МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Белорусский национальный технический университет

Кафедра «Горные машины»

ПРОГРАММИРОВАНИЕ В СРЕДЕ TURBO PASCAL 7.0

Лабораторный практикум для студентов
специальностей 1-36 10 01 «Горные машины и оборудование
(по направлениям)» и 1-36 13 01 «Технология и оборудование
торфяного производства»

В 3 частях

Часть 3

СТРУКТУРИРОВАННЫЕ ТИПЫ ДАННЫХ И ПОДПРОГРАММЫ

Рекомендовано учебно-методическим объединением
по образованию в области горнодобывающей промышленности

Минск
БНТУ
2018

УДК 004.438(076.5)
ББК 22.18я7
П78

Составитель
С. М. Петренко

Рецензенты:
кафедра «Высшая математика и информатика»
ЧУО «Институт современных знаний имени А. М. Широкова»
(зав. кафедрой – *В. Д. Слепцов*);
В. Т. Поляков

П78 **Программирование** в среде Turbo Pascal 7.0 : лабораторный практикум для студентов специальностей 1-36 10 01 «Горные машины и оборудование (по направлениям)» и 1-36 13 01 «Технология и оборудование торфяного производства» : в 3 ч. / сост. С. М. Петренко. – Минск: БНТУ, 2014–2018. – Ч. 3: Структурированные типы данных и подпрограммы. – 2018. – 36 с.
ISBN 978-985-550-803-9 (Ч. 3).

Практикум содержит лабораторные работы по дисциплине «Информатика», выполнение которых позволяет получить практические навыки программирования на алгоритмическом языке Турбо Паскаль с использованием массивов, записей и подпрограмм.

Часть 2 «Программирование типовых алгоритмов» вышла в 2018 г. (сост. С. М. Петренко).

УДК 004.438(076.5)
ББК 22.18я7

ISBN 978-985-550-803-9 (Ч. 3)
ISBN 978-985-550-354-6

© Белорусский национальный технический университет, 2018

ВВЕДЕНИЕ

Лабораторный практикум предназначен для пользователей, которые уже владеют навыками программирования типовых алгоритмов на языке Турбо Паскаль с использованием простых типов данных.

Сложные (структурированные) типы определяют упорядоченные совокупности компонент из базовых простых типов.

Основное назначение практикума – получение пользователями практических навыков разработки алгоритмов и составления по ним программ с применением структурированных типов данных и подпрограмм. В практикум включены лабораторные работы только по наиболее часто используемым сложным типам данных: массивам и записям. Кроме того, рассмотрены типовые алгоритмы использования подпрограмм-функций и подпрограмм-процедур.

Практикум ориентирован прежде всего на студентов-заочников, изучающих программирование самостоятельно. Поэтому задания на каждую лабораторную работу предваряются вводной информацией, в которой кратко изложены основные аспекты алгоритмического языка по рассматриваемой теме, включая примеры программного хода с необходимыми комментариями.

Зарезервированные слова алгоритмического языка, а также имена функций и подпрограмм из модулей библиотеки Турбо Паскаля в примерах и пояснениях выделены жирным шрифтом (кроме комментариев). Символ «;» в конце строки в примерах программ является оператором «продолжать».

Отчет по каждой лабораторной работе должен содержать титульный лист, задание, блок-схему алгоритма, распечатку исходной Паскаль-программы и результатов ее выполнения, анализ результатов. Отчеты оформляются на листах писчей бумаги формата А4 в соответствии с требованиями Стандарта БНТУ (СТП 10-02.01–87. Единая система учебной документации. Отчет о лабораторной работе. Общие требования и правила оформления. Минск, 1987).

Лабораторная работа № 1

ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ МАССИВОВ

Цель работы: освоить типовые алгоритмы обработки массивов.

Вводная информация

Массивы и их объявление в программе

Массив – это структурированный тип данных, который содержит упорядоченный фиксированный набор элементов, имеющих один и тот же базовый тип. Максимальное число элементов массива задается при его объявлении и в процессе работы программы не изменяется (динамические массивы с нефиксированным числом элементов в данной лабораторной работе не рассматриваются). Для объявления массива с фиксированным числом элементов используется специальная конструкция АЯ Паскаль:

ИмяМассива: **Array** [диапазон индексов] **of** ТипКомпонентов;

Все элементы массива имеют одинаковое имя, совпадающее с объявленным именем массива. Доступ к конкретному элементу осуществляется по его индексу в одномерном массиве или по набору индексов в многомерном массиве. В качестве индекса можно использовать выражение любого скалярного типа, кроме вещественного.

