется на переднем плане и его заголовок имеет другой цвет или интенсивность. Оно может перекрывать неактивные окна, частично или полностью загораживая их (поэтому некоторые из них могут быть невидимыми). Чтобы сделать активным видимое окно, следует нажать кнопкой мышки в любом его месте. Для неактивного окна сначала нужно открыть диалоговое окно Task List (Список Задач). Этого можно достичь двойным нажатием кнопкой мышки в любом свободном от икон или пиктограмм месте экрана, а также выбором Switch To (Переключиться На) из Системного Меню программы. Теперь двойное нажатие кнопкой мышки имени нужной программы в поле списка в Task List позволит на нее переключиться.

Отметим, что нажатие кнопок Cascade (Каскад) и Tile (Мозаика) в Task List приводит к соответствующему упорядочиванию окон программ или файлов, а Arrange Icons — к переупорядочиванию пиктограмм (вдоль нижнего края экрана).

Для выхода из программ нужно выбрать команду Exit из меню File программы (либо набрать exit для DOS-программ). Можно также дважды нажать мышкой кнопку Системного Меню в левом верхнем углу экрана или выбрать команду Close из Системного Меню, а также воспользоваться клавишами <Alt>+<F4>.

3.3.4. Использование меню Help для получения справочной информации

В системе Help (Справка) приведены описания основных приемов, необходимых для работы с Windows, клавиатурный эквивалент действий без мышки, а также содержится информация об используемых командах. В зависимости от этого имеются различные варианты вызова Help.

Если выбрать мышкой кнопку Help в строке меню, а затем нужный раздел, то получим или информацию по данному разделу, или список его тем. При вызове Help клавишей <F1> на экране появляется Index (Оглавление) справочной информации программы, с которой вы работаете.

Для поиска информации с помощью Help Index (Оглавления Help) следует нажать клавишу <F1> или выбрать нужный раздел из меню Help и переместиться к нужной теме. Когда указатель мышки станет похожим на указательный палец, нужно нажать кнопку мышки. Это означает, что информация по данной теме имеется в наличии, и при выборе этой темы появляется содержащее ее окно. Для более полного ответа в конце текста справки часто приводится список связанных с ней тем, выбрать которые можно также нажатием кнопки мышки.

Можно получить нужную информацию, соответствующую ключевому слову. Для этого следует открыть Help и нажать мышкой кнопку Search (Поиск). В появившемся диалоговом окне в текстовом поле Search For (Что искать?) нужно набрать ключевое слово или его часть. Ниже в поле списка приводятся ключевые слова, соответствующие набираемым буквам. При нажатии кнопки Search выдается список тем, имеющих отношение к набранному ключевому слову или фразе, из которых следует выбрать нужную. При следующем обращении к Search сохраняется название темы, найденной в прошлый раз.

Большинство меню Help содержит следующие основные опции:

Index (Оглавление) — список всех тем справочной информации в алфавитном порядке;

Keyboard (Клавиатура) — таблицы комбинаций клавиш для работы с клавиатурой в активной программе;

Commands (Команды) — объяснения всех используемых команд;

Procedure (Процедуры) — пошаговые инструкции по использованию активной программы.

Using Help (Работа со Справкой) содержит небольшой урок и информацию о том, как работать с Windows Help. С ним рекомендуется ознакомиться при первоначальном использовании справочной системы.

3.3.5. Program Manager

Центральной программой в Windows является Program Manager (Управление Программами). Она стартует автоматически при запуске Windows и прекращает свою работу только вместе с ним. Предназначена для запуска программ и выполняет организационные функции (объединяет программы в группы по каким-либо признакам). Окна групп содержат пиктограммы заголовков программ, запускающих связанную с ней программу. Команды строки меню Program Manager воздействуют на все окна групп, имеющих только свое собственное Системное Меню.

Пиктограммы групп — это минимизированные окна групп. Они различаются только подписями под каждой пиктограммой и расположены в нижней части окна Program Manager, за пределы которого их, как и окна групп, невозможно переместить. Пиктограммы заголовков программ находятся внутри окон групп. Их можно перемещать из окна одной группы в окно другой, но не непосредственно в окно Program Manager или за его пределы.

При первом запуске Windows Program Manager появляется на экране с открытым окном группы Main (Главная), которое содержит прикладные программы системы: File Manager (Управление Файлами), Control Panel (Панель Управления), Print Manager (Управление Печатью), Clipboard (Буфер), DOS Prompt (Интерпретатор команд DOS), Windows Setup (Установка Windows).

Группа Accessories (Вспомогательные программы) включает текстовый процессор, программу рисования, коммуникационную программу и несколько простых сервисных программ (калькулятор, часы, календарь, блокнот, картотеку), а также Macro Recorder (генератор макрокоманд Windows) и PIF Editor (редактор файлов программной информации для DOS-программ).

Группы Windows Applications (Windows-программы) и Non-Windows Applications (DOS-программы) содержит соответствующие программы, которые были найдены Setup на винчестере во время процедуры установки Windows.

Чтобы запустить программу из группы нужно дважды нажать кнопкой мышки соответствующую пиктограмму группы, из-за чего ее окно откроется. Затем следует также дважды нажать мышкой пиктограмму нужной программы. При запуске программы Program Manager обычно делает текущим каталог, ее содержащий. Чтобы вернуться обратно из какой-либо программы, следует дважды нажать мышкой пиктограмму Program Manager.

Отметим, что в Program Manager можно добавлять или удалять необходимое количество групп или программ, чтобы организовать работу наиболее удобным образом.

Чтобы создать новую группу следует выбрать команду New (Новый) из меню File (Файл). В появившемся диалоговом окне New Program Object (Новый Программный Объект) нужно выбрать опцию Program Group (Программная группа) и нажать ОК. После этого появится следующее диалоговое окно Program Group Properties (Реквизиты Программной Группы), в поле Description (Описание) которого следует ввести имя группы и нажать ОК.

Чтобы добавить программу в группу нужно открыть ее окно (дважды нажав мышкой пиктограмму группы), выбрать также команду New (Новый) из меню File (Файл), но в том же диалоговом окне New Program Object (Новый Программный Объект) следует выбрать опцию Program Item (Программа) и нажать ОК. Затем в следующем диалоговом окне Program Item Properties (Реквизиты Программы) в поле текста Description (Описание) следует ввести имя программы, а в поле Command Line набрать полное имя командного файла (и путь к нему, если соответствующий каталог не указан в команде Path в файле autoexec.bat), после чего нажать ОК. Если имя программного файла неизвестно, то в появившемся на экране после нажатия кнопки Browse (Просмотр) списке файлов и каталогов следует выделить нужное имя файла (и каталог, если он не является текущим), щелкнув по нему мышкой, и нажать ОК, чтобы ввести его в поле Command Line.

Для удаления группы нужно нажать кнопкой мышки ее пиктограмму, выбрать команду Delete (Удалить) из меню File (Файл) и в диалоговом окне выбрать Yes (Да). В результате выделенная группа и все пиктограммы заголовков программ в ней будут удалены, хотя файлы соответствующих программ все же останутся на диске. Для удаления программы из группы следует выполнить аналогичные действия с пиктограммой ее заголовка (предварительно открыв окно нужной группы), что также не приведет к ее удалению с диска.

Для перемещения программ или их копирования из одной группы в другую следует выполнить операцию Drag над пиктограммой программы, причем копирование будет проводиться при нажатии и удерживании клавиши <Ctrl> во все время перемещения.

Чтобы изменить реквизиты заголовка программы, нужно выделить пиктограмму программы, выбрать Properties (Реквизиты) из меню File, внести нужные изменения и нажать кнопку ОК.

Чтобы выйти из Program Manager (и из Windows) сначала следует выйти из всех выполняемых программ, выбрать Exit Windows (Выход из Windows) из меню File и после подтверждения завершения сеанса работы в появившемся диалоговом окне нажать кнопку ОК. Если при этом переключатель Save Changes (Сохранить Изменения) будет помечен, то при следующем запуске Windows рабочая область Program Manager будет иметь тот вид, который она имела во время последнего сеанса.

3.3.6. File Manager

Программа File Manager (Управление Файлами) является инструментом, помогающим содержать файлы и каталоги в порядке. Из нее можно также запускать программы. При первом запуске Windows пиктограмма File Manager появляется на первом месте в группе Main (Главная) в Program Manager.

В File Manager используются окна двух типов. В окне Дерева Каталогов (Directory Tree) показана общая структура каталогов диска, в окне каталога — перечислены файлы и подкаталоги (таких окон можно сделать несколько при помощи меню Window и команды New Windows (Окно и Новое Окно)). Чтобы оба окна расположились без перекрытий, надо нажать <Shift>+<F4>. Тогда File Manager станет похожим на Norton Commander (средства обработки файлов которого более удобны и мощны [26, с. 213]).

Чтобы выдать на экран структуру каталогов другого дискового накопителя (или содержание другого каталога) нужно выбрать мышкой пиктограмму с нужной буквой дисководов (или именем каталога). По умолчанию наличие подкаталогов ничем не отмечается. Если в меню Tree (Дерево) включен переключатель Indicate Expandable Branches (Отмечать Расширяемые Ветви), то на пиктограммах каталогов появится индикация “+” и “-”. Знак “+” означает, что в данном каталоге имеются подкаталоги. Если нажать по нему мышкой или воспользоваться клавишей <+>, то File Manager покажет подкаталоги 1-го уровня этого каталога. Он станет “раскрытым” и на его пиктограмме вместо “+” появится знак “-”. Аналогичные действия над знаком “-” (мышкой или клавишей <->) уберут с экрана подкаталоги, каталог снова станет “закрытым” со знаком “+”. Чтобы увидеть всю ветвь полностью, следует выделить раскрываемый каталог и нажать клавишу <*> (звездочка). Совместное использование <Ctrl>+<*> позволит увидеть одновременно все уровни всех каталогов.

Основные команды для работы с файлами и каталогами (открыть, переместить, копировать, удалить, переименовать, поиск) одинаковы и содержатся в меню File (Файл). Работать можно с каждым файлом или каталогом в отдельности, либо группируя их, для чего их нужно выделить. Для одного файла или каталога это можно сделать, нажав кнопкой мышки его имя. Если их несколько и они расположены последовательно, выделив обычным образом первый элемент группы, для последнего следует нажать и удерживать клавишу <Shift> в момент его выбора мышкой. При их произвольном расположении отмечать каждый новый элемент мышью следует, удерживая при этом <Ctrl>. Для выделения всех файлов удобнее пользоваться клавишами <Ctrl>+</>, а для его снятия — <Ctrl>+<\>.

Отметим, что с помощью операции Drag (Переместить) удобно выполнять перемещение или копирование файлов, каталогов или их выделенных групп. Просто “берите” мышью файл, каталог или группу выделенных файлов и каталогов и “тащите” к целевой пиктограмме каталога или диска. В пределах диска по умолчанию происходит перемещение (а при удерживании <Ctrl> — копирование), с диска на диск по умолчанию выполняется копирование (а при удерживании <Alt> — перемещение).

Если разместить на дисплее Program Manager (Управление Программами) и File Manager (Управление Файлами) так, чтобы были видны окна обеих программ, то операция Drag позволит также устанавливать новые программные элементы в группах Program Manager. Для этого просто “возьмите” мышью пиктограмму исполняемого файла программы в File Manager и “перенесите” ее в ту группу Program Manager, где желательно поместить новый программный элемент. Таким же образом в Program Manager можно установить и файл данных, связанный с прикладной программой (свернутой до пиктограммы), для ее загрузки с этим файлом.

ЧАСТЬ 2. РЕШЕНИЕ ЗАДАЧ И ВЫПОЛНЕНИЕ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ ПО СТАТИКЕ С ИСПОЛЬЗОВАНИЕМ ГОТОВЫХ ПРОГРАММ

ГЛАВА 4. ОСНОВНЫЕ ЭТАПЫ РЕШЕНИЯ ЗАДАЧ СТАТИКИ НА ПЭВМ

Любую задачу статики по определению реакций опор после составления уравнений равновесия можно записать в матричной форме в виде системы линейных алгебраических уравнений:

$$(A) (X) = B, \quad (4.1)$$

где A или $A(I,J)$ — заданная квадратная матрица коэффициентов перед неизвестными, в которой выписаны только отличные от нуля значения элементов матрицы $A(I,J)$;

X или $X(I)$ — неизвестный вектор-столбец с N компонентами (усилия в стержнях, опорные реакции, неизвестные силы и т.п.);

B или $B(I)$ — заданный вектор-столбец с N компонентами — правые части уравнений равновесия, в которые должны быть перенесены все свободные члены уравнений, не содержащие неизвестных.

Число строк матрицы A , т.е. число уравнений равновесия N , равняется числу столбцов, т.е. количеству неизвестных в задаче. Поэтому в задачах статики матрица A всегда квадратная и ее размерность $A(N,N)$.

Важной особенностью систем линейных алгебраических уравнений (СЛАУ) задач статики является их сильно разреженный характер, т.е. наличие в их составе большого количества нулевых элементов.

Универсальные программы STAT и STATN предназначены для решения любых задач статики по определению реакций опор с любым количеством уравнений равновесия (до 30-ти) и в любом количестве вариантов решения. Их отличие заключается в том, что программа STAT предусматривает ввод всех элементов СЛАУ, тогда как STATN учитывает их разреженность и предусматривает ввод только ненулевых элементов с указанием их расположения.

4.1. Формализация уравнений равновесия и выбор используемой программы

Для численного решения любой СЛАУ, возникающей в задачах статики, ее сначала нужно формализовать, приведя к виду (4.1). Это достигается постановкой в соответствие неизвестным в задаче (усилиям в стержнях, опорным реакциям, неизвестным силам и т.п.) определенного элемента одномерного массива $X(I)$ (см. соответствие идентификаторов в пп. 5.1, 5.2, 6).

Вторым этапом следует численное определение всех коэффициентов при неизвестных в уравнениях равновесия и их свободных членов (не содержащих неизвестных). Последние переносятся в правую часть этих уравнений, образуя вектор-столбец свободных членов $B(I)$.

После выполнения 2-го этапа система уравнений любой задачи статики по определению реакций опор приобретает явный вид СЛАУ (4.1), в которой выписаны только отличные от нуля элементы. Для дальнейшего решения с помощью ПЭВМ пособие предоставляет две возможности: ввод всех элементов матрицы $A(I,J)$ и вектора-столбца $B(I)$ для работы с программой STAT (см. п. 4.2.1) или ввод только их ненулевых элементов с указанием их расположения, что требует программа STATN (см. п. 4.2.2).

Для удобства изложения можно разбить все рассматриваемые типовые примеры по формальному признаку количества уравнений равновесия на три группы: с малым (до 6-ти — см. п. 5.1), средним (до 12-ти — см. п. 6) и большим (свыше 12-ти — см. п. 5.2) числом уравнений. Отметим, что достоинства работы с программой STATN резко возрастают с увеличением числа уравнений равновесия.

4.2. Подготовка данных для решения задач статики на ПЭВМ

4.2.1. Работа с универсальной программой STAT

Программа STAT предназначена для решения любых задач статики по определению реакций опор или усилий в стержнях в любом количестве уравнений равновесия (до 30-ти) и любом количестве вариантов решения одновременно. Она предусматривает для своей работы ввод всех значений исходных данных в полном виде или с использованием повторителей для нулевых элементов.

Для работы с универсальной программой STAT квадратную матрицу A коэффициентов перед неизвестными следует дополнить нулями до ее полного

вида NSN . При этом все данные могут быть представлены по желанию как с указанием всех элементов в полном виде, так и в сокращенной форме записи с указанием количества (K^*) для повторяющихся (чаще всего нулевых) элементов. Например, фрагмент строки исходных данных $0,0,0,0,-0.7071,-0.7071$ проще представить в виде $4*0,2*-0.7071$ (чем вследствие его простоты и будем обычно пользоваться).

После этого исходные данные для работы программы STAT следует представить согласно следующей структуре:

1-я строка: номер группы, задания, варианта.

2-я строка: количество уравнений равновесия N в задании, число рассматриваемых вариантов решения M .

3-я — $(2+N)$ -я строки: поэлементный построчный ввод всех значений исходных данных матрицы A .

$(3+N)$ -я строка: перечисление через запятую всех данных вектора-столбца B в одну строку.

Если число рассматриваемых вариантов решения $M > 1$, то в следующих $M*(N+1)$ строках следует поэлементный построчный ввод всех значений исходных данных матрицы A_2 и соответствующего вектора-столбца B_2, A_3 и B_3, \dots, A_m и B_m .

Ввод данных матрицы A_1 и вектора-столбца B_1 осуществляется с помощью оператора ввода, управляемого списком, со встроенным неявным циклом, в котором указывается:

- для матрицы A значение всех ее элементов построчно:

$$\text{READ } *, (A(I, J), J=1, N), I=1, N) \quad (4.2)$$

- для вектора-столбца B значение всех его элементов в одну строку:

$$\text{READ } *, (B(I), I=1, N) \quad (4.3)$$

4.2.2. Работа с универсальной программой STATN

Программа STATN, как и программа STAT, также предназначена для решения любых задач статики по определению реакций опор или усилий в стержнях в любом количестве уравнений равновесия (до 30-ти) и любом количестве вариантов решения одновременно. Она предусматривает для своей работы ввод только значений ненулевых элементов исходных данных с указанием номера строки и столбца его расположения. Предварительно в самой программе производится обнуление всех рабочих массивов матриц A и B (задание нулевых элементов), после чего на нужные места матрицы вводятся значения ненулевых элементов.

Положение элемента $A(I,J)$ в матрице A характеризуется двойным индексом. Первый индекс I означает номер строки, в которой стоит элемент $A(I,J)$, т.е. это порядковый номер уравнения равновесия. Второй индекс J означает номер столбца, т.е. номер идентификатора X_j , при котором стоит элемент $A(I,J)$. Отметим, что нумерация строк производится сверху вниз, а столбцов — слева направо.

Положение элемента вектора — столбца $B(I)$ характеризуется только номером I -й строки, в которой стоит элемент $B(I)$.

Определив положение и значения всех ненулевых элементов матриц A и B , исходные данные для работы программы STATN следует представить согласно следующей структуре:

1-я строка: номер группы, задания, варианта

2-я строка: количество уравнений равновесия N в задании, число рассматриваемых вариантов решения M .

3-я строка: количество объектов равновесия KOR или частей, на которые разбивается ввод матрицы A . Это число обычно выбирают так, что оно определяет количество строк, в которых вводятся ненулевые элементы матрицы A .

4-я — $(3+KOR)$ -я строки: в каждой строке указывается первым число — количество ненулевых элементов KNNEKO каждого объекта равновесия матрицы A (т.е. их количество в данной строке ввода), далее для всех ненулевых элементов данного объекта равновесия указываются номер строки, столбца, значение каждого ненулевого элемента.

$(4+KOR)$ -я строка: указывается количество ненулевых элементов вектора-столбца B , далее номер строки и значение каждого ненулевого элемента вектора-столбца B .

Если число рассматриваемых вариантов решения $M > 1$, то в следующих $M \cdot (1+KOR+1)$ строках следует:

- количество объектов равновесия KOR или частей, на которые разбивается ввод каждой следующей матрицы A_i : A_2, A_3, \dots, A_N .
- далее в каждой следующей KOR-строке указывается первым количество ненулевых элементов KNNEKO каждого объекта равновесия соответствующей матрицы A_i (т.е. их количество в данной строке ввода), далее для всех ненулевых элементов данного объекта равновесия указываются номер строки, столбца, значение каждого ненулевого элемента;
- затем указывается количество ненулевых элементов соответствующего вектора-столбца B_i , далее номер строки и значение каждого ненулевого элемента вектора-столбца B_i .

Ввод матрицы A и вектор-столбца B осуществляется с помощью оператора ввода, управляемого списком, со встроенным неявным циклом, в котором указывается:

- для матрицы A номер строки, столбца, значение ненулевого элемента, перечисление должно быть повторено KNNEKO раз:

$$\text{READ } *, \text{ KNNEKO}, (I, J, A(I, J), J1=1, \text{KNNEKO}) \quad (4.4)$$

- для вектора-столбца B номер строки, значение ненулевого элемента, также повторенное KNNEKO раз:

$$\text{READ } *, \text{ KNNEKO}, (I, B(I), I1=1, \text{KNNEKO}) \quad (4.5)$$

где KNNEKO — количество ненулевых элементов во вводимой строке данных, I — номер строки вводимого ненулевого элемента; J — номер неизвестного, при котором стоит значение самого ненулевого элемента (т.е. это номер столбца); $A(I, J)$ — значение ненулевого элемента матрицы A , стоящее на пересечении I -й строки и J -го столбца; $B(I)$ — значение ненулевого элемента вектора-столбца B , стоящее в I -й строке.

Отметим, что тип данных, вводимых оператором ввода, управляемого списком (4.2)-(4.5), используемый программами STAT и STATN:

READ *, СПИСОК ВВОДИМЫХ ДАННЫХ

определяется по типу, количеству и порядку следования переменных в списке оператора программы, поэтому после вещественных значений элементов $A(I, J)$ и $B(I)$ без дробной части точку ставить необязательно. Мы будем использовать эту возможность во всех рассматриваемых ниже примерах, так как это несколько упрощает запись.

Напомним, что каждая новая строка вводимых данных начинается с крайней левой позиции.

4.3. Решение задачи на ПЭВМ и доведение первоначального решения до правильного. Анализ результатов, их проверка и выводы

4.3.1. Первоначальное решение задания на ПЭВМ

Условимся, что имя Вашего личного архивного файла на диске имеет вид

$$\{IO\}\{NG\}\{NZ\}.\{T\}\{NW\} \quad (4.6)$$

где $\{IO\}$ — инициалы имени и отчества преподавателя, например, wm ; $\{NG\}$ — последние три цифры номера группы, например для 114412 — 412; $\{NZ\}$ — номер задания, например с8 (k1, d3); $\{T\}$ — тип файла: с исходными данными — d , для результатов работы программы — l , для текста подпрограммы на Фортране f ; $\{NW\}$ — номер варианта, например 31. Тогда для указанных

конкретных значений имени файлов данных и результатов согласно (4.6) примут вид (4.7) и (4.8) соответственно:

```
wm412c8.d31 (4.7)
```

```
wm412c8./31 (4.8)
```

Все программы используют исходные данные, находящиеся в единственном рабочем файле с именем

```
имя_файла_программы.dat (4.9)
```

Рассмотрим последовательность действий для создания и редактирования файла исходных данных (4.7) с последующим расчетом по готовой программе, например STATN. При использовании других программ, требующих только подготовки исходных данных, она остается одинаковой. Во всех командах вместо имени программы statn следует указывать имена нужных программ (stat, stat9n, stat9, dinsp или ddinsp).

Сначала нужно сделать текущим соответствующий подкаталог, имя которого совпадает с именем выбранной программы (см. п. 3.1). Для этого следует подвести подсветку курсора к выбранному имени подкаталога (в рассматриваемом случае STATN) и нажать клавишу <Enter> (<Ввод>). Если подкаталог не находится на текущем диске, то предварительно следует нажать <Alt>+<F1> (<ДОП>+<Ф1>) для левой или <Alt>+<F2> (<ДОП>+<Ф2>) для правой панели, клавишами горизонтального перемещения курсора перейти на нужный диск и нажать <Enter>. Далее действия выполняются согласно приведенного сеанса работы.

1. Для создания или редактирования личного архивного файла в текущем каталоге необходимо при любом положении курсора нажать клавиши <Shift>+<F4> (<Верх>+<Ф4>). Затем в появившемся на экране приглашении набрать имя файла типа (4.7) и нажать клавишу <Enter>. Если файл с таким именем существует, то он вызывается на редактирование, если нет — то в появившемся на экране сообщении об этом выделенным окажется вариант “New-file” (“Новый файл”) и нужно еще раз нажать <Enter>.

2. Начать набор исходных данных или исправление ошибок. Для удаления символа используются клавиши: над курсором — (<УДЛ>), влево от курсора — <Back Space> (<ВШ>). Для разделения строки на две (добавление пустой) надо поместить курсор в месте разбиения (в начале или конце строки) и нажать клавишу <Enter> (<ВВОД>). Для соединения двух строк (удаления пустой), надо поместить курсор правее последнего символа первой строки (над пустой) и нажать (<УДЛ>).

3. После окончания ввода исходных данных, исправления ошибок и проверки набранной информации необходимо ее сохранить под архивным и рабочим именами. Для этого нужно сначала нажать клавишу <F2> (<Ф2>) (запись на диск

с именем редактируемого файла (4.7)), затем <Shift>+<F2> (<Верх>+<F2>), в появившейся строке набрать нужное имя (4.9) (statn.dat) и нажать клавишу <Enter> (<ВВОД>). Для выхода из режима редактирования нужно нажать клавишу <F10> (<F10>). На экране опять появляются панели Norton Commander.

4. Подвести подсветку курсора к имени рабочего файла (4.9) (statn.dat) и нажать клавишу <F4> (<F4>). Убедиться в том, что в нем находятся ваши данные и еще раз проверить их. Если при проверке будут обнаружены ошибки, то после их исправления нужно также сначала нажать клавишу <F2> (<F2>) (теперь это запись на диск с именем редактируемого файла (4.9)), затем <Shift>+<F2> (<Верх>+<F2>), в появившейся строке набрать нужное имя (теперь это (4.7) — wm412c8.d31) и нажать клавишу <Enter> (<ВВОД>). Выход из режима редактирования также по <F10> (<F10>). Если под именем (4.9) находится не ваш файл, то нужно внимательнее повторить основные этапы работы, начиная с 1-го.

5. Произвести расчет путем запуска на выполнение файла: имя_файла_программы.exe. Например: statn.exe. Для этого следует подвести подсветку курсора к имя_файла_программы.exe (statn.exe) и нажать клавишу <Enter> (<Ввод>).

6. Нормальное завершение выполнения программы. На экране появляется сообщение “ Stop-Program terminated”. Для просмотра результатов следует нажать клавишу <F3> (<F3>) после совмещения подсветки курсора с именем файла результатов имя_файла_программы.lis (statn.lis). Листание файла выполняется с помощью клавиш <PageUp> (<СтрВверх>) — вперед к концу файла и <PageDn> (<СтрВниз>) — назад к его началу. После окончания просмотра необходимо нажать клавишу <F10> (<F10>). Если результат счета правдоподобен, то перейти к этапу 8.

7. Результат счета неправдоподобен или на экране появляются системные сообщения об ошибках (для их просмотра снятие и появление панелей Norton Commander осуществляется совместным повторным нажатием клавиш <Ctrl>+<O> (<УПР>+<O>)). Следует совместить подсветку курсора с архивным именем файла данных (4.7) и нажатием клавиши <F4> (<F4>) вызвать его на редактирование. Произвести исправление ошибок и повторение сеанса работы с этапа 2.

8. Переименование файла результатов имя_файла_программы.lis (statn.lis) и запись его с архивным именем согласно (4.6) в виде (4.8). Следует совместить подсветку курсора с именем имя_файла_программы.lis (statn.lis), нажать клавишу F6 (<F6>), в появившейся строке набрать имя личного архивного файла (4.8) и нажать клавишу <Enter> (<Ввод>).

9. Поместив копию файла в архив, можно уступить ПЭВМ для работы следующему студенту и заняться проверкой полученных результатов. Теперь файлы данных и результатов записаны на жестком диске с личными архивными идентификаторами (4.7) и (4.8). Они будут сохранены в течение любого нужного нам времени, если после проверки возникнет необходимость в поиске ошибок и их исправлении.

10. Убедившись в правильности решения, выполняем печать результатов расчета. После включения принтера, подключения его к ПЭВМ и заправки бумагой, последовательно подводим подсветку курсора к именам файлов данных (4.7) и результатов (4.8). После каждого раза нажимаем клавишу F5 (<Ф5>) и в появившейся строке набираем `prn`, направляя результаты копирования после нажатия клавиши <Enter> (<Ввод>) на принтер, т.е. на печать.

4.3.2. Поиск ошибок.

Доведение первоначального решения до правильного

Поиск ошибок следует проводить на всех этапах решения задачи.

1. Проверка выбранного рисунка, данных условия задачи и направления реакций опор.

2. Проверка составления уравнений равновесия и правильности замены реакций связей их идентификаторами.

3. Проверка определения расположения и значений ненулевых элементов матриц A и вектора-столбца B .

4. Проверка соответствия введенных в ЭВМ исходных данных из распечатки с теми, которые вы хотели ввести. Данный этап рекомендуется повторять неоднократно перед следующим выходом на ЭВМ, постаравшись найти все сделанные ошибки.

Для доведения первоначального решения до правильного подводим подсветку курсора к имени файла (4.7), нажимаем клавишу <F4> (<Ф4>) и выполняем необходимые исправления. Дальнейший сеанс работы продолжаем с п. 3.

Если после повторного решения проверка также не получается, значит не все сделанные ошибки были найдены сразу, поэтому их поиск и всю последовательность действий при следующем выходе на ЭВМ придется повторить. Последняя редакция файла с исходными данными вашего варианта (4.7) и результатами расчета (4.8) находится в архиве и ждет того момента, когда вы добросовестно выполните все этапы проверки и найдете все сделанные ошибки и неточности.

ГЛАВА 5. ЧИСЛЕННОЕ РЕШЕНИЕ ЗАДАЧ СТАТИКИ НА ПЭВМ С “МАЛЫМ” И “БОЛЬШИМ” КОЛИЧЕСТВОМ УРАВНЕНИЙ РАВНОВЕСИЯ

Рассмотрим решение задач статики с использованием описываемых программ на типовых примерах, приведенных в сборнике [22], которым присвоен 31-й вариант.

5.1. Подготовка данных для решения на ПЭВМ задач по определению усилий в стержнях пространственной конструкции для ССС

5.1.1. Формализация уравнений равновесия

В качестве примера задачи статики с “малым” количеством уравнений равновесия воспользуемся аналитическим решением типового задания по определению усилий в стержнях пространственной конструкции для ССС [22, с. 52-54]. Его уравнения равновесия (1)-(6) после численного определения всех коэффициентов при неизвестных и вычисления свободных членов (не содержащих неизвестных), примут вид:

$$\begin{aligned}
 -2.1192 - 0.7071 S_1 - S_2 - 0.6247 S_3 &= 0, \\
 -2.6491 - 0.7809 S_3 &= 0, \\
 -2.1192 - 0.7071 S_1 &= 0, \\
 0.7071 S_1' + 0.6172 S_6 &= 0, \\
 -S_5 - 0.7715 S_6 &= 0, \\
 0.7071 S_1' + S_4 - 0.1543 S_6 &= 0.
 \end{aligned}
 \tag{5.1}$$

Рассмотрим подготовку исходных данных для численного решения этого примера с использованием программ STAT и STATN.

Отметим, что внутренние силы в первом стержне S_1 и S_1' при мысленном его разбиении для рассмотрения равновесия узлов А и D становятся внешними для каждого из них, выражая действие отброшенной части. Они являются силами действия и противодействия, их модули равны друг другу и они противоположны по направлению. Этот факт выражают векторные равенства:

$$\vec{S}_1 = -\vec{S}_1'.
 \tag{5.2}$$

Обычно векторные уравнения (5.2), выражающие равенство сил действия и противодействия, учитывают на рисунке, направляя соответствующие вектора

для штрихованных и нештрихованных величин в противоположные стороны, как это показано на рис. 53 [22, с.53].

Так как разность направлений штрихованных и нештрихованных величин уже учтена на рисунках (т.е. учтен знак минус в уравнении (5.2)), то этому векторному выражению (5.2) соответствует алгебраическое уравнение (5.3), учитывающее равенство значений этих величин вплоть до знака, так что им можно присвоить один идентификатор:

$$S_1 = S_1' = X_1 \quad (5.3)$$

Эти равенства (5.3) используют при решении задачи. Такой подход мы рекомендуем применять всегда не только при определении усилий в стержнях ферм в п. 5.2, но и при рассмотрении равновесия систем тел при направлении внутренних сил в односторонних связях в п. 6 и промежуточных шарнирах (когда в узле сходятся не более двух стержней).

С учетом вышеизложенного соответствие идентификаторов для рассматриваемого типового примера примет вид:

X_1	X_2	X_3	X_4	X_5	X_6
$S_1=S_1'$	S_2	S_3	S_4	S_5	S_6

Отметим, что при решении задач на определение усилий в стержнях ферм для формализации уравнений равновесия соответствие идентификаторов можно не составлять. Для этого достаточно приравнять друг другу усилие в i -м стержне S_i соответствующему i -му элементу одномерного массива неизвестных X_i : $S_i = X_i$.

Теперь после перенесения свободных членов (не содержащих неизвестных) в правые части уравнений (5.1), они примут следующий формализованный вид (5.4):

$$\begin{aligned}
 1. & -0.7071 S X_1 - X_2 - 0.6247 S X_3 = 2.1192, \\
 2. & -0.7809 S X_3 = 2.6491, \\
 3. & -0.7071 S X_1 = 2.1192, \\
 4. & 0.7071 S X_1 + 0.6172 S X_6 = 0, \\
 5. & -X_5 - 0.7715 S X_6 = 0, \\
 6. & 0.7071 S X_1 + X_4 - 0.1543 S X_6 = 0.
 \end{aligned} \quad (5.4)$$

Для удобства пользования уравнения пронумерованы сверху вниз, начиная с 1, где номером без скобок обозначается порядковый номер строки в системе уравнений (5.4).

5.1.2. Использование универсальной программы STAT

Теперь система уравнений (5.4) приобрела явный вид СЛАУ (4.1), в которой выписаны только отличные от нуля элементы. При использовании программы STAT, которая требует ввода всех элементов матрицы $A(I,J)$ и вектора-столбца $B(I)$, квадратную матрицу A коэффициентов перед неизвестными следует дополнить нулями до ее полного вида NSN . При этом все данные могут быть представлены по желанию как с указанием всех элементов в полном виде:

$$\begin{aligned}
 &0.7071, -1, -0.6247, 0, 0, 0 \\
 &0, 0, -0.7809, 0, 0, 0 \\
 &-0.7071, 0, 0, 0, 0, 0 \\
 &0.7071, 0, 0, 0, 0, 0.6172 \\
 &0, 0, 0, 0, -1, -0.7715 \\
 &0.7071, 0, 0, 1, 0, -0.1543
 \end{aligned} \tag{5.5}$$

так и в сокращенной форме записи с указанием количества (K^*) для повторяющихся нулевых элементов:

$$\begin{aligned}
 &-0.7071, -1, -0.6247, 3*0 \\
 &2*0, -0.7809, 3*0 \\
 &-0.7071, 5*0 \\
 &0.7071, 4*0, 0.6172 \\
 &4*0, -1, -0.7715 \\
 &0.7071, 2*0, 1, 0, -0.1543
 \end{aligned} \tag{5.6}$$

Заметим, что строки данных фрагментов (5.5) и (5.6), которые должны быть введены студентом для рассматриваемого примера задания С-8, соответствуют оператору (4.2).

Дополнив фрагменты (5.5) и (5.6) согласно структуре ввода исходных данных п. 4.2.1 двумя первыми и последней 9-й строкой (в которой перечисляются через запятую все данные вектора-столбца B), получим необходимый вид файла исходных данных `stat.dat` для работы программы STAT:

- с указанием всех элементов в полном виде:

$$\begin{aligned}
 &114413, 8, 31 \\
 &6, 1 \\
 &-0.7071, -1, -0.6247, 0, 0, 0 \\
 &0, 0, -0.7809, 0, 0, 0 \\
 &-0.7071, 0, 0, 0, 0, 0 \\
 &0.7071, 0, 0, 0, 0, 0.6172 \\
 &0, 0, 0, 0, -1, -0.7715 \\
 &0.7071, 0, 0, 1, 0, -0.1543 \\
 &2.1192, 2.6491, 2.1192, 0, 0, 0
 \end{aligned} \tag{5.7}$$

- с использованием повторителей для нулевых элементов:

```

114413, 8, 31
6, 1
-0.7071, -1, -0.6247, 3*0
2*0, -0.7809, 3*0
-0.7071, 5*0
0.7071, 4*0, 0.6172
4*0, -1, -0.7715
0.7071, 2*0, 1, 0, -0.1543
2.1192, 2.6491, 2.1192, 3*0

```

(5.8)

Распечатка результатов решения после окончания работы программы STAT будет иметь вид:

```

*****
РЕЗУЛЬТАТЫ РЕШЕНИЯ СИСТЕМЫ УРАВНЕНИЙ НОМЕР 1
X( 1)= -2.9967 X( 2)= 2.1190 X( 3)= -3.3924
X( 4)= 2.6488 X( 5)= -2.6487 X( 6)= 3.4332
*****

```

5.1.3. Использование универсальной программы STATN

Определим для системы уравнений (5.4) ненулевые элементы матрицы $A(I,J)$ и вектора-столбца $B(I)$. Так как первый индекс I означает номер строки, в которой стоит элемент $A(I,J)$, т.е. это порядковый номер уравнения равновесия, а второй индекс J означает номер столбца, т.е. номер идентификатора X_j , при котором стоит элемент $A(I,J)$, то для матрицы A ненулевыми элементами $A(I,J)$ будут:

- для 1-й строки (уравнение 1.) первый индекс $I=1$, второй индекс J совпадает с номером идентификатора, при котором стоит значение элемента $A(1,J)$: $A(1,1)=-0.7071$ ($J=1$, так как значение этого элемента, равное -0.7071 , стоит при идентификаторе X_1), $A(1,2)=-1$ ($J=2$, так как значение элемента $A(1,2)$ стоит при идентификаторе X_2), $A(1,3)=-0.6247$ ($J=3$, так как стоит при X_3). Напомним, что все остальные элементы первой строки будут равны нулю: $A(1,4)=A(1,5)=A(1,6)=0$.
- для 2-й строки (уравнение 2.) первый индекс $I=2$, второй индекс совпадает с номером идентификатора, при котором стоит значение элемента $A(2,J)$: $A(2,3)=-0.7809$ (значение стоит при X_2 , значит второй индекс $J=2$).
- для 3-й строки (уравнение 3.): $A(3,1)=-0.7071$.

Аналогично далее для оставшихся строк 4-6: $A(4,1)=0.7071$, $A(4,6)=0.6172$, $A(5,5)=-1$, $A(5,6)=-0.7715$, $A(6,1)=0.7071$, $A(6,4)=1$, $A(6,6)=-0.1543$.

Напомним, что в каждой строке по $N=6$ элементов, но значения всех остальных невыписанных элементов равны нулю.

Для вектора-столбца B ненулевыми элементами $B(I)$ будут: для 1-й строки $I=1$ и $B(1)=2.1192$, для 2-й $I=2$ и $B(2)=2.6491$, для 3-й строки $I=3$ и $B(3)=2.1192$.

В столбце также $N=6$ элементов, но остальные равны нулю. Положение элемента вектора — столбца $B(I)$ характеризуется только номером I -й строки, в которой стоит элемент $B(I)$.

Определив положение и значения всех ненулевых элементов матриц A и B , представляем исходные данные для работы программы STATN для рассматриваемого типового примера согласно структуре п. 4.2.2:

$$\begin{aligned}
 &114413, 8, 31 \\
 &6, 1 \\
 &3 \\
 &4, 1, 1, -0.7071, 1, 2, -1., 1, 3, -0.6247, 2, 3, -0.7809 \quad (5.9) \\
 &3, 3, 1, -0.7071, 4, 1, 0.7071, 4, 6, 0.6172 \\
 &5, 5, 5, -1., 5, 6, -0.7715, 6, 1, 0.7071, 6, 4, 1., 6, 6, -0.1543 \\
 &3, 1, 2.1192, 2, 2.6491, 3, 2.1192
 \end{aligned}$$

Здесь первые две строки исходных данных для работы программы STATN совпадают с соответствующими строками данных для программы STAT.

В 3-й строке указано число частей $KOR=3$, на которые разбивается ввод матрицы A . Удобно (но не обязательно) выбирать его так, чтобы ненулевые элементы каждой части с указанием номеров их расположения помещались в одну строку. Тогда количество строк, в которых вводятся ненулевые элементы матрицы A , будет определяться числом KOR (как в примере (5.9)). Это облегчает проверку и поиск ошибок при доведении первоначального решения до правильного, поэтому мы и будем им пользоваться в дальнейшем.

В 4-й – 6-й строках сначала указывается первым количество ненулевых элементов $KNNEKO$ каждой части матрицы A , на которые разбит ее ввод, далее для всех ненулевых элементов данной части указываются номер строки, столбца, значение каждого ненулевого элемента.

В 7-й строке указывается количество ненулевых элементов вектора-столбца B (которое равно 3-м), далее номер строки и значение каждого ненулевого элемента вектора-столбца B .

Заметим, что если число KOR задать равным 1, то исходные данные для ввода ненулевых элементов матрицы (5.5) можно представить следующим образом:

$$\begin{aligned}
 &1 \\
 &12, 1, 1, -0.7071, 1, 2, -1, 1, 3, -0.6247, 2, 3, -0.7809, 3, 1, \quad (5.10) \\
 &-0.7071, 3, 1, -0.7071, 4, 1, 0.7071, 4, 6, 0.6172, 5, 5, -1, \\
 &5, 6, -0.7715, 6, 1, 0.7071, 6, 4, 1, 6, 6, -0.1543
 \end{aligned}$$

Здесь сначала указывается общее количество ненулевых элементов матрицы A , равное 12-ти, а затем идет их перечисление с указанием их расположения. Поэтому по оператору (4.4) двенадцать раз будут считываться номер строки, столбца и значение ненулевого элемента независимо от числа строк, в которых они будут расположены.

Конечно, результаты решения после окончания работы программы STATN совпадут с приведенной в конце п. 5.1.2 распечаткой результатов работы программы STAT.

В заключение отметим, что если требуется определить влияние на результаты изменения нагрузки, геометрических размеров конструкции и т.п., то это можно сделать следующим образом. Сначала нужно задать значение числа M , расположенного 2-м во 2-й строке данных, которое определяет, сколько вариантов решения задачи (т.е. вариантов изменения нагрузки или размеров конструкции) вам необходимо рассмотреть. Затем нужно скопировать $M-1$ раз строки 3-10 файлов (5.7) или (5.8) (для работы программы STAT) или строки 3-7 файла (5.9) (для работы программы STATN). Потом следует внести в изменяющиеся элементы данных необходимые исправления и запустить соответствующие программы STAT или STATN на выполнение. В результате получится решение всех M вариаций задачи.

5.2. Численное решение задач статики на ПЭВМ с “большим” количеством уравнений равновесия

С увеличением количества уравнений равновесия резко возрастают достоинства работы с программой STATN. Они состоят в сокращении вводимых элементов (только ненулевых) и в точном их помещении с указанием места. Все это значительно облегчает ввод данных и поиск допущенных ошибок. Поэтому использование программы STATN для задач статики с “большим” (свыше 12-ти) числом уравнений равновесия становится практически единственно возможным, что мы и рассмотрим на типовом примере численного определения усилий в стержнях плоской фермы [22, с.5-12].

В качестве общего замечания отметим, что для численного определения усилий в стержнях фермы не требуется предварительного нахождения реакций опор, с которого обычно начинается аналитическое решение. Это делается обычно только для упрощения аналитических выкладок, что для численного решения на ЭВМ не имеет никакого значения. Искомые усилия в стержнях, как и реакции опор, получаются из систем уравнений равновесия для всех узлов простой плоской фермы. Поэтому подготовку исходных данных будем начинать сразу с составления соответствия идентификаторов и формализации уравнений равновесия.

Подготовим исходные данные для решения на ПЭВМ с использованием программы STATN системы уравнений равновесия [22, с. 9-11], которая имеет следующий вид:

1. $P + S_2 S \cos(30) = 0,$
2. $-S_1 - S_2 S \cos(60) = 0,$
3. $-S_2' S \cos(30) - S_3' = 0,$
4. $S_2' S \cos(60) - S_6 = 0,$
5. $S_3 + S_5 S \cos(30) = 0,$
6. $S_1' - S_4 - S_5 S \cos(60) = 0,$
7. $-S_5' S \cos(30) - S_7' = 0,$
8. $S_5' S \cos(60) + S_6' - S_{10} = 0,$
9. $S_7 + S_9 S \cos(30) = 0,$
10. $S_4' - S_8 - S_9 S \cos(60) = 0,$
11. $S_8' - R_a = 0,$
12. $-S_9' S \cos(30) + X_b = 0,$
13. $S_9' S \cos(60) + S_{10}' + Y_b = 0.$

Здесь значение $P=11$, а введенные проекции X_b и Y_b неизвестной реакции R_b направлены в сторону положительного направления осей координат рис. 5 [22, с. 10], на котором уже учтена разность направлений штрихованных и нештрихованных реакций связей для каждого стержня. Поэтому вместо векторных уравнений типа (5.2) используются соответствующие скалярные равенства типа (5.3), учитывающие равенство значений этих величин вплоть до знака, так что им можно присвоить один идентификатор: $S_i = S_i' = X_i$.

Соответствие идентификаторов для системы уравнений (5.11)

X1	X2	X3	X4	X5	X6	
S1=S1'	S2=S2'	S3=S3'	S4=S4'	S5=S5'	S6=S6'	
X7	X8	X9	X10	X11	X12	X13
S7=S7'	S8=S8'	S9=S9'	S10=S10'	Ra	Xb	Yb

После численного определения значений тригонометрических функций используемых углов, уравнения (5.11) с учетом соответствия идентификаторов примут следующий формализованный вид:

1. $0.866 S X_2 = -11,$
2. $-X_1 - 0.5 S X_2 = 0,$
3. $-0.866 S X_2 - X_3 = 0,$
4. $0.5 S X_2 - X_6 = 0,$
5. $X_3 + 0.866 S X_5 = 0,$
6. $X_1 - X_4 - 0.5 S X_5 = 0,$
7. $-0.866 S X_5 - X_7 = 0,$

8. $0.5 S X_5 + X_6 - X_{10} = 0,$
9. $X_7 + 0.866 S X_9 = 0,$
10. $X_4 - X_8 - 0.5 S X_9 = 0,$
11. $X_8 - X_{11} = 0,$
12. $-0.866 S X_9 + X_{12} = 0,$
13. $0.5 S X_9 + X_{10} + X_{13} = 0.$

Для удобства пользования уравнения пронумерованы сверху вниз, начиная с 1, где номером без скобок обозначается порядковый номер строки в системе уравнений (5.12). Определив положение и значения всех ненулевых элементов матриц А и В, представляем исходные данные для решения СЛАУ (5.12) с использованием программы STATN согласно структуре п. 4.2.2:

```
114413,1,31
13,1
5
5,1,2,0.866,2,1,-1,2,2,-0.5,3,2,-0.866,3,3,-1
7,4,2,0.5,4,6,-1,5,3,1,5,5,0.866,6,1,1,6,4,-1,6,5,-0.5
7,7,5,-0.866,7,7,-1,8,5,0.5,8,6,1,8,10,-1,9,7,1,9,9,0.866
7,10,4,1,10,8,-1,10,9,-0.5,11,8,1,11,11,-1,12,9,-0.866,12,12,1
3,13,9,0.5,13,10,1,13,13,1
1,1,-11
```

(5.13)

Здесь в 1-й строке, как обычно, указаны номера группы, задания и варианта. Во 2-й: 13 — количество уравнений равновесия в задаче, 1 — число вариантов решения. Для удобства представления исходных данных ввод ненулевых элементов матрицы А разбит на 5 частей (число которых указано в 3-й строке файла данных), соответствующих уравнениям 1-3, 4-6, 7-9, 10-12 и 13 СЛАУ (5.12). В каждой части имеется соответственно 5, 7, 7, 7 и 3 ненулевых элемента, число которых указано в 1-й позиции строк 4-8. В этих строках далее следует перечисление номера строки, столбца, значения каждого ненулевого элемента.

Вектор-столбец В состоит из одного ненулевого элемента, расположенного в 1-й строке СЛАУ (5.12) и имеющего значение минус 11, что и указано в 9-й строке.

Распечатка результатов решения после окончания работы программы STATN с исходными данными (5.13) будет иметь вид:

```
*****
РЕЗУЛЬТАТЫ РЕШЕНИЯ СИСТЕМЫ УРАВНЕНИЙ НОМЕР 1
X( 1)= 6.3510 X( 2)= -12.702 X( 3)= 11.000
X( 4)= 12.702 X( 5)= -12.702 X( 6)= -6.3510
X( 7)= 11.000 X( 8)= 19.053 X( 9)= -12.702
X(10)= -12.702 X(11)= 19.053 X(12)= -11.000
X(13)= 19.053 X(
*****
```

ГЛАВА 6. ЧИСЛЕННОЕ РЕШЕНИЕ ЗАДАЧ СТАТИКИ НА ПЭВМ СО “СРЕДНИМ” КОЛИЧЕСТВОМ УРАВНЕНИЙ РАВНОВЕСИЯ

В качестве примера задачи статики со “средним” количеством уравнений равновесия воспользуемся аналитическим решением типового задания по определению реакций опор составных конструкций с внутренними односторонними связями для системы трех тел (задание С-9 [21, с. 50-59]).

В задаче рассматривается два случая, когда “работает” какая-либо одна из односторонних связей: R_e или R_f , а вторая равна нулю. Соответственно этому необходимо решить на ПЭВМ две системы уравнений равновесия (14) и (24) [21, с. 56 и 57], которые имеют следующий вид:

- 1 случай: $R_f = 0$.

$$\begin{aligned}
 R_e S 2 + M - X_a S 3 + Y_a S 7 &= 0, \\
 X_a + X_c &= 0, \\
 R_e + Y_a + Y_c &= 0, \\
 X_c S 1 + Y_c S 10 - P_2 S 10 &= 0, \\
 -X_c + X_d &= 0, \\
 -Y_c + P_2 + Y_d &= 0, \\
 R_e S 6 + P_1 S \sin(60) S 6 - Q S 1 - Y_d S 2 &= 0, \\
 X_b - X_d - P_1 S \cos(60) &= 0, \\
 -R_e - P_1 S \sin(60) + Y_b - Q - Y_d &= 0.
 \end{aligned} \tag{6.1}$$

- 2 случай: $R_e = 0$.

$$\begin{aligned}
 M - X_a S 3 + Y_a S 7 &= 0, \\
 X_a + X_c &= 0, \\
 Y_a + Y_c &= 0, \\
 R_f S 6 + X_c S 1 + Y_c S 10 - P_2 S 10 &= 0, \\
 -X_c + X_d &= 0, \\
 -R_f - Y_c + P_2 + Y_d &= 0, \\
 -R_f S 4 + P_1 S \sin(60) S 6 - Q S 1 - Y_d S 2 &= 0, \\
 X_b - X_d - P_1 S \cos(60) &= 0, \\
 R_f - P_1 S \sin(60) + Y_b - Q - Y_d &= 0.
 \end{aligned} \tag{6.2}$$

Рассмотрим подготовку исходных данных для численного решения СЛАУ (6.1) и (6.2), отличающихся друг от друга только использованием одной неизвестной величины: (R_e или R_f). Изучим использование не только универсальных программ STAT и STATN, но и специализированных STAT9 и STAT9N, созданных для облегчения выполнения этого задания, поскольку в сборнике [21] оно рекомендовано для применения ЭВМ к решению задач статики.

Составим соответствие идентификаторов для СЛАУ (6.1), присвоив значение реакции $R_e = R_e' = X_1$. Для СЛАУ (6.2) оно будет таким же, только X_1 вместо R_e приравнивается R_f : $R_f = R_f' = X_1$.

Соответствие идентификаторов для СЛАУ (6.1) и (6.2)

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9
$R_e = R_e'$	X_a	Y_a	X_b	Y_b	$X_d = X_d'$	$Y_d = Y_d'$	$X_c = X_c'$	$Y_c = Y_c'$

Перепишем СЛАУ (6.1)–(6.2) с учетом таблицы идентификаторов, перенеся все члены, не содержащие неизвестных, в правую часть и подсчитаем их с учетом условия задачи ($P_1 = 10$ кН, $P_2 = 3$ кН, $M = 20$ кН.м, $Q = 2$ кН). После написания уравнения пронумеруем сверху вниз, для каждой СЛАУ начиная с 1, обозначив номером без скобок порядковый номер строки в системе уравнений:

$$\begin{aligned}
 1. \quad & 2 S X_1 - 3 S X_2 + 7 S X_3 = -M = -20, \\
 2. \quad & X_2 + X_8 = 0, \\
 3. \quad & X_1 + X_3 + X_9 = 0, \\
 4. \quad & X_8 + 10 S X_9 = 10 S P_2 = 30, \\
 5. \quad & X_6 - X_8 = 0, \\
 6. \quad & X_7 - X_9 = -P_2 = -3, \\
 7. \quad & 6 S X_1 - 2 S X_7 = Q - 6 S P_1 S \sin(60) = -49.96, \\
 8. \quad & X_4 - X_6 = P_1 S \cos(60) = 5, \\
 9. \quad & -X_1 + X_5 - X_7 = Q + P_1 S \sin(60) = 10.66;
 \end{aligned} \tag{6.3}$$

$$\begin{aligned}
 1. \quad & -3 S X_2 + 7 S X_3 = -20, \\
 2. \quad & X_2 + X_8 = 0, \\
 3. \quad & X_3 + X_9 = 0, \\
 4. \quad & 6 S X_1 + X_8 + 10 S X_9 = 30, \\
 5. \quad & X_6 - X_8 = 0, \\
 6. \quad & -X_1 + X_7 - X_9 = -3, \\
 7. \quad & -4 S X_1 - 2 S X_7 = -49.96, \\
 8. \quad & X_4 - X_6 = 5, \\
 9. \quad & X_1 + X_5 - X_7 = 10.66.
 \end{aligned} \tag{6.4}$$

Представим формализованные СЛАУ (6.3) и (6.4) в матричной форме в виде $(A_1)(X_1) = B_1$ и $(A_2)(X_2) = B_2$. Из их сравнения видно, что матрицы A_1 и A_2 отличаются только первым столбцом (“работает” реакция R_e или R_f), а вектора-столбцы B_1 и B_2 совпадают, так как нагрузка не меняется.

6.1. Подготовка исходных данных для работы программ STAT и STAT9

6.1.1. Использование программы STAT

Программы STAT и STAT9 предусматривают для своей работы ввод всех значений исходных данных в полном виде или с использованием повторителей для нулевых элементов.

Так как программа STAT является универсальной, то решение СЛАУ (6.3) и (6.4) будем рассматривать как задачу с двумя вариантами, число которых вводится вторым во 2-й строке файла данных (см. п. 4.2.1). Поэтому все значения элементов матрицы A_2 и вектора-столбца B_2 вводятся вслед за A_1 и B_1 . Дополним СЛАУ (6.3) и (6.4) нулевыми элементами до их полного вида и представим исходные данные согласно п. 4.2.1:

- с указанием всех элементов в полном виде:

$$\begin{aligned}
 &114413, 9, 31 \\
 &9, 2 \\
 &2, -3, 7, 0, 0, 0, 0, 0, 0 \\
 &0, 1, 0, 0, 0, 0, 0, 0, 1, 0 \\
 &1, 0, 1, 0, 0, 0, 0, 0, 0, 1 \\
 &0, 0, 0, 0, 0, 0, 0, 0, 1, 10 \\
 &0, 0, 0, 0, 0, 1, 0, -1, 0 \\
 &0, 0, 0, 0, 0, 0, 1, 0, -1 \\
 &6, 0, 0, 0, 0, 0, -2, 0, 0 \\
 &0, 0, 0, 1, 0, -1, 0, 0, 0 \\
 &-1, 0, 0, 0, 1, 0, -1, 0, 0 \\
 &-20, 0, 0, 30, 0, -3, -49.96, 5, 10.66 \\
 &0, -3, 7, 0, 0, 0, 0, 0, 0, 0 \\
 &0, 1, 0, 0, 0, 0, 0, 0, 1, 0 \\
 &0, 0, 1, 0, 0, 0, 0, 0, 0, 1 \\
 &6, 0, 0, 0, 0, 0, 0, 0, 1, 10 \\
 &0, 0, 0, 0, 0, 1, 0, -1, 0 \\
 &-1, 0, 0, 0, 0, 0, 1, 0, -1 \\
 &-4, 0, 0, 0, 0, 0, -2, 0, 0 \\
 &0, 0, 0, 1, 0, -1, 0, 0, 0 \\
 &1, 0, 0, 0, 1, 0, -1, 0, 0 \\
 &-20, 0, 0, 30, 0, -3, -49.96, 5, 10.66
 \end{aligned} \tag{6.5}$$

- с использованием повторителей для нулевых элементов:

```

114413, 9, 31
9, 2
2, -3, 7, 6*0
0, 1, 5*0, 1, 0
1, 0, 1, 5*0, 1
7*0, 1, 10
5*0, 1, 0, -1, 0
6*0, 1, 0, -1
6, 5*0, -2, 2*0
3*0, 1, 0, -1, 3*0
-1, 3*0, 1, 0, -1, 2*0
-20, 2*0, 30, 0, -3, -49.96, 5, 10.66
0, -3, 7, 6*0
0, 1, 5*0, 1, 0
0, 0, 1, 5*0, 1
6, 6*0, 1, 10
5*0, 1, 0, -1, 0
-1, 5*0, 1, 0, -1
-4, 5*0, -2, 2*0
3*0, 1, 0, -1, 3*0
1, 3*0, 1, 0, -1, 2*0
-20, 2*0, 30, 0, -3, -49.96, 5, 10.66

```

(6.6)

Здесь в каждом из файлов данных (6.5) и (6.6) в 1-й строке указаны номера группы, задания и варианта. Во 2-й: 9 — количество уравнений равновесия, 2 — число вариантов решения. Далее поэлементно и построчно представлены все значения матрицы A_1 (строки 3-я–11-я), вектора-столбца B_1 (12-я), матрицы A_2 (13-я–21-я), вектора-столбца B_2 (22-я строка).

6.1.2. Использование программы STAT9

Программа STAT9 специализирована для решения задания С-9 [21]. Она предусматривает, что матрица A_2 отличается от матрицы A_1 только первым столбцом, а векторы-столбцы B_1 и B_2 тождественны. Поэтому в программе предусмотрено копирование вектора-столбца B_1 для B_2 и матрицы A_1 в A_2 . Затем в матрице A_2 замещается первый столбец по оператору `READ *, A(I, 1)`, стоящему в цикле, повторяющем его 9 раз.

Таким образом, ввод исходных данных в строках 1-12 для программы STAT и STAT9 полностью одинаков (только для STAT9 не указывается количество вариантов решения задачи, стоящее 2-м во 2-й строке данных). Он может быть представлен в любой из форм фрагментов (6.5) или (6.6).

Значения элементов первого столбца матрицы A_2 располагаются в следующих $N=9$ строках (13-й-21-й) файла данных вслед за вектором-столбцом B_1 по одному элементу в строку, а вектор-столбец B_2 не вводится вследствие его получения копированием из B_1 в самой программе. Поэтому для СЛАУ (6.3) и (6.4) файл данных для работы программы STAT9 может иметь, например, следующий вид с использованием повторителей для ненулевых элементов:

```

103166, 9, 31
9
2, -3, 7, 6*0
0, 1, 5*0, 1, 0
1, 0, 1, 5*0, 1
7*0, 1, 10
5*0, 1, 0, -1, 0
6*0, 1, 0, -1
6, 5*0, -2, 2*0
3*0, 1, 0, -1, 3*0
-1, 3*0, 1, 0, -1, 2*0
-20, 2*0, 30, 0, -3, -49.96, 5, 10.66
0
0
0
6
0
-1
-4
0
1

```

(6.7)

6.2. Подготовка исходных данных для работы программ STATN и STAT9N

6.2.1. Определение положения ненулевых элементов

Программы STATN и STAT9N предусматривают для своей работы ввод только значений ненулевых элементов исходных данных. Предварительно в самой программе производится обнуление всех рабочих массивов (задание нулевых элементов), после чего на указанные места вводятся значения ненулевых элементов.

Напомним, что положение элемента $A(I,J)$ в матрице A характеризуется двойным индексом. Первый индекс I означает номер строки, в которой стоит элемент $A(I,J)$, т.е. это порядковый номер уравнения равновесия. Вторым индексом J означает номер столбца, т.е. номер идентификатора X_j , при котором стоит элемент $A(I,J)$. Отметим, что нумерация строк производится сверху вниз, а столбцов — слева направо.

Для матрицы A_1 СЛАУ (6.3) ненулевыми элементами $A_1(I,J)$ будут:

- для 1-й строки (уравнение 1.) первый индекс $I=1$, второй индекс J совпадает с номером идентификатора, при котором стоит значение элемента $A_1(1,J)$: $A_1(1,1)=2$ ($J=1$, так как значение этого элемента, равное двум, стоит при идентификаторе X_1), $A_1(1,2)=-3$ ($J=2$, так как значение элемента $A_1(1,2)$ стоит при идентификаторе X_2), $A_1(1,3)=7$ ($J=3$, так как стоит при X_3). Напомним, что все остальные элементы первой строки будут равны нулю: $A_1(1,4)=A_1(1,5)=\dots=A_1(1,9)=0$.
- для 2-й строки (уравнение 2.) первый индекс $I=2$, второй индекс совпадает с номером идентификатора, при котором стоит значение элемента $A_1(2,J)$: $A_1(2,2)=1$ (значение стоит при X_2 , значит второй индекс $J=2$), $A_1(2,8)=1$ (значение стоит при X_8 — второй индекс $J=8$).
- для 3-й строки (уравнение 3.): $A_1(3,1)=1$, $A_1(3,3)=1$, $A_1(3,9)=1$.

Аналогично далее для оставшихся строк 4–9: $A_1(4,8)=1$, $A_1(4,9)=10$, $A_1(5,6)=1$, $A_1(5,8)=-1$, $A_1(6,7)=1$, $A_1(6,9)=-1$, $A_1(7,1)=6$, $A_1(7,7)=-2$, $A_1(8,4)=1$, $A_1(8,6)=-1$, $A_1(9,1)=-1$, $A_1(9,5)=1$, $A_1(9,7)=-1$.

Положение элемента вектора-столбца $B(I)$ характеризуется только номером I -й строки, в которой стоит элемент $B(I)$. Поэтому для вектора-столбца B_1 ненулевыми элементами будут: $B_1(1)=-20$, $B_1(4)=30$, $B_1(6)=-3$, $B_1(7)=-49.96$, $B_1(8)=5$, $B_1(9)=10.66$.

Выполнив аналогичные действия для 2-го случая, когда $Re=0$, определим ненулевые элементы матрицы A_2 СЛАУ (6.4):

$$\begin{aligned} A_2(1,2) &= -3, & A_2(1,3) &= 7, & A_2(2,2) &= 1, & A_2(2,8) &= 1, & A_2(3,3) &= 1, \\ A_2(3,9) &= 1, & A_2(4,1) &= 6, & A_2(4,8) &= 1, & A_2(4,9) &= 10, & A_2(5,6) &= 1, \\ A_2(5,8) &= -1, & A_2(6,1) &= -1, & A_2(6,7) &= 1, & A_2(6,9) &= -1, \\ A_2(7,1) &= -4, & A_2(7,7) &= -2, & A_2(8,4) &= 1, & A_2(8,6) &= -1, \\ A_2(9,1) &= 1, & A_2(9,5) &= 1, & A_2(9,7) &= -1. \end{aligned} \quad (6.8)$$

Матрица A_2 отличается от матрицы A_1 только первым столбцом, так как изменилась только реакция $Rf=X_1$. Заметим, что любой элемент первого столбца имеет второй индекс $J=1$: $A_2(I,1)$. Отличающимися ненулевыми элементами матрицы A_2 , учитывающими вклад реакции $Rf=Rf'$ в уравнения равновесия, будут:

$$A_2(4,1)=6, \quad A_2(6,1)=-1, \quad A_2(7,1)=-4, \quad A_2(9,1)=1. \quad (6.9)$$

6.2.2. Использование программы STATN

Определив положение и значения всех ненулевых элементов матриц A_1 и B_1 , A_2 и B_2 представляем исходные данные для решения СЛАУ (6.3) и (6.4) с использованием программы STATN согласно структуре п. 4.2.2:

```

114413, 9, 31
9, 2
3
8, 1, 1, 2, 1, 2, -3, 1, 3, 7, 2, 2, 1, 2, 8, 1, 3, 1, 1, 3, 3, 1, 3, 9, 1
6, 4, 8, 1, 4, 9, 10, 5, 6, 1, 5, 8, -1, 6, 7, 1, 6, 9, -1
7, 7, 1, 6, 7, 7, -2, 8, 4, 1, 8, 6, -1, 9, 1, -1, 9, 5, 1, 9, 7, -1
6, 1, -20, 4, 30, 6, -3, 7, -49.96, 8, 5, 9, 10.66 (6.10)
3
6, 1, 2, -3, 1, 3, 7, 2, 2, 1, 2, 8, 1, 3, 3, 1, 3, 9, 1
8, 4, 1, 6, 4, 8, 1, 4, 9, 10, 5, 6, 1, 5, 8, -1, 6, 1, -1, 6, 7, 1, 6, 9, -1
7, 7, 1, -4, 7, 7, -2, 8, 4, 1, 8, 6, -1, 9, 1, 1, 9, 5, 1, 9, 7, -1
6, 1, -20, 4, 30, 6, -3, 7, -49.96, 8, 5, 9, 10.66

```

Здесь представление исходных данных в первых двух строках полностью совпадает с любым из возможных видов файла данных (6.5) или (6.6) для работы программы STAT.

В строке 3-й указывается количество частей, на которые разбивается ввод матрицы A_1 СЛАУ (6.3), которое совпадает с количеством объектов равновесия $KOR=3$. В следующих трех строках (4–6) в 1-й позиции указывается количество ненулевых элементов матрицы A_1 8, 6 и 7, которое соответственно имеется в уравнениях 1–3, 4–6, 7–9 СЛАУ (6.3). Далее в строках 4–6 файла данных (6.10) следует перечисление номера строки, столбца, значения каждого ненулевого элемента. Вектор-столбец B_1 СЛАУ (6.3) состоит из 6-ти ненулевых элементов, значения которых с предварительным указанием их расположения представлены в 7-й строке.

Затем в строках 8–12 следует аналогичное представление всех ненулевых элементов матриц A_2 и B_2 СЛАУ (6.4), как и в строках 3–7 для A_1 и B_1 СЛАУ (6.3).

6.2.3. Использование программы STAT9N

Файл исходных данных для работы программы STAT9N совпадает в строках 1–7 с файлом данных (6.10) для программы STATN за исключением того, что во второй строке указывается только количество уравнений равновесия (цифра 9). Программа STAT9N, как и STAT9, специализирована для решения задания С-9 [21, с. 50-59]. В самой программе для сокращения ввода данных производится копирование одинаковых элементов (вектора-столбца B_1 в B_2 и матрицы A_1 в A_2). В матрице A_2 в программе затем обнуляется первый столбец

(присваиваются нулевые значения всем элементам $A_2(I,1)$), после чего вводятся только отличающиеся ненулевые элементы первого столбца второй матрицы A_2 (6.9) с предварительным указанием их количества:

4, 4, 1, 6, 6, 1, -1, 7, 1, -4, 9, 1, 1

Таким образом весь файл данных для решения СЛАУ (6.3)–(6.4) с использованием программы STAT9N будет иметь вид:

```
114413, 9, 31
9
3
8, 1, 1, 2, 1, 2, -3, 1, 3, 7, 2, 2, 1, 2, 8, 1, 3, 1, 1, 3, 3, 1, 3, 9, 1
6, 4, 8, 1, 4, 9, 10, 5, 6, 1, 5, 8, -1, 6, 7, 1, 6, 9, -1          (6.11)
7, 7, 1, 6, 7, 7, -2, 8, 4, 1, 8, 6, -1, 9, 1, -1, 9, 5, 1, 9, 7, -1
6, 1, -20, 4, 30, 6, -3, 7, -49.96, 8, 5, 9, 10.66
4, 4, 1, 6, 6, 1, -1, 7, 1, -4, 9, 1, 1
```

6.3. Описание результатов работы программ STAT9N, STAT9, STATN, STAT

Распечатка результатов решения после окончания работы любой из четырех описываемых программ (STATN, STAT9N, STAT9 и STAT) будет иметь вид ($X(1)=Rf$):

```
*****
X (1)=9.9490 X (2)=11.023 X (3)=1.8671 X (4)=-6.0232 X (5)=5.7929
X (6)=-11.023 X (7)=5.0819 X (8)=-11.023 X (9)=-1.8671 X (
*****
```

Для облегчения поиска ошибок производится контрольная печать входной матрицы A и вектора-столбца B с указанием номера расположения каждого элемента, а также определяется оценка обусловленности матрицы COND [27]. Отметим, что если значение реакция $X(1)$ меньше нуля, то результаты решения на печать программами STAT9N и STAT9 не выдаются.

6.4. Проверка результатов решения и поиск ошибок

Ошибки при решении задачи статики на ПЭВМ с использованием описываемых программ можно разбить на два типа: явные и скрытые.

Явные ошибки препятствуют работе программы. К ним относятся: прекращение выполнения программы при системном прерывании или при обнаружении вырожденности (особенности) матрицы, причиной чего могут

быть грубые ошибки в составлении уравнений равновесия или ввода исходных данных. Программы STAT9N и STAT9 также прекращают работу без выдачи результатов решения при получении обоих отрицательных ответов для реакций Re и Rf.

Скрытые ошибки не препятствуют работе программы, но приводят к получению неправильных ответов. ПЭВМ в этом случае дает правильные результаты решения той системы уравнений, которую в нее ввели, поэтому внешний вид распечатки не отличается от правильной. Но введенная в ЭВМ система уравнений не соответствует той, которую вам нужно решить, поэтому результаты решения для вашего варианта будут неправильные. Следует сделать проверку, для чего нужно убедиться в том, что соблюдаются уравнения равновесия для сил, приложенных ко всей конструкции. При наличии скрытых ошибок хотя бы одно из уравнений равновесия не будет соблюдаться. К этому типу ошибок относятся все ошибки составления уравнений и ввода исходных данных, которые не препятствуют выполнению программы.

В этом случае следует внимательно проверить все этапы решения, начиная от составления уравнений равновесия, определения значений ненулевых элементов и номеров их расположения, обратив внимание на соответствие введенных вами в ЭВМ исходных данных, представленных на распечатке, с теми, которые вы хотели ввести.

Эту длительную и кропотливую работу ни в коем случае не следует прекращать при нахождении первой ошибки. Каждую вновь найденную ошибку в знаке, значении или расположении элемента ни в коем случае нельзя считать последней. Она должна скорее служить стимулом для поиска оставшихся ошибок, а не сигналом для прекращения усилий. Работу по проверке соответствия распечатанных данных с теми, которые вы хотели ввести, желательно повторить несколько раз. Заметим также, что лучше один раз четко и внимательно выполнить с самого начала все этапы работы, проверяя каждый из них по мере выполнения несколько раз, чем выискивать потом весьма труднонаходимые ошибки, сделанные из-за небрежности и невнимательности.

ЧАСТЬ 3. СОСТАВЛЕНИЕ ПРОГРАММ НА ФОРТРАНЕ ДЛЯ РЕШЕНИЯ ЗАДАЧ СТАТИКИ

ГЛАВА 7. ОСНОВНЫЕ СВЕДЕНИЯ ПО АЛГОРИТМИЧЕСКОМУ ЯЗЫКУ ФОРТРАН

Язык программирования Фортран вот уже почти 30 лет является основным для научно-технических приложений. Популярность Фортрана объясняется простотой его синтаксических конструкций, универсальностью и мобильностью программ, а также накопленным за это время колоссальным программным обеспечением. За прошедшие годы Фортран менялся и развивался, что нашло свое отражение в трех стандартах этого языка: Фортран-66, Фортран-77 и Фортран-90.

Однако усовершенствование Фортрана привело к его существенному усложнению: описание только дополнительных возможностей Фортрана-90 в пособии [24] в полтора раза превышает описание обеих предыдущих его версий в работе [8]. Одновременно с этим все трансляторы Фортрана совместимы снизу вверх, что позволяет весь багаж программ, созданных на Фортране, сохранить для дальнейшей эксплуатации.

Последнее обстоятельство позволяет при первоначальном изучении этого языка ограничиваться практическим овладением его простых основ или ядра, чему и посвящено настоящее пособие. После обретения достаточно устойчивых навыков программирования, вы можете, при возникновении необходимости, без затруднений перейти к использованию усовершенствований языка Фортран.

Такой подход позволяет уверенно работать на любых IBM-совместимых ЭВМ. Все приведенные в пособии программы и дополнения проверялись как на персональных IBM PC, так и “больших” IBM-360/370, отечественные аналоги которых соответственно ЕС-1841-1863 и ЕС-1033-1066. При этом использовались получившие наибольшее распространение на ЭВМ соответствующего типа трансляторы: для ПЭВМ — MS-Fortran 5.0 и операционная среда MS DOS, а для “больших” ЭВМ — FORTSE и ОС CBM. Небольшие отличия, касающиеся ввода-вывода данных на ПЭВМ, приведены в п. 7.6.3. Соответствующие операторы напечатаны в тексте малыми латинскими бук-

вами, чтобы выделить их использование только для персональных ЭВМ (на которых в командах и операторах фортран-программ строчные буквы интерпретируются как прописные).

7.1. Описание типов чисел и переменных

Числа в языке Фортран используются двух типов: целые и действительные. Целые числа представляют собой последовательность десятичных цифр со знаком +, – или без знака. О них говорят также, что они представлены в форме I.

Действительные или вещественные числа записываются в двух формах. В основной форме (форме F) число записывается в виде целой и дробной частей, разделенных точкой, со знаком +, – или без знака. Целая или дробная часть числа может отсутствовать, если она равна нулю, но точка присутствует обязательно. Запись числа в форме E соответствует записи чисел в показательной форме, где десятичный порядок обозначается буквой E, а следующая за ней целая константа со знаком или без определяет величину порядка. Например, .15E3 или 150., .15E-3 или 0.00015. Числа в обеих формах F и E могут представляться с обычной или удвоенной точностью. В последнем случае для обозначения порядка вместо буквы E используется буква D.

Переменные — это величины, которым присвоены наименования. Для каждой переменной в памяти машины отводится место, в котором хранится ее значение. В Фортране для обозначения переменных используются символические имена, которые представляют собой набор от 1 до 6 букв или цифр, начинающийся с буквы: N, A1, B2, WORK, KNNEKO. В качестве букв используются буквы латинского алфавита от A до Z, цифры: 0, 1, ..., 9, цифра 0 на ЭВМ перечеркивается, чтобы отличить ее от буквы O.

Тип переменной соответствует типу данных, которые она представляет. Так, целые переменные представляют целые числа, вещественные переменные — вещественные числа и т.д. В программе должно содержаться указание о том, какие значения (целые или вещественные) будут принимать используемые переменные, и с какой точностью (обычной или удвоенной) они должны быть представлены.

Если в программе нет других указаний, то переменные, имена которых начинаются с букв I, J, K, L, M, N — являются целыми, а начинающиеся с любой другой буквы — вещественными. Такое определение типа переменной с помощью первой буквы ее символического имени называется неявным описанием типа. Оно используется для переменных обычной точности.

В Фортране при необходимости можно обойти это ограничение с помощью явного оператора описания типа, который определяет тип переменных и массивов по их полным именам. Он относится к невыполняемым операторам Фортрана. Явный оператор описания типа в простейшем случае имеет вид:

```
тип A1, A2, ..., AK
```

где тип — одно из слов INTEGER (целый), REAL (вещественный), DOUBLE PRECISION (двойной точности); A1, A2, ..., AK — имена переменных. Например, операторы: INTEGER N,D и REAL L1 описывают переменные N и D как целые, а переменную L1 как вещественную. Отметим, что хорошим стилем программирования является выбор имен переменных, которые по смыслу соответствуют описываемым величинам.

Переменные двойной точности всегда должны описываться явно с помощью операторов описания типа DOUBLE PRECISION или REAL *8. Нижеследующие операторы эквивалентны и описывают вещественные переменные COND, T, TOUT, RELERR, ABSERR двойной точности:

```
DOUBLE PRECISION COND, T, TOUT, RELERR, ABSERR
REAL *8 COND, T, TOUT, RELERR, ABSERR
```

Стандартная длина переменных целого и вещественного типа — 4 байта, нестандартная длина переменных целого типа 2 байта, вещественного типа — 8 байт (удвоенная точность).

7.2. Переменные с индексами. Описание массивов

Массив — это упорядоченная последовательность величин, называемых элементами массива, обозначаемая одним символическим именем. Каждый элемент массива определяется именем массива и его положением в массиве, т.е. значениями индексов. Индексы заключаются в круглые скобки и разделяются запятыми, причем для двумерного массива первый индекс определяет номер строки, второй — номер столбца. При обозначении массивов и индексов используются те же правила задания типа, что и при обозначении переменных.

Массивы, используемые в программе, должны быть обязательно описаны. Описание массивов дается в начале программы и содержит информацию об именах, размерностях и максимальных размерах используемых массивов. Для описания массивов обычно используется оператор DIMENSION, общий вид которого:

```
DIMENSION имя-массива
(верхние границы изменения индексов через запятую), ...
```

Например, оператор

$$\text{DIMENSION A}(9, 9), \text{B}(9), \text{IPVT}(9) \quad (7.1)$$

описывает двумерный вещественный массив A с числом строк и столбцов равным 9, для которого обеспечивается резервирование памяти для хранения $9 \cdot 9 = 81$ значения элементов A , одномерные вещественный B и целый $IPVT$ массивы, состоящие из 9 элементов соответственно каждый.

Следует отличать размерность массива от размера массива. Размерность массива — число измерений массива (т.е. число индексов, определяющих положение элемента в массиве). Размер массива — число элементов в массиве. Поэтому в примере (7.1) матрица A является двумерным массивом, размер массива $A=81$ (массив содержит 81 элемент).

Не накладывается никаких ограничений на число переменных в одном предложении `DIMENSION`. Он относится к неисполняемым операторам в том смысле, что не порождает команд в программе: по его описанию выделяется место в памяти для размещения всех указанных массивов.

Вместо термина “двумерный массив” часто употребляется термин “матрица”. Отдельный член массива называется элементом. В задачах статики число строк (число уравнений равновесия N), всегда равняется числу столбцов (количеству неизвестных в задаче), поэтому матрица всегда квадратная $A(N, N)$:

$$\begin{aligned} &A(1, 1), A(1, 2), A(1, 3), \dots, A(1, N) \\ &A(2, 1), A(2, 2), A(2, 3), \dots, A(2, N) \\ &A(3, 1), A(3, 2), A(3, 3), \dots, A(3, N) \\ &\dots\dots\dots \\ &A(N, 1), A(N, 2), A(N, 3), \dots, A(N, N) \end{aligned} \quad (7.2)$$

Первый индекс обозначает номер строки элемента, а второй — номер столбца. Например, элемент $A(2, 3)$ находится во второй строке в третьем столбце. В операторе `DIMENSION` для описания массивов с двумя индексами первое число дает количество строк, а второе количество столбцов.

Одномерные массивы иногда называют векторами. В них положение элемента характеризуется номером I -й строки, в которой стоит элемент $V(I)$. Значения этих элементов хранятся на линейном участке памяти друг за другом: $V(1), V(2), V(3), \dots, V(I)$.

Отметим, что для двумерного массива выделяется также линейный участок памяти, в котором значения элементов (при обычном вводе матрицы — см. п. 7.8.2) хранятся по столбцам. При этом первый индекс является первым изменяющимся индексом. Например, элементы матрицы (7.2) хранятся в таком порядке:

```

A(1,1), A(2,1), A(3,1), ..., A(N,1)
A(1,2), A(2,2), A(3,2), ..., A(N,2)
A(1,3), A(2,3), A(3,3), ..., A(N,3)
.....
A(1,N), A(2,N), A(3,N), ..., A(N,N)

```

Обратите внимание, что быстрее изменяется первый индекс, затем второй: сначала располагается 1-й столбец — значения элементов от $A(1,1)$ до $A(N,1)$, затем 2-й, 3-й, ..., N -й столбец — значения элементов от $A(1,N)$ до $A(N,N)$ (см. п. 9.2.1).

Для описания массивов можно использовать, кроме оператора DIMENSION, также и операторы явного описания типа, причем максимальные значения индексов записываются в той же форме. Нижеследующие два оператора эквивалентны оператору (7.1):

```

REAL A(9,9), B(9)
INTEGER IPVT(9)

```

(7.3)

Для описания вещественных массивов двойной точности часто используется оператор REAL *8, позволяющий совместить описание типа с описанием массива, например:

```

REAL *8 A(9,9), B(9)

```

7.3. Арифметические выражения и системные функции

Обозначение арифметических операций в Фортране следующее: + сложение, − вычитание, * умножение, / деление, ** возведение в степень. Например: $B ** 2 - 4 * A * C$.

Все знаки арифметических действий должны присутствовать в выражении явно. Математическое выражение IJ (значение I умножается на значение J) в Фортране должно быть написано обязательно так: I*J, иначе оно будет понято транслятором как символическое имя IJ и возникнет трудноуловимая ошибка, не препятствующая работе программы.

Рядом не могут находиться два знака арифметических действий, они должны отделяться скобками, например: $R * (-0.01)$. Использование круглых скобок вообще желательно в сомнительных случаях для ясности записи и уверенности, что выражение будет вычисляться правильно. Нужно только следить, чтобы количество открывающих скобок совпадало с количеством закрывающих скобок.

Все арифметические операции, как и скобки, имеют обычный смысл. К особым случаям относятся:

1) в качестве результата деления целой величины на целую берется целая часть частного;

2) в качестве результата степени, основание которой — целая величина, а показатель — целая отрицательная величина, берется также его целая часть;

3) если $X < 0$, а Y — величина вещественного типа, то операция $X^{**}Y$ не имеет смысла, т.к. возведение в вещественную степень осуществляется с помощью логарифмов. Но отрицательная величина может быть возведена в целую степень, т.к. операция возведения в целую степень выполняется как многократное умножение.

Использование в одном выражении величин разных типов разрешено, но не рекомендуется, т.к. в этом случае перед вычислением будет производиться преобразование типа величин, что приведет к построению неэффективной объектной программы.

Следует принимать во внимание, что непосредственное умножение производится быстрее, чем деление или расчет экспоненты для вычисления квадратов. Поэтому запись $X=A/(B^*C)$ более экономична, чем $X=A/B/C$, а для оптимизации, например, следующей последовательности выражений: $X=(A^{**2}+B^{**2})^{**2}$; $Y=1/(A^{**2}+B^{**2})$; $Z=A^{**2}+B^{**2}-1/(A^{**2}+B^{**2}-3)$ программисту следовало бы представить их в виде:

```
AB2=A*A+B*B
X=AB2*AB2
Y=1./AB2
Z=AB2-1./(AB2-3.)
```

Наиболее используемые математические функции оформляются стандартным образом в подпрограммы и называются системными функциями. Для их использования достаточно записать название функции со стоящим после него в круглых скобках аргументом, в качестве которого может быть постоянная, переменная или любое арифметическое выражение, включающее другие функции или даже ту же самую функцию.

К основным системным функциям относятся (вещественный аргумент для краткости обозначен через X):

$SQRT(X)$ — вычислить квадратный корень;

$SIN(X)$, $COS(X)$ или $TAN(X)$ — вычислить соответственно синус, косинус или тангенс угла в радианах;

$ARSIN(X)$, $ARCOS(X)$ или $ATAN(X)$ — вычислить арксинус, арккосинус или арктангенс;

$EXP(X)$, $ALOG(X)$ или $ALOG10(X)$ — вычислить экспоненту, натуральный или десятичный логарифм соответственно;

$ABS(X)$ — вычислить абсолютную величину (модуль).

При работе с двойной точностью аргумент задается также с двойной точностью, а вышеуказанные функции соответственно принимают вид (имя функции начинается с буквы D): DSQRT(X), DSIN(X), DCOS(X), DTAN(X), DARSIN(X), DARCOS(X), DATAN(X), DEXP(X), DLOG(X), DLOG10(X), DABS(X).

Примеры записи системных функций: SIN(3.*3.14159*T*T/4.), SQRT(ALOG(1.+SQRT(X*X+Y*Y+Z*Z))).

7.4. Арифметический оператор присваивания

Для вычисления арифметических выражений и присваивания их значений переменным служит арифметический оператор присваивания. Этот оператор заменяет значение переменной в левой части оператора на значение выражения, стоящего в его правой части, и имеет вид:

переменная = выражение

Между арифметическим оператором присваивания и похожим на него алгебраическим равенством существует очень важное различие. Оператор Фортрана $X=Y+Z$ означает для машины: взять содержимое ячейки памяти, соответствующей переменной Y , сложить с содержимым ячейки переменной Z и передать результат в ячейку памяти, соответствующую переменной X . Знак равенства в арифметическом операторе присваивания имеет значение “необходимо заменить на ...”, а в математике знак равенства означает “то же самое значение, что и ...”.

В Фортране допустимы и часто используются конструкции типа:

$X=X+1.0$ (или $I=I+1$)

С точки зрения обычного алгебраического равенства такое равенство бессмысленно, но в Фортране оно означает: взять содержимое ячейки памяти X (или I), прибавить к нему число 1 и отослать результат обратно в ячейку переменной X (или I), заменив бывшую там информацию на новую. Если, например, ранее в программе было $X=3.0$ (или $I=3$), то после выполнения этого оператора величина X станет равной 4.0 ($I=4$ соответственно), и это число будет использоваться везде в качестве значения X (или I), пока оно не будет снова изменено каким-либо оператором присваивания.

В левой части арифметического оператора присваивания может находиться только простая переменная, а не выражение. Всем переменным перед их использованием в программе должны быть установлены начальные значения (см. п. 7.7).

Если тип переменной в левой части оператора присваивания не совпадает с типом выражения, то после вычисления значения выражения, оно преобразуется в число такого типа, который соответствует левой части оператора присваивания (для целых и вещественных переменных обычной точности).

Например, если $I=4$ и $J=2$, то по оператору $X=I/J$, значение правой части, равное 2, преобразуется к вещественному типу и будет $X=2.0$. Это не приведет к ошибкам. Но для оператора $L=Y/Z$ при $Y=8.0$ и $Z=5.0$, значение правой части, равное 1.6, преобразуется к целому типу путем прямого усечения и будет $L=1$. Это приведет к труднонаходимой ошибке расчета. Поэтому при записи оператора присваивания желательнее использовать однотипные переменные и числа. Чем меньше будет в программе лишних преобразований величин из одного типа в другой, тем она будет эффективнее.

7.5. Операторы управления

7.5.1. Оператор GO TO

Все операторы программы выполняются сверху вниз в той последовательности, в которой они написаны, начиная с первого выполняемого оператора. В случае необходимости для изменения этой последовательности применяются операторы управления. После осуществления перехода в соответствии с оператором управления операторы программы снова выполняются последовательно до тех пор, пока не встретится другой оператор управления. Передача управления может быть условной (см. пп. 7.5.2 и 7.5.3) и безусловной.

Оператор безусловного перехода GO TO (или GOTO, что в Фортране равнозначно) имеет вид:

```
MT GO TO MK
```

где MT — либо пусто, либо метка, помечающая данный оператор GO TO; MK — метка некоторого другого оператора, которому передается управление. Так, например, после выполнения оператора 5 GO TO 40 будет выполняться оператор, помеченный меткой 40, который может быть расположен до или после оператора передачи управления с меткой 5.

Метка является номером оператора, содержит от одной до пяти цифр, расположенных в позициях 1-5 строки бланка программы или экрана дисплея. Напомним, что все операторы программы располагаются обычно с 7-й позиции строки. Любой символ, отличный от 0 или пробела, расположенный в 6-й позиции, является знаком переноса и указывает на продолжение вышераспо-

ложенного оператора. Знаки арифметических действий при переносе не дублируются, как в обычной записи.

Строка комментария, его продолжения или любых пояснений к программе начинается с символа комментария: в 1-й позиции латинская буква C (в последних версиях Фортрана также может использоваться звездочка "*" и восклицательный знак "!").

В программе не должно быть операторов, имеющих одинаковые метки. Нумерация отдельных операторов может иметь любой порядок, однако упорядочение меток по возрастанию помогает ориентироваться в большой программе.

Оператор, следующий за оператором GO TO, всегда должен иметь метку, в противном случае к нему не будет доступа в программе.

Если оператор помечен, а управление ему не передается, то это не является ошибкой программы, хотя компилятор может выдать соответствующее предупреждение.

7.5.2. Условный арифметический оператор IF

Арифметический IF позволяет осуществить разветвление вычислительного процесса на три ветви в зависимости от значения некоторого арифметического выражения AW, входящего в состав этого оператора. Он имеет вид:

```
MT IF(AW) M1, M2, M3
```

где MT — либо пусто, либо метка данного оператора IF; AW — любое арифметическое выражение; M1, M2, M3 — метки операторов, которым передается управление.

Действие условного арифметического оператора заключается в следующем: если значение $AW < 0$, то управление передается оператору с меткой M1; если $AW = 0$, то — оператору с меткой M2; если значение $AW > 0$, то следующим выполняемым оператором будет оператор с меткой M3. Например, оператор, использующий выражение целого типа:

```
IF(K-1) 5, 10, 15
```

при $K < 1$ передаст управление оператору с меткой 5, при $K = 1$ — с меткой 10, и при $K > 1$ — оператору с меткой 15.

Если разветвление производится на две ветви, то две из трех меток могут совпадать. Например, оператор, использующий выражение вещественного типа:

```
IF(Z-0.5) 8, 12, 12
```

при $Z < 0.5$ передаст управление оператору с меткой 8, а при $Z = 0.5$ и при $Z > 0.5$ — оператору с меткой 12.

При использовании оператора IF нужно учитывать, что значение арифметического выражения вещественного типа может отличаться от предполагаемого точного значения, например нулевого, из-за приближенного представления вещественных величин в памяти ЭВМ.

Отметим, что оператор, следующий за арифметическим оператором IF, всегда должен иметь метку, в противном случае к нему не будет доступа в программе.

7.5.3. Условный логический оператор IF

Для разветвления вычислительных процессов очень часто предпочтительнее использование условного логического оператора IF, когда разветвление выполняется при помощи логических выражений.

Простейшим логическим выражением (LW) является отношение. Отношение есть предположение о двух арифметических выражениях, которое может быть либо истинным (т.е. удовлетворяться для входящих в него величин), либо ложным (т.е. не удовлетворяться). В зависимости от этого вычислительный процесс может быть направлен либо по одной, либо по другой ветви. В первом случае говорят, что отношение имеет значение истина (.TRUE.), во втором — ложь (.FALSE.).

Отношение состоит из двух арифметических выражений, соединенных знаком операции отношения, обозначаемых на Фортране: .LT. (меньше), .LE. (меньше или равно), .GT. (больше), .GE. (больше или равно), .EQ. (равно), .NE. (не равно), причем присутствие точек обязательно.

Выражение отношений имеет такой вид:

$$AW1 . ZO . AW2$$

где AW1 и AW2 — любые арифметические выражения целого или вещественного типа, ZO — знак операции отношения (выделяется точками). Например: (B*B-4.*A*C).GE.0., (A+B).NE.(C+D), I.GT.5, SIN(X).LT.COS(X).

Условный логический оператор IF имеет вид:

$$MT \ IF \ (LW) \ S$$

где MT — либо пусто, либо метка, помечающая данный оператор IF; LW — любое логическое выражение; S — любой выполняемый оператор, кроме DO (см. п. 7.5.4) и IF.

При его выполнении анализируется логическое выражение LW, заключенное в скобки. Управление передается оператору S, если значение этого выражения истинно. В противном случае он пропускается и выполняется оператор, следующий в программе за логическим IF. Например:

```
IF (NW.EQ.10) X=SIN(T/4.)+0.5*T**3
IF ((B*B-4.*A*C).GE.0.) GO TO 45
```

Чаще других в логическом условном операторе используется оператор безусловного перехода GO TO МК. Он позволяет передать управление при истинности логического выражения на ту ветвь вычислительного процесса, которая начинается оператором с меткой МК. Вторая ветвь начинается оператором, следующим за условным логическим оператором и выполняется, если значение логического выражения ложно.

Более сложные логические выражения, входящие в оператор IF, могут быть построены с помощью операций над выражениями отношения: .NOT. — отрицание (нет); .AND. — логическое умножение (и); .OR. — логическое сложение (или), причем присутствие точек обязательно.

Операция отрицания применяется к одному операнду и меняет его значение на противоположное. Например, если AW имеет значение истинно, то .NOT. AW ложно и наоборот.

Операции .AND. и .OR. применяются к двум операндам, т.е. к двум выражениям отношения. Их результат выполнения имеет значение истинно, когда оба операнда (для .AND.) или хотя бы один из них (для .OR.) имеют значение истинно.

Заметим, что не следует без необходимости использовать сложных операторов IF, как и любых других. Их лучше представить в виде нескольких более простых выражений.

7.5.4. Оператор цикла DO

Наиболее сложным и мощным из числа операторов управления является оператор цикла DO, позволяющий несколько раз повторить нужную последовательность операторов при различных значениях некоторой переменной, называемой счетчиком цикла. Оператор цикла DO может быть записан в одной из двух форм:

```
MT1 DO МК I=N1, N2, N3
```

или

```
MT2 DO МК I=N1, N2
```

где MT1, MT2 — либо пусто, либо метка, помечающая данный оператор цикла DO;

МК — метка последнего оператора, принадлежащего повторяемой группе операторов и называемого конечным оператором цикла;

I — неиндексированная целая переменная, соответствующая последовательно изменяющейся в цикле величине и называемая параметром или счетчиком цикла;

$N1, N2, N3$ — положительные константы или неиндексированные переменные целого типа (выражения в Фортране-66 запрещены), причем $N1$ и $N2$ — соответственно начальное и конечное значения параметра I , а $N3$ — шаг его изменения при каждом прохождении цикла.

Конструкцию $I=N1, N2, N3$ принято называть заголовком цикла. Если $N3=1$, то в этом случае $N3$ можно не указывать и заголовок цикла будет иметь вид: $I=N1, N2$.

Группа операторов, начиная с оператора DO до конечного оператора цикла с меткой МК, составляет цикл DO. Операторы данного цикла выполняются сначала при значении целой переменной $I=N1$, затем при $I=N1+N3$, затем при $I=N1+2*N3, I=N1+3*N3$ и т.д., пока значение I меньше или равно $N2$. Как только станет значение параметра цикла $I=N2+N3$, то выполнение цикла прекращается и программа переходит к выполнению оператора, следующего непосредственно за конечным оператором цикла с меткой МК. Это называется нормальным выходом из цикла.

Например, в результате выполнения операторов цикла:

```
DO 16 I=1, N
16 B(I)=0.                                     (7.4)
```

будут заданы нулевые значения всем N элементам вектора-столбца B . Отметим, что значение переменной N , присутствующей в заголовке цикла, обязательно должно быть предварительно задано в программе (например, оператором присваивания $N=9$).

Необходимое копирование вектора-столбца $B(N)$ в резервный массив $BD(N)$ может быть выполнено с помощью следующего цикла:

```
DO 50 I=1, N
50 BD(I)=B(I)                                  (7.5)
```

который с помощью арифметического оператора присваивания для всех $N=9$ элементов векторов-столбцов присвоит значение каждого элемента $B(I)$ соответствующему элементу $BD(I)$: $BD(1)=B(1), BD(2)=B(2), \dots, BD(9)=B(9)$.

