Министерство образования Республики Беларусь

Белорусский национальный технический университет

Кафедра «Технической эксплуатации автомобилей»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

**по выполнению курсовой работы
по курсу «Информатика» для студентов специальностей
1-37.01.06 «Техническая эксплуатация автомобилей»
1-37.01.07 «Автосервис»**

Минск 2006

УДК 002(075.8)
ББК 32.81я7
И 74

Составители:
А.С. Сай, В.С. Смольская, А.М. Расолько,
Л.Н. Поклад, А.Д. Пашин

Рецензенты:
Казацкий А.В., Гурский А.С.

В работе приведены общетеоретические и практические рекомендации в составлении алгоритмов вычислительных процессов, которые позволяют, постепенно расчленяя задачи на подзадачи, сводить их решение, в конечном счете, к некоторым типовым фрагментам алгоритмов, а также методические рекомендации по составлению программ, которые позволяют с использованием возможностей средств вычислительной техники, находить оптимальные решения. Приведена структура курсовой работы, правила ее оформления, даны методические рекомендации по выполнению отдельных ее разделов.

1. ОБЩИЕ МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ КУРСОВОЙ РАБОТЫ

1.1. Цель и задачи работы

Курсовая работа способствует развитию у студентов навыков в применении средств вычислительной техники в моделировании различных технологических, конструкторских и исследовательских задач в области технической эксплуатации автомобилей.

Целью работы является умение студентов находить способ решения, например, технологических задач технического обслуживания и диагностирования, расчета основных узлов технологического расчета автотранспортных предприятий и станций технического обслуживания автомобилей и других задач, требующих большого количества вычислений. Умение составить алгоритм моделирования процесса в таком виде, чтобы ЭВМ могла его выполнить, т.е. разбитие алгоритма на элементарные операции, которые можно записать на определенном алгоритмическом языке.

Это дает возможность закрепить общетеоретические и практические навыки в составлении алгоритмов вычислительных процессов, которые позволяют, постепенно расчленяя задачи на подзадачи, сводить их решение, в конечном счете, к некоторым типовым фрагментам алгоритмов, подобно расчленению совершенно непохожих сложных механизмов на одинаковые детали и узлы, только по разному соединенных. Привить студентам умение подходить к процессу составления программы как творческому процессу, позволяющему, с использованием возможностей средств вычислительной техники, находить оптимальные решения.

При выполнении курсовой работы студент должен уметь: сформулировать (поставить) задачу, дать ее математическое описание, собрать необходимые исходные и нормативно-справочные данные, используемые в расчетах, на основании этого уметь составить блок-схему алгоритма решения задачи и программу расчета, на основании которой происходит моделирование процесса с изменением входных параметров и получении графических зависимостей с целью выбора оптимального решения.

1.2. Структура и объем курсовой работы

В связи с тем, что целью работы является закрепление общетеоретических и практических навыков, моделирование технологических, конструкторских и исследовательских задач с использованием средств вычислительной техники, она должна включать в себя следующие разделы: постановка задачи, заключающаяся в формировании решаемых вопросов, которые изучаются предварительно по специальной литературе, например, по технической эксплуатации автомобилей, справочникам по проектированию технологического оборудова-

ния, методикам технологического расчета автотранспортных предприятий и станций технического обслуживания, а также по научно-технической литературе по изучению вопросов влияния различных факторов на моделируемый процесс, например, на износ, на надежность агрегатов и узлов и др.; математическое описание задачи, включающее в себя изложенные в определенной последовательности математические формулы вычисления отдельных величин; нормативно-справочные данные; блок-схему решения задачи; программу; результаты расчета и графические зависимости.

Работа состоит из расчетно-пояснительной записки и графического материала (схем, графиков).

Расчетно-пояснительную записку рекомендуется выполнять на сброшюрованных листах формата А4 297×210), ГОСТ 2.301-68. Титульный лист выполняется по форме, приведенной в приложениях 1,2. На титульном листе указывается название дисциплины, фамилия, имя, отчество студента, факультет, специальность, курс. Записка должна быть набрана на ЭВМ в текстовом редакторе Word на стандартной белой бумаге формата А4 по ГОСТ 2.301-68 с одной стороны листа.

Размер площади текста на странице не должен превышать 245 (высота) × 160 (ширина) мм. Поля: верхнее, нижнее, левое - 25 мм; правое – 10 мм. Интервал между строками и абзацами – полуторный. Размер шрифта должен соответствовать 14 pt Word для Windows /1/. Абзацы в тексте начинают отступом 15-17 мм, одинаковым по всему тексту.

Расчетно-пояснительная записка должна быть написана и оформлена в соответствии с требованиями МИ БНТУ 3.001-2003 /2/.

Текс основной части пояснительной записки должны быть разделены на разделы, подразделы и пункты. Нумерация страниц производится арабскими цифрами по всему тексту записки, включая иллюстрации, таблицы и приложения. Номер страницы проставляется в правом верхнем углу. Титульный лист и задание включаются в общую нумерацию страниц, но номер на них не проставляется.

Разделы, подразделы, пункты и подпункты должны иметь содержательные заголовки. Разделы нумеруются арабскими цифрами с точкой в пределах всей записки. Подразделы нумеруются арабскими цифрами с точкой в пределах раздела (например, 1.5, т.е. пятый подраздел первого раздела). Пункты нумеруются арабскими цифрами с точкой в пределах каждого подраздела (например, 1.5.1). Номера разделов, подразделов и пунктов записываются с абзацного отступа. Внутри пунктов могут быть перечисления, перед каждой позицией перечисления следует ставить тире. Заголовки разделов следует писать прописными буквами с абзацного отступа. Заголовки подразделов следует писать, начиная с прописной буквы строчными буквами, с абзацного отступа. Точка в конце заголовка раздела, подраздела не ставится, название не подчеркивается. Расстояние

между заголовком и текстом должно быть 3-4 интервала. Между заголовками раздела и подраздела – 2 интервала.

Таблицы нумеруются арабскими цифрами порядковой нумерацией в пределах раздела. Номер таблицы состоит из номера раздела и порядкового номера таблицы, разделенных точкой, например, «Таблица 3.1». Таблица каждого приложения обозначают отдельной нумерацией арабскими цифрами с добавлением перед цифрой обозначения приложения, например, «Таблица А.5». На все таблицы пояснительной записки должны быть сделаны ссылки в тексте. При ссылках следует писать: «… по таблице 1.2». Слово «Таблица» с номером указывается один раз слева над первой частью таблицы.

Таблицы должны иметь названия, раскрывающие их содержание. Название следует помещать над таблицей сразу после номера таблицы. Таблицу располагают сразу после ссылки на нее. Если в одной и той же графе таблицы приводятся целые числа и числа с десятичными долями, то следует целые числа без десятичных знаков после запятой дополнять соответствующим числом нулей. Ставить кавычки вместо повторяющихся цифр или символов не допускается. Если цифровые или другие данные в какой-либо строке таблицы не приводятся, то ставится прочерк. Допускается под таблицей делать примечания, поясняющие сущность, как отдельных ее элементов, так и таблицы в целом.

Если таблица не помещается по вертикали листа, то допускается ее располагать по горизонтали, при этом ее располагают так, чтобы при чтении записку нужно было повернуть на 90 градусов по часовой стрелке. При переносе таблицы на следующую страницу головку таблицы следует повторить, а над ней написать слева слова, например, «Продолжение таблицы 2.1.». Если головка очень громоздкая, то допускается при переносе ее не повторять, при этом нумеруются графы, и эту нумерацию повторяют на следующей странице. Заголовок таблицы не повторяется. Если таблица переносится со страницы на станицу несколько раз, то над головкой последней страницы пишутся слова «Окончание таблицы 2.1».

Иллюстрации (схемы, графики, чертежи, фотографии и др.) следует располагать после текста, в котором они упоминаются впервые. Иллюстрации следует нумеровать в пределах раздела арабскими цифрами. Номер рисунка состоит из номере раздела и порядкового номера рисунка, разделенных точкой, например, «Рисунок 2.1». рисунки каждого приложения обозначаются отдельной нумерацией арабскими цифрами с добавлением перед цифрой обозначения приложения, например, «Рисунок А.1». Слово «Рисунок», номер и наименование помещают после рисунка и пояснительных данных (если они имеются).

Формулы набирают с помощью встроенного редактора формул. В формулах в качестве символов следует применять обозначения, установленные соответствующими стандартами. Пояснения символов и числовых коэффициентов, входящих в формулу, если они не пояснены ранее в тексте, должны быть приведены непосредственно под формулой. Пояснения каждого символа следует давать с но-

вой строки в той последовательности, в которой символы приведены в формуле. Первая строка пояснения должна начинаться со слова «где» без двоеточия после него.

Формулы, следующие одна за другой и не разделенные текстом, разделяют запятой.

Переносить формулы на следующую строку допускается только на знаках выполняемых операций, причем знак в начале следующей строки повторяют. При переносе формулы на знаке умножения применяют знак «х».

Формулы должны нумероваться в пределах раздела арабскими цифрами, которые записывают на уровне формулы справа в круглых скобках. Номер формулы состоит номера раздела и порядкового номера формулы, разделенных точкой, например, (3.2).

Формулы в приложениях нумеруются в пределах каждого приложения с добавлением обозначения приложения - (В.1).

Ссылки в тексте на порядковые номера формул дают в скобках, например, «... в формуле (3.2)».

Материал, дополняющий текст пояснительной записки, допускается помещать в приложениях, которые оформляются как продолжение пояснительной записки. Каждое приложение следует начинать с нового листа с указанием наверху по середине страницы слова «Приложение» и его обозначения. Приложения обозначаются заглавными буквами русского (белорусского) алфавита, начиная с А, например, «Приложение А».

Объем записки должен составлять 20 – 35 рукописных листов.

Графическую часть рекомендуется выполнять с помощью встроенной программы построения графиков и диаграмм Microsoft Graph на листах А4, А3 ГОСТ 2.301-68, в зависимости от объема графической части.

При создании диаграммы с помощью Microsoft Graph диаграмма и связанные с ней данные отображаются в таблице, называемой «таблица MS Graph» Данная таблица содержит данные примера, которые показывают куда должны вноситься подписи строк и столбцов, а также данные пользователя.

Данные, используемые для импорта, могут содержать до 4000 строк и до 4000 столбцов, однако на диаграмме может быть представлено не более 255 рядов данных.

Все разделы расчетно-пояснительной записки должны быть взаимосвязаны, должно быть однозначное восприятие входных и выходных параметров, приведены ссылки на соответствующую научно-техническую литературу.

В конце работы должны быть указаны использованные литературные источники.

2. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ КУРСОВОЙ РАБОТЫ

2.1. Введение

Во введении необходимо отразить современное состояние средств вычислительной техники, применение ее в области эксплуатации автомобильного транспорта. Дать описание решаемой задачи, например, описание технологического процесса соответствующего производственного участка, параметры которого подлежат расчету, описание конструкции соответствующего технологического оборудования, описание неисправностей диагностируемого объекта и др.

2.2. Постановка задачи

Поставить задачу – значит сформулировать ее условие. При формулировке задачи обычно применяется математический аппарат, с помощью которого в дальнейшем задача будет решаться. Постановка задачи осуществляется специалистом в определенной области деятельности и требует высокой квалификации.

При выполнении работы необходимо иметь в виду, что задание на курсовую работу формулируется в общем виде и требует конкретизации, т.е. разбиения на элементарные подзадачи с описанием входящих параметров, которые будут запрашиваться при выполнении программы расчета и тех, которые принимаются из нормативно-справочной литературы. Кроме этого, если эти нормативно-справочные данные корректируются для заданных условий, то необходимо это оговорить в этом разделе.

Рассмотри формирование разделов курсовой работы на примере расчета производственной программы по техническому обслуживанию (ТО) автомобилей, заключающегося в решении следующих показателей:

- количество капитальных ремонтов $N_{кр}$ в год и годовая производственная программа по видам ТО (N_1, N_2, N_{eoc}, $N_{eот}$) определяются по следующим формулам:

$$N_{кр} = \frac{L_г}{L_{кр}}; \qquad (2.1)$$

$$N_2 = \frac{L_г}{L_2} - N_{кр}; \qquad (2.2)$$

$$N_1 = \frac{L_г}{L_1} - N_{кр} - N_2; \qquad (2.3)$$

$$N_{EO_C} = \frac{L_г}{L_{сс}}; \qquad (2.4)$$

$$N_{eот} = 1.6(N_1^г + N_2^г), \qquad (2.5)$$

где $L_г$ – годовой пробег парка автомобилей, км; $l_{сс}$ – среднесуточный пробег автомобилей, км; L_1, L_2 – скорректированная периодичность технического обслуживания автомобилей ТО-1 и ТО-2, км.

- годовой пробег автомобилей определяется по выражению

$$L_г = A_и \cdot l_{сс} \cdot a_T \cdot Д_{рг}, \qquad (2.6)$$

где $L_г$ – годовой пробег автомобилей, км; $A_и$ – списочное количество автомобилей, ед; a_T – коэффициент технической готовности парка; $Д_{рг}$ – количество дней работы подвижного состава на линии в течение года, дн.

- коэффициент технической готовности парка определяется по формуле

$$a_T = \frac{1}{1 + l_{сс} \cdot \frac{Д_{ТО,ТР}}{1000}}, \qquad (2.7)$$

где $Д_{ТО,ТР}$ – скорректированная продолжительность простоя автомобиля в ТО и ТР в днях на 1000 км пробега.

- выбранные нормативные пробеги автомобилей до ТО-1,ТО-2 и ремонта приводятся к конкретным условиям эксплуатации подвижного состава с помощью коэффициентов, учитывающих категорию условий эксплуатации ($К_1$), модификацию подвижного состава и организацию его работы ($К_2$), климатические условия ($К_3$).

$$L_1 = L_1^{\,H} \cdot K_1 \cdot K_3 \qquad (2.8)$$

$$L_2 = L_2^{\,H} \cdot K_1 \cdot K_3 \qquad (2.9)$$

$$L_{КР} = L_{КР}^{\,H} \cdot K_1 \cdot K_2 \cdot K_3 \qquad (2.10)$$

где $L_{кр}^{\,н}$, $L_2^{\,н}$, $L_1^{\,H}$ - соответственно нормативные пробеги до КР, ТО-2, ТО-1 для автомобиля данного класса, которые приведены в таблице 2.1.

Таблица 2.1 – Значения нормативов и корректирующих коэффициентов

Тип подвижного состава	Исходное значение норматива				Коэффициент корректирования		
	$L_1^н$, км	$L_2^н$, км	$L_{КР}^н$, км	Д$_{ТО,ТР}$, дн/1000км	K_1	K_2	K_3
Грузовые автомобили	4000	16000	400000	0,35	0,9	1,0	1,0
Автобусы	5000	20000	500000	0,25	0,9	1,0	1,0

- скорректированное значение величины определяется по формуле

$$Д_{ТО,ТР} = Д_{ТО,ТР} \times K_2, \qquad (2.11)$$

где $К_2$ – коэффициент корректирования в зависимости от модификации подвижного состава и организации его работы, выбирается из таблицы 2.1.

Решение этой задачи можно разбить на элементарные подзадачи, которые и указываются в разделе «постановка задачи»:

– определение годового пробега парка автомобилей, входными параметрами которого являются количество автомобилей в парке, среднесуточный пробег автомобилей, количество дней работы подвижного состава в году и коэффициент технической готовности.

Параметры – количество автомобилей в парке, среднесуточный пробег автомобилей задаются пользователем; количество дней работы в году выбирается (252, 302 или 365 дней) в соответствии с заданным типом подвижного состава. Что касается коэффициента технической готовности, то он определяется отдельно с входными параметрами: среднесуточный пробег автомобилей и простой автомобилей в техническом обслуживании и ремонте, который необходимо корректировать в зависимости от заданных условий эксплуатации;
выбор нормативной периодичности технического обслуживания;

– выбор коэффициентов корректирования периодичности ТО (выбор нормативной периодичности ТО и коэффициентов ее корректирования осуществляется из нормативной таблицы 2.1);

– определение периодичности технического обслуживания для заданных условий;

– определение производственной программы по техническому обслуживанию автомобилей.

2.3. Математическое описание задачи

В этом разделе приводятся необходимые математические выражения для решения всех элементарных подзадач, которые описаны выше. Выражения записываются в общем виде, с расшифровкой входящих в него параметров с указанием размерности величин. Если параметры в задании заданы изменяющимися в пределах, то указываются min и max их значения, шаг изменения величины.

Выражения должны следовать друг за другом в такой последовательности, чтобы не оставалось неизвестных величин.

Например. Годовой пробег автомобилей определяется по формуле

$$L_г = A_и \cdot l_{сс} \cdot a_T \cdot Д_{рг}, \qquad (2.12)$$

где $L_г$ – годовой пробег автомобилей, км; $A_и$ – списочное количество автомобилей, ед; $l_{сс}$ – среднесуточный пробег автомобилей, км; a_T – коэффициент технической готовности парка; $Д_{рг}$ – количество дней работы подвижного состава на линии в течение года, дн.

Примечание: в скобках указывается порядковый номер формулы.

Коэффициент технической готовности парка определяется по формуле

$$a_T = \frac{1}{1 + l_{сс} \cdot \frac{Д_{ТО,ТР}}{1000}}, \qquad (2.13)$$

где $Д_{ТО,ТР}$ – скорректированная продолжительность простоя автомобиля в ТО и ТР в днях на 1000 км пробега.

В этом же разделе указывается, что нормативное значение продолжительности простоя автомобиля в ТО и ТР, периодичности технического обслуживания автомобилей, пробега до КР и коэффициенты их корректирования выбирается из таблицы 2.1, которая приводится в разделе «Нормативно-справочные данные» в зависимости от типа подвижного состава, который указывается в задании курсовой работы. Здесь же необходимо описать алгоритм выбора данных из таблицы 2.1, т.е. определяющим при выборе, например, периодичности ТО будет вид подвижного состава для грузовых автомобилей это 4000 и 16000 км (данные из таблицы 2.1), а для автобусов они будут равны соответственно 5000 и 20000 км. Это необходимо для правильного построения в дальнейшем алгоритма и программы расчета.

Скорректированное значение величины простоя в ТО и ремонте определяется по формуле

$$Д_{ТО,ТР} = Д^{н}_{ТО,ТР} \cdot к_2, \qquad (2.14)$$

где $к_2$ – коэффициент корректирования в зависимости от модификации подвижного состава и организации его работы.

В свою очередь коэффициент $К_2$ выбирается из таблицы /1/, которая также приводится в разделе «Нормативно-справочные данные» (в тексте дается просто ссылка на порядковый номер таблицы) и приводится описание алгоритма выбора коэффициента $К_2$.

Если в задании указывается изменение величин от min значения до max, например, количество автомобилей в парке изменяется от 50 единиц до 450 ед. с шагом 50 ед., а среднесуточный пробег от 100 км до 250 км с шагом 10 км, то формулы приобретают другой вид, т.е. в них входят уже не простые переменные A_u и l_{cc}, а элемент одномерных массивов (A_{ui} и l_{ccj}, где i и j – порядковый номер одномерного массива).

А результат, годовой пробег автомобилей, становится двумерным массивом, т.е. $L_{гij}$.

$$L_{гij} = A_{ui} \cdot l_{ccj} \cdot a_{Тj} \cdot Д_{рг}, \qquad (2.15)$$

где $a_{Тj}$ – коэффициент технической готовности пара при j-м значении среднесуточного пробега.

$$a_{Тj} = \frac{1}{1 + l_{ccj} \cdot \frac{Д_{ТО,ТР}}{1000}}, \qquad (2.16)$$

По вычисленным значениям $L_{гij}$ и $a_{Тj}$ будут строиться графические зависимости, т.е. изменение годового пробега автомобилей и коэффициента технической готовности парка в зависимости от количества автомобилей в парке и их среднесуточного пробега.

Аналогичным образом дается описание и более сложных технических задач. При необходимости решение задачи поясняется расчетной схемой (рисунок 2.1, 2.2), на которой наносятся отдельные параметры, входящие в математические формулы. Например, характеристика тарельчатых пружин нелинейная. Она строится по формуле:

$$P = \frac{4\,ES\,l}{(1-m^2)\,D_1^2\,A}\left[(f_3 - l)\left(f_3 - \frac{l}{2}\right) + S^2\right], \quad (2.17)$$

где P – усилие, воспринимаемое пружиной; l - осадка тарелки;

D_1 – внешний диаметр тарелки; f_3 – стрела прогиба тарелки ;

S- толщина металла пластины тарелки; E- модуль упругости;

μ – коэффициент Пуассона.

Эта математическое описание поясняется расчетной схемой, приведенной на рисунке 2.1.

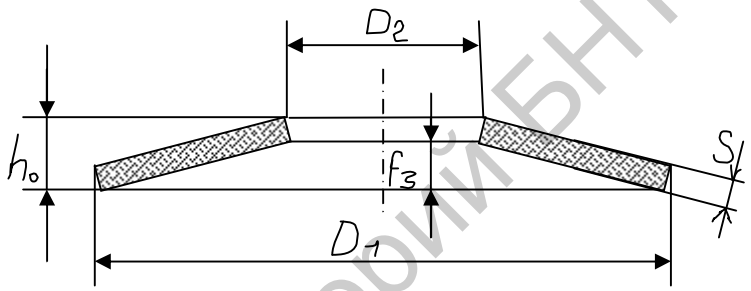


Рисунок 2.1 – Расчетная схема тарельчатой пружины.