Отметим, что рекомендуется выносить из цикла те операторы или группы операторов, значения которых в цикле не изменяются.

7.5.5. Правила использования оператора DO

Правило 1. Первый оператор цикла должен быть выполняемым. Это не может быть оператор типа описания DIMENSION, INTEGER, REAL, DOUBLE PRECISION, FORMAT.

Правило 2. Конечным оператором цикла не могут быть операторы GO TO, IF или другой оператор DO. Это связано с тем, что после выполнения конечного оператора цикла управление должно быть передано на его начало, что противоречило бы действию любого оператора из перечисленных. Это ограничение обычно обходят использованием специального оператора продолжения CONTINUE. Он не выполняет никаких действий, не порождает команд в объектной программе и обычно используется в качестве конечного оператора цикла, даже если его присутствие строго не обязательно. Нижеследующий цикл с использованием оператора CONTINUE эквивалентен примеру (7.5):

```
DO 50 I=1, N
  BD (I) =B (I)
50 CONTINUE
```

Правило 3. Внутри цикла не должно быть операторов, изменяющих значения параметров цикла (I, N1, N2, N3), т.е. значения всех величин из заголовка цикла должны быть полностью определены до входа в цикл или заданы числами в самом заголовке цикла.

Правило 4. Выход из цикла, т.е. передача управления из цикла оператору вне этого цикла, может быть осуществлен в любое время до нормального завершения цикла. В этом случае текущее значение параметра цикла I определено и может быть использовано в дальнейших вычислениях. При нормальном выходе из цикла, когда цикл выполнен полностью, значение параметра I становится неопределенным в Фортране-66 и сохраняет свое максимальное значение в Фортране-77.

Вместе с этим передача управления извне внутрь цикла не разрешается (кроме одного особого случая, который здесь не рассматривается), так как цикл всегда выполняется, начиная с оператора DO. Допускается обращение к подпрограммам (см. п. 7.9) из области действия оператора DO и возврат из них в тот же самый цикл DO с тем условием, чтобы значения величин из заголовка цикла не изменялись: I, N1, N2, N3.

Правило 5. Цикл DO может содержать внутренние циклы. В этом случае их область действия должна полностью находиться в области действия внешнего цикла, т.е. конечный оператор внутреннего цикла должен совпадать с конечным оператором внешнего цикла, либо появляться до него. Такая совокупность циклов называется вложенными циклами. В них быстрее всего изменяются параметры внутреннего цикла, затем внешние, т.е. при начальных внешних параметрах полностью выполняется весь внутренний цикл, затем на шаг изменяется внешний параметр и снова выполняется весь внутренний цикл и т.д.

Примеры. Задание нулевых значений элементам двумерного массива матрицы A(N,N) может быть выполнено следующим вложением двух циклов:

```

DO 16 I=1,N
  DO 16 J=1,N
    A(I,J)=0.
16 CONTINUE

```

(7.6)

Копирование матрицы $A(N,N)$ в резервный массив $AD(N,N)$ (см. программу 8.4) выполняют следующие вложенные циклы:

```

DO 50 I=1,N
  DO 50 J=1,N
    AD(I,J)=A(I,J)
50 CONTINUE

```

(7.7)

Сначала при $I=1$ будет выполнен весь внутренний цикл по J и с помощью оператора присваивания будет заполнена 1-я строка матрицы AD : $AD(1,1)=A(1,1)$, $AD(1,2)=A(1,2)$, ..., $AD(1,9)=A(1,9)$.

Затем параметр внешнего цикла I изменится на шаг и при $I=2$ будет снова выполнен весь внутренний цикл и заполнится вторая строка матрицы AD : $AD(2,1)=A(2,1)$, $AD(2,2)=A(2,2)$, ..., $AD(2,9)=A(2,9)$.

Потом при каждом изменении внешнего параметра I на шаг ($I=3,4,5,6,7,8$) будет выполняться каждый раз весь внутренний цикл и заполняться соответственно 3-я, 4-я, 5-я, 6-я, 7-я и 8-я строки матрицы AD . Наконец, при $I=9$ (т.к. $N=9$) в последний раз выполнится внутренний цикл и заполнится 9-я строка матрицы AD : $AD(9,1)=A(9,1)$, $AD(9,2)=A(9,2)$, ..., $AD(9,9)=A(9,9)$.

После этого управление будет передано следующему за циклом оператору программы, т.е. станет выполняться оператор, следующий в программе за конечным оператором внутреннего и внешнего цикла `CONTINUE` с меткой 50.

Однотипные операции, выполняемые циклами (7.4) и (7.6) — обнуление, и (7.5) и (7.7) — копирование, могут быть совмещены соответственно в один цикл каждый. Например, для одновременного копирования одномерных и двумерных массивов это будет иметь вид:

```

DO 50 I=1,N
  BD(I)=B(I)
  DO 50 J=1,N
    AD(I,J)=A(I,J)
50 CONTINUE

```

(7.8)

Отличие работы вложенных циклов (7.8) от (7.7) состоит в том, что при каждом значении I будет сначала заполняться соответствующий элемент $BD(I)=B(I)$, а затем при этом же значении I будет выполняться весь внутренний цикл по J .

7.6. Операторы ввода-вывода

7.6.1. Общий вид операторов ввода-вывода

Если предстоит произвести вычисления только при одном наборе данных, то все исходные величины можно ввести в программу с помощью арифметических операторов присваивания (см. п. 7.4), операторов DATA (см. п. 7.7.1), REAL и INTEGER (см. п. 7.7.2). Для возможности использования различных исходных данных их обычно помещают в отдельный файл. Затем они вводятся в память ЭВМ во время выполнения программы с помощью оператора ввода READ. Для вывода данных используются операторы WRITE или PRINT. Общий вид операторов READ и WRITE следующий:

READ (N, M) список (7.9)

WRITE (N, M) список (7.10)

где N — условный номер устройства соответственно ввода или вывода данных — это целое число (обычно от 1 до 15).

Дополнительные операторы READ и PRINT предназначены для ввода и вывода данных на обычно используемые системные устройства. Оператор READ используется для чтения данных с системного устройства ввода, оператор PRINT — для вывода данных на системное устройство печати. Их общая форма записи имеет вид:

READ M, список (7.11)

PRINT M, список (7.12)

где для всех операторов (7.9)–(7.12) M — метка оператора FORMAT, который управляет выполнением соответствующего оператора READ, WRITE или PRINT (см. п. 7.6.6);

список — список ввода-вывода, состоящий из имен вводимых или выводимых величин, разделенных запятыми. В списке могут использоваться имена переменных (простых или с индексами), имена массивов или форма неявного цикла DO (см. п. 7.8).

Для каждой переменной, каждого элемента массива, указанных в списке, между памятью и записью передается одно данное, причем передача значений идет последовательно слева направо. Это означает, что при вводе с использованием оператора READ значения, присваиваемые переменным или элементам массива списка ввода, должны быть расположены в строке бланка программы или экрана дисплея в том же порядке, в каком они указаны в списке ввода оператора READ. Оператор READ всегда означает чтение с новой строки.

Аналогичным образом действуют операторы WRITE и PRINT: переменные, значения которых нужно напечатать, перечисляются в том порядке, в котором они должны быть расположены в строчке — слева направо. Операторы WRITE и PRINT также всегда означают печатание с новой строки.

Если в списке используется имя массива, то массив передается целиком в том порядке, в котором он размещен в памяти. Одномерные массивы размещаются в памяти естественным образом, т.е. элементы следуют друг за другом в порядке возрастания индексов. Двумерные массивы, приготовленные для ввода в естественной форме по строкам, как бы “переворачиваются” и располагаются в памяти по столбцам (см. п. 7.2).

Это (“переворачивание”) следует иметь в виду при вводе или выводе двумерных массивов. Поэтому для расположения матриц в памяти ЭВМ в естественной форме по строкам, что нужно для удобной и правильной работы с ними, их предварительно располагают по столбцам (как бы “переворачивают”). Тогда повторное “переворачивание” при вводе расположит их в памяти ЭВМ в естественной форме. Таким образом, при использовании оператора:

```
READ 10, A
```

где A — имя двумерного массива, элементы этого массива должны быть расположены в строке бланка программы или экрана дисплея по столбцам. Аналогичный процесс происходит и при выводе матриц. Поэтому в той же последовательности (по столбцам) будет осуществляться вывод этой же матрицы A при использовании оператора:

```
PRINT 20, A
```

Расположение элементов матрицы по столбцам при вводе или выводе является неудобным, ибо нарушает привычную форму расположения матрицы по строкам. В Фортране имеется возможность при работе с массивами располагать их при вводе или выводе в привычном для нас порядке следования элементов (т.е. по строкам) с помощью формы неявного цикла DO (см. п. 7.8).

7.6.2. Ввод — вывод, управляемый списком

Операторы ввода-вывода, управляемого списком, используются для передачи данных между файлом и элементами, указанными в списке ввода-вывода, причем преобразование данных выполняется в соответствии с типами элементов, указанных в списке. Общая форма операторов:

```
READ (N, *) список (7.13)
```

```
WRITE (N, *) список (7.14)
```

где N — номер устройства, задаваемый в виде целой константы без знака (для ПЭВМ в параметре unit оператора open — см. п. 7.6.3);

список — список ввода-вывода.

Для ввода и вывода данных на обычно используемые стандартные устройства используются дополнительные операторы READ и PRINT соответственно, имеющие более простую форму:

```
READ *, список (7.15)
```

```
PRINT *, список (7.16)
```

Мы будем во всех примерах пользоваться простыми операторами ввода (7.15) и вывода (7.12), работа с которыми возможна при выполнении программ на Фортране на любых IBM-совместимых ЭВМ (см. п. 7.6.3).

7.6.3. Ввод-вывод данных на ПЭВМ

Стандартным устройством, принимаемым по умолчанию, на ЭВМ ЕС 1035-1066 является магнитный диск, откуда по имени файла с типом DATA читаются исходные данные или куда (с типом LIST) записываются результаты расчета.

Для ПЭВМ стандартным устройством, принимаемым по умолчанию, при вводе является клавиатура, а при выводе экран. Останавливать каждый раз работу программы и вводить исходные данные с клавиатуры не всегда удобно. Возможны два способа для реализации чтения-записи данных с диска в процессе работы программы на ПЭВМ. Они используются для установления связи между устройством и так называемым внешним файлом, в котором хранятся предварительно подготовленные исходные данные или куда будут записываться результаты.

1. Явное присоединение файла к устройству (имя файла указывается в параметре FILE оператора OPEN). Связь между устройством и внешним файлом должна быть установлена до обращения к нему по операторам ввода-вывода, например:

```
open (unit=5, file='prs2.dat')
open (unit=6, file='prs2.lis')
read (5,*) n,ksu
write (6,*) k,cond
```

Внешние файлы с именами prs2.dat и prs2.lis присоединяются соответственно к устройствам 5 и 6. Из первого из них выполняется операция чтения данных для переменных N и KSU, а во второй — печать значений переменных K и COND. Файлы остаются присоединенными до тех пор, пока оператор CLOSE не отменит это присоединение или пока не завершится программа. В рассмотренном случае используются операторы ввода-вывода в своем

общем виде (7.13)–(7.14). Заметим, что операторы `open` могут быть также представлены в более простом виде:

```
open (5, file='prs2.dat')
open (6, file='prs2.lis')
```

В вышеприведенном фрагменте фортран-программы маленькие буквы использованы для большего подчеркивания его предназначенности только для ПЭВМ, в отличие от всех остальных программ и фрагментов пособия, одинаково пригодных как для персональных, так и для больших ЭВМ.

2. В команде вызова программы для выполнения используется перенаправление потоков ввода-вывода (см. п. 2.2.3). Это возможно в том случае, если в программе оператор `open` с указанием имени файла не применяется, а используются простые операторы ввода-вывода (7.15), (7.12) или (7.16). Тогда вышеприведенный фрагмент программы примет обычно используемый в пособии вид, одинаково пригодный и для больших машин (ЕС ЭВМ 1035-1066) и для ПЭВМ:

```
READ *, N, KSU
PRINT *, K, COND
```

В этом случае для ПЭВМ при запуске программы `prs2.exe` на выполнение, следует указать соответствующие имена файлов, из которых следует брать исходные данные (`prs2.dat`) и куда помещать результаты расчета (`prs2.lis`):

```
prs2.exe < prs2.dat > prs2.lis
```

Подчеркнем, что для одной и той же программы при очередном ее запуске имена файлов данных и результатов могут быть каждый раз разными, что создает большое удобство при организации работы группы пользователей.

7.6.4. Требования к данным ввода, управляемого списком

Записи, содержащие входные данные для оператора ввода, управляемого списком, komponуются из констант, отделяемых друг от друга разделителями, в качестве которых могут быть использованы запятая или пробел. Например, если ввод данных осуществляется по оператору:

```
READ *, KNNE, N, OD, ZN, ZD
```

то ему соответствует строка данных, например, в форме:

```
8, 3, -1., 4.5, 2.
```

Тип константы должен соответствовать типу элемента в списке ввода-вывода. Он определяется по типу, количеству и порядку следования элементов в списке, например `KNNE` и `N` — целые, а `OD`, `ZN` и `ZD` вещественные (при неявном описании типа — см. п. 7.1). Поэтому после

вещественных значений констант без дробной части точку ставить необязательно, например:

8, 3, -1, 4.5, 2

Двумя последовательными запятыми представляется так называемое отсутствующее данное, например:

8, , -1, 4.5, 2.

В этом случае соответствующий элемент списка ввода (в данном случае 2-й) пропускается, его текущее значение в памяти не изменяется.

Если нужно пропустить 1-й элемент списка ввода, то на его месте в списке ставится запятая, например:

, 3, -1, 4.5, 2.

Количество данных в записи ввода может быть меньше количества элементов в списке. В этом случае за последним данным в записи обязательно должен следовать разделитель “/”:

8, 3, -1/

Для констант и отсутствующих данных может быть указан коэффициент повторения J^* , где J — целая положительная константа, указывающая, сколько раз должна повторяться константа или отсутствующее данное. Например, при вводе по оператору:

READ *, A1

матрицы A1(9,9) примера (6.5), описанной оператором DIMENSION в виде одномерного массива A1(81), вводимые элементы должны быть расположены по столбцам, например, с использованием коэффициента повторения J^* :

2, 0, 1, 3*0, 6, 0, -1
 -3, 1, 7*0
 7, 0, 1, 6*0
 7*0, 1, 0
 8*0, 1
 4*0, 1, 2*0, -1, 0
 5*0, 1, -2, 0, -1
 0, 1, 0, 1, -1, 4*0
 2*0, 1, 10, 0, -1, 3*0

(7.17)

7.6.5. Требования к данным вывода, управляемого списком

Данные в записях, создаваемых по оператору вывода, управляемого списком, выводятся в такой форме, которую требует оператор вывода, управляемо-

го списком. Однако с помощью оператора вывода, управляемого списком, не могут быть выведены отсутствующие данные, коэффициент повторения J* и разделитель /. Например, матрица A1 (7.17) будет выведена по оператору:

```
PRINT *, A1
```

 (7.18)

следующим образом:

```

2.0      .0      1.0      .0      .0      .0      6.0
 .0     -1.0     -3.0      1.0      .0      .0      .0
 .0      .0      .0      .0      7.0      .0      1.0
 .0      .0      .0      .0      .0      .0      .0
 .0      .0      .0      .0      .0      .0      1.0
 .0      .0      .0      .0      .0      .0      .0
 .0      .0      1.0      .0      .0      .0      .0
1.0      .0      .0     -1.0      .0      .0      .0
 .0      .0      .0      1.0     -2.0      .0     -1.0
 .0      1.0      .0      1.0     -1.0      .0      .0
 .0      .0      .0      .0      1.0     10.0      .0
-1.0      .0      .0      .0

```

 (7.19)

Матрица A1 при выводе по оператору (7.18) располагается в полном виде по столбцам в виде одномерного массива из 81-го элемента в соответствие с ее описанием в операторе DIMENSION A1(81) (проверьте это сравнением с естественным видом матрицы A1 (6.5)). Однако ориентироваться в распечатке (7.19) намного труднее, ибо в строку при выбранном машинном формате печати помещается всего 7 элементов, что нарушает привычный порядок расположения матрицы A1(9,9).

Операторы ввода-вывода (7.13)–(7.16), управляемого списком, имеют простую форму. Очень удобно пользоваться операторами (7.13) или (7.15) для ввода данных, что мы рекомендуем делать всегда и чем будем пользоваться во всех примерах.

Использование операторов (7.14) и (7.16) для вывода не совсем удобно (см. распечатку (7.19) матрицы A1) и рекомендуется делать только в начале изучения материала. Это позволит не рассматривать довольно сложный оператор FORMAT (который при первом изучении можно пропустить), и не испытывать затруднений с выводом данных (см. программу 8.1). В дальнейшем, после накопления опыта и появления необходимости в представлении результатов расчета в более удобной и красивой форме с соответствующими пояснениями, раздел 7.6.6 следует изучить и пользоваться операторами вывода в форме (7.10) или (7.12).

7.6.6. Оператор FORMAT

Оператор FORMAT является неисполнимым (или невыполняемым), так как он не порождает команд при выполнении программы. Под управлением оператора FORMAT в Фортране выполняется ввод информации по операторам (7.9) или (7.11) и вывод информации по операторам (7.10) или (7.12). В простейшем случае оператор FORMAT записывается в виде:

```
MT FORMAT (C1, C2 . . . , CI, . . . , CN)
```

где MT — метка (номер) оператора, а каждое CI обозначает спецификатор (описатель) поля, имеющий любой из нижеследующих видов: — для числовых данных: IW, FW.K, EW.K, DW.K, GW.L; — для буквенно-цифровых данных: WH, WX, TW, ' ' .

Здесь символы I, F, E, D, G, H, X, T и ' ' означают тип (код) спецификации; буквы W, K, L — это целые числа, значения которых указывают: W — общее количество позиций, занимаемых вводимым или выводимым данным; K — количество позиций, отводимых под дробную часть числа (т.е. количество цифр после десятичной точки); L — число значащих цифр (только в спецификации типа G).

Оператор FORMAT при вводе указывает, из каких позиций (колонок) должны считываться данные для переменных, перечисленных в списке ввода в операторах (7.9) или (7.11) и в каком формате (формат — форма представления числа) должен осуществляться ввод.

При выводе по операторам (7.10) или (7.12) оператор FORMAT указывает, в какие позиции и в каком формате должен быть осуществлен вывод переменных.

При выполнении операторов (7.9)–(7.12) соответствующий список ввода или вывода просматривается слева направо, в таком же порядке используются спецификации формата для числовых данных. Таким образом с помощью пошагового прохождения спецификации формата и списка ввода-вывода операторов (7.9)–(7.12) происходит соответственно ввод или вывод чисел одного за другим, пока не будет достигнут конец списка.

В простейших случаях число спецификаций формата в операторе FORMAT должно соответствовать числу переменных в списке ввода-вывода операторов (7.9)–(7.12).

Одним из важных понятий ввода-вывода является запись. Обычно это строка символов. Максимальный размер записи при вводе — 80 символов, при выводе — обычно 120 (включая пробелы).

7.6.7. Спецификации формата для числовых данных: I, F, E, D, G

Спецификация типа I (IW) используется для ввода-вывода значений целых переменных. Под число отводится W позиций, причем число в поле вывода

всегда располагается справа, а левая часть заполняется пробелами, если поле слишком велико.

Следует иметь в виду, что первый символ записи на печать не выводится (он используется для управления движением бумаги перед печатью) и должен быть пробелом для обычной печати. Например, при выполнении оператора:

```
PRINT 5, NG, NZ, NW
5 FORMAT (I8, I2, I2) (7.20)
```

если $NG=103167$, $NZ=9$, $NW=7$, то на печать выводится следующая строка:

```
103167 9 7
```

Все числа прижаты к правой границе соответствующего поля и для 2-го и 3-го числа из одной цифры по спецификации I2 слева остается один пробел. Под 1-е число из 6-ти цифр по спецификации I8 отведено 8 позиций, 1-й символ является пробелом и на печать не выводится, поэтому слева от начала текста на распечатке остается также только один пробел.

Для повторения спецификаций несколько раз достаточно указать необходимый коэффициент кратности перед буквой типа спецификации, что для примера (7.20) будет иметь вид:

```
5 FORMAT (I8, 2I2)
```

Спецификация типа F (FW.K) используется для ввода-вывода значений вещественных переменных обычной и двойной точности в виде целой и дробной частей, разделенных точкой (в так называемой основной форме или в форме F). Например: F6.2, F8.3, F10.4, F15.8.

Преимущество спецификатора типа F заключается в том, что он дает выходные данные в естественной форме, удобной для чтения и требует меньшего числа позиций (по сравнению с E и G).

Его недостатком является обязательность знания программистом максимальной величины чисел, которые встретятся, ибо если целая часть числа не поместится в отведенных $W-K-1$ позициях, то все соответствующее поле заполняется звездочками (*). Это относится и к спецификации типа I.

Спецификации типа E (EW.K) и типа D (DW.K) используются для ввода-вывода вещественных чисел в форме с порядком: форме E (для обычной точности) и форме D (для двойной точности). Необходимо выполнение соотношения $W > K+6$. Порядок числа при выводе всегда занимает 4 символа (E или D, пробел или минус и две десятичные цифры экспоненты), кроме того обязательные позиции выделяются для размещения нуля, точки и знака числа. Например: E11.4, E12.5, E10.3, D15.8.

Вывод чисел в формах E или D осуществляется в нормализованной форме, т.е. константа перед порядком имеет нуль целых и первая цифра после точ-

ки — значащая. Данные располагаются в поле справа и левые свободные позиции при $W > K + 7$ заполняются пробелами.

Спецификация типа G (GW.L) является универсальной. Она может задаваться для целых и вещественных чисел (обычной и двойной точности) вместо спецификаций типа I, F, E и D. Для нее также необходимо выполнение требования, аналогичного для спецификаций E и D: $W > L + 6$. Например: G11.4, G12.5.

Спецификация типа G является наиболее удобной для вывода вещественных чисел, а наиболее удобным для ввода чисел является ввод, управляемый списком (см. п. 7.6.2). Они будут наиболее часто использоваться во всех примерах.

7.6.8. Спецификации формата для буквенно-цифровых (литеральных) данных (H, ' ', X, T)

Они используются для выдачи на печать заголовков, пояснений и другой текстовой информации. Спецификация типа H (холлеритовского типа) имеет вид: WH, где W — число позиций, следующих сразу за форматным кодом H, в которых размещаются любые символы. Так, например, оператор FORMAT (с меткой 5) фрагмента (7.20) может быть дополнен текстовой информацией:

```
PRINT 5, NG, NZ, NW
5 FORMAT (7H ГРУППА, I7,                                (7.21)
*40H ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ ПО СТАТИКЕ НОМЕР, I2,
*8H ВАРИАНТ, I2)
```

Литеральным данным является все, что расположено в 7-ми, 40-ка и 8-ми позициях за символом H. Под литеральные данные дополнительная память в машине не отводится и никакие переменные в списке ввода-вывода им не соответствуют. На печать в рассматриваемом примере (7.21), начиная с первой позиции, будет выведено (для данных примера (7.20)):

```
ГРУППА 103167 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ ПО СТАТИКЕ НОМЕР 9 ВАРИАНТ 7
```

Первый пробел в литерале используется в качестве управляющего символа (он на печать не выводится), остальные пробелы служат разделителями литеральных данных и выводимых переменных (причем места для всех пробелов включены в соответствующие числа позиций в спецификации типа H).

Для литеральных данных в операторе FORMAT вместо спецификации типа H могут быть использованы апострофы ' ' (иногда называемые спецификацией типа литерал), между которыми заключаются все передаваемые символы в необходимом порядке. В этом случае не требуется подсчитывать

занимаемое число позиций, что значительно удобнее. Оператор FORMAT фрагмента (7.22) эквивалентен соответствующему оператору примера (7.21):

```
PRINT 5, NG, NZ, NW
5 FORMAT (' ГРУППА' , I7,
* ' ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ ПО СТАТИКЕ НОМЕР' , I2,
* ' ВАРИАНТ' , I2)
```

(7.22)

Косая черта / в операторе FORMAT в качестве разделителя спецификаций вместо запятой всегда означает переход к следующей записи (см. (7.23)). Можно написать несколько косых черт подряд. При выводе в результате этого будет выдано несколько записей, состоящих из пробелов (пустых строк для разделения записей друг от друга), а при вводе это вызовет пропуск соответствующего количества записей.

Спецификация типа X (WX) употребляется для разделения выводимых данных некоторым количеством пробелов, задаваемым числом W перед X. С его помощью часто задают пробел (1X) в качестве управляющего символа печати в выводимой строке. Пример (7.22) с использованием спецификации типа X и с печатью текста в две строки примет вид:

```
PRINT 5, NG, NZ, NW
5 FORMAT (1X, ' ГРУППА' , I7, 1X, ' ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ ' ,
* ' ПО СТАТИКЕ НОМЕР' , I2/23X, ' ВАРИАНТ' , I2)
```

(7.23)

Во 2-й строке печати, отступив от левого края 23 пробела, первый из которых выполняет роль управляющего символа для устройства печати, будет напечатано “ВАРИАНТ 7” (оставшийся в 1-й строке текст сохраняется без изменения):

```
ГРУППА 103167 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ ПО СТАТИКЕ НОМЕР 9
ВАРИАНТ 7
```

Спецификация типа T имеет вид TW и позволяет осуществлять вывод (или ввод) данных по следующей за ней через запятую спецификации, начиная с любой указанной в числе W позиции записи. При использовании спецификации типа T следует помнить, что отсчет позиций в записи начинается с управляющего символа, поэтому первая запись может быть не ранее 2-й позиции: T2.

Обычно оператор FORMAT описывает структуру вводимой или выводимой записи, которая начинается с первой слева позиции в записи и последовательно заполняется вправо. Только спецификация типа T позволяет не учитывать последовательность заполнения записи, подсчитывая использованное количество позиций, а начинать ее в любом нужном месте. Пример (7.23) с использованием спецификации типа T может иметь даже следующий вид, при этом выводимая печать останется без изменений:

```
PRINT 5, NG, NZ, NW
5 FORMAT (T9, I6, T16, ' ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ ПО СТАТИКЕ ' ,
* ' НОМЕР' , I2, T2, ' ГРУППА' / T24, ' ВАРИАНТ' , I2)
```


Последовательность расположения записей на практике обычно не нарушается (здесь это сделано только для показа возможностей спецификации). При выводе заголовков и результатов данных в виде таблиц особенно удобно использовать спецификацию типа T, позволяющую сразу указать нужное место записи.

7.6.9. Повторители и группы спецификаций

Если в списке спецификаций формата две или несколько спецификаций подряд совпадают, то эту спецификацию можно написать лишь один раз, указав требуемое число ее повторений. Поэтому в примере (7.20) оператор FORMAT может быть записан также в виде:

```
5 FORMAT (I7, 2I2)
```

Если нужно повторить группу спецификаций формата, то она заключается в скобки, а перед скобкой ставится целое число повторений группы, например:

```
60 FORMAT (I3, 8 (I2, I2, F7.4))
```

или

```
60 FORMAT (I3, 8 (2I2, F7.4))
```

Отметим, что группы спецификаций играют важную роль при выводе (вводе) длинных списков, когда они просматриваются повторно. При этом повторный просмотр формата начинается каждый раз с последней группы спецификаций, хотя бы повторитель ее был равен единице и не указывался. Например, в результате работы фрагмента программы:

```
PRINT 12, KSU, A, B                                (7.24)
12 FORMAT (/5X, 'СИСТЕМА УРАВНЕНИЙ РАВНОВЕСИЯ НОМЕР', I2/
* (1X, 9F8.3))
```

для матрицы A из 81-го элемента и вектора-столбца B из 9-ти элементов при KSU=1 во 2-й строке с 6-й позиции (1-я строка пустая) на печать сначала будет выведено:

```
СИСТЕМА УРАВНЕНИЙ РАВНОВЕСИЯ НОМЕР 1
```

Затем в 3-й строке будет напечатан 1-й столбец матрицы A, начиная со 2-й позиции: 9 элементов в одну строку. Так как список вывода еще не закончился, то повторный просмотр формата, начинающийся каждый раз с последней группы спецификаций (1X, 9F8.3), выведет в 4-й-11-й строках печати столбцы матрицы A со 2-го по 9-й соответственно (также по 9 элементов в строку). В 12-й строке при последнем просмотре спецификаций формата будет напечатан вектор-столбец B. После этого список вывода будет исчерпан. Повторный просмотр формата прекратится и начнет выполняться следующий оператор программы.

Отметим, что использование групп спецификаций при выводе (вводе) длинных списков позволяет при повторном просмотре игнорировать специфика-

ции форматов, расположенные перед последней группой. Таким образом, оператор FORMAT в примере (7.24) по результатам работы эквивалентен нижеследующему:

```
5 FORMAT (/5X, 'СИСТЕМА УРАВНЕНИЙ РАВНОВЕСИЯ НОМЕР', I2/  
*10 (1X, 9F8.3))
```

7.7. Задание начальных значений

Переменным, элементам массива или массивам нужны начальные значения задаются непосредственно в момент загрузки программы. Это можно сделать двумя способами: с помощью оператора DATA (см. п. 7.7.1) или с помощью операторов явного описания типа REAL и INTEGER (см. п. 7.7.2).

7.7.1. Оператор DATA

Оператор DATA используется для задания начальных значений переменным, элементам массива или массивам и имеет вид:

```
DATA SP1/SK1/, SP2/SK2/, . . . , SPI/SKI/, . . . , SPN/SKN/
```

где SPI — списки имен переменных, элементов массива или массивов; SKI — списки соответствующих значений констант. Все элементы в списке разделяются запятыми.

Между элементами списка имен и списка значений констант должно существовать взаимно-однозначное соответствие: общее количество указанных в каждом списке SPI переменных должно быть согласовано с общим числом заданных числовых значений в соответствующем списке констант SKI, причем их типы также должны совпадать. Этим соответствием устанавливаются начальные значения. Поэтому операторы (7.25) и (7.26) эквивалентны:

```
DATA NG, NZ, NDIM, R, NW/103167, 9, 9, 10., 7/ (7.25)
```

```
DATA NG/103167/, NZ/9/, NDIM/9/, R/10./, NW/7/ (7.26)
```

Если константа повторяется подряд несколько раз, то ее можно записать один раз с повторителем, который отделяется от константы символом * (повторитель не может быть равен нулю):

```
DATA NG/103167/, NZ, NDIM/2*9/, R/10./, NW/7/ (7.27)
```

Если в списке символических имен наряду с переменными встречаются элементы массивов, то каждому элементу массива также присваивается соответственно одно значение:

```
DATA NG/103167/, NZ, NDIM/2*9/, R/10./, NW/7/, (7.28)  
* A1 (1, 1) /2./, A1 (1, 2) /-3./, A1 (1, 3) /7./
```

Если в списке символических имен стоит имя (идентификатор) массива, то в списке значений констант ему должно соответствовать число констант, равное числу элементов массива. Информация о размерности не включается в оператор DATA, поэтому используемые массивы должны быть описаны отдельно с помощью оператора DIMENSION. Например:

```
DIMENSION A1 (9, 9) , B1 (9) , IPVT (9) (7.29)
DATA NG/103167/, NZ, NDIM/2*9/, R/10./, A1/81*0./, B1/9*0./
```

В результате будут присвоены начальные значения: целым переменным NG=103167, NZ=9, NDIM=9, вещественной R=10., всем элементам вещественных массивов матрицы A1 (81 элемент) и вектора-столбца B1 (9 элементов) присвоены значения 0.0, а целочисленный массив IPVT только описывается без присвоения его элементам начальных значений.

Оператор DATA является невыполняемым оператором. В программе может быть любое нужное количество операторов DATA. Значения переменных, определенные оператором DATA, могут быть переопределены, но не с помощью оператора DATA. Это можно сделать, например, с помощью арифметического оператора присваивания (п. 7.4) или оператора ввода READ (п. 7.6.2).

Переменные в операторе DATA получают начальные значения перед выполнением программы в момент ее загрузки независимо от того, в каком месте программы записан оператор DATA. Поэтому он может находиться в любом месте программы после операторов явного описания типа (INTEGER и REAL) и описания массивов (DIMENSION): обычно или сразу за ними или перед оператором, впервые использующим указанную в DATA переменную.

7.7.2. Операторы явного описания типа

Для задания начальных значений могут также использоваться и операторы явного описания типа INTEGER и REAL (при первом чтении можно сразу перейти к п. 7.9). К ним применимы все возможности задания начальных значений переменным, элементам массива и массивам, описанные для оператора DATA. Для этого нужно только вместо оператора DATA использовать операторы INTEGER и REAL для списков из соответствующих однотипных элементов. Поэтому пример (7.28) примет вид:

```
INTEGER NG/103167/, NZ, NDIM/2*9/, NW/7/ (7.30)
REAL R/10./, A1 (1, 1) /2./, A1 (1, 2) /-3./, A1 (1, 3) /7./
```

Кроме этого операторы явного описания типа позволяют совместить задание начальных значений элементам массива с самим описанием массива (не используя оператор DIMENSION). Поэтому в примере (7.29) можно или сохранить оператор DIMENSION и просто разбить оператор DATA на два оператора INTEGER и REAL с соответствующими списками (ана-

логично примеру (7.30)), или включить описание массивов в операторы явного описания типа, исключив оператор DIMENSION (нижеследующий фрагмент эквивалентен примеру (7.29)):

```
INTEGER NG/103167/, NZ, NDIM/2*9/, IPVT(9)
REAL R/10./, A1(9,9)/81*0./, B1(9)/9*0./
```

Из приведенного фрагмента также видно, что в списке элементов оператора явного описания типа, которым присваиваются начальные значения, могут также присутствовать любые элементы, которым не присваивается начальных значений (например, 9 элементов массива IPVT).

При использовании операторов явного описания типа всегда в случае необходимости сохраняется возможность задания нестандартной длины (удвоенной точности):

```
REAL *8 R/10.D0/, A1(9,9)/81*0.D0/, B1(9)/9*0.D0/
```

Задание переменным и элементам массива требуемых значений можно также осуществить с помощью арифметического оператора присваивания или оператора ввода READ (см. пп. 7.4 или 7.6.2).

7.8. Ввод-вывод массивов. Форма неявного цикла DO

7.8.1. Ввод-вывод одномерных массивов

Для одномерных массивов операторы (7.31)–(7.33) эквивалентны (N=9):

```
READ *, B1 (7.31)
```

```
READ *, B1(1), B1(2), B1(3), B1(4), B1(5), B1(6),
*      B1(7), B1(8), B1(9) (7.32)
```

```
READ *, (B1(I), I=1, N) (7.33)
```

В операторе (7.31) указан только идентификатор массива (индекс не определен), поэтому предполагается, что должен быть прочтен весь массив и ввод будет продолжаться, пока не будут введены все 9 элементов. По оператору (7.31) нельзя ввести часть массива или нужные его элементы. Так как используется ввод, управляемый списком, то вводимые элементы могут располагаться (по любому из операторов (7.31)–(7.33)) как в одной строке через запятую, так и в нескольких строках. При использовании ввода по оператору FORMAT расположение вводимых элементов однозначно и строго определяется самим оператором FORMAT, что менее удобно, сложнее и нами не рассматривается.

В операторе (7.32) в списке ввода перечислены все 9 элементов массива, которые должны быть введены в память ЭВМ. Такая форма записи немного гибче. Она позволяет ввести часть массива или нужные его элементы выборочно, но для этого каждый раз приходится переписывать оператор. Поэтому использование оператора (7.32) неудобно, а его запись очень громоздка.

Оператор (7.33) представляет собой сокращенную форму записи оператора (7.32). В нем используется так называемая форма неявного цикла DO (или оператора ввода со встроенным циклом). Он позволяет вводить как весь массив целиком (при N =числу элементов массива, в данном случае $N=9$), так и любую его часть (при других границах 1 и N параметра цикла I , в общем виде $I=N1,N2$). В несколько видоизмененной форме записи (см. оператор (7.35)) его также можно использовать для ввода значений элементов на указанных места в массиве, не меняя каждый раз форму записи оператора.

Форма неявного цикла DO заключается в скобки. Внутри скобок располагаются переменные с индексами или формы неявного цикла DO, разделенные запятыми, если их несколько. За последней переменной записываются индексные параметры (заголовки цикла) $I=N1,N2,N3$, где I -индекс (или параметр цикла), $N1,N2,N3$ — нижняя и верхняя границы и шаг изменения индекса. Они имеют тот же смысл и подчиняются тем же ограничениям, что и для оператора цикла DO. Если $N3=1$, то его можно не писать (см. оператор (7.33) и п. 7.5.4).

Форма неявного цикла DO аналогична обычному оператору цикла DO и для оператора (7.33) имеет вид:

```
DO 7 I=1, N
7 READ *, B1(I) (7.34)
```

В этом частном случае действие обеих групп операторов одинаково: числа последовательно записываются в элементы с $B1(1)$ по $B1(9)$ массива.

Однако в примере (7.34) оператор READ реально встречается каждый раз во время прохождения цикла (9 раз), автоматически выбирая новую строку записи для каждого элемента массива. Поэтому в этом случае необходимо представлять все числа по одному в каждой строке, что громоздко и неудобно для разреженных систем задач статистики.

В операторе (7.33) форма неявного цикла DO является частью оператора READ. В этом случае ввод данных управляется списком и они могут располагаться в одной или нескольких строках. Вводимые данные должны быть представлены в полном виде с указанием всех элементов, причем для одинаковых чисел возможно использование повторителей. Напомним, что с формой оператора (7.33) мы уже встречались под номером (4.3) при подготовке данных для работы программы STAT.

Форма неявного цикла DO позволяет также заполнять массив выборочно в соответствии с нужным номером элемента, задаваемым впереди значения

этого элемента. Такая возможность существует потому, что форма неявного цикла DO может использоваться в операторах ввода-вывода наряду с другими элементами списка. Поэтому по оператору:

```
READ *, KNNEKO, (I, B(I), I1=1, KNNEKO) (7.35)
```

сначала будет прочитано значение KNNEKO, которое является верхней границей изменения параметра цикла I1. Далее начнет выполняться цикл, заключенный в скобки, в котором сначала будет прочитан номер I-го элемента, затем этот указанный I-й номер заполнится следующим за ним в строке данных нужным значением B(I). Затем будет прочтен следующий указанный в строке данных I-й номер, который заполнится следующим за ним в строке данных значением B(I), и т.д. Таких повторений считывания номера элемента и его заполнения указанным за ним значением элемента будет сделано заданное число KNNEKO раз.

Используя предварительное задание всем элементам массива нулевых значений (см. п. 7.7), можно с помощью неявного цикла DO в виде (7.35) значительно упростить ввод данных для разреженных систем (с большим количеством нулевых элементов, какими являются все задачи статики). Напомним, что с формой оператора (7.35) мы уже встречались под номером (4.5) при подготовке данных для работы программы STATN. Например, данные для вектора-столбца B из примера (6.3) для ввода по оператору (7.35) предстанут в виде:

```
6, 1, -20, 4, 30, 6, -3, 7, -49.96, 8, 5, 9, 10.66 (7.36)
```

Аналогично примерам (7.31)–(7.34) может быть организован вывод одномерных массивов. Надеемся, что сомнений в достоинствах формы неявного цикла DO не осталось, поэтому сразу рассмотрим только этот случай (остальные рекомендуем рассмотреть самостоятельно на ЭВМ, для чего оператор READ в примерах (7.31)–(7.34) можно, например, просто заменить на PRINT):

```
PRINT *, (B(I), I=1, N) (7.37)
```

Вывод все же гораздо легче ввода. Поэтому не будем все вопросы представления данных при выводе полностью перелagать на ЭВМ, как в операторе (7.37). Расположим их по порядку в одну строку, сопроводив их заголовком:

```
PRINT 25, (B(I), I=1, N)
25 FORMAT (5X, 'ВЕКТОР ПРАВЫХ ЧАСТЕЙ В' / 2X, 9F8.3)
```

Если вы изучили раздел 7.6.6, то представим вывод с указанием номера выводимого элемента по универсальной спецификации G:

```
PRINT 25, (I, B(I), I=1, N) (7.38)
25 FORMAT (5X, 'ВЕКТОР ПРАВЫХ ЧАСТЕЙ В' /
*5 (2X, 'B(' , I2, ') = ' , G12.5) )
```

В примере (7.38) оператор PRINT под управлением оператора FORMAT сначала в 1-й строке с 6-й позиции напечатает заголовок: “ВЕКТОР ПРАВЫХ ЧАСТЕЙ В”. Затем со 2-й строки начнет выполняться группа спецификаций, заключенная в скобки. Сначала, отступив от левого края два пробела, будет напечатано литеральное данное “В (“. Затем согласно списка вывода оператора PRINT по формату I2 будет напечатан номер 1-го элемента. Потом будет напечатано литеральное данное “)”=“ и после него по спецификации G12.5 будет напечатано значение 1-го элемента. Такая группа спецификаций во 2-й строке через два пробела друг от друга (2X) будет напечатана еще 4 раза для следующих элементов В(2)–В(5), ибо повторитель перед этой группой спецификаций равен пяти. Так как список вывода еще не закончился (N=9), то повторный просмотр формата, начинающийся всегда с последней группы спецификаций, в 3-й строке продолжит вывод для элементов В(6)–В(9). При его 5-м выполнении будет напечатано литеральное данное “В (“ и после него выполнение оператора PRINT закончится, т.к. исчерпан список вывода и значения I=10 в нем нет. В результате на печать для данных вектора-столбца В из примера (6.3) будет выведено:

```

    ВЕКТОР ПРАВЫХ ЧАСТЕЙ В
  В (1)=-20.0 В (2)= .0      В (3)= .0 В (4)=30.0  В (5)= .0
  В (6)=-3.0 В (7)=-49.960 В (8)=5.0 В (9)=10.660 В (

```

В примерах вывода чисел по универсальному формату G нами для экономии места указываются не все пробелы (т.к. расположение чисел в заданных позициях автоматически выполняется ЭВМ).

Отметим, что оператор типа (7.38) используется для выводов одномерных массивов при работе программ STAT, STATN, STAT9 и STAT9N (см. распечатку в п. 6.3).

7.8.2. Ввод-вывод двумерных массивов

Все вышесказанное переносится и на двумерные массивы, т.к. формы неявного цикла DO могут быть вложенными и их достоинства возрастают с увеличением количества элементов в списке. Аналогично примерам (7.31)–(7.33) также возможны 3 формы операторов ввода двумерных массивов, которые рассмотрим на примере ввода матрицы A1 (6.5), состоящей из 9S9 элементов (число строк N=9 равно числу столбцов M). Операторы (7.39)–(7.41) также эквивалентны:

```

  READ *, A1 (7.39)

```

```

  READ *, A1 (1,1), A1 (1,2), ..., A1 (1,9), A1 (2,1), ...,
  * A1 (2,9), ..., A1 (8,9), A1 (9,1), A1 (9,2), ..., A1 (9,9) (7.40)

```

```

  READ *, ((A1 (I,J), J=1,N), I=1,N) (7.41)

```

Так как в операторе (7.39) указан только идентификатор массива, то должен быть прочтен весь массив. Поэтому ввод будет продолжаться, пока не будут введены все (81) элементы, которые должны быть расположены по столбцам в произвольном количестве строк: в 9-ти, т.е. каждый столбец в одну строку, как в примере (7.17), или в произвольном количестве строк по мере их заполнения, которое производится по столбцам:

```
2, 0, 1, 3*0, 6, 0, -1, -3, 1, 7*0, 7, 0, 1, 6*0, 7*0, 1, 0, 8*0, 1,
4*0, 1, 2*0, -1, 0, 5*0, 1, -2, 0, -1, 0, 1, 0, 1, -1, 4*0, 2*0, 1, 10, 0, -1, 3*0
```

Напомним, что матрица всегда при вводе в память ЭВМ и при выводе на печать как бы “переворачивается”.

Если вводимая матрица расположена по строкам, то при вводе она “переворачивается” и располагается в памяти по столбцам в виде одномерного массива. При выводе, например для контрольной печати, выполняемой для проверки введенных элементов, она опять “перевернется” и будет распечатана в естественной форме по строкам, т.е. в таком же виде, как и была приготовлена для ввода. Внешне все кажется правильным, но все расчеты, выполняемые с матрицей, введенной таким образом, будут неверными, т.к. они производятся с элементами, значения которых не соответствуют их номерам из-за расположения матрицы в памяти в “перевернутом” виде по столбцам.

Чтобы результаты расчетов были правильными, матрица должна располагаться в памяти ЭВМ по строкам. Для этого вводимую матрицу предварительно располагают в “перевернутом” виде по столбцам. При вводе она “перевернется” и расположится в памяти ЭВМ нужным образом в естественной форме по строкам (конечно, в виде последовательного одномерного массива). Отметим, что это старый и неудобный способ ввода двумерных массивов.

Оператор (7.40) позволяет обойти неудобство предварительного расположения матрицы по столбцам из-за ее “переворачивания” при вводе, т.к. он сразу располагает значение вводимого элемента по указанному в списке ввода номеру. Для этого все элементы матрицы должны быть перечислены по строкам (без пропусков в виде ...), что весьма трудоемко и громоздко.

Оператор (7.41) представляет собой сокращенную форму записи оператора (7.40), ибо формы неявного цикла DO могут быть вложенными. Обратим внимание, что здесь используется как бы “перевернутый” ввод, т.е. второй индекс J встречается в операторе (7.41) раньше первого индекса I. Поэтому саму матрицу “переворачивать” не нужно и для ввода она может быть расположена в естественной форме по строкам. Это намного удобнее, особенно при проверке и поиске ошибок.

Форма неявного цикла DO по оператору (7.41) позволяет для двумерных массивов вводить как всю матрицу целиком (в данном случае при $N=9=M$), так

и любую ее часть (при других границах параметров циклов по I и J). В несколько видоизмененной форме записи (см. оператор (7.43)) его также можно использовать для ввода значений элементов на указанные места в матрице, не меняя каждый раз форму записи оператора.

Форма неявного цикла DO для двумерных массивов аналогична обычному вложенному оператору цикла DO и для оператора (7.41) имеет вид:

```
DO 7 I=1, N
DO 7 J=1, N
7 READ *, A1 (I, J)
```

(7.42)

Действие обеих групп операторов в этом частном случае одинаково: числа последовательно по строкам записываются в элементы с A1(1,1) по A1(9,9) массива A1.

Однако в примере (7.42) оператор READ реально встречается 81 раз за время выполнения обоих циклов (по I и по J), автоматически выбирая каждый раз новую строку записи для каждого элемента массива. Поэтому в этом случае также необходимо представлять все числа по одному в каждой строке, что для матриц еще более громоздко и неудобно из-за большего (в квадрате) числа элементов и еще большей их разреженности в задачах статистики.

В операторе (7.41) форма неявного цикла DO также является частью оператора READ. В связи с этим форма входных данных и в этом случае, как и для оператора (7.39), может быть произвольной, но только расположенной в естественной форме по строкам. При этом вводимые данные могут быть представлены как в полном виде с указанием всех элементов (см. матрицу A1 в (6.5)), так и с использованием повторителей для нулевых элементов (см. матрицу A1 в (6.6)).

Напомним, что с формой оператора (7.41) мы уже встречались под номером (4.2) при подготовке данных для работы программы STAT.

Аналогично оператору (7.35), дополним список ввода номерами строки I и столбца J вводимого элемента A1(I,J). Организуем также неявный цикл DO по параметру цикла I1. В результате получим возможность выборочного заполнения матрицы:

```
READ *, K1NEKO, (I, J, A1 (I, J) , I1=1, K1NEKO)
```

(7.43)

По оператору (7.43) сначала будет прочитано значение K1NEKO, которое является верхней границей изменения параметра вложенного в оператор READ цикла по I1. Далее начнет выполняться неявный цикл DO, заключенный в скобки. Сначала каждый раз будет прочитан номер строки I и столбца J вводимого элемента, а затем этот указанный номер (I,J) заполнится следующим за ним в строке данных нужным значением A1(I,J). Таких повторений

считывания указанных номеров (I,J) и их заполнения соответствующим значением матрицы A1(I,J) будет сделано заданное число KNNEKO раз. Предварительно задав всем элементам массива нулевые значения (см. п. 7.7), можно по оператору (7.45) организовать точный и надежный способ ввода только ненулевых значений элементов на заранее указанные места. С ним мы уже встречались ранее под номером (4.4) при подготовке данных для работы программы STATN.

Аналогично примерам (7.39)-(7.42) может быть организован вывод двумерных массивов, для чего достаточно в них оператор READ заменить на PRINT (что рекомендуем проделать на ЭВМ самостоятельно). Мы рассмотрим только использование формы неявного цикла DO в операторе PRINT:

```
PRINT *, ( (A1 ( I, J ), J=1, N ), I=1, N) (7.44)
```

которая позволяет матрицу, расположенную в памяти ЭВМ по строкам (в виде одномерного массива), вывести также расположенной по строкам. Обратим внимание, что вывод матрицы также как бы “перевернут” (второй индекс J встречается в операторе (7.44) раньше первого индекса I), как и в операторе (7.39). Поэтому и при выводе не происходит “переворачивания” матрицы и расположения ее по столбцам.

Чтобы расположить данные в естественной форме по N элементов в строке, можно использовать оператор FORMAT с соответствующим повторителем (дополнив для наглядности его еще заглавием):

```
PRINT 20, ( (A1 ( I, J ), J=1, N ), I=1, N) (7.45)
20 FORMAT (5X, 'ВХОДНАЯ МАТРИЦА ЛЕВОЙ ЧАСТИ A1' /
*9 (2X, 9F8.3) )
```

В примере (7.45) оператор PRINT под управлением оператора FORMAT сначала в 1-й строке, отступив 5 пробелов, напечатает заголовок: “ВХОДНАЯ МАТРИЦА ЛЕВОЙ ЧАСТИ A1”. Затем в строках 2-10 будет распечатана вся матрица A1: 9 строк по 9 элементов.

Если вы хорошо изучили раздел 7.6.6, то этот же список вывода можно представить с указанием номера выводимого элемента, например, используя универсальную спецификацию G:

```
PRINT 20, ( (I, J, A1 ( I, J ), J=1, N ), I=1, N) (7.46)
20 FORMAT (5X, 'ВХОДНАЯ МАТРИЦА ЛЕВОЙ ЧАСТИ A1' /
*5 (2X, 'A1 (', I2, ', ', I2, ')=' , G12.5) )
```

После 1-й строки заголовка, одинакового с примером (7.45), со 2-й строки начнет выполняться группа спецификаций, заключенная в скобки. Сначала, отступив 2 пробела, будет напечатано литеральное данное “A1 (“ и согласно списка вывода оператора PRINT по формату I2 будет напечатан I-й номер

1-го элемента, потом будет напечатано литеральное данное “,” (запятая), за которым по формату I2 — J-й номер 1-го элемента. Потом будет напечатано литеральное данное “)” и после него по спецификации G12.5 будет напечатано значение 1-го элемента (в результате получится $A1(1,1)=2.0$). Такая группа спецификаций во 2-й строке через два пробела друг от друга будет еще напечатана 4 раза для следующих 4-х элементов 1-й строки $A1(1,2)$ – $A1(1,5)$, ибо повторитель перед этой группой спецификаций равен пяти. Так как список вывода еще не закончился, то повторный просмотр спецификаций формата, начинающийся всегда с последней группы спецификаций, будет распечатывать всю матрицу A1 построчно по 5 элементов в строке (с указанием номеров I,J каждого элемента и его значения): в 3-й строке $A1(1,6), \dots, A1(1,9), A1(2,1)$; в 4-й строке от $A1(2,2)$ до $A1(2,6)$;...; в 17-й строке от $A1(9,4)$ до $A1(9,8)$. При 17-м просмотре указанной группы спецификаций в 18-й строке будет напечатано $A1(9,9)=.0$ и литеральное данное “A1 (“, после чего выполнение оператора PRINT закончится, т.к. исчерпан список вывода матрицы A1 и 82-го элемента в нем нет.

Отметим, что оператор аналогичного типа используется для вывода матриц при работе программ STAT, STATN, STAT9 и STAT9N.

7.9. Понятие о подпрограммах и функциях

Часто при программировании возникает необходимость неоднократного выполнения в различных частях программы одной и той же последовательности операторов, но при различных значениях входящих в них параметров. В Фортране предусмотрена возможность оформления этой последовательности в виде оператора-функции, подпрограммы-функции FUNCTION или подпрограммы-процедуры SUBROUTINE (см. п. 12.1). Это экономит память, время и оказывается весьма удобным при выполнении расчетов. При таком оформлении указанная последовательность операторов записывается один раз, а в соответствующие места программы помещаются лишь обращения к этой последовательности для ее выполнения с требуемыми значениями параметров.

Подпрограммы являются мощным средством языка Фортран и на нем создано богатейшее программное обеспечение в виде громадного набора готовых подпрограмм практически по всем разделам математики [12], [19], [20], [23], [27]. Поэтому ими нужно научиться пользоваться.

При решении задачи статики по определению реакций опор, которая включает в себя решение системы линейных алгебраических уравнений, можно использовать готовую подпрограмму. Она проделает все необходимые для этого вычисления по выбранному методу численного решения этой системы. Единственное, что при этом необходимо — это написать основную программу, учитывая правила использования подпрограмм.

В настоящей 3-й части пособия рассматривается только использование готовых подпрограмм общего вида SUBROUTINE, к которым относится большинство подпрограмм (составление их и подпрограмм FUNCTION и операторов-функций будут рассмотрены в 5-й части пособия).

7.10. Порядок следования операторов в программной единице

Обычно реальная программа состоит из нескольких программных модулей, к которым относятся основная программа, с которой начинается решение задачи (часто ее называют MAIN или PROGRAM), и различные подпрограммы.

Первым оператором подпрограммы должен быть в зависимости от ее вида оператор FUNCTION или SUBROUTINE, за которым указывается символическое имя подпрограммы, представляющее собой любой набор от 1-го до 6-ти символов латинского алфавита и цифр, причем 1-м символом должна быть буква. После символического имени в скобках записываются через запятую так называемые формальные параметры — идентификаторы, которые внутри подпрограммы используются как имена простых переменных, массивов и подпрограмм. Например, для подпрограммы DECOMP (см. п. 8.2):

```
SUBROUTINE DECOMP (NDIM, N, A, COND, IPVT, WORK) (7.47)
```

В подпрограммах после оператора FUNCTION или SUBROUTINE, а в основной программе с самого начала, следуют операторы описания типа и массива: REAL, INTEGER, DOUBLE PRECISION, DIMENSION (см. п. 7.1 и 7.2), если они используются в данном модуле.

Затем записываются определения используемых операторов функций. После этого следуют исполнимые (выполняемые) операторы Фортрана (см. п. 7.4 и 7.5).

Последним выполняемым оператором обязательно должен быть: — для основной программы оператор останова STOP; — для любой подпрограммы оператор возврата RETURN.

Оператор STOP прекращает выполнение основной программы и возвращает управление операционной системе для перехода к следующей программе.

Оператор RETURN обеспечивает передачу управления (возврат) в вызывающую программу (или подпрограмму), в точку, из которой производилось обращение. Так как обращение к подпрограмме SUBROUTINE осуществляется при помощи специального оператора CALL (см. п. 7.11), то после оконча-

ния работы вызванной подпрограммы по оператору RETURN осуществляется возврат в вызывающую программу (или подпрограмму) к оператору, следующему за CALL.

Невыполняемый оператор END должен быть физически последним оператором в любом программном модуле. Обычно он располагается после оператора STOP или RETURN. Он не порождает команд в программе, а только сигнализирует транслятору о том, что в данном программном модуле операторов больше нет.

Оператор FORMAT (см. п. 7.6.6) может располагаться в любом месте программного модуля до оператора END, но его обычно помещают или сразу за соответствующим оператором WRITE или PRINT, или располагают все операторы FORMAT в одном месте (например, в конце).

7.11. Понятие о формальных и фактических параметрах. Оператор CALL

Главной особенностью любого модуля является его автономность. Поэтому каждый модуль может составлять и транслироваться независимо от других модулей. Метки и переменные в каждом модуле локализованы, поэтому одна и та же переменная с одним и тем же именем (идентификатором) в разных модулях может обозначать совершенно разные величины. Это происходит потому, что память разделяется на локальные секции так, что каждый модуль имеет доступ лишь к своей отдельной секции. Программист должен сам позаботиться об организации информационной связи между отдельными модулями, для чего наиболее гибким и универсальным способом является использование аппарата формальных— фактических параметров.

Формальные параметры — это идентификаторы, которые внутри подпрограммы используются как имена простых переменных, массивов и подпрограмм. Список формальных параметров через запятую помещается в скобках после указания имени подпрограммы (см. (7.47)). Для подпрограммы общего вида это можно записать:

SUBROUTINE имя (список формальных параметров через запятую)

Подпрограмма-процедура выполняется при обращении к ней из другого программного модуля с помощью оператора процедуры CALL, в котором после указания имени вызываемой подпрограммы помещается в скобках список соответствующих фактических параметров через запятую:

CALL имя(список фактических параметров через запятую)

При вызове подпрограммы с данным именем оператором процедуры CALL ей будет передано управление. Она начнет выполняться с начала и до конца, но во всех операторах подпрограммы на место формальных параметров будут поставлены соответствующие фактические параметры, с которыми и будут выполнены все вычисления.

После окончания работы вызванной подпрограммы по оператору RETURN произойдет возврат в вызывающую программу и начнет выполняться оператор, следующий за оператором процедуры CALL. Теперь в вызывающей программе все указанные в списке фактические параметры будут иметь соответственно те значения, которые они получили в подпрограмме после ее выполнения (пока следующие операторы вызывающей программы их не изменят).

Фактические параметры должны соответствовать формальным параметрам вызываемой подпрограммы своим порядком следования, типом, количеством, размерами массива: первому формальному параметру соответствует первый фактический, второму формальному параметру соответствует второй фактический и т.д. Используемые обозначения для идентификаторов не играют здесь абсолютно никакой роли (кроме, конечно, указания типа величины), так как соответствие проводится по порядку следования параметров.

Однако везде в пособии для ясности изложения и удобства чтения используются обозначения, которые аналогичны или даже одинаковы с обозначениями формальных параметров. Это дает возможность повторно не описывать те фактические параметры, обозначения которых совпадают с соответствующими формальными параметрами, используемыми при описании подпрограмм. Они повторно не рассматриваются. Например, для вызова подпрограммы DECOMP (см. оператор(7.47)) для матриц A1 и A2 и определения их обусловленности (см. п. 8.1), можно использовать операторы:

```
CALL DECOMP (NDIM, N, A1, COND1, IPVT1, WORK) (7.48)
```

```
CALL DECOMP (NDIM, N, A2, COND2, IPVT2, WORK) (7.49)
```

Список формальных параметров оператора (7.47) соответствует спискам фактических параметров операторов (7.48) и (7.49) своим порядком следования, типом, количеством и размерами массивов (описание подпрограммы DECOMP см. в п. 8.2).

ГЛАВА 8. ИСПОЛЬЗОВАНИЕ ГОТОВЫХ ПОДПРОГРАММ

Замечено, что многие люди, решающие практические численные задачи, остаются навсегда верными раз использованным методам и даже подпрограммам, т.к. в процессе практической работы большинство инженеров и научных работников не имеют ни времени, ни склонности следить за новинками текущей литературы по численному анализу.

Поэтому авторы работы [27], монографию которых мы настоятельно рекомендуем, решили описать самые лучшие в мире (на момент выхода книги) подпрограммы для решения стандартных математических задач, составившие небольшую библиотеку высококачественных подпрограмм по основным численным методам. К их числу для решения систем линейных алгебраических уравнений относятся описываемые в данной главе подпрограммы DECOMP и SOLVE (представленные в приложении в переделанном варианте с двойной точностью DDECOM и DSOLVE).

8.1. Краткий обзор методов решения систем линейных алгебраических уравнений

Любая система уравнений равновесия задач статики представляет собой систему линейных алгебраических уравнений, которую можно записать в матричной форме в виде:

$$(A) (X) = B,$$

где: A — заданная квадратная матрица коэффициентов перед неизвестными, в которой выписаны только отличные от нуля значения элементов матрицы $A(I,J)$;

X — неизвестный вектор-столбец с N компонентами (искомые реакции, усилия и т.д.);

B — заданный вектор-столбец с N компонентами, которые представляют собой результат всех перенесенных в правую часть членов каждого уравнения, не содержащих неизвестных.

Многие задачи анализа и синтеза физических систем различной природы (механических, гидравлических, электрических и т.п.) сводятся также к решению систем линейных алгебраических уравнений.

Если определитель матрицы A не равен нулю, то значения неизвестных X_I могут быть получены с использованием формулы Крамера:

$$X_I = \text{DET}(A_I) / \text{DET}(A),$$

где $\text{DET}(A_I)$, $\text{DET}(A)$ — соответственно определитель матриц A_I и A , $I=1, \dots, N$. Матрица A_I образуется из матрицы A заменой ее I -го столбца столбцом свободных членов.

Однако формулы Крамера при численных расчетах на ЭВМ не применяются из-за трудоемкости процесса вычисления всех определителей, катастрофически резко возрастающего с увеличением N [27].

Различают два типа матриц:

Хранимая матрица, все NSN элементов которой вводятся в ЭВМ и хранятся в оперативной памяти машины (обычно при $N < 100$). Для их решения используются обычно точные (прямые) методы, которые в предположении, что вычисления ведутся без округлений, позволяют получить точное решение за конечное число арифметических операций. К ним относятся широко применяемые методы Гаусса с выбором главного (ведущего) элемента.

Разреженная матрица, большинство элементов которой — нули, а ненулевые элементы хранятся посредством какой-либо специальной структуры данных или регенерируются по мере необходимости. Они возникают при решении дифференциальных уравнений с частными производными с использованием конечно-элементных методов. Для таких систем высокого порядка ($N = 10^* * 3 — 10^* * 6$) со слабо заполненной матрицей коэффициентов выгодно использовать приближенные (итерационные) методы. Окончательное решение получается из некоторого заданного начального посредством получения все более приближающихся к точному результатов. К итерационным методам относится метод Зейделя [31, с. 160-163].

Эти два типа матриц в известной степени пересекаются. Хранимая матрица может иметь много нулевых элементов и быть в то же время разреженной. Это относится в первую очередь к матрицам задач статики, т.к. их порядок обычно невелик ($N < 20$) и для нулевых элементов можно отвести место в оперативной памяти. Вместе с этим отметим, что в задачах статики уже при $N > 6$ точнее и надежнее задавать только ненулевые элементы, модифицируя методы для разреженных матриц (см. п. 4.2.2), а не всю матрицу целиком (см. п. 4.2.1).

Для линейных систем с хранимыми матрицами наиболее часто используется численный метод последовательного исключения неизвестных, называемый обычно именем Гаусса.

Название “метод Гаусса” является собирательным для большой группы алгоритмов, связанных с решением систем линейных алгебраических уравнений, вычислением определителей, разложением матрицы на множители и т.д. Суть метода состоит в приведении матрицы системы A к треугольной, в которой все элементы под главной диагональю равны нулю. Эта процедура называется прямым ходом метода и состоит из $N-1$ шагов. На k -м шаге кратные k -го уравнения вычитаются из оставшихся уравнений с целью исключить k -е неизвестное.

Диагональные элементы треугольной матрицы (называемой иногда треугольным разложением матрицы A) носят название ведущих элементов:

k -й ведущий элемент есть коэффициент при k -м неизвестном в k -м уравнении на k -м шаге исключения.

Если коэффициент при X_k мал, то рекомендуется переставить уравнения перед исключением. Вычисления требуют деления на ведущие элементы и если они намного < 1 , то резко возрастают ошибки расчета.

Рекомендуемая перестановка осуществляется посредством процесса, известного как частичный выбор ведущего элемента: на k -м шаге прямого хода в качестве ведущего берется наибольший (по абсолютной величине) элемент в неприведенной части k -го столбца. Строка, содержащая этот элемент, переставляется с k -й строкой с тем, чтобы перевести ведущий элемент в позицию (k, k) . Такие же перестановки должны производиться с элементами вектора-столбца B , для чего информация о произведенных перестановках запоминается, после которых он преобразуется в некоторый вектор-столбец D .

В результате прямого хода получается эквивалентная исходной система уравнений, содержащая вместо входной матрицы A на ее месте в памяти ЭВМ нужную верхнюю треугольную матрицу, которая позволяет с помощью элементарных вычислений методом обратной подстановки определить значения неизвестных.

Обратная подстановка заключается в последовательном нахождении неизвестных, начиная с определения $X(N)$ из последнего простейшего уравнения $C(N, N) * X(N)$, в котором все коэффициенты, кроме N -го ведущего элемента $C(N, N)$, равны нулю. Определив $X(N) = D(N) / C(N, N)$, с учетом его значения из предпоследнего уравнения определяют $X(N-1)$ и т.д., пока из 1-го уравнения не будет вычислено $X(1)$.

Метод Гаусса с частичным выбором ведущего элемента надежен, прост и наиболее выгоден для хранимых матриц общего вида, поэтому он широко применяется при решении систем линейных алгебраических уравнений на ЭВМ в различных модификациях подпрограмм.

Заметим, что в случае вырожденной (особенной) матрицы будет получено значение ведущего элемента, равное нулю.

8.2. Описание подпрограмм DECOMP и SOLVE

Подпрограмма DECOMP:

```
SUBROUTINE DECOMP (NDIM, N, A, COND, IPVT, WORK) (8.1)
```

выполняет ту часть Гауссова исключения, которая зависит лишь от матрицы: этап прямого хода.

Ее формальными параметрами являются:

NDIM — заявленный строчный размер массива, содержащего матрицу A , указанный в его описании (в операторе DIMENSION или REAL. Он совпадает с размерами используемых одномерных массивов B , IPVT и WORK, заявленных в этом же операторе описания);

N — реальное число строк (количество уравнений равновесия) данной матрицы A , равное числу ее столбцов, так как матрица квадратная;

A — входная матрица коэффициентов перед неизвестными, расположенная в памяти ЭВМ по строкам и представленная в операторе описания (DIMENSION или REAL) в виде двумерного массива строчного размера NDIM;

COND — оценка обусловленности матрицы A [27, с.54-61]. Определяется подпрограммой DECOMP;

IPVT — целочисленный одномерный массив размера NDIM, содержащий необходимую информацию (о числе перестановок и индексах) для работы подпрограммы SOLVE;

WORK — вещественный одномерный массив размера NDIM, представляющий собой рабочее поле для работы подпрограммы DECOMP. Его содержание не несет важной информации.

Параметры NDIM, N , A содержат входную информацию, необходимую для работы подпрограммы DECOMP.

После работы подпрограммы параметры A , COND и IPVT содержат выходную информацию результатов ее работы. На месте матрицы A будет расположена факторизованная верхняя треугольная матрица, полученная в результате прямого хода Гауссова исключения из входной матрицы A , содержащая в своей нижней части информацию о произведенных перестановках.

Подпрограмма SOLVE:

```
SUBROUTINE SOLVE (NDIM, N, A, B, IPVT) (8.2)
```

использует результаты, полученные подпрограммой DECOMP, чтобы с помощью обратной подстановки получить решение для произвольной правой части (вектора-столбца B).

Ее формальными параметрами являются:

NDIM — заявленный строчный размер массива, содержащего A , указанный при описании массива (в операторе DIMENSION или REAL);

N — число строк входной квадратной матрицы A (значения N должны быть одинаковыми на соответствующих местах в списках параметров подпрограмм DECOMP и SOLVE, также и значения NDIM);

A — факторизованная верхняя треугольная матрица, полученная из DECOMP после окончания ее работы;

B — вектор-столбец правых частей уравнений, который должен быть предварительно введен в память ЭВМ;

IPVT — вектор ведущих элементов, полученный из DECOMP.

Входными величинами подпрограммы SOLVE являются все указываемые параметры списка, из которых задается дополнительно (по отношению к DECOMP) только вектор-столбец B .

После окончания работы подпрограммы SOLVE на месте вектора-столбца B располагается вектор решения X системы линейных алгебраических уравнений $(A)(X)=B$, являющийся единственной выходной величиной.

Обращение к подпрограммам DECOMP и SOLVE осуществляется, как и для всех подпрограмм типа SUBROUTINE, по оператору CALL, например:

```
CALL DECOMP (N,N,A,COND,IPVT,WORK)
CALL SOLVE (N,N,A,B,IPVT)
```

где на соответствующих местах в списках фактических параметров (в скобках) указываются нужные переменные или массивы.

Подпрограмма DECOMP вычисляет также оценку обусловленности матрицы COND, которая, напомним, выполняет роль множителя в увеличении влияния относительной ошибки, вызванной любыми погрешностями (округления, определения и др.), на результаты решения. Число COND является также мерой близости (обратно пропорционально) матрицы A к множеству вырожденных матриц: с возрастанием числа COND близость к вырожденным матрицам увеличивается [27, с.56-57].

Если матрица A чисто вырожденная (особенная), то число COND стремится к бесконечности. В этом случае DECOMP присваивает COND значение 1.E32, чтобы сигнализировать, что обнаружена вырожденность (особенность) матрицы. Это бывает на практике из-за грубых ошибок в составлении уравнений равновесия или ошибок ввода исходных данных, поэтому значение числа COND рекомендуется выводить на печать.

Во всех остальных случаях значение COND заключено в пределах от 1.0 до 1.E32.

Отметим, что желательно не использовать подпрограмму SOLVE, если DECOMP присвоила COND значение 1.E32, а предусмотреть прекращение выполнения программы с выдачей соответствующего диагностического сообщения (см. дополнение 8.1).

Важным достоинством подпрограммы DECOMP и SOLVE является возможность работы с матрицами, расположенными по строкам в естественной форме. В них Гауссово исключение приспособлено для этой цели (все внутренние циклы меняют первый индекс). В этом их отличие от так называемых

стандартных подпрограмм (см. п. 9.2). Поэтому матрицу A можно заявлять в операторе описания в виде двумерного массива и подготавливать для ввода в естественной форме по строкам, используя встроенный цикл DO (7.41).

Подпрограммы DECOMP и SOLVE обеспечивают возможность работы с переменными размерами массива для создания универсальных программ. Для этого в основной программе в операторе описания массива следует задать значение NDIM, равное предполагаемой максимальной величине, например NDIM=30:

$$\text{DIMENSION } A(30, 30), B(30), \text{IPVT}(30), \text{WORK}(30) \quad (8.3)$$

Тогда можно будет работать с любой NSN матрицей A и вектором B из N элементов, где действительный рабочий порядок N меняется от задачи к задаче, оставаясь по величине меньше или равно NDIM (конкретное значение N для данной задачи можно просто каждый раз вводить в программу оператором READ).

Переменные размеры массивов NDIM и N являются параметрами подпрограмм DECOMP и SOLVE, для вызова которых требуется задание значений двух этих величин.

Мы рекомендуем сначала во всех программах задавать NDIM= N и только после приобретения устойчивых навыков программирования (и желательного знакомства с п. 9.2.1) пытаться создавать универсальные программы, позволяющие выполнять на ЭВМ различные задачи статики (с разным числом уравнений равновесия N) по одной программе.

Отметим, что при написании универсальных программ ввод и вывод должен обязательно осуществляться с использованием неявного цикла DO (см. п. 7.8), так как при $N < \text{NDIM}$ заполняется или соответственно распечатывается только занятая часть N массива размером NDIM.

Подпрограммы DDECOM и DSOLVE, представленные в приложении 1, представляют собой варианты подпрограмм DECOMP и SOLVE в удвоенной точности. Напомним, что в обычной точности они содержатся в работе [27, с. 65-70].

8.3. Составление программ для решения задач статики с использованием подпрограмм DECOMP и SOLVE

8.3.1. Простейшая программа для решения задач статики

Будем вести изложение на типовом примере решения задания С-9 из сборника [21, с. 51,55-59], рекомендованного к выполнению с применением ЭВМ и использованного также в части 2-й при работе с готовыми программами. Он приводит к двум системам линейных алгебраических уравнений (6.1) и (6.2).

Простейшая программа для их решения состоит из следующих основных блоков: 1) описание массивов; 2) ввод исходных данных; 3) контрольная печать введенных исходных данных; 4) обращение к подпрограммам; 5) печать выходной информации; 6) операторы STOP и END. Она может быть реализована, например, следующим образом:

```

С      П Р О Г Р А М М А 8.1
      DIMENSION A(9,9),B(9),IPVT(9),WORK(9)           10
      READ *,N,A,B                                     20
      PRINT *,N,A,B                                    30
      CALL DECOMP(9,N,A,COND,IPVT,WORK)               40
      CALL SOLVE(9,N,A,B,IPVT)                       47
      PRINT *,B,COND                                   50
      STOP                                             60
      END                                              61

```

Номера операторов справа проставлены только для удобства дальнейших пояснений. Этот прием будет использоваться и в дальнейшем.

Напомним, что в программах 8.1-8.3 для удобства первоначальной ориентировки 1-я цифра номера указывает на соответствующий блок: например, операторы 40 и 47 относятся к блоку 4) обращение к подпрограммам.

В программе 8.1 оператор 10 задает описание массивов, по которому отводится место в памяти для размещения двумерного массива A, содержащего 81 элемент, и одномерных массивов B, IPVT и WORK, соответственно по 9 элементов каждый.

Оператор 20 осуществляет ввод количества уравнений равновесия N, значений элементов матрицы A и вектора-столбца B и размещает их в соответствующих массивах A и B. Элементы матрицы A для ввода по оператору 20 должны располагаться в строках файла данных по столбцам. Ввод будет продолжаться, пока не будет исчерпан весь список ввода (для рассматриваемого примера $1+81+9=91$ элемент):

```

9
2,0,1,3*0,6,0,-1
-3,1,7*0
7,0,1,6*0
7*0,1,0
8*0,1                                           (8.4)
4*0,1,2*0,-1,0
5*0,1,-2,0,-1
0,1,0,1,-1,4*0
2*0,1,10,0,-1,3*0
-20,2*0,30,0,-3,-49.96,5,10.66

```

Если по ошибке будет пропущен какой-либо элемент (список из 90 элементов), то эта ошибка ввода повлечет системное прерывание и прекращение

выполнения программы. Указанный же в списке лишний (92-й) элемент просто не будет прочитан. В обоих этих случаях поиск ошибки представит собой очень сложную задачу из-за неестественной формы представления элементов матрицы A по столбцам.

Оператор 30 выполнит контрольную печать введенных исходных данных. Так как вывод по оператору 30 управляется списком (без разделения на столбцы и строки), то в нем будет довольно трудно разобраться (см. пример (7.19)). Надеемся, что после первых попыток это вызовет у вас желание лучше изучить п. 7.6.6 и организовать вывод с использованием оператора `FORMAT` (см. программу 8.2).

Операторы 40 и 47 осуществляют обращение к подпрограммам `DECOMP` и `SOLVE` (см. п. 8.2) со следующими входными фактическими аргументами:

9 — заявленный строчный размер `NDIM` массива A , указанный в операторе `DIMENSION` (т.к. для простоты $NDIM=N$, а $N=9$, то значение `NDIM` может быть задано значением переменной N);

N — число строк матрицы A в данном случае (равно 9);

A — входная матрица A (строки 2-10 данных примера (8.4))

B — вектор-столбец правых частей уравнений (последняя строка фрагмента (8.4)).

После выполнения подпрограммы `SOLVE` в векторе-столбце B будет получено решение системы — искомые реакции.

Оператор 50 выводит на печать результат: искомые реакции (9 корней системы) и оценку обусловленности матрицы A `COND`.

Оператор 60 указывает на конец вычислений, а оператор 61 указывает на конец основной программы.

После окончания работы программы 8.1 выполнена только половина задания С-9: если $B(1) < 0$, то следует повторить еще раз выполнение программы. Вводимые данные для 2-го раза будут иметь также вид (8.4), только на месте матрицы $A1$ должна находиться матрица $A2$, также расположенная по столбцам.

Эту же цель проще достигнуть с использованием оператора цикла `DO` (см. п. 7.5.4), расположив его под номером 15 после оператора 10:

```
DO 5 K=1,2 15
```

и изменив оператор 50 на нижеследующий:

```
5 PRINT *, B, COND 50
```

В этом случае исходные данные должны быть представлены в виде (8.4) и входных данных для 2-го случая ($N, A2, B$), расположенных в следующих строках сразу за (8.4). Теперь программа будет выполнена дважды и просчитаны оба случая задания С-9.

Вместо оператора 20 в программе 8.1 возможно использование оператора DATA (см. п. 7.7.1), например:

```
DATA N/9/, A/2., 0., 1., 3*0., 6., 0., -1., -3., 1., 7*0., 7., 0.,
1 1., 13*0., 1., 9*0., 1., 4*0., 1., 2*0., -1., 6*0., 1., -2., 0.,
2 -1., 0., 1., 0., 1., -1., 6*0., 1., 10., 0., -1., 3*0.0/, (8.5)
3 B/-20., 0., 0., 30., 0., -3., -49.96, 5., 10.66/
```

Оператор (8.5) полностью эквивалентен оператору 20: задает начальные значения переменной N, всем элементам матрицы A1 и вектора-столбца B1 (носящим в программе 8.1 идентификаторы A и B). Обратим внимание, что элементы матрицы A1 также расположены по столбцам.

Вместо операторов 10 и 20 в программе 8.1 возможно использование операторов явного описания типа: REAL и INTEGER (см. п. 7.7.2), например:

```
INTEGER N/9/, IPVT(9) (8.6)
REAL A(9,9)/2., 0., 1., 3*0., 6., 0., -1., -3., 1., 7*0., 7., 0.,
1 1., 13*0., 1., 9*0., 1., 4*0., 1., 2*0., -1., 6*0., 1., -2., 0.,
2 -1., 0., 1., 0., 1., -1., 6*0., 1., 10., 0., -1., 3*0.0/, WORK(9),
3 B(9)/-20., 0., 0., 30., 0., -3., -49.96, 5., 10.66/ (8.7)
```

Совместное действие операторов (8.6) и (8.7) полностью эквивалентно действию операторов 10 и 20 программы 8.1: описываются массивы A, B, IPVT, WORK и присваиваются начальные значения переменной N, всем элементам матрицы A и вектора B. Кроме этого еще явно описывается тип переменных. Обратим внимание, что элементы матрицы A1 (носящей в программе 8.1 идентификатор A) также расположены по столбцам, как это делается всегда, когда не используется ввод со встроенным (и “перевернутым”) циклом DO (см. п. 7.8.2).

При использовании операторов (8.5) или (8.6) и (8.7) для задания элементов матрицы A2 уравнений (6.2) нельзя использовать цикл DO, а необходимо повторить еще раз выполнение программы, изменив в операторах (8.5) и (8.7) матрицу A1 на A2. Использование этих операторов для одновременного решения двух систем уравнений (6.1) и (6.2) требует другого программного решения и рассмотрено в дополнениях 8.9 и 8.10 к программе 8.4.

8.3.2. Организация ввода матриц по строкам

Можно довольно легко усовершенствовать программу 8.1, избавившись от одного из ее самых существенных недостатков: ввода матрицы по столбцам. Для возможности ее расположения в естественной форме по строкам нужно использовать для матрицы A ввод со встроенным неявным циклом DO (см. (7.41)), заменив оператор 20 в программе 8.1:

```
READ *, N, ((A(I, J), J=1, N), I=1, N), (B(I), I=1, N) 20
```

Он эквивалентен трем нижеследующим, которыми удобнее пользоваться:

```

READ *, N
READ *, ((A(I, J), J=1, N), I=1, N)
READ *, (B(I), I=1, N)

```

Теперь легко выполнить рекомендации п. 8.3.1 по решению нескольких вариантов задания за один выход на ЭВМ. Для этого будем указывать их количество KSU в первом операторе ввода (READ *,N,KSU) и поставим оператор цикла DO сразу после него. Заодно сделаем некоторые другие усовершенствования, касающиеся ясности представления выходной информации (и не изменяющие структуры основных блоков 1)-6) программы 8.1):

```

C      П Р О Г Р А М М А 8.2
      DIMENSION A(9,9), B(9), WORK(9), IPVT(9)           10
      READ *, N, KSU                                     20
      DO 75 K=1, KSU                                    21
      READ *, ((A(I, J), J=1, N), I=1, N)              22
      READ *, (B(I), I=1, N)                            23
      PRINT 20, K, ((A(I, J), J=1, N), I=1, N)         30
20  FORMAT(/5X, 'ВХОДНАЯ МАТРИЦА ЛЕВОЙ ЧАСТИ A', I2/   31
      * (1X, 9F8.3))
      PRINT 25, K, (B(I), I=1, N)                       32
25  FORMAT(5X, 'ВЕКТОР ПРАВЫХ ЧАСТЕЙ B', I2/1X, 9F8.3) 33
      CALL DECOMP(N, N, A, COND, IPVT, WORK)           40
      PRINT 30, K, COND                                  41
30  FORMAT(5X, 'ОЦЕНКА ОБУСЛОВЛЕННОСТИ МАТРИЦЫ A', I2, 1X, 42
      *' COND=', G12.5)
      CALL SOLVE(N, N, A, B, IPVT)                     47
      PRINT 65, K, (B(I), I=1, N)                      50
65  FORMAT(15X, 'РЕЗУЛЬТАТЫ РЕШЕНИЯ СИСТЕМЫ УРАВНЕНИЙ ', 51
      *' НОМЕР', I2/1X, 9F8.3)
      75 CONTINUE                                       59
      STOP                                             60
      END                                              61

```

Оператор 10 задает описание массивов A, B, IPVT, WORK, по которому им отводится соответствующее место в памяти (значение NDIM=9).

Оператор 20 осуществляет ввод количества уравнений равновесия N в одном варианте и число вариантов KSU, выполняемых за один выход на ЭВМ (для (6.1) и (6.2) N=9 и KSU=2).

Оператор 21 — оператор цикла DO — организует цикл по K, в котором будут выполняться операторы 22-59 (последний помечен меткой 75) при K=1, ..., KSU (см. п. 7.5.4).

Операторы 22 и 23 производят ввод матрицы A и вектора-столбца B , используя форму неявного цикла DO (см. операторы (7.41) и (7.33)). Элементы матрицы A представляются в естественном виде по строкам (в любой из форм (6.5) или (6.6)).

Операторы 30 и 32 под управлением соответственно операторов 31 и 33 и осуществляют контрольную печать исходных данных для проверки правильности ввода матрицы A и вектора-столбца B каждого варианта (см. распечатку (8.9)).

При каждом значении параметра K оператор 30 под управлением оператора 31 сначала пропустит одну строку пробелов (согласно знака перехода к следующей записи “/”). Затем, отступив 5 пробелов, будет напечатан заголовок: “ВХОДНАЯ МАТРИЦА ЛЕВОЙ ЧАСТИ A”, за которым согласно списка вывода оператора 30 будет напечатано значение K (при $K=1$ получится A_1 , при $K=2$ — A_2 и т.д.). В следующую строку (из-за использования разделителя “/”) выводятся 9 элементов матрицы A (ее 1-я строка). Далее, согласно повторяющейся при просмотре группе спецификаций (1X,9F8.3), будут распечатаны по строкам остальные значения элементов матрицы A (при других значениях N для вывода матриц в естественной форме желательно исправить коэффициент 9 в группе спецификаций).

Оператор 32 под управлением оператора 33 при каждом значении K , отступив 5 пробелов, напечатает заголовок: “ВЕКТОР ПРАВЫХ ЧАСТЕЙ B”, за которым также будет напечатано значение K (при $K=1$ получится B_1 и т.д.). В следующей строке будут распечатаны 9 элементов вектора-столбца B по спецификации F8.3: каждое из чисел выводится в 8 позиций, из которых 3 позиции отводятся под дробную часть.

Операторы 40 и 47 осуществляют обращение к подпрограммам DECOMP и SOLVE. Их отличие от аналогичных операторов программы 8.1 состоит в том, что значение заявленного строчного размера NDIM (первый параметр списка) задано значением переменной N (а не цифрой), т.к. пока принимается для простоты $NDIM=N$. Так как в программе 8.2 операторы 40 и 47 вместе с операторами 22 и 23 стоят в цикле по K , то при каждом значении K в программе будут введены (при повторении выполнения операторов 22 и 23) соответствующие массивы: $A_1, B_1, A_2, B_2, \dots, A_k, B_k$. Они поочередно будут подставляться в качестве фактических параметров в операторы 40 и 47. После окончания работы подпрограммы SOLVE в векторе-столбце B будет получено каждый раз решение соответствующей системы — искомые реакции: $X_1(I), X_2(I), \dots, X_k(I)$, где I принимает значение от 1 до N .

Оператор 41 под управлением оператора 42 осуществляет печать для каждой матрицы A_k ее оценки обусловленности COND, определяемой под-

программой DECOMP. Отступив 5 пробелов, будет напечатан заголовок: “ОЦЕНКА ОБУСЛОВЛЕННОСТИ МАТРИЦЫ А”, за которым согласно списка вывода оператора 41 будет напечатано значение параметра цикла К. Далее через 1 пробел (1X) будет напечатано литеральное данное “COND=”, за которым согласно списка вывода оператора 41 будет напечатано его значение по универсальной спецификации G12.5.

Оператор 50 также выполняется в цикле по К и под управлением оператора 51 выводит на печать значения искомым реакций. Отступив 15 пробелов (15X), при каждом значении К будет напечатано текстовое сообщение: “РЕЗУЛЬТАТЫ РЕШЕНИЯ СИСТЕМЫ УРАВНЕНИЙ НОМЕР”, за которым по спецификации I2 будет напечатано значение К (1,2,...), а в следующей строке (т.к. в качестве разделителя использован символ “/”) — 9 значений искомым реакций, соответствующих решаемой системе уравнений.

Оператор 59 — фиктивный оператор CONTINUE (см. п. 7.5.5) используется в качестве конечного оператора цикла. Он передает управление на начало цикла оператору 22, если $K < K_{SU}$ или $K = K_{SU}$, или следующему в программе оператору 60, если $K > K_{SU}$.

Использование оператора CONTINUE в данном случае строго не обязательно. Можно убрать его из программы, переставив метку 75 на оператор 50 (PRINT). Хотя оператор 51 (FORMAT) после этого окажется как бы вне цикла, печать результатов и работа программы от этого не изменится, т.к. он не порождает команд при выполнении программы. Поэтому все операторы 31, 33, 42 и 51 можно сгруппировать в любом ее месте до END (см. п. 7.10).

Оператор 60 указывает на конец вычислений, а оператор 61 указывает на конец основной программы.

Исходные данные для работы программы 8.2 могут быть представлены в любой из двух форм (6.5) или (6.6) (без 1-й строки), например, с использованием повторителей:

```

9,2
2,-3,7,6*0
0,1,5*0,1,0
1,0,1,5*0,1
7*0,1,10
5*0,1,0,-1,0
6*0,1,0,-1
6,5*0,-2,2*0
3*0,1,0,-1,3*0
-1,3*0,1,0,-1,2*0
-20,2*0,30,0,-3,-49.96,5,10.66
0,-3,7,6*0

```

(8.8)

```

0,1,5*0,1,0
0,0,1,5*0,1
6,6*0,1,10
5*0,1,0,-1,0
-1,5*0,1,0,-1
-4,5*0,-2,2*0
3*0,1,0,-1,3*0
1,3*0,1,0,-1,2*0
-20,2*0,30,0,-3,-49.96,5,10.66

```

Результаты работы программы 8.2 для рассматриваемого примера будут иметь нижеследующий вид (при просмотре соответствующего файла или при выводе результатов на экран или принтер):

```

ВХОДНАЯ МАТРИЦА ЛЕВОЙ ЧАСТИ А 1
2.0 -3.0 7.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0
1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 10.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 -1.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 -1.0
6.0 0.0 0.0 0.0 0.0 0.0 -2.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 -1.0 0.0 0.0 0.0
-1.0 0.0 0.0 0.0 1.0 0.0 -1.0 0.0 0.0
ВЕКТОР ПРАВЫХ ЧАСТЕЙ В 1
-20.0 0.0 0.0 30.0 0.0 -3.0-49.96 5.0 0.66
ОЦЕНКА ОБУСЛОВЛЕННОСТИ МАТРИЦЫ А 1 COND= 55.560
РЕЗУЛЬТАТЫ РЕШЕНИЯ СИСТЕМЫ УРАВНЕНИЙ НОМЕР 1
-7.976 10.509 3.926 -5.509 3.734 -10.509 1.051 -10.509
4.051
ВХОДНАЯ МАТРИЦА ЛЕВОЙ ЧАСТИ А 2 (8.9)
0.0 -3.0 7.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0
6.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 10.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 -1.0 0.0
-1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 -1.0
-4.0 0.0 0.0 0.0 0.0 0.0 -2.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 -1.0 0.0 0.0 0.0
1.0 0.0 0.0 0.0 1.0 0.0 -1.0 0.0 0.0
ВЕКТОР ПРАВЫХ ЧАСТЕЙ В 2
-20.0 0.0 0.0 30.0 0.0 -3.0 -49.960 5.0 10.66
ОЦЕНКА ОБУСЛОВЛЕННОСТИ МАТРИЦЫ А 2 COND = 54.689
РЕЗУЛЬТАТЫ РЕШЕНИЯ СИСТЕМЫ УРАВНЕНИЙ НОМЕР 2
9.949 11.023 1.867 -6.023 5.793 -11.023 5.082 -11.023 -1.867

```

Дополнение 8.1. Перед обращением к подпрограмме SOLVE желательно проверить число обусловленности матрицы COND. Его обратная величина ($1/COND$) характеризует относительную близость данной матрицы к множеству обусловленных (вырожденных) матриц. Если из-за грубых ошибок составления уравнений или ввода исходных данных полученная матрица достаточно близка к обусловленным, то нужно прекратить расчет и не обращаться к подпрограмме SOLVE.

Это может возникнуть при выполнении расчета в обычной точности при $COND > 10^{*8}$, а в удвоенной точности при $COND > 10^{*17}$. При таких значениях COND разность между числами на единицу между $COND P1 = COND + 1$ и COND уже не улавливается при машинном представлении числа, т.к. при обычной точности в памяти ЭВМ сохраняется только 7 значащих цифр, при удвоенной — 16 (хотя сама величина числа по модулю может находиться в диапазоне от $1.E-75$ до $1.E+75$). Поэтому для таких чисел обусловленности ЭВМ не в состоянии отличить $COND P1$ от COND.

Это условие можно принять в качестве обобщенного критерия, характеризующего множество обусловленных матриц, при достижении которого выполнение дальнейших расчетов подпрограммой SOLVE не имеет смысла. Он очень удобен, т.к. он не зависит ни от точности расчета, ни от типа ЭВМ. Контроль вышеописанного критерия обусловленности матрицы выполняет следующий фрагмент программы, который мы рекомендуем вставить между операторами 42 и 47 в программе 8.2 и в соответствующие места всех других ваших программ:

```

COND P1=COND+1                                43
IF (COND P1.EQ.COND) PRINT 40,K              44
40 FORMAT(5X,'ВНИМАНИЕ!- ОБНАРУЖЕНА ВЫРОЖДЕННОСТЬ ', 45
*' (ОСОБЕННОСТЬ) МАТРИЦЫ A',I2,' !!!')
IF (COND P1.EQ.COND) STOP                      46

```

Арифметический оператор присваивания 43 присваивает величине $COND P1$ значение $COND + 1$.

Условный логический оператор IF 44 анализирует на истинность логическое выражение ($COND P1.EQ.COND$). Если оно ложно, т.е. ЭВМ может различить неравенство этих двух отличающихся на единицу величин, то оператор 44 пропускается и начинает выполняться следующий исполнимый оператор программы — также условный логический оператор IF 46. Он анализирует еще раз это же логическое выражение, которое в данном случае также ложно. Поэтому он также пропускается и управление передается следующему оператору 47. В этом случае выполнение программы 8.2 ничем не отличается от описанного ранее.

Если выражение (CONDP1.EQ.COND) для ЭВМ истинно, т.е. число COND достигает значений таких величин, что машинное представление отличающихся чисел CONDP1 и COND неразличимо, то управление передается оператору PRINT 40. Этот оператор под управление оператора 45 FORMAT выводит на печать аварийное сообщение: “ВНИМАНИЕ! – ОБНАРУЖЕНА ВЫРОЖДЕННОСТЬ (ОСОБЕННОСТЬ) МАТРИЦЫ А”, после которого по спецификации I2 печатается значение К (номер матрицы 1,2,...,KSU), за которым через один пробел следуют три восклицательных знака.

Следующий условный логический оператор 46, вследствие истинности при данном значении COND выражения (CONDP1.EQ.COND), передает управление оператору STOP, который указывает на конец вычислений и останавливает выполнение программы.

Дополнение 8.2. Желательно дополнить выходную информацию заголовком с указанием номеров группы NG, задания NZ и варианта NW. Это можно сделать, поместив перед оператором 10 в программе 8.2 оператор:

```
READ *, NG, NZ, NW 5
```

Тогда исходные данные примера (8.8) для работы программы 8.2 следует впереди дополнить одной строкой, соответствующей списку ввода оператора 5 (например 114413,9,31), после чего они будут иметь точное совпадение с исходными данными (6.6).

Напечатать заголовок можно по любому из примеров (7.21)-(7.23). Выбранные два оператора (PRINT и соответствующий FORMAT) нужно поместить также перед оператором 10 программы 8.2 после вышеуказанного оператора 5. Если этого не сделать, то нужно хотя бы строкой комментария вверху программы 8.2 указать аналогичные данные.

Дополнение 8.3. Изучив операторы примеров (7.38) и (7.49), можно дополнить указанием номера выводимого элемента операторы 30-33 и 50-51, после чего они примут вид:

```
PRINT 20, K, ((I, J, A(I, J), J=1, N), I=1, N) 30
20 FORMAT (5X, 'ВХОДНАЯ МАТРИЦА ЛЕВОЙ ЧАСТИ А', I2/  
* 5 (2X, 'А(', I2, ', ', I2, ')=' , G12.5) ) 31
PRINT 25, K, (I, B(I), I=1, N) 32
25 FORMAT (5X, 'ВЕКТОР ПРАВЫХ ЧАСТЕЙ В', I2/  
* 5 (2X, 'В(', I2, ')=' , G12.5) ) 33
PRINT 65, K, (I, B(I), I=1, N) 50
65 FORMAT (15X, 'РЕЗУЛЬТАТЫ РЕШЕНИЯ СИСТЕМЫ ',   
*' УРАВНЕНИЙ НОМЕР ', I2//5 (2X, 'X(', I2, ')=' , G12.5) ) 51
```

Подставив новые операторы 30-33 и 50-51 в программу 8.2 на место старых операторов с этими же номерами, мы получим печать всей выводимой

мой информации с указанием номера выводимого элемента массива, процесс печати которого описан в пояснениях к примерам (7.38) и (7.46).

Дополнение 8.4. В сборнике [21, с. 57] на рис. 66 представлена блок-схема программы, которая не предусматривает печати результатов решения, если соответствующая реакция Re или $Rf < 0$ или $= 0$ (т.е. после работы подпрограммы SOLVE $B(1) < 0$ или $= 0$).