Или при расчете поршневого пальца на изгиб рекомендуется применять силовую схему нагрузки, показанную на рисунке 2.2.

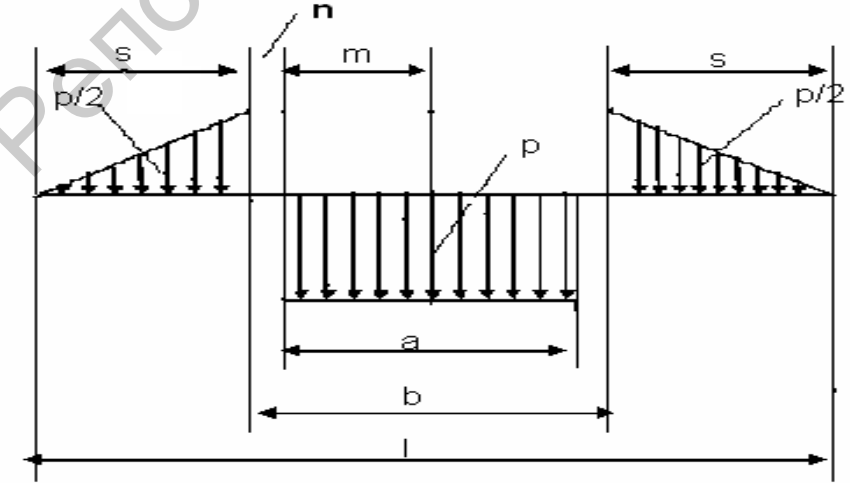


Рисунок 2.2 – Расчетная схема определения напряжения изгиба в поршневом пальце ДВС.

Эта схема поясняет следующее математическое описание расчета напряжения изгиба в поршневом пальце ДВС

$$s_{и} = \frac{P(1+2b-1{,}5a)}{1{,}2d^{3}(l-a^{4})}, \qquad 2.18)$$

где P – нагрузка, действующая на палец, Н; l – рабочая длина пальца, м; b – расстояние между бобышками поршня, м; a – длина поршневой головки шатуна, м; a - отношение внутреннего диаметра пальца к его наружному диаметру d.

2.4. Нормативно-справочные данные

В этом разделе работы приводятся все нормативные данные в виде таблиц или констант, т.е. те данные, которые не вычисляются, а выбираются в зависимости от данных условий. В задании обычно указываются фиксированные значения, например, определенная марка грузового, легкового автомобиля. Это делается для того, чтобы осуществить проверочный расчет и работоспособность программы. В целом же программа должна обладать определенной универсальностью (по согласованию с руководителем работы), например, расчета годового пробега парка любой марки грузового автомобиля, смешанного парка, состоящего, например, из определенного количества грузовых автомобилей различных моделей и определенного количества различного класса автобусов и др.

Поэтому и выбор нормативно-справочных данных должен осуществляться для групп заданных величин. Так, например, для задачи описанной в подразделе 2.2. нормативно-справочные данные выбора продолжительности простоя в ТО и ремонте могут быть представлены в виде таблицы 2.2.

Таблица 2.2 Нормативы продолжительности простоя подвижного состава в техническом обслуживании и ремонте

Тип подвижного состава	Продолжительность простоя $Д_{ТО,ТР}$, дн/1000 км
Грузовые автомобили общего назначения:	
– особо малой грузоподъемности	0,25
– малой грузоподъемности	0,30
– средней грузоподъемности	0,35
– большой грузоподъемности	
свыше 5 т и до 6 т	0,38
свыше 6 т и до 8 т	0,43
– особо большой грузоподъемности	
свыше 8 т и до 10 т	0,48
свыше 10 т и до 16 т	0,53
Автобусы:	
– особо малого класса	0,20
– малого класса	0,25
– среднего класса	0,30
– большого класса	0,35
– особо большого класса	0,45

Как видно из таблицы 2.2 ее можно использовать для выбора нормативной продолжительности простоя автомобилей в ТО и ТР любого грузового пар-

ка (одномарочного, многомарочного), а также смешанного парка грузовых автомобилей и автобусов различного количественного состава.

На этом этапе работы уже необходимо предусмотреть определенный алгоритм выбора величин из нормативно-справочных таблиц. Здесь уже необходимо предвидеть, какие операторы в программе будут осуществлять этот алгоритм, для чего необходимы знания правил записи операторов и порядок их действий, их возможностей в осуществлении вычислительного процесса.

2.5. Разработка алгоритма решения

Решение задачи с использованием компьютера разделяется на несколько этапов. Прежде всего, задача должна быть подробно описана математически и выбран численный метод ее решения. Далее решение описывается в виде алгоритма.

Алгоритм – всякое точное предписание, которое задает вычислительный процесс, начинающийся с произвольного исходного данного и направленный на получение полностью определяемого этими исходными данными результата. Применительно к решению задач на компьютере алгоритм представляет собой последовательность арифметических и логических действий над числовыми значениями переменных, приводящую к решению задачи при изменении исходных данных в широких пределах.

Из существующих способов описания алгоритмов наиболее часто используется описание алгоритмов в виде блок-схемы. Она имеет вид рисунка, состоящего из различных геометрических фигур и стрелок, идущих от одной фигуры к другой. Внутри каждой геометрической фигуры содержится действие. В этом случае каждому действию (ввод исходных данных, вычисление, проверка условий, вывод результатов и т.д.) соответствует определенная геометрическая фигура (блок), а алгоритм определяется геометрически в виде последовательности блоков, выполняющих определенные функции.

Алгоритм должен обладать тремя свойствами:

– определенностью выполнения вычислений, заключающейся в точном и однозначно понимаемом порядке;

– массовостью или универсальностью, т.е. возможностью использовать данный численный метод не только для решения данной конкретной задачи, но и для других однотипных задач;

– результативностью или сходимостью, т.е. неизбежностью получения результата.

Блок-схема алгоритма дает возможность представить логику решения задачи, связь ее отдельных частей в наглядной графической форме. Появляется возможность программировать отдельные блоки различными программистами, почти независимо друг от друга, что уменьшает время разработки всей программы в целом.

В блок-схеме различают три типа блоков:

– линейные – для вычисления по формулам;

– блоки условий (логические) – для проверки выполнения заданных условий;

– вспомогательные.

В таблице 2.3 приведены основные символы блок-схем алгоритмов. При составлении блок-схемы каждый блок нумеруется. Номер записывается обычно слева. Блоки должны иметь сквозную нумерацию.

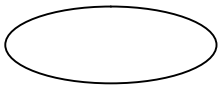
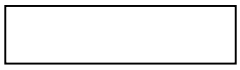
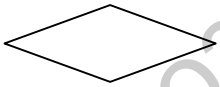
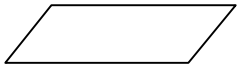
Правила составления блок-схем:

– каждый блок имеет только один вход;

– выполнение действий, описанных в блоке, всегда начинается с первого, входить в середину блока, минуя часть вычислений, не допускается;

– первым выполняется блок, к которому ведет стрелка от блока «начало»;

Таблица 2.3 Символы блок-схем алгоритмов

Обозначение	Назначение
	Начало, конец или прерывание процесса обработки данных или выполнения программы.
	Функция, в результате которой изменяется значение, форма и расположение данных.
	Выбор направления выполнения программы в зависимости от некоторых переменных условий.
	Начало цикла.
	Ввод, вывод данных

– преемник блока указывается исходящей стрелкой, причем логический блок определяет обычно два и более преемников, управление после проверки условия получает один из них;

– передачи управления при отражении цикличности алгоритма рекомендуется показывать стрелкой, входящей слева в первый из повторяющихся в цикле блоков;

– внутри блока обычно указывается формула, проверяемое условие, краткое словесное описание действия.

Например. Блок-схема для определения нормативного простоя автобусов в ТО и ТР (описанная в подразделе 2.3.) приведена на рисунке 2.3.

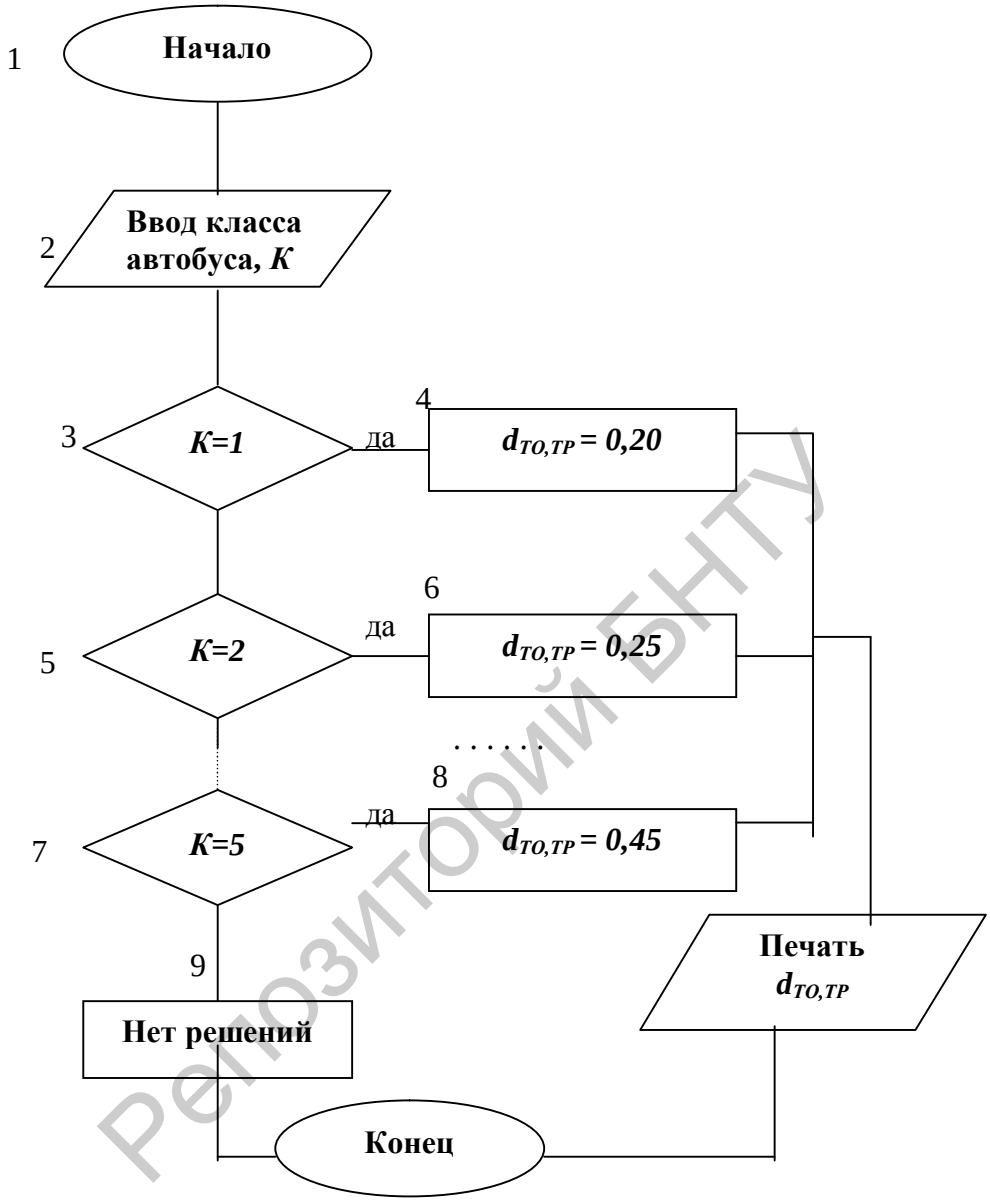


Рисунок 2.3 Блок-схема выбора нормативного простоя автобусов в ТО и ремонте в зависимости от их класса.