Это очень легко может быть достигнуто в любой из рассматриваемых программ использованием условного логического IF в операторе печати результатов решения 50, что для программы 8.2 будет иметь вид:

```
IF (B(1) .GT. 0.0) PRINT 65, K, (B(I), I=1, N) 50
```

Теперь оператор PRINT, осуществляющий печать результатов решения, будет выполняться только в случае значения $B(1) > 0$ (т.е. Re или $Rf > 0$). Если $B(1) < 0$ или $B(1) = 0$, то он пропускается и выполняется следующий в программе оператор 59 (CONTINUE). Он также, как и оператор FORMAT, не выполняет никаких действий, но ему может передаваться управление при выполнении программы.

Так как на конкретном примере всегда описывается некоторый общий метод, в данном случае печати результатов при выполнении определенного условия, то на месте оператора PRINT с меткой 65 может находиться любой аналогичный оператор. При проведении изменений нужно только привести в согласие с используемым оператором PRINT соответствующий ему оператор FORMAT.

Дополнение 8.5. Чтобы полностью удовлетворить блок-схеме программы, представленной в пособии [21, с. 57] на рис. 66, нужно после выполнения дополнения 8.4 добавить в программу 8.2 после оператора 50 и перед оператором 59 еще один условный логический оператор IF, например:

```
IF (B(1) .GT. 0.0) STOP 52
```

Теперь уже при первом выполнении цикла, если реакция Re или Rf будет > 0.0 (т.е. $B(1) > 0.0$), то условный логический оператор IF передаст управление оператору STOP, указывающему на конец вычислений и останавливающему выполнение программы.

Если же заключенное в скобки выражение ложно (т.е. при $B(1) < 0.0$ или $B(1) = 0.0$), то оператор 52 пропускается. Управление передается следующему оператору CONTINUE с меткой 75. Он в свою очередь передает управление на начало цикла оператору 22 для продолжения выполнения программы: ввода исходных данных для 2-го случая и его последующего решения.

Замечание. Из дополнений 8.1 и 8.5 видно, что в программе может присутствовать несколько операторов останова STOP. Хотя по результатам

распечатки можно определить, по какому оператору STOP прекращено выполнение программы, но это проще сделать использованием этого оператора в общем виде:

```
STOP N
```

где N — либо пусто, либо целое число без знака, содержащее от одной до пяти цифр. Если N не пусто, то в процессе выполнения оператора остановка на пульте управления выдается текст STOP N (например STOP 1000 при предварительном задании N=1000).

Дополнение 8.6. Программу 8.2 можно назвать универсальной только с точки зрения решения произвольного количества вариантов за один выход на ЭВМ для задач с числом уравнений N=9. Если же N не равно 9, то приходится изменять оператор 10 описания массивов, приводя его в соответствие с используемыми размерами (значениями N). Если такое необходимое дополнение вас не устраивает, то можно сразу предусмотреть работу с переменными размерами массивов, что позволяют сделать подпрограммы DECOMP и SOLVE (см. п. 8.2). Для этого в основной программе в операторе описания массива следует задать NDIM, равное его предполагаемой максимальной величине, в результате чего оператор 10 примет вид:

```
DIMENSION A (30, 30) , B (30) , IPVT (30) , WORK (30) 10
```

После него следует поместить оператор 11, задающий значение заданных размеров массивов NDIM в операторе DIMENSION (для двумерного массива A это только строчный размер массива, т.е. число строк):

```
NDIM=30 11
```

Теперь нужно переписать обращения к подпрограммам DECOMP и SOLVE 40 и 47 в общем виде, указав на соответствующем месте (первым в списке фактических параметров) значение NDIM, которое зададим именем переменной:

```
CALL DECOMP (NDIM, N, A, COND, IPVT, WORK) 40
CALL SOLVE (NDIM, N, A, B, IPVT) 47
```

С учетом этих изменений программа 8.2 становится действительно универсальной и может работать с массивами переменного размера, задаваемого числом уравнений N в операторе 20 (которое должно быть только меньше 30).

8.3.3. Организация ввода только ненулевых элементов

Программа 8.2 является универсальной программой для решения систем линейных алгебраических уравнений в произвольном количестве вариантов. Однако она не учитывает разреженный характер матриц, возникающих при решении задач статики. Форма неявного цикла DO в операторе READ, использованная

в программе 8.2, позволяет после небольшой переделки заполнять массив выборочно в соответствии с номером ненулевого элемента, задаваемым впереди значения этого ненулевого элемента (см. операторы (7.35) и (7.43)). Напомним, что в п. 4.2.2 описано определение номера ненулевого элемента вектора-столбца В и матрицы А и подготовка исходных данных для программы STATN.

Для осуществления ввода только ненулевых элементов нужно производить каждый раз предварительное обнуление всех используемых массивов, т.е. выполнять задание нулевых элементов. Это можно сделать, например, с помощью арифметических операторов присваивания $B(I)=0.0$ и $A(I,J)=0.0$, помещенных соответственно каждый в обычный (см. (7.4)) или вложенные циклы (см. (7.6)), которые после их выполнения зададут всем элементам А и В нулевые значения.

Удалив из программы 8.2 операторы ввода 22 и 23, организовав предварительное обнуление массивов А и В (операторы 22-25) и используя примеры (7.43) и (7.35) для ввода данных (операторы 28 и 29), получим программу 8.3. Она учитывает разреженный характер матриц задач статики и позволяет вводить только ненулевые элементы. Введем также в программу 8.3 важное дополнение 8.1 (операторы 43-46). Отметим, что все дополнения не изменяют структуру основных блоков 1)-6) программы 8.1.

```

С      П Р О Г Р А М М А 8.3
      DIMENSION A(9,9), B(9), WORK(9), IPVТ(9)          10
      READ *, N, KSU                                     20
      DO 75 K=1, KSU                                     21
      DO 6 I=1, N                                        22
          B(I)=0.                                       23
          DO 6 J=1, N                                    24
              6      A(I, J)=0.                         25
      READ *, KOR                                       26
      DO 7 M=1, KOR                                     27
          7      READ *, KNNEKO, (I, J, A(I, J), I1=1, KNNEKO) 28
      READ *, KNNEKO, (I, B(I), I1=1, KNNEKO)          29
      PRINT 20, K, ((A(I, J), J=1, N), I=1, N)         30
20     FORMAT(/5X, 'ВХОДНАЯ МАТРИЦА ЛЕВОЙ ЧАСТИ А', I2/ 31
*      (1X, 9F8.3))
      PRINT 25, K, (B(I), I=1, N)                       32
25     FORMAT(5X, 'ВЕКТОР ПРАВЫХ ЧАСТЕЙ В', I2/1X, 9F8.3) 33
      CALL DECOMP(N, N, A, COND, IPVТ, WORK)          40
      PRINT 30, K, COND                                 41
30     FORMAT(5X, 'ОЦЕНКА ОБУСЛОВЛЕННОСТИ МАТРИЦЫ А', I2, 1X, 42
*      ' COND=', G12.5)
      CONDP1=COND+1                                    43
      IF (CONDP1.EQ.COND) PRINT 40, K                  44
40     FORMAT(5X, 'ВНИМАНИЕ!- ОБНАРУЖЕНА ВЫРОЖДЕННОСТЬ ', 45

```

* ' (ОСОБЕННОСТЬ) МАТРИЦЫ A', I2, ' !!!')	
IF (CONDP1.EQ.COND) STOP	46
CALL SOLVE (N, N, A, B, IPVT)	47
PRINT 65, K, (B(I), I=1, N)	50
65 FORMAT(15X, 'РЕЗУЛЬТАТЫ РЕШЕНИЯ СИСТЕМЫ УРАВНЕНИЙ ', 51	
* ' НОМЕР ', I2/1X, 9F8.3)	
75 CONTINUE	59
STOP	60
END	61

Операторы 10, 20 и 21 эквивалентны соответствующим операторам программы 8.2.

Оператор 10 задает описание используемых массивов.

Оператор 20 осуществляет ввод числа уравнений N в одном варианте и количества вариантов решения KSU.

Оператор цикла DO 21 организует цикл по K , в котором будут выполнены операторы 22-59 при $K=1, \dots, KSU$.

Однотипные операции обнуления элементов вектора-столбца B и матрицы A (см. примеры (7.4) и (7.6)) в программе 8.3 совмещены соответственно в один цикл каждый, выполняемый операторами 22-25. Работа такого цикла (с добавлением необязательного в данном случае фиктивного оператора CONTINUE) рассмотрена в примере (7.8) для аналогичной операции копирования массивов.

Оператор 26 вводит количество строк KOR, в которых будут вводиться номера и ненулевые элементы матрицы A . Это число является конечным значением счетчика цикла оператора 27. Он организует цикл по M , в котором многократно выполняется оператор 28 с меткой 7 при $M=1, \dots, KOR$. Каждый раз сначала он вводит число ненулевых элементов KNNEKO матрицы A в данной строке, которое является верхней границей изменения параметра неявного (вложенного в оператор READ) цикла по I . Далее начнет выполняться неявный цикл DO, заключенный в скобки. В нем сначала каждый раз будет прочитан номер строки I и столбца J вводимого элемента, а затем этот указанный номер (I, J) заполнится следующим за ним в строке данных нужным значением $A(I, J)$. Таких повторений считывания номеров (I, J) и их заполнения значением матрицы $A(I, J)$ будет сделано заданное число KNNEKO раз для каждой строки.

Оператор 29 вводит для вектора-столбца B ненулевые элементы в соответствии с указанным номером каждого элемента (который задается впереди значения этого элемента). Он тождественен описанному ранее оператору (7.35).

Операторы 30-42 и 47-61 эквивалентны соответствующим операторам программы 8.2, а операторы 43-46 эквивалентны соответствующим операторам дополнения 8.1 к программе 8.2. Все они подробно рассмотрены в соответствующих местах п. 8.3.2.

Исходные данные для работы программы 8.3 могут быть представлены для рассматриваемых примеров (6.3)-(6.4) в виде:

$$\begin{array}{l}
 9, 2 \\
 3 \\
 8, 1, 1, 2, 1, 2, -3, 1, 3, 7, 2, 2, 1, 2, 8, 1, 3, 1, 1, 3, 3, 1, 3, 9, 1 \\
 6, 4, 8, 1, 4, 9, 10, 5, 6, 1, 5, 8, -1, 6, 7, 1, 6, 9, -1 \\
 7, 7, 1, 6, 7, 7, -2, 8, 4, 1, 8, 6, -1, 9, 1, -1, 9, 5, 1, 9, 7, -1 \\
 6, 1, -20, 4, 30, 6, -3, 7, -49.96, 8, 5, 9, 10.66 \\
 3 \\
 6, 1, 2, -3, 1, 3, 7, 2, 2, 1, 2, 8, 1, 3, 3, 1, 3, 9, 1 \\
 8, 4, 1, 6, 4, 8, 1, 4, 9, 10, 5, 6, 1, 5, 8, -1, 6, 1, -1, 6, 7, 1, 6, 9, -1 \\
 7, 7, 1, -4, 7, 7, -2, 8, 4, 1, 8, 6, -1, 9, 1, 1, 9, 5, 1, 9, 7, -1 \\
 6, 1, -20, 4, 30, 6, -3, 7, -49.96, 8, 5, 9, 10.66
 \end{array} \tag{8.10}$$

Проверьте, что они совпадают с исходными данными для программы STATN (6.10), но без 1-й строки. Полное совпадение может быть получено использованием в программе 8.3 дополнения 8.2. Рекомендуем также выполнить дополнения 8.2-8.6, которые также полностью применимы и к программе 8.3.

Результаты работы программы 8.3 (в основном варианте без дополнений 8.2 и 8.3) совпадают с программой 8.2 и представлены в распечатке (8.9).

8.3.4. Использование особенностей задания С-9 [21, с. 50-59]

Программа 8.3 (с учетом дополнений) является удобной универсальной программой для решения произвольного количества вариантов СЛАУ. Она учитывает разреженный характер матриц, возникающих при решении задач статики, и позволяет вводить только их ненулевые элементы сразу на нужные места.

Однако задание С-9, рекомендованное в сборнике [21] для применения ЭВМ к решению задач статики, обладает одной особенностью. Она заключается в том, что матрицы A_1 и A_2 отличаются только первым столбцом (см. уравнения (6.3)-(6.4)). Векторы-столбцы B_1 и B_2 также одинаковы, так как нагрузка не меняется. Это позволяет почти наполовину сократить необходимые вводимые данные. Для этого после ввода матрицы A_1 и вектора-столбца B_1 в программе предусматривается копирование их в резервные массивы, так как после работы подпрограмм DECOMP и SOLVE исходные массивы не сохраняются. Затем, после решений 1-й системы уравнений (6.3), из резервных массивов восстанавливаются исходные. Теперь нужно только ввести элементы 1-го столбца матрицы (6.4). Это можно сделать двумя различными способами: введя все его элементы (программа 8.4) или обнулить их в самой программе и ввести только отличающиеся ненулевые элементы 1-го столбца матрицы (6.4) (программа 8.5).

Так как программы 8.4 и 8.5 являются специализированными для решения задания С-9, то сразу учтем рекомендацию блок-схемы на рис. 66 [21]. Предусмотрим окончание работы программы после решения системы (6.3) при $X(1) > 0$ и отсутствие печати результатов при $X(1) < 0$.

Учет этих особенностей можно сделать с помощью программ (8.4)-(8.5). Они состоят из следующих основных блоков: 1) описание массивов; 2) ввод исходных данных (6.3) и копирование исходных массивов В и А в резервные ВD и АD; 3) контрольная печать В и А; 4) обращение к подпрограммам; 5) проверка значения реакции В(1): если $V(1) > 0$, то переход к блоку 8) для печати выходной информации; в противном случае дальнейшее выполнение программы; 6) копирование резервных массивов ВD и АD в исходные В и А; 7) ввод элементов 1-го столбца матрицы (6.4) и повторное решение с блока 3); 8) печать выходной информации; 9) операторы STOP и END.

Эта блок-схема при вводе всех значений элементов матрицы А из (6.3) может быть реализована на основе программы 8.2, например, следующим образом:

```

С      П Р О Г Р А М М А 8.4
      DIMENSION A(9,9),B(9),WORK(9),IPVT(9),AD(9,9),BD(9) 10
      READ *,N                                             20
      READ *,((A(I,J),J=1,N),I=1,N)                     22
      READ *,(B(I),I=1,N)                                23
      DO 10 I=1,N                                         24
          BD(I)=B(I)                                     25
          DO 10 J=1,N                                    26
10      AD(I,J)=A(I,J)                                  27
          K=0                                            28
15      K=K+1                                            29
          PRINT 20,K,((A(I,J),J=1,N),I=1,N)             30
20      FORMAT(/5X,'ВХОДНАЯ МАТРИЦА ЛЕВОЙ ЧАСТИ A',I2/ 31
          *(1X,9F8.3))
          PRINT 25,K,(B(I),I=1,N)                       32
25      FORMAT(5X,'ВЕКТОР ПРАВЫХ ЧАСТЕЙ В',I2/1X,9F8.3) 33
          CALL DECOMP(N,N,A,COND,IPVT,WORK)             40
          PRINT 30,K,COND                                41
30      FORMAT(5X,'ОЦЕНКА ОБУСЛОВЛЕННОСТИ МАТРИЦЫ A',I2,1X, 42
          *'COND=',G12.5)
          CONDP1=COND+1                                  43
          IF (CONDP1.EQ.COND) PRINT 40,K                44
40      FORMAT(5X,'ВНИМАНИЕ!- ОБНАРУЖЕНА ВЫРОЖДЕННОСТЬ ', 45
          *'(ОСОБЕННОСТЬ) МАТРИЦЫ A',I2,' !!!')
          IF (CONDP1.EQ.COND) STOP                       46
          CALL SOLVE(N,N,A,B,IPVT)                      47

```

IF (B(1).GT.0.0) GO TO 60	51
DO 50 I=1,N	61
B(I)=BD(I)	62
DO 50 J=1,N	63
A(I,J)=AD(I,J)	64
50 CONTINUE	65
DO 55 I=1,N	70
READ *,A(I,1)	71
55 CONTINUE	72
GO TO 15	74
60 PRINT 65,K,(B(I),I=1,N)	80
65 FORMAT(15X,'РЕЗУЛЬТАТЫ РЕШЕНИЯ СИСТЕМЫ УРАВНЕНИЙ ',	
*'НОМЕР ',I2/1X,9F8.3)	81
STOP	90
END	91

Оператор 10 задает описание основных и резервных массивов (A,B,WORK,IPVT,AD,BD), по которому им отводится соответствующее место в памяти. Значение NDIM=9.

Оператор 20 осуществляет ввод количества уравнений равновесия N, которое также равно девяти.

Операторы 22 и 23, совпадающие с такими же (по номеру) операторами программы 8.2, производят ввод матрицы A и вектора-столбца B СЛАУ (6.3). Данные для ввода матрицы A представляются в естественном виде по строкам.

Вложенные циклы из операторов 24-27 производят копирование матрицы A и вектора-столбца B в соответствующие резервные массивы AD и BD.

Арифметические операторы присваивания 28 и 29 соответственно задают начальное значение переменной K=0 и увеличивают значение K на единицу. Переменная K определяет номер решаемой системы уравнений: при 1-м выполнении K=1, при 2-м — K=2.

Операторы 30-47 совпадают с такими же по номеру операторами программы 8.2, где они подробно описаны.

Операторы 30 и 32 под управлением соответственно операторов 31 и 33 осуществляют контрольную печать исходных данных для проверки правильности выполненного ввода матрицы A и вектора-столбца B каждого варианта (см. распечатку (8.9)).

Оператор 40 осуществляет обращение к подпрограмме DECOMP со следующими входными фактическими параметрами: заявленный в операторе DIMENSION строчный размер NDIM=N; число уравнений равновесия N=9; в двумерном массиве A при 1-м решении системы уравнений (K=1) находится матрица A₁, при 2-м (K=2) — A₂. После окончания работы подпрограммы в

массиве A находится факторизованная матрица, нужная для дальнейшей работы подпрограммы SOLVE.

Операторы 41 и 42 осуществляют печать оценки обусловленности матрицы COND (для A_1 или A_2).

Операторы 43-46 выполняют проверку обобщенного критерия обусловленности (особенности) матрицы, при удовлетворении которого выдают на печать аварийное сообщение и прекращают выполнение программы (см. дополнение 8.1).

Оператор 47 осуществляет обращение к подпрограмме SOLVE с фактическими аргументами: NDIM=N; N=9 (поэтому два первых числа в списке заданы значениями одной переменной N); A — факторизованная матрица, полученная из DECOMP; B — на входе в подпрограмму это вектор-столбец свободных членов, на выходе — вектор решения.

Условный логический оператор IF 51 анализирует на истинность логическое выражение, заключенное в скобки. Если оно истинно ($B(1)>0$), то оператор безусловного перехода GO TO передает управление оператору PRINT с меткой 60, выполняющему печать результатов решения, после чего последует прекращение выполнения программы. Если оно ложно ($B(1)<0$) и нужно решение задания C-9 для второго случая, то начинается выполнять следующий оператор 61.

Вложенный цикл из операторов 61-65 производит копирование матрицы AD и вектора BD в соответствующие исходные массивы A и B (см. описание к примеру (7.8)).

Оператор 70 — оператор цикла DO — организует цикл по I, состоящий из одного исполнимого оператора READ, список ввода которого состоит из одного элемента первого столбца $A(I,1)$. В цикле параметр I пробегает значения от 1 до 9 ($N=9$) и при каждом изменении индекса I выполняется оператор 70, вводя каждый раз соответствующий элемент: при $I=1$ — $A(1,1)$, при $I=2$ — $A(2,1)$,..., при $I=9$ — $A(9,1)$. Таким образом оператор 70 повторяется 9 раз, выполняя ввод всех элементов 1-го столбца матрицы A_2 СЛАУ (6.4). Оператор 72 — оператор продолжения CONTINUE — не выполняет никаких действий и используется в качестве конечного оператора цикла. После его выполнения управление передается на начало цикла.

После выполнения цикла из операторов 70-72 в массиве A находится матрица A_2 СЛАУ (6.4), в векторе B — исходный вектор-столбец B_1 СЛАУ (6.3), полностью эквивалентный вектору-столбцу B_2 СЛАУ (6.4).

Оператор безусловного перехода 74 передает управление оператору 29 с меткой 15, который увеличивает значение K на единицу (теперь $K=2$). После этого происходит повторное выполнение операторов 29-51, т.е. решение СЛАУ

(6.4), в котором при правильном составлении уравнений равновесия и отсутствии ошибок ввода должно быть $B(1) > 0$.

Теперь условный логический оператор IF 51, анализируя на истинность выражение отношения $(B(1).GT.0.0)$ определит, что оно выполняется, после чего будет выполнен оператор безусловного перехода GO TO, передающий управление оператору PRINT с меткой 60 (имеющему номер 80).

Операторы 80 и 81 (эквивалентные операторам 50 и 51 программы 8.2) выполняют печать результатов решения (в данном случае при $K=2$ для 2-го случая).

Оператор 90 указывает на конец вычислений, а оператор 91 указывает на конец основной программы.

Исходные данные для работы программы 8.4 (для СЛАУ (6.3)-(6.4)) могут быть представлены в виде:

$$\begin{array}{l}
 9 \\
 2, -3, 7, 6*0 \\
 0, 1, 5*0, 1, 0 \\
 1, 0, 1, 5*0, 1 \\
 7*0, 1, 10 \\
 5*0, 1, 0, -1, 0 \\
 6*0, 1, 0, -1 \\
 6, 5*0, -2, 2*0 \\
 3*0, 1, 0, -1, 3*0 \\
 -1, 3*0, 1, 0, -1, 2*0 \\
 -20, 2*0, 30, 0, -3, -49.96, 5, 10.66 \\
 0 \\
 0 \\
 0 \\
 6 \\
 0 \\
 -1 \\
 -4 \\
 0 \\
 1
 \end{array} \tag{8.11}$$

Проверьте, что они совпадают с исходными данными для программы STAT9 (6.7), но без 1-й строки (полное совпадение может быть получено использованием в программе 8.4 дополнения 8.2). Результаты работы программы 8.4 совпадают с результатами работы программы 8.2 (без дополнений 8.2 и 8.3) и представлены на распечатке (8.9).

Заметим, что для программы 8.4 (и нижеследующей программы 8.5) целесообразны дополнения 8.2 и 8.3. Вместе с этим для программы 8.4 (и 8.5) можно сделать следующие дополнения.

Дополнение 8.7. Из-за ошибок в составлении уравнений равновесия и недостаточной внимательности при подготовке исходных данных, возможны ситуации, когда и Re и Rf получаются отрицательными. В таких случаях необходимо остановить программу после 2-го выполнения (при $K=2$) условного логического оператора IF 51. В таких случаях возникнет ситуация, когда для выполнения оператора 71 READ не будет данных (т.к. 3-го случая в нашем типовом примере (6.3)-(6.4) нет). После этого произойдет системное прерывание с выдачей соответствующего сообщения.

Чтобы предотвратить такую возможность, после оператора 51 можно поместить условный логический оператор IF 54, проверяющий истинность логической операции .AND. (см. п. 7.5.3) над двумя выражениями отношений:

```
IF ( (B(1) .LT. 0.0) .AND. (K.EQ.2) ) STOP 54
```

Результат операции .AND.(и.) имеет значение истинно только тогда, когда оба выражения отношений ($B(1)<0.0$ и $K=2$) выполняются. Только в этом случае оператор STOP останавливает работу программы.

Поэтому при $K=1$ и $B(1)<0.0$ или $K=2$ и $B(1)>0.0$ оператор 52 пропускается и не нарушает вышеописанную последовательность выполнения программы 8.4, а при $K=2$ и $B(1)<0.0$ он предотвращает возникновение аварийной ситуации, останавливая выполнение программы.

Дополнение 8.8. Обычно не просто предотвращают аварийную ситуацию, но и выдают на печать соответствующее сообщение. Это можно сделать, повторив еще раз условный логический оператор IF типа 54, но поместив вместо STOP оператор PRINT (без списка вывода), которому должен соответствовать определенный оператор FORMAT. Естественно, что эта группа операторов должна быть помещена перед оператором 54, иначе она не будет выполнена при возникновении аварийной ситуации и останове программы:

```
IF ( (B(1) .LT. 0.0) .AND. (K.EQ.2) ) PRINT 45 52
45 FORMAT (/5X, 'ВНИМАНИЕ! ОБА ЗНАЧЕНИЯ RE И RF ', 53
*' ОТРИЦАТЕЛЬНЫ !!!')
```

При $K=1$ и $B(1)<0.0$ оператор 52 также пропускается, не нарушая последовательность выполнения программы. Также пропускается он и при $K=2$, но $B(1)>0.0$, т.к. это нарушает одно из двух выражений отношений оператора 52. И только при $K=2$ и $B(1)<0.0$ сначала выдается аварийное сообщение: “ВНИМАНИЕ! ОБА ЗНАЧЕНИЯ RE И RF ОТРИЦАТЕЛЬНЫ!!” (отделяемое для большей ясности строкой пробелов из-за разделителя “/”), после чего оператор 54 останавливает выполнение программы.

Заметим, что дополнения 8.7 и 8.8 иллюстрируют необходимость продумывания возможных непредвиденных ситуаций, на которые следует предусмотреть программно реализованные ответы. Эта работа должна начинаться уже

при разработке алгоритма программы и ее уровень в значительной степени определяется полнотой и качеством проделанной работы.

Дополнение 8.9. Операторы 20-23 в программе 8.4 можно заменить использованием оператора DATA примера (8.5), который также задает начальные значения переменной N, всем элементам матрицы A1 и вектора-столбца B1 СЛАУ (6.3). Но в отличие от программы 8.1, при использовании оператора DATA (8.5) (или операторов явного описания типа примеров (8.6) и (8.7)) в программе 8.4 ее повторять не следует. Нужно в исходных данных (8.11) оставить только 1-й столбец матрицы (6.4).

Дополнение 8.10. Вместо операторов 10 и 20-23 в программе 8.4 возможно использование операторов явного описания типа. Однако в отличие от примеров (8.6) и (8.7), выполняющих такую замену для программы 8.1, оператор 20 дополняется (по сравнению с оператором (8.7)) описанием резервных массивов AD и BD:

```

INTEGER N/9/, IPVT(9)                                10
REAL A(9,9)/2.,0.,1.,3*0.,6.,0.,-1.,-3.,1.,7*0.,7.,0.,
1 1.,13*0.,1.,9*0.,1.,4*0.,1.,2*0.,-1.,6*0.,1.,-2.,0.,-1.,0.,
2 1.,0.,1.,-1.,6*0.,1.,10.,0.,-1.,3*0.0/, WORK(9),AD(9,9),
3 B(9)/-20.,0.,0.,30.,0.,-3.,-49.96,5.,10.66/,BD(9)    20

```

Исходными данными для работы программы 8.4 с учетом дополнений 8.9 и 8.10 будут только 9 последних строк примера (8.11).

Отметим, что при полном вводе матриц с использованием операторов DATA и REAL, элементы двумерных массивов должны перечисляться по столбцам, что неудобно. Предпочтительнее использование оператора DATA для заданий только конкретных ненулевых значений в программе 8.5 (см. дополнение 8.11).

8.3.5. Организация ввода только ненулевых элементов с учетом особенностей задания С-9

Ввод только ненулевых элементов с учетом особенностей задания С-9 [21, с. 51-59] (СЛАУ (6.3)-(6.4)) можно представить следующим образом:

- ввод исходных данных должен быть организован только для ненулевых элементов матрицы A1 и B1, что выполняет программа 8.3. Поэтому операторы с совпадающими номерами в программах 8.3 и 8.5 полностью эквивалентны и повторно не описываются;
- учет особенностей задания С-9 выполняется по образцу программы 8.4. Поэтому операторы, выполняющие в программах 8.4 и 8.5 одинаковые действия, имеют в программах 8.4 и 8.5 совпадающие номера. Чтобы не было

путаницы с какой программой сравнивать, эти совпадающие с программой 8.4 номера в программе 8.5 состоят из самого совпадающего номера оператора (в программе 8.4), точки и цифры 4. Например, совпадающий с оператором 24 в программе 8.4 соответствующий оператор в программе 8.5 имеет номер 24.4. Они также повторно не описываются.

Структура основных блоков программы 8.5 также совпадает с программой 8.4, поэтому 1-я цифра номера оператора для удобства ориентировки, указывает на соответствующий блок 1) — 9), описанный в п. 8.3.4.

Единственная особенность программы 8.5 состоит во вводе только отличающихся ненулевых элементов 1-го столбца матрицы A_2 , для чего производится их предварительное обнуление (после выполнения решения для 1-го случая). Эти действия выполняют новые четыре оператора, которые в программе 8.5 после номера имеют букву N: 70N — 73N. Они описаны в пояснениях к нижеследующей программе 8.5:

```

С      П Р О Г Р А М М А 8.5
      DIMENSION A(9,9),B(9),WORK(9),IPVT(9),AD(9,9),BD(9) 10.4
      READ *,N 20.4
      DO 6 I=1,N 22
        B(I)=0. 23
        DO 6 J=1,N 24
          6 A(I,J)=0. 25
      READ *,KOR 26
      DO 7 M=1,KOR 27
          7 READ *,KNNEKO,(I,J,A(I,J),I1=1,KNNEKO) 28
      READ *,KNNEKO,(I,B(I),I1=1,KNNEKO) 29
      DO 10 I=1,N 24.4
        BD(I)=B(I) 25.4
        DO 10 J=1,N 26.4
          10 AD(I,J)=A(I,J) 27.4
      K=0 28.4
      15 K=K+1 29.4
      PRINT 20,K,((A(I,J),J=1,N),I=1,N) 30
      20 FORMAT(/5X,'ВХОДНАЯ МАТРИЦА ЛЕВОЙ ЧАСТИ A',I2/ 31
      *(1X,9F8.3))
      PRINT 25,K,(B(I),I=1,N) 32
      25 FORMAT(5X,'ВЕКТОР ПРАВЫХ ЧАСТЕЙ B',I2/1X,9F8.3) 33
      CALL DECOMP(N,N,A,COND,IPVT,WORK) 40
      PRINT 30,K,COND 41
      30 FORMAT(5X,'ОЦЕНКА ОБУСЛОВЛЕННОСТИ МАТРИЦЫ A',I2,1X,
      *'COND=',G12.5) 42
      CONDP1=COND+1 43
      IF (CONDP1.EQ.COND) PRINT 40,K 44

```

40	FORMAT(5X, 'ВНИМАНИЕ! - ОБНАРУЖЕНА ВЫРОЖДЕННОСТЬ ', 45	
	* ' (ОСОБЕННОСТЬ) МАТРИЦЫ A', I2, ' !!!')	
	IF (CONDP1.EQ.COND) STOP	46
	CALL SOLVE (N, N, A, B, IPVТ)	47
	IF (B(1).GT.0.0) GO TO 60	51.4
	DO 50 I=1, N	61.4
	B(I)=BD(I)	62.4
	DO 50 J=1, N	63.4
	A(I, J)=AD(I, J)	64.4
50	CONTINUE	65.4
	DO 55 I=1, N	70N
	A(I, 1)=0.	71N
55	CONTINUE	72N
	READ *, KNNEKO, (I, J, A(I, J), I1=1, KNNEKO)	73N
	GO TO 15	74.4
60	PRINT 65, K, (B(I), I=1, N)	80.4
65	FORMAT(15X, 'РЕЗУЛЬТАТЫ РЕШЕНИЯ СИСТЕМЫ УРАВНЕНИЙ ',	
	* 'НОМЕР ', I2/1X, 9F8.3)	81.4
	STOP	90.4
	END	91.4

Операторы 10.4-20.4, 24.4-29.4, 51.4-65.4 и 74.4-91.4 эквивалентны соответствующим операторам программы 8.4, а операторы 22-29 и 30-47 — программе 8.3, где они и описаны.

Оператор 70N — оператор цикла DO — организует цикл по I, состоящий из одного исполнимого арифметического оператора присваивания $A(I,1)=0$. В цикле параметр I пробегает значения от 1 до 9 ($N=9$) и при каждом изменении индекса I выполняется оператор 71N, присваивая каждый раз соответствующему элементу нулевое значение: при $I=1$ — $A(1,1)=0$., при $I=2$ — $A(2,1)=0$., ..., при $I=9$ — $A(9,1)=0$. Таким образом оператор 71N повторяется 9 раз, выполняя обнуление всех элементов 1-го столбца матрицы A2.

Оператор продолжения CONTINUE 72N — не выполняет никаких действий и используется в качестве конечного оператора цикла.

Оператор 73N сначала вводит количество отличающихся ненулевых элементов KNNEKO 1-го столбца матрицы A2, которое является верхней границей изменения параметра неявного (вложенного в оператор READ) цикла по I1. Затем начинает выполняться неявный цикл DO, заключенный в скобки. В нем сначала каждый раз будет прочитан номер строки I и столбца J вводимого элемента, а затем этот указанный номер (I,J) заполнится следующим за ним в строке данных нужным значением $A(I,J)$. Таких повторений считывания указанных номеров (I,J) и их заполнения следующим в строке данных соответствующим значением матрицы $A(I,J)$ будет сделано заданное число KNNEKO раз.

После выполнения операторов 70N-73N в массивах А и В находятся матрица А2 и вектор-столбец В2 для 2-го случая решения задания С-9 — СЛАУ (6.4). Исходные данные для работы программы 8.5 (для СЛАУ (6.3)-(6.4)) могут быть представлены в виде:

$$\begin{array}{l}
 9 \\
 3 \\
 8, 1, 1, 2, 1, 2, -3, 1, 3, 7, 2, 2, 1, 2, 8, 1, 3, 1, 1, 3, 3, 1, 3, 9, 1 \\
 6, 4, 8, 1, 4, 9, 10, 5, 6, 1, 5, 8, -1, 6, 7, 1, 6, 9, -1 \\
 7, 7, 1, 6, 7, 7, -2, 8, 4, 1, 8, 6, -1, 9, 1, -1, 9, 5, 1, 9, 7, -1 \\
 6, 1, -20, 4, 30, 6, -3, 7, -49.96, 8, 5, 9, 10.66 \\
 4, 4, 1, 6, 6, 1, -1, 7, 1, -4, 9, 1, 1
 \end{array} \quad (8.12)$$

Они совпадают с исходными данными (6.11) для программы STAT9N, но без 1-й строки. Полное совпадение может быть получено использованием в программе 8.5 дополнения 8.2. Рекомендуем также выполнить дополнения 8.3, 8.7 и 8.8, которые также полностью применимы и к программе 8.5. Распечатка программы 8.5 (в основном варианте) также совпадает с программой 8.2 и представлена во фрагменте (8.9).

Дополнение 8.11. Операторы 20.4, 22-29 в программе 8.5 можно заменить использованием оператора DATA. В отличие от примера (8.5), в котором задаются начальные значения всем элементам матрицы А1 и вектора-столбца В1, в программе 8.5 нужно задать только их ненулевые элементы. Все нулевые значения задаются предварительно (например, также в операторе DATA):

```

DATA N/9/, A/81*0./, B/9*0./                                20
DATA A(1,1)/2./, A(1,2)/-3./, A(1,3)/7./, A(2,2)/1./,      29
2 A(2,8)/1./, A(3,1)/1./, A(3,3)/1./, A(3,9)/1./,
3 A(4,8)/1./, A(4,9)/10./, A(5,6)/1./, A(5,8)/-1./,
4 A(6,7)/1./, A(6,9)/-1./, A(7,1)/6./, A(7,7)/-2./,
5 A(8,4)/1./, A(8,6)/-1./, A(9,1)/-1./, A(9,5)/1./,
6 A(9,7)/-1./,
7 B(1)/-20./, B(4)/30./, B(6)/-3./, B(7)/-49.96/,
8 B(8)/5./, B(9)/10.66/

```

Исходными данными в этом случае будут только ненулевые элементы 1-го столбца СЛАУ (6.4):

$$4, 4, 1, 6, 6, 1, -1, 7, 1, -4, 9, 1, 1$$

Можно эти же значения задать арифметическими операторами присваивания непосредственно в самой программе. Тогда файла с исходными данными не будет и в программе 8.5 нужно вместо оператора 73N вставить 4 нижеследующих оператора:

$$\begin{array}{l}
 A(4,1)=6. \\
 A(6,1)=-1.
 \end{array}$$

$$A(7, 1) = -4.$$
$$A(9, 1) = 1.$$

Отметим, что эти четыре значения нельзя задать в этом месте с помощью оператора DATA (в середине выполнения программы после использования массива A). Переменные и элементы массивов получают начальные значения с помощью оператора DATA перед выполнением программы в момент ее загрузки независимо от того, в каком месте программы записан оператор DATA (см. п. 7.7.1).

По этой же причине предварительное обнуление массивов A и B (перед оператором 29 DATA дополнения 8.11) нельзя выполнить с помощью выполняемых операторов 22-25 программы 8.5. Но это можно сделать, если вместо оператора 29 DATA в дополнении 8.11 задать ненулевые элементы матрицы A1 с помощью арифметических операторов присваивания.

8.4. Компиляция (трансляция) и редактирование фортран-программ на ЭВМ

ЭВМ работает правильно только при совершенно точной программе и данных. С трудностями доводки первоначального решения до правильного только при подготовке данных вы уже познакомились на практике при выполнении заданий по готовым программам (см. пп. 5 и 6). Эти трудности, конечно, возрастают при создании и доведении до готовности самой программы.

Если программа написана с ошибками, неизбежен ошибочный результат. Человеку свойственно ошибаться... Процесс “вылавливания” ошибок, т.е. доводки первоначальной программы до правильной, носит название тестирования и составляет обычно 70-80% времени создания программы. И только 20-30% - это написание исходного текста.

Обработка программы на Фортране начинается с ввода ее исходного текста (в файл типа .for) с помощью любого текстового редактора. Для получения выполняемой программы (с расширением .exe) нужно произвести ее компиляцию и редактирование. Вызов компилятора (транслятора) производится по команде FL, в которой указываются режимы компиляции, начинающиеся с символа “/” (наклонная черта), и имя обрабатываемого файла. После ее окончания создается объектный файл (тип .obj). Сообщения об ошибках в программе выводятся на экран и помещаются также в отдельный файл (тип .lst).

Если в программе отсутствуют серьезные ошибки, то с помощью команды LINK вызывается Редактор связей для создания выполняемого файла (тип .exe). В процессе редактирования к основному объектному файлу присоединяются необходимые программы из библиотеки Фортрана, а также объектные файлы, дополнительно указанные в команде.

8.4.1. Команда FL

Команда FL имеет следующий формат:

```
FL [режим] [имя-файла] [режим] [имя-файла]...
    [/link [библиотеки] [режимы-редактирования]] (8.12)
```

где режим - один из режимов компилятора (см. [28, с. 142-172]);

имя-файла - идентификатор исходного файла;

библиотеки - список имен библиотечных файлов, явно заданных для редактирования;

режимы-редактирования - режимы Редактора связей (см. п. 7.1.2 [28, с. 177-180]).

Если режим /link используется в команде FL, то компилятор автоматически вызывает Редактор связей для создания выполняемого файла, если нет - то потребуется отдельный шаг редактирования с помощью команды LINK (см. п. 7 [28, с. 175-191]). Команды FL и LINK вследствие своей сложности помещаются обычно в командные файлы (с типом .bat).

Отметим, что режимы могут быть записаны в любом месте команды FL. Они относятся ко всем файлам, имена которых следуют за ними в команде. Не смотря на то, что имена файлов и команд могут задаваться как строчными, так и прописными буквами, режимы в команде FL следует задавать так, как они указаны в пособии, например, /Fpi.

При указании режима /HELP (или /help) выдается справочная информация о команде FL. Например, по команде FL /HELP она выводится на экран, а по FL /HELP >PRN перенаправляется на печатающее устройство.

Пусть исполняемые файлы компилятора Фортрана помещены в подкаталоге C:\F5\BIN, библиотечные - в C:\F5\LIB, графические - в C:\F5\INCLUDE, а временно создаваемые файлы в подкаталоге C:\F5\TMP. Создание этих подкаталогов и помещение в них файлов выполняется автоматически после указания их имен в ответ на соответствующие вопросы при инсталляции Фортрана. Будем также считать, что к ним указан путь в команде PATH файла autoexec.bat (см. (2.4)). В подкаталоге C:\F5\BIN создадим командный файл FFL.BAT для запуска компилятора Фортрана (что позволит вызывать его из любого текущего подкаталога):

```
@ECHO OFF                                10
PATH C:\F5\BIN;                             20
SET LIB=C:\F5\LIB                           30
SET INCLUDE=C:\F5\INCLUDE                   40      (8.13)
SET TMP=C:\F5\TMP                           50
FL /Fs /c /Fpi %1.FOR                       60
```

Команда 10 предназначена для того, чтобы последующие не выводились на экран.

Команда 20 указывает путь в подкаталог, в котором находятся исполняемые файлы компилятора.

Команды 30-50 задают переменные окружения, указывающие расположение подкаталогов, в которых находятся библиотечные, графические и временно создаваемые файлы.

Команда 60 производит вызов программы компилятора Фортрана. Режим /Fs используется для вывода распечатки исходного текста компилируемых модулей программ (с типом .LST), которая полезна при их отладке.

По режиму /c выполняется только компиляция исходных файлов, заданных в команде. По умолчанию объектному файлу (с типом .OBJ) присваивается такое же основное имя, как у соответствующего исходного файла. Если режим /c в команде FL отсутствует, то используется по умолчанию режим /link, полученный объектный файл редактируется и создается выполняемый файл (с типом .EXE). Это удобно при работе с программами, все модули которых располагаются в одном файле, но является ненужным при компиляции подпрограмм, находящихся в отдельных файлах.

Согласование режима /FP, осуществляющего управление операциями с плавающей точкой, и используемой библиотеки Фортрана является особенно важным (см. [28, с. 158-162] и [24, с. 171-174]). Режим /FPi полезен, когда нет уверенности в наличии математического сопроцессора. Если он присутствует, то программа его использует. При отсутствии сопроцессора режим /FPi обеспечивает получение наиболее высокой точности выполнения операций с плавающей точкой. При этом используются библиотеки Фортрана LLIBFORE.LIB или MLLIBFORE.LIB (см. команды 70 фрагментов (8.19) и (8.21)).

Имя компилируемой программы, указываемое при запуске после имени командного файла (FFL в нашем случае), является заменяемым параметром. Оно будет подставлено при выполнении команд вместо символа %1. Тип файла (.FOR) уже указан в процедуре (8.13), поэтому повторно его указывать не нужно. Например, для компиляции программы 8.2, находящейся в файле F8-2.FOR, следует из соответствующего текущего подкаталога ввести команду:

```
FFL F8-2 (8.14)
```

Для компиляции подпрограмм DECOMP и SOLVE, находящихся в одноименных файлах с расширением .FOR, следует соответственно ввести команды:

```
FFL DECOMP (8.15)
```

```
FFL SOLVE (8.16)
```

Всего может быть использовано до девяти заменяемых параметров, обозначаемых символами %1 - %9. Цифра после знака процента определяет

их порядковый номер после имени командного файла. Лишние символы, которым не соответствуют указанные параметры, замещаются пустыми строками. Поэтому, если в процедуре FFL (8.13) команду 60 представить в виде:

```
FFL /Fs /c /Fpi %1.FOR %2.FOR %3.FOR %4.FOR %5.FOR 60
```

то с учетом внесенных изменений могут быть выполнены как команды (8.14) - (8.16), так и команда:

```
FFL F8-2 DECOMP SOLVE (8.17)
```

В обоих случаях будут созданы по три одноименных файла (F8-2, DECOMP и SOLVE) с типами .LST и .OBJ.

Отметим, что компилятор в своих сообщениях указывает только на синтаксические ошибки программы. После их устранения (или внесения любых изменений) перед отправлением программы на счет этапы компиляции и редактирования следует повторить.

Поиск логических ошибок в программе представляет собой значительно более трудоемкую задачу. Для ее облегчения возможно использование различных отладчиков (Code View [28, с. 192-257], WATFOR-77 [7, с. 125-135]). С их помощью можно просмотреть текст отлаживаемой программы, числовые значения переменных, распределение внешних устройств, начать выполнение программы, начать пошаговое вычисление, узнать точку останова программы и числовые значения переменных.

Применение блочного подхода позволяет при работе с пособием избежать логических ошибок при составлении программ или использовании дополнений.

8.4.2. Редактор связей LINK

Вызов Редактора связей производится по команде LINK, которая имеет следующий формат:

```
LINK список-объектных-файлов [, [выполняемый-файл] [, (8.18)
[файл-распечатки] [, [список-библиотек] ]]] [режимы] [;]
```

где список-объектных-файлов - имена файлов с типом .OBJ, участвующих в редактировании, разделенных пробелом или знаком "+";

выполняемый-файл - имя создаваемого выполняемого файла. Если оно не указано, то по умолчанию будет совпадать с именем первого объектного файла и иметь расширение .EXE;

файл-распечатки - имя файла для вывода распечатки редактирования. В нем содержатся сведения о размещении данных, включенных в загрузочный модуль, в памяти ЭВМ. По умолчанию он не создается;

список-библиотек - имена библиотек объектных файлов (также разделенных пробелом или знаком "+"), в которых будет выполняться

поиск требуемых модулей. По умолчанию предполагается, что стандартные библиотеки Фортрана всегда добавляются в этот список;

режимы - список режимов команды LINK.

Чтобы для любого параметра выбрать значение по умолчанию, необходимо вместо него указать запятую. Если в конце команды указать символ “;”, то все оставшиеся параметры принимают значения по умолчанию (см. п. 7.1.2 [28, с. 177-180]). Поэтому нижеследующие команды LINK эквивалентны, только во второй из них конкретно указана используемая библиотека Фортрана, соответствующая выбранному режиму /FPI в команде FL процедуры (8.13):

```
LINK %1 %2 %3 %4, , , ; 70
LINK %1 %2 %3 %4, %1.exe, , C:\F5\LIB\LLIBFORE.LIB; 70
```

Команда 70 в любой из форм может быть помещена вместе с командами 10 - 50 в отдельный командный файл (например, с именем GO.BAT):

```
@ECHO OFF 10
PATH C:\F5\BIN; 20
SET LIB=C:\F5\LIB 30
SET INCLUDE=C:\F5\INCLUDE 40 (8.19)
SET TMP=C:\F5\TMP 50
LINK %1 %2 %3 %4, %1.exe, , C:\F5\LIB\LLIBFORE.LIB; 70
```

Тогда он будет приспособлен только для редактирования и создания выполняемого файла. Для этого в командной строке для рассматриваемого примера нужно будет ввести (после выполнения команд (8.14) - (8.16)):

```
GO F8-2 DECOMP SOLVE (8.20)
```

Команду LINK можно также добавить к командам 10 - 60 фрагмента (8.13), присвоив новому командному файлу, например, имя FU.BAT:

```
@ECHO OFF 10
PATH C:\F5\BIN; 20
SET LIB=C:\F5\LIB 30
SET INCLUDE=C:\F5\INCLUDE 40 (8.21)
SET TMP=C:\F5\TMP 50
FL /Fs /c /Fpi %1.FOR 60
LINK %1 %2 %3 %4, %1.exe, , C:\F5\LIB\LLIBFORE.LIB; 70
```

Тогда он будет приспособлен для компиляции, редактирования и создания выполняемого файла. Для этого в командной строке для рассматриваемого примера нужно будет ввести (после выполнения команд (8.15) - (8.16)):

```
FU F8-2 DECOMP SOLVE (8.22)
```

По этой команде сначала будет произведена компиляция файла F8-2.FOR, в результате чего будет создан файл F8-2.OBJ. Если в нем отсутствуют серьезные ошибки, то с помощью команды LINK (как и по команде (8.20)),

вызывается Редактор связей для создания выполняемого файла F8-2.EXE. В процессе редактирования к основному объектному файлу F8-2.OBJ присоединяются необходимые программы из библиотеки Фортрана, а также объектные файлы DECOMP.OBJ и SOLVE.OBJ, дополнительно указанные в команде.

Программы следующей главы используют стандартные подпрограммы (ARRAY и SIMQ), которые обычно находятся на ПЭВМ в виде одноименных объектных файлов (с расширением .OBJ). Поэтому для компиляции, редактирования и создания выполняемого файла для работы программы 9.2, находящейся, например, в файле F9-2.FOR, нужно из этого текущего подкаталога только ввести команду:

```
FU F9-2 ARRAY SIMQ (8.23)
```

Так как во всех программах пп. 8 и 9 используются одинаковые подпрограммы (DECOMP, SOLVE и ARRAY, SIMQ), то команду LINK процедуры (8.21) можно видоизменить для каждого конкретного случая:

```
LINK %1 DECOMP SOLVE,%1.exe,,C:\F5\LIB\LLIBFORE.LIB; 70
LINK %1 ARRAY SIMQ,%1.exe,,C:\F5\LIB\LLIBFORE.LIB; 70
```

Поместим вместе с командами 10 - 60 фрагмента (8.21) первую из этих команд, например, в файл FSF.BAT, а вторую - в FSP.BAT (конечно, как и все командные файлы, в подкаталоге C:\F5\BIN). Тогда из любого текущего подкаталога можно будет выполнить компиляцию, редактирование и создание выполняемого файла для работы любой программы пп. 8 и 9, указав при вызове соответствующей процедуры только нужное имя файла (с расширением FOR). Для примеров (8.22) и (8.23) это будет соответственно иметь вид:

```
FSF F8-2 (8.24)
```

```
FSP F9-2 (8.25)
```

Компиляция и редактирование всех остальных программ, описываемых в пп. 13 - 15 и 19 - 22, выполняется аналогичным образом. Во всех случаях можно воспользоваться универсальными командными файлами (8.13), (8.19) и (8.21), или видоизменить команду LINK процедуры (8.21), указав в ней имена объектных файлов наиболее часто используемых подпрограмм.

При запуске выполняемого файла (с расширением .EXE) во всех случаях следует учитывать рекомендации п. 7.6.3: предварительно использовать в программах описанный там же оператор open или применять перенаправление потоков ввода-вывода (см. п. 2.2.3). В последнем случае нужно указать соответствующие имена файлов, из которых следует брать исходные данные (обычно с расширением .DAT) и куда помещать результаты расчета (обычно .LIS). Для выполняемых файлов F8-2.EXE и F9-2.EXE это может иметь вид (так как исходные данные (8.8) для работы программ 8.2 и 9.2 одинаковые):

```
F8-2.EXE < F8-2.DAT > F8-2.LIS (8.26)
```

```
F9-2.EXE < F8-2.DAT > F9-2.LIS (8.27)
```

ГЛАВА 9. ИСПОЛЬЗОВАНИЕ СТАНДАРТНЫХ ПОДПРОГРАММ

Рассмотрим составление программ для решения задач статики, приведенных в п. 8.3, с использованием стандартных подпрограмм.

9.1. Пакет научных подпрограмм ПНП-SSP

Пакет научных подпрограмм ПНП-SSP (Scientific Subroutine Package), называемый также библиотекой стандартных подпрограмм (БСП) или библиотекой института математики АН РБ (БИМ), содержит полный набор всех основных вычислительных подпрограмм по различным разделам математики. Это несколько сотен подпрограмм, реализующих важнейшие математические методы [12,19]. Пакет может быть использован для решения многих задач в промышленности, науке и инженерных расчетах, поэтому весьма важно научиться использовать содержащиеся в нем подпрограммы. Основная доля стандартных подпрограмм является подпрограммами общего вида (типа SUBROUTINE). Для каждой подпрограммы имеется описание, включающее краткое изложение используемого метода, характеристику аргументов подпрограммы и указания к ее использованию.

Каждая стандартная подпрограмма имеет свое имя, по которому она может быть вызвана для выполнения. Обращение к стандартным подпрограммам из библиотеки научных подпрограмм осуществляется посредством оператора CALL (см. п. 7.11) с указанием имени подпрограммы и списка ее фактических аргументов. Он должен быть согласован со списком формальных аргументов, приведенных в описании соответствующей подпрограммы.

9.2. Использование подпрограмм SIMQ и ARRAY

9.2.1. Понятие о двумерной и векторной формах памяти

Существуют две формы размещения матриц в памяти ЭВМ в зависимости от их описания в операторе DIMENSION.

Если матрица A , например размером 6×6 , описывается с двойной индексацией (DIMENSION A(6,6)), то используется двумерная форма памяти (см. рис. 9.1). Таким образом, подпрограммы DECOMP и SOLVE (см. п. 8.2) используют двумерную форму памяти.

Если эта же матрица описана в виде одномерного массива (DIMENSION A(36)) то используется векторная форма памяти данных (см. рис. 9.2), применяемая всеми стандартными подпрограммами.

Двумерная и векторная формы размещения данных в памяти совпадают, если число строк и столбцов N данной конкретной матрицы будет таким же, каким оно указано предварительно в операторе описания массивов DIMENSION (т.е. заявленный строчный размер матрицы $NDIM = N$, см. рис. 9.3).

Если используемая матрица меньше отведенной для нее в операторе DIMENSION двумерной области, то две формы памяти не совпадают. Это бывает при составлении универсальной программы для решения, например, разных заданий по статике с различным числом уравнений равновесия N . Тогда в операторе DIMENSION задается максимальное значение числа N (обозначаемое нами ранее как $NDIM$), могущее встретиться в различных задачах. Используемая конкретная матрица обычно оказывается меньше отведенной для нее двумерной области (см. дополнение 8.6).

Рассмотрим распределение данных в памяти, когда операции производятся над матрицей, например $A(3,3)$ в области размером 6×6 ($N=3$, $NDIM=6$), т.е. в задаче число уравнений $N=3$, а в основной программе задано:

DIMENSION A(6,6)

Тогда 9 элементов, подготовленных для ввода в естественной форме по строкам, при обычном считывании на Фортране (без использования формы неявного цикла DO с “перевернутыми” индексами), будут расположены в памяти следующим образом:

		СТОЛБЕЦ					
		1	2	3	4	5	6
С	1	(1)=A(1,1)	(7)=A(1,2)	(13)=A(1,3)	(19)...	(25)...	(31)...
	2	(2)=A(2,1)	(8)=A(2,2)	(14)=A(2,3)	(20)...	(26)...	(32)...
Т	3	(3)=A(3,1)	(9)=A(3,2)	(15)=A(3,3)	(21)...	(27)...	(33)...
	4	(4)...	(10)...	(16)...	(22)...	(28)...	(34)...
Р	5	(5)...	(11)...	(17)...	(23)...	(29)...	(35)...
	6	(6)...	(12)...	(18)...	(24)...	(30)...	(36)...
О							
К							
А							

Рис. 9.1. Размещение в памяти с двойной индексацией — двумерная форма памяти

В Фортране двумерный массив данных располагается по столбцам, длина которых определена в операторе DIMENSION, в виде последовательной

области памяти одномерного массива. Его общая длина равна произведению числа строк и столбцов в операторе DIMENSION. В рассматриваемом примере длина столбца состоит из 6 элементов и общая длина отведенного под матрицу A одномерного массива равна 36-ти.

При двумерной памяти данных последовательная область памяти из 36-ти элементов будет содержать элементы матрицы A в следующем порядке: $A(1,1)$, $A(2,1)$, $A(3,1)$, пустая область для 3-х возможных элементов строк 4-6 1-го столбца, $A(1,2)$, $A(2,2)$, $A(3,2)$, пустая область для 3-х элементов строк 4-6 2-го столбца, $A(1,3)$, $A(2,3)$, $A(3,3)$, пустая область для 3-х элементов строк 4-6 3-го столбца и для 18-ти элементов строк 1-6 для 4-6-го столбцов (см. рис. 9.1).

Если эта же матрица A из 9-ти элементов (представленная в виде $A(9)$) располагается в той же области размером 6×6 , но в основной программе задано:

```
DIMENSION A(36)
```

то эти же 9 элементов разместятся по столбцам (при использовании того же оператора ввода) в векторной форме памяти в виде:

		СТОЛБЕЦ					
		1	2	3	4	5	6
С	1	(1)= $A(1,1)$	(7)= $A(1,3)$	(13)...	(19)...	(25)...	(31)...
	2	(2)= $A(2,1)$	(8)= $A(2,3)$	(14)...	(20)...	(26)...	(32)...
Р	3	(3)= $A(3,1)$	(9)= $A(3,3)$	(15)...	(21)...	(27)...	(33)...
	4	(4)= $A(1,2)$	(10)...	(16)...	(22)...	(28)...	(34)...
К	5	(5)= $A(2,2)$	(11)...	(17)...	(23)...	(29)...	(35)...
	6	(6)= $A(3,2)$	(12)...	(18)...	(24)...	(30)...	(36)...

Рис. 9.2. Векторная форма памяти

При векторной памяти данных эта же последовательная область памяти из 36-ти элементов будет содержать элементы матрицы A в следующем порядке: $A(1,1)$, $A(2,1)$, $A(3,1)$, $A(1,2)$, $A(2,2)$, $A(3,2)$, $A(1,3)$, $A(2,3)$, $A(3,3)$, пустая область для 3-х возможных элементов строк 4-6 2-го столбца и для 24-х элементов строк 1-6 для 3-6-го столбцов (см. рис. 9.2).

В общем случае при разных формах памяти в одинаковых ячейках памяти, определяемых номером в скобках, находятся разные элементы матрицы. Например, в 7-й ячейке рис. 9.1 — $A(1,2)$, рис. 9.2 — $A(1,3)$; в 14-й — $A(2,3)$ и пусто соответственно.

Если число строк и столбцов матрицы в обращении к подпрограмме в операторе CALL будет таким же, каким оно указано в операторе описания массива

вов (DIMENSION A(3,3) или A(9) соответственно для 1-го и 2-го случаев), то двумерная и векторная формы размещения данных в памяти совпадут. В обеих формах памяти в одних и тех же ячейках будут находиться одни и те же элементы матрицы, при этом оба рисунка 9.1 и 9.2 совпадут:

		СТОЛБЕЦ		
		1	2	3
СТРОКА	1	(1)=A(1,1)	(4)=A(1,2)	(7)=A(1,3)
	2	(2)=A(2,1)	(5)=A(2,2)	(8)=A(2,3)
	3	(3)=A(3,1)	(6)=A(3,2)	(9)=A(3,3)

Рис. 9.3. Совпадение двумерной и векторной форм памяти при $NDIM = N$.

Такое совпадение использовано нами для простоты во всех программах, в которых не применяется дополнение 8.6.

При обращении к подпрограммам пакета ПНП-SSP следует иметь в виду, что для любых матричных операций все они используют векторную память для размещения по столбцам. Поэтому все элементы любого двумерного массива должны быть подготовлены для ввода данных по столбцам (см. пример (8.4)) и описаны в операторе DIMENSION в виде одномерного массива (для SLAY (6.3)—(6.4) DIMENSION A(81)).

Эта особенность стандартных подпрограмм при работе с матрицами доставляет большое неудобство, поэтому в пакете имеется подпрограмма ARRAY, которая служит для перехода от одной формы памяти к другой (см. п. 9.2.2).

9.2.2. Описание подпрограмм SIMQ и ARRAY

Подпрограмма SIMQ предназначена для решения системы линейных уравнений вида $(A)(X)=B$ методом Гаусса. Она выполняет, в отличие от подпрограмм DECOMP и SOLVE, обе части гауссова исключения (см. п. 8.1) и этап прямого хода, и обратную подстановку:

```
SUBROUTINE SIMQ(A, B, N, IER)
```

Ее формальными параметрами являются: A — матрица коэффициентов порядка (NSN), расположенная для ввода по столбцам и описанная в операторе DIMENSION в виде одномерного массива; B — вектор-столбец свободных членов системы длины N; N — число уравнений в задаче; IER — код ошибки решения: при IER=0 — результатам решения можно доверять; при IER=1 — обнаружена вырожденность (особенность) матрицы, вследствие чего полученные результаты недостоверны.

После окончания работы подпрограммы SIMQ, на месте вектора-столбца В располагается вектор решения $X \text{ SLAU } (A)(X)=B$, являющийся выходной величиной.

Подпрограмма ARRAY из пакета ПНП-SSP служит для преобразования массива из двумерного в одномерный или наоборот. Эта подпрограмма позволяет обращаться из основной программы, работающей с двумерными массивами, к любым матричным подпрограммам из пакета ПНП-SSP (стандартным), работающим с массивами данных в векторном виде:

```
SUBROUTINE ARRAY (MODE, NI, NJ, NDIMI, NDIMJ, AS, A)
```

Ее формальными параметрами являются: MODE — код режима работы подпрограммы (при MODE=2 — происходит преобразование матрицы из двумерной формы памяти в векторную (что требуется в нашем случае) и при MODE=1 — наоборот); NI и NJ — число строк и столбцов в матрице данных; NDIMI и NDIMJ — число строк и столбцов, определенных для матрицы A в операторе DIMENSION; AS — вектор размерности NINJ=NISNJ, содержащий элементы матрицы данных A порядка NISNJ (при выходе — если MODE=2, как в нашем случае, или при входе — если MODE=1); A — матрица размерности NDIMI*NDIMJ, содержащая матрицу данных размера NISNJ в первых NI строках и NJ столбцах (при MODE=2 матрица A является входным параметром, при MODE=1 — выходным).

Для преобразования матрицы из двумерной формы памяти в одномерную нужно при обращении к подпрограмме ARRAY задать значение MODE=2, при этом входными величинами будут: NI=NJ=N, NDIMI=NDIMJ=NDIM, матрица A. После окончания работы подпрограммы ARRAY в одномерном массиве AS будут расположены в векторной форме памяти данные элементы матрицы A. Теперь при обращении к подпрограмме SIMQ входной матрицей вместо A в общем случае будет являться одномерный массив AS. Заметим, что в случае необходимости для экономии памяти вектор AS может быть расположен на месте матрицы A.

9.2.3. Правила использования стандартных подпрограмм

Все подпрограммы пакета вызываются посредством стандартного оператора Фортрана CALL. Оператор CALL передает управление подпрограмме и заменяет формальные параметры, указанные в списке формальных переменных после имени подпрограммы в ее описании, значением соответствующих фактических аргументов, которые указаны в операторе CALL.

При использовании стандартных подпрограмм придерживаются всех обычных правил Фортрана, касающихся подпрограмм и соответствия формальных и фактических параметров (см. п. 7.11).

Фактические параметры в операторе CALL должны согласовываться с соответствующими аргументами в подпрограмме в порядке, количестве и типе.

Проще всего выполнить указанное правило при использовании стандартных подпрограмм можно, используя в качестве фактических параметров в операторе CALL символические имена переменных, аналогичные указанным в списке формальных параметров при описании этих подпрограмм.

Например, если используется подпрограмма SIMQ, то для решения двух вариантов задания С-9 проще всего обратиться к подпрограмме SIMQ дважды:

```
CALL SIMQ (A1, B1, N, IER1)
CALL SIMQ (A2, B2, N, IER2)
```

Аналогичные символические имена наших фактических параметров A1 и B1, A2 и B2 должны соответствовать по типу формальным параметрам A и B, т.е. A1 и A2 должны быть матрицами коэффициентов порядка (NSN), а B1 и B2 векторами свободных членов длины N. Они должны быть описаны оператором DIMENSION как массивы (для задания С-9 N=9) и предварительно введены в память ЭВМ:

```
DIMENSION A1 (81), B1 (9), A2 (81), B2 (9)
```

Это нужно сделать потому, что все подпрограммы из пакета ППП-SSP чисто вычислительные и не содержат никаких ссылок на устройства ввода-вывода. Поэтому в своей программе нужно предусмотреть ввод-вывод и другие операции, необходимые для полного решения своего варианта задания.

Числовая величина N в качестве соответствующего фактического параметра может быть передана в подпрограмму как имя переменной, которой предварительно в программе задается любым возможным способом нужное значение, например:

```
N=9
CALL SIMQ (A1, B1, N, IER1)
```

или как значение числовой величины на соответствующем месте непосредственно в списке фактических параметров, например:

```
CALL SIMQ (A1, B1, 9, IER1)
```

Эти оба способа задания числовой величины для использования стандартных подпрограмм эквивалентны.

9.3. Составление программ для решения задач статики с использованием стандартных подпрограмм SIMQ и ARRAY

9.3.1. Простейшая программа с использованием подпрограммы SIMQ

По образцу п. 8.3 кратко рассмотрим фортранную реализацию рассмотренных там программ 8.1-8.5 с использованием SIMQ и ARRAY. Новые программы 9.1-9.5, решающие аналогичные задачи, будут иметь ту же вторую цифру номера (1-5) и отличаться только номером главы (9, а не 8). Операторы в программах 9.1-9.5, номера которых совпадают с соответствующими номерами программ 8.1-8.5, полностью эквивалентны и повторно не описываются. Отличающиеся операторы в этих программах имеют для удобства ориентировки после цифры номера букву S. Только они и будут описаны.

Простейшая программа для решения задач статики на ЭВМ с использованием подпрограммы SIMQ для СЛАУ (6.3) имеет вид:

```

C   П Р О Г Р А М М А 9.1
      DIMENSION A(81),B(9)                10S
      READ *,N,A,B                          20
      PRINT *,N,A,B                          30
      CALL SIMQ(A,B,N,IER)                  40S
      PRINT *,B,IER                          50S
      STOP                                    61
      END                                    62

```

Из сравнения программ 8.1 и 9.1 видно, что для использования подпрограммы SIMQ нужно:

- изменить оператор описания DIMENSION, описав матрицу A в виде одномерного массива (оператор 10S);
- обратиться к подпрограмме SIMQ (оператор 40S), удалив из программы 8.1 обращение к подпрограммам DECOMP и SOLVE (операторы 40 и 47);
- изменить оператор печати выходной информации, введя в него печать кода ошибки решения IER (в программе 9.1 все это выполнено в самой простой форме). Отметим, что элементы матрицы A для ввода по оператору 20 также располагаются по столбцам и совпадают с примером (8.4).

9.3.2. Организация ввода матриц по строкам для работы подпрограммы SIMQ с использованием подпрограммы ARRAY

Изменения, касающиеся возможности использования подпрограммы SIMQ, выполним для программы 8.2. Опишем только отличающиеся операторы получившейся программы 9.2:

```

С      П Р О Г Р А М М А 9.2
      DIMENSION A(9,9),B(9),AS(81)          10S
      NDIM=9                                11S
      READ *,N,KSU                          20
      DO 75 K=1,KSU                          21
      READ *,((A(I,J),J=1,N),I=1,N)        22
      READ *,(B(I),I=1,N)                  23
      PRINT 20,K,((A(I,J),J=1,N),I=1,N)    30
20    FORMAT(/5X,'ВХОДНАЯ МАТРИЦА ЛЕВОЙ ЧАСТИ A',I2/  31
      * (1X,9F8.3))
      PRINT 25,K,(B(I),I=1,N)              32
25    FORMAT(5X,'ВЕКТОР ПРАВЫХ ЧАСТЕЙ B',I2/1X,9F8.3)  33
      CALL ARRAY(2,N,N,NDIM,NDIM,AS,A)    40S
      CALL SIMQ(AS,B,N,IER)               47S
      PRINT 65,K,(B(I),I=1,N)             50
65    FORMAT(15X,'РЕЗУЛЬТАТЫ РЕШЕНИЯ СИСТЕМЫ УРАВНЕНИЙ ', 51
      * 'НОМЕР ',I2/1X,9F8.3)
      PRINT 70,IER                         52S
70    FORMAT(5X,'КОД ОШИБКИ IER=',I2)    53S
75    CONTINUE                             59
      STOP                                  60
      END                                   61

```

Оператор 10S задает описание двумерного массива A и одномерных массивов B и AS. В последнем будут находиться элементы матрицы A после их преобразования в векторную форму памяти.

Оператор 11S задает значение строчного размера массива A, указанного при его описании в операторе DIMENSION.

Оператор 40S выполняет обращение к подпрограмме ARRAY. Значение MODE=2, поэтому производится преобразование двумерного массива в одномерный. Число строк и столбцов вследствие своего равенства задается для реальной матрицы A значением числа уравнений N, а для указанных при описании матрицы A в операторе DIMENSION — значением NDIM. Отметим, что в данном частном случае эти 4 фактических параметра можно задать, просто 4 раза указав N. AS — выходной одномерный массив, содержащий входную матрицу A с расположением элементов в векторной форме

памяти. A — входная матрица A , описанная в операторе DIMENSION в виде двумерного массива.

Оператор 47S выполняет обращение к подпрограмме SIMQ со следующими фактическими параметрами: AS — одномерный массив с матрицей A в векторной форме памяти, полученный подпрограммой ARRAY; B — вектор-столбец свободных членов; N — число уравнений в данной задаче; IER — код ошибки решения.

Оператор 52S под управлением оператора 53S осуществляет печать кода ошибки решения. В следующей строке после печати результатов решения, отступив 5 пробелов, будет напечатано литеральное данное: “КОД ОШИБКИ IER=”, за которым по спецификации I2 — значение IER.

Остальные операторы с совпадающими номерами эквивалентны с программой 8.2.

Исходные данные для работы программы 9.2 представлены в примере (8.8). Результаты работы — на распечатке (8.9) с небольшими изменениями, описанными для операторов 52S и 53S.

В программе 9.2 (а также в нижеследующих 9.3-9.5) представлена общая форма использования подпрограмм SIMQ и ARRAY в своей частной реализации, когда значение NDIM=N. Она позволяет во всех этих программах работать с массивами переменной размерности. Для этого значение NDIM в операторах 10S и 11S следует задать равным его предполагаемой максимальной величине.

Однако в частном случае при NDIM=N мы имеем совпадение двумерной и векторной форм памяти (см. п. 9.2.1). Поэтому можно не описывать одномерный массив AS и отказаться от обращения к подпрограмме ARRAY (удалив оператор 40S из соответствующих программ), а сразу обращаться к подпрограмме SIMQ, задавая в качестве первого фактического аргумента исходную матрицу A :

```
DIMENSION A(9,9),B(9)           10S
CALL SIMQ(A,B,N,IER)           47S
```

Однако в том случае, если размер NDIM не совпадет с N (даже если N будет меньше NDIM), то сразу возникнет ошибка решения. По этой причине все нижеследующие программы 9.3-9.5 также приведены в самой общей форме, что не мешает вам проверить на ЭВМ сделанное замечание.

Ниже без описания представлены программы 9.3-9.5, полностью аналогичные соответствующим программам 8.3-8.5. Они используют вместо подпрограмм DECOMP и SOLVE подпрограмму SIMQ из пакета ПНП-SSP. Их ввод исходных данных также одинаков. Операторы с совпадающими номерами в программах 9.3-9.5 и 8.3-8.5 полностью эквивалентны и опи-

связываются в соответствующих местах при описании программ 8.3-8.5. Все отличающиеся операторы (10S, 11S, 40S, 47S, 52S, 53S) описаны в пояснениях к программе 9.2.

```

С      П Р О Г Р А М М А 9.3
      DIMENSION A(9,9),B(9),AS(81)          10S
      NDIM=9                                11S
      READ *,N,KSU                           20
      DO 75 K=1,KSU                           21
          DO 6 I=1,N                          22
              B(I)=0.                         23
              DO 6 J=1,N                      24
                  6      A(I,J)=0.           25
                  READ *,KOR                  26
                  DO 7 M=1,KOR               27
                      7      READ *,KNNEKO,(I,J,A(I,J),I1=1,KNNEKO) 28
                      READ *,KNNEKO,(I,B(I),I1=1,KNNEKO) 29
                      PRINT 20,K,((A(I,J),J=1,N),I=1,N) 30
20     FORMAT(/5X,'ВХОДНАЯ МАТРИЦА ЛЕВОЙ ЧАСТИ A',I2/
*        (1X,9F8.3))                          31
        PRINT 25,K,(B(I),I=1,N)                32
25     FORMAT(5X,'ВЕКТОР ПРАВЫХ ЧАСТЕЙ B',I2/1X,9F8.3) 33
        CALL ARRAY(2,N,N,NDIM,NDIM,AS,A)      40S
        CALL SIMQ(AS,B,N,IER)                 47
        PRINT 65,K,(B(I),I=1,N)              50
65     FORMAT(15X,'РЕЗУЛЬТАТЫ РЕШЕНИЯ СИСТЕМЫ ',
*        'УРАВНЕНИЙ НОМЕР ',I2/1X,9F8.3)     51
        PRINT 70,IER                          52S
70     FORMAT(5X,'КОД ОШИБКИ IER=',I2)       53S
75 CONTINUE                                  59
      STOP                                    60
      END                                     61

С      П Р О Г Р А М М А 9.4
      DIMENSION A(9,9),B(9),AD(9,9),BD(9),AS(81) 10S
      NDIM=9                                11S
      READ *,N                               20
      READ *,((A(I,J),J=1,N),I=1,N)         22
      READ *,(B(I),I=1,N)                   23
      DO 10 I=1,N                             24
          BD(I)=B(I)                          25
          DO 10 J=1,N                          26
10     AD(I,J)=A(I,J)                        27
          K=0                                  28
15     K=K+1                                  29

```

	PRINT 20, K, ((A(I, J), J=1, N), I=1, N)	30
20	FORMAT(/5X, 'ВХОДНАЯ МАТРИЦА ЛЕВОЙ ЧАСТИ A', I2/ * (1X, 9F8.3))	31
	PRINT 25, K, (B(I), I=1, N)	32
25	FORMAT(5X, 'ВЕКТОР ПРАВЫХ ЧАСТЕЙ B', I2/1X, 9F8.3)	33
	CALL ARRAY(2, N, N, NDIM, NDIM, AS, A)	40S
	CALL SIMQ(AS, B, N, IER)	47S
	IF (B(1).GT.0.0) GO TO 60	51
	DO 50 I=1, N	61
	B(I)=BD(I)	62
	DO 50 J=1, N	63
	A(I, J)=AD(I, J)	64
50	CONTINUE	65
	DO 55 I=1, N	70
	READ *, A(I, 1)	71
55	CONTINUE	72
	GO TO 15	74
60	PRINT 65, K, (B(I), I=1, N)	80
65	FORMAT(15X, 'РЕЗУЛЬТАТЫ РЕШЕНИЯ СИСТЕМЫ УРАВНЕНИЙ ', * 'НОМЕР ', I2/1X, 9F8.3)	81
	PRINT 70, IER	52S
70	FORMAT(5X, 'КОД ОШИБКИ', I2)	53S
	STOP	90
	END	91
C	ПРОГРАММА 9.5	
	DIMENSION A(9, 9), B(9), AD(9, 9), BD(9), AS(81)	10S
	NDIM=9	11S
	READ *, N	20.4
	DO 6 I=1, N	22
	B(I)=0.	23
	DO 6 J=1, N	24
6	A(I, J)=0.	25
	READ *, KOR	26
	DO 7 M=1, KOR	27
7	READ *, KNNEKO, (I, J, A(I, J), I1=1, KNNEKO)	28
	READ *, KNNEKO, (I, B(I), I1=1, KNNEKO)	29
	DO 10 I=1, N	24.4
	BD(I)=B(I)	25.4
	DO 10 J=1, N	26.4
10	AD(I, J)=A(I, J)	27.4
	K=0	28.4
15	K=K+1	29.4
	PRINT 20, K, ((A(I, J), J=1, N), I=1, N)	30

```

20 FORMAT (/5X, 'ВХОДНАЯ МАТРИЦА ЛЕВОЙ ЧАСТИ A', I2/ 31
* (1X, 9F8.3))
PRINT 25, K, (B(I), I=1, N) 32
25 FORMAT (5X, 'ВЕКТОР ПРАВЫХ ЧАСТЕЙ B', I2/1X, 9F8.3) 33
CALL ARRAY (2, N, N, N, DIM, DIM, AS, A) 40S
CALL SIMQ (AS, B, N, IER) 47S
IF (B(1).GT.0.0) GO TO 60 51.4
DO 50 I=1, N 61.4
B(I)=BD(I) 62.4
DO 50 J=1, N 63.4
A(I, J)=AD(I, J) 64.4
50 CONTINUE 65.4
DO 55 I=1, N 70N
A(I, 1)=0. 71N
55 CONTINUE 72N
READ *, KNNEKO, (I, J, A(I, J), I1=1, KNNEKO) 73N
GO TO 15 74.4
60 PRINT 65, K, (B(I), I=1, N) 80.4
65 FORMAT (15X, 'РЕЗУЛЬТАТЫ РЕШЕНИЯ СИСТЕМЫ ',
* 'УРАВНЕНИЙ НОМЕР ', I2/1X, 9F8.3) 81.4
PRINT 70, IER 52S
70 FORMAT (5X, 'КОД ОШИБКИ IER=', I2) 53S
STOP 90.4
END 91.4

```

В приложении 1 представлены с достаточным количеством комментариев подпрограммы DDECOM и DSOLVE в качестве материала для УИРС (работа с удвоенной точностью подробно описана в п. 15).

ЧАСТЬ 4. ВЫПОЛНЕНИЕ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ ПО КИНЕМАТИКЕ С ИСПОЛЬЗОВАНИЕМ ГОТОВЫХ ПРОГРАММ

ГЛАВА 10. РАБОТА С ИСПОЛЬЗОВАНИЕМ ГОТОВЫХ ПРОГРАММ KINMP И DKINMP

Решение задач кинематики с применением ЭВМ обычно сводится к численному дифференцированию уравнений движения, функций положения и т.п. Поэтому рассмотрим различные аспекты его применения при работе с готовыми программами.

10.1. Краткое описание основных особенностей программ KINMP и DKINMP

Программа KINMP производит определение кинематических характеристик (скорости, ускорения) и радиуса кривизны траектории материальной точки по заданным уравнениям ее движения. Она предназначена для решения задач на ПЭВМ при изучении кинематики точки или ее сложного движения. В качестве типовых выбраны примеры решения заданий на соответствующие темы: К-1 [21, с. 60-62] или [22, с. 76-79], К-2 [22, с. 82-87], К-7 [21, с. 99-106] или К-10 [22, с. 137-143]. Ссылка на задания производится по их номеру, причем типовым примерам присвоен 31-й вариант.

Студентом подготавливаются исходные данные (см. п. 10.2) и уравнения движения материальной точки в виде подпрограммы SUBROUTINE KINU (см. п. 10.3) для определения программой KINMP кинематических характеристик при различных временах T (что называется численным решением). Затем следует сеанс работы на ПЭВМ. Одновременно с этим обычными методами механики студентом производится параллельное аналитическое решение указанных заданий при одном заданном времени T_1 (которое рассмотрено в сборниках [21], [22] и в пособии не повторяется).

В различных подпрограммах для каждого варианта указанных заданий хранятся как правильные уравнения движения, так и правильные аналитические

решения. Численное решение студента сравнивается с правильным численным при каждом выводимом на печать значении времени T , а с правильным аналитическим — только при заданном времени T_1 .

Программа DKINMP представляет собой программу KINMP с использованием удвоенной точности. Работа с ней полностью аналогична.

Отметим, что требуемые уравнения движения материальной точки задаются в K-1, их составление является необходимым условием для аналитического решения K-2 и дополнительной работой для численного решения K-7 (или K-10), так как аналитическое решение заданий на сложное движение точки проводится другим способом с использованием кинематической теоремы Кориолиса.

Программы KINMP и DKINMP используют для численного дифференцирования формулы для представления первой и второй производных функций в конечных разностях:

$$dF(T)/dT = (F(T+DT) - F(T)) / DT \quad (10.1)$$

$$d(dF(T)) / (dT**2) = (F(T+DT) - 2 * F(T) + F(T-DT)) / DT**2 \quad (10.2)$$

где DT — шаг дифференцирования, т.е. некоторое приращение времени DT (при определенном времени T), от величины которого зависит точность численного вычисления производных.

В целях экономии места исходные данные (в п. 10.2) и подпрограммы с уравнениями движения (в п. 10.3) приводятся сразу для всех трех типовых примеров рассматриваемых заданий (для K-7 [21, с. 99-106] и K-10 [22, с. 137-143] они совпадают). Конечно, при выполнении задания на ЭВМ следует рассматривать только один соответствующий пример.

10.2. Подготовка данных для работы программы KINMP

10.2.1. Структура вводимых данных для работы программы KINMP

1-я строка: перечисляются номера группы NG , задания по кинематике NZ , варианта NW . В рассматриваемых ниже примерах используется: $NG=114419$, $NZ=1$, $NZ=2$ и $NZ=7$, $NW=31$.

2-я строка: указывается заданное в условии время T_1 .

Значения исходных данных, вводимые в строках 3-5, предварительно задаются в программе. Если нет необходимости их изменять, то в первой позиции этих строк следует поставить знак наклонной черты “/”, указывающий на переход к следующей строке данных. В результате этого

оставшиеся в списке ввода элементы пропускаются (все или часть) и их значения в памяти ЭВМ не изменяются.

3-я строка: указываются приращение последовательных значений аргумента HT и шаг дифференцирования DT . В программе эти величины предварительно задаются относительно времени $T1$: $HT=T1/10$ и $DT=HT/10$. Величина HT определяет шаг расчета и печати, т.е. интервал между двумя значениями времени T , при которых происходит расчет кинематических характеристик и печать результатов. Значение DT определяет приращение времени T при определении производных в уравнениях (10.1)-(10.2) и оказывает существенное влияние на погрешность численного решения. Поэтому после получения первоначального решения варьированием величины DT следует постараться уменьшить погрешность решения.

4-я строка: перечисляются значения времени начала TN и окончания TK численного расчета. Предварительно в программе TN приравняется нулю, а TK — удвоенному времени $T1$, заданному для данного варианта.

5-я строка: указывается параметр переключения формата печати $IFORM$ и допустимая величина относительной ошибки EPS (в %) для абсолютного ускорения $DELA1$ (задания $K-7$ и $K-10$) или радиуса кривизны траектории $DELRO1$ (задания $K-1$ и $K-2$). Параметр переключения формата печати $IFORM$ используется только для заданий $K-1$ и $K-2$ и может принимать значения 1, 2 и 3 (см. п. 10.5.3). В программе предварительно задается $IFORM=1$ и $EPS=10$.

10.2.2. Представление исходных данных для программы KINMP

Исходные данные строк 3-4, предварительно задаваемые в программе относительно времени $T1$, выбраны так, чтобы получать оптимальные (в первом приближении) численные значения искомых величин. Поэтому их рекомендуется использовать при первоначальном решении, то есть в строках 3-5 ставить знак наклонной черты “/”. С учетом этого исходные данные согласно структуре п. 10.2.1 для типовых примеров рассматриваемых заданий $K-1$, $K-2$ и $K-7$ будут иметь соответственно следующий вид:

```
114419, 1, 31
0.5
/
/
/
/
```

(10.4)

```
114419, 2, 31
0.33333333
```

(10.5)


```

/
/
/
114419, 7, 31
0.22222222                (10.6)
/
/
/

```

Во второй строке примеров (10.4)-(10.6) указано приведенное в условии время T1. Определив согласно описанию п. 10.2.1 значения величин, предварительно задаваемых в программе (HT, DT, TK, IFORM и EPS), получим эквивалентную полную форму представления исходных данных (10.7)-(10.9). Она наглядно показывает, какие значения перечисленных величин на самом деле применяются при использовании фрагментов (10.4)-(10.6) :

```

114419, 1, 31
0.5                        (10.7)
0.05, 0.005
0., 1.
1, 10.

```

```

114419, 2, 31
0.33333333                (10.8)
0.03333333, 0.00333333
0., 0.66666666
1, 10.

```

```

114419, 7, 31
0.22222222                (10.9)
0.02222222, 0.00222222
0., 0.44444444
1, 10.

```

Для замены предварительно задаваемых в программе величин нужные численные значения следует указать в соответствующих позициях строк 3-5 примеров (10.7)-(10.9). При этом возможны два способа упрощений.

1. Если после измененной величины в строке справа остаются неизмененные, то вместо них следует поставить разделитель "/". Например, чтобы в примере (10.7) в 3-й строке задать начальное приращение аргумента HT равным 0.01, а шаг дифференцирования DT оставить без изменения, то в 3-й строке исходных данных следует указать:

```
0.01/                      (10.10)
```

Соответствующая фрагменту (10.10) запись данных в полной форме будет иметь вид:

$$0.01, 0.005 \quad (10.11)$$

2. Если неизменяемый элемент расположен слева от изменяемого, то он задается как отсутствующее данное. Тогда соответствующий элемент списка ввода пропускается и его значение в памяти ЭВМ не изменяется. Это представляется для элемента внутри списка двумя запятыми, следующими друг за другом “,,”, а для первого отсутствующего вводимого элемента одной запятой “,”, с которой начинается список ввода. Например, чтобы в примере (10.7) в 3-й строке задать значение шага дифференцирования ΔT равным 0.001, а величину ΔT оставить без изменения, то в 3-й строке исходных данных следует указать:

$$, 0.001 \quad (10.12)$$

Соответствующая фрагменту (10.12) запись данных в полной форме будет иметь вид:

$$0.05, 0.001 \quad (10.13)$$

10.3. Представление уравнений движения в виде подпрограммы KINU

10.3.1. Составление уравнений движения

Чтобы составить уравнения движения материальной точки, нужно для произвольного момента времени T выразить ее координаты через данные задачи и параметр T . Время T обычно входит в исходные данные (выражения углов или в уравнения относительного SR и переносного VIE движений). Полученные таким образом выражения координат для одного произвольного положения материальной точки включают переменную T и остаются справедливыми для любого момента времени, являясь уравнениями движения материальной точки в координатном виде в параметрической форме.

Заметим, что во многих вариантах задания К-7 [21, с. 99-106] (и К-10 [21, с. 137-143]) для более наглядного выражения координат нужно дать произвольное перемещение тела, переместив его из заданного положения на рисунке в любое произвольное по направлению переносного движения.

В этих заданиях также не определено нулевое положение переносной угловой координаты и, в зависимости от его выбора, уравнения могут принимать разнообразную форму, отличаясь друг от друга в конечном счете на постоян-

ную. Это приводит к разным значениям координат точек, определяемых по этим разным уравнениям, но к одинаковым значениям скоростей и ускорений.

Для типовых примеров рассматриваемых заданий уравнения движения материальной точки имеют вид:

- для К-1 [21, с. 60]:

$$X=4 * T, \quad Y=16 * T * T-1, \quad (10.14)$$

- для К-2 [22, с. 82]:

$$\begin{aligned} X &= 20 * (6 * \sin(AL) - \sin(VI + AL)), \\ Y &= 20 * (6 * \cos(AL) - \cos(VI + AL) - 5), \end{aligned} \quad (10.15)$$

где $PI=3.14159265$, $VI=PI * T$, $AL=0.2 * VI$;

- для задания К-7 уравнения движения материальной точки не используются при его аналитическом решении в сборнике [21, с. 99-106]. Поэтому они составляются дополнительно с учетом вышеприведенных пояснений:

$$X=-SR/2 * \sin(VIE), \quad Y=SR/2 * \cos(VIE), \quad Z=SR * \cos(PI/6), \quad (10.16)$$

где $SR=16-8 * \cos(3 * PI * T)$, $VIE=0.9 * T ** 2 - 9 * T ** 3$, $PI=3.1415926$;

Отметим, что типовые примеры заданий К-7 [21, с. 99-106] и К-10 [22, с. 137-143] одинаковы. Поэтому их программная реализация совпадает и рассматривается на примере (10.16).

10.3.2. Составление подпрограммы KINU

Чтобы определить те вычисления, которые будет производить подпрограмма KINU, необходимо написать участок программы, содержащий необходимые операторы присваивания, задающие уравнения движения материальной точки. Они могут быть представлены в самых разнообразных формах (см. Дополнения 10.1 и 10.2). Перед этим участком следует расположить заголовок подпрограммы SUBROUTINE KINU со списком использованных формальных параметров (M,T,X,Y,Z), служащих для информационной связи с основной программой KINMP (см. п. 12.1.3).

Вторым в подпрограмме следует оператор описания используемых массивов: DIMENSION T(3),X(3),Y(3),Z(3). По нему для каждой переменной T, X, Y и Z описывается массив, состоящий из трех элементов. Тогда можно в первых элементах соответствующих массивов помещать значения указанных величин при фактическом параметре T, во вторых — при T+DT, в третьих — при T-DT, которые используются в формулах (10.1)-(10.2), и все последующие вычисления выполнять со значениями переменных, расположенных в соответствующих элементах массива. Заканчиваться подпрограмма должна операторами RETURN и END.

Для рассматриваемых примеров (10.14)-(10.16) подпрограмма KINU будет иметь соответственно вид (10.17)-(10.19):

```

SUBROUTINE KINU (M, T, X, Y, Z)
  DIMENSION T (3), X (3), Y (3), Z (3)
  X (M) = 4. * T (M)
  Y (M) = 16. * T (M) * T (M) - 1.
  Z (M) = 0.
  RETURN
END

```

(10.17)

```

SUBROUTINE KINU (M, T, X, Y, Z)
  DIMENSION T (3), X (3), Y (3), Z (3)
  PI = 3.141592653589
  VI = PI * T (M)
  AL = 0.2 * VI
  X (M) = 20. * ( 6. * SIN (AL) - SIN (VI + AL) )
  Y (M) = 20. * ( 6. * COS (AL) - COS (VI + AL) - 5. )
  Z (M) = 0.
  RETURN
END

```

(10.18)

```

SUBROUTINE KINU (M, T, X, Y, Z)
  DIMENSION T (3), X (3), Y (3), Z (3)
  PI = 3.141592653589
  VIE = 0.9 * T (M) * T (M) - 9. * T (M) ** 3
  SR = 16. - 8. * COS ( 3. * PI * T (M) )
  X (M) = -SR / 2. * SIN (VIE)
  Y (M) = SR / 2. * COS (VIE)
  Z (M) = SR * COS (PI / 6. )
  RETURN
END

```

(10.19)

Дополнение 10.1. При описании подпрограмм SUBROUTINE могут использоваться переменные, которых нет в списке формальных параметров (переменные PI, VI, AL и PI, VIE, SR примеров (10.18) и (10.19) соответственно).

Подпрограмму SUBROUTINE KINU в примере (10.17) также можно записать, используя вместо чисел переменные A, B, C (или любые другие). Напомним, что значения этих переменных должны быть определены до первого их использования, например, с помощью оператора DATA (см. п. 7.7.1): DATA A,B,C/4.,16.,1./.

Теперь подпрограмма KINU примера (10.17) примет вид:

```

SUBROUTINE KINU (M, T, X, Y, Z)
  DIMENSION T (3), X (3), Y (3), Z (3)
  DATA A, B, C / 4., 16., 1. /
  X (M) = A * T (M)

```

(10.20)

```

Y (M) =B*T (M) **2-C
Z (M) =0 .
RETURN
END

```

Рекомендуем выполнить аналогичную замену для собственного варианта задания и проверкой на ЭВМ убедиться в эквивалентности примеров (10.17) и (10.20).

Дополнение 10.2. Уравнения движения материальной точки, представленные в примерах (10.17)-(10.19), можно описать в любой удобной для вас форме, т. е. после оператора описания DIMENSION и перед заданием уравнений движения (X(M)=, Y(M)=, Z(M)=), может быть как любое количество операторов, вводящих вспомогательные переменные (для примера (10.19) PI, VIE, SR), так и ни одного. В последнем случае задание уравнений движения происходит сразу с использованием чисел, а не вспомогательных переменных :

```

SUBROUTINE KINU (M, T, X, Y, Z)
DIMENSION T (3), X (3), Y (3), Z (3)
X (M) =- (16.-8.*COS (3.*3.14159*T (M) ) ) /2.*
1      SIN (0.9*T (M) *T (M) -9.*T (M) **3)          (10.21)
Y (M) = (16.-8.*COS (3.*3.14159*T (M) ) ) /2.*
2      COS (0.9*T (M) *T (M) -9.*T (M) **3)
Z (M) = (16.-8.*COS (3.*3.14159*T (M) ) ) *COS (3.14159/6.)
RETURN
END

```

Однако такая громоздкая запись операторов в подпрограмме делает более трудоемким процесс их отладки, не добавляя обычно никаких преимуществ. В примере (10.21) введено значение переменной PI в виде числа 3.14159 и непосредственно при задании уравнений движения используются выражения для VIE и SR. Это дает возможность удалить три арифметических оператора присваивания из примера (10.19), вводящих соответствующие вспомогательные переменные. Подпрограмма (10.21) стала короче (10.19) на три оператора, но она будет дольше выполняться.

Теперь вместо одного представления числа 3.14159 в двоичной системе и помещения его в ячейку для переменной PI, откуда оно очень быстро берется для использования, нужно четыре раза выполнять это преобразование, что требует больше времени. Также трижды приходится вычислять выражение для SR и дважды для VIE в примере (10.21) по сравнению с однократным вычислением в примере (10.19).

Поэтому обычно стараются избегать слишком громоздких записей любых операторов. К тому же в громоздких выражениях гораздо легче сделать ошибку и гораздо труднее ее найти.

10.4. Работа с удвоенной точностью (программа DKINMP)

Необходимость использования удвоенной точности возникает в рамках задач кинематики при варьировании шага дифференцирования DT . При уменьшении DT ошибка численного расчета по сравнению с аналитическим решением будет сначала уменьшаться до определенного предела, но затем станет увеличиваться вплоть до ее лавинообразного нарастания (особенно для ускорения A). Ошибки вычисления становятся очень большими, когда происходит уменьшение величины DT (и особенно DT^{**2}) настолько, что оно становится сравнимым с ошибками ограничения (усечения) и округления чисел при их переводе в понимаемую ЭВМ двоичную форму. Рекомендуем проверить это при работе с использованием программы KINMP.

При использовании удвоенной точности можно легко определить факт возникновения подобной числовой неустойчивости без получения аналитического решения, которое в общем случае для ряда задач может быть весьма трудоемким или даже невозможным. Для этого достаточно сравнить между собой результаты двух численных решений с обычной и удвоенной точностью, между которыми в подобных ситуациях обнаружится существенное расхождение. Использование удвоенной точности позволяет намного отодвинуть границу потери устойчивости численного решения (в сторону уменьшения величины DT в данном случае). При тех значениях DT , когда для обычной точности уже происходит лавинообразное возрастание ошибки расчета, при использовании удвоенной точности еще получают правильные результаты, отчего и возникает расхождение между ними.

Этот эксперимент с варьированием DT позволяет наглядно показать необходимость внимательного и осознанного подхода к численным расчетам на ЭВМ, а также важность умения вычислений с удвоенной точностью (см. п. 15).

Для работы с удвоенной точностью следует использовать готовую программу DKINMP, которая аналогична описанной выше программе KINMP. Исходные данные в файле `dkinmp.dat` и уравнения движения в файле `dkinu.for` должны быть представлены с учетом дополнительных требований, возникающих при использовании удвоенной точности (см. п. 15.3).