Примечание: Цифрами 1,2,…,5 обозначен (закодирован) класс автобуса (особо малый, малый и т.д.).

2.6. Программирование алгоритма расчета

Программирование заключается в переводе алгоритма (блок-схемы алгоритма) расчета на один из алгоритмических языков, изучаемых в курсе «Информатика». Описание одного шага вычислительного процесса, содержащее название одной операции и адреса операндов, называется командой или операто-

ром. Совокупность команд (операндов), описывающих весь вычислительный процесс решения данной задачи, называется программой решения задачи.

Процесс создания программ является довольно сложным, многоступенчатым и во многом творческим процессом. Он включает в себя этапы проектирования, написания, комплектации, компоновки, отладки и запуска программы в работу. Большую помощь в формализации создания программы оказывает метод структурного программирования, применительно к которому и был создан язык Паскаль.

Любая программа на алгоритмическом языке Паскаль состоит из заголовка и блока и имеет следующую структуру:

Program – имя;

Uses – список используемых модулей;

{**Блок**};

Label – раздел меток;

Любой выполняемый оператор может иметь метку – (обычно при разветвлении вычислительного процесса) целую положительную метку (не более 4 цифр). Все используемые в программе метки должны быть описаны в этом разделе. Оператор обычно получает метку в том случае, если требуется пропустить часть вычислительного процесса, в связи с выполнением каких-то условий.

Const – раздел констант;

Обычно в этом разделе приводятся константы, используемые для разных вариантов программы.

type – раздел типов;

В этом разделе описываются имена типов переменных.

Var – раздел переменных;

В этом разделе описываются все переменные, используемые в программе, как вычисляемые, так и входящие в различные выражения.

Procedure, function – раздел процедур и функций;

В этом разделе находят отражение однотипные участки программы, которые выполняют одни и те же вычисления, но с различными данными.

Begin – раздел операторов;

Раздел операторов – выполняемая часть программы, которая записывается в свободной форме и является основой вычислительного процесса.

Наличие интегрированной среды Турбо-Паскаль позволяет объединить написание, компиляцию, компоновку, отладку программы и ее запуск в одном процессе. Это ускоряет создание работоспособной программы, позволяет проще найти ошибки в программе на всех стадиях ее создания.

Начинается работа в среде Турбо-Паскаль с написания текста программы в окне редактирования с помощью редактора среды.

Любая программа начинается со слова Program, после которого указывается имя программы. Имя программы образуется так же, как и имена переменных и по смыслу должно быть близким к теме решаемой задачи, что в дальнейшем облегчает их поиск и классификацию.

В отличие от других алгоритмических языков, где действует принцип умолчания, в разделе **Var** должно быть дано описание всех переменных, используемых в программе. Прежде всего, каждой переменной присваивается имя (индентификатор), которое может иметь длину до 126 символов и различаться первыми 63 символами. Имя переменной состоит из буквенно-цифрового обозначения, близкого к обозначению ее в математических формулах. Пробелы в индентификаторах не допускаются. Запрещается также использовать в качестве индентификаторов ключевые слова Турбо-паскаля (**IF**, **DO**, **then**, **to** и другие). Строчные и прописные буквы в индентификаторах тождественны, т.е. не различаются. Описание типа переменной производится следующим образом

Var список переменных: тип;

Наиболее часто используются переменные целого (integer), вещественного (real), логического (boolean) и символьного (char) типов. Например,

Var x,z,f: real; i,j,k: integer; s1,d: char;

Таким образом, производится описание простых переменных. При описании массивов, представляющих упорядоченную совокупность конечного числа данных одного типа. Индентификаторы массивов образуются так же, как и имена простых переменных. Описание типа данных массива производится следующим образом

Var имя массива:array[m..n] of тип элементов массива (integer, real, boolean, char);

Различают понятие индекса массива, который принимает в вышеприведенном примере значения от m до n.

Например, $X_1, X_2, \dots, X_{10}$, где n = 1, m = 10. В качестве индексов массива могут быть использованы константы и переменные любого скалярного типа, кроме real.

Например, имеется массив X_i, где i = 1,2 … , 10. При этом данные, находящиеся в массиве принимают значения $X_1 = 0{,}25$, $X_2 = 0{,}33$ и т.д. Описание такого массива данных производится в следующем порядке

Var X:array[1..10] of real;

i: integer;

Описание двумерного массива данных осуществляется следующим образом

Var Z:[n..m,k..l] of тип данных,

где n,m – начальное и конечное количество строк в таблице, k,l – начальное и конечное количество столбцов. Общее количество данных определяется произведением m × l.

Например, необходимо описать данные, приведенные в таблице 2.4.

Таблица 2.4 Распределение поступления автомобилей в грузовые автотранспортные предприятия (ГАП).

Автотранспортные предприятия	Год поступления			
	2003	2004	2005	2006
ГАП-1	2	3	5	4
ГАП-2	1	2	4	3
ГАП-3	3	2	5	8

Описание этого массива данных, который обозначим A_{ij}, имеет вид

Var A:array[1..3,1..4] of integer; i,j: integer;

В разделе операторов, начинающийся с ключевого слова Begin и заканчивающийся словом End, описываются все алгоритмические действия, которые необходимо выполнить над исходными данными для получения результата. В начале необходимо ввести исходные данные, при которых будут производиться вычисления. Для ввода исходных данных используется оператор Read. Его общий вид

read (читать данные), readln (читать данные с переводом строки).

Ввод исходных данных может производиться в диалоговом режиме и из внешнего файла данных. Выбор того или иного способа ввода зависит от типа данных. Если данные постоянно меняются при каждом решении и их количество небольшое, то следует применять диалоговый режим, который состоит в следующем, в начале дается сообщение о порядке и количестве вводимых данных с использованием оператора вывода

write (выводить данные), writeln (выводить данные с переводом строки).

Например, при вычислении годового пробега парка автомобилей по формуле 2.12 необходимо вводить постоянно разные значения количества автомобилей и среднесуточного пробега при фиксированном режиме работы (302 дня в году и коэффициенте технической готовности). В этом случае ввод количества автомобилей и среднесуточного пробега осуществим в диалоговом режиме.

Writeln(‘Введите количество автомобилей в парке, $A_и$’);

Readln(Ai);

Writeln(‘Введите среднесуточный пробег автомобилей Lcc, км ’);

Readln(Lcc);

Остальные данные будут вводится из внешнего файла. Структуру программы с возможностью чтения исходных данных из файла и записи результата в файл можно представить в виде

```
Program ProgrTO;
Var   {Раздел описания переменных}
   Ai,Lcc,Drg:integer;
   Lg,Alfa:real;
   Fr:text;   {Объявляем две новые переменные: файл для чтения исходных }
   Fw:text;  {данных (Fr) и файл для записи результатов расчета (Fw)}
```

```
Begin
Assign(Fr,'ProgrTO.dat');  {Внешний файл для чтения исходных данных с именем ProgrTO.dat}
Assign(Fw,'ProgrTO.rez'); {внешний файл для записи результатов  с именем ProgrTO.rez }
Reset(Fr);  {Открытие внешнего файла считывания исходных данных с именем 'ProgrTO.dat }
Rewrite(Fw);  {Открытие  нового файла с именем ProgrTO.rez для записи результатов }
Writeln('Введите количество автомобилей в парке, Аи');
Readln(Ai);
Writeln('Введите среднесуточный пробег автомобилей Lcc, км');
Readln(Lcc);
read(Fr,Drg,Alfa);{Считываем из файла ProgrTO.dat значения Drg и Alfa}
Lg:=Ai*Lcc*Drg*Alfa;{Вычисление годового пробега парка автомобилей }
{Заносим в файл ProgrTO.rez результаты расчета }
writeln(Fw,'Результаты расчета годового пробега автомобилей:');
writeln(Fw);
writeln(Fw,'Lg=',Lg:6:2);
Close(Fw);Close(Fr); {Закрытие файлов исходных данных и результатов расчета}
end.
```

При выводе данных, определенного типа, по умолчанию отводится определенное количество позиций. Существует форматный и бесформатный вывод числовых значений. Бесформатный вывод данных записывается в виде

```
writeln(Fw,'Alfa=',Alfa);
```

В этом случае значение Alfa будет представлено в виде Alfa = 0.8900000000E+00. При выводе целых констант производится в виде

```
writeln(Fw,'Drg=',Drg:m);
```

В этом случае под числовое значение переменной Drg отводится m позиций. При m = 4 Drg = 302. Если число занимает меньше позиций чем отведено в операторе writeln, то свободные позиции слева заполняются пробелами.

Для вывода вещественных констант применяется следующий формат

```
writeln(Fw,'Lg=',Lg:6:2);
```

где 6 – общее число позиций выводимой переменной Lg, включая знак числа, целую часть, точку и дробную часть, 2 – число позиций дробной части. В этом случае переменная Lg выводится в виде константы с фиксированной точкой. Если в результате расчета переменная Lg = 12365422,589 км, то при таком формате вывода ее значение будет выведено в виде Lg = 12365422.59. В курсовой работе при выводе числовых значений необходимо указывать их размерность, например,

```
writeln(Fw,'Lg=',Lg:6:2,' км');
```

В отличие от предыдущего вывода, теперь значение переменной будет равно Lg = 12365422.59 км.

Все вычисления в программе осуществляются с использованием оператора присваивания, который записывается в следующем виде

имя переменной:=арифметическое выражение;

Здесь имя переменной, текущее значение которой заменяется новым значением, определяемым данным арифметическим выражением. В операторе присваивания переменная и результат выполнения арифметического выражения должны иметь один и тот же тип. Разрешено присваивать переменной типа real значение выражения типа integer.

Арифметические выражения строятся из числовых констант, переменных, стандартных функций и операций над ними. В Паскале используются операции сложения (+), вычитания (-), умножения (*), деления (/), целочисленного деления (div), остаток от целочисленного деления (mod). В любом выражении, если одна или более переменных имеют вещественный тип, то результат будет также вещественного типа. В операциях div и mod оба операнда должны быть целого типа. В паскале нет операции возведения в степень. При необходимости ее использования применяют стандартные функции

$$X^a соответствует EXP(a * Ln(X));$$

В арифметических выражениях принят следующий приоритет действий: вычисление значений стандартных функций, умножение и деление, сложение и вычитание. При необходимости изменить приоритет действий можно использованием скобок.

Наиболее часто используемые стандартные функции Турбо-Паскаля приведены в таблице 2.5.

Таблица 2.5 Встроенные стандартные функции

Функции	Назначение
Abs(x)	Вычисление абсолютного значения (x)
Sqr(x)	Вычисления квадрата (x)
Sin(x)	Вычисление синуса (x)
Cos(x)	Вычисление косинуса (x)
Arctan(x)	Вычисление арктангенса (x)
Exp(x)	Вычисление экспоненты (x)
Ln(x)	Вычисление натурального логарифма (x)
Sqrt(x)	Вычисление корня квадратного (x)
Trunc(x)	Вычисление целой части (x)
Round(x)	Округление (x) в сторону ближайшего целого
Odd(x)	True, если x – нечетное, False, если x-четное
Sin(x)/cos(x)	Tg(x)
Cos(x)/Sin(x)	Ctg(x)

Окончание таблицы 2.5

Функции	Назначение
1/Sin(x)	Csc(x)
$Arctg(x/(1-x^2))^{1/2}$	Arcsin(x)
$P_I/2$-Arcsin(x)	Arccos(x)
$P_I/2$-Arctg(x)	Arcctg(x)
Ln(x)/Ln(a)	$Log_a(x)$

В тригонометрических функциях синуса и косинуса аргумент должен быть задан только в радианах. Если аргумент необходимо задать в градусах (G), то для их перевода используется формула $X = G*P_i/180$.

Если в программе имеются несколько ветвей, отличающихся друг от друга содержанием вычислений, то выход вычислительного процесса на ту или иную ветвь алгоритма определяется исходными данными задачи.

Оператор управления IF реализует алгоритмическую структуру «Разветвление» и изменяет порядок действий в зависимости от истинности или ложности некоторого условия.

Существует две формы записи оператора IF- полная и краткая.

Полная форма записи имеет вид:

IF <условие> THEN <оператор 1> ELSE <оператор 2>;

Краткая форма записи:

IF <условие> THEN <оператор>;

Оператор может быть как простым, так и составным. Составной оператор представляет собой совокупность последовательно выполняемых операторов, заключенных в операторные скобки BEGIN и END.

В языке Паскаль также используется вложенная конструкция оператора IF.

IF <условие 1> THEN <оператор 1> ELSE

IF <условие 2> THEN <оператор 2> ELSE <оператор 3>;

Условие, использующееся в записи оператора условного перехода IF, записывается или в виде выражения отношения или логического выражения.

Логические выражения строятся из логических констант и переменных, логических операций и операций отношения. В операциях отношения могут использоваться, помимо переменных и констант, арифметические и логические выражения, символьные константы. Используются следующие логические операции в порядке их приоритета, при этом действия выполняются над логическими операндами: Not – отрицание, and – логическое «и», т.е. объединяющее два логических условия, которые должны выполнятся одновременно, or – логическое «или», которое предполагает выполнение одного из условий записанных в выражении, xor – исключающее «или».

Таблица 2.6 Операции отношения

Условное обозначение	Действие	Пример записи
=	Равно	X= X+B или a=в
< >	Не равно	X < > Y
<	Меньше	A+b < c
>	Больше	C > a+b
<=	Меньше или равно	C<=1
>=	Больше или равно	C>=1

Выполнение действий в логическом выражении осуществляется в следующей последовательности: вычисление стандартных функций, арифметических операций, операций отношения и логические операции. Изменить порядок действий можно с использованием круглых скобок.

Пример: Составить фрагмент программы для выбора нормативного простоя автомобилей в ТО и ремонте в соответствии с таблицей 2.2.

```
Program TOP;
VAR
Dto,G,A:real;
BEGIN
 Writeln('Введите грузоподъемность грузового автомобиля G');
Writeln('особо малой грузоподъемности - 1');
 Writeln('малой грузоподъемности - 2');
Writeln('средней грузоподъемности - 3');
Writeln('большой грузоподъемности:');
Writeln('свыше 5 т и до 6т - 4');
Writeln('свыше 6 т и до 8т - 5');
Writeln('особо большой грузоподъемности:');
Writeln('свыше 8 т и до 10 т - 6');
Writeln('свыше 10 т и до 16 т - 7');
 Readln(G);
If G=1 then Dto:=0.25;
If G=2 then Dto:=0.30;
If G=3 then Dto:=0.35;
If G=4 then Dto:=0.38;
If G=5 then Dto:=0.43;
If G=6 then Dto:=0.48;
If G=7 then Dto:=0.53;
Writeln('Введите класс автобуса  A');
Writeln('особо малого - 1');
 Writeln('малого - 2');
Writeln('среднего - 3');
Writeln('большого - 4');
```

```
  Writeln('особо большого - 5');
Readln(A);
  If A=1 then Dto:=0.20;
  If A=2 then Dto:=0.25;
  If A=3 then Dto:=0.30;
  If A=4 then Dto:=0.35;
  If A=5 then Dto:=0.45;
Writeln('Dto=',Dto:7:3)
   END.
```

При необходимости пропуска части программы или выполнения отдельных ее ветвей используется оператор безусловного перехода. Оператор безусловного перехода имеет вид: **GOTO** <метка>

Метка в Турбо Паскале — это произвольный идентификатор, позволяющий именовать некоторый оператор программы и таким образом ссылаться на него. В качестве меток допускается использовать целые числа без знака.

Метка располагается непосредственно перед помечаемым оператором и отделяется от него двоеточием.

Перед тем как использовать метку в программе, она должна быть описана. Описание меток состоит из зарезервированного слова **LABEL** (метка), за которым следует список меток.

Действие оператора **GOTO** состоит в передаче управления соответствующему меченому оператору.

Правила использования меток:

- метка, на которую ссылается оператор **GOTO,** должна быть описана в разделе описаний;

- метки, описанные в процедуре (функции) локализуются в ней.

Например, если в предыдущем примере требуется сделать разветвление в зависимости от типа подвижного состава, т. е. для грузовых автомобилей выполнить одну часть программы, а для автобусов другую, то программа примет вид.

```
      Program TO1;
      Label 1,2;
      VAR
Dto,G,A,T:real;
                  BEGIN
   Writeln('Произвести выбор для автобусов - 1');
   Writeln('Произвести выбор для грузовых автомобилей - 2');
   Readln(T); {T обозначает тип подвижного состава, т.е. грузовой автомобиль
или автобус}
      IF T=1 then Goto 1
   Writeln('Введите грузоподъемность грузового автомобиля G');
  Writeln('особо малой грузоподъемности - 1');
```

```
 Writeln('малой грузоподъемности - 2');
Writeln('средней грузоподъемности - 3');
Writeln('большой грузоподъемности:');
Writeln('свыше 5 т и до 6т - 4');
Writeln('свыше 6 т и до 8т - 5');
Writeln('особо большой грузоподъемности:');
Writeln('свыше 8 т и до 10 т - 6');
Writeln('свыше 10 т и до 16 т - 7');
 Readln(G);
If G=1 then Dto:=0.25;
If G=2 then Dto:=0.30;
If G=3 then Dto:=0.35;
If G=4 then Dto:=0.38;
If G=5 then Dto:=0.43;
If G=6 then Dto:=0.48;
If G=7 then Dto:=0.53; Goto 2;
1:Writeln('Введите класс автобуса  A');
Writeln('особо малого - 1');
 Writeln('малого - 2');
Writeln('среднего - 3');
Writeln('большого - 4');
Writeln('особо большого - 5');
Readln(A);
If A=1 then Dto:=0.20;
If A=2 then Dto:=0.25;
If A=3 then Dto:=0.30;
If A=4 then Dto:=0.35;
If A=5 then Dto:=0.45;
2:Writeln('Dto=',Dto:7:3)
 END.
```

Оператор выбора **CASE** является обобщением оператора **IF** и позволяет сделать выбор из произвольного числа имеющихся вариантов. Он состоит из выражения, называемого селектором, и списка операторов, каждому из которых предшествует список констант выбора.

Формат оператора:

CASE <выражение-селектор> **OF**
<константа 1>: <оператор 1>;
<константа 2>: <оператор 2>;
…
<константа n>: <оператор n>
ELSE <оператор>
END.

Значение выражения и констант должно быть одного и того же скалярного типа, кроме вещественного. Использование строкового типа в качестве селектора запрещено.

Ветвь оператора **ELSE** является необязательной. Если она отсутствует, и значение выражения не совпадает ни с одной из перечисленных констант, то весь оператор рассматривается как пустой. В отличие от оператора IF перед словом **ELSE** точку с запятой можно ставить. Рассмотрим вышеприведенный пример с использованием оператора выбора **CASE**.

```
Program TO2;
VAR
Dto:real; ,G,A:integer;
BEGIN
 Writeln('Введите грузоподъемность грузового автомобиля G');
Writeln('особо малой грузоподъемности - 1');
 Writeln('малой грузоподъемности - 2');
Writeln('средней грузоподъемности - 3');
Writeln('большой грузоподъемности:');
Writeln('свыше 5 т и до 6т - 4');
Writeln('свыше 6 т и до 8т - 5');
Writeln('особо большой грузоподъемности:');
Writeln('свыше 8 т и до 10 т - 6');
Writeln('свыше 10 т и до 16 т - 7');
 Readln(G);
Case G of
  1: Dto:=0.25;
  2: Dto:=0.30;
  3: Dto:=0.35;
  4: Dto:=0.38;
  5: Dto:=0.43;
  6: Dto:=0.48;
  7: Dto:=0.53; End;
Writeln('Введите класс автобуса  A');
  Writeln('особо малого - 1');
   Writeln('малого - 2');
  Writeln('среднего - 3');
  Writeln('большого - 4');
  Writeln('особо большого - 5');
Readln(A);
Case A of
  1: Dto:=0.20;
```

```
  2: Dto:=0.25;
  3: Dto:=0.30;
  4: Dto:=0.35;
  5: Dto:=0.45; End:
Writeln('Dto=',Dto:7:3)
   END.
```