Для рассматриваемых типовых примеров (10.14)-(10.16) подпрограмма DKINU удвоенной точности будет иметь, соответственно вид (10.23)-(10.25):

```

SUBROUTINE DKINU (M, T, X, Y, Z)
  IMPLICIT REAL *8 (A-H, O-Z)
  DIMENSION T (3), X (3), Y (3), Z (3)
  X (M) = 4. D0 * T (M)
  Y (M) = 16. D0 * T (M) * T (M) - 1. D0
  Z (M) = 0. D0
  RETURN
END

```

(10.23)

```

SUBROUTINE DKINU (M, T, X, Y, Z)
  IMPLICIT REAL *8 (A-H, O-Z)
  DIMENSION T (3), X (3), Y (3), Z (3)
  PI = 3. 141592653589D0
  VI = PI * T (M)
  AL = 0. 2D0 * VI
  X (M) = 20. D0 * ( 6. D0 * DSIN (AL) - DSIN (VI + AL) )
  Y (M) = 20. D0 * ( 6. D0 * DCOS (AL) - DCOS (VI + AL) - 5. D0 )
  Z (M) = 0. D0
  RETURN
END

```

(10.24)

```

SUBROUTINE DKINU (M, T, X, Y, Z)
  IMPLICIT REAL *8 (A-H, O-Z)
  DIMENSION T (3), X (3), Y (3), Z (3)
  PI = 3. 141592653589D0
  VIE = 0. 9D0 * T (M) * T (M) - 9. D0 * T (M) ** 3
  SR = 16. D0 - 8. D0 * DCOS ( 3. D0 * PI * T (M) )
  X (M) = -SR / 2. D0 * DSIN (VIE)
  Y (M) = SR / 2. D0 * DCOS (VIE)
  Z (M) = SR * DCOS (PI / 6. D0)
  RETURN
END

```

(10.25)

Здесь оператор `IMPLICIT REAL *8 (A-H, O-Z)` описывает все переменные, начинающиеся с букв A-H или O-Z, в удвоенной точности в каждой из подпрограмм (10.23)-(10.25). В удвоенной точности используются также все функции (`DSIN` и `DCOS`) и все вещественные числа, для чего использована экспоненциальная форма их записи (единичный множитель `D0`).

Исходные данные для работы программы `DKINMP` будут полностью аналогичны фрагментам (10.4)-(10.9) для соответствующих типовых примеров рассматриваемых заданий, только все вещественные числа должны быть указаны в удвоенной точности (что проще всего достигается указанием единичного множителя `D0` после значения вещественного числа).

10.5. Описание результатов работы программ KINMP и DKINMP

Представление результатов работы программ KINMP и DKINMP, отличающихся только используемой точностью вычислений, полностью совпадает. После печати заголовка выводимой информации сами результаты расчета программ KINMP и DKINMP выводятся в виде таблицы, которая имеет две основные формы (одну для K-1, K-2 и вторую для K-7, K-10).

Таблица результатов расчета для заданий K-1 и K-2 может быть представлена еще в трех модификациях в зависимости от значения параметра переключения формата печати IFORM. Это позволяет осуществлять проверку большего числа этапов аналитического расчета (значений скорости, полного или тангенциального ускорения).

Значение IFORM задается в 1-й позиции 5-й строки исходных данных и может быть равным 1, 2 или 3.

При IFORM=1 печатаются при каждом T координаты точек X и Y, значения скорости V, ускорения A и радиуса кривизны RO и их относительные ошибки расчета для V, A и RO по отношению к соответствующим (точным) значениям VT, AT и ROT, полученным из правильных уравнений движения, хранящихся в памяти ЭВМ. Такая форма представления результатов является основной, так как IFORM=1 предварительно задается в программе.

При IFORM=2 печатаются значения расчета по заданным и правильным уравнениям движения для скорости (V и VT), ускорения (A и AT) и их соответствующие ошибки (как всегда для заданий K-7 и K-10), а также радиус кривизны траектории RO и ошибка его определения.

При IFORM=3 формат печати в основном совпадает с IFORM=1, но вместо полного ускорения A распечатывается тангенциальное ускорение и ошибка его расчета.

В программах KINMP и DKINMP предусмотрены различные защиты от необдуманных действий, сопровождающиеся соответствующими аварийными сообщениями, после которых прекращается их выполнение.

Дополнение 10.3. В программах KINMP и DKINMP предусмотрена также защита от аварийной ситуации, которая может возникнуть в некоторых вариантах (2, 7, 12, 17, 20, 23, 26, 29) задания K-1 в случае прямолинейной траектории движения. В этом случае при точных аналитических расчетах полное ускорение A равняется его тангенциальному ускорению ATANG=A, а нормальное ускорение ANORM=0. Однако при численных расчетах таких точных равенств не достигается вследствие различных ошибок вычисления, ограничения (усечения) и округления (см. п. 2 в [33]). Они возникают вследствие того, что арифметические действия проводятся с конечным чис-

лом значащих цифр, с помощью которых часто невозможно представить необходимую величину конечной дробью в принятой (двоичной) системе счисления (например, $1/3$). Поэтому могут возникать различные частные случаи.

Вследствие разного распространения ошибок (см. п. 2 в [10] и дополнение 13.2) при вычислении двух почти равных чисел по разным формулам может получиться и $ATANG > A$. Это противоречит физическому смыслу (часть больше целого), но иногда может реально возникнуть в процессе вычислений, что приводит к отрицательному подкоренному выражению при определении нормального ускорения $ANORM$.

В этом случае выдается соответствующее сообщение, после которого программы продолжают свою работу обычным образом.

10.6. Методические указания по организации сеанса работы на ПЭВМ при работе с готовыми программами, требующими подготовки подпрограмм

Условимся, что имена личных архивных файлов на диске согласно (4.6) для приведенных там значений, например, для задания К-1, имеют вид:

<code>wm412k1.d31</code>	(10.26)
<code>wm412k1.f31</code>	(10.27)
<code>wm412k1.l31</code>	(10.28)

В файле (10.26) хранятся исходные данные, в (10.27) — подпрограмма `kinu`, задающая уравнения движения, а в (10.28) — результаты работы программы. Соответствующие им рабочие файлы, которые использует программа, например `kinmp`, имеют вид (10.29) — (10.31):

<code>kinmp.dat</code>	(10.29)
<code>kinu.for</code>	(10.30)
<code>kinmp.lis</code>	(10.31)

При использовании других программ, требующих подготовки исходных данных и одной подпрограммы, нижеописанная последовательность действий остается одинаковой. Во всех командах вместо имен `kinmp` и `kinu` следует соответственно указывать имена нужных программ (`dkinmp`, `idusp` или `didusp`) или подпрограмм (`dkinu`, `sdu1p` или `dsdu1p`).

Сначала нужно сделать текущим подкаталог, имя которого совпадает с именем выбранной программы. Для этого к нему (`KINMP`) следует подвести подсветку курсора и нажать клавишу `<Enter>` (`<Ввод>`). Далее действия выполняются согласно этапам приведенного сеанса работы, перед которым желательно повторить пп. 3.1 и 4.3.1.

1. Для создания или редактирования архивного файла в текущем каталоге необходимо при любом положении курсора нажать клавиши <Shift>+<F4> (<Верх>+<Ф4>). Затем в появившемся на экране приглашении набрать имя файла типа (10.26) (wm412k1.d31) и нажать клавишу <Enter>. Если файл с таким именем существует, то он вызывается на редактирование, если нет — то в сообщении об этом выделенным окажется вариант “New-file” (“Новый файл”) и нужно еще раз нажать <Enter>.

2. Выполнить набор исходных данных или исправление ошибок.

3. Сохранить информацию под архивными и рабочими именами. Для этого нужно сначала нажать клавишу <F2> (<Ф2>) (запись на диск с именем редактируемого файла (10.26)), затем <Shift>+<F2> (<Верх>+<Ф2>), в появившейся строке набрать нужное имя (10.29) (kinmp.dat) и нажать клавишу <Enter> (<ВВОД>). Для выхода из режима редактирования нужно нажать клавишу <F10> (<Ф10>). На экране опять появляются панели Norton Commander.

4. Подвести подсветку курсора к имени рабочего файла (10.29) (kinmp.dat) и нажать клавишу <F4> (<Ф4>). Убедиться в том, что в нем находятся ваши данные и еще раз проверить их. Если при проверке будут обнаружены ошибки, то после их исправления нужно также сначала нажать клавишу <F2> (<Ф2>) (теперь это запись на диск с именем редактируемого файла (10.29)), затем <Shift>+<F2> (<Верх>+<Ф2>), в появившейся строке набрать нужное имя (теперь это (10.26) — wm412k1.d31) и нажать клавишу <Enter> (<ВВОД>). Выход из режима редактирования также по <F10> (<Ф10>). Если под именем (10.29) находится не ваш файл, то внимательнее повторите основные этапы работы, начиная с 1-го.

5. Нажать клавиши <Shift>+<F4> (<Верх>+<Ф4>). В появившемся на экране приглашении набрать архивное имя подпрограммы с уравнениями движения типа (10.27) и нажать клавишу <Enter> (два раза, если это новый файл).

6. Выполнить набор исходного текста подпрограммы или исправление ошибок.

7. Сохранить информацию под архивными и рабочими именами. Нажать клавишу <F2> (<Ф2>) (запись на диск с именем редактируемого файла (10.27)), затем <Shift>+<F2> (<Верх>+<Ф2>), в появившейся строке набрать нужное имя (10.30) (kinu.for) и нажать клавишу <Enter> (<ВВОД>). Выход из режима редактирования по <F10> (<Ф10>).

8. Вызвать по <F4> (<Ф4>) рабочий файл (10.30) (kinu.for). Убедиться в том, что в нем находятся ваши уравнения и еще раз проверить их. Если пришлось исправить ошибки, то нажать клавишу <F2> (<Ф2>) (запишется kinu.for), затем <Shift>+<F2> (<Верх>+<Ф2>), набрать архивное имя (10.27) (wm412k1.f31) и нажать <Enter> (<ВВОД>). Выйти по <F10> (<Ф10>). Если под именем (10.30) (kinu.for) находится не ваш файл, то повторить сеанс работы с этапа 5.

9. Выполнить компиляцию подпрограммы при помощи команды FL [28, с. 140-174]. Для нее обычно создают командный файл, например, `fortfl.bat`. Тогда команда примет вид `fortfl ИФ` (где ИФ — имя компилируемой подпрограммы), например `fortfl kinu`. О нормальном ее завершении свидетельствует отсутствие диагностических сообщений. При их наличии вызвать по <F4> (<Ф4>) рабочий файл (10.30) (`kinu.for`), исправить синтаксические ошибки и продолжить с этапа 8.

10. Вызов Редактора связей (команда LINK [28, с. 175-191]) для создания выполняемого файла (см. п. 8.4). Для нее также обычно создают командный файл, например, имя_файла_программы.bat (`kinmp.bat`). Тогда команда примет вид `kinmp.bat ИФ` (где ИФ — имя дополняемой подпрограммы), например `kinmp.bat kinu`. При наличии диагностических сообщений следует обратиться к системному программисту. После нормального завершения работы команды в текущем каталоге должен появиться файл имя_файла_программы.exe (`kinmp.exe`).

11. Произвести расчет по программе путем запуска на выполнение файла имя_файла_программы.exe. Для этого в нашем случае следует подвести подсветку курсора к `kinmp.exe` и нажать клавишу <Enter> (<Ввод>).

12. Нормальное завершение расчета (сообщение “Stop-Program terminated”). Для просмотра результатов совместить подсветку курсора с именем файла результатов имя_файла_программы.lis (`kinmp.lis`) и нажать клавишу <F3> (<Ф3>) (или <F4> (<Ф4>), чтобы не было искажения таблицы). Выход в обоих случаях по <F10> (<Ф10>). Если указанная погрешность решения меньше допустимой, то перейти к этапу 14.

13. Если погрешность расчета велика, то выполнить проверку вводимых исходных данных и уравнений, произвести исправление ошибок и внимательно повторить сеанс работы.

14. Убедившись в правильности решения, выполняем печать результатов расчета. После включения принтера и заправки его бумагой, последовательно подводим подсветку курсора к именам файлов исходных данных (10.26), уравнений (10.27) и результатов (10.31). После каждого раза нажимаем клавишу F5 (<Ф5>), в появившейся строке набираем `prt` и нажимаем <Enter> (<Ввод>).

ГЛАВА 11. ОПРЕДЕЛЕНИЕ УГЛОВЫХ СКОРОСТЕЙ И УГЛОВЫХ УСКОРЕНИЙ ЗВЕНЬЕВ МЕХАНИЗМА МАНИПУЛЯТОРА ПО ЗАДАННОМУ ДВИЖЕНИЮ РАБОЧЕЙ ТОЧКИ С ИСПОЛЬЗОВАНИЕМ ПРОГРАММ К9МР И ДК9МР

Манипулятор промышленного робота схематично можно представить в виде плоского механизма (рис. 11.1, а). Звенья этого устройства образуют “механическую руку” с захватом в точке А, уравнения движения которой заданы.

Программа К9МР предназначена для определения кинематических характеристик звеньев механизма манипулятора по заданным законам движения рабочих органов (задание К-9 [21, с. 115-123]). Программа производит сравнение результатов расчета, определяя относительную ошибку отклонения расчетной траектории от заданной на каждом шаге проводимых вычислений, а также выполняет проверку вводимых студентом исходных данных и недостающих начальных условий. Программа ДК9МР представляет собой программу К9МР с использованием удвоенной точности.

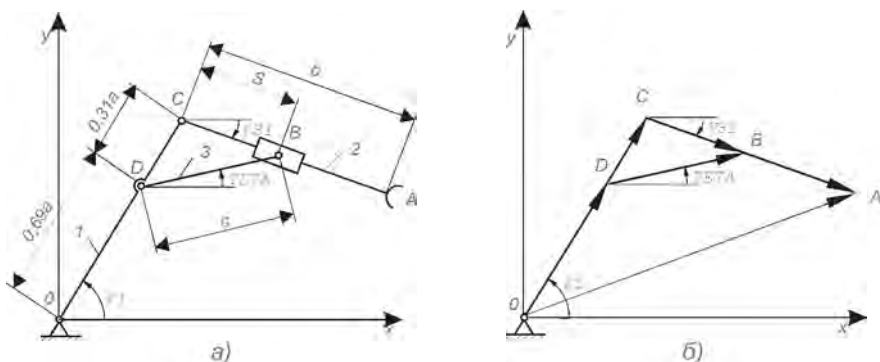


Рис. 11.1. Схема механизма манипулятора.

Рассмотрим решение типового примера задания К-9 [21, с. 119-123]. На рис. 11.1, а изображен в начальном положении механизм манипулятора и показаны положительные направления отсчета углов Ψ , Θ , Φ и расстояния S (для простоты и удобства оформления для обозначения углов на рисунке, в тексте и в программах выбраны одинаковые идентификаторы). Размеры звеньев (м): $a=1.30$, $b=1.10$, $c=0.55$. Захват (точка А) в течение 1 сек. движется согласно уравнениям:

$$\begin{aligned} X_a(t) &= 1.5329 - 0.2 * T, \\ Y_a(t) &= 0.5487 - 0.089 * T. \end{aligned} \quad (11.1)$$

В начальный момент времени (при $T=0$) $FI = FI_0 = 62$ град, $PSI = PSI_0 = 33$ град. Следует определить значения углов FI , PSI , $TETA$ и расстояние S для указанного промежутка времени. Нужно вычислить также угловые скорости FI' , PSI' , $TETA'$, угловые ускорения FI'' , PSI'' , $TETA''$, относительные скорость S' и ускорение S'' точки B . Вычисления следует произвести для моментов времени 0, 0.2, 0.4, 0.6, 0.8, 1 с.

Обратим основное внимание на его выполнение на ПЭВМ по программам К9МР и ДК9МР, остановившись на их особенностях и реализуемых ими возможностях. Аналитическая часть решения задания рассмотрена в [21], и подробно со всеми дополнениями, учитывающими особенности решения различных вариантов задания, в работе [3].

11.1. Описание этапов аналитического расчета и подготовки исходных данных

1) Для рассматриваемой схемы (рис. 11.1, б) составим уравнения связей в векторной форме (представленные под номерами (2)–(3) в [21, с. 119]):

$$\overline{OA} = \overline{OC} + \overline{CA} \dots \text{основной механизм}, \quad (11.2)$$

$$\overline{OB} = \overline{DC} + \overline{CB} \dots \text{управляющий механизм}. \quad (11.3)$$

2) Получим уравнения связей механизма в координатной форме, для чего спроектируем векторные соотношения (11.2)-(11.3) на оси декартовой системы координат (см. (5)-(6) в [21, с. 119-120]):

$$X_a = A * \cos(FI) + B * \cos(PSI); \quad (11.4)$$

$$Y_a = A * \sin(FI) - B * \sin(PSI);$$

$$C * \cos(TETA) = 0.31 * A * \cos(FI) + S * \cos(PSI); \quad (11.5)$$

$$C * \sin(TETA) = 0.31 * A * \sin(FI) - S * \sin(PSI).$$

Отметим, что левые части уравнений проекций основного механизма (11.4) X_a и Y_a , определяющие траекторию движения захвата (точки A), являются заданными в (11.1). Поэтому значения проекций координат точки A , определенные при любом времени T из уравнения (11.1), должны совпадать со значениями проекций, определенных правой частью уравнений (11.4) для соответствующего этому времени положению механизма. Согласованность задания начальных условий проверяется таким образом при $T=0$, в чем желательно убедиться перед выполнением расчета. Например, для рассматриваемого типового примера при $T=0$ будем иметь:

$$\begin{aligned}
 & (11.1) \quad (11.4) \\
 X_a & === 1.5329 === A \cdot \cos(\text{FI0}) + B \cdot \cos(\text{PSI0}) = \\
 & = 1.3 \cdot \cos(1.0821) + 1.1 \cdot \cos(0.57596) = 1.3 \cdot 0.46947 + \\
 & + 1.1 \cdot 0.83867 = 0.61031 + 0.92254 = 1.53285 \text{ (M)}, \quad (11.6)
 \end{aligned}$$

$$\begin{aligned}
 & (11.1) \quad (11.4) \\
 Y_a & === 0.5487 === A \cdot \sin(\text{FI0}) - B \cdot \sin(\text{PSI0}) = \\
 & = 1.3 \cdot \sin(1.0821) - 1.1 \cdot \sin(0.57596) = 1.3 \cdot 0.88295 - \\
 & - 1.1 \cdot 0.54464 = 1.14783 - 0.59910 = 0.54873 \text{ (M)}. \quad (11.7)
 \end{aligned}$$

3) Определим при времени $T=0$ из уравнений (11.5) недостающие начальные значения координаты S_0 и угла TETA0 . Для этой цели уравнения (11.5) следует представить в виде двух тождественных соотношений с выделенными в левой части функциями угла TETA , которые нужно рассмотреть при времени $T=0$, подставив в него соответствующие численные значения:

$$\begin{aligned}
 C \cdot \cos(\text{TETA0}) & = 0.31 \cdot A \cdot \cos(\text{FI0}) + S_0 \cdot \cos(\text{PSI0}); \quad (11.8) \\
 C \cdot \sin(\text{TETA0}) & = 0.31 \cdot A \cdot \sin(\text{FI0}) - S_0 \cdot \sin(\text{PSI0}).
 \end{aligned}$$

Здесь неизвестными являются S_0 и TETA0 . Возведем оба выражения (11.8) (с выделенными в левой части функциями угла TETA0) в квадрат и сложим соответственно их левые и правые части:

$$\begin{aligned}
 C^{**2} & = (0.31 \cdot A)^{**2} + 2 \cdot S_0 \cdot 0.31 \cdot A \cdot \cos(\text{FI0}) \cdot \cos(\text{PSI0}) - \\
 & - 2 \cdot S_0 \cdot 0.31 \cdot A \cdot \sin(\text{FI0}) \cdot \sin(\text{PSI0}) + S_0^{**2} = \\
 & = (0.31 \cdot A)^{**2} + 2 \cdot S_0 \cdot 0.31 \cdot A \cdot \cos(\text{FI0} + \text{PSI0}) + S_0^{**2}.
 \end{aligned}$$

В результате получим квадратное уравнение для определения S_0 . Для большей наглядности представим его в более привычном виде:

$$S_0^{**2} + 2 \cdot S_0 \cdot 0.31 \cdot A \cdot \cos(\text{FI0} + \text{PSI0}) + (0.31 \cdot A)^{**2} - C^{**2} = 0.$$

Подставим числовые значения входящих в него величин:

$$\begin{aligned}
 S_0^{**2} & + 2 \cdot S_0 \cdot 0.31 \cdot 1.3 \cdot \cos(1.0821 + 0.57596) + \\
 & + (0.31 \cdot 1.3)^{**2} - 0.55^{**2} = \\
 & = S_0^{**2} - 2 \cdot S_0 \cdot 0.403 \cdot \cos(1.65806) + \\
 & + 0.16241 - 0.3025 = 0.
 \end{aligned}$$

Или

$$S_0^{**2} - 2 \cdot S_0 \cdot 0.03512 - 0.14009 = 0.$$

Корнями квадратного уравнения будут следующие значения S_{01} и S_{02} :

$$\begin{aligned}
 S_{01} & = 0.03512 + \text{SQRT}(0.03512^{**2} - (-0.14009)) = 0.03512 + \\
 & + \text{SQRT}(0.001234 + 0.14009) = 0.03512 + \text{SQRT}(0.14132) = \\
 & = 0.03512 + 0.37593 = 0.4110 \text{ (M)};
 \end{aligned}$$

$$\begin{aligned}
 S_{02} & = 0.03512 - \text{SQRT}(0.03512^{**2} - (-0.14009)) = \\
 & = 0.03512 - 0.37593 = -0.3408 \text{ (M)};
 \end{aligned}$$

Из физических соображений значение S_0 не может быть отрицательным, так как отрицательному направлению отсчета относительной координаты S влево от точки A не соответствует реальное физическое содержание в виде продолжения стержня влево от точки C (см. рис. 11.1). Поэтому для рассматриваемого типового примера начальное значение $S_0 = S_{01} = 0.411$ (м).

Для получения начального значения $TETA_0$ сначала определим его тангенс, для чего следует разделить равенства (11.8) одно на другое (нижнее уравнение на верхнее), после чего будем иметь:

$$\begin{aligned} \text{TAN}(TETA_0) &= (0.31 \cdot A \cdot \text{SIN}(FI_0) - S_0 \cdot \text{SIN}(PSI_0)) / \\ &/ (0.31 \cdot A \cdot \text{COS}(FI_0) + S_0 \cdot \text{COS}(PSI_0)) = \\ &= (0.31 \cdot 1.3 \cdot \text{SIN}(1.0821) - 0.41105 \cdot \text{SIN}(0.57596)) / \\ &/ (0.31 \cdot 1.3 \cdot \text{COS}(1.0821) + 0.41105 \cdot \text{COS}(0.57596)) = \\ &= (0.403 \cdot 0.88295 - 0.41105 \cdot 0.54464) / (0.403 \cdot 0.46947 + \\ &+ 0.41105 \cdot 0.83867) = (0.3558279 - 0.22388) / \\ &/ (0.18920 + 0.34474) = 0.13195 / 0.53394 = 0.24713; \end{aligned}$$

откуда $TETA_0 = \text{ATAN}(0.24713) = 0.2423$ рад. = 13.88 град.

Напомним, что во всех расчетах на ЭВМ используются значения углов, выраженные в радианах. Отметим также, что в общем случае для определения правильного угла $TETA_0$ из верхнего или нижнего уравнения типа (11.8) следует определить еще какую-нибудь функцию угла $TETA_0$ (кроме тангенса): его косинус или синус (подробнее смотри дополнение 4 в [3]).

4) Получим дифференцированием уравнений связей (11.4)-(11.5) систему соотношений для определения линейной скорости S' и угловых скоростей звеньев механизма, которые используют для этого уже найденные значения координаты S и углов для данного времени T (см. (7)-(8) в [21, с. 120]). Для большей наглядности поместим неизвестные переменные с обозначениями скоростей в конце каждого слагаемого и дополнительно пронумеруем уравнения по порядку:

$$1. -A \cdot \text{SIN}(FI) \cdot FI' - B \cdot \text{SIN}(PSI) \cdot PSI' = Xa'; \quad (11.9)$$

$$2. A \cdot \text{COS}(FI) \cdot FI' - B \cdot \text{COS}(PSI) \cdot PSI' = Ya';$$

$$3. -0.31 \cdot A \cdot \text{SIN}(FI) \cdot FI' - S \cdot \text{SIN}(PSI) \cdot PSI' + \text{COS}(PSI) \cdot S' + C \cdot \text{SIN}(TETA) \cdot TETA' = 0; \quad (11.10)$$

$$4. 0.31 \cdot A \cdot \text{COS}(FI) \cdot FI' - S \cdot \text{COS}(PSI) \cdot PSI' - \text{SIN}(PSI) \cdot S' - C \cdot \text{COS}(TETA) \cdot TETA' = 0.$$

Присвоим неизвестным скоростям, входящим в систему уравнений (11.9) — (11.10), соответствующие идентификаторы:

$$XV(1) = FI', \quad XV(2) = PSI', \quad XV(3) = S', \quad XV(4) = TETA'. \quad (11.11)$$

Соотношения (11.9) и (11.10) представляют собой систему алгебраических уравнений, которая для каждого момента времени T линейна относительно

но скоростей. С учетом использования соответствующих идентификаторов их можно представить в матричной форме в следующем виде:

$$(AV) (XV) = BV , \quad (11.12)$$

где: AV — заданная квадратная матрица коэффициентов перед неизвестными, в которой выписаны только отличные от нуля значения элементов матрицы $AV(I,J)$;

XV — неизвестная матрица-столбец с N компонентами — искомые угловые и линейные скорости в (11.11): $XV(1)=FI'$, $XV(2)=PSI'$, $XV(3)=S'$, $XV(4)=TETA'$;

BV — заданная матрица-столбец с N компонентами — правые части уравнений (11.9)-(11.10): $BV(1)=Xa'$, $BV(2)=Ya'$, $BV(3)=0.$, $BV(4)=0.$

Положение элемента $AV(I,J)$ в матрице AV характеризуется двойным индексом. Первый индекс I означает номер строки, в которой стоит элемент $AV(I,J)$, т.е. это порядковый номер уравнения проекций скоростей 1-4. Вторым индексом J означает номер столбца, т.е. номер идентификатора $XV(J)$, при котором стоит элемент $AV(I,J)$. Значение J также меняется от 1-го до 4-х. Отметим, что нумерация строк производится сверху вниз, а столбцов — слева направо. Положение элемента матрицы — столбца $BV(I)$ характеризуется только номером I -й строки, в которой стоит элемент $BV(I)$.

Систему (11.9) — (11.10) с учетом уравнений (11.12) для любого варианта задания К-9 можно представить, например, в таком общем виде, удобном для дальнейшего решения:

$$\begin{aligned} AV(1,1) * FI' + AV(1,2) * PSI' + AV(1,3) * S' + AV(1,4) * TETA' &= BV(1) ; \\ AV(2,1) * FI' + AV(2,2) * PSI' + AV(2,3) * S' + AV(2,4) * TETA' &= BV(2) ; \\ AV(3,1) * FI' + AV(3,2) * PSI' + AV(3,3) * S' + AV(3,4) * TETA' &= BV(3) ; \\ AV(4,1) * FI' + AV(4,2) * PSI' + AV(4,3) * S' + AV(4,4) * TETA' &= BV(4) . \end{aligned} \quad (11.13)$$

Если в уравнениях отсутствуют слагаемые, содержащие какие-либо неизвестные, то это означает, что соответствующий коэффициент $AV(I,J)$ равен нулю.

Для типового примера решения задания К-9 в [21] определим элементы матрицы коэффициентов $AV(I,J)$ и матрицы-столбца свободных членов $BV(I)$, для чего воспользуемся соответствием систем уравнений (11.9)-(11.10) и (11.13) между собой:

$$\begin{aligned} AV(1,1) &= -A * SIN(FI) , \quad AV(1,2) = -B * SIN(PSI) , \quad AV(1,3) = 0. , \\ AV(1,4) &= 0. , \quad AV(2,1) = A * COS(FI) , \quad AV(2,2) = -B * COS(PSI) , \\ AV(2,3) &= 0. , \quad AV(2,4) = 0. , \quad AV(3,1) = -0.31 * A * SIN(FI) , \\ AV(3,2) &= -S * SIN(PSI) , \quad AV(3,3) = COS(PSI) , \\ AV(3,4) &= C * SIN(TETA) , \\ AV(4,1) &= 0.31 * A * COS(FI) , \quad AV(4,2) = -S * COS(PSI) , \\ AV(4,3) &= -SIN(PSI) , \quad AV(4,4) = -C * COS(TETA) ; \\ BV(1) &= Xa' , \quad BV(2) = Ya' , \quad BV(3) = 0. , \quad BV(4) = 0. \end{aligned} \quad (11.14)$$

где Xa' и Ya' определяются дифференцированием уравнений движения захвата (11.1):

$$Xa' = -0.2 \text{ м/с}, \quad Ya' = -0.089 \text{ м/с}.$$

Заметим, что из уравнений (11.9)-(11.10), зная значения углов и координаты S для данного времени T , можно определить значения неизвестных скоростей.

5) Аналогичным дифференцированием уравнений (11.9)-(11.10) получим систему соотношений для определения линейного ускорения S'' и угловых ускорений звеньев механизма, которые используют для этого не только значения координаты S и углов FI , PSI и $TETA$, но и найденные уже для данного времени T значения их скоростей S' , FI' , PSI' и $TETA'$.

Перепишем их в виде, тождественном уравнениям (9)-(10) в [21, с. 120], соблюдая следующую последовательность неизвестных FI'' , PSI'' , S'' , $TETA''$ при расположении слагаемых. Также для большей наглядности поместим эти неизвестные переменные с обозначениями ускорений в конце каждого слагаемого и дополнительно пронумеруем уравнения по порядку:

$$1. -A * \sin(FI) * FI'' - B * \sin(PSI) * PSI'' = \\ = Xa'' + A * \cos(FI) * FI' ** 2 + B * \cos(PSI) * PSI' ** 2; \quad (11.15)$$

$$2. A * \cos(FI) * FI'' - B * \cos(PSI) * PSI'' = \\ = Ya'' + A * \sin(FI) * FI' ** 2 - B * \sin(PSI) * PSI' ** 2;$$

$$3. -0.31 * A * \sin(FI) * FI'' - S * \sin(PSI) * PSI'' + \cos(PSI) * S'' + \\ + C * \sin(TETA) * TETA'' = 0.31 * A * \cos(FI) * FI' ** 2 + \\ + 2 * \sin(PSI) * PSI' * S' + S * \cos(PSI) * PSI' ** 2 - C * \cos(TETA) * TETA' ** 2; \quad (11.16)$$

$$4. 0.31 * A * \cos(FI) * FI'' - S * \cos(PSI) * PSI'' - \sin(PSI) * S'' - \\ - C * \cos(TETA) * TETA'' = 0.31 * A * \sin(FI) * FI' ** 2 + \\ + 2 * \cos(PSI) * PSI' * S' - S * \sin(PSI) * PSI' ** 2 - C * \sin(TETA) * TETA' ** 2.$$

По аналогии со скоростями, присвоим неизвестным ускорениям, входящим в систему уравнений (11.15) — (11.16), соответствующие идентификаторы:

$$XW(1) = FI'', \quad XW(2) = PSI'', \quad XW(3) = S'', \quad XW(4) = TETA''. \quad (11.17)$$

Соотношения (11.15) и (11.16) представляют собой также систему алгебраических уравнений, которая для каждого момента времени T линейна относительно ускорений, поэтому их можно представить с учетом использования соответствующих идентификаторов в матричной форме в следующем виде:

$$(AW) (XW) = BW, \quad (11.18)$$

где: AW — заданная квадратная матрица коэффициентов перед неизвестными, в которой выписаны только отличные от нуля значения элементов матрицы $AW(I,J)$;

XW — неизвестная матрица-столбец с N компонентами — искомые угловые и линейные ускорения: $XW(1)=FI''$, $XW(2)=PSI''$, $XW(3)=S''$, $XW(4)=TETA''$;

BW — заданная матрица-столбец с N компонентами — правые части уравнений (9) - (10): $BW(1)=Xa''$, $BW(2)=Ya''$, $BW(3)=0.$, $BW(4)=0$.

Заметим, что из уравнений (11.15)-(11.16), зная значения углов, координаты S и их скоростей для данного времени T , можно определить значения неизвестных ускорений.

Для дальнейшего решения нам будут также нужны значения проекций ускорений точки A Xa'' и Ya'' , которые определяются двойным дифференцированием уравнений движения захвата (11.1): $Xa'' = 0.0$ м/с², $Ya'' = 0.0$ м/с².

Сравнением соответственно левых частей уравнений (11.9)-(11.10) и (11.15)-(11.16) устанавливается, что матрицы AV и AW для систем алгебраических уравнений относительно скоростей и ускорений одинаковы.

Аналогичное сравнение правых частей этих равенств говорит о их различии (BV и BW), поэтому выпишем элементы матрицы-столбца BW :

$$\begin{aligned} BW(1) &= A * FI' * * 2 * COS(FI) + B * PSI' * * 2 * COS(PSI) ; \\ BW(2) &= A * FI' * * 2 * SIN(FI) - B * PSI' * * 2 * SIN(PSI) ; \\ BW(3) &= 0.31 * A * FI' * * 2 * COS(FI) + 2. * S' * PSI' * SIN(PSI) + \\ &+ S * PSI' * * 2 * COS(PSI) - C * TETA' * * 2 * COS(TETA) ; \\ BW(4) &= 0.31 * A * FI' * * 2 * SIN(FI) + 2. * S' * PSI' * COS(PSI) - \\ &- S * PSI' * * 2 * SIN(PSI) - C * TETA' * * 2 * SIN(TETA) . \end{aligned} \quad (11.19)$$

Подчеркнем, что никакие тождественные преобразования на данных этапах решения (как умножение на -1 и т. п.) недопустимы, так как матрицы коэффициентов перед неизвестными относительно ускорений и скоростей должны совпадать: $AW = AV$, что использовано в программе расчета.

Не может быть изменена и последовательность неизвестных ускорений при записи левых частей уравнений (11.15)-(11.16) (FI'' , PSI'' , S'' , $TETA''$), так как она также заложена в программе расчета на ЭВМ.

11.2. Общее описание алгоритма решения задачи

Для любого момента времени T , которому соответствует I -е положение механизма, мы можем, зная значения углов и координаты S , последовательным решением СЛАУ (11.9)-(11.10) и (11.15)-(11.16) определить соответствующие значения скоростей и ускорений, что удобно сделать с помощью ЭВМ.

Для определения значений углов и координаты S для следующего $(I+1)$ -го положения механизма при времени $T+HT$ можно воспользоваться разложением функции в ряд Тейлора (11.20), где через HT обозначено приращение времени или шаг расчета. Можно с достаточной точностью ограничиться

лишь двумя первыми членами ряда Тейлора, тогда для возможности определения следующих (I+1)-х значений углов и координаты S потребуется только знание их 1-х и 2-х I-х производных (т. е. скоростей и ускорений, уже определенных нами для предыдущего времени T):

$$\begin{aligned} FI(I+1) &= FI(I) + FI'(I) * HT + FI''(I) * HT**2/2 ; \\ PSI(I+1) &= PSI(I) + PSI'(I) * HT + PSI''(I) * HT**2/2 ; \quad (11.20) \\ S(I+1) &= S(I) + S'(I) * HT + S''(I) * HT**2/2 ; \\ TETA(I+1) &= TETA(I) + TETA'(I) * HT + TETA''(I) * HT**2/2 ; \end{aligned}$$

Теперь после однократного выполнения этапов 1)-5) (определения стартовых начальных условий S0 и TETA0 и получения выражений для элементов матриц AV=AW и матриц-столбцов BV и BW), весь последующий расчет неоднократного его повторения путем численного решения систем уравнений (11.9)-(11.10) и (11.15)-(11.16) с изменяющимися после каждого шага значениями углов, координаты S, их скоростей и ускорений хорошо выполнит ЭВМ.

11.3. Подготовка подпрограмм K9AV и K9BW

Чтобы определить те вычисления, которые будут производить подпрограммы SUBROUTINE K9AV и K9BW, нужно написать соответствующие участки программы, содержащие необходимые операторы присваивания, задающие выражения для элементов соответствующих матриц AV и BW. Перед ними следует расположить слово SUBROUTINE и наименование подпрограммы (K9AV или K9BW) со списком использованных формальных параметров (FI,PSI,S,TETA,AV или FI,PSI,S,TETA,FIP,PSIP,SP,TETAP,BW соответственно), служащих для информационной связи с основной программой K9MP (см. п. 12.1.3).

После заголовка подпрограммы следуют операторы описания. Оператор общих областей COMMON (см. п. 12.3), позволяет пользоваться в подпрограммах K9AV и K9BW введенными в основную программу значениями констант A, B, C, определяющих размеры соответствующих звеньев механизма. Операторы DIMENSION AV(4,4) и DIMENSION BW(4) задают описания используемых массивов двумерной матрицы AV в подпрограмме K9AV и одномерной BW в подпрограмме K9BW, содержащих 16 и 4 элемента. Заканчивается каждая подпрограмма операторами RETURN и END.

Подпрограмма SUBROUTINE K9AV задает выражения для элементов матрицы коэффициентов перед неизвестными SLAY (11.9)-(11.10), которые представлены выписанными отдельно в (11.14):

```

SUBROUTINE K9AV (FI, PSI, S, TETA, AV)
COMMON /ABC/A, B, C
DIMENSION AV (4, 4)
AV (1, 1) = -A * SIN (FI)
AV (1, 2) = -B * SIN (PSI)
AV (1, 3) = 0.
AV (1, 4) = 0.
AV (2, 1) = A * COS (FI)
AV (2, 2) = -B * COS (PSI)
AV (2, 3) = 0.
AV (2, 4) = 0.
AV (3, 1) = -0.31 * A * SIN (FI)
AV (3, 2) = -S * SIN (PSI)
AV (3, 3) = COS (PSI)
AV (3, 4) = C * SIN (TETA)
AV (4, 1) = 0.31 * A * COS (FI)
AV (4, 2) = -S * COS (PSI)
AV (4, 3) = -SIN (PSI)
AV (4, 4) = -C * COS (TETA)
RETURN
END

```

(11.21)

Подпрограмма SUBROUTINE K9BW задает выражения для свободных членов (правых частей) СЛАУ (11.15)-(11.16), которые представлены выписанными отдельно в (11.19):

```

SUBROUTINE K9BW (FI, PSI, S, TETA,
*   FIP, PSIP, SP, TETAP, BW)
COMMON /ABC/A, B, C
DIMENSION BW (4)
BW (1) = A * FIP ** 2 * COS (FI) + B * PSIP ** 2 * COS (PSI)
BW (2) = A * FIP ** 2 * SIN (FI) - B * PSIP ** 2 * SIN (PSI)
BW (3) = 0.31 * A * FIP ** 2 * COS (FI) + 2. * SP * PSIP *
*   SIN (PSI) + S * PSIP ** 2 * COS (PSI) - C * TETAP ** 2 * COS (TETA)
BW (4) = 0.31 * A * FIP ** 2 * SIN (FI) + 2. * SP * PSIP * COS (PSI) -
*   S * PSIP ** 2 * SIN (PSI) - C * TETAP ** 2 * SIN (TETA)
RETURN
END

```

(11.22)

Отметим, что в подпрограммах K9AV и K9BW можно не использовать оператор COMMON /ABC/A,B,C и при задании элементов матриц AV и BW пользоваться численными значениями этих констант. Однако это не освобождает от необходимости ввода размеров звеньев во второй строке исходных данных, так как они проверяются обеими программами (K9MP и DK9MP) на их соответствие условию вашего номера варианта.

11.4. Подготовка данных для работы программы К9МР

Исходные данные для работы программы К9МР имеют следующую структуру (числовые значения приводятся для рассматриваемого примера и представлены в полном виде в файле (11.24)):

1-я строка: перечисляются номера группы NG (114418), задания по кинематике NZ (9), варианта NW (31): 114418,9,31;

2-я строка: задаются размеры звеньев a (1.3 м), b (1.1), c (0.55 м): 1.3,1.1,0.55;

3-я строка: указываются для точки А при $T=0$ значения проекций на оси x и y скорости (-0.2 и -0.089) и ускорения (0 и 0): -0.2,-0.089,0,0;

4-я строка: перечисляются начальные значения углов (в радианах) и расстояния S (в метрах) при $T=0$ в следующей последовательности: FI, PSI, S, TETA. Для FI и PSI они даны в таблице и только переводятся в радианы, а для S и TETA определяются из системы (11.8): 1.082, 0.576, 0.41105, 0.24227;

5-я строка: указываются значения начального и конечного времени расчета TN и ТК, а также приращения аргумента HT и шага печати результатов расчета HPR. В программе предварительно задается: TN=0.0; ТК=1.0; HT=0.2; HPR=0.2; .

6-я строка: перечисляются допустимые значения относительных погрешностей расчета EPSX и определения начальных условий (S0 и TETA0) EPS0, а также абсолютной ошибки задания исходных данных DELC. В программе предварительно задается: EPSX=1.0, EPS0=1.0, DELC=0.01;

7-я строка: указываются значения следующих параметров: переключения формата печати IFORM и управления выбора значений начальных условий (S0 и TETA0 — IPARQU) или исходных данных (звена C и уравнения движения Ya — IPARID). В программе им присваиваются значения: IFORM=1, IPARQU=1, IPARID=1.

При IFORM=2 происходит увеличение числа выводимых на печать величин. IPARQU=2 приводит к использованию для сравнения с вашими значениями величин S0 и TETA0, соответствующих отрицательному корню квадратного уравнения, получающемуся при решении системы (11.8) (если для этого существует физическая возможность в данном варианте). При IPARID=2 для сравнения при определении погрешностей решения используются исправленные значения размера звена C и уравнения движения Ya в вариантах 1, 2, 6, 8, 16, 23, 24, 25 и 26 (см. дополнение 11.1).

Величины, вводимые в строках 5-7, предварительно задаются в программе. Они выбраны так, чтобы получать оптимальные (в первом приближении) численные значения искомым величин. Поэтому их рекомендуется использовать при первоначальном решении, поставив в этих

строках знак наклонной черты “/”. Для рассматриваемого типового примера исходные данные для работы программы К9МР в самой простой форме будут иметь вид:

```
114418, 9, 31
1.3, 1.1, 0.55
-0.2, -0.089, 0., 0.
1.082, 0.576, 0.41105, 0.24227
/
/
/
```

(11.23)

Знак наклонной черты “/” указывает на переход к следующей строке данных. В результате этого предварительно заданные значения вводимых в этой строке величин в памяти ЭВМ не изменяются. Поэтому данные примера (11.23) эквивалентны (11.24), в котором вместо отсутствующих данных в строках 5-7 согласно вышеописанной структуре указаны предварительно задаваемые в программе значения соответствующих величин:

```
114418, 9, 31
1.3, 1.1, 0.55
-0.2, -0.089, 0., 0.
1.082, 0.576, 0.41105, 0.24227
0., 1., 0.2, 0.2
1., 1., 0.01
1, 1, 1
```

(11.24)

После получения достаточно правильного решения, следует варьированием значений величин, вводимых в строках 5-7, изучить дополнительные возможности программы К9МР. Для изменения значения какой-либо величины в соответствующей ей позиции данной строки следует указать нужное число. При этом возможны оба способа упрощения записи (10.10) и (10.12).

В программе предусмотрена защита от необдуманных действий при варьировании данных в строках 5-6, могущих повлечь за собой резкое увеличение выходной печати (до 22-х положений механизма) или времени счета (до 100000 повторений расчета). При превышении этих значений соответствующие переменные принимают свои предварительно задаваемые в программе значения.

11.5. Работа с удвоенной точностью с использованием программы DK9МР

Для работы с удвоенной точностью следует использовать готовую программу DK9МР, работа с которой полностью аналогична описанной

выше работе с программой K9MP. Матрицы AV и BW соответственно в файлах dk9av.for и dk9bw.for и исходные данные в файле dk9mp.dat должны быть представлены с учетом дополнительных требований, возникающих при использовании удвоенной точности (см. п. 15.3). Для рассматриваемого типового примера это может иметь следующий вид:

```

SUBROUTINE DK9AV (FI, PSI, S, TETA, AV)
  IMPLICIT REAL *8 (A-H, O-Z)
  COMMON /ABC/A, B, C
  DIMENSION AV (4, 4)
  AV (1, 1) = -A*DSIN (FI)
  AV (1, 2) = -B*DSIN (PSI)
  AV (1, 3) = 0. D0
  AV (1, 4) = 0. D0
  AV (2, 1) = A*DCOS (FI)
  AV (2, 2) = -B*DCOS (PSI)
  AV (2, 3) = 0. D0
  AV (2, 4) = 0. D0
  AV (3, 1) = -0. 31D0*A*DSIN (FI)
  AV (3, 2) = -S*DSIN (PSI)
  AV (3, 3) = DCOS (PSI)
  AV (3, 4) = C*DSIN (TETA)
  AV (4, 1) = 0. 31D0*A*DCOS (FI)
  AV (4, 2) = -S*DCOS (PSI)
  AV (4, 3) = -DSIN (PSI)
  AV (4, 4) = -C*DCOS (TETA)
  RETURN
  END

```

(11.25)

```

SUBROUTINE DK9BW (FI, PSI, S, TETA,
*   FIP, PSIP, SP, TETAP, BW)
  IMPLICIT REAL *8 (A-H, O-Z)
  COMMON /ABC/A, B, C
  DIMENSION BW (4)
  BW (1) = A*FIP**2*DCOS (FI) + B*PSIP**2*DCOS (PSI)
  BW (2) = A*FIP**2*DSIN (FI) - B*PSIP**2*DSIN (PSI)
  BW (3) = 0. 31D0*A*FIP**2*DCOS (FI) + 2. D0*SP*PSIP*
*   DSIN (PSI) + S*PSIP**2*DCOS (PSI) - C*TETAP**2*DCOS (TETA)
  BW (4) = 0. 31D0*A*FIP**2*DSIN (FI) + 2. D0*SP*PSIP*DCOS (PSI) -
*   S*PSIP**2*DSIN (PSI) - C*TETAP**2*DSIN (TETA)
  RETURN
  END

```

114418, 9, 31

1. 3D0, 1. 1D0, 0. 55D0

-0. 2D0, -0. 089D0, 0. D0, 0. D0

```
1.082D0,0.576D0,0.41105D0,0.24227D0
```

(11.27)

/

/

/

Представление исходных данных (11.27) в полной форме для работы с удвоенной точностью примет вид:

114418,9,31

1.3D0,1.1D0,0.55D0

-0.2D0,-0.089D0,0.D0,0.D0

1.082D0,0.576D0,0.41105D0,0.24227D0

(11.28)

0.D0,1.D0,0.2D0,0.2D0

1.D0,1.D0,0.01D0

1,1,1

При варьировании величин, задаваемых в строках данных 5-7, возможны оба способа упрощения записи (10.10) и (10.12), только переменные должны указываться удвоенной точности.

11.6. Замечания по использованию программ K9MP и DK9MP на ПЭВМ

В п. 10.6 описана возможная последовательность сеанса работы по решению заданий по кинематике на ПЭВМ с помощью программы KINMP. При использовании программ K9MP или (DK9MP) последовательность действий остается аналогичной, а во всех командах вместо имен программы KINMP и подпрограммы KINU следует указывать соответственно имена нужных программ K9MP (или DK9MP) и подпрограмм K9AV, K9BW (или DK9AV, DK9BW).

Единственным отличием при работе с программой K9MP (или DK9MP) будет то, что она требует для своей работы наличия двух подпрограмм K9AV и K9BW (или DK9AV и DK9BW). Это приводит к тому, что перед сборкой выполняемого модуля программы k9mp.exe (или dk9mp.exe) и запуском его на выполнение нужно протранслировать обе подпрограммы K9AV и K9BW (или DK9AV и DK9BW), добившись для каждой из них нормального завершения выполнения трансляции с отсутствием диагностических сообщений.

Также все команды для работы с подпрограммой KINU, описанные в п. 10.6, следует повторять аналогичным образом дважды для подпрограмм K9AV и K9BW (или DK9AV и DK9BW при использовании удвоенной точности). С учетом этих замечаний последовательность действий для работы с программами K9MP и DK9MP полностью совпадает с описанной в п. 10.6.

11.7. Описание результатов работы программ К9МР и ДК9МР, их проверка и поиск ошибок

Представление результатов работы программ К9МР и ДК9МР, отличающихся только используемой точностью вычислений, полностью совпадает. После печати заголовка выводимой информации сами результаты расчета программ К9МР и ДК9МР выводятся в виде таблицы. Указываются полученные значения угловых и линейных скоростей и ускорений звеньев механизма манипулятора для различных его положений от начального времени ТН до конечного времени ТК.

Задание К-9 не предусматривает аналитического решения для всех положений механизма. Поэтому программой К9МР выполняются различные проверки вводимых исходных данных и результатов решения. По ним определяются соответствующие погрешности, также указываемые в распечатке.

Если значения этих погрешностей выходят за установленные для них соответствующие диапазоны (которые можно менять), то появляются аварийные сообщения, после которых прекращается выполнение программ. В этом случае следует внимательно проверить все этапы постановки задачи, ее решения и подготовки вводимых данных, а также правильность их введения по распечатке, после чего нужно повторить выход на ЭВМ.

Отметим, что наибольшую трудность представляют собой исправление и поиск ошибок, возникших из-за неправильного определения начальных значений координаты S и угла ТЕТА, а также при определении или задании матриц коэффициентов при неизвестных AV и свободных членов BW.

Убедившись в правильности всех выражений для элементов матриц AV и BW, следует варьированием величины шага расчета НТ, вводимого третьим в пятой строке файла данных, постараться уменьшить относительную ошибку определения скоростей и ускорений звеньев механизма при последующих выходах на ЭВМ, а потом перейти к вычислениям с удвоенной точностью с использованием программы ДК9МР (см. п. 11.5).

Ранее (см. п. 10.4) вы выполняли подобное исследование при использовании для численного дифференцирования формул в конечных разностях (10.1)-(10.2), которые относятся к числу неустойчивых алгоритмов (шаг дифференцирования ДТ в знаменателе и в формуле (10.2) даже в квадрате). Поэтому при уменьшении ДТ в этом случае погрешность расчета сначала уменьшается, а потом резко возрастает вплоть до катастрофической потери точности.

При использовании формулы Тейлора (11.20) мы имеем дело с устойчивым алгоритмом, при котором шаг расчета НТ находится всегда в числителе и при уменьшении своего значения приводит только к уменьшению ошибок.

Дополнение 11.1. Следует отметить, что вследствие досадных неточностей некоторые варианты задания К-9 не обеспечивают согласованность начальных условий (варианты 2, 6, 8, 23, 25 и 26). Это приводит к геометрическому разрыву механизма (невозможности его существования с данными размерами) сразу в начальный момент времени. В этом случае соответствующие ошибки расчета возникают сразу при $T=0$ и в дальнейшем имеют примерно такой же порядок. Неточности этого типа будем называть геометрическими.

Они выявляются после проверки согласованности задания начальных условий (см. уравнения (11.6) и (11.7)). Для их устранения нужно менять значения размеров исходных данных или свободных констант в уравнениях движения таблицы 37 в [21, с. 115], что можно сделать, например, следующим образом:

Т а б л и ц а 11.1.

**Основные геометрические несогласованности
исходных данных индивидуального задания К-9**

№ варианта	Неисправленные значения	Исправленные значения
2	$C = 0.24$	$C = 0.4$
6	$C = 0.6$	$C = 0.8$
8	$Y_a = 0.0947 + 0.3 \cdot T$	$Y_a = 0.09047 + 0.3 \cdot T$
23	$C = 0.4$	$C = 0.6$
25	$Y_a = 0.6839$	$Y_a = 0.6829$
26	$C = 0.34$	$C = 0.54$

Второй вид неточностей, которые мы будем называть кинематическими, возникает вследствие несогласованности величины скорости или ее направления. Поэтому механизм не может обеспечить нужной траектории движения в течение требуемого промежутка времени от 0 до 1 сек. В этом случае относительная погрешность расчета с течением времени нарастает, оставаясь меньше 1 %, а при некотором значении T резко увеличивается (когда начинается значительное расхождение реальной траектории захвата точки А от требуемой), что случается в вариантах 1, 16 и 24.

Для исправления кинематических неточностей в задании К-9 нужно уменьшить длину отрезка траектории захвата, проходимого им за заданное время. Это

можно сделать различными способами, например уменьшением величины коэффициента или изменением знака слагаемого с “Т” в уравнении движения.

Т а б л и ц а 11.2.

**Основные кинематические несогласованности
исходных данных индивидуального задания К-9**

№ варианта	Неисправленные значения	Исправленные значения
1	$Y_a = 0.7436 - 0.3 \cdot T$	$Y_a = 0.7436 - 0.1 \cdot T$
16	$Y_a = 0.5961 + 0.2 \cdot T$	$Y_a = 0.5961 - 0.2 \cdot T$
24	$Y_a = 1.2260 + 0.2 \cdot T$	$Y_a = 1.2260 - 0.2 \cdot T$

С исправленными значениями исходных данных выполнение указанных в таблицах 11.1 и 11.2 вариантов не отличается от остальных для задания К-9 в [21].

ЧАСТЬ 5. СОСТАВЛЕНИЕ ПРОГРАММ НА ФОРТРАНЕ ДЛЯ РЕШЕНИЯ ЗАДАЧ И ВЫПОЛНЕНИЯ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ ПО КИНЕМАТИКЕ

ГЛАВА 12. ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ ПО АЛГОРИТМИЧЕСКОМУ ЯЗЫКУ ФОРТРАН

Часто возникает необходимость в многократных выполнениях действий при различных значениях входящих в них параметров. Для уменьшения размера программы целесообразно выделить повторяющиеся операции в подпрограмму, где описывается процедура их выполнения при формальных параметрах. В основной программе в требуемых местах осуществляется обращение к подпрограмме при заданных значениях фактических параметров.

В языке Фортран используются следующие подпрограммы: системные (или стандартные) функции, оператор-функции (или операторные функции), подпрограммы-функции и подпрограммы общего вида. Различие заключается в их описании, а также в реализуемых ими возможностях.

Системные функции не нужно описывать. Для их использования достаточно указать название функции со стоящим после него в круглых скобках аргументом, в качестве которого может быть постоянная, переменная или любое арифметическое выражение, включающее другие функции или ту самую функцию (см. п. 7.3).

Остальные функции и подпрограммы будут рассмотрены ниже с примерами их практического использования, естественно возникающими при решении задач по кинематике точки с использованием численного дифференцирования.

Формальные и фактические параметры для любых типов подпрограмм должны согласовываться по количеству, порядку следования, типу и длине.

12.1. Функции и подпрограммы

12.1.1. Оператор-функция

Для записи подпрограммы, состоящей из одного оператора, применяется оператор-функция:

$$F(X_1, X_2, \dots, X_N) = AW$$

где F — имя (идентификатор) оператора-функции; X_1, X_2, \dots, X_N — формальные параметры, которые могут быть только простыми переменными; AW — любое арифметическое или логическое выражение.

Имя оператора-функции подчиняется тем же правилам, что и имена переменных, т.е. имя должно содержать набор от 1 до 6 букв или цифр, начинающийся с буквы; целый и действительный тип можно определять косвенно с помощью неявного описания типа, т.е. по первой букве.

Все описания операторов-функций должны находиться в программе до появления первого исполняемого оператора (см. п. 7.10). Они являются только определением функции и сами по себе не обуславливают никаких вычислений.

Например, для возможности неоднократного вычисления при различных T значений координат X и Y , задающих уравнения движения материальной точки (10.14), можно записать следующие операторы-функции:

$$X(T) = 4 * T \quad (12.1)$$

$$Y(T) = 16 * T ** 2 - 1. \quad (12.2)$$

Здесь именами операторов-функций являются однобуквенные идентификаторы X и Y , а список формальных параметров состоит из одного аргумента T .

Формальные параметры не занимают никакого места в памяти. Они могут быть такими же, как и наименования переменных в другом месте программы, и совершенно несущественны (за исключением указания типа). Они используются только для обозначения последовательности операций, которые нужно будет выполнить над соответствующими фактическими параметрами. Поэтому ничего не изменится, если в (12.1)–(12.2) вместо формального параметра T использовать, например, X_1 и X_2 :

$$X(X_1) = 4 * X_1 \quad (12.3)$$

$$Y(X_2) = 16 * X_2 ** 2 - 1. \quad (12.4)$$

Обращение к операторам-функциям (12.1)–(12.2) или (12.3)–(12.4) будет одинаковым.

Для обращения следует в любом нужном месте программы записать имя оператора-функции F, указав при этом нужные фактические параметры $F(A_1, A_2, \dots, A_N)$, которые могут быть константами, простыми или индексными переменными, выражениями. Результат присваивается имени оператора-функции, которое определяет его тип (с использованием соглашения по умолчанию или с помощью операторов явного описания типа).

Например, для вычисления значений рассматриваемых функций (10.14) в трех различных точках программы при $T+DT$, T и $T-DT$, необходимо просто записать обращение к этим операторам-функциям в виде: $X(T+DT)$, $X(T)$ и $X(T-DT)$ или $Y(T+DT)$, $Y(T)$ и $Y(T-DT)$ соответственно. При этом каждый раз указанный фактический параметр $T+DT$, T и $T-DT$, имеющий соответствующее значение в данном месте программы, будет подставлен вместо формального параметра (T в (12.1)-(12.2) или X_1 в (12.3) и X_2 в (12.4) — все равно). С ним в этом месте программы будут проделаны все необходимые вычисления и определено значение функции:

$$AX = (X(T+DT) - 2 * X(T) + X(T-DT)) / DT ** 2 \quad (12.5)$$

$$AY = (Y(T+DT) - 2 * Y(T) + Y(T-DT)) / DT ** 2 \quad (12.6)$$

Отметим, что выражение в правой части определения оператора-функции может содержать переменные, которых нет в списке формальных параметров. Например, операторы-функции (12.1)-(12.2) можно записать, используя вместо чисел переменные A_1 , B , C . Эти переменные являются общими для всего программного модуля (в отличие от формального параметра T). Значения этих переменных должны быть определены последующими операторами, но обязательно до первого обращения к функции, где эти переменные используются, например, с применением оператора DATA (см. п. 7.7.1):

$$\begin{aligned} X(T) &= A_1 * T \\ Y(T) &= B * T ** 2 - C \end{aligned} \quad (12.7)$$

$$\begin{aligned} & \dots \dots \dots \\ & DATA A_1, B, C / 4., 16., 1. / \end{aligned} \quad (12.8)$$

$$\begin{aligned} AX &= (X(T+DT) - 2 * X(T) + X(T-DT)) / DT ** 2 \\ AY &= (Y(T+DT) - 2 * Y(T) + Y(T-DT)) / DT ** 2 \end{aligned}$$

Последние два оператора, иллюстрирующие обращение к операторам-функциям, остаются, конечно, без изменений и совпадают с операторами (12.5)-(12.6).

Выражение в правой части определения оператора-функции может содержать системные функции (см. п. 7.3), подпрограммы-функции (см. п. 12.1.2), а также другие операторы-функции, определенные

раньше. Например, нижеследующие операторы-функции эквивалентны операторам (12.1)–(12.2):

$$\begin{aligned} X(T) &= 4 * T \\ Y(T) &= X(T) ** 2 - 1. \end{aligned} \quad (12.9)$$

12.1.2. Подпрограмма-функция

Для организации подпрограмм, допускающих любое количество операторов, используется подпрограмма-функция. Ее структура имеет следующий основной вид:

```
FUNCTION F (X1, X2, . . . , XN)
<операторы подпрограммы>
F = . . .
RETURN
END
```

(12.10)

где F — имя (идентификатор) подпрограммы-функции, которое подчиняется тем же правилам, что и имена переменных и операторов-функций; X1, X2, . . . , XN — формальные параметры, которые могут быть простыми переменными, именами массивов и других подпрограмм. Если в качестве формального аргумента используется имя массива, то массив должен быть описан в подпрограмме-функции при помощи оператора DIMENSION. Оператор RETURN осуществляет возврат в вызывающую программу, а оператор END сигнализирует транслятору о том, что в данном программном модуле операторов больше нет.

Подпрограмма-функция является подпрограммой в полном смысле этого слова: наименования переменных и метки в ней полностью независимы от наименований переменных и меток в основной программе и в других подпрограммах. Это позволяет использовать ее с различными основными программами, а также отлаживать и транслировать независимо от них. Подпрограмма-функция оформляется в виде отдельной программной единицы и располагается или в отдельном одноименном файле, или после оператора END основной программы или другой подпрограммы.

Вычисления рассматриваемого примера (10.14) можно определить с использованием подпрограмм-функций следующим образом:

```
FUNCTION FX (T)
FX = 4 * T
RETURN
END
```

(12.11)

```

FUNCTION FY (T)
FY=16.*T**2-1. (12.12)
RETURN
END

```

В примерах (12.11)–(12.12) список формальных параметров состоит из одной переменной T, а FX и FY — имена соответствующих подпрограмм-функций. Их тип задан неявно, т.е. определяется первой буквой имени. Впереди оператора FUNCTION (в той же строке) возможно также использование явного описателя типа для подпрограммы-функции: INTEGER, REAL, DOUBLE PRECISION (см. операторы 110, 140 и 170 п. 15.4).

Подпрограмма-функция выполняется после обращения к ней из другой программной единицы. Для этого необходимо только написать наименование функции там, где необходимо получить ее значение, и после него подставить в скобки соответствующий список фактических параметров F(A1,A2,...,AN). С их значениями будут проведены вычисления вместо формальных параметров X1,X2,...,XN, указанных в операторе FUNCTION. В результате переменной, имеющей то же имя, что и подпрограмма (FX в (12.11) и FY в (12.12)), присваивается вычисленное значение функции и управление по оператору RETURN передается к месту ее вызова.

В списке формальных и фактических параметров можно указывать не только входные, но и выходные величины. Однако имени подпрограммы присваивается значение только одного результата.

Вычисления примеров (12.5) и (12.6) можно выполнить с использованием подпрограмм-функций (12.11)–(12.12), например, следующим образом:

$$AX = (FX(T+DT) - 2. * FX(T) + FX(T-DT)) / DT**2 \quad (12.13)$$

$$AY = (FY(T+DT) - 2. * FY(T) + FY(T-DT)) / DT**2 \quad (12.14)$$

Также ничего не изменится, если при определении подпрограмм-функций (12.11)–(12.12), вместо идентификатора формального параметра переменной T использовать любой идентификатор вещественной переменной, например, X1 и X2 (как использовано в операторах-функциях (12.3)–(12.4)):

```

FUNCTION FX (X1)
FX=4.*X1 (12.15)
RETURN

```

```

END

```

```

FUNCTION FY (X2)
FY=16.*X2**2-1. (12.16)
RETURN

```

```

END

```


В этом случае подпрограмма SUBROUTINE может возвращать результаты вычислений в вызывающую программу через оператор общих областей COMMON (см. п. 12.3), или не возвращать результаты вычислений вообще, выполняя вспомогательную роль (служебную или сервисную), например, печать результатов вычислений и т.п.

Вычисления рассматриваемого примера (10.14) можно определить с использованием подпрограммы SUBROUTINE, например, с именем K1UD следующим образом:

```

SUBROUTINE K1UD(T, X, Y)
  X=4.*T
  Y=16.*T**2-1.
  RETURN
END

```

(12.19)

Список формальных параметров в примере (12.19) состоит из трех переменных: входной величиной является время T, а значения X и Y, определяемые в результате вычислений по подпрограмме (12.19), являются выходными формальными параметрами.

Для обращения к подпрограмме SUBROUTINE используется специальный оператор CALL, после которого указывается наименование подпрограммы F и перечисляются в скобках фактические параметры A1, A2, ..., AN:

```
CALL F(A1, A2, . . . , AN)
```

(12.20)

Правила записи фактических параметров и их соответствие с формальными те же, что и у подпрограммы-функции.

Чтобы вычислить значения X и Y, определенные в программе (12.19), при некоторых значениях, например, T+DT, T или T-DT, где значения T и DT заданы предварительно в программе, следует записать в нужном месте оператор CALL в соответствующем виде:

```

CALL K1UD(T+DT, X, Y)
CALL K1UD(T, X, Y)
CALL K1UD(T-DT, X, Y)

```

(12.21)

В примере (12.21) подпрограмма (12.19) выполняется трижды и переменные X и Y каждый раз принимают значения функций, вычисленных при использовании фактических параметров T+DT, T и T-DT соответственно.

Для подпрограмм SUBROUTINE наименования формальных параметров, как и в случае операторов-функций и подпрограмм функций, со-

вершено несущественны (кроме указания типа переменных). Поэтому ничего не изменится, если при определении подпрограммы SUBROUTINE (12.19) вместо идентификаторов формальных параметров T, X, Y использовать любые другие идентификаторы вещественного типа, например, X1, X2, X3:

```
SUBROUTINE K1UD(X1,X2,X3)
X2=4.*X1
X3=16.*X1**2-1.
RETURN
END
```

(12.22)

Обращение к подпрограммам (12.19) и (12.22) будет также одинаковым и может быть выполнено по любому из операторов (12.21).

Заметим, что формальные параметры, использованные для выражения одних функций, внутри той же подпрограммы могут быть использованы в качестве идентификаторов этих функций в любых других выражениях. Например, в обоих примерах (12.19) и (12.22) соответственно Y или X3 могут быть выражены через X или X2:

$$Y=X**2-1. \quad (12.23)$$

$$X3=X2**2-1. \quad (12.24)$$

Конечно, от такой замены в подпрограммах (12.19) и (12.22) обращения к ним (12.21) и результаты вычислений нисколько не изменятся.

В заключение сведем воедино основные характеристики функций, подпрограмм-функций и подпрограмм SUBROUTINE. Для полноты охвата и лучшего понимания системные функции, рассмотренные в п. 7.4, также включены в сводку, которая представлена на следующей странице.

После выполнения подпрограммы-функции по оператору RETURN происходит возврат в вызывающую программу в точку вызова подпрограммы-функции в том же операторе, после чего обращение к функции заменяется вычисленным значением функции и продолжается дальнейшее выполнение этого же оператора.

После выполнения подпрограммы SUBROUTINE по оператору RETURN тоже происходит возврат в вызывающую программу. Все выходные фактические параметры принимают значения, вычисленные при работе подпрограммы, после чего начинается дальнейшее выполнение программы с оператора, следующего за оператором процедуры CALL.

Основные характеристики функций и подпрограмм.

Системные функции	Оператор-функция	Подпрограмма-функция	Подпрограмма SUBROUTINE
1. Наименование:			
	1–6 букв или цифр; первой должна быть буква		
зарезервированы идентификаторы SIN, COS, TAN, ARSIN, ARCOS, ATAN, ...	при неявном способе задания типа первая буква определяет тип: начинающиеся с I,J,K,L,M,N — целые функции		первая буква не используется для указания типа подпрограмм
2. Способ определения:			
предусмотрены в программе- трансляторе	один арифметический оператор перед началом программы	любое количество операторов после заголовка подпрограммы:	
		FUNCTION	SUBROUTINE
3. Способ использования:			
написать наименование там, где необходимо вычислить значение функции			оператор CALL
4. Количество аргументов (в зависимости от определения):			
один или более			любое количество, в том числе и ни одного
5. Количество выходных значений:			
одно	одно соответствует наименованию функции	любое количество	

12.2. Оператор EXTERNAL

При описании подпрограммы-функции и подпрограммы SUBROUTINE в качестве формального параметра может использоваться имя другой подпрограммы. В этом случае соответствующим фактическим параметром может быть также только имя внешней функции или подпрограммы. Оно должно быть описано в вызывающей программе в операторе EXTERNAL, имеющем следующий общий вид:

```
EXTERNAL F1, F2 (12.25)
```

где F1, F2, ... — имена внешних подпрограмм-функций или подпрограмм SUBROUTINE, используемых в качестве фактических параметров.

Этот оператор позволит транслятору различить имена подпрограмм и имена переменных, перечисленных в качестве фактических параметров. Он располагается в вызывающем программном модуле перед первым выполняемым оператором (впереди определений операторов-функций, если они присутствуют в программе).

Оператор EXTERNAL часто применяется при использовании стандартных подпрограмм из пакета научных подпрограмм ПНП-SSP (см. п. 9.1). Это обусловлено тем, что многие подпрограммы пакета требуют указания наименования подпрограммы-функции пользователя как элемента списка фактических параметров оператора CALL. Например, при использовании подпрограммы DCAR для численного дифференцирования функций, заданных аналитически (см. п. 14.1.1):

```
SUBROUTINE DCAR(T, H, IH, FCT, D) (12.26)
```

Требуется составить подпрограмму дифференцируемой функции FCT и при обращении к подпрограмме DCAR (см. п. 14.2) указать ее имя в операторе CALL. Наименование функции FCT, находящейся в списке фактических параметров, также должно находиться в операторе EXTERNAL в начале программы:

```
EXTERNAL FCT
```

Дифференцирование уравнений движения материальной точки (10.14), заданных в виде подпрограмм-функций FX (12.11) и FY (12.12), можно осуществить, дважды обратясь к подпрограмме DCAR, указывая каждый раз в списке фактических параметров соответствующее наименование используемой функции FX или FY. Их имена следует указать в операторе EXTERNAL (см. операторы 10, 40 и 41 программы 14.1):

```
C      ОСНОВНАЯ ПРОГРАММА
      EXTERNAL FX, FY (12.27)
```

```
.....
```

```
CALL DCAR (T, DT, 1, FX, VX (M) ) (12.28)
```

```
CALL DCAR (T, DT, 1, FY, VY (M) ) (12.29)
```

Обращение (12.28)-(12.29), конечно, останется без изменения, если подпрограммы-функции FX и FY будут использоваться в виде (12.15)-(12.16), а не в описываемом (12.11)-(12.12), с другими обозначениями формальных параметров.

Обращение (12.28)-(12.29) и результаты вычислений останутся также без изменения, если подпрограмму-функцию FY определить через FX:

```
FUNCTION FX (T)
FX=4. *T (12.30)
```

```
RETURN
```

```
END
```

```
FUNCTION FY (T)
```

```
FY=FX (T) **2-1. (12.31)
```

```
RETURN
```

```
END
```

Оператор EXTERNAL может не использоваться в подпрограмме FY (12.31), вызывающей для исполнения подпрограмму FX (12.30). В ней происходит обычный вызов подпрограммы FX по имени из другого программного модуля в тексте подпрограммы, а не через аппарат формальных-фактических параметров. Хотя оператор

```
EXTERNAL FX
```

является в данном случае необязательным, его присутствие в подпрограмме (12.31) не приводит к ошибке. Вообще объявление имен вызываемых подпрограмм в программной единице, даже если оно и необязательно, является признаком хорошего стиля программирования и облегчает перенос программ из одной операционной среды в другую [28]-[30].

12.3. Оператор COMMON

Подпрограммы (FUNCTION и SUBROUTINE) представляют собой миниатюрные программы, называемые программными единицами или модулями. Метки и переменные в каждом модуле локализованы. Переменная с одним и тем же наименованием в разных модулях может обозначать совершенно разные величины. Это происходит потому, что память разделяется на локальные секции так, что каждая программная единица имеет доступ лишь к своей секции.

Обмен информацией между отдельными программными единицами осуществляется через механизм формальных и фактических параметров или через оператор общих областей COMMON, который относится к числу невыполняемых операторов.

— Оператор `COMMON` должен встретиться по крайней мере в двух программных единицах. Описанные в этих операторах `COMMON` переменные и массивы будут принадлежать одной общей области.

Элементы в области `COMMON` располагаются в порядке, определяемом порядком их написания в операторе `COMMON`, т.е. для соответствия элементов в двух операторах `COMMON` в разных программных единицах используется тот же позиционный принцип, что и для соответствия формальных и фактических параметров.

Оператор `COMMON` имеет вид:

$$\text{COMMON } B_1, B_2, \dots, B_N \quad (12.32)$$

где B_1, B_2, \dots, B_N — список переменных или массивов через запятую, т.е. идентификаторы переменных, массивов или массивов с указателем размерности.

Смысл оператора `COMMON` состоит в следующем. Пусть в основной программе присутствует оператор

$$\text{COMMON } A, B, C$$

Под переменные и массивы, указанные в нем, отведены некоторые ячейки памяти. Пусть в подпрограмме опять встретился оператор

$$\text{COMMON } U, V, W$$

Тогда переменная (или массив) U расположится в тех же ячейках, что и A ; переменная (или массив) V расположится в тех же ячейках, что и B и т.д. Если оператор `COMMON` встретится в другой подпрограмме еще раз, то новые переменные (или массивы) опять будут располагаться в соответствии с описанным правилом.

Оператор `COMMON` служит для того, чтобы записывать данные из разных подпрограмм в одни и те же ячейки памяти, т.е. общие области. Таким образом можно передавать данные из одной подпрограммы в другую и устанавливать соответствие между ними.

В качестве элементов общей области могут использоваться: имя переменной, имя массива, имя массива с указанием его размерности. Если размерность массива не указана в `COMMON`, то она обязательно должна быть указана в операторе `DIMENSION`, который должен предшествовать оператору `COMMON` (так как массив описывается всегда при первом его упоминании). Например:

$$\begin{aligned} &\text{DIMENSION } BW(4) \\ &\text{COMMON } BW \end{aligned} \quad (12.33)$$

Если в `COMMON` указать размерность массива, то запрещается ее указывать в `DIMENSION` (поэтому оператор `DIMENSION` для описания этого массива просто не используется):

$$\text{COMMON } BW(4) \quad (12.34)$$

Количество, тип и длина элементов общей области в одной программной единице должны совпадать с количеством, типом и длиной соответствующих элементов общей области в другой программной единице. Здесь также соответствие проводится по порядку следования элементов в списке. Их обозначение тоже не играет никакой роли и только для удобства чтения программы мы будем обычно использовать одинаковые обозначения для совпадающих элементов общей области.

В список элементов общей области не разрешается включать идентификаторы формальных параметров.

В качестве примера опишем подпрограмму SUBROUTINE K1UD (12.19), задающую уравнения (10.14) с помощью переменных, значения которым присваиваются в основной программе:

```
C      ОСНОВНАЯ ПРОГРАММА
      COMMON A1, B, C                                (12.35)
      A1=4.
      B=16.
      C=1.
```

.....

STOP

END

SUBROUTINE K1UD (T, X, Y)

```
      COMMON A, B, C                                (12.36)
```

X=A*T

Y=B*T**2-C

RETURN

END

По первому оператору COMMON (12.35) в общей области резервируются три ячейки для размещения переменных A1, B, C из основной программы: первая отводится для A1, вторая — для B, третья — для C; а по второму оператору COMMON (12.36) — те же три ячейки в той же позиционной последовательности элементов списка отводятся для размещения переменных A, B, C из подпрограммы SUBROUTINE. Это равносильно тождественности A1 и A, B и B, C и C в разных программных единицах, т.к. одинаковость обозначений (B и C) в разных модулях без оператора COMMON совсем ничего не означает, а разность идентификаторов (A1 и A) при использовании общей области не играет никакой роли.

Если же в рассмотренном примере использовать подпрограммы-функции (12.11)-(12.12), задающие уравнения движения (10.14) с помощью переменных, значения которым присваиваются также в основной программе, например, по фрагменту (12.35), то они примут вид:

```
      FUNCTION FX (T)
```

```
      COMMON A, B, C                                (12.37)
```

FX=A*T


```

RETURN
END
FUNCTION FY (T)
COMMON A, B, C                               (12.38)
FY=B**T2-C
RETURN
END

```

Сегмент основной фортран-программы (12.35) остается без изменения, оператор COMMON (12.36) встречается дважды в подпрограммах (12.37) и (12.38), поэтому описанная выше тождественность устанавливается теперь уже не для двух, а для трех программных модулей.

Однако в операторе COMMON из (12.37) два элемента (B и C), а в операторе COMMON из (12.38) один элемент (A), являются неиспользуемыми (фиктивными) и в своих программных модулях не применяются. Они служат для выравнивания объема памяти области COMMON (12.37) и для установления необходимых соответствий.

В операторе COMMON из (12.37) неиспользуемые (фиктивные) параметры B и C служат только для выравнивания длин общих блоков в разных программных единицах, так как соответствие производится с начала следования элементов в списке.

В операторе COMMON из (12.38) первый элемент A в списке является фиктивным. Он может быть заменен на любой другой, неиспользуемый в подпрограмме идентификатор, например FORM:

```
COMMON FORM, B, C
```

Обойтись без него в операторе COMMON из (12.38) нельзя, т.к. он нужен для установления соответствий с основной программой. При его отсутствии, т.е. замене оператора (12.38) на просто

```
COMMON B, C
```

будет установлено совсем другое соответствие: A1 и B (в основной программе) будут эквивалентны соответственно B и C (в подпрограмме). Это приведет к заданию совсем другого уравнения движения FY. Таким образом, если оператор

```
COMMON A, B, C
```

появляется в разных программных единицах (12.37) и (12.38), то использование A, B или C в любой из них соответствует обращению к первой, второй или третьей ячейке в области COMMON.

Возможно еще несоответствие из-за несовпадения типа соответствующих элементов в списках. При его возникновении одно нарушение может привести к несоответствию границ памяти всех следующих элементов, так как соответствие производится по позиционному принципу с начала списка и величины разного типа занимают в памяти ЭВМ разный объем. Ответственность за

соблюдение границ при расположении переменных в области COMMON ложится на программиста, ибо транслятором она не проверяется.

Самый верный и простейший способ обеспечения соответствия границ памяти состоит в том, чтобы указывать первыми все более длинные однотипные переменные: сначала все удвоенной точности (если они есть в программе), затем все вещественные и после них все целые. При этом будет гарантировано расположение переменных в нужных границах и можно не вдаваться в многочисленные тонкости этого вопроса. Например, если:

```
DOUBLE PRECISION D, H
REAL HN, HK
INTEGER K, N
```

то

```
COMMON D, H, HN, HK, K, N
```

поставит более длинные переменные первыми, чем будет гарантировано соблюдение соответствующих границ памяти.

Отметим также, что при указании имени массива в операторе COMMON (см. (12.33)), общая область включает в себя весь массив целиком. Массив не может быть частично в операторе COMMON, частично вне его. Это также следует иметь в виду при расчете количества элементов в списке для установления соответствия по объему памяти операторов COMMON.

Следующий материал до конца главы при первоначальном изучении рекомендуется опустить.

12.3.1. Помеченный блок COMMON

Область COMMON может быть разделена на несколько блоков. Один из них может быть непомеченным, а всем остальным даются имена, удовлетворяющие обычному правилу (1–6 алфавитно-цифровых символов, первым из которых должна быть буква). Имя блока заключается в прямые наклонные скобки в операторе COMMON и предшествует списку переменных в этом блоке. Пробелы, заключенные между прямыми наклонными скобками, указывают на возобновление непомеченного блока COMMON. Поэтому оператор:

```
COMMON A1/BB/B,C/ZID/TN,TK,DT//T,HT/NGZW/NG,NZ,NW (12.39)
```

обеспечивает размещение переменных A1, T и HT в непомеченном блоке; переменных B,C — в помеченном блоке с именем BB; переменных TN, TK, DT — в блоке ZID; целых переменных NG, NZ, NW в помеченном блоке с именем NGZW.

В общем виде помеченный блок COMMON можно представить следующим образом:

COMMON /имя1/список1.../имяN/списокN...

где имя1, ..., имяN — имена помеченных общих блоков памяти,

список1, ..., списокN... — элементы соответствующей именованной общей области (/имя1/, ..., /имяN/), перечисленные через запятую.

Идентификаторы элементов списка именованной общей области могут быть разными в разных программных единицах, так как для соответствия между ними используется позиционный принцип.

Имена помеченных общих блоков памяти в разных программных единицах должны быть те же самые. Количество именованных общих областей в программе не ограничивается.

Все неименованные общие области считаются одноименными и представляют собой одну неименованную общую область, которая в программе может быть только одна (см. оператор (12.39)).

Все одноименные общие области, относящиеся к разным программным модулям, будут совмещены друг с другом, т.е. будут размещены с одного и того же места памяти.

Каждый именованный общий блок /имяN/ рассматривается как отдельный описанный выше в п. 12.3 оператор COMMON. Поэтому оператор (12.39) эквивалентен нижеследующему фрагменту:

```
COMMON A1, T, HT
COMMON /BB/B, C
COMMON /ZID/TN, TK, DT
COMMON /NGZW/NG, NZ, NW
```

(12.40)

В список элементов именованной и неименованной общей области не разрешается включать идентификаторы формальных параметров.

Помеченные блоки удобно использовать в тех случаях, когда общая область велика или некоторые подпрограммы обращаются только к небольшой части общих данных. Сегмент основной фортран-программы (12.35) и подпрограммы (12.37)-(12.38) можно переписать с использованием именованных общих блоков:

```
COMMON /AA/A1/BB/B, C
COMMON /AA/A
COMMON /BB/B, C
```

(12.41)
(12.42)
(12.43)

Здесь одна неименованная общая область примеров (12.35), (12.37)-(12.38) разбивается на две именованные с именами AA и BB. Оператор (12.41) помещается на место оператора (12.35) в основную программу, а операторы (12.42) и (12.43) — на место операторов (12.37) и (12.38) в подпрограммы.

12.3.2. Подпрограмма начальных значений BLOCK DATA

Величины, входящие в неименованные общие области, не могут иметь начальных значений, заданных с помощью операторов явного описания типа или DATA. Переменным, находящимся в помеченных блоках COMMON, можно присвоить начальные значения и только с помощью подпрограммы BLOCK DATA, которая имеет вид

```
BLOCK DATA
<невыполняемые операторы>
.....
END
```

После заголовка подпрограммы начальных значений BLOCK DATA и перед оператором END могут находиться только невыполняемые операторы (COMMON, DIMENSION, IMPLICIT, DATA, операторы явного описания типа INTEGER, REAL, DOUBLE PRECISION).

Подпрограмма BLOCK DATA будет содержать хотя бы один оператор COMMON. Он должен специфицировать каждую переменную в именованном общем блоке независимо от того, присваивается ей какое-либо значение или нет.

В одной подпрограмме можно присваивать начальные значения нескольким помеченным блокам. Так, значения переменным A1, B и C, задаваемые во фрагменте основной фортран-программы (12.35) операторами присваивания, можно с использованием помеченных блоков (12.41) — (12.43) задать с помощью подпрограммы начальных значений:

```
BLOCK DATA
COMMON /AA/A1/BB/B,C (12.44)
REAL A1/4./,B/16./,C/1./ (12.45)
END
```

Подпрограмма начальных значений начинается со своего названия BLOCK DATA, после которого оператор COMMON (12.44) специфицирует каждую переменную, входящую в именованные общие блоки AA и BB. Затем оператор явного описания типа REAL (12.45) описывает A1, B и C как вещественные переменные, присваивая им начальные значения (соответственно 4., 16. и 1.). Оператор END указывает на конец подпрограммы.

Подпрограмма BLOCK DATA не компилируется отдельно. Она включается обычно в основную программу и располагается или впереди ее первого оператора, или сразу после оператора END.

Заданное начальное значение переменной сохраняется до тех пор, пока эта переменная не будет пересчитана по программе (не появится в списке вводимых величин, в левой части оператора присваивания или в качестве аргумента в операторе CALL).

ГЛАВА 13. СОСТАВЛЕНИЕ ПРОГРАММ НА АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ ФОРТРАН ДЛЯ РЕШЕНИЯ ЗАДАЧ ПО КИНЕМАТИКЕ С ИСПОЛЬЗОВАНИЕМ ЧИСЛЕННОГО ДИФФЕРЕНЦИРОВАНИЯ

В качестве примеров для составления программ на алгоритмическом языке Фортран с использованием численного дифференцирования выберем рассматривавшиеся ранее уравнения движения (10.14) и (10.16). Для первых из них будем определять радиус кривизны траектории (см. задание К-1 [21, с. 60-62]), а для вторых — только скорость и ускорение точки (см. задание К-7 [21, с. 99-106]). Описываемые здесь программы можно также использовать для решения любых задач по кинематике с применением численного дифференцирования, в частности индивидуальных заданий К-1, К-2, К-9 и К-10 из сборника [22].

До начала изучения настоящего материала желательно, чтобы студент уже освоил работу с использованием готовых программ KINMP и DKINMP (см. п. 10), убедился в правильности уравнений движения для своей задачи и теперь все внимание мог обратить на различные вопросы составления программ.

При написании программ с использованием операторов-функций, подпрограмм-функций и подпрограмм SUBROUTINE (пп. 13.1-13.3) для численного дифференцирования используются формулы для представления первой и второй производных функций в конечных разностях (10.1) и (10.2).

При использовании стандартных подпрограмм (см. п. 14) применяются другие методы. Для численного дифференцирования аналитически заданных функций подпрограммы DCAR и DBAR используют метод экстраполяции Ричардсона и Ромберга. Для функций, заданных таблицей значений в равноотстоящих точках, подпрограммы DET3 и DET5 вычисляют производную от интерполяционного многочлена Лагранжа, построенного по трем (DET3) или пяти (DET5) последовательным точкам.

13.1. Использование операторов-функций

Простейшая программа для определения радиуса кривизны траектории уравнений движения (10.14) с использованием операторов-функций состоит из следующих основных блоков: 1) описание используемых операторов-функций; 2) задание или ввод исходных данных и присваивание начальных значений переменным; 3) печать заголовка таблицы результатов и контрольная печать; 4) обращение к операторам-функциям и определение проекций скорости точки; 5) обращение к операторам-функциям и определение проекций ускорения точки; 6) определение тангенциального ускорения точки;

7) определение нормального ускорения точки и радиуса кривизны траектории; 8) печать выходной информации; 9) изменение на шаг времени T и его проверка на окончание цикла, т.е. соответствие с конечным временем TK ; 10) операторы `STOP` и `END`.

Она может быть реализована, например, следующим образом:

```

C      П Р О Г Р А М М А 13.1
      X (T)=4.*T                      10
      Y (T)=16.*T*T-1.                11
      DATA TN,TK,HT,DT/0.,1.,0.05,0.005/ 20
      T=TN                             21
      PRINT 15                          30
15  FORMAT (6X,'T',12X,'V',12X,'A',10X,'ATANG',10X,'RO') 31
20  VX=(X(T+DT)-X(T))/DT              40
      VY=(Y(T+DT)-Y(T))/DT            41
      V=SQRT(VX*VX+VY*VY)             43
      AX=(X(T+DT)-2.*X(T)+X(T-DT))/(DT*DT) 50
      AY=(Y(T+DT)-2.*Y(T)+Y(T-DT))/(DT*DT) 51
      A=SQRT(AX*AX+AY*AY)             53
      ATANG=(VX*AX+VY*AY)/V           60
      ANORM=SQRT(A*A-ATANG*ATANG)     70
      RO=V*V/ANORM                    76
      PRINT 40,T,V,A,ATANG,RO         80
40  FORMAT (5(1X,G12.5))              81
      T=T+HT                           90
      IF (T.LE.TK) GOTO 20            91
      STOP                             100
      END                              101

```

Для удобства первоначальной ориентировки 1-я цифра номера обычно указывает на соответствующий блок: например, операторы 50, 51 и 53 относятся к блоку 5).

Операторы 10 и 11 описывают операторы-функции $X(T)$ и $Y(T)$, задающие определение уравнений движения (10.14) с использованием формального параметра T (см. п. 12.1.1).

Оператор 20 `DATA` (см. п. 7.7.1) используется для задания начальных значений следующим переменным: начальному TN и конечному TK значению временного интервала, на котором проводится дифференцирование функций и другие расчеты; HT — временной интервал или шаг по времени, с которым проводятся расчеты; DT — шаг дифференцирования, т.е. некоторое приращение времени DT для вычисления производных в формулах (10.1) и (10.2). Указанные переменные получают в момент загрузки программы значения $TN=0.$, $TK=1.$, $HT=0.05$, $DT=0.005$ сек.

Арифметический оператор присваивания 21 переменной T присваивает значение времени TN, с которого начинают выполняться расчеты.

Оператор 30 под руководством оператора 31 с меткой 15 осуществляет печать заголовка таблицы результатов расчета. Отступив от левого края 6 пробелов, первый из которых выполняет роль управляющего символа для устройства печати, будет напечатано литеральное данное T. Затем, каждый раз через 12 пробелов, будут выведены на печать литеральные данные V и A, а через 10 пробелов еще ATANG и RO. Они идентифицируют выводимые в дальнейшем оператором 80 под ними результаты вычислений (см. п. 7.6.8).

Оператор 40 с меткой 20 производит вычисление проекции скорости VX по формуле (10.1). Из значения описанной оператором 10 функции X при времени T+DT вычитается значение этой функции для текущего времени T, а разность делится на величину шага дифференцирования DT, причем результат помещается в ячейку для переменной VX.

Оператор 41 выполняет полностью аналогичные оператору 40 действия, но только с использованием оператора-функции 11, определяя проекцию скорости VY.

Оператор 43 определяет значение величины скорости V, извлекая квадратный корень из суммы квадратов ее проекций VX и VY.

Оператор 50 производит вычисление проекции ускорения AX по формуле (10.2). Из значения описанной оператором 10 функции X при времени T+DT вычитается удвоенное значение этой функции для T и прибавляется ее значение при T-DT. Результат делится на величину квадрата шага дифференцирования DT*2, после чего полученное значение помещается в ячейку для переменной AX.

Оператор 51 выполняет полностью аналогичные оператору 50 действия, но только с использованием оператора-функции 11, определяя проекцию ускорения AY.

Оператор 53 определяет значение величины ускорения A, извлекая квадратный корень из суммы квадратов его проекций AX и AY.

Оператор 60 вычисляет тангенциальное ускорение точки по общеизвестной формуле для его численного определения с использованием значений проекций скорости и ускорения.

Оператор 70 по найденным полному и тангенциальному ускорениям определяет нормальное ускорение точки, извлекая квадратный корень из их разности квадратов.

Оператор 76 вычисляет значение радиуса кривизны траектории RO из формулы для нормального ускорения, деля квадрат скорости точки V*V на ее нормальное ускорение ANORM.

Оператор 80 под руководством оператора 81 с меткой 40 выводит на печать результаты расчета: значение текущего времени T , при котором проводились расчеты, величины скорости материальной точки V и ускорения A , а также значения тангенциального ускорения $ATANG$ и радиуса кривизны траектории RO .

Арифметический оператор присваивания 90 заменяет значение текущего времени T на $T+NT$. Берется содержимое ячейки памяти T (после первого раза выполнения программы $T=TN=0$.) К нему прибавляется значение приращения времени $NT=0.05$. Результат отсылается обратно в ячейку переменной T , заменяя бывшую там информацию на новую (теперь значение текущего времени $T=0.05$).

Условный логический оператор IF 91 (см п. 7.5.3) анализирует на истинность логическое выражение, заключенное в скобки.

Если оно истинно, т.е. текущее значение T меньше его конечного значения $TK=1$., то выполняется оператор безусловного перехода $GOTO$ 20 (см п. 7.5.1). После него станет выполняться оператор с меткой 20 и работа программы будет повторяться с определения проекций скорости VX , VY и т.д. при следующем значении T .

Далее оператор 80 выведет на печать результаты расчета при данном T (на втором шаге $T=0.05$ сек). Затем оператор 90 снова заменит значение времени T на $T+NT$ (и станет $T=0.1$ сек), а условный логический оператор 91 снова передаст управление оператору с меткой 20 для очередного повторения программы при следующем времени T (т.к. теперь $T=0.1$ и также меньше $TK=1$.).

Так будет повторяться до тех пор, пока после очередного выполнения программы по арифметическому оператору присваивания 90 значение T не станет равным TK , после чего оператор 91 опять передаст управление оператору 20, т.к. выражение отношения в $(T.LE.TK)$ все еще останется истинным при $T=TK$.

Программа будет повторена в последний раз и после очередного выполнения оператора 90 значение T станет больше TK . Тогда выражение в скобках $(T.LE.TK)$ в условном логическом операторе 91 станет ложным, поэтому оператор $GOTO$ 20 пропускается и выполняется следующий в программе за логическим IF оператор 100, указывающий на конец вычислений.

Оператор 101 указывает на конец основной программы.

В программе 13.1 и далее, выполняя рекомендацию п. 7.3.1, для записи квадратов переменных обычно используется их непосредственное умножение друг на друга. Оно производится быстрее, чем расчет экспоненты для вычисления квадратов.

Операторы-функции 10-11 в программе 13.1 могут быть описаны с использованием любой из форм (12.3)-(12.4), (12.7), (12.9), что предлагаем вам проверить самостоятельно.

Для использования примера (12.7) в программу 13.1 нужно добавить задание значений переменным A1, B, C. Это можно сделать, например, с помощью оператора DATA:

```
X (T) = A1 * T                                10
Y (T) = B * T * T - C                          11
DATA A1, B, C / 4., 16., 1. /                  35
```

Конечно, от таких замен и дополнений результаты вычислений несколько не изменятся.

Дополнение 13.1. Защита от возможного переполнения порядка.

При расчете по программе 13.1 для какого-либо задания и значения времени T может случиться, что значения скорости V окажется равным или весьма близким к нулю. Так как в операторе 60 происходит деление на эту величину, то это может вызвать переполнение порядка. В памяти ЭВМ максимальная величина числа ограничивается его допустимым порядком (для вещественных чисел около 10^{*75}). Поэтому нужно предусмотреть защиту от возможной ошибки переполнения порядка. Для этого в программу 13.1 можно вставить перед оператором 60, например, следующий условный логический оператор IF:

```
IF (V.LT.1.E-33) V=1.E-33                      59
```

Теперь, если значение скорости V по какой-либо причине (в результате вычислений или вследствие допущенных ошибок в составлении уравнений движения) окажется меньше 1.E-33, то переменной V будет присвоено значение 1.E-33. Оно достаточно мало, чтобы не исказить результаты дальнейшего решения, а обратная к ней величина гораздо меньше уровня переполнения и не вызовет соответствующей ошибки в операторе 60.

Программу 13.1 можно еще дополнить сообщением о случае возможности переполнения порядка, для чего перед оператором 59 можно вставить, например, такую группу операторов:

```
IF (V.LT.1.E-33) PRINT 25, T, V                57
25 FORMAT (5X, 'В Н И М А Н И Е ! ПРИ ВРЕМЕНИ T=', F6.3, 58
* 'СЕК ЗНАЧЕНИЕ СКОРОСТИ V' / 2X, 'МЕНЬШЕ 1.E-33 СМ/СЕК ',
* ' И СОСТАВЛЯЕТ V=', G12.5, ' СМ/СЕК.')
```

Теперь, если $V < 1.E-33$, то из-за истинности логического выражения в скобках (V.LT.1.E-33) в операторе 57 будет выполнен оператор PRINT, выводящий на печать под руководством оператора FORMAT с меткой 25 аварийное сообщение 58.

При $V > 1.E-33$ в обоих условных логических операторах 57 и 59 выражение в скобках окажется ложным. В результате находящиеся за ними в тех же строках операторы будут пропускаться и начнут выполняться следующие за

ними операторы программы 59 и 60 соответственно, т.к. следующий за оператором 57 оператор FORMAT с меткой 25 является невыполняемым и может быть расположен в любом месте программы. В этом случае операторы 57-59 не оказывают воздействия на ход выполнения программы.

Дополнение 13.2. При составлении программы всегда следует анализировать возможные аварийные ситуации при численных расчетах и предусматривать соответствующие защиты от них.