Можно упростить несколько выбор переменной D_{to}, если производить выбор одновременно для грузовых автомобилей и автобусов, для этого обозначим простой соответственно D_{tog} и D_{toa}. В этом случае в операторе выбора Case необходимо использовать операторные скобки BEGIN и END. Программа будет иметь вид

```
        Program TO3;
        VAR
Dto:real; ,G,A:integer;
        BEGIN
    Writeln('Введите грузоподъемность грузового автомобиля G');
   Writeln('особо малой грузоподъемности - 1');
    Writeln('малой грузоподъемности - 2');
   Writeln('средней грузоподъемности - 3');
   Writeln('большой грузоподъемности:');
   Writeln('свыше 5 т и до 6т - 4');
   Writeln('свыше 6 т и до 8т - 5');
   Writeln('особо большой грузоподъемности:');
   Writeln('свыше 8 т и до 10 т - 6');
   Writeln('свыше 10 т и до 16 т - 7');
    Readln(G);
Writeln('Введите класс автобуса  A');
   Writeln('особо малого - 1');
    Writeln('малого - 2');
   Writeln('среднего - 3');
   Writeln('большого - 4');
   Writeln('особо большого - 5');
Readln(A); {Т.к. переменные A и G принимают частично одинаковые значения,
то в качестве селектора выбираем G}
Case G of
  1: Begin Dtoa:=0.20; Dtog:=0.25; end;
  2: Begin Dtoa:=0.25; Dtog:=0.30; end;
  3: Begin Dtoa:=0.30; Dtog:=0.35; end;
  4: Begin Dtoa:=0.35; Dtog:=0.38; end;
  5: Begin Dtoa:=0.45; Dtog:=0.43; end;
```

```
  6: Dtog:=0.48;
  7: Dtog:=0.53; End;
Writeln('Dtoa=',Dtoa:7:3, 'Dtog=',Dtog:7:3)
   END.
```

Для программирования циклических алгоритмов используются три вида операторов, позволяющие многократно повторить какой-то фрагмент программы. Различают оператор цикла с предварительным условием While, который имеет вид

While логическое выражение **do** оператор;

При необходимости повторения нескольких операторов используется составной оператор **While**

While логическое выражение **do Begin** операторы **End**;

Оператор цикла **While** действует следующим образом. Предварительно проверяется логическое выражение, оно может принимать значения **True**– истина, или **False** – ложь. Если True, то выполняется оператор или составной оператор, записанные после ключевого слова **do**. После чего снова производится проверка логического условия. Выход из цикла осуществляется, если логическое условие ложно. Если условие изначально ложно, то цикл пропускается и управление передается оператору, записанному после этого цикла. Этот вид цикла целесообразно использовать в том случае, если какой-то параметр изменяется с дробным шагом, например 0,1; 0,5 и т.д. Рассмотрим это на примере расчета годового пробега автомобилей по формуле 2.12, при условии, что коэффициент технической готовности $\alpha_т$ изменяется от 0,5 до 1,0 с шагом 0,1.

```
Program God;
Var
   Ai,Lcc,Drg:integer;
   Lg,Alfa:real;
                  BEGIN
   Writeln('Введите количество автомобилей в парке, Аи');
   Readln(Ai);
   Writeln('Введите среднесуточный пробег автомобилей Lcc, км');
   Readln(Lcc);
   Writeln('Введите количество дней работы автомобилей в году, Dрг');
   Readln(Drg); Alfa:=0.5;
   writeln('Результаты расчета годового пробега автомобилей:');
   While Alfa<=1 do Begin Lg:=Ai*Lcc*Drg*Alfa; {Начало цикла}
   writeln('Lg=',Lg:6:2); Alfa:= Alfa+0.1; End;        {Конец цикла }
end.
```

В данном примере распечатка результатов производится за каждым шагом расчета, при необходимости сохранения всех промежуточных значений Lg,

эту переменную необходимо описать в разделе Var, объявить как массив, т.е. Lg_i.

Аналогичное назначение имеет и оператор цикла Repeat, отличие его состоит в том, что границы цикла заранее определены. Цикл начинается с ключевого слова Repeat и заканчивается ключевым словом Until, после которого записывается логическое условие. Еще одно его отличие, он выполняется хотя бы один раз и не требует составного оператора. Выход из цикла происходит в том случае, если логическое условие принимает значение False. Программу расчета для предыдущего примера можно представить в следующем виде

```
Program God1;
Var
   Ai,Lcc,Drg:integer;
   Lg,Alfa:real;
Begin
   Writeln('Введите количество автомобилей в парке, Аи');
   Readln(Ai);
   Writeln('Введите среднесуточный пробег автомобилей Lcc, км');
   Readln(Lcc);
   Writeln('Введите количество дней работы автомобилей в году, Dрг');
   Readln(Drg); Alfa:=0.5;
   writeln('Результаты расчета годового пробега автомобилей:');
   Repeat   Lg:=Ai*Lcc*Drg*Alfa; {Начало цикла}
   writeln('Lg=',Lg:6:2); Alfa:= Alfa+0.1; Until   Alfa>=1;       {Конец цикла }
end.
```

Если необходимо использовать в расчетах упорядоченный массив данных, то в этом случае целесообразно использовать оператор цикла For, общий вид которого имеет вид

For i:=n **to** m **do** оператор;

При необходимости повторения нескольких операторов используется составной оператор

For i:=n **to** m **do Begin** операторы **end**;

Здесь переменная i служит параметром цикла, значение которой изменяется при каждом шаге расчета от n до m с шагом 1. Параметры i, n, m должны быть одного и того же скалярного типа (кроме real), или выражением кроме вещественного типа real. Выход из цикла осуществляется при достижении параметром i своего конечного значения m. В нутрии цикла нельзя изменять ни начальное n, ни конечное значение m параметра цикла i, а также само его значение. После завершения цикла значение параметра i становится неопределенным, за исключением выхода из цикла с использованием оператора безусловного перехода **Goto**.

Рассмотрим действие этого оператора на примере расчета годового пробега автомобилей по формуле 2.12 с учетом того, что определяем годовой пробег для парков с различным количеством автомобилей A_i, где i =1..100.

```
Program God1;
Var
   Lcc,Drg:integer; Ai:array[1..100] of integer;
   Alfa:real; Lg:array[1..100] of real;
                              Begin
   Writeln(‘Введите количество автомобилей в парке, Ai’);
   For i:=1 to 100 do Readln(A[i]);
   Writeln(‘Введите среднесуточный пробег автомобилей Lcc, км’);
   Readln(Lcc);
   Writeln(‘Введите количество дней работы автомобилей в году, Dрг’);
   Readln(Drg);
   Writeln(‘Введите коэффициент технической готовности, αт’);
   Readln(Alfa);
   For i:=1 to 100 do Lg[i]:=A[i]*Lcc*Drg*Alfa;
   writeln('Результаты расчета годового пробега автомобилей:');
   For i:=1 to 100 do writeln('Lg’,’(‘,I,’)=',Lg[i]:6:2);
                                 end.
```

При необходимости изменения нескольких переменных, которые заданы в виде таблиц (матриц), применяются вложенные циклы.

Например, требуется определить годовой пробег автомобилей различных моделей (количество их одинаковое), работающих на разных маршрутах, скорость движения V_{ij} на них заданы в виде матрицы. Автомобили работают на линии 8 часов.

Таблица 2.7 Скорости движения автомобилей на маршрутах

№ маршрута	Модель автомобиля		
	МАЗ	ЗИЛ	ГАЗ
1	60,2	55,8	62,3
2	58,9	52,3	60,1
3	56,3	50,9	55,6

```
Program God2;
Var
   A,Drg:integer; V,Lg:array[1..3,1..3] of real;
   Lcc ,Alfa:real;
Begin
   Writeln(‘Введите количество автомобилей в парке, A’);
   Readln(A);
   Writeln(‘Введите количество дней работы автомобилей в году, Dрг’);
```

```
  Readln(Drg);
  Writeln('Введите коэффициент технической готовности, αт');
  Readln(Alfa);
  {Расчет среднесуточного пробега}
  For i:=1 to 3 do {Перебираем маршруты}
  For j:=1 to 3 do Lg[i,j]:=A*(V[i,j]*8)*Drg*Alfa; {j- модели автомобилей}
  writeln('Результаты расчета годового пробега автомобилей:');
  For i:=1 to 3 do
  For j:=1 to 3 do writeln('Lg','(',i,j,')=',Lg[i,j]:6:2);
end.
```

После составления программы осуществляется компиляция текста в машинный код, которая осуществляется специальной программой среды.

В результате компоновки программы происходит объединение частей программы, добавление стандартных функций и подпрограмм, установление взаимосвязи отдельных частей программы.

Важным этапом является отладка программы, т.е. выявление ошибок, которые были сделаны в программе на различных этапах. Достоинством интегрированной среды является то, что устранение ошибок осуществляется, не выходя из среды.

Выявление ошибок на этом этапе не заканчивается. После запуска программы и ее выполнения могут быть выявлены связанные с ошибками организации вычислительного процесса, неправильным объявлением типа переменных и другие. Сообщения об ошибках, выдаваемые на различных этапах отладки программы, приведены в приложении 3 /3/.

Таким образом, после выполнения программы или ее части, т.е. при выводе промежуточных результатов, необходимо провести их анализ на реальность полученных значений и только в этом случае программа считается приемлемой.

В курсовой работе программа представляется в виде распечатки программного файла. В программе должен быть предусмотрен вывод такой информации: назначение программы (в соответствии с заданием), наименование факультета, группы, Ф.И.О. исполнителя и даты завершения работы с программой.

2.7. Инструкция пользователя

В этом разделе необходимо привести характеристику разработанной программы: назначение, область применения для реальных автотранспортных предприятий, ограничений, связанных, например, с количеством единиц подвижного состава, количеством и типом подвижного состава и др. Здесь же необходимо оговорить порядок ввода исходных данных, их формат. Указать, какие данные приняты как константы для данной программы, какие вносятся в отдельный файл данных, а какие вводятся с клавиатуры в ходе выполнения про-

граммы, размерность величин. Приводятся сообщения, которые выдаются программой в ходе выполнения, и как выйти из конфликтных ситуаций. Какие промежуточные результаты могут быть получены при дополнительной доработке программы.

2.8. Результаты расчета

Этот раздел представляет собой распечатку результатов расчета.

Величины, которые определяются по определенной формуле, должны выводиться с указанием определяемой величины и в виде формулы с подстановкой числовых значений.