При определении радиуса кривизны траектории по программе 13.1 в случаях прямолинейной траектории движения полное ускорение A равняется его тангенциальному ускорению $ATANG$, а нормальное ускорение $ANORM=0$. Это вызывает необходимость производить защиту от деления на 0 в операторе 76, т.е. от возможного переполнения порядка по дополнению 13.1.

Однако при численных расчетах таких точных равенств не достигается вследствие различных ошибок вычисления, ограничения (усечения) и округления (см. п. 2 в [10]). Они возникают вследствие того, что арифметические действия проводятся с конечным числом значащих цифр, с помощью которых часто невозможно представить необходимую величину конечной дробью в принятой системе счисления (например, $1/3$).

Возможны различные частные случаи и нужно не только проводить защиту от деления на 0 перед выполнением оператора 76, но может возникнуть даже и подкоренное отрицательное выражение в операторе 70, когда из-за различных вычислительных ошибок вместо $ATANG=A$ получится $ATANG>A$.

Нарушение этого равенства при прямолинейной траектории точки может произойти потому, что при определении двух почти равных чисел по разным формулам (A по операторам 50, 51 и 53, и $ATANG$ по операторам 40, 41, 43, 50, 51 и 60) происходит разное распространение ошибок.

Это приводит к тому, что при вычитании двух почти равных чисел относительная ошибка разности может оказаться очень большой. Из-за того, что знак ошибки обычно неизвестен, может получиться и нежелательное нарушение теоретического равенства $ATANG=A$ в сторону неравенства $ATANG>A$ в результате реальных численных расчетов. Это противоречит физическому смыслу (часть больше целого), но может реально возникнуть в процессе вычислений. Поэтому следует также предусмотреть защиту и от возможной аварийной ситуации при вычислении подкоренного отрицательного выражения в операторе 70, для чего вместо него вставим в программу 13.1 группу операторов:

```

DAN=A*A-ATANG*ATANG                                70
IF (DAN.LT.0.) PRINT 27,T,DAN                        71
27 FORMAT(5X,'В Н И М А Н И Е ! ! ! ПРИ ОПРЕДЕЛЕНИИ ',  72
1 'НОРМАЛЬНОГО УСКОРЕНИЯ ПО ЗАДАНЫМ УРАВНЕНИЯМ ДВИЖЕНИЯ' /

```

```

2 2X, 'ПРИ ВРЕМЕНИ T=' , F6.3, ' СЕК ПОД КОРНЕМ ОТРИЦАТЕЛЬНОЕ ',
3 'ВЫРАЖЕНИЕ, РАВНОЕ ', G12.5, ', ' / 2X, 'КОТОРОЕ ДЛЯ ВОЗМОЖНОСТИ',
4 ' ДАЛЬНЕЙШЕГО РАСЧЕТА УСЛОВНО ПРИНИМАЕТСЯ РАВНЫМ НУЛЮ.')
   IF (DAN.LT.0.) DAN=0.                                73
   ANORM=SQRT (DAN)                                     74

```

Теперь оператор 70 только вычисляет подкоренное выражение для определения нормального ускорения ANORM и присваивает его переменной DAN.

Затем условный логический оператор 71 проверяет: DAN<0 или нет. Если DAN<0., то из-за истинности логического выражения в скобках (DAN.LT.0.) будет выполнен оператор PRINT. Он выведет на печать под управлением оператора FORMAT с меткой 27 аварийное сообщение 72.

Такой же условный логический оператор 73 при DAN<0 приведет к выполнению находящийся с ним в одной строке арифметический оператор присваивания DAN=0. Выбор нулевого значения DAN для этого случая вытекает из физического анализа возникающей ситуации. При аналитических расчетах для прямолинейной траектории должно быть ATANG=A, ANORM=0 и DAN=0.

Теперь оператор 74 для рассматриваемого случая извлечет квадратный корень из нулевого значения DAN, получив ANORM=0., как это и должно быть при точных аналитических расчетах.

Если после выполнения оператора 70 дополнения 13.2 окажется DAN>0., то из-за ложности выражения в скобках (DAN.LT.0.) в условном логическом операторе 71 будет пропущен оператор PRINT 27 и станет выполняться такой же условный логический оператор 73 (т.к. стоящий в следующей строке оператор FORMAT с меткой 27 является невыполняемым и может быть расположен в любом месте программы). В операторе 73, также из-за ложности в этом случае выражения отношения (DAN.LT.0.), будет пропущен оператор DAN=0. Затем станет выполняться стоящий за ним в следующей строке оператор 74, который просто извлечет квадратный корень из вычисленного оператором 70 значения DAN.

В результате при DAN>0 операторы 70 и 74 дополнения 13.2 произведут необходимые действия для вычисления нормального ускорения ANORM, а операторы 71-73 не окажут никакого воздействия на ход выполнения программы.

Защиту от переполнения порядка при использовании операторов 70-74 уже нужно делать обязательно, так как при DAN<0 оператор 76 будет делить на ANORM=0. Ее можно выполнить аналогично оператору 59 дополнения 13.1, например:

```

IF (ANORM.LT.1.E-33) ANORM=1.E-33                      75

```

Следует отметить, что для большинства задач дополнения 13.1 и 13.2 не произведут никаких заметных изменений в работе ваших программ. Они

только повысят надежность их работы, защитив от аварийных экстремальных ситуаций.

В случае прямолинейной траектории погрешности вычислений обычно делают относительно существенной разницу между A и $ATANG$ при численном их определении. Это приводит (при $DAN > 0$.) просто к большим значениям радиуса кривизны RO (порядка тысяч или десятков тысяч). При использовании удвоенной точности (см. п. 15) порядок RO возрастает (в сотни и тысячи раз), оставаясь все же просто большим числом. Для более резкого выделения близости к бесконечности радиуса кривизны RO нужны специальные программные дополнения, аналогичные описанным выше. Их предлагаем желающим сделать самостоятельно.

Дополнение 13.3. Для решения любой плоской задачи кинематики точки по определению радиуса кривизны траектории RO в программе 13.1 достаточно только соответствующим образом описать операторы-функции 10-11. При необходимости также нужно изменить задание начальных значений в операторе 20.

При решении задач на сложное движение точки ее тангенциальное и нормальное ускорения, а также радиус кривизны траектории определять не требуется. Поэтому в этом случае не следует загружать ЭВМ работой, результаты которой мы не можем сравнить хотя бы в одной точке с аналитическим расчетом. Нужно убрать из программы 13.1 операторы 60, 70, 76, определяющие эти величины, соответственно сократив список переменных, выводимых на печать по оператору 80. Для пространственных задач нужно еще внести изменения, связанные с наличием координаты Z . Для уравнений движения (10.16) это может иметь следующий вид:

```

SR (T) = 16. - 8. * COS (3. * PI * T)                10N
VIE (T) = .9 * T * T - 9. * T ** 3                 11N
X (T) = -SR (T) / 2. * SIN (VIE (T) )              10
Y (T) = SR (T) / 2. * COS (VIE (T) )               11
Z (T) = SR (T) * COS (PI / 6. )                    12
PI = 3.141592653589                                 15
DATA TN, TK, HT, DT / 0., 0.44444, 0.02222, 0.00222 / 20
15 FORMAT (6X, ' T ', 12X, ' V ', 12X, ' A ' )       31
VZ = (Z (T+DT) - Z (T) ) / DT                       42
V = SQRT (VX * VX + VY * VY + VZ * VZ)              43
AZ = (Z (T+DT) - 2. * Z (T) + Z (T-DT) ) / (DT * DT) 52
A = SQRT (AX * AX + AY * AY + AZ * AZ)              53
PRINT 40, T, V, A                                    80

```

Операторы 10N-11N задают уравнения относительного SR и переносного VIE движения материальной точки примера (10.16) в виде операторов-

функций $SR(T)$ и $VIE(T)$. Они используются при описании операторов-функций 10-12, задающих уравнения движения материальной точки относительно координатных осей X , Y и Z .

Значение переменной PI задает арифметический оператор присваивания 15.

В результате соответствия, определенного оператором 20, указанные в его списке переменные получают в момент загрузки программы на выполнение следующие значения: $TN=0.$, $TK=0.44444$, $HT=0.02222$, $DT=0.00222$ сек.

Оператор 31, руководящий выводом на печать заголовка таблицы результатов оператором 30, определит укороченный список литеральных данных, которые будут напечатаны: T , V , A (соответственно через 6, 12 и 12 пробелов каждый).

Операторы 42 и 52 с использованием оператора-функции 12 определяют проекции скорости VZ и ускорения AZ соответственно.

Видоизмененные операторы 43 и 53 определяют значения абсолютных скорости V и ускорения A , извлекая квадратный корень из суммы квадратов всех трех их проекций.

Оператор 80 выводит на печать результаты расчета текущего времени T , абсолютных скорости V и ускорения A под руководством оператора 81. Его повторитель группы спецификаций (равный пяти) теперь не соответствует списку выводимых величин (состоящему из трех переменных), что допускается. Поэтому желающие привести их в соответствие с помощью оператора:

```
40 FORMAT (3 (1X, G12.5) ) 81
```

не обнаружат никаких изменений в печати результатов работы своей программы. Однако, мы все же рекомендуем приводить в соответствие операторы `PRINT` и `FORMAT` до полного и ясного понимания механизма их работы в каждом конкретном случае. Ибо в данном случае в операторе 81 повторитель группы спецификаций может быть равен и единице:

```
40 FORMAT (1X, G12.5)
```

Но теперь на печать значения T , V и A будут выводиться не в одну строку, а в виде столбца из трех строк, в каждой из которых будет только одно напечатанное число. Это сделает бессмысленными операторы 30 и 31, печатающие заголовок таблицы результатов расчета, а также представит вывод полученной информации в весьма неудобной для обработки форме.

Дополнение 13.4. В программе 13.1 обращение к операторам-функциям для моментов времени T и $T+DT$ происходит дважды для каждой координаты. Поэтому ее можно оптимизировать. Для этого обращаются

к оператору-функции один раз и присваивают вычисленное значение некоторой переменной, которую потом используют для дальнейших вычислений. Это оказывается быстрее уже при двукратном обращении к функции. Для программы 13.1 с учетом дополнения 13.3 для уравнений (10.16) это может принять следующий вид:

20	$X_T = X(T)$	40N
	$Y_T = Y(T)$	41N
	$Z_T = Z(T)$	42N
	$X_{TDT} = X(T+DT)$	43N
	$Y_{TDT} = Y(T+DT)$	44N
	$Z_{TDT} = Z(T+DT)$	45N
	$VX = (X_{TDT} - X_T) / DT$	40
	$VY = (Y_{TDT} - Y_T) / DT$	41
	$VZ = (Z_{TDT} - Z_T) / DT$	42
	$AX = (X_{TDT} - 2 * X_T + X(T-DT)) / (DT * DT)$	50
	$AY = (Y_{TDT} - 2 * Y_T + Y(T-DT)) / (DT * DT)$	51
	$AZ = (Z_{TDT} - 2 * Z_T + Z(T-DT)) / (DT * DT)$	52

Опишем только измененные и добавленные операторы с номерами 40N-45N, 40-42 и 50-52. Остальные описаны в пояснениях к программе 13.1 и в дополнении 13.3.

Оператор 40N с меткой 20 производит обращение к оператору-функции $X(T)$. Фактическим параметром является значение текущего времени T . Вычисленное значение функции присваивается переменной X_T . Метка 20 определяет собой начало очередного повторения вычислений по программе. Поэтому она переставлена сюда с оператора 40 программы 13.1, с которого ранее начинались обращения к операторам-функциям.

Операторы 41N-42N выполняют аналогичные обращения к операторам-функциям $Y(T)$ и $Z(T)$, присваивая вычисленные значения переменным Y_T и Z_T .

Операторы 43N-45N (соответственно каждый) выполняют обращения к тем же операторам-функциям, что и операторы 40N-42N, но только при значении фактического параметра $T+DT$, присваивая вычисленные значения переменным X_{TDT} , Y_{TDT} и Z_{TDT} .

Обращение к операторам-функциям при $T-DT$ происходит только один раз для каждой функции, поэтому нецелесообразно использовать дополнительные переменные.

Операторы 40-42 и 50-52, хотя форма их записи изменилась, выполняют ту же роль и с тем же результатом, что и раньше.

13.2. Использование подпрограмм-функций

Применение подпрограмм-функций и операторов-функций полностью аналогично. Поэтому программа с их использованием для определения радиуса кривизны траектории не только состоит из тех же основных блоков 2)-9), но может состоять даже из тех же основных операторов, что и программа 13.1.

Для этого нужно выбрать в качестве имен подпрограмм-функций те же идентификаторы, что были и у соответствующих операторов-функций. Вместо описания операторов-функций в основной программе, нужно описать соответствующие подпрограммы-функции в одноименных программных модулях.

Таким образом, программу 13.1 (вместе с дополнениями 13.1-13.2) можно непосредственно использовать для определения радиуса кривизны траектории с помощью подпрограмм-функций. Для этого нужно удалить из программы все определения операторов-функций 10 и 11, а также дополнить ее описанием подпрограмм-функций в любой из форм (12.11)-(12.12), (12.15)-(12.16) или (12.30)-(12.31). В этих фрагментах нужно только заменить имена функций FX и FY на X и Y соответственно (в заголовке функции и в операторе присваивания).

Теперь, операторы 40-41 и 50-51, оставшись по внешнему виду в такой модифицированной программе 13.1 неизменными, используют для определения проекций скорости (40-41) и ускорения точки (50-51) обращение уже к подпрограммам-функциям.

Дополнение 13.5. Используемые идентификаторы для формальных параметров при описании подпрограмм-функций могут быть любыми (сравните описание функций (12.11)-(12.12), (12.15)-(12.16) и (12.30)-(12.31)). Имена функций должны быть одинаковыми при описании подпрограммы-функции и при обращении к ней.

Поэтому можно также оставить описание подпрограмм-функций (12.11)-(12.12), (12.15)-(12.16) или (12.30)-(12.31) неизменным, а согласовать с ним соответствующим образом имена при обращении к этим функциям в операторах 40-41 и 50-51 основной программы (заменив здесь уже X и Y на FX и FY):

20	$VX = (FX(T+DT) - FX(T)) / DT$	40
	$VY = (FY(T+DT) - FY(T)) / DT$	41
	$AX = (FX(T+DT) - 2 * FX(T) + FX(T-DT)) / (DT * DT)$	50
	$AY = (FY(T+DT) - 2 * FY(T) + FY(T-DT)) / (DT * DT)$	51

С таким видоизменением модификацию программы 13.1 можно также непосредственно использовать с подпрограммами-функциями (12.11)-(12.12), (12.15)-(12.16) или (12.30)-(12.31).

Дополнение 13.6. При описании подпрограмм-функций также, как и для операторов-функций (см. (12.7) и (12.8)), могут использоваться переменные, которых нет в списке формальных параметров. Например, подпрограммы-функции (12.11)-(12.12) можно записать, используя вместо чисел также переменные А, В, С. Каждая из этих переменных является общей только для того программного модуля, в котором она используется. Значения этих переменных должны быть определены до первого их использования, например, с помощью оператора DATA (см. п. 7.7.1):

```

FUNCTION FX(T)
DATA A/4./
FX=A*T
RETURN
END

```

(13.1)

```

FUNCTION FY(T)
DATA B,C/16.,1./
FY=B*T**2-C
RETURN
END

```

(13.2)

Значения переменным (А, В, С) в подпрограммы можно передать и с использованием оператора COMMON, задавая их в основной программе (см. сегмент (12.35)). Тогда подпрограммы-функции FX и FY должны иметь вид по фрагментам (12.37) и (12.38).

Для задания значений переменным во фрагменте основной программы (12.35) может также использоваться оператор ввода READ. Для этой цели не могут использоваться операторы задания начальных значений (DATA и операторы явного описания типа), так как величины, входящие в неименованные общие области не могут иметь начальных значений.

Дополнение 13.7. Если использовать именованные общие области (см. примеры (12.41)-(12.43)), то находящимся в них переменным можно присвоить начальные значения с помощью подпрограммы BLOCK DATA (см. примеры (12.44)-(12.45)) следующим образом:

```

BLOCK DATA
COMMON /AA/A1/BB/B,C
REAL A1/4./,B/16./,C/1./
END
FUNCTION FX(T)
COMMON /AA/A
FX=A*T
RETURN
END

```

103
105
107
109
110
115
120
130
131

FUNCTION FY (T)	140
COMMON /BB/B, C	145
FY=B*T**2-C	150
RETURN	160
END	161

Оператор 103 представляет собой заголовок подпрограммы начальных значений BLOCK DATA. Она располагается в данном случае после оператора END программы 13.1, модифицированной по дополнению 13.5 и использующей для своей работы подпрограммы-функции FX и FY.

Оператор 105 обеспечивает размещение переменной A1 в помеченном блоке с именем AA, а переменных B, C — с именем BB.

Оператор 107 явного описания типа REAL описывает переменные A1, B и C как вещественные, присваивая им одновременно начальные значения соответственно 4., 16. и 1.

Оператор 109 указывает на конец подпрограммы начальных значений.

Оператор 110 представляет собой заголовок подпрограммы-функции с именем FX, имеющей в качестве списка формальных параметров одну переменную T.

Оператор 115 устанавливает соответствие между переменными A1 и A, входящими в одноименный общий блок с именем AA. Так как в подпрограмме BLOCK DATA переменной A1 было присвоено начальное значение A1=4., то теперь и в подпрограмме-функции FX A=4.

Оператор 120 описывает подпрограмму-функцию FX и присваивает вычисленное значение одноименной переменной FX, являющейся выходом подпрограммы.

По оператору 130 осуществляется возврат в вызывающую программу, обратившуюся к подпрограмме FX для вычисления значения функции в процессе своей работы.

Оператор 131 указывает на конец подпрограммы-функции FX.

Оператор 140 определяет заголовок подпрограммы-функции FY.

Оператор 145 устанавливает соответствие между переменными, входящими в общий блок с именем BB. Так как переменным общей области с именем BB в подпрограмме BLOCK DATA были присвоены начальные значения, то и в подпрограмме FY B=16., C=1.

Оператор 150 описывает подпрограмму-функцию FY и присваивает вычисленное значение одноименной переменной FY.

По оператору 160 осуществляется возврат в вызывающую программу, а оператор 161 указывает на конец подпрограммы-функции FY.

Дополнение 13.8. Рассмотрим более подробно пространственную задачу на сложное движение точки (10.16) с использованием подпрограмм-функций. Задание ее уравнений движения возможно в разных формах:

- непосредственно в прямом виде:

```
FUNCTION FX (T)
FX=- (16.-8.*COS (3.*3.14159*T) ) /2.*
* SIN (0.9*T*T-9.*T**3)
RETURN
END
```

(13.3)

```
FUNCTION FY (T)
FY=(16.-8.*COS (3.*3.14159*T) ) /2.*
* COS (0.9*T*T-9.*T**3)
RETURN
END
```

(13.4)

```
FUNCTION FZ (T)
FZ=(16.-8.*COS (3.*3.14159*T) ) *COS (3.14159/6.)
RETURN
END
```

(13.5)

- или с использованием вспомогательных переменных:

```
FUNCTION FX (T)
PI=3.141592653589
SR=16.-8.*COS (3.*PI*T)
VIE=.9*T*T-9.*T**3
FX=-SR/2.*SIN (VIE)
RETURN
END
```

(13.6)

```
FUNCTION FY (T)
PI=3.141592653589
SR=16.-8.*COS (3.*PI*T)
VIE=.9*T*T-9.*T**3
FY=SR/2.*COS (VIE)
RETURN
END
```

(13.7)

```
FUNCTION FZ (T)
PI=3.141592653589
SR=16.-8.*COS (3.*PI*T)
FZ=SR*COS (PI/6.)
RETURN
END
```

(13.8)

FUNCTION FX (T)	110
COMMON PI, SR, VIE	115
PI=3.141592653589	117
SR=16.-8.*COS(3.*PI*T)	118
VIE=.9*T*T-9.*T**3	119
FX=-SR/2.*SIN(VIE)	120
RETURN	130
END	131
FUNCTION FY (T)	140
COMMON PI, SR, VIE	145
FY=SR/2.*COS(VIE)	150
RETURN	160
END	161
FUNCTION FZ (T)	170
COMMON PI, SR, VIE	175
FZ=SR*COS(PI/6.)	180
RETURN	190
END	191

В результате соответствия, устанавливаемого оператором 20 DATA, перечисленные в нем переменные приобретают начальные значения $TN=0.$, $TK=0.44444$, $HT=0.02222$, $DT=0.00222$ сек.

Арифметический оператор присваивания 21 текущему времени T присваивает значение начального времени TN, с которого начинают выполняться расчеты.

Оператор 30 под руководством оператора 31 с меткой 15 осуществляет печать заголовка таблицы результатов расчета: отступив от левого края 6 пробелов, первый из которых выполняет роль управляющего символа для устройства печати, сначала будет напечатано литеральное данное T, а затем, каждый раз через 12 пробелов, соответственно V и A.

Операторы 40N-45N аналогичны операторам с совпадающими номерами дополнения 13.4, только производят обращение к подпрограммам-функциям FX, FY и FZ, присваивая их вычисленные значения при фактических параметрах T и T+DT соответственно переменным XT, YT, ZT и XTDT, YTDT, ZTDT.

Операторы 40-43 и 53 полностью тождественны, а операторы 50-52 аналогичны операторам с совпадающими номерами дополнений 13.4 и 13.3. Они производят вычисления проекций скорости и ускорения, а также их абсолютных значений V и A. Небольшое отличие находится в операторах 50-52, в которых обращение производится к соответствующим подпрограммам-функциям.

Оператор 80 под руководством оператора 81 с меткой 40 выводит на печать результаты расчета: значения текущего времени T, абсолютной скорости V и ускорения A.

Операторы 90-101 эквивалентны соответствующим операторам программы 13.1 и повторно не описываются.

Оператор 110 представляет собой заголовок подпрограммы-функции с именем FX, имеющей в качестве списка формальных параметров одну переменную T.

По оператору 115 в неименованной общей области COMMON резервируются три ячейки для размещения переменных PI, SR и VIE из подпрограммы-функции FX: первая отводится для PI, вторая — для SR, третья — для VIE.

Оператор 117 присваивает постоянное значение переменной PI.

Операторы 118 и 119 описывают вычисления заданных законов относительного SR и переносного VIE движений с использованием формального параметра T. Их результаты присваиваются переменным SR и VIE соответственно.

Оператор 120 описывает вычисления функции FX, используя значения переменных SR и VIE. Вычисленное значение присваивается одноименной с функцией переменной FX, являющейся выходом подпрограммы. Формальный параметр T в данном случае используется при вычислении функции FX не непосредственно, а через значения переменных SR и VIE, в описания вычисления которых он входит в явном виде.

По оператору 130 осуществляется возврат в вызывающую программу в операторы 40N или 43N, откуда происходит обращение к подпрограмме FX для вычисления значения функции при фактических параметрах T или T+DT соответственно.

Оператор 131 указывает на конец подпрограммы-функции FX.

Операторы 140 и 170 представляют собой заголовки подпрограмм-функций с именами FY и FZ соответственно, имеющих в качестве списка формальных параметров также только одну переменную T.

По операторам 145 и 175 в неименованной общей области резервируются три ячейки для размещения переменных PI, SR и VIE из подпрограмм-функций FY и FZ. Это равнозначно тождественности значений одноименных переменных PI, SR и VIE во всех подпрограммах.

Хотя переменные SR и VIE в разных подпрограммах и тождественны, но определяются они только в подпрограмме FX при том значении фактического параметра, с которым происходит обращение к ней. Они остаются таковыми при любых обращениях к подпрограммам-функциям FY и FZ до тех пор, пока при следующем обращении к FX с другим значением фактического параметра они не изменят свое значение. Поэтому нужно организовать основную программу таким образом, чтобы обращение к FY и FZ происходило с тем же значением фактического параметра, с которым последний раз производилось обращение к подпрограмме FX.

Операторы 150 и 180 описывают вычисления функций FY и FZ, используя значения переменных SR и VIE, определенных в подпрограмме-функции FX, и присваивают вычисленные значения одноименным с функциями переменным FY и FZ, которые являются выходными значениями соответствующих подпрограмм.

По операторам 160 и 190 осуществляется возврат в вызывающую программу в операторы 41N и 44N или 42N и 45N, откуда происходит обращение к подпрограммам FY или FZ при фактических параметрах T и T+DT соответственно. Операторы 161 и 191 указывают на конец подпрограмм FY и FZ соответственно.

Отметим, что в программе 13.2 вместо подпрограмм, состоящих из операторов 110-191, конечно, можно использовать любые другие формы представления этих функций, например, подпрограммы (13.3)-(13.5) или (13.6)-(13.8). Все три варианта дают при выполнении программы 13.2 одинаковые и правильные результаты.

Однако, если бы мы удалили оптимизирующую вставку из операторов 40N-45N из программы 13.2, представив операторы 40-42 и 50-52 с использованием непосредственного обращения к подпрограммам

$$\begin{array}{ll}
 20 \quad VX = (FX(T+DT) - FX(T)) / DT & 40 \\
 \quad \quad VY = (FY(T+DT) - FY(T)) / DT & 41 \\
 \quad \quad VZ = (FZ(T+DT) - FZ(T)) / DT & 42 \quad (13.9) \\
 \quad \quad AX = (FX(T+DT) - 2 * FX(T) + FX(T-DT)) / (DT * DT) & 50 \\
 \quad \quad AY = (FY(T+DT) - 2 * FY(T) + FY(T-DT)) / (DT * DT) & 51 \\
 \quad \quad AZ = (FZ(T+DT) - 2 * FZ(T) + FZ(T-DT)) / (DT * DT) & 52
 \end{array}$$

то результаты расчета вдруг стали бы ошибочными. Вместе с этим применение подпрограмм-функций без оператора COMMON (13.3)-(13.5) и (13.6)-(13.8) давало бы правильные результаты.

Причина этого состоит в том, что переменные SR и VIE вычисляются при обращении к функции FX с указанным значением фактического параметра. Они используются затем в других подпрограммах FY и FZ с тем же значением фактического параметра, с которым последний раз производилось обращение к подпрограмме FX (из-за образования общей области).

Оптимизирующая вставка из операторов 40N-45N программы 13.2 производит обращение к функциям в правильном порядке. Переменные SR и VIE, использующиеся при вычислении функций FY и FZ, имеют соответствующие указанному фактическому параметру значения. Поэтому результаты получаются правильными. После присваивания их значений соответствующим переменным, они уже не зависят от последовательности их использования.

Все происходит правильно и точно также, как при описании подпрограмм-функций без использования оператора COMMON по примерам (13.3)-(13.5) и (13.6)-(13.8).

При удалении из программы 13.2 операторов 40N-45N и использовании примера (13.9) происходит нарушение нужного соответствия. В операторах 41-42 и 51-52 значения функций FY и FZ вычисляются по значениям SR и VIE, определенным не по указанным при обращении к этим функциям фактическим параметрам, а по последнему обращению к функции FX в операторах 40 и 50 (соответственно при T и T-DT). Это и приводит к ошибке. Надеемся, что дополнение 13.8 убедило вас в необходимости ясно осознавать производимые в программах изменения, всегда стараясь понимать процесс их работы.

Дополнение 13.9. Рассмотрим пример создания более универсальной программы для решения задач с применением ЭВМ на темы “Кинематика точки” (задания К-1 и К-2 [22]) и “Сложное движение точки” (задания К-7 [21] и К-10 [22]). В первом случае для плоской задачи (10.14) необходимо определить радиус кривизны траектории, во втором — для пространственной (10.16) только скорость и ускорение точки. Потребуем составления во всех случаях трех подпрограмм-функций, задающих уравнения движения материальной точки. Тогда для всех плоских задач, для которых требуется составить только два уравнения движения X и Y, третья подпрограмма-функция для координаты Z будет выглядеть следующим образом:

```
FUNCTION FZ (T)
  FZ=0.
  RETURN
END
```

(13.10)

В качестве исходной выберем программу 13.2. Совпадающие с ней операторы универсальной программы 13.3 имеют тот же номер, а совпадающие с программой 13.1 имеют после своего номера точку и цифру 1. Они повторно не описываются. Добавленные новые операторы имеют после своего номера букву U. Такую совмещенную программу можно представить, например, в следующем виде:

```
С      П Р О Г Р А М М А 13.3
      READ *, NG, NZ, NW                20U
      READ *, TN, TK, HT, DT            21U
      T=TN                               21
      PRINT 5, NG, NZ, NW                25U
5     FORMAT (/5X, ' ГРУППА ', I8, 5X,   26U
1     ' ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ' /7X, ' ПО КИНЕМАТИКЕ НОМЕР ',
2     I2, 5X, ' ВАРИАНТ ', I2)
      IF( (NZ.EQ.1) .OR. (NZ.EQ.2) ) PRINT 15                30.1U
```

15	FORMAT (6X, 'T', 12X, 'V', 12X, 'A', 10X, 'ATANG',	31.1
*	10X, 'RO')	
	IF ((NZ.EQ.7).OR.(NZ.EQ.10)) PRINT 17	30.2U
17	FORMAT (6X, 'T', 12X, 'V', 12X, 'A')	31.2U
20	XT=FX (T)	40N
	YT=FY (T)	41N
	ZT=FZ (T)	42N
	XTDT=FX (T+DT)	43N
	YTDT=FY (T+DT)	44N
	ZTDT=FZ (T+DT)	45N
	VX=(XTDT-XT)/DT	40
	VY=(YTDT-YT)/DT	41
	VZ=(ZTDT-ZT)/DT	42
	V=SQRT (VX*VX+VY*VY+VZ*VZ)	43
	AX=(XTDT-2.*XT+FX (T-DT))/ (DT*DT)	50
	AY=(YTDT-2.*YT+FY (T-DT))/ (DT*DT)	51
	AZ=(ZTDT-2.*ZT+FZ (T-DT))/ (DT*DT)	52
	A=SQRT (AX*AX+AY*AY+AZ*AZ)	53
	IF ((NZ.EQ.7).OR.(NZ.EQ.10)) GOTO 42	54U
	ATANG=(VX*AX+VY*AY)/V	60.1
	ANORM=SQRT (A*A-ATANG*ATANG)	70.1
	RO=V*V/ANORM	76.1
	PRINT 40, T, V, A, ATANG, RO	80.1
40	FORMAT (5 (1X, G12.5))	81.1
42	IF ((NZ.EQ.7).OR.(NZ.EQ.10)) PRINT 40, T, V, A	82U
	T=T+HT	90
	IF (T.LE.TK) GOTO 20	91
	STOP	100
	END	101

Оператор 20U осуществляет ввод номеров группы NG, задания NZ и варианта NW. По оператору 21U производится ввод начального TN и конечного TK значений времени движения, шага HT, с которым проводятся расчеты, и шага дифференцирования DT. Использование оператора READ позволяет вводить различные исходные данные без изменения основной программы.

Оператор 25U под руководством оператора 26U осуществляет печать заголовка, указывающего номера группы, задания и варианта (см. описание различных вариантов программного решения этого фрагмента в п. 7.6.8).

Условный логический оператор IF 30.1U анализирует на истинность логическое выражение, заключенное в скобки. Если оно истинно, т.е. NZ=1 или NZ=2, то управление передается оператору PRINT 15, который под управлением оператора 15 FORMAT выводит на печать заголовки таблицы расчета для

заданий К-1 и К-2 (что описано в программе 13.1). Если оно ложно, то начинает выполняться следующий исполнимый оператор программы — также условный логический оператор IF 30.2U.

Оператор 30.2U также анализирует аналогичное логическое выражение ((NZ.EQ.7).OR.(NZ.EQ.10)). Если оно истинно, т.е. NZ=7 или NZ=10, то выполняется стоящий с ним в одной строке оператор PRINT 17, выводящий на печать под управлением оператора 17 FORMAT заголовок таблицы расчета для заданий К-7 и К-10 (соответствующий оператор FORMAT, описанный в программе 13.2, имеет метку 15). Если оно ложно, то выполняется следующий за ним исполнимый оператор программы 40N.

В результате выполнения фрагмента из операторов 30.1U-31.2U будет напечатан один раз заголовок таблицы результатов для соответствующих заданий К-1 и К-2 или К-7 и К-10.

Условный логический оператор IF 54U при NZ=7 или NZ=10 будет передавать управление через оператор безусловного перехода GOTO оператору с меткой 42 для печати результатов расчета. При NZ=1 или NZ=2 выполняется следующий в программе оператор 60.1.

Условный логический оператор IF 82U с меткой 42 выполняется уже для обеих ветвей программы. При NZ=7 или NZ=10 логическое выражение в операторе 82U является истинным, вследствие чего будет выполнен стоящий с ним в одной строке оператор PRINT 40. Он выводит на печать под руководством оператора FORMAT (с меткой 40) результаты расчета текущего времени T, скорости V и ускорения A для заданий К-7 или К-10. Напомним, что список вывода может быть меньше количества спецификаций в операторе FORMAT.

При NZ=1 или NZ=2 выражение в операторе 82U станет ложным, поэтому выполняется следующий в программе оператор 90 (печать результатов расчета в этом случае уже выполнена оператором 80.1 под руководством оператора 81.1).

Три подпрограммы-функции (FX, FY и FZ) задающие уравнения движения материальной точки для программы 13.3, могут быть представлены в любой из рассматриваемых в пособии форм с добавлением подпрограммы FZ (13.10).

Исходные данные для работы программы 13.3 могут быть представлены для типовых примеров заданий К-1 (10.14) и К-7 (10.16) сборника [21] соответственно в виде (13.11) и (13.12):

$$114418, 1, 31 \quad (13.11)$$

$$0., 1., 0.05, 0.005$$

$$114418, 7, 31 \quad (13.12)$$

$$0., 0.44444, 0.02222, 0.00222$$

Рекомендуем при использовании программы 13.3 для решения своей задачи обязательно учесть дополнения 13.1 и 13.2, выполняющих защиту от возможных аварийных ситуаций.

13.3. Использование подпрограмм SUBROUTINE

При численном дифференцировании с использованием формул для производных функций в конечных разностях (10.1) и (10.2) удобнее описывать уравнения движения материальной точки в виде подпрограммы SUBROUTINE (см. п. 12.1.3). Она может иметь любое количество выходных значений, что позволяет помещать все уравнения движения в один программный модуль.

Вместе с этим необходимость оператора CALL для вызова подпрограммы требует другого программного решения. Для каждой переменной T, X, Y и Z можно описать массив, состоящий из трех элементов. Тогда можно в первых из них поместить значения указанных величин при фактическом параметре T, во вторых — при T+DT, в третьих — при T-DT. Теперь все последующие вычисления выполняются аналогично вышерассмотренным программам со значениями переменных, расположенных в соответствующих элементах массива. При этом сохраняется структура основных блоков, что и для программы 13.1.

Простейшая программа с использованием подпрограммы SUBROUTINE для определения радиуса кривизны траектории для уравнений (10.14) может быть реализована следующим образом:

```

C      П Р О Г Р А М М А 13.4
      DIMENSION T(3),X(3),Y(3)           10S
      DATA TN,TK,HT,DT/0.,1.,0.05,0.005/ 20.1
      T(1)=TN                             21
      PRINT 15                             30.1
15  FORMAT(6X,'T',12X,'V',12X,'A',10X,'ATANG',10X,'RO') 31.1
20  T(2)=T(1)+DT                          40S
      T(3)=T(1)-DT                         41S
      DO 30 M=1,3                          42S
          CALL K1U (M,T,X,Y)               43S
30  CONTINUE                              44S
      VX=(X(2)-X(1))/DT                    40
      VY=(Y(2)-Y(1))/DT                    41
      V=SQRT(VX*VX+VY*VY)                 43.1
      AX=(X(2)-2.*X(1)+X(3))/(DT*DT)     50
      AY=(Y(2)-2.*Y(1)+Y(3))/(DT*DT)     51
      A=SQRT(AX*AX+AY*AY)                 53.1
      ATANG=(VX*AX+VY*AY)/V              60.1

```

ANORM=SQRT (A*A-ATANG*ATANG)	70.1
RO=V*V/ANORM	76.1
PRINT 40, T (1) , V, A, ATANG, RO	80
40 FORMAT (5 (1X, G12.5))	81.1
T (1) =T (1) +HT	90
IF (T (1) .LE. TK) GOTO 20	91
STOP	100.1
END	101.1
SUBROUTINE K1U (M, T, X, Y)	110
DIMENSION T (3) , X (3) , Y (3)	112
X (M) =4. *T (M)	120
Y (M) =16. *T (M) **2-1.	121
RETURN	130
END	131

Программа 13.4 составлена на основании 13.1, поэтому операторы, совпадающие с ней, имеют после своего номера точку и цифру 1 (они повторно не описываются). Аналогичные операторам программы 13.1, но отличающиеся только использованием массивов для своей работы, после своего номера не имеют никаких добавлений, а новые операторы имеют после своего номера букву S.

Оператор 10S описывает одномерные массивы переменных T, X, Y, Z, под каждый из которых отводится место в памяти ЭВМ для размещения трех элементов.

Арифметический оператор присваивания 21 значение времени TN, с которого начинают выполняться расчеты, присваивает первому элементу массива T. Операторы 40S-41S заполняют вторые и третьи элементы одномерного массива T.

Оператор 42S — оператор цикла DO — организует цикл по M, в котором будут выполняться операторы 43S-44S (последний помечен меткой 30) при M=1, 2, 3.

Оператор 43S выполняется в цикле по M и осуществляет обращение к подпрограмме K1U со следующими входными фактическими параметрами: M — текущее значение параметра цикла или счетчика цикла; T — одномерный массив из трех элементов, в котором расположены: T(1) — текущее значение времени, при котором на данном шаге выполняются расчеты; T(2) — значение времени T(1)+DT; T(3) — значение времени T(1)-DT.

Выходными значениями подпрограммы K1U являются одномерные массивы X и Y из трех элементов каждый, содержащие значения уравнений движения материальной точки X и Y при соответствующем времени T(M): X(1) и Y(1) — при значении времени T(1); X(2) и Y(2) — при T(1)+DT; X(3) и Y(3) — при значении времени T(1)-DT.

Оператор 44S (CONTINUE) используется в качестве конечного оператора цикла. В зависимости от значения параметра цикла М, он передает управление на начало цикла оператору 43S, если $M < 3$ или $M = 3$, или передает управление следующему в программе оператору 40, если $M > 3$. Его использование в данном случае строго не обязательно. Можно убрать его из программы, поставив метку 30 на оператор CALL 43S.

Операторы 40-41 и 50-51 производят вычисления проекций скорости V и ускорения А по формулам (10.1) и (10.2), используя значения элементов массивов X и Y, в которых предварительно были помещены нужные значения уравнений движения материальной точки.

Первый элемент массива T выполняет роль текущего времени T в программе 13.1, поэтому операторы 80, 90 и 91 эквивалентны соответствующим операторам программы 13.1, но только теперь значение текущего времени присвоено T(1).

Оператор 110 представляет собой заголовок подпрограммы SUBROUTINE с именем K1U, имеющей в качестве списка формальных параметров целую переменную M и вещественные массивы T, X, Y.

Оператор 112 задает описание массивов T, X, Y, по которому им отводится место в памяти (по три элемента для каждого).

Операторы 120 и 121 вычисляют арифметические выражения, определяющие значения уравнений движения (10.14) в зависимости от значения элемента массива T(M), и присваивают вычисленные значения соответствующим элементам X(M) и Y(M).

По оператору 130 осуществляется возврат в вызывающую программу, а оператор 131 указывает на конец подпрограммы.

Дополнение 13.10. Подпрограмма K1U программы 13.4 отличается от (12.19) именем, введением номера элемента массива M в список формальных параметров оператора 110, дополнительным оператором 112 с описанием массивов T, X, Y, Z и заменой в операторах 120-121 ранее использованных переменных соответствующими элементами массива T(M), X(M), Y(M).

С учетом этих видоизменений подпрограмму K1U можно представить в любой из рассмотренных выше форм (12.22)-(12.24) или (12.36), что для подпрограммы (12.22) будет иметь вид:

```

SUBROUTINE K1U (M, X1, X2, X3)           110
DIMENSION X1 (3), X2 (3), X3 (3)       112
X2 (M) = 4. * X1 (M)                   120
X3 (M) = 16. * X1 (M) ** 2 - 1.        121
RETURN                                  130
END                                      131

```

Дополнение 13.11. При описании подпрограмм SUBROUTINE могут использоваться переменные, которых нет в списке формальных параметров. Например, SUBROUTINE K1U в программе 13.4 и в дополнении 13.10 можно записать, используя вместо чисел также переменные A, B, C (или любые другие). Их значения должны быть определены до первого их использования, например, с помощью оператора DATA (см. п. 7.7.1):

```
DATA A, B, C/4., 16., 1./ 115
```

Теперь в обеих подпрограммах с учетом вставляемого оператора 115 изменятся операторы 120-121, которые примут вид:

- в программе 13.4:

```
X (M) = A * T (M) 120
```

```
Y (M) = B * T (M) ** 2 - C 121
```

- в дополнении 13.10:

```
X2 (M) = A * X1 (M) 120
```

```
X3 (M) = B * X1 (M) ** 2 - C 121
```

Значения переменным (A, B, C) в подпрограмму можно передать и с использованием оператора COMMON, задавая их в основной программе, например, с помощью арифметических операторов присваивания (см. сегмент фортран-программы (12.35)). Тогда в обоих приведенных фрагментах вместо оператора 115 DATA следует использовать оператор COMMON:

```
COMMON A, B, C 115
```

Дополнение 13.12. Переменным, находящимся в помеченных блоках COMMON, можно присвоить начальные значения с помощью подпрограммы BLOCK DATA. Ее расположим после оператора END основной программы 13.4 и представим в виде, реализующем дополнение 13.11:

```
BLOCK DATA 103
```

```
COMMON /AA/A, B, C 105
```

```
REAL A/4./, B/16./, C/1./ 107
```

```
END 109
```

Оператор 103 представляет собой заголовок подпрограммы BLOCK DATA. Оператор 105 обеспечивает размещение переменных A, B и C в помеченном блоке с именем AA. Оператор 107 явного описания типа REAL описывает переменные A, B, C как вещественные, присваивая им одновременно начальные значения соответственно 4., 16., 1. Оператор 109 указывает на конец подпрограммы.

Теперь в подпрограмме SUBROUTINE K1U в любой из форм, представленных в дополнении 13.11, в операторе 115 следует использовать помеченный блок COMMON:

```
COMMON /AA/A, B, C 115
```

Оператор 115 устанавливает соответствие между одноименными переменными (А, В, С) в разных программных единицах, входящими в одноименный общий блок с именем АА. В подпрограмме BLOCK DATA им были присвоены начальные значения, поэтому и в подпрограмме SUBROUTINE K1UA=4., B=16., C=1.

Дополнение 13.13. Согласно дополнению 13.8 переделаем программу 13.4 для решения пространственных задач. Учтем специфику использования массивов (в данном случае при работе с подпрограммой SUBROUTINE), когда вместо применявшихся ранее переменных Т, Х, Y и Z используются элементы соответствующих одноименных массивов. Тогда дополнение к программе 13.4 для выполнения задания К-7 [21] с уравнениями движения (10.16) может иметь следующий вид:

```

DIMENSION T(3), X(3), Y(3), Z(3)                                10S
DATA TN, TK, HT, DT/0., 0.44444, 0.02222, 0.00222/             20
15 FORMAT(6X, ' T', 12X, ' V', 12X, ' A' )                      31
      CALL K7U (M, T, X, Y, Z)                                   43S
VZ=(Z(2)-Z(1))/DT                                              42
V=SQRT(VX*VX+VY*VY+VZ*VZ)                                       43
AZ=(Z(2)-2.*Z(1)+Z(3))/(DT*DT)                                  52
A=SQRT(AX*AX+AY*AY+AZ*AZ)                                       53
PRINT 40, T(1), V, A                                           80
SUBROUTINE K7U(M, T, X, Y, Z)                                    110
DIMENSION T(3), X(3), Y(3), Z(3)                                112
PI=3.141592653589                                              117
VIE=0.9*T(M)*T(M)-9.*T(M)**3                                    118
SR=16.-8.*COS(3.*PI*T(M))                                       119
X(M)=-SR/2.*SIN(VIE)                                           120
Y(M)=SR/2.*COS(VIE)                                            121
Z(M)=SR*COS(PI/6.)                                             122
RETURN                                                         130
END                                                             131

```

Операторы 20, 31, 43 и 53 полностью совпадают с такими же (по номеру) операторами программы 13.2, где они и описаны.

Оператор 10S описывает дополнительно к оператору с тем же номером в программе 13.4 еще один массив Z, состоящий из трех элементов, который входит дополнительно в список фактических параметров при вызове подпрограммы K7U оператором CALL 43S.

Операторы 42 и 52 определяют (аналогично операторам 40-41 и 50-51 соответственно) проекции скорости и ускорения материальной точки по

значениями элементов массива Z и присваивают вычисленные значения переменным VZ и AZ .

Отличие оператора 80 от такого же оператора в программе 13.2 состоит в том, что в нем роль текущего времени T выполняет первый элемент массива $T(1)$.

Оператор 110 отличается именем подпрограммы `SUBROUTINE K7U`, а также наличием в списке формальных параметров дополнительного массива Z , описываемого оператором 112.

Операторы 117-119 присваивают значения переменным: 117 — PI , 118 и 119 — VI и SR в зависимости от значения элемента массива $T(M)$.

Операторы 120-122 вычисляют арифметические выражения, определяющие значения уравнений движения (10.16), используя введенные переменные, и присваивают вычисленные значения соответствующим элементам $X(M)$, $Y(M)$ и $Z(M)$.

Операторы 130-131 эквивалентны соответствующим операторам программы 13.4.

Дополнение 13.14. Согласно дополнения 13.9 рассмотрим создание универсальной программы с применением подпрограммы `SUBROUTINE`. Для этого достаточно осознанно скомпоновать универсальную программу 13.3, выполняющую поставленную задачу, но использующую для своей работы подпрограммы-функции, и программу 13.4, в которой решены вопросы применения подпрограммы `SUBROUTINE`. Это можно сделать следующим образом:

```

С      П Р О Г Р А М М А 13.5
      DIMENSION T(3), X(3), Y(3), Z(3)                10
      READ *, NG, NZ, NW                               20U
      READ *, TN, TK, HT, DT                           21U
      T(1)=TN                                           21
      PRINT 5, NG, NZ, NW                               25U
5     FORMAT (/5X, ' ГРУППА ', I8, 5X,                26U
1     ' ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ' /7X, ' ПО КИНЕМАТИКЕ НОМЕР ',
2     I2, 5X, ' ВАРИАНТ ', I2)
      IF((NZ.EQ.1).OR.(NZ.EQ.2)) PRINT 15              30.1U
15    FORMAT(6X, ' T ', 12X, ' V ', 12X, ' A ', 10X, ' ATANG ', 31.1
*     10X, ' RO ')
      IF((NZ.EQ.7).OR.(NZ.EQ.10)) PRINT 17            30.2U
17    FORMAT(6X, ' T ', 12X, ' V ', 12X, ' A ')       31.2U
20    T(2)=T(1)+DT                                       40S
      T(3)=T(1)-DT                                       41S
      DO 30 M=1, 3                                       42S
          CALL KINU (M, T, X, Y, Z)                   43S

```

30	CONTINUE	44S
	VX=(X(2)-X(1))/DT	40
	VY=(Y(2)-Y(1))/DT	41
	VZ=(Z(2)-Z(1))/DT	42
	V=SQRT(VX*VX+VY*VY+VZ*VZ)	43
	AX=(X(2)-2.*X(1)+X(3))/(DT*DT)	50
	AY=(Y(2)-2.*Y(1)+Y(3))/(DT*DT)	51
	AZ=(Z(2)-2.*Z(1)+Z(3))/(DT*DT)	52
	A=SQRT(AX*AX+AY*AY+AZ*AZ)	53
	IF((NZ.EQ.7).OR.(NZ.EQ.10)) GOTO 42	54U
	ATANG=(VX*AX+VY*AY)/V	60.1
	ANORM=SQRT(A*A-ATANG*ATANG)	70.1
	RO=V*V/ANORM	76.1
	PRINT 40,T(1),V,A,ATANG,RO	80.1
40	FORMAT (5(1X,G12.5))	81.1
42	IF((NZ.EQ.7).OR.(NZ.EQ.10)) PRINT 40,T(1),V,A	82U
	T(1)=T(1)+HT	90
	IF(T(1).LE.TK) GOTO 20	91
	STOP	100
	END	101
	SUBROUTINE KINU(M,T,X,Y,Z)	110
	DIMENSION T(3),X(3),Y(3),Z(3)	112
	PI=3.141592653589	117
	VIE=0.9*T(M)*T(M)-9.*T(M)**3	118
	SR=16.-8.*COS(3.*PI*T(M))	119
	X(M)=-SR/2.*SIN(VIE)	120
	Y(M)=SR/2.*COS(VIE)	121
	Z(M)=SR*COS(PI/6.)	122
	RETURN	130
	END	131

Напомним, что введенные в базовых программах 13.3 и 13.4 операторы отмечены буквами после их номера: U — 13.3 и S — 13.4, которые они также сохраняют в программе 13.5. Эти операторы повторно не описываются.

Оператор 10 задает описание используемых массивов.

Арифметический оператор присваивания 21 начальное значение времени TN, с которого начинаются расчеты, присваивает первому элементу массива T.

Операторы 40S-44S, 40-41, 50-51 эквивалентны соответствующим операторам программы 13.4, а операторы 42-43 и 52-53 — операторам дополнения 13.13, где они подробно и рассмотрены.

Вследствие универсальности программы 13.5, имя вызываемой подпрограммы в операторе 43S выбрано KINU, так как оно не должно быть связано с номером задания (как было ранее K1U или K7U). Поэтому в подпрограммах, задающих уравнения движения материальной точки, при их работе с программой 13.5 следует изменить их имена на KINU. Этим достигается их согласование с именем вызываемой подпрограммы в операторе 43S программы 13.5.

Операторы 60.1-80.1 эквивалентны соответствующим операторам программы 13.1, на что указывает цифра 1 после их номера, а операторы 90-101 — операторам программы 13.4, где они и описаны.

Заметим, что во всех программах 13.3-13.5, в которых может определяться радиус кривизны траектории RO, желательно использовать дополнения 13.1 и 13.2. При этом следует обращать внимание только на номер оператора, так как цифра 1 после точки указывает на его эквивалентность соответствующему оператору в программе 13.1.

Подпрограмма SUBROUTINE (операторы 110-131) может быть представлена в любой из рассмотренных выше форм, но имя подпрограммы должно быть KINU и ее список формальных параметров также должен быть согласован с подпрограммой KINU:

- на первом месте указывается целая переменная, используемая в подпрограмме в качестве указателя номера элемента массива;
- на втором месте указывается массив, выполняющий роль времени при описании уравнений движения;
- третье-пятое место занимают массивы для переменных, определяющих проекции уравнений движения материальной точки на оси координат. Присутствие всех трех массивов обязательно и для плоских задач. В этом случае оператор 122 принимает форму

$$Z(M)=0.$$

122

Исходные данные для работы программы 13.5 для типовых примеров заданий К-1 и К-7 сборника [21] могут быть представлены в виде (13.11) и (13.12) соответственно.

ГЛАВА 14. ИСПОЛЬЗОВАНИЕ СТАНДАРТНЫХ ПОДПРОГРАММ ДЛЯ РЕШЕНИЯ ЗАДАЧ КИНЕМАТИКИ

В четвертом выпуске математического обеспечения ЕС ЭВМ [12] описывается 10 подпрограмм из пакета ПНП-SSP (см. п. 9) для численного дифференцирования функций: 5 обычной и 5 удвоенной точности (см. п. 15). Из них четыре (2 простой: DCAR, DBAR и 2 удвоенной точности: DDCAR и DDBAR) для функций, заданных аналитически и шесть (3 простой: DET3, DET5, DGT3 и 3 удвоенной точности: DDET3, DDET5 и DDGT3) для функций, заданных таблицей значений в равноотстоящих (DET3, DDET3, DET5, DDET5) и неравноотстоящих (DGT3, DDGT3) точках.

В задачах кинематики уравнения движения материальной точки обычно задаются аналитически, но стандартные подпрограммы могут производить только одно дифференцирование и с их помощью можно получить только значение скорости материальной точки.

Для получения ускорения материальной точки нужно сначала получить с помощью подпрограмм численного дифференцирования аналитических функций (DCAR, DDCAR, DBAR, DDBAR) значения проекций скорости в разных равноотстоящих точках. Затем полученную таблицу значений проекций скорости следует обработать с помощью подпрограмм численного дифференцирования функций, заданных таблично (DET3, DDET3, DET5, DDET5).

Рассмотрим эти подпрограммы подробнее, так как они имеют определенную самостоятельную ценность (кроме использования в задачах кинематики), особенно для численного дифференцирования функций, заданных таблицей значений.

14.1. Описание стандартных подпрограмм для численного дифференцирования

14.1.1. Подпрограммы DCAR, DBAR и DDCAR, DDBAR

Подпрограммы DCAR и DBAR производят дифференцирование аналитически заданной функции методом экстраполяции Ричардсона и Ромберга в центре интервала (DCAR) и на его границе (DBAR).

Подробное изложение используемого метода и его математического обоснования для пользования этими подпрограммами не столь существен-

но. Интересующиеся более полным описанием, чем излагаемое ниже, могут прочесть об этом для DCAR в [12, вып. 4, с. 23-29] и для DBAR в [12, вып. 4, с. 33-40].

Подпрограмма DCAR

```
SUBROUTINE DCAR (T, H, IH, FCT, D)
```

предназначена для вычисления в точке T производной от аналитически заданной функции FCT , которая дифференцируема 11 раз в области, содержащей замкнутый симметричный интервал $[T-H/, T+H/]$ радиуса $H/$. Производная определяется по значениям функции из этого замкнутого интервала. Параметр FCT должен быть описан оператором EXTERNAL (см. п. 12.2).

Ее формальными параметрами являются: T — точка, в которой вычисляется производная; H — число, абсолютное значение которого определяет замкнутый симметричный интервал $H/$; IH — входной параметр: если $IH.NE.0$ — подпрограмма вычисляет внутреннее значение HH ; при $IH=0$ — внутреннее значение HH полагается равным $H/$ (см. также нижеследующие замечания); FCT — имя внешней подпрограммы-функции, которая вычисляет необходимые значения функции (одно из уравнений движения материальной точки, задаваемых в виде подпрограммы FUNCTION $FCT(T)$); D — вычисленное значение производной.

Замечания: 1. $H=0.$, то вычислений нет.

2. Внутреннее значение HH , которое определяется в соответствии с IH , есть максимальная величина шага, используемого при вычислении центральных разделенных разностей (см. замечание 3). Если $IH.NE.0$, то подпрограмма вычисляет HH по критерию, который уравнивает ошибку округления и ошибку усечения. $HH.LE.H/$, так что все вычисления имеют место в замкнутом интервале $[T-H/, T+H/]$.

3. Вычисление производной D основано на методе экстраполяции Ричардсона и Ромберга. Он применяется к последовательности центральных разделенных разностей, соответствующих парам точек: $(T-(K*HH)/5, T+(K*HH)/5)$, $K=1, \dots, 5$.

Подпрограмма DBAR

```
SUBROUTINE DBAR (T, H, IH, FCT, D)
```

также предназначена для вычисления в точке T производной от аналитически заданной функции FCT . Она использует тот же метод, что и подпрограмма DCAR, и для ее работы справедливы те же замечания. Ее формальные параметры полностью совпадают с описанными для подпрограммы DCAR. Отличие заключается в том, что величина H

определяет другой замкнутый интервал $[T, T+H]$, на котором и проводятся все вычисления.

Подпрограммы DCAR и DBAR оформлены таким образом, что при обращении к ним задаваемые фактические параметры могут быть одинаковыми, что удобно при практической работе. При вызове для выполнения этих подпрограмм отличие может быть только в имени вызываемой подпрограммы. Например (см. п. 14.2):

```
CALL DCAR (T, DT, 1, FX, VX (M) )
CALL DBAR (T, DT, 1, FX, VX (M) )
```

Подпрограммы DDCAR и DDBAR

```
SUBROUTINE DDCAR (T, H, IH, FCT, D)
SUBROUTINE DDBAR (T, H, IH, FCT, D)
```

полностью аналогичны соответствующим подпрограммам DCAR и DBAR, но только составлены в удвоенной точности, правила и примеры работы с которой описаны в п. 15.

14.1.2. Подпрограммы DET3, DET5 и DDET3, DDET5

Подпрограммы DET3 и DET5 вычисляют вектор $D=(D_1, \dots, D_N)$ значений производной функции, заданной вектором $X=(X_1, \dots, X_N)$ ее значений X_I в N равноотстоящих точках $T(I)$ с шагом $T(I) - T(I-1) = H$ ($I=2, \dots, N$). За исключением конечных точек $T(1)$ и $T(N)$ для DET3 и точек $T(1), T(2), T(N-1)$ и $T(N)$ для DET5, $D(I)$ есть производная от интерполяционного многочлена Лагранжа 2-й (DET3) или 4-й (DET5) степени, построенного по трем (DET3) или пяти (DET5) последовательным точкам.

Интересующиеся более полным описанием, чем излагаемое ниже, могут прочесть об этом в работе [12, вып. 4] для DET3 на страницах 12-13 и для DET5 на 17-19.

Подпрограммы DET3 и DET5

```
SUBROUTINE DET3 (H, X, D, NDIM, IER)
SUBROUTINE DET5 (H, X, D, NDIM, IER)
```

предназначены для вычисления вектора значений производной функции, заданной таблицей значений в равноотстоящих точках. Их формальными параметрами являются: H — приращение последовательных значений аргумента (шаг $H > 0$, если они возрастают, и $H < 0$ — убывают); X — входной вектор значений функции размером $NDIM$; D — вычисленный вектор значений производной размером $NDIM$; $NDIM$ — размер векторов

X и D ; IER — код ошибки. При $IER=0$ ошибки нет и результатам вычислений можно доверять. Если $IER=-1$, то задано $NDIM < 3$ для $DET3$ или $NDIM < 5$ для $DET5$, поэтому не хватает точек для построения интерполяционного многочлена Лагранжа и никаких вычислений не производится. $IER=1$ указывает на то, что при задании фактических параметров приращение последовательных значений аргумента H оказалось равным нулю, что является ошибкой.

Отметим, что вектор-столбец D может совпадать для экономии используемой памяти с вектором столбцом X . Тогда при вызове подпрограммы в операторе `CALL` для этих двух формальных параметров будет использоваться на соответствующих местах один идентификатор в качестве фактического параметра. Например:

```
CALL DET3 (DT, VX, VX, NDIM, IERX)
```

В этом случае входной вектор-столбец VX после окончания работы подпрограммы не сохраняется и на его месте располагается вычисленный вектор значений производной.

Если X не совпадает с D , то он сохраняется, что для нашего примера может иметь вид:

```
SUBROUTINE DET3 (DT, VX, AX, NDIM, IERX)
```

Теперь вычисленный вектор значений производных располагается в одномерном массиве AX , а входной вектор-столбец VX сохраняет свои значения после окончания работы подпрограммы.

Подпрограммы DDET3 и DDET5

```
SUBROUTINE DDET3 (H, X, D, NDIM, IER)
SUBROUTINE DDET5 (H, X, D, NDIM, IER)
```

также полностью аналогичны соответствующим вышеописанным подпрограммам $DET3$ и $DET5$, но только составлены в удвоенной точности (см. п. 15).

14.2. Использование стандартных подпрограмм для численного дифференцирования

Простейшая программа для определения радиуса кривизны траектории с использованием стандартных подпрограмм состоит из тех же основных блоков (см. п. 13.1), что и программы 13.1-13.5, но с учетом специфических особенностей: изменяются блоки 1), 4) и 5), а 9) совмещается с блоком 4).

В блоке 1) указываются имена внешних функций в операторе EXTERNAL и приводится описание используемых массивов.

В блоке 4) подпрограммами DCAR или DBAR производится формирование массивов проекций скорости, осуществляющих их заполнение с шагом дифференцирования DT. Определяются абсолютные значения скорости на каждом шаге.

В блоке 5) определяются проекции ускорения материальной точки. Используются значения проекций скорости в равноотстоящих точках и подпрограммы DET3 или DET5. Вычисляются абсолютные значения ускорения в этих точках.

Такая модифицированная блок-схема может быть реализована с использованием подпрограмм DCAR и DET3 следующим образом:

```

C   П Р О Г Р А М М А 14.1
C   ЧИСЛЕННОЕ ДИФФЕРЕНЦИРОВАНИЕ ЗАДАНИЯ К-1 С ИСПОЛЬЗОВАНИЕМ
C   СТАНДАРТНЫХ ПОДПРОГРАММ DCAR И DET3
      EXTERNAL FX, FY                                10
      DIMENSION VX(21), VY(21), V(21), AX(21), AY(21), A(21),
*      ATANG(21), ANORM(21), RO(21)                11
      DATA TN, TK, DT, NDIM/0.45, 0.55, 0.005, 21/  20
      T=TN                                           21
      M=1                                            22
      PRINT 15                                       30
15  FORMAT (6X, 'T', 12X, 'V', 12X, 'A', 10X, 'AT', 10X, 'RO') 31
20  CALL DCAR(T, DT, 1, FX, VX(M))                  40
      CALL DCAR(T, DT, 1, FY, VY(M))                41
      V(M)=SQRT(VX(M)*VX(M)+VY(M)*VY(M))            43
      T=T+DT                                         44
      M=M+1                                          45
      IF(T.LE.TK) GOTO 20                           46
      CALL DET3(DT, VX, AX, NDIM, IERX)              50
      CALL DET3(DT, VY, AY, NDIM, IERY)              51
      DO 22 N=1, NDIM                                54
      A(N)=SQRT(AX(N)*AX(N)+AY(N)*AY(N))             56
      ATANG(N)=(VX(N)*AX(N)+VY(N)*AY(N))/V(N)        60
      ANORM(N)=SQRT(A(N)*A(N)-ATANG(N)*ATANG(N))     70
      RO(N)=V(N)*V(N)/ANORM(N)                      76
22  CONTINUE                                         77
      T=TN                                           78
      DO 50 N=1, NDIM                                79
      PRINT 40, T, V(N), A(N), ATANG(N), RO(N)       80
40  FORMAT (5(1X, G12.5))                            81
50  T=T+DT                                           82

```

PRINT 60, IERX, IERY	83
60 FORMAT (5X, ' КОДЫ ОШИБОК: IERX=', I2, 3X, ' IERY=', I2)	84
STOP	100
END	101

Оператор 10 объявляет FX и FY как имена внешних подпрограмм, которые будут использоваться далее в качестве фактических параметров при обращении к подпрограмме DCAR.

Оператор 11 задает описание массивов, по которому отводится место в памяти для их размещения, причем в каждом содержится 21 элемент. Это число NDIM=21 соответствует количеству равноотстоящих точек, в которых будет определяться значения проекций скорости, а затем ускорения и всех других величин. Оно равняется числу повторений всех циклов в программе и должно быть согласовано со значением начального TN и конечного ТК времени расчета и шагом дифференцирования DT

$$NDIM = ((TK - TN) / DT) + 1, \quad (14.1)$$

где итоговый результат преобразовывается к целому числу.

В результате соответствия, определенного оператором 20, указанные в нем переменные получают в момент загрузки программы следующие значения: TN=0.45, ТК=0.55, DT=0.005, NDIM=21 (значения TN и ТК желательно выбирать так, чтобы величина времени T1 была посередине между ними).

Операторы 21, 30 и 31 эквивалентны соответствующим операторам программы 13.1.

Оператор 22 целой переменной M, использующейся при заполнении массивов в качестве указателя номера элемента, присваивает его начальное значение, равное единице.

Оператор 40 CALL (с меткой 20) осуществляет обращение к стандартной подпрограмме DCAR со следующими фактическими параметрами: T — текущее значение времени, при котором вычисляется производная; DT — шаг дифференцирования; l — значение внутреннего параметра подпрограммы IH, при котором в процессе дифференцирования определяется оптимальное значение DT (см. п. 14.1.1); FX — имя внешней подпрограммы-функции, вычисляющей значения соответствующего уравнения движения; VX(M) — значение производной, которое помещается в ячейку массива VX с номером M.

Оператор 41 также осуществляет обращение к DCAR при тех же значениях фактических параметров T, DT и IH, но с использованием подпрограммы-функции FY и помещением результата дифференцирования в ячейку массива VY с тем же номером M.

Оператор 43 определяет абсолютное значение скорости, вычисляя корень квадратный из суммы квадратов ее проекций $VX(M)$ и $VY(M)$, и присваивает его элементу массива V с номером M .

Арифметический оператор присваивания 44 заменяет значение текущего времени T на $T+DT$ (после первого раза выполнения операторов 40-43 $T=TN=0.45$, $DT=0.005$ и станет $T=0.455$).

Арифметический оператор присваивания 45 заменяет значение целой переменной M , используемой в качестве указателя номера элемента массива, на $M+1$, которое теперь становится равным 2 (т.е. теперь будут заполняться вторые элементы массивов).

Условный логический оператор IF 46, эквивалентный оператору 91 программы 13.1, анализирует на истинность логическое выражение, заключенное в скобки.

Если оно истинно, т.е. текущее T меньше его конечного значения $TK=0.55$, то выполняется оператор безусловного перехода $GOTO$ (см п. 7.5.1). Он передает управление оператору с меткой 20, осуществляющему обращение к подпрограмме $DCAR$. Теперь операторы программы 40-43 станут повторяться при следующих значениях $T=0.455$ и $M=2$. В результате будут вычислены и заполнены вторые элементы массивов $VX(2)$, $VY(2)$ и $V(2)$.

Затем оператор 44 снова заменит текущее значение T на $T+DT$ (станет $T=0.46$ сек), оператор 45 изменит значение M на $M+1$ (будет $M=3$), а условный логический оператор 46 снова передаст управление оператору с меткой 20 для очередного повторения операторов 40-46 (пока T меньше $TK=0.55$ сек.).

Так будет повторяться до тех пор, пока после очередного выполнения по арифметическому оператору присваивания 44 значение T не станет равным TK , а $M=21$. После этого оператор 46 опять передаст управление оператору 20, т.к. выражение отношения в $(T.LE.TK)$ все еще останется истинным при $T=TK$.

Теперь программа будет повторена в последний раз, заполнятся 21-е элементы массивов VX , VY , V и после очередного выполнения оператора 44 значение T станет больше TK (а $M=22$). Тогда выражение в скобках $(T.LE.TK)$ в условном логическом операторе 46 станет ложным, поэтому оператор $GOTO$ будет пропущен и станет выполняться следующий в программе за логическим IF оператор 50.

Оператор 50 осуществляет обращение к стандартной подпрограмме $DET3$ с фактическими параметрами: DT — приращение аргумента T (в операторе 44), с которым происходило заполнение массивов; VX — входной вектор значений функции; AX — вычисленный вектор значений производной; $NDIM$ — размер одномерных массивов VX и AX (для программы 14.1 $NDIM=21$); $IERX$ — код ошибки при определении проекций ускорения AX .

Оператор 51 осуществляет аналогичное оператору 50 обращение к DET3 при тех же значениях фактических параметров DT и NDIM, но при задании вектора значений проекции скорости VY. Вычисленный вектор проекций ускорения точки расположится в одномерном массиве AY, а в IERY — код ошибки.

Отметим, что подпрограмма DET3 определяет значения производных проекций скорости, т.е. проекции ускорения, сразу во всех указанных точках, поэтому обращение к подпрограмме производится только один раз для каждой координаты.

Оператор 54 — оператор цикла DO — организует цикл по N, в котором будут выполняться операторы 56-77 (последний помечен меткой 22) при $N=1, \dots, \text{NDIM}$. При каждом значении N они производят вычисления полного $A(N)$, тангенциального $\text{ATANG}(N)$ и нормального $\text{ANORM}(N)$ ускорений (при первом выполнении цикла заполняются первые элементы массивов, ..., при 21-м — 21-е).

Операторы 78-84 организуют вывод результатов расчета на печать.

Арифметический оператор присваивания 78, эквивалентный оператору 21, переменной T опять присваивает значение времени TN, с которого начинались выполняться расчеты.

Оператор 79 — оператор цикла DO — организует цикл по N, в котором будут выполняться операторы 80-82 (последний помечен меткой 50) при $N=1, \dots, \text{NDIM}$.

Оператор 80 выполняется в цикле по N и производит под руководством оператора 81 вывод на печать значений времени T, скорости $V(N)$, полного $A(N)$ и тангенциального $\text{ATANG}(N)$ ускорений, радиуса кривизны $\text{RO}(N)$ (при первом выполнении цикла печатаются первые элементы массивов, ..., при 21-м — 21-е).

Оператор 82 при каждом повторении цикла увеличивает значение текущего времени T на используемое ранее при расчетах в операторе 44 приращение DT. Этим достигается соответствие при выводе на печать между величиной T и результатами вычислений, хранящихся в ячейках массивов.

Оператор 83 под руководством оператора 84 печатает результаты кодов ошибок при определении проекций ускорения. Они должны быть равны нулю при правильной работе подпрограммы DET3 (см. п. 14.1.2).

Оператор 100 указывает на конец вычислений, оператор 101 — на конец основной программы.

С программой 14.1 можно использовать подпрограммы-функции FX и FY в любой из рассмотренных выше форм (12.11)-(12.12), (12.15)-(12.16), (12.30)-(12.31), по дополнениям 13.6 и 13.7, а также в виде (12.37)-(12.38) с дополнением в основную программу фрагмента (12.35).

Использование подпрограмм DBAR и DET5 в программе 14.1 аналогично рассмотренному, так как они описываются с применением одинаковых формальных параметров. Для этого нужно просто в программе 14.1 исправить имена вызываемых подпрограмм: вместо DCAR — DBAR, вместо DET3 — DET5.

Дополнение 14.1. Для большей наглядности в программе 14.1 печать результатов производилась отдельно после определения всех результатов расчета. Поэтому использовались два цикла: для вычислений (операторы 54-77) и для печати результатов (операторы 79-82). Такой подход требует большого количества используемых массивов. Требуемый объем памяти можно существенно сократить, совместив оба цикла, используя простые переменные для V, A, ATANG и RO и выполняя печать результатов вычислений сразу после их определения. Для этого в программе 14.1 следует выполнить следующие изменения:

DIMENSION VX(21), VY(21), AX(21), AY(21)	11
T=TN	53/78/
DO 50 N=1, NDIM	54
V=SQRT(VX(N)*VX(N)+VY(N)*VY(N))	55/43/
A=SQRT(AX(N)*AX(N)+AY(N)*AY(N))	56
ATANG=(VX(N)*AX(N)+VY(N)*AY(N))/V	60
ANORM=SQRT(A*A-ATANG*ATANG)	70.1
RO=V*V/ANORM	76.1
PRINT 40, T, V, A, ATANG, RO	80.1
40 FORMAT (5(1X, G12.5))	81.1
50 T=T+DT	82

Теперь оператор 11 задает описание массивов из 21-го элемента только для проекций скорости VX, VY и ускорения AX, AY.

Оператор 43 убирается со своего места в программе 14.1 и помещается в измененном виде под номером 55. Оператор 78 выносится со своего места и располагается также перед началом цикла под номером 53. Для них старые номера указываются в наклонных скобках после их нового номера 55/43/ и 53/78/.

Первый и второй циклы DO объединяются: в начальном операторе первого цикла 54 указывается метка 50 конечного оператора 82 второго цикла, а конечный оператор первого цикла 77 и начальный оператор второго цикла 79 удаляются из программы.

С учетом этих изменений операторы 53-82 дополнения 14.1 располагаются сразу после оператора 51 программы 14.1.

Оператор 53 переменной T опять присваивает значение TN, с которого начались выполняться расчеты.

Оператор 54 — оператор цикла DO — организует цикл по N, в котором будут выполняться операторы 55-82 (последний помечен меткой 50) при N=1, ..., NDIM.

Операторы 55/43/, 56 и 60 вычисляют при первом выполнении цикла значения скорости, полного и тангенциального ускорений. При этом используются значения первых элементов массивов VX, VY, AX, AY, определенных при T=TN, и результат присваивается соответствующим переменным V, A и ATANG.

Операторы 70, 76, 80 и 81 эквивалентны соответствующим операторам программы 13.1, на что указывает цифра 1, стоящая после их номера. Операторы 70 и 76 определяют нормальное ускорение ANORM и радиус кривизны RO. Оператор 80 под руководством оператора 81 выводит на печать соответствующее значение времени T и результаты расчета V, A, ATANG и RO после каждого выполнения цикла (на 1-м шаге T=TN, на 21-м T=TK).

Оператор присваивания 82 увеличивает значение T на приращение DT и повторяется, начиная с оператора 55/43/, выполнение цикла при N=2 (используются 2-е элементы массивов VX, VY, AX, AY). После определения значений переменных V, A, ATANG и RO они выводятся на печать операторами 80-81, а оператор 82 опять увеличивает T на приращение DT.

Затем рассматриваемая последовательность операторов 55/43/-82 будет повторяться каждый раз при N=3, ..., 21. На каждом шаге будут использоваться соответственно 3-е, ..., 21-е элементы массивов VX, VY, AX, AY, определяться значения V, A, ATANG и RO для данного времени T и вместе с указанием его значения выводиться на печать. Конечный оператор цикла 82 будет увеличивать текущее значение T на DT, приводя его в соответствие с вычисляемыми величинами следующего шага.

Вариант решения данного вопроса по дополнению 14.1 с использованием переменных экономит требуемую для работы программы память ЭВМ. Использование массивов сохраняет результаты расчетов. Это является достоинством в том случае, если их нужно использовать для последующих вычислений.

Дополнение 14.2. Согласно дополнению 13.13 переделаем программу 14.1 для решения пространственных задач. Учтем специфику использования стандартных подпрограмм. Удалим из программы 14.1 операторы, вычисляющие величины (тангенциальное ATANG и нормальное ANORM ускорения, радиус кривизны RO), которые не требуется определять при изучении сложного движения точки. С учетом этого дополнение к программе 14.1 для выполнения задания К-7 [21] может иметь следующий вид:

```

EXTERNAL FX, FY, FZ                                10
DIMENSION VX (21), VY (21), VZ (21), V (21), AX (21),    11
*           AY (21), AZ (21), A (21)
DATA TN, TK, DT, NDIM/0.2, 0.24444, 0.00222, 21/        20
15 FORMAT (6X, ' T ', 12X, ' V ', 12X, ' A ' )          31
CALL DCAR (T, DT, 1, FZ, VZ (M) )                      42
V (M) =SQRT (VX (M) *VX (M) +VY (M) *VY (M) +VZ (M) *VZ (M) ) 43
CALL DET3 (DT, VZ, AZ, NDIM, IERZ)                    52
A (N) =SQRT (AX (N) *AX (N) +AY (N) *AY (N) +AZ (N) *AZ (N) ) 56
PRINT 40, T, V (N), A (N)                              80
PRINT 60, IERX, IERY, IERZ                             83
60 FORMAT (5X, ' КОДЫ ОШИБОК: IERX=' , I2, 3X, ' IERY=' , I2, 84
*           3X, ' IERZ=' , I2)

```

Оператор 10 объявляет дополнительно (к программе 14.1) FZ как имя внешней подпрограммы. Оно будет использоваться в качестве фактического параметра при обращении к подпрограмме DCAR.

Оператор 11 задает описание массивов, по которому отводится место в памяти для их размещения, причем добавляются массивы для производных скорости VZ и ускорения AZ, а удаляются для ATANG, ANORM и RO по сравнению с программой 14.1.

В результате соответствия, устанавливаемого оператором 20 DATA, перечисленные в нем переменные приобретают следующие начальные значения: TN=0.2, TK=0.24444, DT=0.00222, NDIM=21, которые также связаны между собой соотношением (14.1).

Оператор 31 эквивалентен соответствующему оператору программы 13.2.

Оператор 42 осуществляет обращение к стандартной подпрограмме DCAR при тех же значениях фактических параметров T, DT и IH, что и описанный в программе 14.1 оператор 40, но с использованием подпрограммы-функции FZ и помещением результата дифференцирования в ячейку массива VZ с тем же номером M.

Операторы 43 и 56 по значениям проекций скорости VX(M), VY(M), VZ(M) и ускорения AX(M), AY(M), AZ(M) определяют их абсолютные значения, вычисляя корень квадратный из суммы квадратов проекций, и присваивают их соответственно элементам массивов V и A с номером M.

Оператор 52 осуществляет аналогично оператору 50 программы 14.1 обращение к стандартной подпрограмме DET3 при тех же значениях фактических параметров DT и NDIM, но при задании проекции скорости VZ и с помещением результата дифференцирования в одномерном массиве AZ, причем код ошибки работы подпрограммы будет содержать целая переменная IERZ.

Оператор 80 выполняется в цикле по N и производит вывод на печать под руководством оператора 81 значений времени T, скорости V(N) и ускорения A(N).

Оператор 83 под руководством оператора 84 выводит на печать дополнительно к программе 14.1 код ошибки IERZ при определении проекций ускорения AZ подпрограммой DET3.

В дополнении 14.2 можно использовать подпрограммы FX, FY и FZ в любой из рассмотренных выше форм (13.3)-(13.5), (13.6)-(13.8) и др., кроме их представления с использованием общей области COMMON в программе 13.2 (операторы 110-191). Для этого случая не произведено согласование обращений к подпрограммам FX, FY и FZ, чтобы оно происходило с одинаковым значением фактического параметра.

Рекомендуем также для дополнения 14.2 выполнить замену используемых массивов V и A соответствующими переменными по типу дополнения 14.1. Форму подпрограммы-функции FZ (13.13) также необходимо использовать при решении плоских задач.

Дополнение 14.3. Согласно дополнения 13.9 рассмотрим создание универсальной программы с применением стандартных подпрограмм. Для этого достаточно осознанно скомпоновать две программы: универсальную программу 13.3, выполняющую поставленную задачу, но использующую для своей работы подпрограммы-функции, и программу 14.1, в которой решены вопросы применения стандартных подпрограмм. Учтем также дополнение 14.1, представив программу 14.2 с минимальным использованием массивов. Это можно сделать следующим образом:

```

С      П Р О Г Р А М М А 14.2
      EXTERNAL FX, FY, FZ                10
      DIMENSION VX(21), VY(21), VZ(21), AX(21), AY(21),    11
      *      AZ(21)
      READ *, NG, NZ, NW                  20U
      READ *, TN, TK, DT, NDIM            21U
      T=TN                                21
      M=1                                  22
5     PRINT 6, NG, NZ, NW                 25U
6     FORMAT (/5X, ' ГРУППА ', I8, 5X,    26U
1     ' ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ' /7X, ' ПО КИНЕМАТИКЕ НОМЕР ',
2     I2, 5X, ' ВАРИАНТ ', I2)
      IF((NZ.EQ.1).OR.(NZ.EQ.2)) PRINT 15    30.1U
15    FORMAT (6X, ' T ', 12X, ' V ', 12X, ' A ', 10X, ' ATANG ',  31.1
      *      10X, ' RO ')
      IF((NZ.EQ.7).OR.(NZ.EQ.10)) PRINT 17  30.2U
17    FORMAT (6X, ' T ', 12X, ' V ', 12X, ' A ')  31.2U

```

20	CALL DCAR(T,DT,1,FX,VX(M))	40
	CALL DCAR(T,DT,1,FY,VY(M))	41
	CALL DCAR(T,DT,1,FZ,VZ(M))	42
	T=T+DT	44
	M=M+1	45
	IF(T.LE.TK) GOTO 20	46
	CALL DET3(DT,VX,AX,NDIM,IERX)	50
	CALL DET3(DT,VY,AY,NDIM,IERY)	51
	CALL DET3(DT,VZ,AZ,NDIM,IERZ)	52
	T=TN	53/78/
	DO 50 N=1,NDIM	54
	V=SQRT(VX(N)*VX(N)+VY(N)*VY(N)+VZ(N)*VZ(N))	55/43/
	A=SQRT(AX(N)*AX(N)+AY(N)*AY(N)+AZ(N)*AZ(N))	56
	IF((NZ.EQ.7).OR.(NZ.EQ.10)) GOTO 42	58
	ATANG=(VX(N)*AX(N)+VY(N)*AY(N))/V	60
	ANORM=SQRT(A*A-ATANG*ATANG)	70.1
	RO=V*V/ANORM	76.1
	PRINT 40,T,V,A,ATANG,RO	80.1
40	FORMAT(5(1X,G12.5))	81.1
42	IF((NZ.EQ.7).OR.(NZ.EQ.10)) PRINT 40,T,V,A	82U
50	T=T+DT	82
	IF((NZ.EQ.1).OR.(NZ.EQ.2)) PRINT 60,IERX,IERY	83
60	FORMAT(5X,'КОДЫ ОШИБОК: IERX=',I2,3X,'IERY=',I2)	84
	IF((NZ.EQ.7).OR.(NZ.EQ.10)) PRINT 61,IERX,IERY,	85
	* IERZ	
61	FORMAT(5X,'КОДЫ ОШИБОК: IERX=',I2,3X,'IERY=',	86
	* I2,3X,'IERZ=',I2)	
	STOP	100
	END	101

Операторы, введенные в базовых программах 13.1 или 13.3, отмечены соответственно точкой и цифрой 1 или буквой U после их номера. Они повторно не описываются.

Оператор 10 объявляет FX, FY, FZ как имена внешних подпрограмм, которые будут использоваться далее в качестве фактических параметров при обращении к подпрограмме DCAR.

Оператор 11 задает описание используемых массивов.

Оператор 21 начальное значение TN, с которого начинаются расчеты, присваивает переменной T.

Оператор 22 целой переменной M, используемой при заполнении массивов в качестве указателя номера элемента, присваивает его начальное значение, равное единице.

Операторы 40-41 и 44-51 эквивалентны соответствующим операторам программы 14.1, а операторы 42 и 52 — операторам дополнения 14.2, где они подробно рассмотрены.

Оператор 53/78/ переменной Т опять присваивает значение TN, с которого начался расчет. Этим подготавливается соответствие текущего времени Т выводимым результатам расчета на печать.

Оператор 54 организует цикл по N, в котором будут выполняться операторы 55/43/-82 при N=1, ..., NDIM.

Операторы 55/43/-56 выполняются в цикле по N и при каждом значении N (от 1 до 21) производят вычисления скорости V и полного ускорения A. При этом применяются N-ные значения элементов используемых массивов, но результат вычислений каждый раз присваивается переменным V и A.

Условный логический оператор IF 58 анализирует логическое выражение, заключенное в скобки. Если оно истинно (для заданий К-7 [21] или К-10 [22]), то результат расчета по оператору безусловного перехода 42 отправляется на печать. Если оно ложно (для заданий К-1 [21] или К-2 [22]), то выполняется нижеследующая группа операторов 60-80.1. Они определяют тангенциальное ATANG и нормальное ANORM ускорения, радиус кривизны RO и также выводят на печать результаты вычислений.

Во всех программах, в которых может определяться радиус кривизны траектории RO (13.3-13.5 и 14.1-14.2), желательно использовать дополнения 13.1 и 13.2.

Конечный оператор цикла 82 увеличивает значение Т на используемое ранее при расчетах в операторе 44 приращение DT. Этим достигается согласование времен Т при определении проекций скорости операторами 40-46 (хранящихся в массивах VX, VY, VZ) и при вычислениях данного цикла, сопровождающихся на каждом шаге печатью результатов.

Операторы 83 и 85 осуществляют печать соответствующих кодов ошибок подпрограммы DET3 при определении проекций ускорения. При правильной работе коды ошибок равны нулю.

Оператор 100 указывает на конец вычислений, а оператор 101 указывает на конец основной программы.

Исходные данные к программе 14.2 для типовых примеров заданий К-1 [21, с. 60] и К-7 [21, с. 99] могут быть представлены соответственно в виде (14.2) и (14.3):

$$\begin{aligned} 114418, 1, 31 & & (14.2) \\ 0.45, 0.55, 0.005, 21 & & \end{aligned}$$

$$\begin{aligned} 114418, 7, 31 & & (14.3) \\ 0.2, 0.24444, 0.00222, 21 & & \end{aligned}$$

Напомним, что соотношение между величинами TN, TK, DT и NDIM определяется равенством (14.1).

ГЛАВА 15. ВЫЧИСЛЕНИЯ С УДВОЕННОЙ ТОЧНОСТЬЮ

Применяемые в п. 10 для численного дифференцирования формулы с использованием конечных разностей (10.1) и (10.2) очень чувствительны к изменению величины шага дифференцирования ΔT .

В этом можно убедиться при работе с программами 13.1-13.5. Сначала с уменьшением шага дифференцирования ΔT ошибка расчета будет уменьшаться. При дальнейшем уменьшении ΔT она станет увеличиваться вплоть до лавинообразного нарастания и катастрофической потери верных знаков.

Особенно просто и наглядно это проверяется при работе с использованием программы KINMP (см. п. 10). Она не только производит сравнение вашего решения с правильным численным решением, но и с правильным аналитическим при заданном времени T_1 . Из него только и можно определить ошибку численного решения и ее конкретное возрастание при сильном уменьшении ΔT .

Этот эксперимент с варьированием ΔT позволяет в данном случае наглядно показать необходимость внимательного и осознанного подхода к численным расчетам на ЭВМ, а также важность умения вычислений с удвоенной точностью.

При использовании удвоенной точности можно определить факт возникновения подобной числовой неустойчивости без получения аналитического решения, которое в общем случае для ряда задач может быть весьма трудоемким или даже невозможным. Для этого достаточно сравнить между собой результаты двух численных решений задачи с обычной и удвоенной точностью, между которыми в подобных ситуациях обнаружится существенное расхождение.

Использование удвоенной точности позволяет намного отодвинуть границу возникновения катастрофической потери верных знаков расчета (в сторону уменьшения величины ΔT). При тех значениях ΔT , когда для обычной точности уже происходит лавинообразное возрастание ошибки расчета (вследствие нарастания ошибок округления и т.п.), при использовании удвоенной точности еще получают точные и правильные результаты, отчего и возникает расхождение между ними.

Возможен целый ряд других ситуаций, некоторые из которых рассмотрены в [27], когда необходимо работать с использованием удвоенной точности. Рассмотрим кратко те вопросы, с которыми приходится сталкиваться при использовании удвоенной точности на примере рассмотренных программ для решения задач кинематики.

15.1. Переменные удвоенной точности

Данные каждого типа характеризуются длиной, определяющей размер занимаемой памяти ЭВМ. Вещественные переменные обычной точности (или стандартной длины) занимают одну ячейку длиной 4 байта (единица памяти). Они называются также данными типа REAL и могут дополнительно не описываться. Тогда согласно неявному описанию типа к ним относятся все переменные, начинающиеся с букв от A до H и от O до Z.

Если нужно обойти это ограничение, то для описания вещественных переменных обычной точности используются эквивалентные операторы REAL или REAL *4.

При вычислениях с удвоенной точностью для значений переменных отводят по две последовательные ячейки памяти, каждая из которых имеет длину 4 байта. Таким образом, переменная удвоенной точности занимает в памяти ЭВМ всего 8 байт (поэтому иногда ее называют удвоенной значностью).

Числа с удвоенной точностью описываются с помощью специального оператора DOUBLE PRECISION (двойная точность). Он имеет вид

```
DOUBLE PRECISION A, B, C, ...
```

(15.1)

где A, B, C, ... — идентификаторы соответствующих переменных, массивов или наименования подпрограмм-функций.

Вместо оператора DOUBLE PRECISION для описания удвоенной точности применяют также оператор REAL *8 (т.е. в операторе явного описания типа REAL просто указывают длину переменных удвоенной точности). Тогда оператор (15.1) примет вид:

```
REAL *8 A, B, C, ...
```

(15.2)

Поэтому нижеследующие два оператора в подпрограмме KINU эквивалентны:

```
DOUBLE PRECISION PI, SR, VIE, T, X, Y, Z
```

111

```
REAL *8 PI, SR, VIE, T, X, Y, Z
```

111

Для записи чисел с удвоенной точностью используется экспоненциальная форма записи, которая имеет такой же вид, как и для действительных чисел. Только вместо буквы E пишется буква D: 0.D0, 1.D0, 0.05D0, 0.005D0, 12.7D3, -7.23D-4.

15.2. Неявный оператор типа — IMPLICIT

Оператор IMPLICIT определяет тип и длину величин по первой букве имени, указанной в нем. Его общая форма имеет вид:

```
IMPLICIT тип1 (S1) , тип2 (S2) , . . . , типN (SN)
```

где тип I (I=1,2, ..., N) — указатель типа REAL, INTEGER (а также не рассматриваемые здесь COMPLEX и LOGICAL);

SI (I=1,2,...,N) — список, содержащий один или несколько символов из набора A, B, ..., Z, разделенных запятыми.

Величины (переменные, массивы и функции), имена которых начинаются с указанных в соответствующем списке в скобках букв, относятся к тому типу, который определяет стоящий перед этими скобками соответствующий указатель типа (REAL или INTEGER).

Например, оператор 111 можно представить с использованием неявного оператора типа IMPLICIT в виде:

- для обычной точности:

```
IMPLICIT REAL (P,S,V,T,X,Y,Z) 111
```

- для удвоенной точности:

```
IMPLICIT REAL *8 (P,S,V,T,X,Y,Z) 111
```

Все величины (переменные, массивы и подпрограммы-функции), имена которых начинаются с указанных в списке букв (т.е. переменные PI, SR, VIE, массивы T, X, Y, Z для подпрограммы KINU программы 13.5), относятся к вещественным переменным соответственно обычной (REAL) или удвоенной точности (REAL *8).

Последовательность символов, расположенных в алфавитном порядке, может быть записана в виде диапазона. Диапазон задается первым и последним символами, разделенными знаком минус. Например, запись вида A,B,C идентична записи A–C.

Эту возможность часто используют при описании переменных удвоенной точности в универсальной форме, поместив в любом программном модуле оператор:

```
IMPLICIT REAL *8 (A–H,O–Z) (15.3)
```

Теперь все величины, удовлетворяющие неявному описанию типа, имена которых начинаются с любых букв, кроме I, J, K, L, M, N (зарезервированных для целых чисел), описаны в данном программном модуле величинами удвоенной точности.

Каждый символ, заданный любым из способов, не должен повторяться ни в одном из списков оператора IMPLICIT.

15.3. Требования к программам для работы с использованием удвоенной точности

1. Все величины, использующиеся с удвоенной точностью, должны быть в каждом программном модуле описаны соответствующим образом. Для этой цели может применяться любой из операторов типа (15.1)-(15.3).

Очень удобным для практического использования является применение оператора IMPLICIT в форме (15.3).

Заметим, что его использование в подпрограмме-функции не отменяет обязательности описания имени этой функции в заголовке подпрограммы FUNCTION. Например, описание подпрограммы-функции FX из программы с использованием удвоенной точности может иметь вид:

```

REAL FUNCTION DFX*8 (T)                                110
IMPLICIT REAL *8 (A-H,O-Z)                            111
COMMON PI, SR, VIE                                     115
PI=3.141592653589D0                                    117
SR=16.D0-8.D0*DCOS(3.D0*PI*T)                         118      (15.4)
VIE=0.9D0*T*T-9.D0*T**3                               119
DFX=-SR/2.D0*DSIN(VIE)                                120
RETURN                                                 130
END                                                    131

```

Хотя оператор 111 описывает в удвоенной точности сразу все величины (переменные, массивы и подпрограммы-функции), начинающиеся с букв от А до Н и от О до Z и входящие в данный программный модуль, имя самой функции DFX также описывается в удвоенной точности в заголовке подпрограммы FUNCTION в операторе 110. Отметим, что изменение имени функции с FX на DFX здесь совершенно несущественно и служит только для отличия подпрограмм FX и DFX друг от друга.

2. Системные функции должны использоваться также удвоенной точности. Имена их обычно состоят из их названия (для обычной точности) с добавлением впереди буквы D. Это в данном случае весьма существенно, так как определяет совсем другую системную функцию, название и описание которой мы изменить не можем.

К основным системным функциям удвоенной точности относятся (вещественный аргумент для краткости обозначен через X): DSQRT(X),

DSIN(X), DCOS(X), DTAN(X), DARSIN(X), DARCOS(X), DATAN(X), DEXP(X), DLOG(X), DLOG10(X), DABS(X) (см. п. 7.3).

Например, в операторах 118 и 120 подпрограммы (15.4) используется вычисление синуса и косинуса удвоенной точности.

3. Значения величин, описанных в удвоенной точности, должны задаваться также с удвоенной значностью (в экспоненциальной форме записи, в которой вместо буквы E используется буква D).

15.4. Вычисления с удвоенной точностью

С учетом описанных в п. 15.3 требований, можно достаточно уверенно переделывать программы из обычной точности в удвоенную. Что мы кратко и рассмотрим. Из соображения экономии места будут приводиться только изменяемые или дополняемые в соответствующие программы операторы.

1. Для вычислений с удвоенной точностью с использованием операторов-функций в программу 13.1 следует внести следующие изменения и дополнения:

IMPLICIT REAL *8 (A-H, O-Z)	9
X(T)=4.D0*T	10
Y(T)=16.D0*T*T-1.D0	11
DATA TN, TK, HT, DT/0.D0, 1.D0, 0.05D0, 0.005D0/	20
V=DSQRT (VX*VX+VY*VY)	43
A=DSQRT (AX*AX+AY*AY)	53

Неявный оператор типа IMPLICIT 9 описывает все величины, начинающиеся с букв от A до H и от O до Z удвоенной точности.

Операторы 10 и 11 описывают с использованием постоянных удвоенной точности операторы-функции X(T) и Y(T), задающие определение уравнений движения материальной точки.

В результате соответствия, определенного оператором 20, указанные переменные получают в момент загрузки программы следующие значения удвоенной точности: TN=0.D0, TK=1.D0, HT=0.05D0, DT=0.005D0 сек.

Операторы 43 и 53 определяют с использованием системной функции удвоенной точности DSQRT значения скорости V и ускорения A.

Отметим, что вместо неявного оператора типа IMPLICIT 9 возможно использование нижеследующих операторов явного описания типа в любой из приведенных форм:

DOUBLE PRECISION X, Y, T, TN, TK, HT, DT, VX, VY, V, AX, AY, A	9
REAL *8 X, Y, T, TN, TK, HT, DT, VX, VY, V, AX, AY, A	9

Подпрограммы-функции могут располагаться в одноименном файле в памяти ЭВМ. Чтобы не было путаницы с подпрограммами обычной точности, имена функций в удвоенной точности изменены DFX, DFY и DFZ.

В результате соответствия, устанавливаемого оператором 20 DATA, перечисленные в нем переменные приобретают следующие начальные значения удвоенной точности: TN=0.D0, TK=0.44444D0, NT=0.02222D0, DT=0.00222D0 сек.

Операторы 40N-45N производят обращение к подпрограммам-функциям удвоенной точности DFX, DFY и DFZ, присваивая их вычисленные значения соответствующим переменным.

Операторы 43 и 50-53 производят вычисления абсолютной скорости V, проекций ускорения и его абсолютного значения A.