Например, годовой пробег автомобилей (количество автомобилей в парке равно 200 ед., годовой пробег автомобилей – 100 км, коэффициент технической готовности – 0,9, количество дней работы подвижного состава в году – 365 дн.)

$$L_г = 200 \cdot 100 \cdot 0{,}9 \cdot 365 = 6570000 \text{ км}$$

Вышеприведенная программа Progr TO для вывода этой формулы будет иметь вид

```
Program ProgrTO;
Var
   Ai,Lcc,Drg:integer;
   Lg,Alfa:real;
   Fr:text;   Fw:text;
              Begin
   Assign(Fr,'ProgrTO.dat');
   Assign(Fw,'ProgrTO.rez');
   Reset(Fr);  Rewrite(Fw
   Writeln(‘Введите количество автомобилей в парке, Аи’);
   Readln(Ai);
   Writeln(‘Введите среднесуточный пробег автомобилей Lcc, км’);
   Readln(Lcc);
   read(Fr,Drg,Alfa);
   Lg:=Ai*Lcc*Drg*Alfa;
   writeln(Fw,'Результаты расчета годового пробега автомобилей:');
   writeln(Fw);
   {Вывод расчетной формулы с подстановкой исходных данных}
   writeln(Fw,'Lг=',Ai:3,’ * ‘,Lcc:3,’ * ‘,Drg:3,’ * ‘,Alfa:4:2,’= ‘,Lg:6:2,’ км’);
   Close(Fw);Close(Fr);
              end.
```

Если расчетная величина является массивом данных, то она приводится в виде таблицы, с указанием всех входных и выходных параметров и их размерности.

Эти величины, кроме того, приводятся в виде графических зависимостей и оформляются как рисунки (например, рисунок 2.4, 2.5), для построения которых используются средства статистического и инженерного анализа /5/.

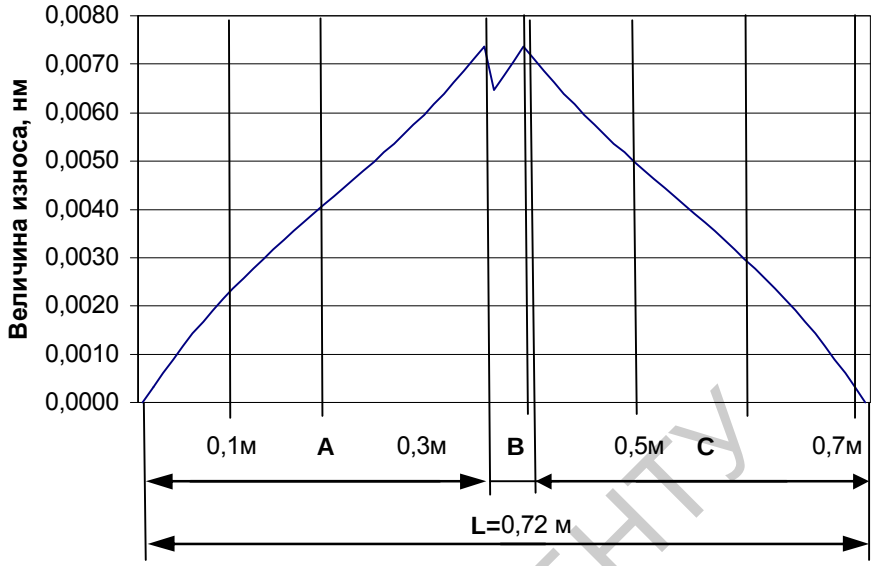


Рисунок 2.4 – Распределение износа направляющих горизонтально- шлифовального станка (материал направляющих – СЧ 21-40, материал ползуна – сталь 40Х, среднее давление распределенной нагрузки – 0,15 МПа)

В состав MS Excel 2000 входят очень мощные и удобные средства анализа данных (составляющие так называемый Пакет анализа), предназначенные для решения сложных статистических, инженерных задач. Для проведения анализа с помощью этих инструментов достаточно указать входные данные и выбрать параметры обработки. Анализ проводится с помощью соответствующей статистической или инженерной макрофункции, а результат размещается в выходной диапазон.

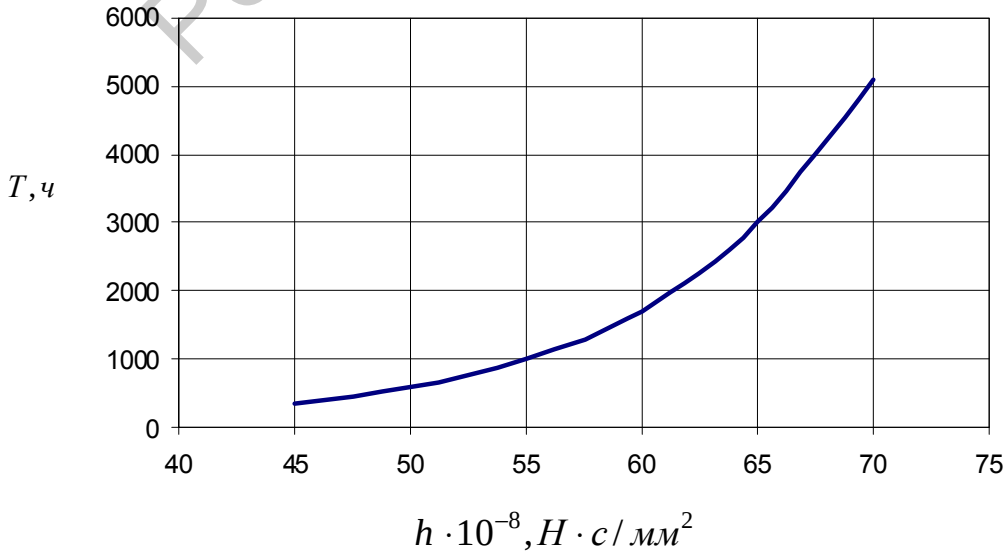


Рисунок 2.5 – Зависимость ресурса работы подшипника от вязкости смазки.

Используя графические средства, результаты анализа можно представить в виде диаграмм или графиков.

Аппарат статистического анализа в MS Excel представлен:

- функциями дисперсионного анализа;
- аппаратом построения гистограмм;
- генератором последовательностей случайных чисел, распределенных по нормальному, равномерному и другим законам;
- функциями, относящимися к описательной статистике: среднее значение, отклонение от среднего, наиболее вероятное значение, степень надежности.

Кроме этого статистические инструменты MS Excel позволяют моделировать случайные процессы и анализировать данные с использованием методов факторного и регрессионного анализа.

В список инженерных функций включены функции ошибок, средства преобразования данных из одной физической системы единиц в другую, функции преобразования данных из одной системы счисления в другую, комплексная арифметика.

2.8.1 Построение гистограммы числовых распределений по результатам расчета

Построение гистограммы распределения случайных чисел рассмотрим на примере распределения наработки автомобиля на отказ, представляющую собой ряд неупорядоченных чисел. Для построения гистограммы выполним следующие действия:

– создадим новый файл MS Excel и введем последовательность чисел в ячейки A1:An, где n – обозначает количество исходных данных;

– в столбце B в ячейках B3:Bm введем границы частотных интервалов от min до max с некоторым шагом, где m – количество интервалов (рекомендуется 7…10). Можно и не указывать границы частотных интервалов, в этом случае машина сама определит количество равных интервалов;

– далее необходимо открыть окно диалога инструмента Сервис\ Анализ данных/ Гистограмма;

– теперь необходимо ввести параметры. Переходим в поле «Входной интервал», в котором указываются номера ячеек с исходной информацией. Для этого выделяем мышью (n) ячеек. Входной интервал задан;

– вводим интервал карманов. Для этого переходим в поле «Интервал карманов» и выделяем мышью ячейки B3:Bm;

– отмечаем «Вывод графика»;

– в разделе «Параметры вывода» переходим на переключатель «Выходной интервал»;

– переходим в поле «Выходной интервал» и указываем ячейку C1 в качестве левой верхней ячейки для выходных данных. После этого осуществляется редактирование гистограммы:

– редактирование названия и размерности осей;

– редактирование названия самой гистограммы;

– осуществляется выбор вида гистограммы.

Чтобы описать экспериментальное распределение выделяем точки полученной ломанной линии, нажатием левой кнопки мыши, затем нажимая правую кнопку выделяем в подменю «Добавить линию тренда». Проводим аппроксимацию и сглаживание, выбирая «Тип» линии тренда:

- линейная;
- логарифмическая;
- полиномиальная;
- степенная;
- экспоненциальная;
- линейная фильтрация.

Выбирая «Параметры» ставим флажок на «Показывать уравнение на диаграмме» и «Поместить на диаграмму величину достоверности аппроксимации», на поле диаграммы будет записано уравнение, описывающее распределение случайной величины и достоверность аппроксимации. Пример диаграммы приведен на рисунке 2.6.

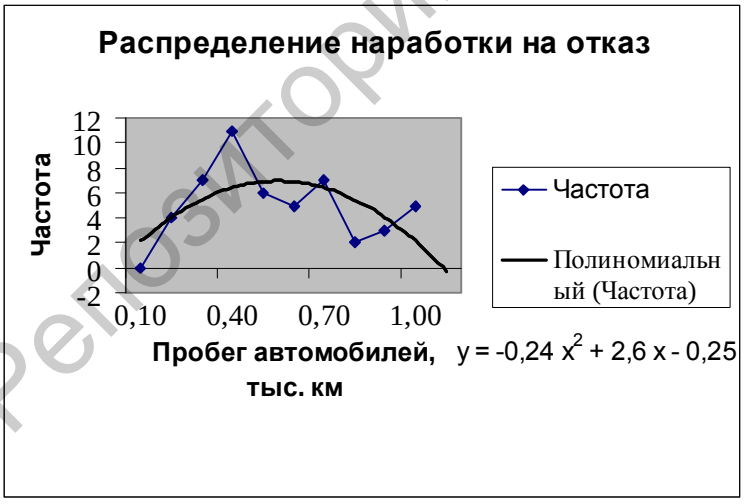


Рисунок 2.6 Диаграмма распределения наработки автомобилей на отказ.

Инструмент «Описательная статистика» позволяет построить статистический отчет для исходных данных. Выходная таблица содержит два столбца информации для каждого набора данных. Левый столбец содержит названия статистических показателей, а правый – значения этих параметров.