Метка 20 определяет собой также начало очередного повторения вычислений по программе, если увеличенное на NT значение переменной T все же оказывается меньше конечного времени TK в операторе условного перехода IF 91.

Оператор 110 представляет собой заголовок подпрограммы-функции удвоенной точности с именем DFX, имеющей в качестве списка формальных параметров одну переменную T.

Оператор 111 описывает вещественные переменные PI, SR, VIE и T удвоенной точности. По оператору 115 в неименованной общей области COMMON резервируется три ячейки для размещения переменных PI, SR и VIE из подпрограммы-функции DFX: первая отводится для PI, вторая — для SR, третья — для VIE.

Оператор 117 присваивает постоянное значение удвоенной точности переменной PI. Операторы 118 и 119 описывают вычисления заданных законов относительного SR и переносного VIE движений с использованием формального параметра T, значений постоянных и системной функции DCOS удвоенной точности.

Оператор 120 описывает вычисления функции DFX удвоенной точности, используя значения переменных SR и VIE, постоянной (2.D0) и системной функции DSIN удвоенной точности. Вычисленное значение присваивается одноименной переменной DFX, являющейся выходом подпрограммы.

Операторы 140 и 170 представляют собой заголовки подпрограмм-функций удвоенной точности с именами DFY и DFZ соответственно, имеющих в качестве списка формальных параметров также только одну переменную T.

Операторы 141 и 171 описывают вещественные переменные PI, SR и VIE в удвоенной точности.

По операторам 145 и 175 в той же неименованной общей области резервируются те же три ячейки в той же позиционной последовательности элементов

списка для размещения одноименных переменных PI, SR и VIE из подпрограмм-функций DFY и DFZ соответственно. Это равнозначно тождественности значений одноименных переменных PI, SR и VIE во всех трех подпрограммах.

Хотя переменные SR и VIE в разных подпрограммах и тождественны, но определяются они только в подпрограмме DFX при том значении фактического параметра, с которым происходит обращение к ней. Они остаются такими при любых обращениях к подпрограммам-функциям DFY и DFZ до тех пор, пока при следующем обращении к подпрограмме-функции DFX они не изменят свое значение при следующем значении фактического параметра T. Поэтому нужно организовать основную программу так, чтобы обращение к подпрограммам-функциям DFY и DFZ происходило с тем же значением фактического параметра, с которым последний раз производилось обращение к подпрограмме-функции DFX.

Операторы 150 и 180 описывают вычисления функций DFY и DFZ удвоенной точности, используя значения переменных SR и VIE, постоянных и системной функции DCOS удвоенной точности.

По операторам 130, 160 и 190 осуществляется возврат в вызывающую программу, а операторы 131, 161 и 191 указывают на конец соответствующих подпрограмм-функций DFX, DFY или DFZ.

С использованием оператора DOUBLE PRECISION могут быть представлены операторы 110, 140 и 170:

DOUBLE PRECISION FUNCTION DFX (T)	110
DOUBLE PRECISION FUNCTION DFY (T)	140
DOUBLE PRECISION FUNCTION DFZ (T)	170

и операторы 111, 141 и 171 (которые совмещены в один оператор без номера, так как они одинаковы):

```
DOUBLE PRECISION PI, SR, VIE, T
```

3. Для вычислений с удвоенной точностью с использованием подпрограмм SUBROUTINE в программу 13.5 следует внести следующие изменения и дополнения:

IMPLICIT REAL *8 (A-H, O-Z)	9
CALL DKINU (M, T, X, Y, Z)	43S
V=DSQRT (VX*VX+VY*VY+VZ*VZ)	43
A=DSQRT (AX*AX+AY*AY+AZ*AZ)	53
ANORM=DSQRT (A*A-ATANG*ATANG)	70.1
SUBROUTINE DKINU (M, T, X, Y, Z)	110
IMPLICIT REAL *8 (A-H, O-Z)	111
PI=3.141592653589D0	117
VIE=0.9D0*T (M) *T (M) -9.D0*T (M) **3	118

SR=16.D0-8.D0*DCOS(3.D0*PI*T(M))	119
X(M)=-SR/2.D0*DSIN(VIE)	120
Y(M)=SR/2.D0*DCOS(VIE)	121
Z(M)=SR*DCOS(PI/6.D0)	122

Оператор 43S выполняется в цикле по M и осуществляет обращение к подпрограмме DKINU удвоенной точности с фактическими параметрами, описанными при рассмотрении программы 13.5.

В операторе 43S изменено только название подпрограммы (добавлена буква D), что сделано во избежание путаницы с такой же подпрограммой обычной точности.

Операторы 43, 53 и 70.1 определяют с использованием системной функции удвоенной точности DSQRT значения скорости V, полного A и нормального ANORM ускорений.

Оператор 110 представляет собой заголовок подпрограммы SUBROUTINE удвоенной точности с именем DKINU, имеющей в качестве списка формальных параметров целую переменную M и вещественные массивы T, X, Y и Z. Имя подпрограммы SUBROUTINE не используется для указания ее типа, поэтому заголовок подпрограммы не описывается в удвоенной точности.

Неявный оператор типа IMPLICIT 111 описывает все величины, начинающиеся с букв от A до H и от O до Z, в удвоенной точности.

Операторы 117-119 присваивают в удвоенной точности значения переменной PI и выражений переносного и относительного законов движения материальной точки — переменным VIE и SR.

Операторы 120-122 вычисляют в удвоенной точности выражения, определяющие значения уравнений движения материальной точки (10.16), и присваивают вычисленные значения соответствующим элементам X(M), Y(M) и Z(M).

Программа 13.5 является универсальной для решения задач с применением ЭВМ на темы “Кинематика точки” и “Сложное движение точки”. Рассмотрим, какие изменения и дополнения нужно сделать в подпрограмме SUBROUTINE K1U (из программы 13.4) для работы по программе 13.5 с удвоенной точностью (для типового примера K-1 [21, с. 60]). Приведем всю подпрограмму целиком:

SUBROUTINE DKINU(M, T, X, Y, Z)	110
REAL *8 T(3), X(3), Y(3), Z(3)	111
X(M)=4.D0*T(M)	120
Y(M)=16.D0*T(M)**2-1.D0	121
Z(M)=0.D0	122
RETURN	130
END	131

Оператор 110 представляет собой заголовок подпрограммы SUBROUTINE удвоенной точности с именем DKINU, имеющей в качестве списка формальных параметров целую переменную M и вещественные массивы T, X, Y, Z. Имя используемой подпрограммы SUBROUTINE для универсальной программы 13.5 должно быть одинаковым (для всех заданий и вариантов) и согласованным с именем вызываемой подпрограммы в операторе 43S. Также должны быть согласованы и соответствующие списки формальных и фактических параметров в операторах 110 и 43S.

Оператор REAL *8 111 описывает массивы T, X, Y, Z удвоенной точности с одновременным указанием их размеров.

Операторы 120 и 121 вычисляют с удвоенной точностью арифметические выражения, определяющие значения уравнений движения материальной точки (10.14) и присваивают вычисленные значения соответствующим элементам X(M) и Y(M).

Оператор 122 имеет форму, одинаковую для всех плоских задач с использованием удвоенной точности, в которых координате Z при любом значении T присваивается нулевое значение.

Исходные данные к программе 13.5 в удвоенной точности для типовых примеров заданий K-1 [21, с. 60] и K-7 [21, с. 99] могут быть представлены соответственно в виде (15.5) и (15.6):

```
114418, 1, 31                                     (15.5)
0.D0, 1.D0, 0.05D0, 0.005D0
```

```
114418, 7, 31                                     (15.6)
0.D0, 0.44444D0, 0.02222D0, 0.00222D0
```

4. Для вычислений с удвоенной точностью с использованием стандартных подпрограмм в программу 14.2 следует внести следующие изменения и дополнения:

IMPLICIT REAL *8 (A-H, O-Z)	9
EXTERNAL DFX, DFY, DFZ	10
20 CALL DDCAR (T, DT, 1, DFX, VX (M))	40
CALL DDCAR (T, DT, 1, DFY, VY (M))	41
CALL DDCAR (T, DT, 1, DFZ, VZ (M))	42
CALL DDET3 (DT, VX, AX, NDIM, IERX)	50
CALL DDET3 (DT, VY, AY, NDIM, IERY)	51
CALL DDET3 (DT, VZ, AZ, NDIM, IERZ)	52
V=DSQRT (VX (N) *VX (N) +VY (N) *VY (N) +VZ (N) *VZ (N))	55/43/
A=DSQRT (AX (N) *AX (N) +AY (N) *AY (N) +AZ (N) *AZ (N))	56
ANORM=DSQRT (A *A-ATANG*ATANG)	70.1

Оператор 10 объявляет DFX, DFY и DFZ как имена внешних подпрограмм, которые будут использоваться в качестве фактических параметров при обращении к подпрограмме DDCAR удвоенной точности.

Операторы 40-42 и 50-52 осуществляют обращение к подпрограммам DDCAR и DDET3 с фактическими параметрами, описанными при рассмотрении программы 14.1. Они должны быть представлены с учетом требований для использования удвоенной точности.

Операторы 55/43/, 56 и 70.1 определяют с использованием системной функции удвоенной точности DSQRT значения скорости V, полного A и нормального ANORM ускорений.

Программу 14.2 в удвоенной точности можно использовать с подпрограммами-функциями DFX, DFY и DFZ в любой из рассмотренных выше форм, переделанных в удвоенную точность по образцу (15.4) и дополненных для плоских задач нулевой функцией. Исключение составляет применение общей области COMMON, в которую входят переменные SR и VIE (операторы 110-191 в программе 13.2). В программе 14.2 не предусмотрено необходимое для этого случая согласование обращений к подпрограммам DFX, DFY и DFZ, чтобы оно происходило с одинаковым значением фактического параметра.

Исходные данные к программе 14.2 удвоенной точности для типовых примеров заданий К-1 [21, с. 60] и К-7 [21, с. 99] могут быть представлены соответственно в виде (15.7) и (15.8):

```
114418, 1, 31
0.45D0, 0.55D0, 0.005D0, 21
```

(15.7)

```
114418, 7, 31
0.2D0, 0.24444D0, 0.00222D0, 21
```

(15.8)

Напомним, что соотношение между величинами TN, TK, DT и NDIM определяется равенством (14.1).

Надеемся, что вы проверите все вышеописанные варианты и дополнения, переделав их самостоятельно в удвоенную точность. Для ответа на возможные вопросы, которые могут появиться в процессе этой работы, в приложении 3 приведены все основные программы 13.1–13.5 и 14.1–14.2 в удвоенной точности. Рекомендуем обращаться к ним только после возникновения трудностей, которые вы не сможете преодолеть самостоятельно.

Напомним, что при выполнении всех описываемых в пособии программ на персональных ЭВМ следует учесть в любой удобной для вас форме материал п. 7.6.3.

ЧАСТЬ 6. РЕШЕНИЕ ЗАДАЧ И ВЫПОЛНЕНИЕ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ ПО ДИНАМИКЕ С ИСПОЛЬЗОВАНИЕМ ГОТОВЫХ ПРОГРАММ

ГЛАВА 16. РАБОТА С ИСПОЛЬЗОВАНИЕМ СПЕЦИАЛИЗИРОВАННЫХ ПРОГРАММ DINSP И DDINSP

Решение задач динамики с применением ЭВМ обычно сводится к численному интегрированию дифференциальных уравнений. Поэтому сначала рассмотрим различные аспекты его применения при работе с готовыми программами.

16.1. Краткое описание программ DINSP и DDINSP

Программа DINSP предназначена для решения на ПЭВМ строго определенных заданий из сборника [21] при исследовании относительного (Д-4 [21, с. 148-155]) или колебательного (Д-3 [21, с. 138-148]) движения материальной точки, а также при изучении свободных колебаний механической системы с одной степенью свободы (Д-23 [21, с. 312-320]). В качестве типовых выбраны примеры решения заданий на указанные темы. Ссылка на задания производится по их номеру, причем типовым примерам присвоен 31-й вариант. Все перечисленные задания решаются численно на ПЭВМ и аналитически обычными методами.

Для численного решения на ЭВМ нужно составить дифференциальное уравнение движения материальной точки и определить числовые значения его коэффициентов (см. п. 16.2), правильность которых проверяется программой. Значения этих коэффициентов с указанием их номеров вместе с другими исходными данными, описываемыми в п. 16.3, должны быть представлены в файле `dinsp.dat`. Затем следует сеанс работы на ПЭВМ. Специализированные программы по динамике требуют для своей

работы только подготовки файла исходных данных, как и программы по статике. Поэтому их сеансы работы на ПЭВМ для получения численного решения полностью аналогичны (отличаясь только именем используемой программы) и описаны в п. 4.3.

Одновременно с этим обычными методами механики студентом производится параллельное аналитическое решение указанных заданий при одном заданном времени T_1 (которое рассмотрено в сборниках [21], [22] и в пособии не повторяется).

В различных подпрограммах для каждого варианта указанных заданий хранятся как правильные дифференциальные уравнения, так и правильные аналитические решения. Численное решение студента сравнивается с правильным численным при каждом выводимом на печать значении времени T , а с правильным аналитическим только при заданном времени T_1 . По желанию студента его аналитическое решение также может быть сравнено с правильным при времени T_1 с определением соответствующей погрешности решения.

Программа DDINSP представляет собой программу DINSP с использованием удвоенной точности. Работа с ней полностью аналогична и исходные данные должны находиться в файле ddinsp.dat.

Программы DINSP и DDINSP в зависимости от желания студента реализуют (также как программы IDUSP и DIDUSP в п. 17) различные численные методы (в скобках указываются имена используемых подпрограмм): Рунге-Кутта 3-4-го порядка (RKGS и DRKGS), предиктор-корректор метод Хемминга (HPCG и DHPCG), трехточечный (MAD3) и обобщенный (OMAD) методы Адамса.

Обратите внимание, что программы удвоенной точности (DDINSP и DIDUSP) реализуют только два первых из указанных методов численного интегрирования, поскольку подпрограмм удвоенной точности, использующих трехточечный или обобщенный методы Адамса в пакете научных подпрограмм ПНП-SSP нет (см. [12], [19]). Это является единственным их отличием от соответствующих программ DINSP и IDUSP.

16.2. Представление интегрируемого дифференциального уравнения для ввода в ЭВМ при работе со специализированными программами

Общая форма дифференциального уравнения любого из вариантов заданий Д-3, Д-4 и Д-23 может быть представлена в виде:

$$d(dx) / (dT**2) = C(1) * X + C(2) * dx/dT + C(3) * SIN(C(4) * T) + C(5) + C(6) * T + C(7) * COS(C(8) * T) . \quad (16.1)$$

Каждый частный случай интегрируемого уравнения, возникающего при решении указанных заданий, может быть получен из данного общего, полагая в нем все невыписанные (т.е. не представленные в конкретном уравнении) коэффициенты равными нулю.

Номер ненулевого коэффициента в дифференциальном уравнении определяется по номеру его аналога в уравнении общего вида (16.1): число, стоящее при переменной X , есть $C(1)$; коэффициент при производной dX/dT — $C(2)$; число, стоящее при выражении для SIN от переменного аргумента, есть $C(3)$, а коэффициент при его аргументе T — $C(4)$; свободный член постоянной в правой части дифференциального уравнения есть $C(5)$; значение коэффициента при аргументе T — $C(6)$; число, стоящее при выражении для COS от переменного аргумента, есть $C(7)$, а коэффициент при его аргументе T — $C(8)$.

Для подготовки исходных данных интегрируемое дифференциальное уравнение следует представить в виде с выделенной в левой части второй производной, определив численные значения всех входящих в него коэффициентов. Затем следует определить номера полученных коэффициентов, под которыми обозначены их аналоги $C(I)$ перед соответствующими функциями в уравнении (16.1).

В программах DINSP и DDINSP производится предварительное обнуление всего массива коэффициентов $C(I)$, чем достигается задание нулевых элементов. Поэтому для задания полученного дифференциального уравнения достаточно ввести численные значения его ненулевых коэффициентов с указанием номеров их расположения.

Это осуществляется с помощью оператора ввода, управляемого списком, со встроенным неявным циклом. В нем сначала указывается общее количество ненулевых коэффициентов KNNK, затем следует последовательное попарное перечисление номера ненулевого элемента, его значения, ..., I , $C(I)$, ..., перечисление должно быть повторено KNNK раз:

```
READ *,KNNK,(I,C(I),I1=1,KNNK)
```

Согласно этому оператору вводимые значения ненулевых коэффициентов интегрируемого уравнения располагаются в 3-й строке файла данных следующим образом: указывается количество ненулевых коэффициентов KNNK интегрируемого уравнения, номер первого коэффициента, значение первого коэффициента, ..., I , $C(I)$,..., номер последнего (KNNK-го) коэффициента, значение последнего (KNNK-го) коэффициента. Величина KNNK в зависимости от варианта и задания может иметь значения от 1 до 4.

Для типовых примеров рассматриваемых заданий интегрируемые дифференциальные уравнения после проведения нужных вычислений и выделения в левой части второй производной, могут быть представлены в виде:

- для Д-3 ([21, с. 147]):

$$d(dx) / (dT^{**2}) = -(K^{**2}) * X + H * SIN(P * T) = -300 * X + 6 * SIN(10 * T), \quad (16.2)$$

- для Д-4 (см. в [21, с. 155] уравнение (2)):

$$d(dx) / (dT^{**2}) = C1 * X + C5 = -97.533 * X + 12.491, \quad (16.3)$$

где: $PI=3.14159$, $AL=PI/6$, $G=9.81$, $R=0.2$, $C=1$, $L0=0.2$, $M=0.01$,

$$C1 = -(C/M - PI * PI * (SIN(AL))^{**2}) = -97.533;$$

$$C5 = PI * PI * R * SIN(AL) - G * COS(AL) + C * L0 / M = 12.491.$$

- для Д-23 (см. в [21, с. 319] уравнение (2)):

$$d(dx) / (dT^{**2}) = -744.114 * X \quad (16.4)$$

Определим номера и ненулевые коэффициенты для ввода уравнений (16.2)-(16.4) в 3-й строке соответствующих файлов данных (16.5)-(16.7).

В (16.2) количество ненулевых коэффициентов интегрируемого уравнения $KNNK=3$: номер первого ненулевого коэффициента 1, его значение -300 . (т.е. $C(1)=-300$.), номер второго ненулевого коэффициента 3, его значение 6. (т.е. $C(3)=6$.), номер последнего (третьего) коэффициента 4, его значение равно 10. Поэтому третья строка данных (16.5) примет вид:

3, 1, -300., 3, 6., 4, 10.

В (16.3) количество ненулевых коэффициентов интегрируемого уравнения $KNNK=2$: номер первого ненулевого коэффициента 1, его значение -97.533 (т.е. $C(1)=-97.533$), номер второго (и последнего) ненулевого коэффициента 5, его значение равно 12.491 (т.е. $C(5)=12.491$). В результате в (16.6) будем иметь:

2, 1, -97.533, 5, 12.491

В (16.4) количество ненулевых коэффициентов интегрируемого уравнения $KNNK=1$: номер первого (и последнего) ненулевого коэффициента 1, его значение равно -744.114 (т.е. $C(1)=-744.114$), что определит следующий вид 3-й строки (16.7):

1, 1, -744.114

Определение коэффициентов интегрируемого уравнения является самой ответственной частью при подготовке исходных данных, поэтому на него следует обратить особое внимание.

Отметим, что с аналогичным вводом данных мы уже встречались ранее при выполнении индивидуальных заданий по статике с использованием программ STATN и STAT9N (см. п. 5.3).

16.3. Подготовка данных для работы специализированных программ

16.3.1. Структура вводимых данных для работы программы DINSP

1-я строка: указываются номера группы NG, задания NZ, варианта NW. В рассматриваемых ниже примерах используются: NG=1 14429, NZ=3, NZ=4 и NZ=23, NW=31.

2-я строка: перечисляются начальные значения при времени $T=0$: координаты $X(1)=X_0$ и скорости $X(2)=d(X_0)/dT$ материальной точки (эти данные для заданий Д-4 и Д-23 берутся из условий для соответствующих вариантов, для задания Д-3 определяются по данным условия при постановке задачи).

3-я строка: по описанному выше в п. 16.2 указывается количество ненулевых коэффициентов KNNK интегрируемого уравнения, номер первого ненулевого коэффициента, значение первого коэффициента, ..., I, C(I), ..., номер последнего (KNNK-го) коэффициента, значение последнего (KNNK-го) ненулевого коэффициента.

4-я строка: в зависимости от номера задания состоит из следующих различных элементов:

- для задания Д-3 указывается произвольное время T_1 (находящееся между начальным T_N и конечным T_K временами интервала интегрирования), при котором будет проводиться сравнение вашего аналитического решения с численным на ЭВМ. В программе предварительно задано ($T_1=0.5$).
- для задания Д-4 указывается значение заданного времени T_1 и найденное значение реакции стенки канала N.
- для задания Д-23 (как и для Д-3) указывается произвольное время T_1 (в программе предварительно задано $T_1=0.5$), а также определенные из вашего решения значения циклической частоты ZK, периода TAU и амплитуды A.

5-я строка: перечисляются величины начального PRMT(1)= T_N и конечного PRMT(2)= T_K значений переменной интегрирования T, величины начального шага интегрирования PRMT(3) и верхней границы погрешности вычислений PRMT(4). Все четыре значения PRMT предварительно задаются в программе (PRMT(1)=0., PRMT(2)=1.2, PRMT(3)=0.01, PRMT(4)=0.001).

6-я строка: указываются шаг печати результатов HPR, значение максимальной абсолютной погрешности EPSC задания коэффициентов интегрируемого уравнения C(I), а также величина допустимой относительной ошибки EPSX выполнения расчета. В программе заданы величины HPR=0.1, EPSC=0.2, EPSX=10.

7-я строка: перечисляются значения IPARPP и IPARX (в программе предварительно задано IPARPP=1 и IPARX=1):

- IPARPP — параметр выбора используемой подпрограммы, реализующей соответствующий метод численного интегрирования. Может принимать значения 1, 2, 3 или 4, в зависимости от чего используется подпрограмма RKGS, HPCG, MAD3 или OMAD.
- IPARX — параметр, определяющий необходимость ввода в 8-й строке значений аналитических результатов для координаты и скорости, определенных студентом при времени T1, для сравнения их с соответствующими правильными значениями для данного варианта. При IPARX=2 сравнения производятся и данные в 8-й строке вводятся, а при IPARX=1 — нет.

8-я строка: перечисляются значения аналитических результатов для координаты и скорости, определенных при времени T1.

16.3.2. Представление исходных данных для программы DINSP

Исходные данные в самой простой форме для рассматриваемых типовых примеров (16.2)-(16.4) будут иметь согласно п. 16.3.1 соответственно следующий вид:

```
114429, 3, 31
-0.0245, 0.
3, 1, -300., 3, 6., 4, 10.
/
/
/
/
/
```

(16.5)

```
114429, 4, 31
0.3, 2.0
2, 1, -97.533, 5, 12.491
0.2, 0.106
/
/
/
/
```

(16.6)

```
114429, 23, 31
0.2E-2, 8.E-2
1, 1, -744.114
, 27.27845, 0.2303351, 3.549765E-3
/
/
/
```

(16.7)

Время T1=0.5, предварительно заданное в программе для задания Д-23, нет необходимости изменять. Поэтому в файле (16.7) в 4-й строке в 1-й позиции стоит запятая “,”, определяющая отсутствующее данное.

Разделитель “/” указывает на переход к следующей строке данных. В результате этого предварительно заданные значения вводимых в этой строке величин в памяти ЭВМ не изменяются. Поэтому данные примеров (16.5)-(16.7) эквивалентны нижеследующим (16.8)-(16.10) соответственно, в которых вместо отсутствующих данных указаны предварительно задаваемые в программе значения соответствующих величин (см. п. 16.3.1):

```
114429, 3, 31
-0.0245, 0.
3, 1, -300., 3, 6., 4, 10. (16.8)
```

```
0.5
0., 1.2, 0.01, 0.001
0.1, 0.2, 10.
1, 1
```

```
114429, 4, 31
0.3, 2.0
2, 1, -97.533, 5, 12.491 (16.9)
```

```
0.2, 0.106
0., 1.2, 0.01, 0.001
0.1, 0.2, 10.
1, 1
```

```
114429, 23, 31
0.2E-2, 8.E-2
1, 1, -744.114 (16.10)
```

```
0.5, 27.27845, 0.2303351, 3.549765E-3
0., 1.2, 0.01, 0.001
0.1, 0.2, 10.
1, 1
```

Для изменения значения какой-либо величины в соответствующей ей позиции данной строки следует указать нужное число. При этом возможны оба способа упрощения записи (10.10) и (10.12). Для повторяющихся отсутствующих данных используется коэффициент повторения “J*”, где J — число повторений.

В программе предусмотрена защита от необдуманных действий при изменении данных в 5-й и 6-й строках, могущих повлечь за собой резкое увеличение количества выходной печати или времени счета. Поэтому следующие величины не могут быть заданы меньше: шаг интервала печати HPR — чем 0.01, начальный шаг интегрирования PRMT(3) — чем 0.001, верхняя граница погрешности вычислений PRMT(4) — чем 0.00001, а разность конечного PRMT(2) и начального PRMT(1) значений переменной интегрирования не может быть больше 10. Выход за допустимые границы для указанных величин влечет за собой присваивание им первоначально задаваемых в программе значений (см. п. 16.3.1).

Для работы с удвоенной точностью следует использовать программу DDINSP, которая полностью аналогична описанной выше программе DINSP.

Исходные данные, которые должны быть заданы в файле `ddinsp.dat`, будут полностью аналогичны фрагментам (16.5)-(16.10) для соответствующих типовых примеров рассматриваемых заданий. Все вещественные числа только должны быть указаны в удвоенной точности (что проще всего достигается указанием единичного множителя `D0` после значения вещественного числа).

16.3.3. Сравнение аналитического решения с соответствующим правильным решением

Значение `IPARX`, указываемое вторым в списке в 7-й строке данных, для выполнения сравнения задается равным 2, например:

, 2

Теперь в дополнительной 8-й строке должны быть указаны величины координаты `X1` и скорости `XP1`, определенные при времени `T1` из аналитического решения. Для рассматриваемых типовых примеров (16.2) — (16.4) при значениях `T1` 0.5, 0.2 и 0.5 сек соответственно это будет иметь вид:

$$-2.306074E-2, 59.56459E-2 \quad (16.11)$$

$$0.246, -2.33 \quad (16.12)$$

$$3.531826E-3, -9.722526E-3 \quad (16.13)$$

16.4. Работа на ПЭВМ и представление полученных результатов

Последовательность сеанса работы на ПЭВМ в диалоговом режиме при работе с программами `DINSP` или `DDINSP` совпадает с аналогичным сеансом работы по решению задач статики (см. п. 4.3). Это становится возможным потому, что эти программы требуют для своей работы также только ввода исходных данных, находящихся соответственно в файлах `dinsp.dat` или `ddinsp.dat`. Во всех командах в качестве имени следует указывать соответствующее имя программы `DINSP` или `DDINSP` для работы с удвоенной точностью.

Представление результатов работы специализированных программ `DINSP` и `DDINSP`, отличающихся только используемой точностью вычислений, полностью совпадает. После печати заголовка выводимой информации сами результаты расчета программ `DINSP` и `DDINSP` выводятся в виде таблицы. При каждом выводимом на печать значении времени `T` печатаются расчетные и точные (полученные из правильных дифференциальных уравнений) значения координаты и скорости. По результатам их сравнения указываются абсолютные ошибки расчета.

При заданном времени `T1` указываются относительные погрешности численного на ЭВМ (а при `IPARX=2` и аналитического) определения искомых величин по отношению к их правильному аналитическому решению.

ГЛАВА 17. РАБОТА С ИСПОЛЬЗОВАНИЕМ УНИВЕРСАЛЬНЫХ ПРОГРАММ IDUSP И DIDUSP

Универсальная программа IDUSP предназначена для численного интегрирования на ЭВМ произвольных дифференциальных уравнений второго порядка, получающихся в различных задачах динамики. Они должны быть представлены в виде системы дифференциальных уравнений (СДУ) 1-го порядка (см. п. 18.3.1). Программа DIDUSP представляет собой программу IDUSP с использованием удвоенной точности.

Для работы программы IDUSP (или DIDUSP) студентом, кроме исходных данных в файле idusp.dat (или didusp.dat), задается интегрируемое дифференциальное уравнение в виде СДУ 1-го порядка в файле sdu1p.for (или dsdu1p.for) (см. соответственно пп. 17.1-17.3 или 17.4).

Универсальные программы, как и специализированные п. 16, реализуют те же различные численные методы (Рунге-Кутта, Хемминга, трехточечный и обобщенный методы Адамса). Они также допускают возможность сравнений численного и аналитических решений. В этом случае результаты аналитического решения представляются в виде соответствующих функций от времени для координаты и скорости в подпрограмме DINU (или DDINU).

17.1. Подготовка интегрируемого дифференциального уравнения для ввода в ЭВМ при работе с универсальными программами

В теоретической механике дифференциальные уравнения обычно описывают движение материальной точки или системы и являются дифференциальными уравнениями второго порядка (ДУ2П). Для численного интегрирования любого дифференциального уравнения (ДУ) его нужно представить в виде системы дифференциальных уравнений первого порядка (СДУ1П), что подробно описывается в п. 18.3.1 (см. замену произвольного ДУ2П (18.3) СДУ1П (18.4)-(18.5)). В формализованном виде соответствующий элемент фортран-программы в SDUIP имеет структуру (18.6)-(18.7).

Для рассмотрения в качестве типового выбран пример решения задания Д-27, которое рекомендовано в [21, с. 358-361] для выполнения на ЭВМ. Его дифференциальное уравнение записано там же под номером (12) и имеет вид:

$$12.6 * d(dx) / (dt**2) = -(78+1944*x*x) * x / ABS(x) - 684*x - 3888*x**3 \quad (17.1)$$

Уравнение (17.1) можно также представить с использованием вспомогательных переменных:

$$d(dX) / (dT^{**2}) = -(B+C*X*X) * X / ABS(X) - D*X - E*X**3, \quad (17.2)$$

где A=12.6, B=78./A, C=1944./A, D=684./A, E=3888./A.

Рассматривается также ДУ (16.3) типового примера задания Д-4, интегрирование которого описывалось в п. 16. Это сделано для обеспечения более плавного перехода от специализированных к использованию универсальных программ.

17.2. Представление системы двух дифференциальных уравнений первого порядка (С2ДУ1П) в виде подпрограммы SDU1P

Заголовок подпрограммы SUBROUTINE SDU1P(T,X,XP) определяет ее имя и задает список используемых формальных параметров (в который входит время T и массивы X и XP), служащих для информационной связи с основной программой IDUSP.

Следующий вторым оператор DIMENSION X(2),XP(2) задает описание используемых массивов (см. п. 18.3.1) переменной X и ее первой производной (X(1)=X, X(2)=dX/dT) и производных XP (XP(1)=dX/dT, XP(2)=d(dX)/dT**2), содержащих по два элемента.

Далее следует задание интегрируемой системы уравнений 1-го порядка согласно структуре (18.6)-(18.7), что для примера (16.3) будет иметь вид:

$$\begin{aligned} XP(1) &= X(2) \\ XP(2) &= -97.533 * X(1) + 12.491 \end{aligned} \quad (17.3)$$

Если при этом используются вспомогательные переменные, то предварительно им должны быть заданы определенные значения (см. соответствующие операторы в (17.5)).

Заканчивается каждая подпрограмма операторами RETURN и END.

С учетом вышеописанной структуры подпрограммы SDU1P могут иметь вид:

- для примера (16.3) (задание Д-4):

```
SUBROUTINE SDU1P(T,X,XP)
  DIMENSION X(2),XP(2)
  XP(1)=X(2)
  XP(2)=-97.533*X(1)+12.491
  RETURN
END
```

(17.4)

- для примера (17.2) (задание Д-27):

```
SUBROUTINE SDU1P(T,X,XP)
  DIMENSION X(2),XP(2)
```

C *** ЗАДАНИЕ ВСПОМОГАТЕЛЬНЫХ ПЕРЕМЕННЫХ:

```

A=12.6
B=78./A
C=1944./A
D=684./A
E=3888./A

```

(17.5)

C ***** ЗАДАНИЕ ИНТЕГРИРУЕМОЙ СИСТЕМЫ УРАВНЕНИЙ:

```

XP(1)=X(2)
XP(2)=- (B+C*X(1)*X(1))*X(1)/ABS(X(1))-D*X(1)-E*X(1)**3
RETURN
END

```

Для численного интегрирования типовых примеров заданий Д-3 (16.2) и Д-23 (16.4) следует изменить во фрагменте (17.4) только правую часть оператора XP(2), задающего интегрируемое уравнение (см. дополнение 19.6).

Описание защиты от возможной ошибки переполнения порядка в операторе, задающем значение XP(2), находится в дополнении 19.8, а представление интегрируемого уравнения (17.2) без использования вспомогательных переменных — в дополнении 19.9.

17.3. Подготовка данных для работы универсальных программ

17.3.1. Структура вводимых данных для работы программы IDUSP

1-я строка: указываются номера группы NG, задания NZ, варианта NW. В рассматриваемых ниже примерах используются: NG=114410, NZ=4 и NZ=27, NW=31.

2-я строка: перечисляются начальные значения при времени $T=0$ координаты $X(1)=X_0$ и скорости $X(2)=d(X_0)/dT$ материальной точки (определяются по данным условия). В типовом примере задания Д-4 [21, с. 149] $X_0=0.3$ м, $X_0'=2$ м/с, а в Д-27 [21, с. 358] $X_0=0.5$ рад, $X_0'=0$.

3-я строка: перечисляются величины начального PRMT(1)=TN и конечного PRMT(2)=TK значений переменной интегрирования T, величины начального шага интегрирования PRMT(3) и верхней границы погрешностей вычислений PRMT(4). Значения величин в строках 3-4 предварительно задаются в программе применительно к выполнению типового примера задания Д-27: PRMT(1)=0., PRMT(2)=0.341, PRMT(3)=0.001, PRMT(4)=0.0001;

4-я строка: указываются значение шага интервала печати HPR для времени T и величина допустимой относительной ошибки EPSX выполнения расчета. В программе задано: HPR=0.011, EPSX=10.

5-я строка: перечисляются задаваемые значения IPARPP, IPARX1 и IPARXA (предварительно задано IPARPP=1, IPARX1=1 и IPARXA=1):

- IPARPP — параметр выбора используемой подпрограммы, реализующей соответствующий метод численного интегрирования. Может принимать значения 1, 2, 3 или 4, в зависимости от чего используется подпрограмма RKGS, HPCG, MAD3 или OMAD;
- IPARX1 — параметр, определяющий необходимость ввода в 6-й строке значений для координаты X1 и скорости XP1, определенных студентом при времени T1 из его аналитического решения, для их сравнения с результатами численного интегрирования на ЭВМ. При IPARX1=2 сравнения производятся и данные в 6-й строке вводятся, а при IPARX1=1 — нет;
- IPARXA — параметр, определяющий необходимость задания в подпрограмме DINU аналитического решения для координаты XA и скорости XPA для сравнения его с соответствующими значениями численного интегрирования на ЭВМ при каждом выводимом на печать времени T. При IPARXA=2 — сравнения производятся и подпрограмма DINU задается, а при IPARXA=1 — нет.

6-я строка: ввод при IPARX1=2 времени T1, значений аналитических результатов при T1 для координаты X1 и скорости XP1 (в последовательности T1, X1, XP1) для сравнения их с соответствующими численными результатами. Если значение IPARXA=2 и IPARX1=1, то в 6-й строке задается только значение времени T1, при котором программа определяет относительную ошибку численного и аналитического решений. Если IPARX1=1 и IPARXA=1, то 6-я строка исходных данных может отсутствовать.

17.3.2. Представление исходных данных для программы IDUSP

Исходные данные в самой простой форме для рассматриваемых примеров (16.3) и (17.2) согласно п. 17.3.1 будут иметь соответственно вид (17.6)-(17.7):

```
114410, 4, 31
0.3, 2.0
, 1.2/
0.1/
/
/
/
/
/
```

(17.6)

```
114410, 27, 31
0.5, 0.
/
/
/
/
/
```

(17.7)

Здесь во фрагменте (17.6) использованы оба способа упрощения записи (10.10) и (10.12) для предварительно задаваемых в программе величин. Значения конечного времени интегрирования ТК (1.2) и шага интервала

печати HPR (0.1) заданы совпадающими с фрагментами (16.6) и (16.9) для возможности сравнения результатов работы специализированных и универсальных программ.

Данные примеров (17.6)-(17.7) эквивалентны нижеследующим (17.8)-(17.9) соответственно, в которых вместо отсутствующих данных указаны предварительно задаваемые в программе значения соответствующих величин (см. п. 17.3.1):

```
114410, 4, 31
0.3, 2.0
0., 1.2, 0.001, 0.0001 (17.8)
0.1, 10.
1, 1, 1
```

```
114410, 27, 31
0.5, 0.
0., 0.341, 0.001, 0.0001 (17.9)
0.011, 10.
1, 1, 1
```

17.3.3. Сравнение результатов, полученных из аналитического решения при времени T1, с соответствующим численным на ЭВМ

Если значение IPARX1, указываемое вторым в списке в 5-й строке данных, задается равным 2, например:

, 2/

то производится сравнение соответствующего численного расчета на ЭВМ и аналитических результатов, определенных при времени T1 для координаты X1 и скорости XP1. Для этого они должны быть введены в таком же порядке в дополнительной 6-й строке данных, которая для примера (16.3) задания Д-4 примет вид:

```
0.2, 0.246, -2.33 (17.10)
```

17.3.4. Сравнение аналитических зависимостей, полученных для координаты XА и скорости XРА, с численным решением на ЭВМ

В этом случае значение IPARXA, указываемое последним (третьим) в списке в 5-й строке данных, задается равным 2, например:

, , 2

Сравнение производится при каждом выводимом на печать значении времени T. Выражения для координаты XА и скорости XРА должны быть

представлены в виде соответствующих функций от времени T в файле `dinu.for` в виде подпрограммы `DINU`, которая для примера (16.3) задания Д-4 может иметь следующий вид:

```

SUBROUTINE DINU (T, XA, XPA)
  ZK=9.876
  C1=0.172
  C2=0.202
  B=0.128
  ZKT=ZK*T
  XA=C1*COS (ZKT)+C2*SIN (ZKT)+B
  XPA=-C1*ZK*SIN (ZKT)+C2*ZK*COS (ZKT)
  RETURN
END

```

(17.11)

17.4. Работа с удвоенной точностью с использованием программы `DIDUSP`

Работа с программами `DIDUSP` и `IDUSP` полностью аналогична. Интегрируемые уравнения и аналитические решения соответственно в файлах `dsdu1p.for` и `ddinu.for`, а также исходные данные в файле `didusp.dat` должны быть представлены с учетом требований для использования удвоенной точности (см. п. 15.3):

- для примера (16.3):

```

SUBROUTINE DSDU1P (T, X, XP)
  IMPLICIT REAL *8 (A-H, O-Z), INTEGER *4 (I-N)
  DIMENSION X (2), XP (2)
  XP (1)=X (2)
  XP (2)=-97.533D0*X (1)+12.491D0
  RETURN
END

```

(17.12)

```

SUBROUTINE DDINU (T, XA, XPA)
  IMPLICIT REAL *8 (A-H, O-Z), INTEGER *4 (I-N)
  ZK=9.876D0
  C1=0.172D0
  C2=0.202D0
  B=0.128D0
  ZKT=ZK*T
  XA=C1*DCOS (ZKT)+C2*DSIN (ZKT)+B
  XPA=-C1*ZK*DSIN (ZKT)+C2*ZK*DCOS (ZKT)
  RETURN
END

```

(17.13)


```

114410,4,31
0.3D0,2.0D0
,1.D0/
0.1D0/
/

```

(17.14)

- для примера (17.2) (с учетом дополнения 19.8):

```

SUBROUTINE DSDU1P(T,X,XP)
IMPLICIT REAL *8 (A-H,O-Z), INTEGER *4 (I-N)
DIMENSION X(2),XP(2)

```

C *** ЗАДАНИЕ ВСПОМОГАТЕЛЬНЫХ ПЕРЕМЕННЫХ:

```

A=12.6D0
B=78.D0/A
C=1944.D0/A
D=684.D0/A
E=3888.D0/A

```

C ***** ЗАДАНИЕ ИНТЕГРИРУЕМОЙ СИСТЕМЫ УРАВНЕНИЙ:

```

XP(1)=X(2)
IF(DABS(X(1)).LE.1.D-33) GOTO 5
XP(2)=- (B+C*X(1)*X(1))*X(1)/DABS(X(1))-D*X(1)-E*X(1)**3
GOTO 10
5 XP(2)=-D*X(1)-E*X(1)**3
10 RETURN
END

```

(17.15)

```

114410,27,31
0.5D0,0.D0
/
/
/

```

(17.16)

Представление исходных данных в полной форме, соответствующее примерам (17.14) и (17.16) для работы с удвоенной точностью соответственно примет вид:

```

114410,4,31
0.3D0,2.0D0
0.D0,1.D0,0.001D0,0.0001D0
0.1D0,10.D0
1,1,1

```

(17.17)

```

114410,27,31
0.5D0,0.D0
0.D0,0.341D0,0.001D0,0.0001D0
0.011D0,10.D0
1,1,1

```

(17.18)

Для сравнения результатов, полученных из аналитического решения при задаваемом времени $T1$, с соответствующим численным решением на ЭВМ, 5-я и 6-я строка данных фрагмента (17.14) файла `didusp.dat` соответственно примет вид:

$$\begin{aligned} & , 2/ \\ & 0.2D0, 0.246D0, -2.33D0 \end{aligned} \quad (17.19)$$

17.5. Работа на ПЭВМ, представление и анализ полученных результатов

Последовательность сеанса работы на ЭВМ в диалоговом режиме при работе с универсальными программами совпадает с аналогичным сеансом работы по решению заданий по кинематике (см. п. 10.6). При использовании программ `IDUSP` и `DIDUSP` при `IPARXA=1` во всех командах вместо имен программы `KINMP` и подпрограммы `KINU` следует указывать соответствующие имена программы `IDUSP` (или `DIDUSP`) и подпрограммы `SDU1P` (или `DSDU1P` для работы с удвоенной точностью).

При `IPARXA=2` небольшое отличие будет касаться только того факта, что при дополнительном в этом случае задании подпрограммы `DINU` (или `DDINU`), последовательность действий по редактированию и трансляции подпрограммы (описанная в п. 10.6 на примере подпрограммы `KINU`) должна быть повторена дважды до достижения нормального ее завершения: для подпрограммы `SDU1P` (или `DSDU1P`) и для `DINU` (или `DDINU` соответственно при использовании удвоенной точности).

Одновременно с этим следует отметить, что если задано `IPARXA=1` и не предполагается сравнения с аналитическим решением, все равно в текущем подкаталоге должен находиться оттранслированный файл с именем `dinu.obj` (или `ddinu.obj`) и произвольным содержанием. Для этой цели можно использовать имеющийся тестовый файл `dinu.obj` рассматриваемых типовых примеров или скопировать содержимое файла `sdu1p.obj` с именем `dinu.obj`. Его имя также должно обязательно указываться наряду с именем `sdu1p` в параметрах команды для сборки выполняемого модуля программы `idusp.exe`: `idusp.bat sdu1p dinu` (или `didusp.bat dsdu1p ddinu` при работе с удвоенной точностью).

Представление результатов работы универсальных программ `IDUSP` и `DIDUSP`, отличающихся только используемой точностью вычислений, полностью совпадает. Их структура и аварийные сообщения также почти полностью совпадают с описанным в п. 16.4 представлением результатов работы программы `DINSP`. Отметим их небольшие отличия.

В заголовке выводимой информации не указывается конкретный вид интегрируемого уравнения вследствие его произвольности, а только указываются значения начальных условий (X_0 и X_0'), а также метод численного интегрирования и наименование реализующей этот метод подпрограммы.

При IPARXA=2 заголовок дополняется сообщением о проведении сравнения численного решения для X и XP , получаемых из заданного студентом дифференциального уравнения, с его аналитическим решением, обозначаемым “точное X_A ” и “точное X_{PA} ”. Хотя сами результаты расчета универсальных программ также выводятся в виде таблицы, однако форма ее различна в зависимости от задаваемых значений IPARXA.

В программе предусмотрена защита от необдуманных действий при варьировании данных в 3-5-й строках, могущих повлечь за собой резкое увеличение количества выходной печати или времени счета. Выход за допустимые границы для этих величин влечет за собой присваивание им первоначально задаваемых в программе значений (см. п. 17.3.1).

Обращаем ваше внимание, что сравнение результатов численного и аналитического решений производится при произвольно задаваемом времени T_1 , при котором значения координаты X или скорости XP в данной задаче могут иметь малые абсолютные значения. Так как результаты сравнения при времени T_1 представляются в относительной форме, то в этом случае малая абсолютная ошибка расчета может дать довольно большую относительную погрешность.

Например, для типового примера решения задания Д-23 из [21] при времени $T_1=0.5$ относительная погрешность аналитического определения скорости по отношению к численному решению на ЭВМ около 100 %. В этом случае следует более точно выполнить расчет, определяя коэффициенты и постоянные интегрирования, чем это сделано в рассматриваемом типовом примере в [21, с. 319-320]. При более точном расчете относительная погрешность аналитического определения скорости по отношению к численному решению на ЭВМ составит уже около 1 %.

ЧАСТЬ 7. СОСТАВЛЕНИЕ ПРОГРАММ НА ФОРТРАНЕ ДЛЯ РЕШЕНИЯ ЗАДАЧ И ВЫПОЛНЕНИЯ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ ПО ДИНАМИКЕ

ГЛАВА 18. НЕОБХОДИМЫЕ СВЕДЕНИЯ ДЛЯ ЧИСЛЕННОГО ИНТЕГРИРОВАНИЯ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

Рассмотрим различные аспекты практического применения численного интегрирования дифференциальных уравнений, необходимые для самостоятельного составления программ.

18.1. Постановка задачи

Дифференциальное уравнение первого порядка с одним начальным условием можно записать в виде:

$$dX/dT = X' = F(T, X), \quad (18.1)$$

$$X(T_0) = X_0. \quad (18.2)$$

Уравнение (18.1) можно рассматривать как определение кривой через ее производную в координатной плоскости OTX . Мы знаем, как вычислить производную в каждой точке этой кривой через координаты этой точки T и X . В общем случае уравнению (18.1) удовлетворяет целое семейство кривых. Начальное условие (18.2) позволяет выбрать из этого семейства одну определенную кривую, которая проходит через заданную точку.

Численное решение дифференциального уравнения в принципе находится следующим образом. Дифференциальное уравнение задает наклон кривой в любой точке как функцию от T и от X . В начальный момент мы знаем только одну точку, через которую проходит кривая, а именно с координатами T_0, X_0 . Поэтому мы начинаем с этой точки, вычисляем наклон кривой при $T=T_0$ и $X=X_0$ по формуле $X_0' = F(T_0, X_0)$ и продвигаемся на некоторое малое расстояние вдоль получившейся касательной.

Если шаг по T обозначить через H , то мы приходим в точку $T_1=T_0+H$, в которой приближение X_1 к точному значению $X(T_1)$ можно найти, например, беря два первых члена ряда Тейлора: $X_1=X_0+H \cdot F(T_0, X_0)$. Полагаем затем $T_2=T_1+H$ и находим $X_2=X_1+H \cdot F(T_1, X_1)$. Продолжая эту процедуру дальше, мы получаем последовательность коротких отрезков прямой, которые, как мы надеемся, являются достаточно хорошим приближением к искомой функции: $X(N+1)=X_N+H \cdot F(T_N, X_N)$.

Естественно, в этом простейшем методе численного решения дифференциальных уравнений, который носит название метода Эйлера, имеется множество недостатков. Они вытекают из того основного факта, что мы пытаемся приблизить кривую отрезками прямой, а это с самого начала будет приводить к затруднениям. Достаточно часто будет возникать такая ситуация, когда последовательность отрезков прямых существенно отклонится от искомой кривой. Эта проблема называется проблемой устойчивости метода. Метод Эйлера, конечно, на практике не используется, но он дает общее представление и описывает основной алгоритм действий, производимых при численном интегрировании дифференциальных уравнений.

Для учета кривизны кривой искомого решения существует огромное количество усложненных в различной степени методов, которые можно разбить на 2 широких класса: одноступенчатые и многоступенчатые (или многошаговые) методы. Ознакомимся в общих чертах с некоторыми применениями их, имеющими достаточно хорошую фортранную реализацию, обращая особое внимание не столько на общее описание, сколько на их достоинства и недостатки, которые нужно знать при выборе метода для практического использования.

18.2. Методы численного интегрирования дифференциальных уравнений

18.2.1. Одноступенчатые методы

В одноступенчатых методах, называемых иногда одношаговыми, используется только информация о самой кривой в одной точке и не производится итерации, то есть дальнейшие приближения к значению искомой функции. Одним из таких методов является представление приближений функции с помощью рядов Тейлора, простейшим частным случаем которого является рассмотренный выше метод Эйлера. Методы рядов Тейлора обычно довольно неудобны для практического использования, поэтому их применение слишком ограничено.

К практически удобным методам этого класса относятся методы Рунге-Кутты, обладающие следующими отличительными свойствами:

1) Эти методы являются одноступенчатыми: чтобы найти $X(N+1)$, нужна информация только о предыдущей точке T_N, X_N .

2) Они согласуются с рядом Тейлора вплоть до членов порядка $H * P$, где степень P различна для различных методов и называется порядком метода.

3) Они не требуют определения производных $F'(T, X)$, а используют только вычисления самой функции, поэтому они более удобны для практических расчетов, нежели методы рядов Тейлора.

Классический метод Рунге-Кутты четвертого порядка может рассматриваться как обобщение на дифференциальные уравнения квадратичной формулы Симпсона для численного интегрирования функций и выражается формулой, приводимой здесь только для общего представления:

$$X(N+1) = X_N + (K_0 + 2 * K_1 + 2 * K_2 + K_3) \cdot H,$$

где $K_0 = H * F(T_N, X_N)$, $K_1 = H * F(T_N + H/2, X_N + K_0/2)$, $K_2 = H * F(T_N + H/2, X_N + K_1/2)$, $K_3 = H * F(T_N + H, X_N + K_2)$. Заметим, что на каждом шаге требуется четыре вычисления функции $F(X, T)$. Это та цена, которую приходится платить за право не определять никаких производных, но цена более чем умеренная.

Описание подпрограммы RKGS, реализующей на Фортране этот метод Рунге-Кутты, подробные указания и примеры работы с ней приведены в разделах 18.4.1 и 19.1.

Методы Рунге-Кутты имеют несколько достоинств: — их легко программировать; — они численно устойчивы для широкого класса задач; — для получения решения $X(N+1)$ в следующей точке требуется знать значение решения X_N только в одной предшествовавшей точке, поэтому метод

является самостартующим; — величину шага H можно изменить на любом этапе вычислений.

Кроме неоднократного вычисления функции $F(X, T)$, эти методы имеют тот недостаток, что делают трудоемкой оценку допустимой ошибки, помогающей в выборе величины шага. Контроль точности и выбор величины шага интегрирования H подпрограммой RKGS производится путем сравнения результатов счета в одной и той же точке с шагами H и $2H$, так как ни ошибка метода, ни ее оценка не получается в процессе вычислений.

Отметим, что последний недостаток может быть почти полностью устранен: в п. 22 описывается современная модификация Фельберга метода Рунге-Кутты и реализующая ее подпрограмма RK45, которая включает контроль длины шага.

18.2.2. Многоступенчатые или многошаговые методы

В методах, рассматривавшихся до сих пор, значение $X(N+1)$ вычисляется посредством функции, которая зависит лишь от T_N, X_N и длины шага H . Логично предположить, что можно увеличить точность расчета, используя информацию в предшествовавших точках, а именно $X(N-1)$, $X(N-2)$, ..., X_1 и вычисленные в них ранее соответствующие значения функции $F(T_i, X_i)$. Многошаговые методы, основанные на этой идее, весьма эффективны, особенно если требуется высокая точность.

Большинство методов этого класса называются методами прогноза и коррекции (или методы предиктор-корректор). Они состоят из различных шагов для вычисления значения функции в следующей точке: — шаг-предсказание 1 (или прогноз), который мы обозначим через P , когда вычисляется начальное приближение к $X(N+1)$; — шаг-вычисление 2, назовем его E ; — шаг-коррекция 3, обозначаемый C ; — шаг 4 повторного вычисления лучшего приближения, обозначаемый также через E . Таким образом, получается схема предиктор-корректор, наиболее часто используемая для так называемых методов Адамса.

Возможны различные сочетания шагов предиктор-корректор. Например, шаг 4 может быть заменен требованием выполнения ровно M итераций корректора, что обозначают $P(EC)^*M$. Часто его завершают заключительным вычислением функции, дающим более точное значение для следующего шага, что в принятых обозначениях примет вид $P(EC)^*ME$. Эти различные сочетания шагов предиктор-корректор определяют многообразие используемых модификаций многошаговых методов и их названий, а также их реализаций на Фортране.

Отметим, что хотя и имеются некоторые трудности, связанные с использованием итерационной процедуры (ЕС) и с получением нескольких начальных точек решения, но они уравниваются тем фактом, что оценку ошибки при использовании этого метода легко получить в качестве побочного продукта вычислений, не изменяя значения шага H . Поэтому порядок метода, зависящий от величины локальной ошибки на данном шаге, может выбираться автоматически и динамически изменяться. Тем самым получаются методы, работающие для очень широкого круга задач.

Эти достоинства приобретаются ценой усложнения подпрограммы для численного интегрирования. Это ни в коей мере не должно пугать пользователя ЭВМ, поскольку в пакете научных подпрограмм ПНП-SSP [12] имеется несколько хороших фортранных реализаций многошаговых методов. Из них мы выберем только те, практическое использование которых почти совсем не потребует новых знаний. Для этого их список формальных параметров и требуемые для работы внешние подпрограммы должны почти полностью совпадать с описываемой ниже подпрограммой RKGS. К ним относятся: HPCG — использующая модифицированный предиктор-корректор метод Хемминга; MAD3 — реализующая трехточечный метод Адамса; OMAD — использующая обобщенный метод Адамса 4-го порядка с переменным шагом интегрирования.

Освоение этих подпрограмм значительно расширяет возможности пользователя, так как во многих случаях эти два класса одно- и многошаговых методов приходится сочетать разумным образом, учитывая их достоинства и недостатки, что бывает особенно незаменимо для проверки результатов интегрирования при выполнении расчета двумя принципиально разными методами.

Все рассмотренные выше методы легко обобщаются на системы уравнений первого порядка, для интегрирования которых и предназначены упомянутые выше подпрограммы. Отметим, что все уравнения высших порядков можно свести к системе уравнений первого порядка, чему посвящен следующий раздел.

В теоретической механике дифференциальные уравнения обычно описывают движение материальной точки или системы и являются уравнениями второго порядка. В зависимости от характера задачи: линейная, плоская или пространственная — интегрируемая система может состоять соответственно из одного, двух или трех дифференциальных уравнений второго порядка. Рассмотрим возможные частные случаи, после изучения которых интегрирование систем с числом уравнений более трех уже может проводиться по аналогии.

18.3. Представление дифференциальных уравнений высших порядков в виде системы дифференциальных уравнений первого порядка

18.3.1. Замена дифференциального уравнения второго порядка системой двух дифференциальных уравнений первого порядка

При интегрировании одного дифференциального уравнения его нужно представить в виде, в котором высшая производная (вторая в данном случае) выделена и находится в левой части дифференциального уравнения:

$$d(dX)/dT^*2 = F(T, X, dX/dT) . \quad (18.3)$$

Введем обозначение: $dX/dT = XP$, тогда $d(dX)/dT^*2 = d(XP)/dT$ и уравнение (18.3) можно переписать в виде системы двух дифференциальных уравнений первого порядка:

$$dX/dT = XP, \quad (18.4)$$

$$d(XP)/dT = F(T, X, XP) . \quad (18.5)$$

Таким образом, мы выполнили поставленную задачу и представили дифференциальное уравнение второго порядка (18.3) в виде системы из двух дифференциальных уравнений первого порядка (18.4)-(18.5).

Теперь, чтобы задать систему дифференциальных уравнений первого порядка, запись уравнений (18.4) и (18.5) нужно представить в формализованном виде. С этой целью вводятся две матрицы-столбца, размер которых равен количеству дифференциальных уравнений первого порядка KDU (в данном случае $KDU=2$):

$X(KDU)$ — массив значений переменной X и ее первой производной: $X(1)=X$, $X(2)=dX/dT$;

$XP(KDU)$ — массив производных XP , в котором $XP(1)=dX/dT$, $XP(2)=d(dX)/dT^*2$.

Такая формализация выполняется для задания системы (18.4)-(18.5) в подпрограмме $SDU1P$, соответствующий элемент фортран-программы которой имеет следующую структуру:

$$XP(1) = X(2) \quad (18.6)$$

$$XP(2) = F(T, X(1), X(2)) \quad (18.7)$$

При задании системы дифференциальных уравнений 1-го порядка в подпрограмме $SDU1P$ нужно определить массив производных $XP(KDU)$ через принятые обозначения для переменной $X(1)$ и ее производной по времени $X(2)$.

Отметим, что оператор (18.6) будет одинаков для всех линейных (одномерных) задач, ибо он приравнивает обозначение для производной $X(2)$ первому элементу массива производных $XP(1)$.

Оператор (18.7) присваивает второму элементу массива производных значение правой части уравнения (18.3), которое нужно представить с помощью правил записи арифметических выражений на алгоритмическом языке Фортран с учетом принятых обозначений $X=X(1)$, $dX/dT=X(2)$.

Таким образом, из каждого дифференциального уравнения N-го порядка получится в общем случае система из N дифференциальных уравнений 1-го порядка.

18.3.2. Замена системы двух дифференциальных уравнений второго порядка (С2ДУ2П) системой четырех дифференциальных уравнений первого порядка (С4ДУ1П)

Оба интегрируемых уравнения следует представить в виде с выделенной в левой части второй производной, что в общем случае можно записать так:

$$d(dX)/dT^{**2}=F1(T, X, Y, dX/dT, dY/dT), \quad (18.8)$$

$$d(dY)/dT^{**2}=F2(T, X, Y, dX/dT, dY/dT). \quad (18.9)$$

Вводим обозначения: $dX/dT=XP$ и $dY/dT=YP$, с использованием которых левые части уравнений (18.8)-(18.9) примут вид: $d(dX)/dT^{**2}=d(XP)/dT$, $d(dY)/dT^{**2}=d(YP)/dT$.

Теперь уравнения (18.8) и (18.9) можно переписать в виде системы из четырех дифференциальных уравнений 1-го порядка:

$$dX/dT=XP, \quad (18.10)$$

$$dY/dT=YP, \quad (18.11)$$

$$d(XP)/dT=F1(T, X, Y, XP, YP), \quad (18.12)$$

$$d(YP)/dT=F2(T, X, Y, XP, YP). \quad (18.13)$$

Обратим внимание, что последовательность, то есть алгоритм производимых действий при замене двух дифференциальных уравнений полностью совпадает с рассмотренным выше случаем замены одного уравнения. Он сохраняется и для случая трех (см. п. 18.3.3) и более уравнений.

Также вводятся две матрицы-столбца, размер которых равен количеству дифференциальных уравнений первого порядка KDU (в данном случае KDU=4):

$X(KDU)$ — массив значений переменных X, Y и их первых производных: $X(1)=X$, $X(2)=Y$, $X(3)=dX/dT$, $X(4)=dY/dT$;

$XP(KDU)$ — массив производных XP и YP, в котором $XP(1)=dX/dT$, $XP(2)=dY/dT$, $XP(3)=d(dX)/dT^{**2}$, $XP(4)=d(dY)/dT^{**2}$.

В качестве буквы для обозначения этих массивов может быть выбрана любая буква или их сочетание. Мы выбрали X, чтобы сохранить преемственность изложения.

Теперь нам также нужно определить массив производных $XP(KDU)$ через принятые обозначения для переменных $X=X(1)$, $Y=X(2)$ и их производных по времени $dX/dT=X(3)$, $dY/dT=X(4)$.

Элемент фортран-программы, задающий систему дифференциальных уравнений первого порядка (18.10)-(18.13) в подпрограмме `SDU1P`, будет иметь следующую структуру:

$$XP(1) = X(3) \quad (18.14)$$

$$XP(2) = X(4) \quad (18.15)$$

$$XP(3) = F1(T, X(1), X(2), X(3), X(4)) \quad (18.16)$$

$$XP(4) = F2(T, X(1), X(2), X(3), X(4)) \quad (18.17)$$

Операторы (18.14) и (18.15) будут также одинаковыми для всех плоских (двумерных) задач, ибо они просто присваивают введенные обозначения для производных $dX/dT=X(3)$ и $dY/dT=X(4)$ соответственно первому и второму элементу массива производных $XP(1)$ и $XP(2)$.

Операторы (18.16) и (18.17) присваивают третьему $XP(3)$ и четвертому $XP(4)$ элементам массива производных соответствующие выражения для правых частей интегрируемых уравнений (18.8) и (18.9), запись которых следует произвести на Фортране с учетом принятых обозначений $X=X(1)$, $Y=X(2)$, $dX/dT=X(3)$, $dY/dT=X(4)$.

18.3.3. Замена системы трех дифференциальных уравнений второго порядка (СЗДУ2П) системой шести дифференциальных уравнений первого порядка (С6ДУ1П)

Представляем интегрируемые уравнения в виде с выделенной в левой части второй производной:

$$d(dX)/dT**2 = F1(T, X, Y, Z, dX/dT, dY/dT, dZ/dT), \quad (18.18)$$

$$d(dY)/dT**2 = F2(T, X, Y, Z, dX/dT, dY/dT, dZ/dT), \quad (18.19)$$

$$d(dZ)/dT**2 = F3(T, X, Y, Z, dX/dT, dY/dT, dZ/dT). \quad (18.20)$$

Вводим обозначения: $dX/dT=XP$, $dY/dT=YP$ и $dZ/dT=ZP$, с использованием которых левые части уравнений (18.18)-(18.20) примут вид:

$$d(dX)/dT**2 = d(XP)/dT,$$

$$d(dY)/dT**2 = d(YP)/dT,$$

$$d(dZ)/dT**2 = d(ZP)/dT.$$

Теперь интегрируемая система уравнений (18.18)-(18.20) может быть представлена в виде системы из шести дифференциальных уравнений первого порядка:

$$dX/dT=XP, \quad (18.21)$$

$$dY/dT=YP, \quad (18.22)$$

$$dZ/dT=ZP, \quad (18.23)$$

$$d(XP)/dT=F1(T, X, Y, Z, XP, YP, ZP), \quad (18.24)$$

$$d(YP)/dT=F2(T, X, Y, Z, XP, YP, ZP), \quad (18.25)$$

$$d(ZP)/dT=F3(T, X, Y, Z, XP, YP, ZP). \quad (18.26)$$

Вводим две матрицы-столбца, размер которых равен количеству дифференциальных уравнений первого порядка KDU=6:

X(KDU) — массив значений переменных X,Y,Z и их первых производных: X(1)=X, X(2)=Y, X(3)=Z, X(4)=dX/dT, X(5)=dY/dT, X(6)=dZ/dT;

XP(KDU) — массив производных XP, YP и ZP, в котором XP(1)=dX/dT, XP(2)=dY/dT, XP(3)=dZ/dT, XP(4)=d(dX)/dT**2, XP(5)=d(dY)/dT**2, XP(6)=d(dZ)/dT**2.

Теперь систему дифференциальных уравнений первого порядка (18.21)-(18.26), то есть массив производных XP(KDU), можно задать в подпрограмме SDU1P с помощью принятых обозначений элементом фортран-программы следующей структуры:

$$XP(1)=X(4) \quad (18.27)$$

$$XP(2)=X(5) \quad (18.28)$$

$$XP(3)=X(6) \quad (18.29)$$

$$XP(4)=F1(T, X(1), X(2), X(3), X(4), X(5), X(6)) \quad (18.30)$$

$$XP(5)=F2(T, X(1), X(2), X(3), X(4), X(5), X(6)) \quad (18.31)$$

$$XP(6)=F3(T, X(1), X(2), X(3), X(4), X(5), X(6)) \quad (18.32)$$

Операторы (18.27)-(18.29) также будут одинаковыми для всех пространственных (трехмерных) задач, ибо они только присваивают введенные обозначения для производных dX/dT=X(4), dY/dT=X(5) и dZ/dT=X(6) соответственно первому, второму и третьему элементам массива производных XP(1), XP(2) и XP(3).

При записи правых частей операторов (18.30)-(18.32) также не следует обращать внимание на их довольно громоздкие выражения в общем виде, а просто записать конкретные для решаемой задачи соответствующие правые части интегрируемых уравнений (18.18)-(18.20) по правилам алгоритмического языка Фортран с учетом принятых обозначений.

Напомним, что обозначения аргумента T, массивов переменных X и производных XP, входящих в подпрограмму SUBROUTINE SDU1P(T,X,XP), а также само название подпрограммы SDU1P, являются формальными параметрами и могут быть обозначены любыми символическими именами вещественного типа. На их место при выполнении программы будут поставлены значения фактических параметров, с которыми и будут произведены все действия.

18.4. Описание используемых подпрограмм

18.4.1. Подпрограмма RKGS

Все используемые в этом разделе стандартные подпрограммы из пакета ПНП-SSP [12] для численного интегрирования (RKGS, HPCG, MAD3, OMAD) выбраны так, чтобы они были составлены с использованием одинаковых формальных параметров, хотя сами подпрограммы и используют разные методы. Тогда обращение к ним будет через одинаковые фактические параметры.

Вследствие этого описание выбранных подпрограмм будет весьма значительно отличаться друг от друга и мы подробно рассмотрим только одну из них (RKGS). Для других подпрограмм (HPCG, MAD3, OMAD), использующих те же формальные параметры, основное внимание будет обращено на их отличия от подпрограммы RKGS. Совпадающие же параметры с RKGS описываться не будут.

Подпрограмма RKGS из пакета ПНП-SSP [12] интегрирует методом Рунге-Кутты систему обыкновенных дифференциальных уравнений первого порядка, обычно имеющей в приложениях к задачам теоретической механики вид (18.6)-(18.7), (18.14)-(18.17) или (18.27)-(18.32) соответственно для линейных, плоских или пространственных задач.

Интегрирование происходит на заданном отрезке $[A, B]$, где $A=T_0$ есть начальная, а $B=TK$ — конечная точка, с учетом начальных условий при $T=0$ (обозначается T_0): $X_1(T_0)=X_{10}$, $X_2(T_0)=X_{20}$, ..., $X_N(T_0)=X_{N0}$ (задача Коши). Последовательное деление шага интегрирования пополам выполняется до тех пор, пока не будет определена величина шага, обеспечивающая заданную точность.

Обращение к подпрограмме RKGS имеет вид:

```
CALL RKGS (PRMT, X, XP, KDU, IH2, SDU1P, OUTF, AUX)
```

Ее параметры имеют следующий смысл:

PRMT — входной массив размером не менее 5, компонентами которого являются: PRMT(1) и PRMT(2) — начальная $A=T_0$ и конечная $B=TK$ границы интервала интегрирования, PRMT(3) — начальный шаг интегрирования, PRMT(4) — верхняя граница погрешности вычислений, PRMT(5) — параметр, служащий для связи подпрограммы OUTF с RKGS. При входе в подпрограмму RKGS PRMT(5) присваивается значение 0. Если PRMT(5)=0, RKGS будет работать до тех пор, пока система уравнений не будет проинтегрирована на всем отрезке $[A, B]$.

При работе подпрограммы RKGS периодически происходит обращение к подпрограмме OUTF, ведущей обработку выходных величин. Если необходи-

мо закончить интегрирование в какой-либо промежуточной точке отрезка $[A,B]$, в OUTP следует заменить значение PRMT(5) на ненулевое. Если PRMT(5) не равно 0, происходит выход из RKGS, независимо от того, закончено интегрирование системы уравнений на отрезке $[A,B]$ или нет.

X — входной массив начальных значений ($X10, X20, \dots, XN0$). В дальнейшем X становится вектором-решения системы ($X1, X2, \dots, XN$) в соответствующих точках отрезка $[A,B]$.

XP — входной вектор $XP(1), XP(2), \dots, XP(N)$ весовых коэффициентов погрешности (сумма его компонент должна быть равна 1). В процессе работы становится вектором производных функций $X1, X2, \dots, XN$, обозначаемых $XP(1), XP(2), \dots, XP(N)$, в соответствующих точках отрезка $[A,B]$.

KDU — количество дифференциальных уравнений первого порядка. KDU равно 2, 4, или 6 соответственно для линейной, плоской или пространственной задачи динамики материальной точки.

ИН2 — результирующее значение, которое определяет число делений пополам начального шага. Выход из подпрограммы RKGS с присвоением ИН2 следующих значений происходит:

ИН2=11 — если для достижения заданной точности требуется более 10-ти делений шага пополам;

ИН2=12 — если начальный шаг интегрирования PRMT(3) вследствие ошибки задан равным нулю;

ИН2=13 — если $SIGN(PRMT(3)).NE.SIGN(PRMT(2)-PRMT(1))$, т.е. знак начального шага интегрирования PRMT(3) не совпадает со знаком разности между конечным $PRMT(2)=TK$ и начальным $PRMT(1)=T0$ значениями интервала интегрирования.

SDU1P — имя внешней подпрограммы, которая задает интегрируемую систему дифференциальных уравнений 1-го порядка (см. п. 18.3). Список ее формальных аргументов должен включать T,X,XP.

OUTP — имя внешней подпрограммы, которая ведет обработку выходных величин подпрограммы RKGS. В OUTP полученные результаты выводятся на печать, причем она может быть организована таким образом, что результаты будут выводиться на печать не после каждого шага (т.к. шаг интегрирования может быть достаточно мал), а лишь в выбранных точках отрезка $[A,B]$. Подпрограмма RKGS общается к OUTP после выполнения каждого шага интегрирования, поэтому в OUTP можно выполнять любые вычисления с использованием результатов расчета до того, как продолжить процесс интегрирования. Список формальных параметров OUTP должен включать T,X,XP,ИН2,KDU,PRMT.

AUX — рабочий массив размером 8SKDU: AUX(8,KDU).

18.4.2. Описание других стандартных подпрограмм

Все нижеследующие подпрограммы используют многоступенчатые или многошаговые методы (см. п. 18.2.2).

Подпрограмма HPCG выполняет интегрирование системы дифференциальных уравнений первого порядка модифицированным предиктор-корректор методом Хемминга. Обращение к подпрограмме HPCG имеет вид:

```
CALL HPCG (PRMT, X, XP, KDU, IH2, SDU1P, OUTP, AUX)
```

Все параметры этой подпрограммы совпадают с RKGS, только рабочий массив AUX, описываемый в операторе DIMENSION, должен иметь размер $16 \times KDU$ (где вместо KDU, конечно, указывается количество дифференциальных уравнений 1-го порядка в данной задаче: 2, 4 или 6):

```
DIMENSION AUX(16, KDU)
```

Подпрограмма OMAD использует обобщенный метод Адамса четвертого порядка с переменным шагом интегрирования. Все параметры этой подпрограммы полностью совпадают с RKGS. Обращение к подпрограмме OMAD имеет вид:

```
CALL OMAD (PRMT, X, XP, KDU, IH2, SDU1P, OUTP, AUX)
```

Подпрограмма MAD3 выполняет интегрирование системы дифференциальных уравнений первого порядка трехточечным методом Адамса. Обращение к подпрограмме MAD3 имеет вид:

```
CALL MAD3 (PRMT, X, XP, KDU, IH2, SDU1P, OUTP, AUX)
```

Параметры MAD3 также совпадают с RKGS за исключением следующих отличий: рабочий массив AUX, описываемый в операторе DIMENSION, должен иметь размер $6 \times KDU$: DIMENSION AUX(6, KDU); а результирующее значение IH2 при выходе из MAD3 принимает следующие значения: IH2=-2 — текущий шаг интегрирования стал меньше $0.1E-4$; IH2=-1 — начальный шаг интегрирования PRMT(3) равен нулю или имеет неверный знак.

Следует отметить, что при работе MAD3, в отличие от других рассматриваемых подпрограмм, последняя точка интегрирования не точно совпадает с конечным значением интервала интегрирования PRMT(2)=TK, достаточно близко приближаясь к нему.

ГЛАВА 19. ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ ОДНОМЕРНЫХ ЗАДАЧ ДИНАМИКИ

Описание программ численного интегрирования проведем на основе подпрограммы RKGS, использующей метод Рунге-Кутты. Для практической работы со стандартными подпрограммами, использующими другой метод, следует внести незначительные изменения в указанные операторы (см. пп. 19.2.1-19.2.3). Совпадающие части программ описываться не будут.

Работа с более совершенной подпрограммой RKF45 [27] обладает большими отличиями и описывается в п. 22.

Везде далее в качестве примеров также выбраны рассмотренные ранее дифференциальные уравнения (16.2)-(16.4) и (17.6)-(17.7), численное интегрирование которых описывалось в 6-й части настоящего пособия. Это сделано для обеспечения более плавного перехода от использования готовых программ к самостоятельному их написанию.

19.1. Использование подпрограммы RKGS

Ниже приводится программа 19.1 для численного интегрирования одного дифференциального уравнения 2-го порядка типового примера задания Д-4 (16.3), представленного с использованием вспомогательных переменных:

```

С      П Р О Г Р А М М А 19.1
      EXTERNAL SDU1P,OUTP                10
      DIMENSION PRMT(5),X(2),XP(2),AUX(8,2) 20
      COMMON TPR                          30
      DATA PRMT/0.,1.2,0.01,0.001,0./      40
      DATA X/0.3,2./                      45
      DATA XP/2*0.5/                      50
      TPR=0.                               54
      PRINT 5                              60
5     FORMAT(T4,'ВРЕМЯ, СЕК',T16,'КООРДИНАТА X, М',
*      T32,'СКОРОСТЬ XP, М/С')
      CALL RKGS(PRMT,X,XP,2,IH2,SDU1P,OUTP,AUX) 65
      STOP                                 98
      END                                  99
      SUBROUTINE SDU1P(T,X,XP)            100
      DIMENSION X(2),XP(2)               110
      DATA PI/3.14159/, ZM/0.01/, R/0.2/, C/1./, 120
*      ZL0/0.2/, G/9.81/
      AL=PI/6.                            130
      C1=- (C/ZM-PI*PI*(SIN(AL))**2)      134

```


C5=PI*PI*R*SIN(AL)-G*COS(AL)+C*ZL0/ZM	136
XP(1)=X(2)	140
XP(2)=C1*X(1)+C5	150
RETURN	196
END	198
SUBROUTINE OUTP(T,X,XP,IH2,KDU,PRMT)	200
DIMENSION X(2),XP(2),PRMT(5)	210
COMMON TPR	220
IF(ABS(T-TPR)-0.0001) 20,20,40	230
20 PRINT 25,T,X(1),X(2)	240
25 FORMAT(2X,3(2X,G12.5))	241
TPR=TPR+0.1	250
40 RETURN	260
END	262

Оператор 10 объявляет имена SDU1P и OUTP внешних подпрограмм, являющихся фактическими параметрами при обращении к RKGS.

Оператор 20 задает описание массивов PRMT, X, XP и AUX, содержащих соответственно 5, 2, 2 и 16 элементов.

Оператор 30 описывает общую область COMMON, содержащую переменную TPR, которая определяет значение времени, при котором будет осуществляться печать результатов решения.

Оператор 40 DATA используется для задания начальных значений элементам массива PRMT (см. п. 18.4.1), которым в момент загрузки программы присваивается: PRMT(1)=0, PRMT(2)=1.2, PRMT(3)=0.01, PRMT(4)=0.001, PRMT(5)=0. Напомним, что при входе в подпрограмму RKGS устанавливается PRMT(5)=0, а начальный шаг интегрирования PRMT(3) должен быть выбран меньше, чем шаг, используемый для вывода на печать.

Оператор 45 задает начальные значения переменной X, носящей в программе обозначение X(1), и ее производной X(2) при времени T=0, которые приведены в условии рассматриваемого примера: X(1)=0.3, X(2)=2.

Оператор 50 присваивает начальные значения весовым коэффициентам погрешности в массиве производных XP, которые в сумме должны быть равны единице (в нашем случае все весовые коэффициенты равны XP(1)=XP(2)=0.5).

Оператор 54 задает начальное значение переменной TPR, т.е. определяет значение времени T, с которого будет начат вывод результатов решения на печать.

Оператор 60 под управлением оператора 61 выводит на печать заголовков таблицы результатов решения, рассчитанной так, чтобы значения соответствующих величин располагались под их обозначениями. В одной строке соответственно с 4-й, 16-й и 32-й позиции будет напечатано:

ВРЕМЯ, СЕК КООРДИНАТА X, М СКОРОСТЬ XP, М/С

Оператор 65 осуществляет обращение к подпрограмме RKGS с указанными фактическими параметрами. Те из них, обозначения которых совпадают с использованными в п. 18.4.1 формальными параметрами (PRMT, X, XP, IH2, SDU1P, OUTF, AUX) согласно принятому соглашению имеют тот же смысл. Количество дифференциальных уравнений первого порядка KDU задано числом 2. В результате выполнения подпрограммы RKGS будут вычислены и выведены на печать значения аргумента T и соответствующие ему значения координаты материальной точки X(1) и ее скорости X(2) в точках 0., 0.1, 0.2, 0.3, ..., 1.2 (вывод на печать обеспечивается подпрограммой OUTF).

Операторы 98, 99 указывают на конец вычислений и окончание основной программы.

Далее следует подпрограмма SDU1P, задающая интегрируемое уравнение второго порядка в виде системы дифференциальных уравнений первого порядка (см. п. 18.3.1) и вычисляющая правые части уравнений этой системы. Напомним, что подпрограмму SDU1P нам уже приходилось ранее составлять при работе с универсальными программами для тех же примеров (см. п. 17.2).

Оператор 100 именуется подпрограмму SDU1P и задает ее формальные параметры: T, X и XP.

Оператор 110 задает описание формальных массивов X(2) и XP(2). Для простоты они указаны постоянного размера, равного количеству дифференциальных уравнений первого порядка KDU (см. дополнение 19.1).

Следующая группа операторов 120-136 является вспомогательной и служит для подготовки к заданию с использованием дополнительных переменных правой части интегрируемого уравнения (16.3).

В результате соответствия, устанавливаемого оператором 120 DATA, перечисленные в нем переменные приобретают следующие начальные значения: PI=3.14159, ZM=0.01, R=0.2, C=1., ZL0=0.2, G=9.81.

Арифметический оператор присваивания 130 определяет значение вспомогательной переменной AL, являющейся аргументом для вычисления тригонометрических функций в операторах 134-136.

Оператор 134 вычисляет выражение для коэффициента при переменной X уравнения (16.3), обозначенного C1, а оператор 136 определяет значение постоянной составляющей правой части этого уравнения, обозначенной C5.

Операторы 140-150 представляют интегрируемое дифференциальное уравнение второго порядка в виде системы двух дифференциальных уравнений первого порядка и задают формулы (в данном случае с учетом операторов 120-136) для вычисления правых частей уравнений этой системы (см. п. 18.3.1).

Оператор 196 осуществляет возврат в вызывающую программу (в подпрограмму RKGS).

Оператор 198 указывает на конец самостоятельной программной единицы — подпрограммы SDU1P.

Подпрограмма OUTP, следующая далее, обеспечивает печать результатов решения в заданных точках отрезка $[T_0, T_K]$, т.е. $[PRMT(1), PRMT(2)]$.

Оператор 200 определяет, именуется подпрограмму OUTP и задает ее формальные параметры: T, X, XP, IH2, KDU и PRMT, имеющие тот же смысл, что и соответствующие параметры подпрограммы RKGS, при рассмотрении которой они и описаны (см. п. 18.4.1).

Оператор 210 задает описание формальных массивов. Их также для составления более универсальной программы можно задать с регулируемые размерами для X и XP (см. дополнение 19.1).

Оператор 220 описывает общую область, содержащую переменную TPR. В соответствии с этим под переменную TPR будет выделена та же ячейка памяти, что и под переменную TPR, включенную в общую область в основной программе. Для удобства чтения программы эти переменные обозначены здесь одинаковыми идентификаторами, но без оператора общих областей COMMON одинаковые обозначения не играли бы никакой роли и эти одноименные переменные в двух разных программных единицах были бы независимы друг от друга.

Условный арифметический оператор 230 осуществляет разветвление вычислительного процесса на две ветви. Если текущее значение аргумента T совпадает с точностью 0.0001 со значением TPR (т.е. разность $(T-TPR)-0.0001$ меньше или равна нулю), то осуществляется переход к оператору с меткой 20 (оператору PRINT). В противном случае — переход к оператору с меткой 40 для возврата в подпрограмму RKGS.

Оператор 240 (с меткой 20) под управлением оператора 241 (с меткой 25) выводит на печать значения аргумента T и искомым функций: координаты $X(1)$ и ее скорости $X(2)$ в точке $T=TPR$.

Матрица-столбец X представляет собой выход подпрограммы RKGS с результатами решения для каждого значения времени T, изменяющегося с шагом, равным шагу интегрирования. Однако печатаются они с выбранным шагом вывода на печать (0.1), на который оператор 250 изменяет значение величины TPR после очередной печати результатов операторами 240-241.

Оператор 260 (с меткой 40) осуществляет возврат в вызывающую программу (подпрограмму RKGS), а оператор 262 указывает на конец подпрограммы OUTP.

Дополнение 19.1. Задание массивов переменных размеров в подпрограммах важно для создания универсальных программ и может быть реализовано двумя способами. По первому из них указатели размера массивов в операторе DIMENSION в качестве максимальных значений индексов содержат простые переменные целого типа, например, X(KDU) и XP(KDU). Соответствующее значение KDU должно присваиваться в основной программе и передаваться в подпрограмму до обращения к ней. В программе 19.1 применить для этой цели аппарат формальных-фактических параметров невозможно, т.к. их список ограничен подпрограммой RKGS. Поэтому следует использовать оператор COMMON:

COMMON /KK/KDU	32
KDU=2	53
DIMENSION X(KDU), XP(KDU)	110
COMMON /KK/KDU	112

Во втором способе используется то обстоятельство, что размер формального массива определяется размером соответствующего фактического массива, заданного в основной программе. Так как компилятор Фортрана не проверяет индекс на превышение границы, то в программе 19.1 в операторе 110 следует просто указать: DIMENSION X(1), XP(1). После этого можно работать с массивами X и XP нужной длины, конкретно указанной в основной программе для соответствующих фактических параметров.

Дополнение 19.2. Определение шага интегрирования подпрограммы RKGS.

Подпрограмма RKGS автоматически изменяет шаг интегрирования для достижения заданной точности в зависимости от характера интегрируемого уравнения. Чтобы проследить за этим изменением шага нужно убрать из подпрограммы оператор 230, а также становящиеся после этого ненужными операторы 220 и 250.

Тогда оставшиеся операторы 240 и 241 будут выводить на печать значения результатов решения для каждого значения аргумента T, изменяющегося с шагом интегрирования (конечно, количество выдаваемых на печать результатов резко возрастет). Обратившись затем к выдаче рассматриваемой программы для значений аргумента T, можно проследить за тем, как изменяется шаг интегрирования во времени в процессе вычислений.

Дополнение 19.3. Варианты обхода соглашения по умолчанию.

В программе 19.1 в операторах 120-136 в обозначении для массы шарика ZM и длины недеформированной пружины ZL0 добавлена буква Z по сравнению с условием (16.3), чтобы обойти неявное описание типа, согласно которому переменные, начинающиеся с букв M и L, должны быть целыми.

Заметим, что второй возможностью обойти соглашение по умолчанию, оставив неизменными в программе 19.1 обозначения переменных M и L0 условия (16.3), является использование в программе 19.1 оператора явного описания типа:

```

REAL M,L0                                111
DATA PI/3.14159/, M/0.01/, R/0.2/, C/1./, 120
*   L0/0.2/, G/9.81/
C1=- (C/M-PI*PI*(SIN(AL))**2)           134
C5=PI*PI*R*SIN(AL)-G*COS(AL)+C*L0/M    136

```

Дополнение 19.4. Простая форма представления подпрограммы SDU1P.

Отметим, что группа операторов (120-136) приведена в качестве примера записи правой части интегрируемого дифференциального уравнения в общем виде. Она может иметь разную форму в зависимости от вида уравнения и должна конкретно задаваться студентом для своего варианта задачи. Если это покажется неудобным, то можно на микрокалькуляторе определить нужные коэффициенты (в данном случае C1 и C5) и с их помощью записать требуемое выражение, предварительно удалив из программы всю группу операторов 120-136. Оператор 140 при этом всегда будет оставаться без изменений, а оператор 150 примет вид:

```

XP(2)=-97.533*X(1)+12.491                150

```

В такой более простой форме подпрограмма SDU1P применялась нами ранее при работе с универсальными программами (см. фрагмент (17.4)). Однако такая более короткая форма записи подпрограммы SDU1P имеет ряд неудобств. Она требует большего количества предварительных вычислений на микрокалькуляторе, в результате которых также снижается точность вычисленных коэффициентов.

Дополнение 19.5. Сравнение результатов численного и аналитического решений.

Для этого нужно в базовой программе 19.1 для рассматриваемого примера (16.3) внести следующие изменения и дополнения:

```

5  FORMAT(1X,T4,'ВРЕМЯ, СЕК',T16,'КООРДИНАТА X, M', 61
*   T32,' ТОЧНОЕ XT',T48,'RASX=X-XT',T59,' СКОРОСТЬ XP,M/C',
*   T76,' ТОЧНОЕ XPT',T90,'RASXP=XP-XPT')
COMMON /ZCDU/C1,C5                                112
COMMON TPR /ZCDU/D1,D5                            120
20 ZK=SQRT(-D1)                                    231
   B=-D5/D1                                         232
   C1=0.3-B                                         233
   C2=2./ZK                                         234
   ZKT=ZK*T                                         235

```

$XT=C1 * \cos (ZKT) + C2 * \sin (ZKT) + B$	236
$XPT=ZK * (-C1 * \sin (ZKT) + C2 * \cos (ZKT))$	237
$RASX=X(1) - XT$	238
$RASXP=X(2) - XPT$	239
$PRINT 25, T, X(1), XT, RASX, X(2), XPT, RASXP$	240
$25 FORMAT(2X, 7(2X, G12.5))$	241

Оператором основной программы 60 под управлением замененного оператора 61 (с меткой 5) в заглавии таблицы результатов решения дополнительно к времени T и координате X будет напечатано: с 32-й позиции “ТОЧНОЕ XT ”, с 48-й — “ $RASX=X-XT$ ”, с 59-й — “СКОРОСТЬ XP , м/с”, с 76-й — “ТОЧНОЕ XPT ” и с 90-й позиции “ $RASXP=X-XP$ ”. Расположение этих обозначений предварительно рассчитывается так, чтобы под ними печатались соответствующие величины группой операторов 240-241. Заметим, что эпитет “точное” понимается в смысле “аналитическое” и используется из-за своей краткости.

Оператор 112 вставляется после оператора 110 в подпрограмме SDUIP. Он описывает именованную общую область с именем ZCDU (в нее войдут значения коэффициентов дифференциального уравнения $C1$ и $C5$, задаваемого подпрограммой SDUIP).

Оператор 220 заменяет соответствующий оператор в подпрограмме OUTP. Он описывает две общие области: неименованную (в нее войдет величина TPR) и с именем ZCDU (в нее войдут величины $D1$, $D5$). В соответствии с этим под переменную TPR будет выделена та же ячейка памяти, что и под переменную TPR , включенную в неименованную общую область в основной программе. Под переменные $D1$ и $D5$ будут выделены те же ячейки памяти, что и под переменные $C1$ и $C5$ соответственно в подпрограмме SDUIP, так как эти величины включены в одну общую область с именем ZCDU в разных программных единицах.

Теперь определенные в подпрограмме SDUIP коэффициенты дифференциального уравнения $C1$ и $C5$ можно использовать под обозначениями $D1$ и $D5$ при записи точного аналитического решения интегрируемого дифференциального уравнения в подпрограмме OUTP. Используемая постоянная интегрирования $C1$ в подпрограмме OUTP не имеет никакого отношения к одноименной величине коэффициента дифференциального уравнения $C1$ в подпрограмме SDUIP, так как они не включены в одну общую область и являются абсолютно независимыми величинами.

Группа операторов 231-239 вставляется в подпрограмму OUTP, из них 231-235 являются вспомогательными для записи операторами 236-237 точного решения (координаты XT и скорости XPT) рассматриваемого примера (16.3).

Операторы 238 и 239 определяют разности $RASX$ и $RASXP$ между численными и аналитическими решениями для координаты и скорости соответственно.

Оператор 240 (с меткой 20) под управлением оператора 241 (с меткой 25) выводит на печать значение аргумента Т, численно найденных значений координаты $X=X(1)$ и скорости $XP=X(2)$, аналитически определенных значений XT и XPT , а также их соответствующих разностей $RASX=X-XT$ и $RASXP=XP-XPT$.

По оператору 241, отступив 2 позиции от левого края (первое 2X), семь раз будут выведены значения величин, указанных в списке оператора 240 PRINT, при этом на печать числа по формату G12.5 отводится 12 позиций, и еще 2 позиции пропускаются между числами согласно указания 2X перед G12.5. В соответствии с этим значения выводимых величин будут располагаться в следующих позициях: 5-16, 19-30, 33-44, 47-58, 61-72, 75-86, 89-100. Такой расчет предварительно нужно проделать, чтобы в операторе 61 при печати заголовка таблицы результатов решения расположить обозначения соответствующих величин над распечатываемыми числами. Для этого используется спецификация типа "Т" с указанием номера позиции, с которого будет начинать печататься следующая за ним запись.

Дополнение 19.6. Численное интегрирование типовых примеров заданий Д-3 (16.2) и Д-23 (16.4).

Для интегрирования любых других дифференциальных уравнений одномерных задач механики следует изменить при необходимости задание начальных значений следующим переменным:

- в операторе 40 соответственно первых четырех компонент массива PRMT: нижней PRMT(1) и верхней PRMT(2) границы интервала интегрирования, начального шага интегрирования PRMT(3) и допустимой погрешности вычислений PRMT(4);
- в операторе 45 переменной X, носящей в программе обозначение X(1), и ее производной X(2) при времени T=0. Их значения приводятся или определяются из условия рассматриваемой задачи.

Следует также в подпрограмме SDU1P заменить оператор 150, задающий правую часть интегрируемого уравнения.

Оставим без изменения начальные значения параметров интегрирования в операторе 40. Если использовать для рассматриваемых типовых примеров (16.2) и (16.4) задание уравнений с предварительно определенными числовыми коэффициентами, то после удаления из программы 19.1 операторов 120-136, определяющих вспомогательные переменные, в ней нужно будет заменить только операторы 45 и 150:

- для задания Д-3 (16.2):

DATA X/-.0245,0./	45
XP(2)=-300.*X(1)+6.*SIN(10.*T)	150

- для задания Д-23 (16.4):

```
DATA X/0.002,0.08/           45
XP(2)=-744.114*X(1)         150
```

Здесь в операторе 45 изменены только значения начальных условий координаты $X(1)$ и скорости $X(2)$ при $T=0$, поэтому в результате соответствия, определенного этим оператором, указанные переменные получают в момент загрузки программы следующие значения: для задания Д-3: $X(1)=-0.0245$, $X(2)=0$; для задания Д-23: $X(1)=0.002$, $X(2)=0.08$.

Дополнение 19.7. Численное интегрирование типового примера задания Д-27 (17.2). В этом случае также следует выполнить действия, рассмотренные в дополнении 19.6. Вследствие относительной сложности уравнения (17.2) для его представления используются вспомогательные переменные A , B , C , D и E (как и ранее во фрагменте (17.5)). Поэтому в программе 19.1 следует заменить и вставить следующие операторы:

```
DATA PRMT/0.,0.341,0.001,0.0001,0./   40
DATA X/0.5,0./                         45
A=12.6                                 120
B=78./A                                124
C=1944./A                               130
D=684./A                                134      (19.1)
E=3888./A                               136
XP(2)=- (B+C*X(1) *X(1) ) *X(1) /ABS (X(1) ) - 150
*      D*X(1) -E*X(1) **3
TPR=TPR+0.011                          250
```

В результате соответствия, определенных операторами 40 и 45, указанные переменные получают в момент загрузки программы следующие значения: $PRMT(1)=0.$, $PRMT(2)=0.341$, $PRMT(3)=0.001$, $PRMT(4)=0.0001$, $PRMT(5)=0$ и $X(1)=0.5$, $X(2)=0$.

В операторе 40 границы интервала интегрирования ($PRMT(1)$ и $PRMT(2)$) выбраны одинаковыми с соответствующими величинами в [21, с. 361] для возможности сравнения результатов. Из-за уменьшения интервала интегрирования значения начального шага $PRMT(3)$ и допустимой погрешности вычислений $PRMT(4)$ уменьшены на порядок по сравнению с базовой программой 19.1.

Вспомогательные переменные, определенные операторами 120-136, используются затем в операторе 150, задающем правую часть дифференциального уравнения.

Шаг вывода на печать (0.11), на который оператор 250 изменяет значение величины TPR , также выбран совпадающим с [21, с. 361] для возможности сравнения результатов.

Дополнение 19.8. Защита от возможной ошибки переполнения порядка.

При задании интегрируемого уравнения в операторе 150 фрагмента (19.1) в знаменателе стоит функция $ABS(X(1))$.

При расчете по программе 19.1 с использованием фрагмента (19.1) для какого-либо времени T может случиться, что значение координаты $X(1)$ окажется равным или весьма близким к нулю. Так как в операторе 150 происходит деление на эту величину, то это может вызвать искажение расчета или переполнение порядка (при значении числа более 10^{*75}).

Для защиты от возможной ошибки переполнения порядка в программу 19.1 после добавления фрагмента (19.1) следует вставить следующую группу операторов:

```

      IF (ABS (X (1)) .LE. 1.E-33) GOTO 5          145
      XP (2) = -(B+C*X (1) *X (1) *X (1) /ABS (X (1)) - 150   (19.2)
      *      D*X (1) -E*X (1) **3
      GOTO 10                                     152
5     XP (2) = -D*X (1) -E*X (1) **3           154
10    RETURN                                     196

```

Условный логический оператор IF 145 анализирует на истинность логическое выражение, заключенное в скобки $(ABS(X(1)), LE. 1.E-33)$. Если значение абсолютной величины координаты $X(1)$ по какой-либо причине окажется меньше $1.E-33$, то выражение в скобках будет истинным. Поэтому оператор безусловного перехода GOTO передаст управление оператору с меткой 5 и значение $XP(2)$ будет вычисляться по оператору 154. В нем учтено условие, что при $X(1)=0$ значение $X(1)/ABS(X(1))$ также равно нулю.

Во всех остальных случаях выражение в скобках будет ложным, вследствие чего оператор 145 пропускается и выполняется следующий за ним оператор 150 (полностью совпадающий с имеющимся в (19.1)), задающий правую часть дифференциального уравнения $XP(2)$ для общего случая, когда $X(1)$ не равно нулю.

После его выполнения следующий за ним оператор безусловного перехода 152 передаст управление оператору 196 с меткой 10 RETURN, который осуществит возврат в вызывающую программу.

Используемое в подобных случаях значение диапазона $1.E-33$ выбрано потому, что оно достаточно мало, чтобы не исказить результаты дальнейшего решения. Вместе с этим обратная к ней величина гораздо меньше уровня переполнения и не вызовет аварийной ситуации.

Отметим, что задание интегрируемой системы уравнений по фрагменту (19.2) можно применять и при использовании вместо выражения $X(1)/ABS(X(1))$ функции присваивания знака SIGN.

Дополнение 19.9. Заметим, что интегрируемую систему дифференциальных уравнений в подпрограмме SDU1P, представленную в примерах (19.1) и (19.2), можно описать в любой кажущейся вам удобной форме. После оператора описания DIMENSION и перед заданием интегрируемой системы уравнений, может быть как любое количество операторов, вводящих используемые вспомогательные переменные (для (19.1) и (19.2) A, B, C, D, E), так и ни одного. В последнем случае задание уравнений движения происходит сразу с использованием чисел, что для примера (19.2) может принять, например, следующий вид:

```

SUBROUTINE SDU1P(T,X,XP)                                100
DIMENSION X(2),XP(2)                                    110
XP(1)=X(2)                                              140
IF(ABS(X(1)).LE.1.E-33) GOTO 5                          145
XP(2)=- (78./12.6+(1944./12.6)*X(1)*                    150
* X(1))*X(1)/ABS(X(1))-(684./12.6)*
* X(1)-(3888./12.6)*X(1)**3                               (19.3)
GOTO 10                                                  152
5 XP(2)=- (684./12.6)*X(1)-
* (3888./12.6)*X(1)**3                                    154
10 RETURN                                               196
END                                                       198

```

19.2. Использование других стандартных подпрограмм, реализующих многшаговые методы

19.2.1. Использование подпрограммы HPCG

При любом численном решении всегда встает вопрос о достоверности полученных результатов. В учебных задачах часто возможно аналитическое решение, результаты которого хотя бы в одной точке следует сравнить с численным решением. Если это сделать в самой программе, то сравнение результатов будет происходить для каждой выводимой на печать точки (см. дополнение 19.5).

Однако в подавляющем большинстве случаев такого сравнения сделать невозможно. В подобных ситуациях достаточно хорошей формой оценки достоверности полученных результатов является численное решение задачи двумя (или несколькими) принципиально различными методами.

Для осуществления такой возможности, как уже отмечалось, специально выбраны такие подпрограммы, обращение к которым полностью совпадает с обращением к подпрограмме RKGS. Тогда и сама основная

программа, использующая другой метод интегрирования, и ее внешние подпрограммы, составленные пользователем, будут практически полностью совпадать с описанными в разделе 19.1. Незначительные отличия каждой из них будут рассмотрены ниже.

Для практического использования подпрограммы HPCG, реализующей модифицированный предиктор-корректор метод Хемминга, следует заменить в программе 19.1 два оператора, номера которых совпадают с указанными ниже:

```
DIMENSION PRMT(5), X(2), XP(2), AUX(16,2)           20
CALL HPCG(PRMT, X, XP, 2, IH2, SDU1P, OUTP, AUX)     65
```

Отметим, что при работе с подпрограммой HPCG, как и с другими описываемыми ниже стандартными подпрограммами, сохраняется возможность использования любых из рассматриваемых дополнений.

19.2.2. Использование подпрограммы OMAD

Все параметры подпрограммы OMAD, реализующей обобщенный метод Адамса четвертого порядка с переменным шагом интегрирования, совпадают с подпрограммой RKGS. Поэтому для ее практического использования в основной программе 19.1 следует заменить один оператор:

```
CALL OMAD(PRMT, X, XP, 2, IH2, SDU1P, OUTP, AUX)    65
```

19.2.3. Использование подпрограммы MAD3

Для практического использования подпрограммы MAD3, в программе 19.1 должны быть заменены два оператора:

```
DIMENSION PRMT(5), X(2), XP(2), AUX(6,2)           20
CALL MAD3(PRMT, X, XP, 2, IH2, SDU1P, OUTP, AUX)    65
```

Дополнение 19.10. Число делений начального шага пополам IH2.

Подпрограмма MAD3 присваивает в случае аварийного завершения работы и выхода в основную программу величине IH2 значения, сообщающие о причинах выхода (см. п. 18.4.2), которые отличаются от аналогичных сообщений подпрограммы RKGS (см. п. 18.4.1). Поэтому часто за этой величиной производят наблюдение, выводя ее значение на печать вместе с результатами решения. Для этого нужно в программе 19.1 выполнить замены п. 19.2.3 и операторы с указанными номерами дополнить до следующего вида:

```
5  FORMAT(1X, T4, 'ВРЕМЯ, СЕК', T16, 'КООРДИНАТА X, M', 61
   * T32, 'СКОРОСТЬ XP, М/С', T50, 'ЗНАЧЕНИЕ IH2')
20 PRINT 25, T, X(1), X(2), IH2                       240
25 FORMAT(2X, 4(2X, G12.5))                            241
```

Оператором основной программы 60 под управлением оператора 61 (с меткой 5) в заголовии таблицы результатов решения с 50-й позиции будет дополнительно напечатано “ЗНАЧЕНИЕ IH2”.

Под ним в подпрограмме OUTP оператором 240 (с меткой 20) под управлением оператора 241 (с меткой 25) вместе с результатами решения будет дополнительно печататься значение IH2, т.е. количество делений пополам начального шага интегрирования. Формат G, использованный в операторе 241, универсальный: по нему можно выводить целые числа и вещественные (даже удвоенной точности), поэтому он для простоты также использовался нами для вывода на печать целых значений IH2.

Аналогичным образом можно выводить на печать другие величины, которые могут нас заинтересовать, например, ускорение XP(2).

Дополнение 19.11. Для получения универсальной программы для численного интегрирования дифференциального уравнения любым из рассматриваемых методов в программу 19.1 следует внести следующие изменения и дополнения:

```

DIMENSION PRMT (5) , X (2) , XP (2) , AUX (8, 2) ,           20
* AUXH (16, 2) , AUXM (6, 2)
IPARPP=2                                                    62
IF (IPARPP.EQ.1) CALL RKGS (PRMT, X, XP, 2, IH2,           65
* SDU1P, OUTF, AUX)                                       (19.4)
IF (IPARPP.EQ.2) CALL HPCG (PRMT, X, XP, 2, IH2,           66
* SDU1P, OUTF, AUXH)
IF (IPARPP.EQ.3) CALL MAD3 (PRMT, X, XP, 2, IH2,           67
* SDU1P, OUTF, AUXM)
IF (IPARPP.EQ.4) CALL OMAD (PRMT, X, XP, 2, IH2,           68
* SDU1P, OUTF, AUX)

```

Оператор 20 задает описание всех массивов, используемых разными подпрограммами: AUX(8,2) для RKGS и OMAD, AUX(16,2) для HPCG и AUX(6,2) для MAD3.

Арифметический оператор присваивания 62 задает значение параметра выбора подпрограмм IPARPP, которое для обычной точности может лежать в диапазоне от 1 до 4-х.

Четыре условных логических оператора 65-68 выполняют однотипные действия для разных значений параметра выбора подпрограммы IPARPP, но при этом только одно логическое выражение, заключенное в скобки, окажется истинным для конкретного значения IPARPP. Если IPARPP=1 — используется RKGS, IPARPP=2 — HPCG, IPARPP=3 — MAD3, при IPARPP=4 — используется OMAD.

Например, при IPARPP=2 окажется истинным логическое выражение в скобках оператора 66, вследствие чего будет осуществлено по оператору CALL обращение к подпрограмме HPCG и для численного интегрирования будет использован предиктор-корректор метод Хемминга.

Во всех случаях для трех других условных логических операторов, анализирующих на истинность другие значения IPARPP (для нашего примера при IPARPP=1, 3 и 4), выражения в скобках окажутся ложными и соответствующие операторы будут пропущены.

Таким образом, при любом значении IPARPP от 1 до 4 будет обращаться с вызовом к подпрограмме по оператору CALL всегда только один оператор из четырех 65-68 (для IPARPP=2 оператор 66), и будет отработана только одна подпрограмма (в нашем случае HPCG), реализующая нужный нам метод численного интегрирования.

Напомним, что если задать значение параметра выбора подпрограммы IPARPP любым способом в самой программе (например, оператором DATA или арифметическим оператором присваивания 62), то после изменения его значения каждый раз нужно проводить транслирование программы (см. п. 8.4). Это неудобно, поэтому целесообразно значение этого параметра задать в программе оператором ввода READ, например:

```
READ *, IPARPP 62
```

Теперь в файле с выбранным вами именем с 1-й позиции в 1-й строке следует задать значение IPARPP, а при его изменении можно сразу отправлять программу на выполнение.

Однако, при ошибке задания IPARPP и выходе его значения за допустимые пределы, будут ложными выражения в скобках во всех 4-х условных логических операторах 65-68. В результате этого все они будут пропущены и по операторам 60-61 напечатается только заголовок таблицы результатов решения. Для сообщения о подобной ошибке в программу 19.1 после добавления фрагмента (19.4) можно вставить, например, следующую группу операторов:

```
IF ( (IPARPP.LT.1) .OR. (IPARPP.GT.4) ) PRINT 20 90  
20 FORMAT (/5X, 'ВНИМАНИЕ! ОШИБКА В ЗАДАНИИ IPARPP!') 92
```

Оператор PRINT, осуществляющий печать аварийного сообщения, будет работать, когда значение IPARPP будет выходить за допустимые пределы (будет меньше 1 или больше 4-х). При правильном задании IPARPP выражение в скобках будет ложным, оператор PRINT будет пропущен и операторы 90-92 не будут оказывать никакого воздействия на работу программы.

ГЛАВА 20. ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ ДВУМЕРНЫХ И ТРЕХМЕРНЫХ ЗАДАЧ ДИНАМИКИ

Описание программ численного интегрирования плоских и пространственных задач динамики также проведем на основе подпрограммы RKGS, использующей метод Рунге-Кутты. При работе с другими стандартными подпрограммами следует внести изменения с учетом размерности задачи согласно пп. 19.2.1-19.2.3.

20.1. Двумерные (плоские) задачи динамики

Численное интегрирование плоских задач динамики, описываемых двумя дифференциальными уравнениями второго порядка, рассмотрим на типовом примере задания Д-2 [21, с. 130]. Интегрируемая система дифференциальных уравнений с учетом силы сопротивления, приведенная в [21, с. 137] в форме системы (9), после вычисления коэффициентов и выделения в левой части второй производной примет вид:

$$\begin{aligned} d(dX/dT)/dT &= -2*dX/dT - 4*X, \\ d(dZ/dT)/dT &= -2*dZ/dT - 4*Z - 9.81 \end{aligned} \quad (20.1)$$

Ее начальные условия при $T=0$ равны [21, с. 130]: для координат точки $X_0=0$ и $Z_0=10$ (м), для скоростей $XP_0=20$ и $ZP_0=0$ (м/с).

Ниже приводится программа 20.1 для численного интегрирования двух дифференциальных уравнений 2-го порядка (20.1). Из-за больших отличий от программы 19.1 она представлена полностью, однако совпадающие операторы не описываются. Для отличия они имеют после своего номера точку и цифру 1:

```

C      П Р О Г Р А М М А 20.1
      EXTERNAL SDU1P,OUTP                10.1
      DIMENSION PRMT(5),X(4),XP(4),AUX(8,4) 20
      COMMON TPR                          30.1
      DATA PRMT/0.,1.2,0.01,0.001,0./      40.1
      DATA X/0.,10.,20.,0./              45
      DATA XP/4*0.25/                    50
      TPR=0.                               54.1
      PRINT 5                              60.1
5  FORMAT(T4,'ВРЕМЯ, СЕК',T21,'X, М',T35,'Z, М',
*      T49,'XP, М/С',T62,'ZP, М/С')      61
      CALL RKGS(PRMT,X,XP,4,IH2,SDU1P,OUTP,AUX) 65
      STOP                                98.1
      END                                 99.1

```

SUBROUTINE SDUIP(T,X,XP)	100.1
DIMENSION X(4),XP(4)	110
XP(1)=X(3)	140
XP(2)=X(4)	142
XP(3)=-4.*X(1)-2.*X(3)	150
XP(4)=-4.*X(2)-2.*X(4)-9.81	152
RETURN	196.1
END	198.1
SUBROUTINE OUTF(T,X,XP,IH2,KDU,PRMT)	200.1
DIMENSION X(4),XP(4),PRMT(5)	210
COMMON TPR	220.1
IF(ABS(T-TPR)-0.0001) 20,20,40	230.1
20 PRINT 25,T,(X(I),I=1,4)	240
25 FORMAT(2X,5(2X,G12.5))	241
TPR=TPR+0.1	250.1
40 RETURN	260.1
END	262.1

Оператор 20 задает описание массивов PRMT, X, XP и AUX, содержащих соответственно 5, 4, 4 и 32 элемента.

Оператор 45 задает начальные значения при времени $T=0$ переменным X и Z, обозначенным в программе X(1) и X(2), и их производным X(3) и X(4): X(1)=0, X(2)=10, X(3)=20, X(4)=0.

Оператор 50 присваивает начальные значения весовым коэффициентам погрешности в массиве производных XP, которые в сумме должны быть равны единице (в нашем случае все весовые коэффициенты равны $XP(1)=XP(2)=XP(3)=XP(4)=0.25$).

Оператор 60.1 под управлением оператора 61 выводит на печать заголовок таблицы результатов решения, рассчитанной так, чтобы значения соответствующих величин располагались под их обозначениями. В одной строке соответственно с 4-й, 21-й, 35-й, 49-й и 62-й позиции будет напечатано:

ВРЕМЯ, СЕК X, М Z, М XP, М/С ZP, М/С

Оператор 65 осуществляет обращение к подпрограмме RKGS с фактическими параметрами, совпадающими с параметрами соответствующего оператора программы 19.1 за исключением: 4 — количество дифференциальных уравнений первого порядка KDU для плоской задачи; AUX — рабочий массив размером 8S4.

Обратим ваше внимание, что хотя в операторе 65 имена внешних подпрограмм совпадают, сами же эти подпрограммы составлены для случая плоской задачи и могут существенно отличаться от используемых в программе 19.1 (особенно SDUIP).

В результате выполнения подпрограммы RKG5 будут вычислены и выведены на печать значения аргумента T и соответствующие ему значения координат материальной точки X(1), X(2) и проекций ее скорости X(3), X(4) в точках 0., 0.1, 0.2, 0.3, ..., 1.2 (вывод на печать обеспечивается подпрограммой OUPP).

Оператор 110 задает описание формальных массивов X(4) и XP(4) в подпрограмме SDUIP. Для простоты они указаны постоянного размера, равного количеству дифференциальных уравнений первого порядка KDU (см. также дополнение 19.1).

Операторы 140-152 представляют систему из двух интегрируемых дифференциальных уравнений второго порядка (20.1) в виде системы четырех дифференциальных уравнений первого порядка (см. п. 18.3.2).

Оператор 210 задает описание формальных массивов X(4), XP(4), PRMT(5) в подпрограмме OUPP.

Оператор 240 (с меткой 20) под управлением оператора 241 (с меткой 25) выводит на печать значения аргумента T, координат материальной точки X(1), X(2) и проекций ее скорости X(3) и X(4) в каждой точке T=TPR, при этом используется форма оператора вывода со встроенным циклом (см. пример (7.38)).

Отметим, что с программой 20.1 могут работать все дополнения 19.1-19.11, в чем мы предлагаем убедиться самостоятельно (с учетом в дополнении 19.5 особенностей двумерного характера задачи, описанных в дополнении 20.1). Если возникнут трудности при использовании программы 20.1 для численного интегрирования линейных задач, то следует обратиться к дополнению 20.2.

Дополнение 20.1. Определение относительной ошибки при сравнении численного и аналитического решений для двумерных задач. Для этого нужно в базовой программе 20.1 для рассматриваемого примера (20.1) внести следующие изменения:

```

5  FORMAT (T4, 'ВРЕМЯ, СЕК', T21, 'X, M', T35,           61
   * ' ТОЧНОЕ XT', T49, 'ORASX, %', T64, 'Z, M',
   * T76, ' ТОЧНОЕ ZT', T92, 'ORASZ, %' )
20 ST=SQRT (3.) * T                                     231
   XT=11.56*EXP (-T) * SIN (ST)                        232
   ZT=12.45*EXP (-T) * (COS (ST)+0.577*SIN (ST)) -2.45 233
   RASX=X (1) -XT                                       234
   RASZ=X (2) -ZT                                       235
   IF (ABS (XT) .LE.1.E-33) XT=1.E-33                 236
   IF (ABS (ZT) .LE.1.E-33) ZT=1.E-33                 237
   ORASX= (RASX/XT) *100.                               238
   ORASZ= (RASZ/ZT) *100.                               239
   PRINT 25, T, X (1), XT, ORASX, X (2), ZT, ORASZ     240
25  FORMAT (2X, 7 (2X, G12.5) )                         241

```


Оператором основной программы 60.1 под управлением замененного оператора 61 (с меткой 5) в заглавии таблицы результатов решения дополнительно к времени и координате будет напечатано: с 35-й позиции “ТОЧНОЕ ХТ”, с 49-й — “ORASX, %”, с 64-й — “Z, M”, с 76-й — “ТОЧНОЕ ZT” и с 92-й позиции “ORASZ, %”. Расположение этих обозначений предварительно рассчитывается так, чтобы под ними печатались соответствующие величины группой операторов 240-241.

Следующая группа операторов 231-239 вставляется после оператора 230 в подпрограмме OUPR. Из них оператор 231 является вспомогательным. Он вычисляет произведение $\text{SQRT}(3) * T$, которое в двух нижеследующих операторах встречается 3 раза.

Операторы 232 и 233 записывают точное решение рассматриваемого примера [21, с. 138] в виде уравнений движения материальной точки, присваивая им обозначения ХТ и ZТ. Напомним также, что эпитет “точное” следует понимать в смысле “аналитическое”. В нем также могут быть допущены ошибки.

Операторы 234 и 235 определяют разности RASX и RASZ между численным и аналитическим решением для координат X и Z.

Два условных логических оператора 236 и 237 выполняют защиту от переполнения порядка. Если значение координат ХТ или ZТ по какой-либо причине окажется меньше $1.E-33$, то этим переменным будет присвоено значение $1.E-33$. Оно достаточно мало, чтобы не исказить результаты дальнейшего решения, а обратная к ней величина гораздо меньше уровня переполнения (около $10 * 75$) и не вызовет соответствующей ошибки в операторах 238 и 239.

Операторы 238 и 239 определяют относительные ошибки численного решения по отношению к аналитическому, обозначаемые соответственно ORASX, ORASZ и выраженные в процентах, для каждого выводимого на печать значения времени T.

Оператор 240 под управлением оператора 241 (с меткой 25) выводит на печать значения аргумента T, численно и аналитически определенных координат $X=X(1)$ и ХТ, $Z=X(2)$ и ZТ, а также относительные ошибки их определения ORASX и ORASZ, выраженные в процентах.

Рассмотренный фрагмент дополняет работу базовой программы 20.1 сравнением результатов численного расчета с аналитическим решением. Все остальные неуказанные во фрагменте операторы базовой программы как обычно остаются на своих местах без изменения, выполняя те же функции.

Дополнение 20.2. Использование программы 20.1 для численного интегрирования линейных задач.

Если задача поставлена в более общей двумерной постановке, то не мешает ее проверить на частном одномерном случае. Отметим одну общую закономер-

ность: любая одномерная задача в двумерной постановке при соответствующем выборе осей будет иметь одно дифференциальное уравнение движения 2-го порядка, равное нулю, с соответствующими нулевыми начальными условиями.

При представлении одномерного дифференциального уравнения 2-го порядка в виде системы дифференциальных уравнений 1-го порядка для двумерной постановки также будет только одно нулевое дифференциальное уравнение (а не два, как можно сразу подумать, раз из одного дифференциального уравнения 2-го порядка получается 2 дифференциальных уравнений 1-го порядка).

Это объясняется тем, что первые два дифференциальных уравнения 1-го порядка устанавливают просто соотношения между производными и не зависят от их конкретных значений (см. уравнения (18.10) и (18.11)). Таким образом, для численного интегрирования примера 16.3 (одномерная задача), использовавшегося в программе 19.1, в двумерной постановке в программу 20.1 следует внести следующие необходимые изменения:

DATA X/0.3, 0., 2., 0. /	45
XP (3) = -97.533 * X (1) + 12.491	150
XP (4) = 0.	152

Оператор 45 задает начальные условия для координаты X и проекции скорости на эту координату, которые для двумерной постановки представляют собой соответственно 1-й и 3-й элементы массива X: X(1)=0.3, X(3)=2. Соответствующие же величины для координаты Y равны нулю: X(2)=0, X(4)=0.

Операторы 150-152 задают формулы для вычисления правой части дифференциального уравнения (16.3) в двумерной постановке для работы по программе 20.1. Здесь для простоты использована числовая форма представления правой части оператора 150 по дополнению 19.4.

20.2. Трехмерные (пространственные) задачи динамики

Численное интегрирование систем из трех дифференциальных уравнений второго порядка рассмотрим на примере 28-го варианта задания Д-2 [21, с. 133, 135]. Интегрируемая система после деления на массу точки и выделения в левой части второй производной примет вид:

$$\begin{aligned} d(dX/dT) / dT &= -0.5 * dX/dT, \\ d(dY/dT) / dT &= -0.5 * dY/dT, \\ d(dZ/dT) / dT &= -0.5 * dZ/dT + 9.81, \end{aligned} \quad (20.2)$$

Ее начальные условия при T=0 равны [21, с. 135]: для координат точки X0=0, Y0=0, Z0=0 (м), для скоростей XP0=1, YP0=1, ZP0=2 (м/с).

Программа 20.2 предназначена для численного интегрирования трех дифференциальных уравнений 2-го порядка (20.2) с использованием подпрограммы RKGS. Она также представлена полностью и операторы, совпадающие с программой 19.1, не описываются (имеют после своего номера точку и цифру 1):

```

C      П Р О Г Р А М М А 20.2
      EXTERNAL SDU1P,OUTP                               10.1
      DIMENSION PRMT(5),X(6),XP(6),AUX(8,6)            20
      COMMON TPR                                       30.1
      DATA PRMT/0.,1.2,0.01,0.001,0./                40.1
      DATA X/0.,0.,0.,1.,1.,2./                      45
      DATA XP/6*0.166666/                             50
      TPR=0.                                           54.1
      PRINT 5                                           60.1
5  FORMAT(T4,'ВРЕМЯ, СЕК',T21,'X, M',T35,'Y, M',
*  T49,'Z, M',T62,'XP, M/C',T76,'YP, M/C',T90,'ZP, M/C')
      CALL RKGS(PRMT,X,XP,6,IH2,SDU1P,OUTP,AUX)        65
      STOP                                             98.1
      END                                             99.1
      SUBROUTINE SDU1P(T,X,XP)                          100.1
      DIMENSION X(6),XP(6)                             110
      XP(1)=X(4)                                        140
      XP(2)=X(5)                                        142
      XP(3)=X(6)                                        144
      XP(4)=-0.5*X(4)                                  150
      XP(5)=-0.5*X(5)                                  152
      XP(6)=-0.5*X(6)+9.81                             154
      RETURN                                           196.1
      END                                             198.1
      SUBROUTINE OUTP(T,X,XP,IH2,KDU,PRMT)              200.1
      DIMENSION X(6),XP(6),PRMT(5)                    210
      COMMON TPR                                       220.1
      IF(ABS(T-TPR)-0.0001) 20,20,40                  230.1
20 PRINT 25,T,(X(I),I=1,6)                             240
25 FORMAT(2X,7(2X,G12.5))                              241
      TPR=TPR+0.1                                       250.1
40 RETURN                                             260.1
      END                                             262.1

```

Оператор 20 задает описание массивов PRMT, X, XP и AUX, содержащих соответственно 5, 6, 6 и 48 элементов.

Оператор 45 задает начальные значения при времени $T=0$ переменным X, Y и Z, обозначенным в программе X(1), X(2) и X(3), и их производным соответственно X(4), X(5) и X(6): X(1)=0, X(2)=0, X(3)=0, X(4)=1, X(5)=1, X(6)=2.

Оператор 50 присваивает начальные значения весовым коэффициентам погрешности в массиве производных XP, которые в сумме должны

быть равны единице (в нашем случае все весовые коэффициенты равны $XP(1) = XP(2) = XP(3) = XP(4) = XP(5) = XP(6) = 0.166666$).

Оператор 60.1 под управлением оператора 61 выводит на печать заголовок таблицы результатов решения. В одной строке соответственно с 4-й, 18-й, 31-й, 43-й, 55-й, 69-й и 83-й позиции будет напечатано (вследствие несовпадения длин строк пособия и распечатки нижеследующий заголовок таблицы сжат):

ВРЕМЯ, СЕК X, М Y, М Z, М XP, М/С YP, М/С ZP, М/С

Оператор 65 осуществляет обращение к подпрограмме RKGS с фактическими параметрами, совпадающими с параметрами соответствующего оператора программы 19.1 за исключением: 6 — количество дифференциальных уравнений первого порядка KDU для пространственной задачи; AUX — рабочий массив размером 8S6.

В результате выполнения подпрограммы RKGS будут вычислены и напечатаны значения аргумента T, координат материальной точки X(1), X(2), X(3) и проекций ее скорости X(4), X(5), X(6) в точках 0, 0.1, 0.2, 0.3, ..., 1.2 (вывод на печать обеспечивается подпрограммой OUPP).

Оператор 110 задает описание формальных массивов X(6) и XP(6) в подпрограмме SDUIP.

Операторы 140-154 представляют систему из трех интегрируемых дифференциальных уравнений второго порядка (20.2) в виде системы шести дифференциальных уравнений первого порядка (см. п. 18.3.3).

Оператор 210 задает описание формальных массивов X(6), XP(6) и PRMT(5) в подпрограмме OUPP. Их также можно задать с регулируемыми размерами для X и XP (см. дополнение 19.1).

Оператор 240 (с меткой 20) под управлением оператора 241 (с меткой 25) выводит на печать значения аргумента T, координат материальной точки X(1), X(2), X(3) и проекций ее скорости X(4), X(5) и X(6) в каждой точке $T=TPR$, при этом используется форма оператора вывода со встроенным циклом (см. пример (7.38)).

Отметим, что с программой 20.2 могут работать все 12 дополнений, описанных в пп. 19 и 20.1, в чем мы предлагаем убедиться самостоятельно (с учетом в дополнениях 19.5 и 20.1 особенностей трехмерного характера задачи).

Программа 20.2 представляет собой самый общий случай базовой программы. Покажем, как можно использовать ее для решения плоских (дополнение 20.3) и линейных (дополнение 20.4) задач на типовых примерах программ 20.1 и 19.1 соответственно.

Кроме естественного изменения соответствующих правых частей дифференциальных уравнений и начальных условий, а также изменения их обозначений в программе (связанной с увеличением размеров используемых массивов)

вов), для плоской задачи всегда будет одно недостающее дифференциальное уравнение равняться нулю ($XP(5)=0$ или $XP(6)=0$ в зависимости от обозначений), а для линейной задачи — два: $XP(5)=0$., $XP(6)=0$.

Дополнение 20.3. Использование программы 20.2 для численного интегрирования плоских задач. С этой целью в программу 20.2 для интегрирования двумерного примера программы 20.1 следует внести следующие изменения:

DATA X/0.,0.,10.,20.,0.,0./	45
XP(4)=-4.*X(1)-2.*X(4)	150
XP(5)=0.	152
XP(6)=-4.*X(3)-2.*X(6)-9.81	154

Оператор 45 используется для задания начальных значений при времени $T=0$ переменным X , Y и Z , носящим в программе 20.2 обозначения $X(1)$, $X(2)$ и $X(3)$, и их производным соответственно $X(4)$, $X(5)$ и $X(6)$. В результате они получают в момент загрузки программы следующие значения: $X(1)=0$, $X(2)=0$, $X(3)=10$, $X(4)=20$, $X(5)=0$, $X(6)=0$.

Операторы 150-154 задают формулы для вычисления правых частей системы дифференциальных уравнений (20.1), рассматривавшихся в качестве плоской задачи в программе 20.1, для численного интегрирования в трехмерной постановке в программе 20.2.

Дополнение 20.4. Использование программы 20.2 для численного интегрирования линейных задач. С этой целью в программу 20.2 для интегрирования одномерного примера программы 19.1 следует внести следующие изменения:

DATA X/0.3,0.,0.,2.,0.,0./	45
XP(4)=-97.533*X(1)+12.491	150
XP(5)=0.	152
XP(6)=0.	154

Оператор 45 используется для задания начальных значений переменной X , обозначенной в программе 20.2 также $X(1)$, и ее производной $X(4)$ при времени $T=0$. Соответствующие величин для координат Y ($X(2)$ и $X(5)$) и Z ($X(3)$ и $X(6)$) равняются нулю.

Операторы 150-154 задают формулы для вычисления правых частей системы дифференциальных уравнений (16.3), рассматривавшихся в качестве линейной задачи в программе 19.1, для численного интегрирования в трехмерной постановке в программе 20.2. Заметим, что пример одномерной задачи (16.3) рассматривался в двумерной постановке по программе 20.1 в дополнении 20.2.

Конечно, результаты работы программы 19.1, программы 20.1 с дополнением 20.2 и программы 20.2 с дополнением 20.4 будут одинаковыми, в чем желательно убедиться самостоятельно получением соответствующих распечаток сначала для рассматриваемого примера, а затем и для ваших задач.

ГЛАВА 21. ВЫЧИСЛЕНИЯ С УДВОЕННОЙ ТОЧНОСТЬЮ

В п. 15 была показана важность умения вычислений с удвоенной точностью и приведены необходимые для этого сведения. С учетом описанных в п. 15.3 требований, можно достаточно уверенно переделывать программы из обычной точности в удвоенную, что мы для приобретения навыка кратко и рассмотрим на примерах численного интегрирования одномерных (программа 19.1), двумерных (20.1) и трехмерных (программа 20.2) задач динамики.

Для краткости будут приводиться только изменяемые или дополняемые в соответствующие программы операторы. Надеемся, что вы не только с пониманием восстановите (по номерам операторов) и проверите работу этих программ с использованием удвоенной точности, но сделаете это и для приведенных в пособии дополнений и ваших задач.

1. Для вычислений с удвоенной точностью в программу 19.1 следует внести следующие изменения и дополнения:

IMPLICIT REAL *8 (A-H, O-Z)	9
DIMENSION PRMT(5), X(2), XP(2), AUX(16, 2)	20
DATA PRMT/0.D0, 1.2D0, 0.01D0, 0.001D0, 0.D0/	40
DATA X/0.3D0, 2.D0/	45
DATA XP/2*0.5D0/	50
TPR=0.D0	54
CALL DRKGS (PRMT, X, XP, 2, IH2, SDU1P, OUTP, AUX)	65
IMPLICIT REAL *8 (A-H, O-Z)	109
DATA PI/3.141592653589D0/, ZM/0.01D0/, R/0.2D0/,	120
* C/1.D0/, ZL0/0.2D0/	
AL=PI/6.D0	130
C1=-(C/ZM-PI*PI*(DSIN(AL))**2)	134
C5=PI*PI*R*DSIN(AL)-9.81D0*DCOS(AL)+C*ZL0/ZM	136
IMPLICIT REAL *8 (A-H, O-Z)	209
IF (DABS (T-TPR)-0.0001D0) 20, 20, 40	230
TPR=TPR+0.1D0	250

Неявный оператор типа IMPLICIT 9 описывает все величины основной программы, начинающиеся с букв от А до Н и от О до Z, в удвоенной точности. Он встречается еще дважды под номерами 109 и 209, описывая в подпрограммах SDU1P и OUTP все величины в удвоенной точности.

В операторе 20, задающем описание массивов, изменяется только размер рабочего массива AUX, который для работы подпрограммы удвоенной

точности DRKGS с системой двух дифференциальных уравнений первого порядка должен содержать $16S2=32$ элемента.

В операторах 40, 45, 50 и 54 значения описанных в программе 19.1 величин задаются в удвоенной точности.

В операторе 65, осуществляющем обращение к подпрограмме удвоенной точности DRKGS, изменяется только ее имя, однако все фактические параметры должны быть представлены с учетом требований п. 15.3 для использования удвоенной точности.

В операторах 120, 130, 134-136, 230 и 250 значения описанных в программе 19.1 величин также задаются в удвоенной точности и используются функции DSIN, DCOS и DABS удвоенной точности.

2. Для вычислений с удвоенной точностью в программу 20.1 следует внести следующие изменения и дополнения:

```

IMPLICIT REAL *8 (A-H, O-Z)                9.1D
DIMENSION PRMT(5), X(4), XP(4), AUX(16,4)  20
DATA PRMT/0.D0, 1.2D0, 0.01D0, 0.001D0, 0.D0/ 40.1D
DATA X/0.D0, 10.D0, 20.D0, 0.D0/          45
DATA XP/4*0.25D0/                          50
TPR=0.D0                                     54.1D
CALL DRKGS (PRMT, X, XP, 4, IH2, SDU1P, OUTP, AUX) 65
IMPLICIT REAL *8 (A-H, O-Z)                109.1D
XP(3)=-4.D0*X(1)-2.D0*X(3)                 150
XP(4)=-4.D0*X(2)-2.D0*X(4)-9.81D0         152
IMPLICIT REAL *8 (A-H, O-Z)                209.1D
IF (DABS (T-TPR) -0.0001D0) 20,20,40      230.1D
TPR=TPR+0.1D0                              250.1D

```

Операторы, полностью совпадающие с соответствующими операторами фрагмента п. 21.1, здесь и в п. 21.3 повторно не описываются. Для отличия они имеют после своего номера точку, цифру 1 и букву D и представлены вследствие своего отличия от аналогичных операторов переделываемых программ 20.1 и 20.2.

В операторе 20, задающем описание массивов, изменяется по отношению к программе 20.1 только размер рабочего массива AUX, который для работы подпрограммы DRKGS с системой 4-х уравнений 1-го порядка должен содержать $16S4=64$ элемента.

В операторах 45 и 50 значения описанных в программе 20.1 величин задаются в удвоенной точности.

Оператор 65 осуществляет обращение к подпрограмме DRKGS с фактическими параметрами, совпадающими с параметрами программы 20.1, но которые должны быть представлены с учетом требований для использования удвоенной точности.

Операторы 150-152 задают формулы для вычисления правых частей системы дифференциальных уравнений (20.1) с использованием числовых величин удвоенной точности.

3. Для вычислений с удвоенной точностью в программу 20.2 следует внести следующие изменения и дополнения:

IMPLICIT REAL *8 (A-H, O-Z)	9.1D
DIMENSION PRMT (5), X (6), XP (6), AUX (16, 6)	20
DATA PRMT/0.D0, 1.2D0, 0.01D0, 0.001D0, 0.D0/	40.1D
DATA X/0.D0, 0.D0, 0.D0, 1.D0, 1.D0, 2.D0/	45
DATA XP/6*0.166666D0/	50
TPR=0.D0	54.1D
CALL DRKGS (PRMT, X, XP, 6, IH2, SDU1P, OUTP, AUX)	65
IMPLICIT REAL *8 (A-H, O-Z)	109.1D
XP (4)=-0.5D0*X (4)	150
XP (5)=-0.5D0*X (5)	152
XP (6)=-0.5D0*X (6)+9.81D0	154
IMPLICIT REAL *8 (A-H, O-Z)	209.1D
IF (DABS (T-TPR)-0.0001D0) 20, 20, 40	230.1D
TPR=TPR+0.1D0	250.1D

В операторе 20, задающем описание массивов, изменяется по отношению к программе 20.2 только размер рабочего массива AUX, который для работы подпрограммы DRKGS с системой шести дифференциальных уравнений первого порядка должен содержать 16S6=96 элементов.

В операторах 45 и 50 значения описанных в программе 20.2 величин задаются в удвоенной точности.

Оператор 65 осуществляет обращение к подпрограмме DRKGS с фактическими параметрами, совпадающими с параметрами программы 20.2, но которые должны быть представлены с учетом требований для использования удвоенной точности.

Операторы 150-154 задают формулы для вычисления правых частей системы дифференциальных уравнений (20.2) с использованием числовых величин удвоенной точности.

Дополнение 21.1. Использование стандартной подпрограммы удвоенной точности DHPCG для численного интегрирования задач динамики.

Из всех трех рассмотренных в п. 19.2 стандартных подпрограмм (HPCG, OMAD и MAD3), реализующих многошаговые методы, в удвоенной точности существует только одна стандартная подпрограмма DHPCG, реализующая модифицированный предиктор-корректор метод Хемминга. Для ее практического использования в программах 19.1-20.2, соответственно преобразованных по пп. 21.1-21.3 для работы в удвоенной точности, следует заменить следующие два оператора, номера которых совпадают с указанными ниже:

- для программы 19.1:

DIMENSION PRMT (5) , X (2) , XP (2) , AUX (32 , 2)	20
CALL DHPCG (PRMT , X , XP , 2 , IH2 , SDU1P , OUTF , AUX)	65

- для программы 20.1:

DIMENSION PRMT (5) , X (4) , XP (4) , AUX (32 , 4)	20
CALL DHPCG (PRMT , X , XP , 4 , IH2 , SDU1P , OUTF , AUX)	65

- для программы 20.2:

DIMENSION PRMT (5) , X (6) , XP (6) , AUX (32 , 6)	20
CALL DHPCG (PRMT , X , XP , 6 , IH2 , SDU1P , OUTF , AUX)	65

Отметим, что при работе с подпрограммой DHPCG сохраняется возможность использования любых из рассмотренных дополнений.

Надеемся, что вы проверите все вышеописанные варианты и дополнения, переделав их самостоятельно в удвоенную точность. Для ответа на возможные вопросы, которые могут появиться в процессе этой работы, в приложении 4 приведены все основные программы 19.1, 20.1–20.2 в удвоенной точности. Рекомендуем обращаться к ним только после возникновения трудностей, которые вы не сможете преодолеть самостоятельно.

Напомним, что при выполнении всех описываемых в пособии программ на персональных ЭВМ следует учесть в любой удобной для вас форме материал п. 7.6.3.

ГЛАВА 22. ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ ЗАДАЧ ДИНАМИКИ С ИСПОЛЬЗОВАНИЕМ ПОДПРОГРАММЫ RKF45. МАТЕРИАЛЫ ДЛЯ УИРС И НИРС

В работе [27, с. 146-164] представлена более совершенная подпрограмма RKF45, использующая также метод Рунге-Кутты, как и RKGS. После исправления опечаток, препятствующих ее использованию, она переделана нами в удвоенную точность и представлена в виде подпрограммы DRKF45 в приложении 2.

Подпрограмма RKF45 обладает более широкими возможностями и является более совершенной, чем описанные в п. 18.4 стандартные подпрограммы, что обуславливает ее некоторую сложность. Поэтому изучение работы с ней предполагается в виде дополнительной индивидуальной работы в рамках УИРС или НИРС.

22.1. Подпрограмма RKF45

Подпрограмма RKF45 требует шести вычислений функции за шаг. Четыре из этих значений берутся с одним набором коэффициентов, приводя к методу четвертого порядка, и все шесть значений комбинируются с другими коэффициентами, что дает метод пятого порядка. Сравнение двух вычисленных величин позволяет получить оценку ошибки, которая используется для контроля длины шага.

Обращение к подпрограмме RKF45 имеет вид:

```
CALL RKF45 (SDU1P, KDU, X, T, TOUT, GOPX, GPX, IFLAG, WORK, IWORK)
```

Параметры этой подпрограммы имеют следующий смысл:

SDU1P — имя внешней подпрограммы, которая задает систему дифференциальных уравнений 1-го порядка (см. п. 18.3). Список ее формальных параметров должен включать T, X, XP и она полностью совпадает с использовавшимися в пп. 19-20.2 подпрограммами SDU1P для соответствующих типовых примеров;

KDU — число интегрируемых дифференциальных уравнений первого порядка (2, 4, или 6 соответственно для линейной, плоской или пространственной задачи динамики материальной точки);

X — входной массив начальных значений (X10, X20, ..., XN0). В дальнейшем X становится результирующим вектором-решения системы (X1, X2, ..., XN) в соответствующих точках отрезка [A, B]. Полностью совпадает с использовавшимися везде ранее одноименным массивом X(KDU);

T — независимая переменная, задание которой в качестве параметра не требуют все применяемые ранее стандартные подпрограммы;

TOUT — точка выхода, в которой нужно определить значение решения. $T=TOUT$ возможно лишь при первом обращении к подпрограмме RKF45. В этом случае выход из нее происходит при нормальном значении для параметра $IFLAG=2$, если можно продолжать интегрирование (см. ниже описание параметра $IFLAG$);

GOPX и GPX — границы для относительной и абсолютной локальных погрешностей. Эти границы должны быть неотрицательны. Подпрограмме RKF45, вообще говоря, не следует задавать границу для относительной ошибки, меньшую, чем примерно $1.E-8$, чтобы избежать трудностей, связанных с очень высокими запросами к точности. Не разрешается задание только абсолютной ошибки GPX. Если же задано значение относительной ошибки GOPX меньше допустимого, то подпрограмма RKF45 увеличивает значение GOPX надлежащим образом и возвращает управление пользователю, прежде чем продолжать интегрирование.

Параметр $IFLAG$ является важной контрольной переменной. При первом обращении к RKF45 ему следует присвоить значение 1. Обычно RKF45 изменяет его значение на 2, и при последующих обращениях нужно сохранить это значение. Значения, не равные 2, полученные на выходе RKF45, сигнализируют о наличии различных нерегулярностей или ошибок, подробно описываемых в комментариях к распечатке подпрограммы DRKF45 в приложении 2.

$IFLAG=3$ указывает, что затребована слишком высокая относительная точность GOPX.

$IFLAG=4$ или $IFLAG=7$ — это предупреждения, что подпрограмме будет очень трудно получить требуемую точность. Пользователь может продолжать процесс, либо увеличить границы погрешностей, либо перейти к подпрограмме многошагового метода (см. п. 18.4.2).

$IFLAG=5$ или $IFLAG=6$ означают, что прежде, чем продолжать, нужно изменить допуски на ошибку.

$IFLAG=8$ — указание на неправильность вызова подпрограммы RKF45.

Мы настоятельно советуем пользователю включить в свою основную программу проверку параметра $IFLAG$ (см. дополнение 22.1).

WORK — рабочий массив, содержащий информацию, внутреннюю для подпрограммы RKF45, которая необходима при ее последующих вызовах для других значений TOUT из интервала интегрирования. Его размер должен быть не меньше $3+6*KDU$ элементов, что составит: для одномерных, двумерных и трехмерных задач соответственно 15, 27 и 39 элементов.

IWORK — целочисленный рабочий массив. Его размер должен быть не меньше 5 и может не меняться в зависимости от типа задачи.

Обратим ваше внимание, что в отличие от изучавшихся в пп. 19-20.2 стандартных подпрограмм, RKF45 не требует дополнительной внешней

подпрограммы, которая ведет обработку ее выходных величин: это может быть сделано в самой основной программе сразу после обращения к RKF45. Однако для создания большего подобия с работой вышеописанных программ 19.1-20.2 и с учетом удобств применения ранее описанных дополнений, соответствующий фрагмент вывода результатов расчета на печать оформлен также в виде подпрограммы OUTF.

В качестве примеров для численного интегрирования одномерных, двумерных и трехмерных задач динамики с использованием подпрограммы RKF45 выбраны те же условия типовых примеров, которые были соответственно взяты и для программ 19.1-20.2.

22.2. Численное интегрирование одномерных задач динамики с использованием подпрограммы RKF45

Ниже приводится программа 22.1 для численного интегрирования одного дифференциального уравнения 2-го порядка (16.3) с использованием подпрограммы RKF45. Она выполняет ту же задачу, что и программа 19.1, но вследствие использования подпрограммы RKF45 имеет довольно большие отличия. Поэтому для удобства пользования она представлена полностью. Однако операторы, совпадающие с программой 19.1, не описываются. Для отличия они имеют после своего номера точку и цифру 1:

```

С      ПРОГРАММА 22.1
С      (ПРОГРАММА 19.1 С ИСПОЛЬЗОВАНИЕМ ПОДПРОГРАММЫ RKF45)
      EXTERNAL SDU1P                10
      DIMENSION X(2), IWORK(5), WORK(15)      20
      DATA T, TK/0., 1.2/, GPX, GOPX/0., 1.E-8/  40
      DATA X/0.3, 2./              45.1
      KDU=2                          50
      IFLAG=1                        52
      HPR=0.1                        54
      TOUT=T                          56
      PRINT 5                        60.1
5     FORMAT (T4, 'ВРЕМЯ, СЕК', T16, ' КООРДИНАТА X, М',
*          T32, ' СКОРОСТЬ XP, М/С' )      61.1
10    CALL RKF45 (SDU1P, KDU, X, T, TOUT, GOPX, GPX, IFLAG,
*              WORK, IWORK)              65
      CALL OUTF (T, X)                 67
      TOUT=T+HPR                      74
      IF (T.LT.TK) GOTO 10             75
      STOP                             98.1
      END                              99.1

```

SUBROUTINE SDU1P(T,X,XP)	100.1
DIMENSION X(2),XP(2)	110.1
DATA PI/3.141592653589/, ZM/0.01/, R/0.2/,	120.1
* C/1./, ZL0/0.2/	
AL=PI/6.	130.1
C1=- (C/ZM-PI*PI*(SIN(AL))**2)	134.1
C5=PI*PI*R*SIN(AL)-9.81*COS(AL)+C*ZL0/ZM	136.1
XP(1)=X(2)	140.1
XP(2)=C1*X(1)+C5	150.1
RETURN	196.1
END	198.1
SUBROUTINE OUTF(T,X)	200
DIMENSION X(2)	210
PRINT 25,T,X(1),X(2)	240.1
25 FORMAT(2X,3(2X,G12.5))	241.1
RETURN	260.1
END	262.1

Оператор 10 объявляет имя внешней подпрограммы SDU1P, являющейся одним из фактических параметров при обращении к подпрограмме RKF45.

Оператор 20 задает описание массивов X, IWORK, WORK, содержащих соответственно 2, 5 и 15 элементов.

Оператор 40 DATA используется для задания начальных значений нижней T и верхней TK границы интервала интегрирования, а также величин абсолютной GPX и относительной GOPX допустимых погрешностей вычислений, которым в момент загрузки программы присваивается: T=0, TK=1.2, GPX=0, GOPX=1.E-8.

Оператор 50 задает целой переменной KDU значение количества дифференциальных уравнений 1-го порядка (для одномерной задачи равно двум), а оператор 52 присваивает начальное значение параметру IFLAG, которое при первом обращении к RKF45 должно быть равно единице.

Оператор 54 задает значение шага печати результатов HPR=0.1, а оператор 56 присваивает переменной TOUT значение независимой переменной T, что возможно лишь при первом обращении к RKF45.

Оператор 65 осуществляет обращение к подпрограмме RKF45 с указанными фактическими параметрами. Так как их обозначения совпадают с использованными в п. 22.1 формальными параметрами (SDU1P, KDU, X, T, TOUT, GOPX, GPX, IFLAG, WORK, IWORK), то согласно принятому нами соглашению они имеют тот же смысл.

Подпрограмма RKF45 в конце своей работы приравнивает значение точки выхода TOUT текущему аргументу T (TOUT=T). После первого обращения к RKF45 это все еще будет нулевое значение.

Оператор 67 осуществляет обращение к подпрограмме OUTF, которая выводит на печать значения текущего аргумента T, координаты материальной точки X(1) и ее скорости X(2).

Арифметический оператор присваивания 74 увеличивает значение TOUT на величину шага печати результатов HPR по сравнению со значением текущей переменной T. Теперь значение следующей точки выхода TOUT, в которой нужно найти решение для элементов массива X, равняется 0.1.

Условный логический оператор 75 проверяет на истинность выражение, заключенное в скобки. Так как значение текущей переменной T после первого обращения к подпрограмме RKF45 равно нулю, а $TK=1.2$, то (T.LT.TK) и вследствие истинности этого логического выражения управление передается оператору с меткой 10, осуществляющему повторное обращение к подпрограмме RKF45. Значения фактических параметров, определенные на выходе из этой подпрограммы, будут являться входными параметрами при следующем обращении к ней.

Теперь в результате повторного выполнения подпрограммы RKF45 будут вычислены значения координаты материальной точки X(1) и ее скорости X(2) в точке, равной значению $TOUT=0.1$. Подпрограмма RKF45 в конце своей работы опять приравняет значение точки выхода TOUT текущему аргументу T (см. оператор с меткой 300 в распечатке подпрограммы DRKF45 в приложении 2). После второго обращения к RKF45 $T=0.1$.

Далее после вызова оператором 67 подпрограммы печати результатов OUTF будет распечатано следующее значение текущего аргумента $T=0.1$ и соответствующих ему значений координаты материальной точки X(1) и ее скорости X(2).

Арифметический оператор присваивания 74 опять увеличивает значение TOUT на величину шага печати результатов $HPR=0.1$ по сравнению со значением текущей переменной T. Теперь значение следующей точки выхода, в которой нужно найти решение для элементов массива X равняется 0.2.

Условный логический оператор 75 опять проверяет на истинность выражение, заключенное в скобки. Так как значение текущей переменной T после второго обращения к подпрограмме RKF45 равно 0.1, а $TK=1.2$, то (T.LT.TK) и вследствие истинности этого логического выражения управление передается оператору с меткой 10, осуществляющему повторное обращение к подпрограмме RKF45. Значения фактических параметров, определенные на выходе из этой подпрограммы, теперь также будут являться входными параметрами при следующем обращении к ней.

Так опять будет найдено решение для координаты материальной точки X(1) и ее скорости X(2) при $TOUT=0.2$, подпрограмма RKF45 приравняет текущее значение $T=TOUT=0.2$ и по оператору 67 будут распечатаны новые значения при $T=0.2$ координаты X(1) и скорости X(2).

Далее оператор 74 опять увеличит значение TOUT на $HPR=0.1$ по сравнению с текущим $T=0.2$, в результате чего TOUT станет равным 0.3.

Условный логический оператор 75 вследствие истинности выражения, заключенного в скобки, опять передаст управление оператору с меткой 10, осуществляющему следующее обращение к подпрограмме RKF45 с новыми значениями фактических параметров для нахождения очередного решения.

В результате такого циклического повторения будут вычислены и выведены на печать значения аргумента T и соответствующие ему значения координаты материальной точки $X(1)$ и ее скорости $X(2)$ в следующих точках 0.3, 0.4, 0.5, 0.6, ..., 1.2 (вывод на печать обеспечивается подпрограммой OUTF).

При последнем повторении текущее значение аргумента T в условном логическом операторе 75 окажется вследствие ошибок усечения при представлении числа 0.1 в двоичной системе, используемой ЭВМ (а такое повторялось 12 раз), несколько меньшим, чем принятое нами конечное значение интервала интегрирования $TK=1.2$. В результате этих накопленных ошибок логическое выражение в скобках опять окажется истинным и вычисления будут повторены еще один лишний раз при $TOUT=1.3$.

После распечатывания результатов при текущем значении $T=1.3$, арифметический оператор присваивания 74 увеличивает значение TOUT опять на 0.1 и оно становится равным 1.4, что уже не является существенным. Только теперь наконец становится ложным значение логического выражения, заключенного в скобки в операторе 75, так как текущее значение $T=1.3$, что больше $TK=1.2$. Вследствие этого оператор 75 пропускается и выполняются следующие за ним операторы 98 и 99, указывающие на конец вычислений и окончание основной программы.

Напомним, что подпрограмма SDU1P, задающая интегрируемые уравнения второго порядка в виде системы дифференциальных уравнений первого порядка (см. п. 18.3) и вычисляющая правые части уравнений этой системы, полностью совпадает с использовавшимися в пп. 19-20.2 подпрограммами SDU1P для соответствующих типовых примеров. В данном случае все ее операторы 100-198 описаны в пояснениях к программе 19.1, на что указывает точка и цифра 1 после их номера.

Оператор 200 определяет, именуется подпрограмму OUTF и задает ее формальные параметры, из которых T — аргумент, а X — массив переменной и ее первой производной, т.е. вектор-решение, являющееся выходной характеристикой.

Обратим ваше внимание, что список формальных параметров подпрограммы OUTF может быть любым, то есть он может содержать любые нужные нам переменные или массивы, которые мы сочтем необходимым обработать или просто вывести на печать. В этом главное

ее отличие от подпрограммы OUTF, осуществляющей вывод на печать результатов решения при работе со стандартными подпрограммами, где соответствующий список строго определен (см. описание подпрограммы OUTF в п. 18.4.1).

Оператор 210 задает описание формального массива X, размер которого для линейной задачи равняется двум.

Нижеследующие операторы 240-262 также совпадают с описанными в пояснениях к программе 19.1 (только оператор 260 уже не нуждается в метке).

Заметим, что печать результатов решения можно организовать и в основной программе. Для этого операторы 240-241 следует расположить сразу после обращения к подпрограмме RKF45 по оператору 65, например под номерами 67 и 68:

```

PRINT 25, T, X(1), X(2)           67
25 FORMAT(2X, 3(2X, G12.5))      68

```

При этом из программы 22.1 следует удалить обращение к подпрограмме OUTF и операторы 200, 210, 260.1 и 262.1.

Дополнение 22.1. Организация контроля за значением параметра IFLAG. Приведем один из возможных вариантов включения в свою основную программу проверки параметра IFLAG, например, с помощью нижеследующего фрагмента изменений и дополнений к программе 22.1:

```

GOTO(80, 20, 30, 40, 50, 60, 70, 80), IFLAG           72
GOTO 80                                                73
20 TOUT=T+HPR                                         74
STOP                                                  76
30 PRINT 31, GOPX, GPX, IFLAG                          77
31 FORMAT(5X, 'ЗАТРЕБОВАНА СЛИШКОМ ВЫСОКАЯ ',       78
1 'ОТНОСИТЕЛЬНАЯ ТОЧНОСТЬ, '/2X, 'ВСЛЕДСТВИЕ ЧЕГО ',
2 'ПАРАМЕТР GOPX БЫЛ ИЗМЕНЕН НАДЛЕЖАЩИМ ДЛЯ'/
3 2X, 'ПРОДОЛЖЕНИЯ ИНТЕГРИРОВАНИЯ ОБРАЗОМ: '/
4 5X, 'GOPX=', G12.5, 2X, 'GPX=', G12.5, 2X, 'IFLAG=', I2)
GOTO 10                                               79
40 PRINT 41, GOPX, GPX, IFLAG                          80
41 FORMAT(5X, 'ИНТЕГРИРОВАНИЕ НЕ ВЫЛО ЗАКОНЧЕНО ',   81
1 'ЗА 3000 ВЫЧИСЛЕНИЙ ФУНКЦИИ (500 ШАГОВ) .' /
2 2X, 'МОЖНО УВЕЛИЧИТЬ ГРАНИЦЫ ПОГРЕШНОСТЕЙ ИЛИ ',
3 'ПРОДОЛЖАТЬ ПРОЦЕСС: ВУДУТ РАЗРЕШЕНЫ' /
4 2X, 'ЕЩЕ 3000 ВЫЧИСЛЕНИЙ ФУНКЦИИ.' /
5 5X, 'GOPX=', G12.5, 2X, 'GPX=', G12.5, 2X, 'IFLAG=', I2)
GOTO 10                                               82
50 GPX=1.0E-9                                         83
PRINT 51, GOPX, GPX, IFLAG                            84

```



```

51 FORMAT(5X,'РЕШЕНИЕ ОБРАТИЛОСЬ В НУЛЬ, ВСЛЕДСТВИЕ ', 85
1 'ЧЕГО ТЕСТ ТОЛЬКО ОТНОСИТЕЛЬНОЙ ОШИБКИ НЕ ПРОХОДИТ.' /
2 2X,'ДЛЯ ПРОДОЛЖЕНИЯ ИНТЕГРИРОВАНИЯ ЗНАЧЕНИЕ ',
3 'ПАРАМЕТРА GPX УСТАНОВЛЕНО НЕНУЛЕВЫМ И СОСТАВЛЯЕТ:' /
4 5X,'GORX=',G12.5,2X,'GPX=',G12.5,2X,'IFLAG=',I2)
   GOTO 10 86
60 GORX=10.0*GORX 87
   PRINT 61,GORX,GPX,IFLAG 88
61 FORMAT(5X,'ТРЕБУЕМАЯ ТОЧНОСТЬ НЕ МОГЛА БЫТЬ ', 89
1 'ДОСТИГНУТА ДАЖЕ ПРИ НАИМЕНЬШЕЙ ДОПУСТИМОЙ ВЕЛИЧИНЕ',
2 ' ШАГА.' /2X,'ДЛЯ ПРОДОЛЖЕНИЯ ИНТЕГРИРОВАНИЯ СЛЕДУЕТ',
3 ' УВЕЛИЧИТЬ ЗНАЧЕНИЕ GORX ЛИБО GPX, ' /
4 2X,'ЛИБО ОБОИХ ВМЕСТЕ, А ТАКЖЕ ЗАДАТЬ IFLAG=2:' /
5 5X,'GORX=',G12.5,2X,'GPX=',G12.5,2X,'IFLAG=',I2)
   IFLAG=2 90
   GOTO 10 91
70 PRINT 71 92
71 FORMAT (5X,'ЖЕЛАТЕЛЬНО ПЕРЕЙТИ К РЕЖИМУ ПОШАГОВОГО', 93
1 ' ИНТЕГРИРОВАНИЯ С ВЕЛИЧИНОЙ ШАГА, ОПРЕДЕЛЯЕМОЙ ' /
2 2X,'ПРОГРАММОЙ, ИЛИ ПОПРОБОВАТЬ ИСПОЛЬЗОВАТЬ МЕТОДЫ',
3 ' АДАМСА.' /2X,'ПРИ УПОРНОМ ЖЕЛАНИИ ПРОДОЛЖАТЬ ',
4 'ИНТЕГРИРОВАНИЕ ПО ПОДПРОГРАММЕ RKF45 НУЖНО ЗАДАТЬ',
5 ' IFLAG=2.' )
   IFLAG=2 94
   GOTO 10 95
80 PRINT 81 96
81 FORMAT (2X,'ИНТЕГРИРОВАНИЕ НЕЛЬЗЯ ПРОДОЛЖАТЬ, ', 97
1 'ПОКА НЕ БУДУТ ИСПРАВЛЕНЫ ОШИБОЧНЫЕ ВХОДНЫЕ ПАРАМЕТРЫ.')

```

Оператор 72 представляет собой впервые встречаемый нами вычисляемый оператор GOTO. Его общая форма имеет вид:

```
GOTO (M1,M2,M3, . . . ,MN) , I
```

где: M1,M2,M3,...MN — метка исполнимого оператора, а I — неиндексированная целая переменная, текущее значение которой находится в диапазоне от 1 до N. Этот оператор вызывает передачу управления оператору с меткой M1, M2, M3, ... или MN в зависимости от того, является ли текущее значение I равным соответственно 1, 2, 3, ... или N. Если значение I находится вне допустимого диапазона, выполняется оператор, следующий в программе за вычисляемым оператором GOTO.

В нашем случае роль целой переменной I выполняет параметр IFLAG, которому подпрограммой RKF45 может быть присвоено значение в диапазоне

от 1 до 8. В зависимости от этого будет выполняться оператор с той меткой, позиция расположения которой (от 1 до 8) в списке меток вычисляемого оператора GOTO совпадает со значением IFLAG.

Если значение IFLAG=1 или IFLAG=8, то начнет выполняться оператор с меткой 80, которая встречается в списке дважды соответственно в 1-й и 8-й позиции, который под управлением оператора 97 выведет на печать следующее аварийное сообщение: “Интегрирование нельзя продолжать, пока не будут исправлены ошибочные входные параметры.” После этого следующий в программе 22.1 оператор STOP под номером 98 укажет на конец вычислений, после чего будет остановлена работа программы.

При IFLAG=2 станет выполняться оператор с меткой 20 (под номером 74). Совместная работа операторов 74 и 75 характеризует нормальный режим интегрирования и описана в пояснениях к программе 22.1. После достижения $T > TK$, оператор 75 пропускается и следующий за ним оператор STOP 76 укажет на конец вычислений.

Если значение IFLAG=3, то оператор с меткой 30 (под номером 77) под управлением оператора 78 выведет на печать следующее сообщение: “Затребуется слишком высокая относительная точность, вследствие чего, параметр GOPX был изменен надлежащим для продолжения интегрирования образом: GOPX= ..., GPX= ..., IFLAG= ...”. Далее оператор 79 передаст управление оператору с меткой 10 (под номером 65) программы 22.1 для последующего обращения к подпрограмме RKF45 и попытки дальнейшего продолжения интегрирования.

При IFLAG=4 начнет выполняться оператор с меткой 40 (под номером 80). На печать под управлением оператора 81 будет выведено: “Интегрирование не было закончено за 3000 вычислений функции (500 шагов). Можно увеличить границы погрешностей или продолжать процесс: будут разрешены еще 3000 вычислений функции. GOPX= ..., GPX= ... IFLAG= ...”. Затем оператор безусловного перехода 82 передаст управление оператору с меткой 10 для следующего обращения к подпрограмме RKF45 и попытки дальнейшего продолжения интегрирования.

Если значение IFLAG=5, то станет выполняться оператор с меткой 50 (и номером 83), который присвоит абсолютной ошибке интегрирования GPX ненулевое значение $1.E-9$. Следующий за ним оператор 84 под управлением оператора 85 выведет на печать сообщение: “Решение обратилось в нуль, вследствие чего тест только относительной ошибки не проходит. Для продолжения интегрирования значение параметра GPX установлено ненулевым и составляет: GOPX= ..., GPX= ..., IFLAG= ...”. Затем оператор безусловного перехода 86 передаст управление оператору с меткой 10 для попытки дальнейшего продолжения интегрирования.

При $IFLAG=6$ начнет выполняться оператор с меткой 60 (и номером 87), который в 10 раз увеличит значение относительной ошибки интегрирования $GOPX$. Следующий за ним оператор 88 под управлением оператора 89 выведет на печать: “Требуемая точность не могла быть достигнута даже при наименьшей допустимой величине шага. Для продолжения интегрирования следует увеличить значение $GOPX$ либо GPX , либо обоих вместе, а также задать $IFLAG=2$: $GOPX=...$, $GPX=...$, $IFLAG=...$ ”.

Затем оператор 90 задает $IFLAG=2$ для возможности дальнейшего использования подпрограммы $RKF45$, а следующий за ним оператор безусловного перехода 91 передаст управление оператору с меткой 10 для попытки дальнейшего продолжения интегрирования. Если она окажется неудачной, то следует выполнить рекомендации полученного сообщения в более полном объеме: увеличить и значение GPX , поместив соответствующий арифметический оператор присваивания после оператора 87 рассматриваемого фрагмента, и продолжить попытку решения.

Если $IFLAG=7$, то начнет выполняться оператор с меткой 70 (и номером 92), который под управлением оператора 93 выведет на печать сообщение: “Желательно перейти к режиму пошагового интегрирования с величиной шага, определяемой программой, или попробовать использовать методы Адамса. При упорном желании продолжать интегрирование по подпрограмме $RKF45$ нужно задать $IFLAG=2$.” Затем оператор 94 присваивает $IFLAG=2$, а следующий за ним оператор безусловного перехода 95 передаст управление оператору с меткой 10 (под номером 65) программы 22.1 для попытки дальнейшего продолжения интегрирования.

Напомним вам, что рекомендованные в данном случае обобщенный и трехточечный методы Адамса реализованы соответственно в стандартных подпрограммах $OMAD$ и $MAD3$ (см. п. 18.4.2), работа с которыми рассматривалась в пп. 19.2.2-19.2.3.

Если $IFLAG < 1$ (режим пошагового интегрирования) или больше 8 (что может быть только вследствие грубой ошибки в начальном задании этого параметра в операторе 52), то будет выполняться следующий за ним оператор безусловного перехода 73, передающий управление оператору с меткой 80 для печати аварийного сообщения об ошибке с указанием значения параметра $IFLAG$.

В заключение отметим, что для работы с подпрограммой $RKF45$ справедливы все дополнения 19.1, 19.3-19.9 для численного интегрирования одномерных задач динамики (не касающиеся специальных вопросов использования стандартных подпрограмм). Надеемся, что вы их рассмотрите не только для приведенных примеров, но и для ваших задач.

22.3. Численное интегрирование двумерных задач динамики с использованием подпрограммы RKF45

Использование подпрограммы RKF45 для численного интегрирования плоских задач механики, описываемых двумя дифференциальными уравнениями второго порядка, рассмотрим в программе 22.2 на примере (20.1). Хотя в программе 22.2 будет только 3 оператора, не встречавшихся ранее и не описанных в программах 19.1, 20.1 или 22.1, для удобства пользования мы приведем ее полностью. Однако операторы, совпадающие с программами 19.1, 20.1 или 22.1, повторно не описываются. Для отличия они имеют после своего номера точку и цифры 1, 2 или 4, соответствующие порядковому номеру указанных программ в седьмой части пособия:

```

С      ПРОГРАММА 22.2
С      (ПРОГРАММА 20.1 С ИСПОЛЬЗОВАНИЕМ ПОДПРОГРАММЫ RKF45)
      EXTERNAL SDU1P                                10.4
      DIMENSION X(4), IWORK(5), WORK(27)           20
      DATA T, TK/0., 1.2/, GPX, GOPX/0., 1.E-9/    40.4
      DATA X/0., 10., 20., 0./                   45.2
      KDU=4                                         50
      IFLAG=1                                       52.4
      HPR=0.1                                       54.4
      TOUT=T                                        56.4
      PRINT 5                                       60.2
5     FORMAT (T4, 'ВРЕМЯ, СЕК', T21, 'X, M', T35, 'Z, M',
*          T49, 'XP, M/C', T62, 'ZP, M/C')         61.2
10    CALL RKF45(SDU1P, KDU, X, T, TOUT, GOPX, GPX, IFLAG,
*              WORK, IWORK)                        65.4
      CALL OUTF(T, X)                               67.4
      TOUT=T+HPR                                    74.4
      IF (T.LT.TK) GOTO 10                          75.4
      STOP                                          98.1
      END                                           99.1
      SUBROUTINE SDU1P(T, X, XP)                   100.1
      DIMENSION X(4), XP(4)                       110.2
      XP(1)=X(3)                                    140.2
      XP(2)=X(4)                                    142.2
      XP(3)=-4.*X(1)-2.*X(3)                       150.2
      XP(4)=-4.*X(2)-2.*X(4)-9.81                 152.2
      RETURN                                       196.1
      END                                           198.1
      SUBROUTINE OUTF(T, X)                        200.4
      DIMENSION X(4)                               210
      PRINT 25, T, (X(I), I=1, 4)                  240.2
25    FORMAT (2X, 5(2X, G12.5))                   241.2

```

RETURN	260.1
END	262.1

Оператор 20 задает описание массива переменной X и рабочих массивов IWORK и WORK, которым отводится место в памяти для размещения соответственно 4, 5 и 27 элементов.

Оператор 50 присваивает переменной KDU значение количества дифференциальных уравнений 1-го порядка, равное четырем.

Оператор 210 задает описание формального массива X, состоящего из четырех элементов.

Отметим, что с программой 22.2 могут работать дополнения 19.1, 19.3-19.9, 20.1-20.2, 22.1 (с учетом в дополнении 19.5 особенностей двумерного характера задачи, описанных в дополнении 20.1). При возникновении трудностей в случае линейных задач следует обратиться к дополнению 20.2.

22.4. Численное интегрирование трехмерных задач динамики с использованием подпрограммы RKF45

Использование подпрограммы RKF45 для численного интегрирования пространственных задач механики, описываемых тремя дифференциальными уравнениями второго порядка, рассмотрим на примере (20.2). Хотя в программе 22.3 (как и в программе 22.2) будет также только 3 оператора, не встречавшихся ранее в другом наборе программ 19.1, 20.2 или 22.1, но для удобства пользования и применения ранее описанных дополнений мы тоже приведем ее полностью. Однако операторы, совпадающие с программами 19.1, 20.2 или 22.1, повторно не описываются. Для отличия они имеют после своего номера точку и соответственно цифры 1, 3 или 4, также соответствующие порядковому номеру указанных программ в седьмой части пособия:

```

С      ПРОГРАММА 22.3
С      (ПРОГРАММА 20.2 С ИСПОЛЬЗОВАНИЕМ ПОДПРОГРАММЫ RKF45)
      EXTERNAL SDU1P                                10.4
      DIMENSION X(6), IWORK(5), WORK(39)           20
      DATA T, TK/0., 1.2/, GPX, GOPX/0., 1.E-9/    40.4
      DATA X/0., 0., 0., 1., 1., 2./              45.3
      KDU=6                                          50
      IFLAG=1                                       52.4
      HPR=0.1                                       54.4
      TOUT=T                                        56.4
      PRINT 5                                       60.3
5      FORMAT(T4, 'ВРЕМЯ, СЕК', T21, 'X, M', T35, 'Y, M',
*      T49, 'Z, M', T62, 'XP, M/C', T76, 'YP, M/C', T90, 'ZP, M/C')
10     CALL RKF45(SDU1P, KDU, X, T, TOUT, GOPX, GPX, IFLAG,
*      WORK, IWORK)                                65.4

```

CALL OUTF(T,X)	67.4
TOUT=T+HPR	74.4
IF(T.LT.TK) GOTO 10	75.4
STOP	98.1
END	99.1
SUBROUTINE SDU1P(T,X,XP)	100.1
DIMENSION X(6),XP(6)	110.3
XP(1)=X(4)	140.3
XP(2)=X(5)	142.3
XP(3)=X(6)	144.3
XP(4)=-0.5*X(4)	150.3
XP(5)=-0.5*X(5)	152.3
XP(6)=-0.5*X(6)+9.81	154.3
RETURN	196.1
END	198.1
SUBROUTINE OUTF(T,X)	200.4
DIMENSION X(6)	210
PRINT 25,T,(X(I),I=1,6)	240.3
25 FORMAT(2X,7(2X,G12.5))	241.3
RETURN	260.1
END	262.1

Оператор 20 задает описание массивов X, IWORK и WORK, содержащих соответственно 6, 5 и 39 элементов.

Оператор 50 присваивает переменной KDU значение количества дифференциальных уравнений 1-го порядка, равное шести.

Оператор 210 задает описание формального массива X, состоящего также из шести элементов.

Отметим, что с программой 22.3 могут работать дополнения 19.1, 19.3-19.9, 20.1-20.4 и 22.1 (с учетом в дополнениях 19.5 и 20.1 особенностей трехмерного характера задачи). При возникновении трудностей при использовании программы 22.3 для численного интегрирования плоских и линейных задач следует обратиться к дополнениям 20.3 и 20.4 соответственно.

22.5. Вычисления с удвоенной точностью (подпрограмма DRKF45)

Рассмотрим кратко те вопросы, с которыми приходится сталкиваться при использовании подпрограммы DRKF45 для численного интегрирования одномерных (программа 22.1), двумерных (программа 22.2) и трехмерных (программа 22.3) задач динамики. Из соображения экономии места будут приводиться только изменяемые или дополняемые в

соответствующие программы операторы. Надеемся, что вы не только с пониманием восстановите (по номерам операторов) и проверите работу этих программ с использованием удвоенной точности на рассматриваемых примерах, но сделаете это и для приведенных в пособии дополнений и ваших задач. Операторы, совпадающие с фрагментом п. 21.1, повторно не описываются. Для отличия они имеют после своего номера точку, цифру 1 и букву D.

1. Для численного интегрирования одного дифференциального уравнения 2-го порядка (16.3) в программу 22.1 для вычислений с удвоенной точностью следует внести следующие изменения:

```

IMPLICIT REAL *8 (A-H, O-Z)                                9.1D
DATA T, TK/0.D0, 1.2D0/, GPX, GOPX/0.D0, 1.D-9/           40
DATA X/0.3D0, 2.D0/                                        45.1D
HPR=0.1D0                                                  54
10 CALL DRKF45 (SDU1P, KDU, X, T, TOUT, GOPX, GPX, IFLAG, 65
*                WORK, IWORK)
IMPLICIT REAL *8 (A-H, O-Z)                                109.1D
DATA PI/3.141592653589D0/, ZM/0.01D0/, R/0.2D0/,
*   C/1.D0/, ZL0/0.2D0/                                    120.1D
AL=PI/6.D0                                                 130.1D
C1=- (C/ZM-PI*PI* (DSIN (AL) ) **2)                       134.1D
C5=PI*PI*R*DSIN (AL) -9.81D0*DCOS (AL) +C*ZL0/ZM        136.1D
IMPLICIT REAL *8 (A-H, O-Z)                                209.1D

```

Оператор 40 DATA используется для задания начальных значений удвоенной точности нижней T и верхней TK границы интервала интегрирования, а также границ абсолютной GPX и относительной GOPX допустимых погрешностей вычислений: T=0.D0, TK=1.2D0, GPX=0.D0, GOPX=1.D-8.

Оператор 54 задает значение шага печати результатов в удвоенной точности HPR=0.1D0.

Оператор 65 осуществляет обращение к подпрограмме удвоенной точности DRKF45. Все его фактические параметры описаны в программы 22.1, но должны быть представлены с учетом требований п. 15.3 для использования удвоенной точности.

2. Для вычислений с удвоенной точностью в программу 22.2 следует внести следующие изменения и дополнения:

```

IMPLICIT REAL *8 (A-H, O-Z)                                9.1D
DATA T, TK/0.D0, 1.2D0/, GPX, GOPX/0.D0, 1.D-9/           40
DATA X/0.D0, 10.D0, 20.D0, 0.D0/                          45
HPR=0.1D0                                                  54
10 CALL DRKF45 (SDU1P, KDU, X, T, TOUT, GOPX, GPX, IFLAG, 65
*                WORK, IWORK)

```

IMPLICIT REAL *8 (A-H, O-Z)	109.1D
XP (3)=-4.D0*X (1)-2.D0*X (3)	150
XP (4)=-4.D0*X (2)-2.D0*X (4)-9.81D0	152
IMPLICIT REAL *8 (A-H, O-Z)	209.1D

Операторы 40, 54 и 65 полностью совпадают с вышеприведенным в п. 22.5.1 фрагментом изменений к программе 22.1 и повторно не описываются.

Оператор 45 задает начальные значения удвоенной точности переменным X и Z, носящим в программе обозначения X(1) и X(2), и их производным соответственно X(3) и X(4) при времени T=0, которые приведены в условии рассматриваемого примера: X(1)=0.D0, X(2)=10.D0, X(3)=20.D0, X(4)=0.D0.

Операторы 150-152 задают формулы для вычисления правых частей системы дифференциальных уравнений рассматриваемого типового примера с использованием числовых величин удвоенной точности.

3. Для вычислений с удвоенной точностью в программу 22.3 следует внести следующие изменения и дополнения:

IMPLICIT REAL *8 (A-H, O-Z)	9.1D
DATA T, TK/0.D0, 1.2D0/, GPX, GOPX/0.D0, 1.D-9/	40
DATA X/0.D0, 0.D0, 0.D0, 1.D0, 1.D0, 2.D0/	45
HPR=0.1D0	54
10 CALL DRKF45 (SDU1P, KDU, X, T, TOUT, GOPX, GPX, IFLAG, 65	
* WORK, IWORK)	
IMPLICIT REAL *8 (A-H, O-Z)	109.1D
XP (4)=-0.5D0*X (4)	150
XP (5)=-0.5D0*X (5)	152
XP (6)=-0.5D0*X (6)+9.81D0	154
IMPLICIT REAL *8 (A-H, O-Z)	209.1D

Здесь также операторы 40, 54 и 65 полностью совпадают с фрагментом изменений к программе 22.1, приведенным в п. 22.5.1, и повторно не описываются.

Оператор 45 присваивает начальные значения удвоенной точности переменным X, Y и Z, носящим в программе обозначения X(1), X(2) и X(3), и их производным соответственно X(4), X(5) и X(6) при времени T=0, которые приведены в условии рассматриваемого примера: X(1)=0.D0, X(2)=0.D0, X(3)=0.D0, X(4)=1.D0, X(5)=1.D0, X(6)=2.D0.

Операторы 150-154 задают формулы для вычисления правых частей системы дифференциальных уравнений (20.2) с использованием числовых величин удвоенной точности.

Полный вид программ 22.1 — 22.3 в удвоенной точности приведен в приложении 4.

ЛИТЕРАТУРА

1. Браух В. Программирование на Фортране-77 для инженеров: Пер. с нем. – М.: Мир, 1987.
2. Бухтияров А.М., Маликова Ю.П., Фролов Г.Д. Практикум по программированию на Фортране (ОС ЕС ЭВМ): Учеб.пособие. – М.: Наука, 1988.
3. Выполнение заданий для курсовых работ по теоретической механике с применением ЭВМ типа ДВК: Метод. пособие по комплекс. преподаванию теор. механики, вычислит. мат. и программирования / [В.М. Носов, Ф.И. Подгайский, А.С. Ковеня и др.; Под общей ред. В.М. Носова]. – Мн.: БГПА, 1992.
4. Грунд Ф. Программирование на языке Фортран-IV. – М.: Мир, 1978.
5. Денисов В. Windows 95 с самого начала. – СПб.: Питер, 1996.
6. Зверев С.А. Практическая работа в MS DOS / Под общ. ред. Ю.В. Кузьменко. – М.: Воениздат, 1991.
7. Инструментальные средства персональных ЭВМ. В 10 кн. Кн. 3. Программирование на языке Фортран-77: Практ. пособие / В.Е. Алексеев, А.С. Ваулин, Е.Э. Иванцова и др.; Под ред. Б.Г. Трусова. – М.: Высш.шк., 1993.
8. Колдербэнк В.Дж. Курс программирования на Фортране: Фортран-66 и Фортран-77. – М.: Радио и связь, 1986.
9. Кузьменко Ю.В., Зверев С.А. IBM PC: все для начинающего пользователя / Под общ.ред.Кузьменко Ю.В. – М.: Воениздат, 1992.
10. Мак-Кракен Д., Дорн У. Численные методы и программирование на Фортране. – М.: Мир, 1977.
11. Microsoft Windows: Версия 3.0 для операционной системы MS-DOS / Руководство пользователя: [Перевод]. – М.: Эрус, 1991.
12. Математическое обеспечение ЕС ЭВМ: [Сборник]. – Мн.: ИМ АН БССР, 1973-1983, вып. 1-44.
13. Новожилов И.В., Зацепин М.Ф. Типовые расчеты по теоретической механике на базе ЭВМ: Учеб.пособие для вузов. – М.: Высш.шк., 1986.
14. Носов В.М. Выполнение заданий для курсовых работ по теоретической механике с применением ЭВМ: Метод. пособие по комплекс. преподаванию теор. механики, вычислит. мат. и программирования. – Мн.: БПИ – БГПА, 1989 – 1993. – Ч. 1 – 6.
15. Носов В.М. Методические указания к выполнению индивидуальных заданий по курсу теоретической механики. – Мн.: БПИ, 1984 – 1985. – Ч. 1 – 4.

16. Носов В.М. Определение скоростей и ускорений с использованием их аналогов для основных (базовых) механизмов: Учеб.-метод. пособие для мех. спец. – Мн.: БГПА, 1994.
17. Офицеров Д.В. и др. Программирование на персональных ЭВМ: Учеб. пособие / Д.В. Офицеров, А.Б. Долгий, В.А. Старых; Под общ.ред. Д.В. Офицера. – Мн.: Вышэйш.шк., 1993.
18. Программирование на Фортране-77 / Дж.Ашкрафт, Р.Элдридж, Р.Полсон, Г.Уилсон: Пер.с англ. – М.: Радио и связь, 1990.
19. Программное обеспечение ЕС ЭВМ: [Сборник]. – Мн.: ИМ АН БССР, 1983 – 1985, вып. 45 – 67.
20. Программное обеспечение персональных ЭВМ: Справочное пособие. – Киев: Навук. думка, 1989.
21. Сборник заданий для курсовых работ по теоретической механике: Учеб. пособие для техн. вузов/ [Яблонский А.А., Норейко С.С., Вольфсон С.А. и др.; Под ред. А.А. Яблонского]. – 4-е изд. – М.: Высш. шк., 1985.
22. Сборник заданий для курсовых работ по теоретической механике: Учеб. пособие для техн. вузов/ [Яблонский А.А., Норейко С.С., Вольфсон С.А. и др.; Под ред. А.А. Яблонского]. – 3-е изд. – М.: Высш. шк., 1978.
23. Сборник научных программ на Фортране. – М.: Статистика, 1974, вып. 1–2.
24. Соловьев П.В. Fortran для персонального компьютера: Справоч. пособие. – М.: Arist, 1991.
25. Уорд Т., Бромхед Э. Фортран и искусство программирования на персональных ЭВМ. – М.: Радио и связь, 1993.
26. Фигурнов В.Э. IBM PC для пользователя. – 6-е изд., перераб. и доп. – М.: ИНФРА-М, 1995.
27. Форсайт Дж., Малькольм М., Моулер К. Машинные методы математических вычислений. – М.: Мир, 1980.
28. Фортран-77 для ПЭВМ ЕС / З.С.Брич, Д.В.Капилевич, Н.А.Клецкова. – М.: Финансы и статистика, 1991.
29. Фортран-77 ЕС ЭВМ: Справ.изд. / [З.С.Брич и др.] – М.: Финансы и статистика, 1989.
30. Фортран ЕС ЭВМ: Справ.изд. / [З.С.Брич и др.] – М.: Финансы и статистика, 1985.
31. Фурунжиев Р.И. Вычислительная техника: Практикум. – Мн.: Вышэйш. шк., 1985.
32. Шкаев А.В. Справочное руководство по работе на персональном компьютере. – М.: Радио и связь, 1992.
33. Шуп Т. Решение инженерных задач на ЭВМ: Практическое руководство. – М.: Мир, 1982.

ПРИЛОЖЕНИЕ

Приложение 1.

```

SUBROUTINE DDECOM (NDIM, N, A, COND, IPVT, WORK)
C ПОДПРОГРАММА ПЕРЕДЕЛАНА В УДВОЕННУЮ ТОЧНОСТЬ
  INTEGER NDIM, N
  REAL *8 A (NDIM,N), COND, WORK (N)
  INTEGER IPVT (N)
C
  REAL *8 EK, T, ANORM, YNORM, ZNORM
  INTEGER NM1, I, J, K, KP1, KB, KM1, M
C
  IPVT (N) = 1
  IF (N.EQ.1) GO TO 80
  NM1 = N - 1
C
C   ВЫЧИСЛИТЬ ПЕРВУЮ НОРМУ МАТРИЦЫ А
C
  ANORM = 0.0D0
  DO 10 J = 1, N
    T = 0.0D0
    DO 5 I = 1, N
      T = T + DABS (A (I, J))
    5   CONTINUE
    IF (T.GT.ANORM) ANORM = T
  10  CONTINUE
C
C   ГАУССОВО ИСКЛЮЧЕНИЕ С ЧАСТИЧНЫМ ВЫБОРОМ ВЕДУЩЕГО ЭЛЕМЕНТА
C
  DO 35 K = 1, NM1
    KP1 = K + 1
C
C   НАЙТИ ВЕДУЩИЙ ЭЛЕМЕНТ
C
    M = K
    DO 15 I = KP1, N
      IF (DABS (A (I, K)) .GT. DABS (A (M, K))) M = I
    15  CONTINUE
    IPVT (K) = M

```

```

IF (M.NE.K) IPVT(N)=-IPVT(N)
T=A(M,K)
A(M,K)=A(K,K)
A(K,K)=T
C
C ПРОПУСТИТЬ ЭТОТ ШАГ, ЕСЛИ ВЕДУЩИЙ ЭЛЕМЕНТ = 0
C
IF (T.EQ.0.0D0) GO TO 35
C
C ВЫЧИСЛИТЬ МНОЖИТЕЛИ
C
DO 20 I=KP1,N
A(I,K)=-A(I,K)/T
20 CONTINUE
C
C ПЕРЕСТАВЛЯТЬ И ИСКЛЮЧАТЬ ПО СТОЛБЦАМ
C
DO 30 J=KP1,N
T = A(M,J)
A(M,J)=A(K,J)
A(K,J)=T
IF (T.EQ.0.0D0) GO TO 30
DO 25 I=KP1,N
A(I,J)=A(I,J)+A(I,K)*T
25 CONTINUE
30 CONTINUE
35 CONTINUE
C
C РЕШИТЬ СИСТЕМУ (ТРАНСПОНИРОВАННАЯ ДЛЯ A) S Y=E
C
DO 50 K=1,N
T=0.0D0
IF (K.EQ.1) GO TO 45
KM1=K-1
DO 40 I=1,KM1
T=T+A(I,K)*WORK(I)
40 CONTINUE
45 EK=1.0D0
IF (T.LT.0.0D0) EK=-1
IF (A(K,K).EQ.0.0D0) GO TO 90
WORK(K)=- (EK+T)/A(K,K)
50 CONTINUE
DO 60 KB=1,NM1
K=N-KB
T=0.0D0
KP1=K+1

```

```
DO 55 I=KPL,N
    T=T+A(I,K)*WORK(K)
55  CONTINUE
    WORK(K)=T
    M=IPVT(K)
    IF (M.EQ.K) GO TO 60
    T=WORK(M)
    WORK(M)=WORK(K)
    WORK(K)=T
60  CONTINUE
C
    YNORM=0.0D0
    DO 65 I=1,N
        YNORM=YNORM+DABS(WORK(I))
65  CONTINUE
C
C    РЕШИТЬ СИСТЕМУ (A)(Z)=Y
C
    CALL DSOLVE(NDIM,N,A,WORK,IPVT)
C
    ZNORM=0.0D0
    DO 70 I=1,N
        ZNORM=ZNORM+DABS(WORK(I))
70  CONTINUE
C
C    ОЦЕНИТЬ ОБУСЛОВЛЕННОСТЬ
C
    COND=ANORM*ZNORM/YNORM
    IF (COND.LT.1.0D0) COND=1.0D0
    RETURN
C
C    СЛУЧАЙ МАТРИЦЫ 1s1
C
80  COND=1.0D0
    IF (A(1,1).NE.0.0D0) RETURN
C
C    ТОЧНАЯ ВЫРОЖДЕННОСТЬ
C
90  COND=1.0D+32
    RETURN
    END
SUBROUTINE DSOLVE(NDIM,N,A,B,IPVT)
C    ПРОГРАММА ПЕРЕДЕЛАНА В УДВОЕННОЙ ТОЧНОСТИ
    INTEGER NDIM,N,IPVT(N)
```

```

REAL *8 A (NDIM,N) , B (N)
C
C ПОДПРОГРАММУ НЕ СЛЕДУЕТ ИСПОЛЬЗОВАТЬ, ЕСЛИ DBESOM ОБНАРУЖИЛА
C ВЫРОЖДЕННОСТЬ
INTEGER KB, KM1, NM1, KP1, I, K, M
REAL *8 T
C
IF (N.EQ.1) GO TO 50
NM1=N-1
DO 20 K=1, NM1
  KP1=K+1
  M=IPVT (K)
  T=B (M)
  B (M)=B (K)
  B (K)=T
  DO 10 I=KP1, N
    B (I)=B (I)+A (I, K) *T
10  CONTINUE
20  CONTINUE
C
DO 40 KB=1, NM1
  KM1=N-KB
  K=KM1+1
  B (K)=B (K) /A (K, K)
  T=-B (K)
  DO 30 I=1, KM1
    B (I)=B (I)+A (I, K) *T
30  CONTINUE
40  CONTINUE
50  B (1)=B (1) /A (1, 1)
    RETURN
    END

```

Приложение 2.

```

SUBROUTINE DRKF45 (SDU1P, KDU, X, T, TOUT, GOPX, GPX, IFLAG,
*                WORK, IWORK)
C ПОДПРОГРАММА ПЕРЕДЕЛАНА В ВЫЧИСЛЕНИЯ С УДВОЕННОЙ ТОЧНОСТЬЮ.
C МЕТОД РУНГЕ-КУТТА-ФЕЛЬВЕРГА ЧЕТВЕРТОГО-ПЯТОГО ПОРЯДКА.
C
C DRKF45 ПРЕДНАЗНАЧЕНА, ГЛАВНЫМ ОБРАЗОМ, ДЛЯ РЕШЕНИЯ НЕЖЕСТКИХ И
C СЛАБОЖЕСТКИХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ, КОГДА ВЫЧИСЛЕНИЕ
C ПРОИЗВОДНЫХ НЕ СЛИШКОМ ДОРОГОСТОЯЩЕЕ. DRKF45, ВООБЩЕ ГОВОРЯ,
C НЕ СЛЕДУЕТ ИСПОЛЬЗОВАТЬ, ЕСЛИ ПОЛЬЗОВАТЕЛЮ ТРЕБУЕТСЯ ВЫСОКАЯ
C ТОЧНОСТЬ.

```

С РЕЗЮМЕ

С

С ПОДПРОГРАММА DRKF45 ИНТЕГРИРУЕТ СИСТЕМУ ИЗ KDU

С ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ ПЕРВОГО ПОРЯДКА СЛЕДУЮЩЕГО ВИДА:

С $dx(I)/dt = F(T, X(1), X(2), \dots, X(KDU))$,

С ГДЕ $X(I)$ ЗАДАНЫ В Т.

С ОБЫЧНО ПОДПРОГРАММУ ПРИМЕНЯЮТ ДЛЯ ИНТЕГРИРОВАНИЯ ОТ Т ДО TOUT.

С ОДНАКО ЕЕ МОЖНО ИСПОЛЬЗОВАТЬ И КАК ОДНОШАГОВЫЙ ИНТЕГРАТОР,

С ЧТОБЫ ПРОДОЛЖИТЬ РЕШЕНИЕ НА ОДИН ШАГ В НАПРАВЛЕНИИ TOUT. НА

С ВЫХОДЕ ПАРАМЕТРАМ, ФИГУРИРУЮЩИМ В СПИСКЕ ВЫЗОВА, ПРИСВАИВАЮТСЯ

С ЗНАЧЕНИЯ, НЕОБХОДИМЫЕ ДЛЯ ПРОДОЛЖЕНИЯ ИНТЕГРИРОВАНИЯ. ПОЛЬЗОВА-

С ТЕЛЮ НУЖНО ЛИШЬ ЕЩЕ РАЗ ОБРАТИТЬСЯ К DRKF45 (И, ВОЗМОЖНО, ОПРЕ-

С ДЕЛИТЬ ЗНАЧЕНИЕ ДЛЯ TOUT). В ДЕЙСТВИТЕЛЬНОСТИ DRKF45 - ЭТО

С ПОДПРОГРАММА ИНТЕРФЕЙСА, КОТОРАЯ ВЫЗЫВАЕТ ПОДПРОГРАММУ RKFS,

С ОСУЩЕСТВЛЯЮЩУЮ ПРОЦЕСС РЕШЕНИЯ. RKFS В СВОЮ ОЧЕРЕДЬ ВЫЗЫВАЕТ

С ПОДПРОГРАММУ FENL, КОТОРАЯ ВЫЧИСЛЯЕТ ПРИВЛИЖЕННОЕ

С РЕШЕНИЕ НА ОДИН ШАГ.

С

С ПАРАМЕТРЫ ПОДПРОГРАММЫ:

С SDU1P - ПОДПРОГРАММА SDU1P(T, X, XP) ДЛЯ ВЫЧИСЛЕНИЯ ПРОИЗВОДНЫХ

С $XP(I) = dx(I)/dt$;

С KDU - ЧИСЛО ИНТЕГРИРУЕМЫХ УРАВНЕНИЙ;

С X(*) - РЕШЕНИЕ В ТОЧКЕ Т;

С Т - НЕЗАВИСИМАЯ ПЕРЕМЕННАЯ;

С TOUT - ТОЧКА ВЫХОДА, В КОТОРОЙ НУЖНО ОПРЕДЕЛИТЬ ЗНАЧЕНИЕ

С РЕШЕНИЯ;

С GORX, GRX - ГРАНИЦЫ ОТНОСИТЕЛЬНОЙ И АБСОЛЮТНОЙ ПОГРЕШНОСТИ ДЛЯ

С ТЕСТА ЛОКАЛЬНОЙ ОШИБКИ. НА КАЖДОМ ШАГЕ ПРОГРАММА ТРЕБУЕТ

С ВЫПОЛНЕНИЯ УСЛОВИЯ:

С $ABS(LOCAL ERROR) \cdot LE \cdot GORX \cdot ABS(X) + GRX$ ДЛЯ КАЖДОЙ КОМПО-

С НЕНТЫ ВЕКТОРОВ ЛОКАЛЬНОЙ ОШИБКИ И РЕШЕНИЯ;

С IFLAG - УКАЗАТЕЛЬ РЕЖИМА ИНТЕГРИРОВАНИЯ.

С WORK(*) - МАССИВ, СОДЕРЖАЩИЙ ИНФОРМАЦИЮ, ВНУТРЕННЮЮ ДЛЯ DRKF45,

С КОТОРАЯ НЕОБХОДИМА ПРИ ПОСЛЕДУЮЩИХ ВЫЗОВАХ. ЕГО РАЗМЕР

С ДОЛЖЕН БЫТЬ НЕ МЕНЬШЕ $3+6 \cdot KDU$;

С IWORK - ЦЕЛЫЙ МАССИВ, СОДЕРЖАЩИЙ ИНФОРМАЦИЮ, ВНУТРЕННЮЮ ДЛЯ

С DRKF45, КОТОРАЯ НЕОБХОДИМА ПРИ ПОСЛЕДУЮЩИХ ВЫЗОВАХ. ЕГО

С РАЗМЕР ДОЛЖЕН БЫТЬ НЕ МЕНЬШЕ 5.