Порядок действий для определения статистических характеристик:

– выбирается команда Сервис\Анализ данных\Описательная статистика;

– задаются параметры определения характеристик распределения:

– ссылка на входной диапазон, содержащий анализируемые данные. Она должна состоять не менее чем из двух смежных диапазонов данных, данные в

которых расположены по строкам или столбцам. Например, указываются первая и последняя ячейки расположения исходных данных т.е. A1:A50;

– группирование данных осуществляется установкой переключателя в положение «По столбцам» или «По строкам» в зависимости от расположения данных во входном диапазоне;

– для установления уровня надежности необходимо ввести требуемое значение. Например, значение 95%;

– определяем левую верхнюю ячейку выходного диапазона. Этот инструмент анализа выводит два столбца сведений для каждого набора данных. Левый столбец содержит метки статистических данных; правый столбец содержит статистические данные. Состоящий их двух столбцов диапазон статистических данных будет выведен для каждого столбца или для каждой строки входного диапазона в зависимости от положения переключателя «Группирование»;

– для вывода параметров статистического распределения «Итоговая статистика» необходимо отметить в выходном диапазоне по одному полю для каждого из следующих видов статистических данных: среднее, стандартная ошибка (среднего), медиана, мода, стандартное отклонение, дисперсия выборки, эксцесс, асимметричность, интервал, минимум, максимум, сумма, счет и т.д. Например:

Установите флажок, если в выходном диапазоне необходимо получить по одному полю для каждого из следующих видов статистических данных: Среднее, Стандартная ошибка (среднего), Медиана, Мода, Стандартное отклонение, Дисперсия выборки, Эксцесс, Асимметричность, Интервал, Минимум, Максимум, Сумма, Счет, Наибольшее (#), Наименьшее (#), Уровень надежности. Например:

Характеристики распределения	
Среднее	0,522
Стандартная ошибка	0,034
Медиана	0,48
Мода	0,48
Стандартное отклонение	0,24
Дисперсия выборки	0,059
Эксцесс	-0,99
Асимметричность	0,39
Интервал	0,87
Минимум	0,12
Максимум	0,99
Сумма	25,58
Счет	49
Наибольший(1)	0,99
Наименьший(1)	0,12
Уровень надежности(95,0%)	0,69

На основании анализа графических зависимостей дается заключение о выборе оптимального варианта решения поставленной задачи

Список использованных источников

1. Ильина М.М. Vord 2000. - М.: Лаборатория базовых знаний, 2000. – 544 с.

2. МИ БНТУ 3.001-2003. Единая система стандартизации БНТУ. дипломное проектирование. – Мн.: БНТУ, 2003.- 41 с.

3. Офицеров Д.З., Старых В.А. Программирование в интегрированной среде Турбо-Паскаль. – Мн.: Беларусь, 1992. – 240 с.

4. Информатика: Путеводитель абитуриента и старшеклассника. Авт.-сост. Н.А.Подольская . – М.: Научно-технический центр «Университетский», 1998. – 128 с.

5. Симонович С.В., Евсеев Г.А., Алексеев А.Г. Специальная информатика: Учебное пособие. – М.: АСТ-ПРЕСС:Инфорком-Пресс, 1999 – 480 с.

6. Бондарев В.М., Рублинецкий В.И., Качко Е.Г. Основы программирования. – Харьков: Фолио; Ростов н/Д: Феникс, 1997. – 368 с.

7. Рахмина Г.В. Excel 2000. Руководство пользователя с примерами. – М.: Лаборатория базовых знаний, 2001. – 592 с.

Приложение 1

Министерство образования Республики Беларусь

Белорусский национальный технический университет

Автотракторный факультет

Кафедра «Техническая эксплуатация автомобилей»

Курсовая работа

по дисциплине «Информатика»

Тема: (например) Расчет годовой производственной программы
по техническому обслуживанию автомобилей

Исполнитель	студент АТФ, курс (№ группы) Ф.И.О.
Руководитель, должность	Ф.И.О. преподавателя

Минск – (год)

Приложение 2

Министерство образования Республики Беларусь

Белорусский национальный технический университет

Автотракторный факультет

Кафедра «Техническая эксплуатация автомобилей»

Пояснительная записка

к курсовой работе по дисциплине «Информатика»

Тема: (например) Расчет годовой производственной программы
по техническому обслуживанию автомобилей

Исполнитель	студент АТФ, курс (№ группы) Ф.И.О.
Руководитель, должность	Ф.И.О. преподавателя

Минск – (год)

Приложение 3

Сообщение об ошибках при отладке программы

Код ошибки	Содержание
Сообщения компилятора TURBO PASCAL (версии 5.0 и 5.5) об ошибках	
1	выход за границы памяти
2	не указан идентификатор
3	неизвестный идентификатор
4	повторный идентификатор
5	синтаксическая ошибка
6	ошибка в действительной константе
7	ошибка в целой константе
8	строковая константа превышает размеры строки
9	слишком много вложенных файлов
10	неправильный конец файла
11	строка слишком длинная
12	нужен идентификатор типа
13	слишком много открытых файлов
14	неверное имя файла
15	файл не найден
16	диск заполнен
17	неправильная директива компилятора
18	слишком много файлов
19	неопределенный тип в определении ссылки
20	нужен идентификатор переменной
21	ошибка в определении типа
22	слишком большая структура
23	базовый тип множества нарушает границы
24	компонентами файла не могут быть файлы (и объекты в версии 5.5)
25	неверная длина строки
26	несоответствие типов
27	неправильный базовый тип отрезка типа
28	нижняя граница больше верхней
29	нужен порядковый тип
30	нужна целая константа
31	нужна константа
32	нужна целая или действительная константа
33	нужен идентификатор типа
34	неправильный тип результата функции
35	нужен идентификатор метки
36	нужен BEGIN
37	нужен END
38	нужно выражение типа Integer
39	нужно выражение перечисляемого типа
40	нужно выражение типа Boolean
41	типы операндов не соответствуют оператору
42	ошибка в выражении

Продолжение приложения 3

Код ошибки	Содержание
43	неверное присваивание
44	нужен идентификатор поля
45	объектный файл слишком большой (больше 64 Кбайт)
46	неопределенная внешняя процедура
47	неправильная запись объектного файла
48	сегмент кода слишком большой (больше 65520 байт)
49	сегмент данный слишком велик
50	нужен оператор DO
51	неверное определение PUBLIC
52	неправильное определение EXTRN
53	слишком много определений типа EXTRN (больше 256)
54	требуется OF
55	требуется интерфейсная секция
56	недействительная перемещаемая ссылка
57	требуется THEN
58	требуется TO или DOWNTO
59	неопределенное опережающее описание
60	слишком много процедур (больше 512 в одном модуле)
61	неверное преобразование типа
62	деление на нуль
63	неверный файловый тип
64	нет возможности считать или записать переменные данного типа
65	нужно использовать переменную-указатель
66	нужна строковая переменная
67	нужно выражение строкового типа
68	программный модуль не найден
69	несоответствие имен программных модулей
70	несоответствие версий программных модулей
71	повторное имя программного модуля
72	ошибка формата файла программного модуля
73	требуется секция реализации
74	типы констант и тип выражения оператора case не соответствуют друг другу
75	нужна переменная типа запись
76	константа нарушает границы
77	нужна файловая переменная
78	нужно выражение типа указатель
79	нужно выражение типа real или integer
80	метка не находится внутри текущего блока
81	метка уже определена
82	неопределенная метка в предыдущем разделе операторов
83	недействительный аргумент оператора @
84	нужно UNIT
85	нужно указать “;”
86	нужно указать “:”
87	нужно указать “,”

Продолжение приложения 3

Код ошибки	Содержание
88	нужно указать “(”
89	нужно указать “)”
90	нужно указать “=”
91	нужно указать “:=”
92	нужно “[” или “(.”
93	нужно “]” или “.)”
94	нужно “.”
95	нужно “..”
96	слишком много переменных
97	неправильная переменная цикла оператора FOR
98	нужна переменная целого типа
99	здесь не допускаются файлы
100	несоответствие длины строковой переменной или константы
101	неверный порядок полей
102	нужна константа строкового типа
103	нужна переменная типа integer или real
104	нужна переменная перечисляемого типа
105	ошибка в операторе INLINE
106	предшествующее выражение должно иметь символьный тип
107	слишком много перемещаемых элементов
108	недостаточно памяти для выполнения программы
109	нет возможности найти файл .EXE
110	модуль выполнять нельзя
111	компиляция прервана с помощью клавиш <Ctrl-Break>
112	константа оператора CASE находится вне границ
113	ошибка в операторе
114	нет возможности вызвать процедуру прерывания
115	для компиляции необходимо наличие сопроцессора 8087
116	для компиляции необходим режим 8087
117	адрес назначения не найден
118	в такой ситуации включаемые файлы не допускаются
119	ошибка формата файла .TPU
120	нужен NIL
121	неверный квалификатор переменной
122	недействительная ссылка на переменную
123	слишком много символов (больше 64 Кбайт)
124	слишком большой раздел операторов (больше 24 Кбайт)
125	в модуле нет отладочной информации
126	файлы должны иметь параметры VAR
127	слишком много условных символов
128	пропущена условная директива
129	пропущена директива ENDIF
130	ошибка в начальных условных определениях
131	заголовок не соответствует предыдущему определению
132	критическая ошибка диска
133	нельзя вычислить данное выражение

Продолжение приложения 3

Код ошибки	Содержание
134	некорректное завершение выражения
135	неверный спецификатор формата
136	недопустимая косвенная ссылка
137	здесь не допускается использование структурной переменной
138	нельзя вычислить без блока System
139	доступ к данному символу отсутствует
140	недопустимая операция с плавающей запятой
141	нельзя выполнить компиляцию оверлеев в память
142	должна использоваться переменная – процедура или функция
143	недопустимая ссылка на процедуру или функцию
144	этот модуль не может использоваться в качестве оверлейного
Новые сообщения компилятора версии 5.5	
147	в данном контексте ожидается объектный тип
148	определение объектного типа внутри процедур или функций недопустимо
149	пропущено ключевое слово VIRTUAL
150	в данном контексте ожидается идентификатор метода
151	конструктор должен быть статическим методом
152	в данном контексте ожидается идентификатор конструктора
153	в данном контексте ожидается идентификатор деструктора
154	стандартная процедура Fail может быть использована только внутри конструкторов
Сообщения об ошибках выполнения программ	
1	не найден файл
3	не найден маршрут
4	слишком много открытых файлов
5	отказано в доступе к файлу
6	недопустимый файловый канал
12	недействительный код доступа к файлам
15	недопустимый номер дисковода
16	нельзя удалить текущий каталог
17	нельзя при переименовании указывать разные дисководы
100	ошибка чтения диска
101	ошибка записи на диск
102	файлу не присвоено имя
103	файл не открыт
104	файл не открыт для ввода
105	файл не открыт для вывода
106	неверный числовой формат
150	диск защищен от записи
151	неизвестный модуль
152	дисковод находится в состоянии “не готов”
153	неопознанная команда
154	ошибка в исходных данных
155	при запросе к диску указана неверная длина структуры
156	ошибка при операции установки головок на диске

Окончание приложения 3

Код ошибки	Содержание
157	неизвестный тип носителя
158	сектор не найден
159	кончилась бумага на устройстве печати
160	ошибка при записи на устройство
161	ошибка при чтении с устройства
162	сбой аппаратуры
200	деление на нуль
201	ошибка при проверке границ
202	переполнение стека
203	переполнение динамически распределяемой области памяти
204	недействительная операция ссылки
205	переполнение при операции с плавающей запятой
206	исчезновение порядка при операции с плавающей запятой
207	недопустимая операция с плавающей запятой
208	не установлена подсистема управлениями оверлеями
209	ошибка чтения оверлейного файла

СОДЕРЖАНИЕ