С

С ПЕРВОЕ ОБРАЩЕНИЕ К DRKF45

С ПОЛЬЗОВАТЕЛЬ ДОЛЖЕН ПРЕДУСМОТРЕТЬ В СВОЕЙ ВЫЗЫВАЮЩЕЙ ПРОГРАММЕ

С ПАМЯТЬ ДЛЯ СЛЕДУЮЩИХ МАССИВОВ, ФИГУРИРУЮЩИХ В СПИСКЕ ВЫЗОВА -

С $X(KDU)$, $WORK(3+6 \cdot KDU)$, $IWORK(5)$;

С КРОМЕ ТОГО ОН ДОЛЖЕН ОБЪЯВИТЬ SDU1P В ОПЕРАТОРЕ EXTERNAL,

С ПОДГОТОВИТЬ ПОДПРОГРАММУ SDU1P(T,X,XP) И ПРИСВОИТЬ
 С НАЧАЛЬНЫЕ ЗНАЧЕНИЯ ПАРАМЕТРАМ:
 С KDU - ЧИСЛО ИНТЕГРИРУЕМЫХ УРАВНЕНИЙ (KDU.GE.1);
 С X(*) - ВЕКТОР НАЧАЛЬНЫХ УСЛОВИЙ;
 С T - НАЧАЛЬНАЯ ТОЧКА ИНТЕГРИРОВАНИЯ, T ДОЛЖНО БЫТЬ ПЕРЕМЕННОЙ;
 С TOUT - ТОЧКА ВЫХОДА, В КОТОРОЙ НУЖНО НАЙТИ ЗНАЧЕНИЕ РЕШЕНИЯ.
 С T=TOUT ВОЗМОЖНО ЛИШЬ ПРИ ПЕРВОМ ОБРАЩЕНИИ. В ЭТОМ СЛУЧАЕ ВЫХОД
 С ИЗ DRKF45 ПРОИСХОДИТ СО ЗНАЧЕНИЕМ ПАРАМЕТРА IFLAG=2, ЕСЛИ
 С МОЖНО ПРОДОЛЖАТЬ ИНТЕГРИРОВАНИЕ.
 С GORX,GRX - ГРАНИЦЫ ДЛЯ ОТНОСИТЕЛЬНОЙ И АБСОЛЮТНОЙ ЛОКАЛЬНЫХ
 С ПОГРЕШНОСТЕЙ. ЭТИ ГРАНИЦЫ ДОЛЖНЫ БЫТЬ НЕОТРИЦАТЕЛЬНЫ.
 С GORX ДОЛЖНА БЫТЬ ПЕРЕМЕННОЙ, А GRX МОЖЕТ БЫТЬ И CONST.
 С ПРОГРАММЕ, ВООБЩЕ ГОВОРЯ, НЕ СЛЕДУЕТ ЗАДАВАТЬ ГРАНИЦУ ДЛЯ ОТНО-
 С СИТЕЛЬНОЙ ОШИБКИ, МЕНЬШЮ, ЧЕМ ПРИМЕРНО 1.0E-8. ДАВЫ ИЗБЕЖАТЬ
 С ТРУДНОСТЕЙ, СВЯЗАННЫХ С ОЧЕНЬ ВЫСОКИМИ ЗАПРОСАМИ К ТОЧНОСТИ,
 С ПРОГРАММА ТРЕБУЕТ, ЧТОБЫ GORX БЫЛА БОЛЬШЕ, ЧЕМ НЕКОТОРЫЙ
 С ПАРАМЕТР ОТНОСИТЕЛЬНОЙ ОШИБКИ, ВЫЧИСЛЯЕМЫЙ ВНУТРИ НЕЕ И ЗАВИСЯЩИЙ
 С ОТ МАШИНЫ, В ЧАСТНОСТИ, НЕ РАЗРЕШАЕТСЯ ЗАДАНИЕ ТОЛЬКО АБСОЛЮТНОЙ
 С ОШИБКИ. ЕСЛИ ЖЕ ЗАДАНО ЗНАЧЕНИЕ GORX, МЕНЬШЕ ДОПУСТИМОГО,
 С ТО DRKF45 УВЕЛИЧИВАЕТ GORX НАДЛЕЖАЩИМ ОБРАЗОМ И ВОЗВРАЩАЕТ
 С УПРАВЛЕНИЕ ПОЛЬЗОВАТЕЛЮ, ПРЕЖДЕ ЧЕМ ПРОДОЛЖАТЬ ИНТЕГРИРОВАНИЕ.
 С IFLAG=+1,-1. ЭТО УКАЗАТЕЛЬ НАСТРОЙКИ ПРОГРАММЫ ДЛЯ КАЖДОЙ НОВОЙ
 С ЗАДАЧИ. НОРМАЛЬНОЕ ВХОДНОЕ ЗНАЧЕНИЕ =+1. ПОЛЬЗОВАТЕЛЬ ДОЛЖЕН
 С ЗАДАВАТЬ IFLAG=-1 ЛИШЬ В ТОМ СЛУЧАЕ, КОГДА НЕОБХОДИМО УПРАВЛЕНИЕ
 С ОДНОШАГОВЫМ ИНТЕГРАТОРОМ. В ЭТОМ СЛУЧАЕ DRKF45 ПЫТАЕТСЯ ПРОДОЛ-
 С ЖИТЬ РЕШЕНИЕ НА ОДИН ШАГ В НАПРАВЛЕНИИ TOUT ПРИ КАЖДОМ ОЧЕРЕД-
 С НОМ ВЫЗОВЕ. ПОСКОЛЬКУ ЭТОТ РЕЖИМ РАБОТЫ ВЕСЬМА НЕЭКОНОМИЧЕН,
 С ЕГО СЛЕДУЕТ ПРИМЕНЯТЬ ЛИШЬ В СЛУЧАЕ КРАЙНЕЙ НЕОБХОДИМОСТИ.
 С
 С ИНФОРМАЦИЯ НА ВЫХОДЕ:
 С X(*) - РЕШЕНИЕ В ТОЧКЕ T
 С T - ПОСЛЕДНЯЯ ТОЧКА, ДОСТИГНУТАЯ ПРИ ИНТЕГРИРОВАНИИ.
 С IFLAG=2 - ПРИ ИНТЕГРИРОВАНИИ ДОСТИГНУТО TOUT. ЭТО ЗНАЧЕНИЕ
 С ПАРАМЕТРА УКАЗЫВАЕТ НА УСПЕШНЫЙ ВЫХОД И ЯВЛЯЕТСЯ НОРМАЛЬНЫМ
 С РЕЖИМОМ ДЛЯ ПРОДОЛЖЕНИЯ ИНТЕГРИРОВАНИЯ.
 С =-2 - БЫЛ ПРЕДПРИНЯТ ОДИН ШАГ В НАПРАВЛЕНИИ TOUT, ОКАЗАВШИЙСЯ
 С УСПЕШНЫМ. ЭТО НОРМАЛЬНЫЙ РЕЖИМ ДЛЯ ПРОДОЛЖЕНИЯ ПОШАГОВОГО
 С ИНТЕГРИРОВАНИЯ.
 С =3 - ИНТЕГРИРОВАНИЕ НЕ БЫЛО ЗАКОНЧЕНО ИЗ-ЗА ТОГО, ЧТО ЗАДАННОЕ
 С ЗНАЧЕНИЕ ГРАНИЦЫ ДЛЯ ОТНОСИТЕЛЬНОЙ ОШИБКИ ОКАЗАЛОСЬ СЛИШКОМ
 С МАЛО. ДЛЯ ПРОДОЛЖЕНИЯ ИНТЕГРИРОВАНИЯ GORX БЫЛО НАДЛЕЖАЩИМ
 С ОБРАЗОМ УВЕЛИЧЕНО.
 С =4 - ИНТЕГРИРОВАНИЕ НЕ БЫЛО ЗАКОНЧЕНО ИЗ-ЗА ТОГО, ЧТО ПОТРЕВО-
 С ВАЛОСЬ БОЛЕЕ 3000 ВЫЧИСЛЕНИЙ ПРОИЗВОДНОЙ. ЭТО СООТВЕТСТВУЕТ
 С ПРИБЛИЗИТЕЛЬНО 500 ШАГАМ.

C =5 - ИНТЕГРИРОВАНИЕ НЕ БЫЛО ЗАКОНЧЕНО ИЗ-ЗА ТОГО, ЧТО РЕШЕНИЕ
C ОБРАТИЛОСЬ В НУЛЬ, ВСЛЕДСТВИЕ ЧЕГО ТЕСТ ТОЛЬКО ОТНОСИТЕЛЬНОЙ
C ОШИБКИ НЕ ПРОХОДИТ. ДЛЯ ПРОДОЛЖЕНИЯ НЕОБХОДИМО НЕНУЛЕВОЕ ЗНАЧЕ-
C НИЕ ПАРАМЕТРА GRX. ИСПОЛЬЗОВАНИЕ НА ОДИН ШАГ РЕЖИМА ПОШАГО-
C ВОГО ИНТЕГРИРОВАНИЯ ЯВЛЯЕТСЯ РАЗУМНЫМ ВЫХОДОМ ИЗ ПОЛОЖЕНИЯ.
C =6 -ИНТЕГРИРОВАНИЕ НЕ БЫЛО ЗАКОНЧЕНО ИЗ-ЗА ТОГО, ЧТО ТРЕБУЕМАЯ
C ТОЧНОСТЬ НЕ МОГЛА БЫТЬ ДОСТИГНУТА ДАЖЕ ПРИ НАИМЕНЬШЕЙ ДОПУСТИ-
C МОЙ ВЕЛИЧИНЕ ШАГА.ПОЛЬЗОВАТЕЛЬ ДОЛЖЕН УВЕЛИЧИТЬ ГРАНИЦУ ПОГРЕШ-
C НОСТИ, ПРЕЖДЕ ЧЕМ МОЖНО БУДЕТ ПОПЫТАТЬСЯ ПРОДОЛЖАТЬ ИНТЕГРИРО-
C ВАНИЕ.
C =7 - ПО ВСЕЙ ВИДИМОСТИ, DRKF45 НЕЭФФЕКТИВНА ПРИ РЕШЕНИИ ЭТОЙ
C ЗАДАЧИ. СЛИШКОМ БОЛЬШОЕ ЧИСЛО ТРЕБУЕМЫХ ВЫХОДНЫХ ТОЧЕК
C ПРЕПЯТСТВУЕТ ВЫБОРУ ЕСТЕСТВЕННОЙ ВЕЛИЧИНЫ ШАГА. СЛЕДУЕТ
C ИСПОЛЬЗОВАТЬ РЕЖИМ ПОШАГОВОГО ИНТЕГРИРОВАНИЯ.
C =8 - НЕПРАВИЛЬНОЕ ЗАДАНИЕ ВХОДНЫХ ПАРАМЕТРОВ.ЭТО ЗНАЧЕНИЕ
C ПОЯВЛЯЕТСЯ, ЕСЛИ ДОПУЩЕНА ОДНА ИЗ СЛЕДУЮЩИХ ОШИБОК:
C KDU.LE.0
C T=TOUT И IFLAG.NE.+1 ИЛИ -1
C GORX ИЛИ GRX.LT.0.
C IFLAG.EQ.0 ИЛИ .LT.-2 ИЛИ .GT.8
C WORK(*), IWORK(*) - ИНФОРМАЦИЯ, КОТОРАЯ ОБЫЧНО НЕ ПРЕДСТАВЛЯЕТ
C ИНТЕРЕСА ДЛЯ ПОЛЬЗОВАТЕЛЯ,НО НЕОБХОДИМА ПРИ ПОСЛЕДУЮЩИХ ВЫ-
C ЗОВАХ. WORK(1), . . . ,WORK(KDU+1) СОДЕРЖАТ ПЕРВЫЕ ПРОИЗВОДНЫЕ
C ВЕКТОРА РЕШЕНИЯ X В ТОЧКЕ T. WORK(KDU+1) ХРАНИТ ВЕЛИЧИНУ
C ШАГА H, С КОТОРОЙ МОЖНО ПОПЫТАТЬСЯ ПРОВЕСТИ СЛЕДУЮЩИЙ ШАГ.
C В IWORK(1) СОДЕРЖИТСЯ СЧЕТЧИК ЧИСЛА ВЫЧИСЛЕНИЙ ПРОИЗВОДНЫХ.
C
C ПОСЛЕДУЮЩИЕ ОБРАЩЕНИЯ К DRKF45
C НА ВЫХОДЕ ПОДПРОГРАММЫ DRKF45 ИМЕЕТСЯ ВСЯ ИНФОРМАЦИЯ, НЕОБХОДИ-
C МАЯ ДЛЯ ПРОДОЛЖЕНИЯ ИНТЕГРИРОВАНИЯ. ЕСЛИ ПРИ ИНТЕГРИРОВАНИИ
C ДОСТИГНУТО TOUT, ТО ПОЛЬЗОВАТЕЛЮ ДОСТАТОЧНО ОПРЕДЕЛИТЬ НОВОЕ
C ЗНАЧЕНИЕ TOUT И СНОВА ОБРАТИТЬСЯ К DRKF45. В РЕЖИМЕ ПОШАГОВОГО
C ИНТЕГРИРОВАНИЯ (IFLAG=-2) ПОЛЬЗОВАТЕЛЬ ДОЛЖЕН ИМЕТЬ В ВИДУ,
C ЧТО КАЖДЫЙ ШАГ ВЫПОЛНЯЕТСЯ В НАПРАВЛЕНИИ ТЕКУЩЕГО ЗНАЧЕНИЯ
C TOUT (СИГНАЛИЗИРУЕМОМ ИЗМЕНЕНИЕМ IFLAG НА 2). ПОЛЬЗОВАТЕЛЬ
C ДОЛЖЕН ЗАДАТЬ НОВОЕ ЗНАЧЕНИЕ TOUT И ПЕРЕОПРЕДЕЛИТЬ IFLAG
C НА -2, ЧТОБЫ ПРОДОЛЖАТЬ В РЕЖИМЕ ПОШАГОВОГО ИНТЕГРИРОВАНИЯ.
C ЕСЛИ ИНТЕГРИРОВАНИЕ НЕ БЫЛО ЗАКОНЧЕНО, НО ПОЛЬЗОВАТЕЛЬ ХОЧЕТ
C ПРОДОЛЖАТЬ (СЛУЧАИ IFLAG=3,4), ОН ПОПРОСТУ СНОВА ОБРАЩАЕТСЯ К
C DRKF45.
C ПРИ IFLAG=3 ПАРАМЕТР GORX БЫЛ ИЗМЕНЕН НАДЛЕЖАЩИМ ДЛЯ
C ПРОДОЛЖЕНИЯ ИНТЕГРИРОВАНИЯ ОБРАЗОМ. В СЛУЧАЕ IFLAG=4 СЧЕТЧИК
C ЧИСЛА ЗНАЧЕНИЙ ФУНКЦИИ БУДЕТ ПЕРЕОПРЕДЕЛЕН НА 0, И БУДУТ
C РАЗРЕШЕНЫ ЕЩЕ 3000 ВЫЧИСЛЕНИЙ ФУНКЦИИ.

С ОДНАКО В СЛУЧАЕ IFLAG=5, ПРЕЖДЕ ЧЕМ МОЖНО БУДЕТ ПРОДОЛЖАТЬ
 С ИНТЕГРИРОВАНИЕ, ПОЛЬЗОВАТЕЛЬ ДОЛЖЕН СНАЧАЛА ИЗМЕНИТЬ КРИТЕРИЙ
 С ОШИБКИ, ЗАДАВ ПОЛОЖИТЕЛЬНОЕ ЗНАЧЕНИЕ ДЛЯ GRX. ЕСЛИ ОН НЕ
 С СДЕЛАЕТ ЭТОГО, ВЫПОЛНЕНИЕ ПРОГРАММЫ БУДЕТ ПРЕКРАЩЕНО. ТОЧНО
 С ТАК ЖЕ, В СЛУЧАЕ IFLAG=6, ПРЕЖДЕ ЧЕМ ПРОДОЛЖАТЬ ИНТЕГРИРОВА-
 С НИЕ, ПОЛЬЗОВАТЕЛЮ НЕОБХОДИМО ПЕРЕОПРЕДЕЛИТЬ IFLAG НА 2 (ИЛИ -2,
 С ЕСЛИ ИСПОЛЬЗУЕТСЯ РЕЖИМ ПОШАГОВОГО ИНТЕГРИРОВАНИЯ) И УВЕЛИЧИТЬ
 С ЗНАЧЕНИЕ ДЛЯ GRX ЛИБО GORX, ЛИБО И ДЛЯ ТОГО , И ДЛЯ ДРУГОГО.
 С ЕСЛИ ЭТО НЕ БУДЕТ СДЕЛАНО, ВЫПОЛНЕНИЕ ПРОГРАММЫ ПРЕКРАЩАЕТСЯ.
 С ПОЯВЛЕНИЕ IFLAG=6 УКАЗЫВАЕТ НА НЕРЕГУЛЯРНОСТЬ (РЕШЕНИЕ ВЫСТРО
 С МЕНЯЕТСЯ ИЛИ, ВОЗМОЖНО, ИМЕЕТСЯ ОСОБЕННОСТЬ), И ЧАСТО В ПОДОВНЫХ
 С СЛУЧАЯХ НЕ ИМЕЕТ СМЫСЛА ПРОДОЛЖАТЬ ИНТЕГРИРОВАНИЕ.
 С ЕСЛИ ПОЛУЧЕНО ЗНАЧЕНИЕ IFLAG=7, ТО ПОЛЬЗОВАТЕЛЬ ДОЛЖЕН ПЕРЕЙТИ
 С К РЕЖИМУ ПОШАГОВОГО ИНТЕГРИРОВАНИЯ С ВЕЛИЧИНОЙ ШАГА, ОПРЕДЕЛЯЕ-
 С МОЙ ПРОГРАММОЙ, ИЛИ РАССМОТРЕТЬ ВОЗМОЖНОСТЬ ПЕРЕХОДА НА ПРОГРАМ-
 С МЫ МЕТОДОВ АДАМСА. ЕСЛИ ВСЕ ЖЕ ПОЛЬЗОВАТЕЛЬ ХОЧЕТ ПРОДОЛЖАТЬ
 С ИНТЕГРИРОВАНИЕ ПО ПОДПРОГРАММЕ DRKF45, ОН ДОЛЖЕН ДО НОВОГО
 С ОБРАЩЕНИЯ К НЕЙ ПЕРЕОПРЕДЕЛИТЬ IFLAG НА 2. В ПРОТИВНОМ СЛУЧАЕ
 С ВЫПОЛНЕНИЕ ПРОГРАММЫ БУДЕТ ПРЕКРАЩЕНО.
 С ЕСЛИ ПОЛУЧЕНО ЗНАЧЕНИЕ IFLAG=8, ТО ИНТЕГРИРОВАНИЕ НЕЛЬЗЯ ПРОДОЛ-
 С ЖАТЬ, ПОКА НЕ БУДУТ ИСПРАВЛЕНЫ ОШИБОЧНЫЕ ВХОДНЫЕ ПАРАМЕТРЫ.
 С НУЖНО ОТМЕТИТЬ, ЧТО МАССИВЫ WORK И IWORK СОДЕРЖАТ ИНФОРМАЦИЮ,
 С НЕОБХОДИМУЮ ДЛЯ ДАЛЬНЕЙШЕГО ИНТЕГРИРОВАНИЯ. ПОЭТОМУ В ЭТИ
 С МАССИВЫ НЕЛЬЗЯ ВНОСИТЬ ИЗМЕНЕНИЙ.

```
INTEGER KDU, IFLAG, IWORK (5)
REAL *8 X (KDU), T, TOUT, GORX, GRX, WORK (1)
```

С ЕСЛИ ТРАНСЛЯТОР ПРОВЕРЯЕТ ИНДЕКСЫ, ТО ЗАМЕНИТЬ WORK(1) НА
 С WORK(3+6*KDU) .

```
EXTERNAL SDU1P
INTEGER K1, K2, K3, K4, K5, K6, K1M
```

С ВЫЧИСЛИТЬ ИНДЕКСЫ ДЛЯ РАСЩЕПЛЕНИЯ РАБОЧЕГО МАССИВА:

```
K1M=KDU+1
K1=K1M+1
K2=K1+KDU
K3=K2+KDU
K4=K3+KDU
K5=K4+KDU
K6=K5+KDU
```

С ЭТА ПРОМЕЖУТОЧНАЯ ПРОГРАММА ПРОСТО СОКРАЩАЕТ ДЛЯ ПОЛЬЗОВАТЕЛЯ
 С ДЛИННЫЙ СПИСОК ВЫЗОВА ПУТЕМ РАСЩЕПЛЕНИЯ ДВУХ РАБОЧИХ МАССИВОВ.
 С ЕСЛИ ЭТО НЕ СОВМЕСТИМО С ТРАНСЛЯТОРОМ, КОТОРЫЙ ИМЕЕТСЯ
 С В РАСПОРЯЖЕНИИ ПОЛЬЗОВАТЕЛЯ, ТО ОН ДОЛЖЕН ОБРАЩАТЬСЯ
 С НЕПОСРЕДСТВЕННО К ПОДПРОГРАММЕ RKFS.

```

CALL RKFS (SDU1P, KDU, X, T, TOUT, GOPX, GPX, IFLAG,
1  WORK (1), WORK (K1M), WORK (K1), WORK (K2), WORK (K3),
2  WORK (K4), WORK (K5), WORK (K6), WORK (K6+1), IWORK (1),
3  IWORK (2), IWORK (3), IWORK (4), IWORK (5))
RETURN
END
SUBROUTINE RKFS (SDU1P, KDU, X, T, TOUT, GOPX, GPX, IFLAG,
1  XP, H, F1, F2, F3, F4, F5, SAVRE, SAVAE, NFE, KOP, INIT, JFLAG, KFLAG)
C  RKFS ИНТЕГРИРУЕТ СИСТЕМУ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕ-
C  НИЙ ПЕРВОГО ПОРЯДКА МЕТОДОМ РУНГЕ-КУТТА-ФЕЛЬБЕРГА 4-5-ГО ПОРЯД-
C  КА (СМ. КОММЕНТАРИЙ К DRKF45). МАССИВЫ XP, F1, F2, F3, F4, И F5
C  (РАЗМЕРНОСТИ ПО КРАЙНЕЙ МЕРЕ KDU) И ПЕРЕМЕННЫЕ H, SAVRE, SAVAE,
C  NFE, KOP, INIT, JFLAG И KFLAG ИСПОЛЬЗУЮТСЯ ВНУТРИ ПРОГРАММЫ
C  И ВЫНЕСЕНЫ В СПИСОК ВЫЗОВА, ЧТОБЫ СОХРАНИТЬ ИХ ОПРЕДЕЛЕННОСТЬ
C  ПРИ ПОВТОРНОМ ОБРАЩЕНИИ. ПОЭТОМУ ИХ ЗНАЧЕНИЯ НЕ ДОЛЖНЫ
C  ИЗМЕНЯТЬСЯ ПОЛЬЗОВАТЕЛЕМ.
C  ВОЗМОЖНЫЙ ИНТЕРЕС ПРЕДСТАВЛЯЮТ ПАРАМЕТРЫ:
C  XP - ПРОИЗВОДНАЯ ВЕКТОРА РЕШЕНИЯ В ТОЧКЕ T;
C  H - ПРЕДПОЛАГАЕМЫЙ РАЗМЕР ШАГА ДЛЯ ОЧЕРЕДНОГО ЭТАПА;
C  NFE - СЧЕТЧИК ЧИСЛА ВЫЧИСЛЕНИЙ ФУНКЦИИ.
C
C  LOGICAL HFAILD, OUTPUT
C  INTEGER KDU, IFLAG, NFE, KOP, INIT, JFLAG, KFLAG
C  REAL *8 X (KDU), T, TOUT, GOPX, GPX, H, XP (KDU), F1 (KDU), F2 (KDU),
1  F3 (KDU), F4 (KDU), F5 (KDU), SAVRE, SAVAE
C  EXTERNAL SDU1P
C  REAL *8 A, AE, DT, EE, EEOET, ESTTOL, ET, HMIN, REMIN, RER, S, SCALE,
1  TOL, TOLN, U26, EPSP1, EPS, YPK
C  INTEGER K, MAXNFE, MFLAG
C  REAL AMAX1, AMIN1
C
C  REMIN - ЭТО МИНИМАЛЬНОЕ ДОПУСТИМОЕ ЗНАЧЕНИЕ ДЛЯ GOPX. ПОПЫТКИ
C  ПОЛУЧИТЬ ПО ЭТОЙ ПОДПРОГРАММЕ БОЛЕЕ ВЫСОКУЮ ТОЧНОСТЬ ОБЫЧНО
C  СТОЯТ ОЧЕНЬ ДОРОГО И ЗАЧАСТУЮ БЕЗУСПЕШНЫ.
C  DATA REMIN/1.D-12/
C  СТОИМОСТЬ СЧЕТА КОНТРОЛИРУЕТСЯ ТРЕБОВАНИЕМ, ЧТОБЫ КОЛИЧЕСТВО
C  ВЫЧИСЛЕНИЙ ФУНКЦИИ БЫЛО ОГРАНИЧЕНО ВЕЛИЧИНОЙ, ПРИВЛИЗИТЕЛЬНО
C  РАВНОЙ ЗНАЧЕНИЮ ПАРАМЕТРА MAXNFE. ПРИНЯТОЕ ЗДЕСЬ ЗНАЧЕНИЕ
C  ПРИМЕРНО СООТВЕТСТВУЕТ 500 ШАГАМ.
C  DATA MAXNFE/3000/
C  ПРОВЕРИТЬ ВХОДНЫЕ ПАРАМЕТРЫ:
C  IF (KDU.LT.1) GO TO 10
C  IF ((GOPX.LT.0.0D0).OR. (GPX.LT.0.0D0)) GO TO 10
C  MFLAG=IABS (IFLAG)
C  IF ((MFLAG.EQ.0).OR. (MFLAG.GT.8)) GO TO 10
C  IF (MFLAG.NE.1) GO TO 20

```

```

C ПЕРВЫЙ ВЫЗОВ, ВЫЧИСЛИТЬ МАШИННОЕ ЭПСИЛОН:
  EPS=1.0D0
  5  EPS=EPS/2.0D0
    EPSP1=EPS+1.0D0
    IF (EPSP1.GT.1.0) GO TO 5
    U26=26.0D0*EPS
    GO TO 50
C ОШИБКА ВО ВХОДНОЙ ИНФОРМАЦИИ
  10 IFLAG=8
    RETURN
C ПРОВЕРИТЬ ВОЗМОЖНОСТЬ ПРОДОЛЖЕНИЯ
C
  20 IF ((T.EQ.TOUT) .AND. (KFLAG.NE.3)) GO TO 10
    IF (MFLAG.NE.2) GO TO 25
C
C   IFLAG=+2 ИЛИ -2
   IF ((KFLAG.EQ.3) .OR. (INIT.EQ.0)) GO TO 45
   IF (KFLAG.EQ.4) GO TO 40
   IF ((KFLAG.EQ.5) .AND. (GPX.EQ.0.0D0)) GO TO 30
   IF ((KFLAG.EQ.6) .AND. (GOPX.LE.SAVRE) .AND. (GPX.LE.SAVAE))
1GO TO 30
   GO TO 50
C
C   IFLAG=3,4,5,6,7 ИЛИ 8
  25 IF (IFLAG.EQ.3) GO TO 45
    IF (IFLAG.EQ.4) GO TO 40
    IF ((IFLAG.EQ.5) .AND. (GPX.GT.0.0D0)) GO TO 45
C
C ИНТЕГРИРОВАНИЕ НЕЛЬЗЯ ПРОДОЛЖАТЬ, ПОСКОЛЬКУ ПОЛЬЗОВАТЕЛЬ НЕ ВЫ-
C ПОЛНИЛ ИНСТРУКЦИЙ, СООТВЕТСТВУЮЩИХ ЗНАЧЕНИЯМ IFLAG=5,6,7 ИЛИ 8.
  30 STOP
C
C ПЕРЕОПРЕДЕЛИТЬ СЧЕТЧИК ЧИСЛА ВЫЧИСЛЕНИЙ ФУНКЦИИ
  40 NFE=0
    IF (MFLAG.EQ.2) GO TO 50
C
C ПЕРЕОПРЕДЕЛИТЬ ЗНАЧЕНИЕ FLAG, УСТАНОВЛЕННОЕ ПРИ ПРЕДЫДУЩЕМ
C ОБРАЩЕНИИ:
  45 IFLAG=JFLAG
    IF (KFLAG.EQ.3) MFLAG=IABS (IFLAG)
C
C СОХРАНИТЬ ВХОДНОЕ ЗНАЧЕНИЕ IFLAG И УСТАНОВИТЬ ЗНАЧЕНИЕ KFLAG,
C СООТВЕТСТВУЮЩЕЕ ПРОДОЛЖЕНИЮ ДЛЯ БУДУЩЕЙ ВХОДНОЙ ПРОВЕРКИ:
  50 JFLAG=IFLAG
    KFLAG=0
C
C СОХРАНИТЬ ЗНАЧЕНИЯ GOPX И GPX ДЛЯ ВХОДНОЙ ПРОВЕРКИ ПРИ
C ПОСЛЕДУЮЩИХ ОБРАЩЕНИЯХ:

```

```

  SAVRE=GOPX
  SAVAE=GPX
C
C УСТАНОВИТЬ ЗНАЧЕНИЕ ГРАНИЦЫ ДЛЯ ОТНОСИТЕЛЬНОЙ ПОГРЕШНОСТИ, РАВ-
C НОЕ КАК МИНИМУМ 2*EPS+REMIN, ЧТОБЫ ИЗБЕЖАТЬ ТРУДНОСТЕЙ, СВЯЗАН-
C НЫХ С ТРЕБОВАНИЕМ НЕДОСТИЖИМОЙ ТОЧНОСТИ:
  RER=2.0D0*EPS+REMIN
  IF (GOPX.GE.RER) GO TO 55
C
C ЗАДАННАЯ ГРАНИЦА ОТНОСИТЕЛЬНОЙ ПОГРЕШНОСТИ СЛИШКОМ МАЛА:
  GOPX=RER
  IFLAG=3
  KFLAG=3
  RETURN
C
  55 DT=TOUT-T
C
  IF (MFLAG.EQ.1) GO TO 60
  IF (INIT.EQ.0) GO TO 65
  GO TO 80
C
C ПРИСВОЕНИЕ НАЧАЛЬНЫХ ЗНАЧЕНИЙ (ИНИЦИИРОВАНИЕ) - УСТАНОВИТЬ
C ЗНАЧЕНИЕ УКАЗАТЕЛЯ ОКОНЧАНИЯ ИНИЦИИРОВАНИЯ INIT;
C УСТАНОВИТЬ ЗНАЧЕНИЕ УКАЗАТЕЛЯ СЛИШКОМ ВОЛЬШОГО ЗАТРЕБОВАННОГО
C ЧИСЛА ВЫХОДНЫХ ТОЧЕК KOP;
C ВЫЧИСЛИТЬ НАЧАЛЬНЫЕ ПРОИЗВОДНЫЕ;
C УСТАНОВИТЬ ЗНАЧЕНИЕ СЧЕТЧИКА ЧИСЛА ВЫЧИСЛЕНИЙ ФУНКЦИИ NFE;
C ОЦЕНИТЬ НАЧАЛЬНУЮ ВЕЛИЧИНУ ШАГА.
C
  60 INIT=0
    KOP=0
C
  A=T
  CALL SDU1P (A,X,XP)
  NFE=1
  IF (T.NE.TOUT) GO TO 65
  IFLAG=2
  RETURN
C
  65 INIT=1
    H=DABS (DT)
    TOLN=0.D0
    DO 70 K=1,KDU
      TOL=GOPX*DABS (X(K)) +GPX
      IF (TOL.LE.0.D0) GO TO 70
    TOLN=TOL

```

```

        YPK=DABS (XP (K))
        IF (YPK*H**5.GT.TOL) H=(TOL/YPK)**0.2D0
70  CONTINUE
        IF (TOLN.LE.0.0D0) H=0.0D0
        H=DMAX1 (H,U26*DMAX1 (DABS (T),DABS (DT)))
        JFLAG=ISIGN (2,JFLAG)
C
C  ПРИСВОИТЬ ВЕЛИЧИНЕ ШАГА ЗНАК, СООТВЕТСТВУЮЩИЙ ИНТЕГРИРОВАНИЮ В
C  НАПРАВЛЕНИИ ОТ Т К ТOUT:
        80  H=DSIGN (H,DT)
C
C  ПРОВЕРКА, НАСКОЛЬКО СЕРЬЕЗНО ВЛИЯНИЕ НА DRKF45 СЛИШКОМ БОЛЬШОГО
C  ЗАТРЕБОВАННОГО ЧИСЛА ВЫХОДНЫХ ТОЧЕК:
C
        IF (DABS (H).GE.2.0D0*DABS (DT)) KOP=KOP+1
        IF (KOP.NE.100) GO TO 85
C  ЧРЕЗМЕРНАЯ ЧАСТОТА ВЫХОДОВ
        KOP=0
        IFLAG=7
        RETURN
C
        85  IF (DABS (DT).GT.U26*DABS (T)) GO TO 95
C
C  ЕСЛИ ОЧЕНЬ ВЛИЗКО К ТОЧКЕ ВЫХОДА, ПРОЭКСТРАПОЛИРОВАТЬ И ВЕР-
C  НУТЬСЯ ПО МЕСТУ ВЫЗОВА:
        DO 90 K=1,KDU
        90  X (K)=X (K)+DT*XP (K)
        A=TOUT
        CALL SDU1P (A,X,XP)
        NFE=NFE+1
        GO TO 300
C
C  ПРИСВОИТЬ НАЧАЛЬНОЕ ЗНАЧЕНИЕ ИНДИКАТОРУ ТОЧКИ ВЫХОДА:
        95  OUTPUT=.FALSE.
C
C  ЧТОВЫ ИЗБЕЖАТЬ НЕОПРАВДАННОГО МАШИННОГО НУЛЯ ПРИ ВЫЧИСЛЕНИИ
C  ФУНКЦИИ ОТ ГРАНИЦ ПОГРЕШНОСТЕЙ,ПРОМАСШТАБИРОВАТЬ ЭТИ ГРАНИЦЫ:
        SCALE=2.0D0/GOPX
        AE=SCALE*GFX
C
C  ПОШАГОВОЕ ИНТЕГРИРОВАНИЕ
C
        100 HFAILED=.FALSE.

```

```
C
C УСТАНОВИТЬ НАИМЕНЬШУЮ ДОПУСТИМУЮ ВЕЛИЧИНУ ШАГА:
      HMIN=U26*DABS (T)
C
C ИСПРАВИТЬ ПРИ НЕОБХОДИМОСТИ ВЕЛИЧИНУ ШАГА, ЧТОБЫ ДОСТИГНУТЬ
C ТОЧКИ ВЫХОДА. РАССЧИТАТЬ НА ДВА ШАГА ВПЕРЕД, ЧТОБЫ ИЗБЕЖАТЬ
C СЛИШКОМ РЕЗКИХ ИЗМЕНЕНИЙ В ВЕЛИЧИНЕ ШАГА И ТЕМ САМЫМ УМЕНЬШИТЬ
C ВЛИЯНИЕ ВЫХОДНЫХ ТОЧЕК НА ПРОГРАММУ.
      DT=TOUT-T
      IF (DABS (DT) .GE.2.0D0*DABS (H) ) GO TO 200
      IF (DABS (DT) .GT.DABS (H) ) GO TO 150
C
C СЛЕДУЮЩИЙ УСПЕШНЫЙ ШАГ ЗАВЕРШИТ ИНТЕГРИРОВАНИЕ ДО УКАЗАННОЙ
C ТОЧКИ ВЫХОДА:
      OUTPUT=.TRUE.
      H=DT
      GO TO 200
150 H=0.5D0*DT
C
C ВНУТРЕННИЙ ОДНОШАГОВЫЙ ИНТЕГРАТОР
C ГРАНИЦЫ ПОГРЕШНОСТЕЙ ВЫЛИ ПРОМАСШТАБИРОВАНЫ, ЧТОБЫ ИЗБЕЖАТЬ
C НЕОПРАВДААННОГО МАШИНОГО НУЛЯ ПРИ ВЫЧИСЛЕНИИ ФУНКЦИИ ЕТ ОТ НИХ.
C ЧТОБЫ ИЗБЕЖАТЬ ОБРАЩЕНИЯ В НУЛЬ ЗНАМЕНАТЕЛЯ В ТЕСТЕ, ОТНОСИ-
C ТЕЛЬНАЯ ОШИБКА ИЗМЕРЯЕТСЯ ПО ОТНОШЕНИЮ К СРЕДНЕМУ ИЗ ВЕЛИЧИН
C РЕШЕНИЯ В НАЧАЛЕ И КОНЦЕ ШАГА.
C В ФОРМУЛЕ, ОЦЕНИВАЮЩЕЙ ОШИБКУ, ПРОИЗВЕДЕНА ГРУППИРОВКА СЛАГАЕ-
C МЫХ, УМЕНЬШАЮЩАЯ ПОТЕРЮ ВЕРНЫХ ЗНАКОВ.
C ЧТОБЫ РАЗЛИЧАТЬ МЕЖДУ СОВОЙ РАЗНЫЕ АРГУМЕНТЫ, ДЛЯ Н НЕ ДОПУСКА-
C ЮТСЯ ЗНАЧЕНИЯ, МЕНЬШИЕ УМНОЖЕННОЙ НА 26 ОШИБКИ ОКРУГЛЕНИЯ В Т.
C ВВЕДЕНЫ ПРАКТИЧЕСКИЕ ОГРАНИЧЕНИЯ НА СКОРОСТЬ ИЗМЕНЕНИЯ ВЕЛИЧИНЫ
C ШАГА, ЧТОБЫ СГЛАДИТЬ ПРОЦЕСС ВЫБОРА ЭТОЙ ВЕЛИЧИНЫ И ИЗБЕЖАТЬ
C ЧРЕЗМЕРНОГО ЕЕ РАЗВРОСА В ЗАДАЧАХ С НАРУШЕНИЕМ НЕПРЕРЫВНОСТИ.
C ИЗ ПРЕДОСТОРОЖНОСТИ ПРОГРАММА БЕРЕТ 9/10 ОТ ТОЙ ВЕЛИЧИНЫ ШАГА,
C КОТОРАЯ НУЖНА ПО ЕЕ ОЦЕНКЕ.
C ЕСЛИ НА ДАННОМ ШАГЕ БЫЛА НЕУДАЧНАЯ ПОПЫТКА, ТО ПРИ ПЛАНИРОВАНИИ
C СЛЕДУЮЩЕГО УВЕЛИЧЕНИЕ ДЛИНЫ ШАГА НЕ ДОПУСКАЕТСЯ. ЭТО ПОВЫШАЕТ
C ЭФФЕКТИВНОСТЬ ПРОГРАММЫ ДЛЯ ЗАДАЧ С РАЗРЫВАМИ И В ОБЩЕМ СЛУЧАЕ,
C ПОСКОЛЬКУ ИСПОЛЬЗУЕТСЯ ЛОКАЛЬНАЯ ЭКСТРАПОЛЯЦИЯ И ДОПОЛНИТЕЛЬНАЯ
C ПРЕДОСТОРОЖНОСТЬ КАЖЕТСЯ ОПРАВДААННОЙ.
C
C ПРОВЕРИТЬ ЧИСЛО ВЫЧИСЛЕНИЙ ПРОИЗВОДНЫХ, ЕСЛИ ОНО НЕ ПРЕВЫШАЕТ
C УСТАНОВЛЕННОГО ПРЕДЕЛА, ПОПРОБОВАТЬ ПРОДОЛЖИТЬ ИНТЕГРИРОВАНИЕ
C С Т ДО Т+Н:
200 IF (NFE.LE.MAXNFE) GO TO 220
```

```

С
С СЛИШКОМ БОЛЬШАЯ РАБОТА.
  IFLAG=4
  KFLAG=4
  RETURN

С
С ПРОДОЛЖИТЬ ПРИБЛИЖЕННОЕ РЕШЕНИЕ НА ОДИН ШАГ ДЛИНЫ Н:
220 CALL FENL(SDU1P,KDU,X,T,H,XP,F1,F2,F3,F4,F5,F1)
  NFE=NFE+5

С
С ВЫЧИСЛИТЬ И СРАВНИТЬ ДОПУСТИМЫЕ ГРАНИЦЫ И ОЦЕНКИ ЛОКАЛЬНОЙ
С ОШИБКИ, А ЗАТЕМ СНЯТЬ МАСШТАБИРОВАНИЕ ГРАНИЦ. ЗАМЕЬТЕ, ЧТО
С ОТНОСИТЕЛЬНАЯ ОШИБКА ИЗМЕРЯЕТСЯ ПО ОТНОШЕНИЮ К СРЕДНЕМУ ИЗ
С ВЕЛИЧИН РЕШЕНИЯ В НАЧАЛЕ И В КОНЦЕ ШАГА.
  ЕЕОЕТ=0.0D0
  DO 250 K=1,KDU
    ET=DABS(X(K))+DABS(F1(K))+AE
    IF(ET.GT.0.0D0) GO TO 240
С НЕПРАВИЛЬНАЯ ГРАНИЦА ПОГРЕШНОСТИ
  IFLAG=5
  RETURN
240 EE=DABS((-2090.D0*XP(K)+(21970.D0*F3(K)-15048.D0*F4(K)))
  1 + (22528.D0*F2(K)-27360.D0*F5(K)))
250 ЕЕОЕТ=DMAX1(ЕЕОЕТ,ЕЕ/ЕТ)

С
  ESTTOL=DABS(H)*ЕЕОЕТ*SCALE/752400.0D0
  IF(ESTTOL.LE.1.0D0) GO TO 260
С НЕУДАЧНЫЙ ШАГ
С УМЕНЬШИТЬ ВЕЛИЧИНУ ШАГА И СНОВА ПОПРОБОВАТЬ
С УМЕНЬШЕНИЕ ОГРАНИЧИВАЕТСЯ СНИЗУ МНОЖИТЕЛЕМ 1/10
  NFAILD=.TRUE.
  OUTPUT=.FALSE.
  S=0.1D0
  IF(ESTTOL.LT.59049.0D0) S=0.9D0/ESTTOL**0.2D0
  H=S*H
  IF(DABS(H).GT.HMIN) GO TO 200
С ЗАДАННАЯ ГРАНИЦА ОШИБКИ НЕДОСТИЖИМА ДАЖЕ ПРИ НАИМЕНЬШЕЙ
С ДОПУСТИМОЙ ВЕЛИЧИНЕ ШАГА:
  IFLAG=6
  KFLAG=6
  RETURN

С
С УСПЕШНЫЙ ШАГ
С ПОМЕСТИТЬ В МАССИВ X РЕШЕНИЕ В ТОЧКЕ Т+Н И ВЫЧИСЛИТЬ
С ПРОИЗВОДНЫЕ В ЭТОЙ ТОЧКЕ:
260 T=T+H

```

```

DO 270 K=1, KDU
270   X(K)=F1(K)
      A=T
      CALL SDU1P(A, X, XP)
      NFE=NFE+1
C
C ВЫБРАТЬ ВЕЛИЧИНУ СЛЕДУЮЩЕГО ШАГА. УВЕЛИЧЕНИЕ ОГРАНИЧЕНО
C МНОЖИТЕЛЕМ 5. ЕСЛИ НА ДАННОМ ШАГЕ БЫЛА НЕУДАЧНАЯ ПОПЫТКА, ТО
C ДЛЯ СЛЕДУЮЩЕГО НЕ ДОПУСКАЕТСЯ ВЫБОР БОЛЬШЕЙ ВЕЛИЧИНЫ ШАГА.
      S=5.0D0
      IF (ESTTOL.GT.1.889568D-4) S=0.9D0/ESTTOL**0.2D0
      IF (HFAILD) S=DMIN1(S, 1.0D0)
      H=DSIGN(DMAX1(S*DABS(H), HMIN), H)
C КОНЕЦ ОДНОШАГОВОГО ИНТЕГРАТОРА
C
C   НУЖНО ЛИ ДЕЛАТЬ ОЧЕРЕДНОЙ ШАГ
      IF (OUTPUT) GO TO 300
      IF (IFLAG.GT.0) GO TO 100
C ИНТЕГРИРОВАНИЕ УСПЕШНО ЗАВЕРШЕНО
C
C   РЕЖИМ ОДНОШАГОВОГО ИНТЕГРИРОВАНИЯ
      IFLAG=-2
      RETURN
C
C   РЕЖИМ ИНТЕГРИРОВАНИЯ НА ИНТЕРВАЛЕ
300 T=TOUT
      IFLAG=2
      RETURN
      END
      SUBROUTINE FENL(SDU1P, KDU, X, T, H, XP, F1, F2, F3, F4, F5, S)
C
C МЕТОД РУНГЕ-КУТТА-ФЕЛЬБЕРГА 4-5-ГО ПОРЯДКА
C ИНТЕГРИРУЕТ СИСТЕМУ ИЗ KDU ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ
C УРАВНЕНИЙ ПЕРВОГО ПОРЯДКА СЛЕДУЮЩЕГО ВИДА:
      dx(I)/dt=F(T, X(1), ..., X(KDU)),
C ГДЕ НАЧАЛЬНЫЕ ЗНАЧЕНИЯ X(I) И НАЧАЛЬНЫЕ ПРОИЗВОДНЫЕ XP(I) ЗАДА-
C НЫ В НАЧАЛЬНОЙ ТОЧКЕ T. FENL ПРОДОЛЖАЕТ РЕШЕНИЕ НА ФИКСИРОВАН-
C НЫЙ ШАГ H И ПОМЕЩАЕТ В МАССИВ S(I) ПРИВЛИЖЕНИЕ К РЕШЕНИЮ В ТОЧ-
C КЕ T+H, ИМЕЮЩЕЕ 5-Й ПОРЯДОК ТОЧНОСТИ (ЛОКАЛЬНЫЙ ПОРЯДОК =6).
C F1, ..., F5 - МАССИВЫ РАЗМЕРНОСТИ KDU, НЕОБХОДИМЫЕ ВНУТРИ
C ПОДПРОГРАММЫ.
C В ФОРМУЛАХ ПРОИЗВЕДЕНА ГРУППИРОВКА С ЦЕЛЬЮ УМЕНЬШИТЬ ПОТЕРЮ
C ВЕРНЫХ ЗНАКОВ.
C ЧТОБЫ МОЖНО БЫЛО РАЗЛИЧАТЬ РАЗНЫЕ НЕЗАВИСИМЫЕ АРГУМЕНТЫ, ПРИ
C ОВРАЩЕНИИ К FENL НЕ СЛЕДУЕТ ЗАДАВАТЬ ДЛЯ H ЗНАЧЕНИЕ, МЕНЬШЕЕ
C УМНОЖЕННОЙ НА 13 ОШИБКИ ОКРУГЛЕНИЯ В T.

```



```

C
    INTEGER KDU
    REAL *8 X(KDU), T, H, XP(KDU), F1(KDU), F2(KDU), F3(KDU),
1    F4(KDU), F5(KDU), S(KDU)
    REAL *8 CH
    INTEGER K

C
    CH=H/4.0D0
    DO 221 K=1, KDU
221  F5(K)=X(K)+CH*XP(K)
    CALL SDU1P(T+CH, F5, F1)

C
    CH=3.0D0*H/32.0D0
    DO 222 K=1, KDU
222  F5(K)=X(K)+CH*(XP(K)+3.0D0*F1(K))
    CALL SDU1P(T+3.0D0*H/8.0D0, F5, F2)

C
    CH=H/2197.0D0
    DO 223 K=1, KDU
223  F5(K)=X(K)+CH*(1932.0D0*XP(K)+(7296.0D0*F2(K)-
*      7200.0D0*F1(K)))
    CALL SDU1P(T+12.0D0*H/13.0D0, F5, F3)

C
    CH=H/4104.0D0
    DO 224 K=1, KDU
224  F5(K)=X(K)+CH*((8341.0D0*XP(K)-845.0D0*F3(K))+
*      (29440.0D0*F2(K)-32832.0D0*F1(K)))
    CALL SDU1P(T+H, F5, F4)

C
    CH=H/20520.0D0
    DO 225 K=1, KDU
225  F1(K)=X(K)+CH*((-6080.0D0*XP(K)+(9295.0D0*F3(K)-5643.0D0*
*      F4(K)))+(41040.0D0*F1(K)-28352.0D0*F2(K)))
    CALL SDU1P(T+H/2.0D0, F1, F5)

C
C    ВЫЧИСЛИТЬ ПРИВЛИЖЕННОЕ РЕШЕНИЕ В ТОЧКЕ T+H
    CH=H/7618050.0D0
    DO 230 K=1, KDU
230  S(K)=X(K)+CH*((902880.0D0*XP(K)+(3855735.0D0*F3(K)-
1    1371249.0D0*F4(K)))+(3953664.0D0*F2(K)+
2    277020.0D0*F5(K)))
    RETURN
    END

```

Приложение 3.

C	ПРОГРАММА 13.1	
C	С ИСПОЛЬЗОВАНИЕМ УДВОЕННОЙ ТОЧНОСТИ	
	IMPLICIT REAL *8 (A-H, O-Z)	9
	X (T)=4.D0*T	10
	Y (T)=16.D0*T*T-1.D0	11
	DATA TN, TK, HT, DT/0.D0, 1.D0, 0.05D0, 0.005D0/	20
	T=TN	21
	PRINT 15	30
15	FORMAT (6X, ' T', 12X, ' V', 12X, ' A', 10X, ' ATANG',	31
	* 10X, ' RO')	
20	VX=(X (T+DT)-X (T)) /DT	40
	VY=(Y (T+DT)-Y (T)) /DT	41
	V=DSQRT (VX*VX+VY*VY)	43
	AX=(X (T+DT)-2.D0*X (T)+X (T-DT)) / (DT*DT)	50
	AY=(Y (T+DT)-2.D0*Y (T)+Y (T-DT)) / (DT*DT)	51
	A=DSQRT (AX*AX+AY*AY)	53
	ATANG=(VX*AX+VY*AY) /V	60
	ANORM=DSQRT (A*A-ATANG*ATANG)	70
	RO=V*V/ANORM	76
	PRINT 40, T, V, A, ATANG, RO	80
40	FORMAT (5 (1X, G12.5))	81
	T=T+HT	90
	IF (T.LE.TK) GOTO 20	91
	STOP	100
	END	101
C	ПРОГРАММА 13.2 (ДЛЯ К-7)	
C	С ИСПОЛЬЗОВАНИЕМ УДВОЕННОЙ ТОЧНОСТИ	
	IMPLICIT REAL *8 (A-H, O-Z)	9
	DATA TN, TK, HT, DT/0.D0, 0.44444D0, 0.02222D0,	20
	* 0.00222D0/	
	T=TN	21
	PRINT 15	30
15	FORMAT (6X, ' T', 12X, ' V', 12X, ' A')	31
20	XT=DFX (T)	40N
	YT=DFY (T)	41N
	ZT=DFZ (T)	42N
	XTDT=DFX (T+DT)	43N
	YTDT=DFY (T+DT)	44N
	ZTDT=DFZ (T+DT)	45N
	VX=(XTDT-XT) /DT	40
	VY=(YTDT-YT) /DT	41
	VZ=(ZTDT-ZT) /DT	42
	V=DSQRT (VX*VX+VY*VY+VZ*VZ)	43
	AX=(XTDT-2.*XT+DFX (T-DT)) / (DT*DT)	50

AY=(YTDT-2.*YT+DFY(T-DT))/(DT*DT)	51
AZ=(ZTDT-2.*ZT+DFZ(T-DT))/(DT*DT)	52
A=DSQRT(AX*AX+AY*AY+AZ*AZ)	53
PRINT 40,T,V,A	80
40 FORMAT(3(1X,G12.5))	81
T=T+HT	90
IF(T.LE.TK) GOTO 20	91
STOP	100
END	101
REAL FUNCTION DFX*8(T)	110
REAL *8 PI,SR,VIE,T	111
COMMON PI,SR,VIE	115
PI=3.141592653589D0	117
SR=16.D0-8.D0*DCOS(3.D0*PI*T)	118
VIE=0.9D0*T*T-9.D0*T**3	119
DFX=-SR/2.D0*DSIN(VIE)	120
RETURN	130
END	131
REAL FUNCTION DFY*8(T)	140
REAL *8 PI,SR,VIE,T	141
COMMON PI,SR,VIE	145
DFY=SR/2.*DCOS(VIE)	150
RETURN	160
END	161
REAL FUNCTION DFZ*8(T)	170
REAL *8 PI,SR,VIE,T	171
COMMON PI,SR,VIE	175
DFZ=SR*DCOS(PI/6.)	180
RETURN	190
END	191
C	
ПРОГРАММА 13.3	
C	
С ИСПОЛЬЗОВАНИЕМ УДВОЕННОЙ ТОЧНОСТИ	
IMPLICIT REAL *8(A-H,O-Z)	9
READ *,NG,NZ,NW	20U
READ *,TN,TK,HT,DT	21U
T=TN	21
PRINT 5,NG,NZ,NW	25U
5 FORMAT(/5X,'ГРУППА',I8,5X,	26U
1 'ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ'/7X,'ПО КИНЕМАТИКЕ НОМЕР ',	
2 I2,5X,'ВАРИАНТ ',I2)	
IF((NZ.EQ.1).OR.(NZ.EQ.2)) PRINT 15	30.1U
15 FORMAT(6X,'T',12X,'V',12X,'A',10X,'ATANG',	31.1
* 10X,'RO')	
IF((NZ.EQ.7).OR.(NZ.EQ.10)) PRINT 17	30.2U
17 FORMAT(6X,'T',12X,'V',12X,'A')	31.2U

20 XT=DFX(T)	40N
YT=DFY(T)	41N
ZT=DFZ(T)	42N
XTDT=DFX(T+DT)	43N
YTDT=DFY(T+DT)	44N
ZTDT=DFZ(T+DT)	45N
VX=(XTDT-XT)/DT	40
VY=(YTDT-YT)/DT	41
VZ=(ZTDT-ZT)/DT	42
V=DSQRT(VX*VX+VY*VY+VZ*VZ)	43
AX=(XTDT-2.*XT+DFX(T-DT))/(DT*DT)	50
AY=(YTDT-2.*YT+DFY(T-DT))/(DT*DT)	51
AZ=(ZTDT-2.*ZT+DFZ(T-DT))/(DT*DT)	52
A=DSQRT(AX*AX+AY*AY+AZ*AZ)	53
IF((NZ.EQ.7).OR.(NZ.EQ.10)) GOTO 42	54U
ATANG=(VX*AX+VY*AY)/V	60.1
ANORM=DSQRT(A*A-ATANG*ATANG)	70.1
RO=V*V/ANORM	76.1
PRINT 40,T,V,A,ATANG,RO	80.1
40 FORMAT(5(1X,G12.5))	81.1
42 IF((NZ.EQ.7).OR.(NZ.EQ.10)) PRINT 40,T,V,A	82U
T=T+HT	90
IF(T.LE.TK) GOTO 20	91
STOP	100
END	101
REAL FUNCTION DFX*8(T)	110
IMPLICIT REAL *8(A-H,O-Z)	111
COMMON PI,SR,VIE	115
PI=3.141592653589D0	117
SR=16.D0-8.D0*DCOS(3.D0*PI*T)	118
VIE=.9D0*T*T-9.D0*T**3	119
DFX=-SR/2.D0*DSIN(VIE)	120
RETURN	130
END	131
REAL FUNCTION DFY*8(T)	140
IMPLICIT REAL *8(A-H,O-Z)	141
COMMON PI,SR,VIE	145
DFY=SR/2.D0*DCOS(VIE)	150
RETURN	160
END	161
REAL FUNCTION DFZ*8(T)	170
IMPLICIT REAL *8(A-H,O-Z)	171
COMMON PI,SR,VIE	175
DFZ=SR*DCOS(PI/6.D0)	180
RETURN	190
END	191

C		ПРОГРАММА 13.4	
C		С ИСПОЛЬЗОВАНИЕМ УДВОЕННОЙ ТОЧНОСТИ	
		IMPLICIT REAL *8 (A-H,O-Z)	9
		DIMENSION T(3),X(3),Y(3)	10S
		DATA TN,TK,HT,DT/0.D0,1.D0,0.05D0,0.005D0/	20.1
		T(1)=TN	21
		PRINT 15	30.1
15		FORMAT(6X,'T',12X,'V',12X,'A',10X,'ATANG',	31.1
	*	10X,'RO')	
20		T(2)=T(1)+DT	40S
		T(3)=T(1)-DT	41S
		DO 30 M=1,3	42S
		CALL DK1U (M,T,X,Y)	43S
30		CONTINUE	44S
		VX=(X(2)-X(1))/DT	40
		VY=(Y(2)-Y(1))/DT	41
		V=DSQRT(VX*VX+VY*VY)	43.1
		AX=(X(2)-2.*X(1)+X(3))/(DT*DT)	50
		AY=(Y(2)-2.*Y(1)+Y(3))/(DT*DT)	51
		A=DSQRT(AX*AX+AY*AY)	53.1
		ATANG=(VX*AX+VY*AY)/V	60.1
		ANORM=DSQRT(A*A-ATANG*ATANG)	70.1
		RO=V*V/ANORM	76.1
		PRINT 40,T(1),V,A,ATANG,RO	80
40		FORMAT(5(1X,G12.5))	81.1
		T(1)=T(1)+HT	90
		IF(T(1).LE.TK) GOTO 20	91
		STOP	100.1
		END	101.1
		SUBROUTINE DK1U(M,T,X,Y)	110
		IMPLICIT REAL *8 (A-H,O-Z)	111
		DIMENSION T(3),X(3),Y(3)	116
		X(M)=4.D0*T(M)	120
		Y(M)=16.D0*T(M)**2-1.D0	121
		RETURN	130
		END	131
C		ПРОГРАММА 13.5	
C		С ИСПОЛЬЗОВАНИЕМ УДВОЕННОЙ ТОЧНОСТИ	
		IMPLICIT REAL *8 (A-H,O-Z)	9
		DIMENSION T(3),X(3),Y(3),Z(3)	10
		READ *,NG,NZ,NW	20U
		READ *,TN,TK,HT,DT	21U
		T(1)=TN	21

	PRINT 5,NG,NZ,NW	25U
5	FORMAT (/5X, 'ГРУППА', I8, 5X,	26U
	1 'ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ'/7X, 'ПО КИНЕМАТИКЕ НОМЕР ',	
	2 I2, 5X, 'ВАРИАНТ ', I2)	
	IF((NZ.EQ.1).OR.(NZ.EQ.2)) PRINT 15	30.1U
15	FORMAT(6X, 'T', 12X, 'V', 12X, 'A', 10X, 'ATANG',	31.1
	* 10X, 'RO')	
	IF((NZ.EQ.7).OR.(NZ.EQ.10)) PRINT 17	30.2U
17	FORMAT(6X, 'T', 12X, 'V', 12X, 'A')	31.2U
20	T(2)=T(1)+DT	40S
	T(3)=T(1)-DT	41S
	DO 30 M=1, 3	42S
	CALL DKINU (M, T, X, Y, Z)	43S
30	CONTINUE	44S
	VX=(X(2)-X(1))/DT	40
	VY=(Y(2)-Y(1))/DT	41
	VZ=(Z(2)-Z(1))/DT	42
	V=DSQRT(VX*VX+VY*VY+VZ*VZ)	43
	AX=(X(2)-2.*X(1)+X(3))/(DT*DT)	50
	AY=(Y(2)-2.*Y(1)+Y(3))/(DT*DT)	51
	AZ=(Z(2)-2.*Z(1)+Z(3))/(DT*DT)	52
	A=DSQRT(AX*AX+AY*AY+AZ*AZ)	53
	IF((NZ.EQ.7).OR.(NZ.EQ.10)) GOTO 42	54U
	ATANG=(VX*AX+VY*AY)/V	60.1
	ANORM=DSQRT(A*A-ATANG*ATANG)	70.1
	RO=V*V/ANORM	76.1
	PRINT 40, T(1), V, A, ATANG, RO	80.1
40	FORMAT (5(1X,G12.5))	81.1
42	IF((NZ.EQ.7).OR.(NZ.EQ.10)) PRINT 40, T(1), V, A	82U
	T(1)=T(1)+HT	90
	IF(T(1).LE.TK) GOTO 20	91
	STOP	100
	END	101
	SUBROUTINE DKINU(M, T, X, Y, Z)	110
	IMPLICIT REAL *8(A-H, O-Z)	111
	DIMENSION T(3), X(3), Y(3), Z(3)	116
	PI=3.141592653589D0	117
	VIE=0.9D0*T(M)*T(M)-9.D0*T(M)**3	118
	SR=16.D0-8.D0*DCOS(3.D0*PI*T(M))	119
	X(M)=-SR/2.D0*DSIN(VIE)	120
	Y(M)=SR/2.D0*DCOS(VIE)	121
	Z(M)=SR*DCOS(PI/6.D0)	122
	RETURN	130
	END	131

```

C      П Р О Г Р А М М А 14.1
C      С ИСПОЛЬЗОВАНИЕМ УДВОЕННОЙ ТОЧНОСТИ
      IMPLICIT REAL *8 (A-H,O-Z)          9
      EXTERNAL DFX,DFY                    10
      DIMENSION VX(21),VY(21),V(21),AX(21),AY(21),
*      A(21),ATANG(21),ANORM(21),RO(21)  11
      DATA TN,TK,DT,NDIM/0.45D0,0.55D0,0.005D0,21/ 20
      T=TN                                 21
      M=1                                   22
      PRINT 15                              30
15     FORMAT (6X,'T',12X,'V',12X,'A',12X,'AT',
*          11X,'RO',15X,'IERX',10X,'IERY')  31
20     CALL DDCAR(T,DT,1,DFX,VX(M))        40
      CALL DDCAR(T,DT,1,DFY,VY(M))        41
      V(M)=DSQRT(VX(M)*VX(M)+VY(M)*VY(M)) 43
      T=T+DT                               44
      M=M+1                                 45
      IF(T.LE.TK) GOTO 20                  46
      CALL DDET3(DT,VX,AX,NDIM,IERX)      50
      CALL DDET3(DT,VY,AY,NDIM,IERY)      51
      DO 22 N=1,NDIM                       54
      A(N)=DSQRT(AX(N)*AX(N)+AY(N)*AY(N))  56
      ATANG(N)=(VX(N)*AX(N)+VY(N)*AY(N))/V(N) 60
      ANORM(N)=DSQRT(A(N)*A(N)-ATANG(N)*ATANG(N)) 70
      RO(N)=V(N)*V(N)/ANORM(N)            76
22     CONTINUE                           77
      T=TN                                 78
      DO 50 N=1,NDIM                       79
      PRINT 40,T,V(N),A(N),ATANG(N),RO(N)  80
40     FORMAT (5(1X,G12.5))                81
50     T=T+DT                              82
      PRINT 60,IERX,IERY                   83
60     FORMAT (5X,'КОДЫ ОШИБОК: IERX=',I2,3X,
*          'IERY=',I2)                    84
      STOP                                  100
      END                                   101
      BLOCK DATA                          103
      COMMON /AA/A1/BB/B,C                 105
      REAL *8 A1/4.D0/,B/16.D0/,C/1.D0/   107
      END                                   109
      REAL FUNCTION DFX*8(T)               110
      REAL *8 T,A                          111
      COMMON /AA/A                          115
      DFX=A*T                               120
      RETURN                                130
      END                                   131

```

```

      REAL FUNCTION DFY*8(T)               140
      REAL *8 T,B,C                        141
      COMMON /BB/B,C                       145
      DFY=B*T**2-C                          150
      RETURN                                160
      END                                   161
      REAL FUNCTION DFZ*8(T)               170
      DFZ=0.D0                             180
      RETURN                                190
      END                                   191

C      П Р О Г Р А М М А 14.2
C      С ИСПОЛЬЗОВАНИЕМ УДВОЕННОЙ ТОЧНОСТИ
      IMPLICIT REAL *8 (A-H,O-Z)          9
      EXTERNAL DFX,DFY,DFZ                10
      DIMENSION VX(21),VY(21),VZ(21),AX(21),AY(21),
*      AZ(21)                             11
      READ *,NG,NZ,NW                      20U
      READ *,TN,TK,DT,NDIM                 21U
      T=TN                                  21
      M=1                                   22
5     PRINT 6,NG,NZ,NW                     25U
6     FORMAT (/5X,'ГРУППА',I8,5X,
1     'ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ'/7X,'ПО КИНЕМАТИКЕ НОМЕР ',
2     I2,5X,'ВАРИАНТ ',I2)
      IF((NZ.EQ.1).OR.(NZ.EQ.2)) PRINT 15   30.1U
15     FORMAT (6X,'T',12X,'V',12X,'A',10X,'ATANG',
*      10X,'RO')                          31.1
      IF((NZ.EQ.7).OR.(NZ.EQ.10)) PRINT 17 30.2U
17     FORMAT (6X,'T',12X,'V',12X,'A')    31.2U
20     CALL DDCAR(T,DT,1,DFX,VX(M))        40
      CALL DDCAR(T,DT,1,DFY,VY(M))        41
      CALL DDCAR(T,DT,1,DFZ,VZ(M))        42
      T=T+DT                               44
      M=M+1                                 45
      IF(T.LE.TK) GOTO 20                  46
      CALL DDET3(DT,VX,AX,NDIM,IERX)      50
      CALL DDET3(DT,VY,AY,NDIM,IERY)      51
      CALL DDET3(DT,VZ,AZ,NDIM,IERZ)      52
      T=TN                                  53/78/
      DO 50 N=1,NDIM                       54
      V=DSQRT(VX(N)*VX(N)+VY(N)*VY(N)+VZ(N)*VZ(N)) 55/43/
      A=DSQRT(AX(N)*AX(N)+AY(N)*AY(N)+AZ(N)*AZ(N)) 56

```

IF((NZ.EQ.7).OR.(NZ.EQ.10)) GOTO 42	58U
ATANG=(VX(N)*AX(N)+VY(N)*AY(N))/V	60
ANORM=DSQRT(A*A-ATANG*ATANG)	70.1
RO=V*V/ANORM	76.1
PRINT 40,T,V,A,ATANG,RO	80.1
40 FORMAT (5(1X,G12.5))	81.1
42 IF((NZ.EQ.7).OR.(NZ.EQ.10)) PRINT 40,T,V,A	82U
50 T=T+DT	82
IF((NZ.EQ.1).OR.(NZ.EQ.2)) PRINT 60,IERX,IERY	83
60 FORMAT (5X,'КОДЫ ОШИБОК: IERX=',I2,3X,	84
* ' IERY=',I2)	
IF((NZ.EQ.7).OR.(NZ.EQ.10)) PRINT 61,IERX,	85
* IERY,IERZ	
61 FORMAT (5X,'КОДЫ ОШИБОК: IERX=',I2,3X,' IERY=',	86
* I2,3X,' IERZ=',I2)	
STOP	100
END	101
REAL FUNCTION DFX*8(T)	110
REAL *8 PI,SR,VIE,T	111
COMMON PI	115
PI=3.141592653589D0	117
SR=16.D0-8.D0*DCOS(3.D0*PI*T)	118
VIE=0.9D0*T*T-9.D0*T**3	119
DFX=-SR/2.D0*DSIN(VIE)	120
RETURN	130
END	131
REAL FUNCTION DFY*8(T)	140
REAL *8 PI,SR,VIE,T	141
COMMON PI	145
SR=16.D0-8.D0*DCOS(3.D0*PI*T)	148
VIE=0.9D0*T*T-9.D0*T**3	149
DFY=SR/2.*DCOS(VIE)	150
RETURN	160
END	161
REAL FUNCTION DFZ*8(T)	170
REAL *8 PI,SR,VIE,T	171
COMMON PI	175
SR=16.D0-8.D0*DCOS(3.D0*PI*T)	178
VIE=0.9D0*T*T-9.D0*T**3	179
DFZ=SR*DCOS(PI/6.D0)	180
RETURN	190
END	191

C	ПРОГРАММА 19.1 В УДВОЕННОЙ ТОЧНОСТИ	
	IMPLICIT REAL *8(A-H,O-Z)	9
	EXTERNAL SDU1P,OUTP	10
	DIMENSION PRMT(5),X(2),XP(2),AUX(16,2)	20
	COMMON TPR	30
	DATA PRMT/0.D0,1.2D0,0.01D0,0.001D0/	40
	DATA X/0.3D0,2.D0/	45
	DATA XP/2*0.5D0/	50
	TPR=0.D0	54
	PRINT 15	60
15	FORMAT(T4,'ВРЕМЯ, СЕК',T16,'КООРДИНАТА X, М',	61
*	T32,'СКОРОСТЬ XP, М/С')	
	CALL DRKGS(PRMT,X,XP,2,IH2,SDU1P,OUTP,AUX)	65
	STOP	98
	END	99
	SUBROUTINE SDU1P(T,X,XP)	100
	IMPLICIT REAL *8(A-H,O-Z)	109
	DIMENSION X(2),XP(2)	110
	DATA PI/3.141592653589D0/, ZM/0.01D0/, R/0.2D0/,	
*	C/1.D0/, ZL0/0.2D0/	120
	AL=PI/6.D0	130
	C1=-(C/ZM-PI*PI*(DSIN(AL))**2)	134
	C5=PI*PI*R*DSIN(AL)-9.81D0*DCOS(AL)+C*ZL0/ZM	136
	XP(1)=X(2)	140
	XP(2)=C1*X(1)+C5	150
	RETURN	196
	END	198
	SUBROUTINE OUTP(T,X,XP,IH2,KDU,PRMT)	200
	IMPLICIT REAL *8(A-H,O-Z)	209
	DIMENSION X(2),XP(2),PRMT(5)	210
	COMMON TPR	220
	IF(DABS(T-TPR)-0.0001D0) 20,20,40	230
20	PRINT 25,T,X(1),X(2)	240
25	FORMAT(2X,3(2X,G12.5))	241
	TPR=TPR+0.1D0	250
40	RETURN	260
	END	262
C	ПРОГРАММА 20.1 В УДВОЕННОЙ ТОЧНОСТИ	
	IMPLICIT REAL *8(A-H,O-Z)	9.1D
	EXTERNAL SDU1P,OUTP	10.1D
	DIMENSION PRMT(5),X(4),XP(4),AUX(16,4)	20
	COMMON TPR	30.1D

	DATA PRMT/0.D0,1.2D0,0.01D0,0.001D0/	40.1D
	DATA X/0.D0,10.D0,20.D0,0.D0/	45
	DATA XP/4*0.25D0/	50
	TPR=0.D0	54.1D
	PRINT 15	60.1D
15	FORMAT (T4,'ВРЕМЯ, СЕК',T21,'X, M',T35,'Z, M',	61
*	T49,'XP, M/C',T62,'ZP, M/C')	
	CALL DRKGS (PRMT,X,XP,4,IH2,SDU1P,OUTP,AUX)	65
	STOP	98.1D
	END	99.1D
	SUBROUTINE SDU1P (T,X,XP)	100.1D
	IMPLICIT REAL *8 (A-H,O-Z)	109.1D
	DIMENSION X(4), XP(4)	110
	XP(1)=X(3)	140
	XP(2)=X(4)	142
	XP(3)=-4.D0*X(1)-2.D0*X(3)	150
	XP(4)=-4.D0*X(2)-2.D0*X(4)-9.81D0	152
	RETURN	196.1D
	END	198.1D
	SUBROUTINE OUTP (T,X,XP,IH2,KDU,PRMT)	200.1D
	IMPLICIT REAL *8 (A-H,O-Z)	209.1D
	DIMENSION X(4),XP(4),PRMT(5)	210
	COMMON TPR	220.1
	IF (DABS (T-TPR)-0.0001D0) 20,20,40	230.1D
20	PRINT 25,T,(X(I),I=1,4)	240
25	FORMAT (2X,5(2X,G12.5))	241
	TPR=TPR+0.1D0	250.1D
40	RETURN	260.1D
	END	262.1D
C	ПРОГРАММА 20.2 В УДВОЕННОЙ ТОЧНОСТИ	
	IMPLICIT REAL *8 (A-H,O-Z)	9.1D
	EXTERNAL SDU1P,OUTP	10.1D
	DIMENSION PRMT(5),X(6),XP(6),AUX(16,6)	20
	COMMON TPR	30.1D
	DATA PRMT/0.D0,1.2D0,0.01D0,0.001D0/	40.1D
	DATA X/0.D0,0.D0,0.D0,1.D0,1.D0,2.D0/	45
	DATA XP/6*0.166666D0/	50
	TPR=0.D0	54.1D
	PRINT 15	60.1D
15	FORMAT (T4,'ВРЕМЯ, СЕК',T21,'X, M',T35,'Y, M',	61
*	T49,'Z, M',T62,'XP, M/C',T76,'YP, M/C',T90,'ZP, M/C')	
	CALL DRKGS (PRMT,X,XP,6,IH2,SDU1P,OUTP,AUX)	65
	STOP	98.1D
	END	99.1D

SUBROUTINE SDU1P(T,X,XP)	100.1D
IMPLICIT REAL *8(A-H,O-Z)	109.1D
DIMENSION X(6),XP(6)	110
XP(1)=X(4)	140
XP(2)=X(5)	142
XP(3)=X(6)	144
XP(4)=-0.5D0*X(4)	150
XP(5)=-0.5D0*X(5)	152
XP(6)=-0.5D0*X(6)+9.81D0	154
RETURN	196.1D
END	198.1D
SUBROUTINE OUTF(T,X,XP,IH2,KDU,PRMT)	200.1D
IMPLICIT REAL *8(A-H,O-Z)	209.1D
DIMENSION X(6),XP(6),PRMT(5)	210
COMMON TPR	220.1D
IF(DABS(T-TPR)-0.0001D0) 20,20,40	230.1D
20 PRINT 25,T,(X(I),I=1,6)	240
25 FORMAT(2X,7(2X,G12.5))	241
TPR=TPR+0.1D0	250.1D
40 RETURN	260.1D
END	262.1D

C	ПРОГРАММА 22.1 В УДВОЕННОЙ ТОЧНОСТИ С DRKF45	
	IMPLICIT REAL *8(A-H,O-Z)	9.1D
	EXTERNAL SDU1P	10.4
	DIMENSION X(2),IWORK(5),WORK(15)	20.4
	DATA T,TK/0.D0,1.2D0/,GPX,GOPX/0.D0,1.D-9/	40
	DATA X/0.3D0,2.D0/	45.1D
	KDU=2	50.4
	IFLAG=1	52.4
	HPR=0.1D0	54
	TOUT=T	56.4
	PRINT 5	60.1
5	FORMAT(T4,'ВРЕМЯ, СЕК',T16,'КООРДИНАТА X, М',	61.1
	* T32,'СКОРОСТЬ XP, М/С')	
10	CALL DRKF45(SDU1P,KDU,X,T,TOUT,GOPX,GPX,IFLAG,	65
	* WORK,IWORK)	
	CALL OUTF(T,X)	67.4
	TOUT=T+HPR	74.4
	IF(T.LT.TK) GOTO 10	75.4
	STOP	98.1
	END	99.1
	SUBROUTINE SDU1P(T,X,XP)	100.1
	IMPLICIT REAL *8(A-H,O-Z)	109.1D
	DIMENSION X(2),XP(2)	110.1

	DATA PI/3.141592653589D0/,ZM/0.01D0/,R/0.2D0/,	120.1D
	* C/1.D0/,ZL0/0.2D0/	
	AL=PI/6.D0	130.1D
	C1=- (C/ZM-PI*PI* (DSIN(AL)) **2)	134.1D
	C5=PI*PI*R*DSIN(AL) -9.81D0*DCOS(AL) +C*ZL0/ZM	136.1D
	XP(1)=X(2)	140.1
	XP(2)=C1*X(1)+C5	150.1
	RETURN	196.1
	END	198.1
	SUBROUTINE OUTF(T,X)	200.4
	IMPLICIT REAL *8(A-H,O-Z)	209.1D
	DIMENSION X(2)	210.4
	PRINT 25,T,X(1),X(2)	240.1
25	FORMAT(2X,3(2X,G12.5))	241.1
	RETURN	260.1
	END	262.1
C	ПРОГРАММА 22.2 В УДВОЕННОЙ ТОЧНОСТИ С DRKF45	
	IMPLICIT REAL *8(A-H,O-Z)	9.1D
	EXTERNAL SDU1P	10.5
	DIMENSION X(4),IWORK(5),WORK(27)	20.5
	DATA T,TK/0.D0,1.2D0/,GPX,GOPX/0.D0,1.D-9/	40
	DATA X/0.D0,10.D0,20.D0,0.D0/	45
	KDU=4	50.5
	IFLAG=1	52.4
	HPR=0.1D0	54
	TOUT=T	56.4
	PRINT 5	60.2
5	FORMAT(T4,'ВРЕМЯ, СЕК',T21,'X, M',T35,'Z, M',	61.2
	* T49,'XP, M/C',T62,'ZP, M/C')	
10	CALL DRKF45(SDU1P,KDU,X,T,TOUT,GOPX,GPX,IFLAG,	65
	* WORK,IWORK)	
	CALL OUTF(T,X)	67.4
	TOUT=T+HPR	74.4
	IF(T.LT.TK) GOTO 10	75.4
	STOP	98.1
	END	99.1
	SUBROUTINE SDU1P(T,X,XP)	100.1
	IMPLICIT REAL *8(A-H,O-Z)	109.1D
	DIMENSION X(4),XP(4)	110.2
	XP(1)=X(3)	140.5
	XP(2)=X(4)	142.5
	XP(3)=-4.D0*X(1)-2.D0*X(3)	150
	XP(4)=-4.D0*X(2)-2.D0*X(4)-9.81D0	152
	RETURN	196.1
	END	198.1

	SUBROUTINE OUTF(T,X)	200.4
	IMPLICIT REAL *8(A-H,O-Z)	209.1D
	DIMENSION X(4)	210.5
	PRINT 25,T,(X(I),I=1,4)	240.5
25	FORMAT(2X,5(2X,G12.5))	241.5
	RETURN	260.1
	END	262.1
C	ПРОГРАММА 22.3 В УДВОЕННОЙ ТОЧНОСТИ С DRKF45	
	IMPLICIT REAL *8(A-H,O-Z)	9.1D
	EXTERNAL SDU1P	10.4
	DIMENSION X(6),IWORK(5),WORK(39)	20.6
	DATA T,TK/0.D0,1.2D0/,GPX,GOPX/0.D0,1.D-9/	40
	DATA X/0.D0,0.D0,0.D0,1.D0,1.D0,2.D0/	45
	KDU=6	50.6
	IFLAG=1	52.4
	HPR=0.1D0	54
	TOUT=T	56.4
	PRINT 5	60.3
5	FORMAT(T4,'ВРЕМЯ, СЕК',T21,'X, M',T35,'Y, M',	61.3
*	T49,'Z, M',T62,'XP, M/C',T76,'YP, M/C',T90,'ZP, M/C')	
10	CALL DRKF45(SDU1P,KDU,X,T,TOUT,GOPX,GPX,IFLAG,	65
*	WORK,IWORK)	
	CALL OUTF(T,X)	67.4
	TOUT=T+HPR	74.4
	IF(T.LT.TK) GOTO 10	75.4
	STOP	98.1
	END	99.1
	SUBROUTINE SDU1P(T,X,XP)	100.1
	IMPLICIT REAL *8(A-H,O-Z)	109.1D
	DIMENSION X(6),XP(6)	110.3
	XP(1)=X(4)	140.3
	XP(2)=X(5)	142.3
	XP(3)=X(6)	144.3
	XP(4)=-0.5D0*X(4)	150
	XP(5)=-0.5D0*X(5)	152
	XP(6)=-0.5D0*X(6)+9.81D0	154
	RETURN	196.1
	END	198.1
	SUBROUTINE OUTF(T,X)	200.1
	IMPLICIT REAL *8(A-H,O-Z)	209.1D
	DIMENSION X(6)	210.6
	PRINT 25,T,(X(I),I=1,6)	240.3
25	FORMAT(2X,7(2X,G12.5))	241.3
	RETURN	260.1
	END	262.1

СОДЕРЖАНИЕ

Предисловие	4
Методические рекомендации	6
Основные обозначения	10
Часть 1. Краткие сведения для практической работы на персональных ЭВМ (ПЭВМ)	11
1. Общее описание ПЭВМ	11
1.1. Сравнительное описание клавиатур зарубежных и отечественных ПЭВМ	12
1.2. Действия при “зависании” компьютера	15
1.3. Диски	15
2. Основы операционной системы для ПЭВМ	18
2.1. Основные термины и понятия	18
Д о п о л н е н и е 2.1.	22
Д о п о л н е н и е 2.2.	23
Д о п о л н е н и е 2.3.	23
2.2. Основные сведения о командах DOS	24
2.2.1. Соглашения об обозначениях форматов команд	24
2.2.2. Структура команд DOS	25
2.2.3. Перенаправление потоков ввода и вывода	26
2.3. Основные команды DOS	26
3. Практическая работа на ПЭВМ с использованием сервисных программ (Norton Commander, PCTOOLS) и Microsoft Windows	31
3.1. Программа Norton Commander (NC)	31
3.2. Программа PC Tools	35
3.3. Microsoft Windows	37
3.3.1. Основные элементы Windows	38
3.3.2. Работа с диалоговыми окнами	40
3.3.3. Работа с программами и файлами	41
3.3.4. Использование меню Help для получения справочной информации	43
3.3.5. Program Manager	44
3.3.6. File Manager	46
Часть 2. Решение задач и выполнение индивидуальных заданий по статике с использованием готовых программ	48
4. Основные этапы решения задач статики на ПЭВМ	48
4.1. Формализация уравнений равновесия и выбор используемой программы	49
4.2. Подготовка данных для решения задач статики на ПЭВМ	49
4.2.1. Работа с универсальной программой STAT	49
4.2.2. Работа с универсальной программой STATN	50
4.3. Решение задачи на ПЭВМ и доведение первоначального решения до правильного. Анализ результатов, их проверка и выводы	52
4.3.1. Первоначальное решение задания на ПЭВМ	52

4.3.2. Поиск ошибок. Доведение первоначального решения до правильного.	55
5. Численное решение задач статики на ПЭВМ с “малым” и “большим” количеством уравнений равновесия	56
5.1. Подготовка данных для решения на ПЭВМ задач по определению усилий в стержнях пространственной конструкции для СССР.	56
5.1.1. Формализация уравнений равновесия.	56
5.1.2. Использование универсальной программы STAT.	58
5.1.3. Использование универсальной программы STATN.	59
5.2. Численное решение задач статики на ПЭВМ с “большим” количеством уравнений равновесия.	61
6. Численное решение задач статики на ПЭВМ со “средним” количеством уравнений равновесия.	64
6.1. Подготовка исходных данных для работы программ STAT и STAT9.	66
6.1.1. Использование программы STAT.	66
6.1.2. Использование программы STAT9.	67
6.2. Подготовка исходных данных для работы программ STATN и STAT9N.	68
6.2.1. Определение положения ненулевых элементов	68
6.2.2. Использование программы STATN	70
6.2.3. Использование программы STAT9N.	70
6.3. Описание результатов работы программ STAT9N, STAT9, STATN, STAT.	71
6.4. Проверка результатов решения и поиск ошибок.	71
Часть 3. Составление программ на Фортране для решения задач статики	73
7. Основные сведения по алгоритмическому языку Фортран.	73
7.1. Описание типов чисел и переменных.	74
7.2. Переменные с индексами. Описание массивов.	75
7.3. Арифметические выражения и системные функции	77
7.4. Арифметический оператор присваивания	79
7.5. Операторы управления	80
7.5.1. Оператор GO TO.	80
7.5.2. Условный арифметический оператор IF	81
7.5.3. Условный логический оператор IF	82
7.5.4. Оператор цикла DO	83
7.5.5. Правила использования оператора DO.	84
7.6. Операторы ввода-вывода	87
7.6.1. Общий вид операторов ввода-вывод.	87
7.6.2. Ввод - вывод, управляемый списком	88
7.6.3. Ввод-вывод данных на ПЭВМ	89
7.6.4. Требования к данным ввода, управляемого списком	90
7.6.5. Требования к данным вывода, управляемого списком	91
7.6.6. Оператор FORMAT	93
7.6.7. Спецификации формата для числовых данных: I, F, E, D, G	93

7.6.8. Спецификации формата для буквенно-цифровых (литеральных) данных (H, ' ', X, T)	95
7.6.9. Повторители и группы спецификаций	97
7.7. Задание начальных значений	98
7.7.1. Оператор DATA	98
7.7.2. Операторы явного описания типа	99
7.8. Ввод-вывод массивов. Форма неявного цикла DO	100
7.8.1. Ввод-вывод одномерных массивов	100
7.8.2. Ввод-вывод двумерных массивов	103
7.9. Понятие о подпрограммах и функциях	107
7.10. Порядок следования операторов в программной единице.	108
7.11. Понятие о формальных и фактических параметрах.	
Оператор CALL	109
8. Использование готовых подпрограмм	111
8.1. Краткий обзор методов решения систем линейных алгебраических уравнений	111
8.2. Описание подпрограмм DECOMP и SOLVE	113
8.3. Составление программ для решения задач статики с использованием подпрограмм DECOMP И SOLVE	116
8.3.1. Простейшая программа для решения задач статики.	116
Программа 8.1.	117
8.3.2. Организация ввода матриц по строкам	119
Программа 8.2.	120
Дополнение 8.1	124
Дополнение 8.2	125
Дополнение 8.3	125
Дополнение 8.4	126
Дополнение 8.5	126
Дополнение 8.6	127
8.3.3. Организация ввода только ненулевых элементов	127
Программа 8.3.	128
8.3.4. Использование особенностей задания C-9 при вводе всех элементов массивов	130
Программа 8.4.	131
Дополнение 8.7	135
Дополнение 8.8	135
Дополнение 8.9	136
Дополнение 8.10	136
8.3.5. Организация ввода только ненулевых элементов с учетом особенностей задания C-9	136
Программа 8.5.	137
Дополнение 8.11	139
8.4. Компиляция (трансляция) и редактирование фортран-программ на ПЭВМ	140

8.4.1. Команда FL	141
8.4.2. Редактор связей LINK	143
9. Использование стандартных подпрограмм	146
9.1. Пакет научных подпрограмм ПНП-SSP.	146
9.2. Использование подпрограмм SIMQ И ARRAY	146
9.2.1. Понятие о двумерной и векторной формах памяти	146
9.2.2. Описание подпрограмм SIMQ И ARRAY	149
9.2.3. Правила использования стандартных подпрограмм	150
9.3. Составление программ для решения задач статики с использованием стандартных подпрограмм SIMQ и ARRAY.	152
9.3.1. Простейшая программа для решения задач статики на ЭВМ с использованием подпрограммы SIMQ	152
Программа 9.1	152
9.3.2. Организация ввода матриц по строкам для работы подпрограммы SIMQ с использованием подпрограммы ARRAY	153
Программа 9.2	153
Программа 9.3	155
Программа 9.4	155
Программа 9.5	156
Часть 4. Выполнение индивидуальных заданий по кинематике с использованием готовых программ	158
10. Работа с использованием готовых программ KINMP и DKINMP ...	158
10.1. Краткое описание основных особенностей программ KINMP и DKINMP.	158
10.2. Подготовка данных для работы программы KINMP.	159
10.2.1. Структура вводимых данных для работы программы KINMP.	159
10.2.2. Представление исходных данных для программы KINMP.	160
10.3. Представление уравнений движения в виде подпрограммы KINU.	162
10.3.1. Составление уравнений движения	162
10.3.2. Составление подпрограммы KINU.	163
Дополнение 10.1	164
Дополнение 10.2	165
10.4. Работа с удвоенной точностью с использованием программ DKINMP	166
10.5. Описание результатов работы программ KINMP и DKINMP ...	168
Дополнение 10.3	168
10.6. Методические указания по организации сеанса работы на ЭВМ при работе с готовыми программами, требующими подготовки подпрограмм.	169
11. Определение угловых скоростей и угловых ускорений звеньев механизма манипулятора по заданному движению рабочей точки с использованием программ K9MP и DK9MP	172

11.1. Описание этапов аналитического расчета и подготовки исходных данных	173
11.2. Общее описание алгоритма решения задачи	178
11.3. Подготовка подпрограмм K9AV и K9BW.	179
11.4. Подготовка данных для работы программы K9MP	181
11.5. Работа с удвоенной точностью с использованием программы DK9MP	182
11.6. Замечания по использованию программ K9MP и DK9MP на ПЭВМ	184
11.7. Описание результатов работы программ K9MP и DK9MP, их проверка и поиск ошибок.	185
Д о п о л н е н и е 11.1.	186
Часть 5. Составление программ на Фортране для решения задач и выполнения индивидуальных заданий по кинематике	188
12. Дополнительные сведения по алгоритмическому языку Фортран	188
12.1. Функции и подпрограммы.	189
12.1.1. Оператор-функция	189
12.1.2. Подпрограмма-функция	191
12.1.3. Подпрограмма SUBROUTINE.	193
12.2. Оператор EXTERNAL	197
12.3. Оператор COMMON	198
12.3.1. Помеченный блок COMMON	202
12.3.2. Подпрограмма начальных значений BLOCK DATA	204
13. Составление программ на алгоритмическом языке Фортран для решения задач по кинематике с использованием численного дифференцирования.	205
13.1. Использование операторов-функций.	205
Программа 13.1	206
Д о п о л н е н и е 13.1.	209
Д о п о л н е н и е 13.2.	210
Д о п о л н е н и е 13.3.	212
Д о п о л н е н и е 13.4.	213
13.2. Использование подпрограмм-функций	215
Д о п о л н е н и е 13.5.	215
Д о п о л н е н и е 13.6.	216
Д о п о л н е н и е 13.7.	216
Д о п о л н е н и е 13.8.	218
Программа 13.2	219
Д о п о л н е н и е 13.9.	223
Программа 13.3	223
13.3. Использование подпрограмм SUBROUTINE	226
Программа 13.4.	226
Д о п о л н е н и е 13.10.	228
Д о п о л н е н и е 13.11	229

Дополнение 13.12.....	229
Дополнение 13.13.....	230
Дополнение 13.14.....	231
Программа 13.5.	231
14. Использование стандартных подпрограмм для решения задач кинематики.....	234
14.1. Описание стандартных подпрограмм для численного дифференцирования.	234
14.1.1. Подпрограммы DCAR, DBAR и DDCAR, DDBAR	234
14.1.2. Подпрограммы DET3, DET5 и DDET3, DDET5	236
14.2. Использование стандартных подпрограмм для численного дифференцирования	237
Программа 14.1.	238
Дополнение 14.1.	242
Дополнение 14.2.	243
Дополнение 14.3.	245
Программа 14.2.	245
15. Вычисления с удвоенной точностью.....	248
15.1. Переменные удвоенной точности.....	249
15.2. Неявный оператор типа - IMPLICIT.....	250
15.3. Требования к программам для работы с использованием удвоенной точности.	251
15.4. Вычисления с удвоенной точностью.....	252
Часть 6. Решение задач и выполнение индивидуальных заданий по динамике с использованием готовых программ.....	259
16. Работа с использованием специализированных программ DINSP и DDINSP.	259
16.1. Краткое описание программ DINSP и DDINSP.	259
16.2. Представление интегрируемого дифференциального уравнения для ввода в ЭВМ при работе со специализированными программами ..	260
16.3. Подготовка данных для работы специализированных программ	263
16.3.1. Структура вводимых данных для работы программы DINSP.	263
16.3.2. Представление исходных данных для программы DINSP. .	264
16.3.3. Сравнение аналитического решения с соответствующим правильным решением.....	266
16.4. Работа на ПЭВМ и представление полученных результатов....	266
17. Работа с использованием универсальных программ IDUSP и DIDUSP.	267
17.1. Подготовка интегрируемого дифференциального уравнения для ввода в ЭВМ при работе с универсальными программами	267
17.2. Представление системы двух дифференциальных уравнений первого порядка (С2ДУП) в виде подпрограммы SDUIP.	268

17.3. Подготовка данных для работы универсальных программ	269
17.3.1. Структура вводимых данных для работы программы IDUSP.	269
17.3.2. Представление исходных данных для программы IDUSP.	270
17.3.3. Сравнение результатов, полученных из аналитического решения при времени T1, с соответствующим численным на ЭВМ.	271
17.3.4. Сравнение аналитических зависимостей, полученных для координаты XА и скорости XРА, с численным решением на ЭВМ.	271
17.4. Работа с удвоенной точностью с использованием программы DIDUSP	272
17.5. Работа на ПЭВМ, представление и анализ полученных результатов	274
Часть 7. Составление программ на Фортране для решения задач и выполнения индивидуальных заданий по динамике	276
18. Необходимые сведения для численного интегрирования обыкновенных дифференциальных уравнений	276
18.1. Постановка задачи	276
18.2. Методы численного интегрирования дифференциальных уравнений	278
18.2.1. Одноступенчатые методы	278
18.2.2. Многоступенчатые или многошаговые методы	279
18.3. Представление дифференциальных уравнений высших порядков в виде системы дифференциальных уравнений первого порядка	281
18.3.1. Замена дифференциального уравнения второго порядка системой двух дифференциальных уравнений первого порядка.	281
18.3.2. Замена системы двух дифференциальных уравнений второго порядка (С2ДУ2П) системой четырех дифференциальных уравнений первого порядка (С4ДУ1П)	282
18.3.3. Замена системы трех дифференциальных уравнений второго порядка (С3ДУ2П) системой шести дифференциальных уравнений первого порядка (С6ДУ1П)	283
18.4. Описание используемых подпрограмм	285
18.4.1. Подпрограмма RKGS	285
18.4.2. Описание других стандартных подпрограмм	287
19. Численное интегрирование одномерных задач динамики.	288
19.1. Использование подпрограммы RKGS	288
Программа 19.1	288
Дополнение 19.1.	292
Дополнение 19.2.	292
Дополнение 19.3.	292
Дополнение 19.4.	293

Дополнение 19.5.	293
Дополнение 19.6.	295
Дополнение 19.7.	296
Дополнение 19.8.	297
Дополнение 19.9.	298
19.2. Использование других стандартных подпрограмм, реализующих многошаговые методы	298
19.2.1. Использование подпрограммы HPCG	298
19.2.2. Использование подпрограммы OMAD	299
19.2.3. Использование подпрограммы MAD3.	299
Дополнение 19.10.	299
Дополнение 19.11.	300
20. Численное интегрирование двумерных и трехмерных задач динамики.	302
20.1. Двумерные (плоские) задачи динамики	302
Программа 20.1.	302
Дополнение 20.1.	304
Дополнение 20.2.	305
20.2. Трехмерные (пространственные) задачи динамики	306
Программа 20.2.	307
Дополнение 20.3.	309
Дополнение 20.4.	309
21. Вычисления с удвоенной точностью.	310
Дополнение 21.1.	313
22. Численное интегрирование задач динамики с использованием подпрограммы RK45. Материалы для УИРС и НИРС	314
22.1. Подпрограмма RK45.	314
22.2. Численное интегрирование одномерных задач динамики с использованием подпрограммы RK45	316
Программа 22.1.	316
Дополнение 22.1.	320
22.3. Численное интегрирование двумерных задач динамики с использованием подпрограммы RK45	324
Программа 22.2.	324
22.4. Численное интегрирование трехмерных задач динамики с использованием подпрограммы RK45	325
Программа 22.3.	325
22.5. Вычисления с удвоенной точностью с использованием подпрограммы DRKF45	326
Литература	329
Приложение	331
Содержание.	360

Учебное издание

Носов Валерий Михайлович

Программирование на персональных ЭВМ задач
теоретической механики

Редактор *В.М.Носов*

Технический редактор *Я.А.Мельдис, В.М.Носов*

Подписано в печать с диапозитивов 24.02.97. Формат 60x90 1/16.
Бум. офсетная. Печать офсетная. Усл. печ.л. 23. Уч. изд. л. 23,8.
Тираж 2000 экз. Зак. 2375.

Издательство "Технопринт". Лицензия ЛВ № 1075.
220027, г. Минск, пр-т Ф.Скорины, 65, корп. 14, комн. 111.

Отпечатано с готовых диапозитивов на Минском ордена Трудового
Красного Знамени полиграфкомбинате МППО им. Я. Коласа.
220007, г. Минск, ул. Красная, 23.