Примеры описания одномерных массивов:

Vector: **array** [-10..10] **of real**; {одномерный массив из 21 элемента}

Sps: **array** [‘a’..’z’] **of integer**; {одномерный массив из 26 элементов}

Для описания многомерных массивов можно использовать в качестве типа компонентов массива объявление другого массива

Matr: **array** [1..20] **of array** [1..10] **of byte**; {матрица 20 × 10}

или записывать диапазоны индексов в квадратных скобках, разделяя их запятыми.

Matr: **array** [1..20, 1..10] **of byte**; {матрица 20 × 10}.

Первый вариант наиболее наглядно представляет описание матрицы как массива массивов, второй вариант имеет более компактную запись.

При использовании в программе нескольких массивов целесообразно объявить пользовательский тип:

```
Type Massiv = array [1..5, 1..10, 1.. 3] of integer;
Var
A,B,C,D: Massiv; {4 трехмерных массива}.
```

В некоторых случаях удобно применять массивы, описанные как типизированные константы, элементы которых можно легко изменять. Например, для задания пунктов меню можно использовать следующую конструкцию:

```
Type Menu = array [1..3] of string [9];
Const
Punkts: Menu = (‘ Справка ’, ‘ Решение ‘, ‘Результат ‘);
```

В памяти массивы хранятся как сплошные последовательности элементов, при этом в многомерных массивах с несколькими индексами быстрее всего изменяется самый «дальний» индекс.

Размерность массивов в Турбо Паскале не ограничена, но сумма размеров всех компонентов массива не должна превышать 65520 байт (почти 64 К).

Использование массивов в выражениях

Массивы в целом могут участвовать только в операции присваивания вида

Ms1:= Ms2;

и в операциях отношения «равно» или «не равно», например

If Ms1< > Ms2 **then** k:= k+1;

При этом массивы Ms1 и Ms2 должны иметь одинаковую структуру и тип элементов.

В арифметических и логических выражениях используются элементы массива. Над элементами массивов можно выполнять те же действия, что и над обычными переменными:

```
Vector[1] := x+y;
z:= B[1,7,2] * 3.3;
D[1,3,2] := A[2,4,1] + B[3,7,2];
```

Как правило, при обработке массивов необходимо последовательно обращаться к конкретному числу элементов массива или ко всем элементам массива (выполнять перебор элементов). В этих случаях удобно использовать цифровые индексы, которые можно легко изменять с помощью циклов с параметром.

Ввод элементов массива с клавиатуры

Фрагмент программного кода ввода одномерного массива B(10)

B: **array** [1..10] **of byte**;

с использованием цикла с параметром имеет вид:

```
For i := 1 to 10 do {i – индекс элемента массива}
  begin
  Write (‘ Введите B[‘ , i:2, ‘]= ’); {вывод подсказки об индексе элемента}
  Readln (B[i]); {чтение значения элемента с текущим индексом i }
  end; .
```

При вводе двумерного массива C(4 × 5)

```
C: array [1..4, 1..5] of real;
```

используются два цикла, один – внешний, второй – вложенный:

```
For i := 1 to 4 do {i –номер строки, внешний цикл}
For j := 1 to 5 do {j –номер столбца, вложенный цикл}
begin
Write (‘C[‘, i:2, ’, ‘ , j:2, ‘]= ?’); {вывод подсказки об индексах элемента}
  Readln (C[i, j]); {чтение значения элемента с текущими индексами i и j}
  end; .
```

Здесь элементы матрицы вводятся по строкам, так как внешний цикл – по номерам строк i. При внешнем цикле по номерам столбцов j реализуется ввод по столбцам.

Соответственно при вводе трехмерного массива необходимо использовать три цикла, четырехмерного – четыре и т. д.

Вывод элементов массива на дисплей

Вывод элементов одномерного массива B(10) из примера ввода по пять элементов в строке с указанием индексов реализуется следующим программным кодом:

```
For i := 1 to 10 do {i – индекс элемента массива}
  begin
  write (‘B[‘ , i:2, ‘]= ’, B[i]:5); {вывод 5-ти элементов в одной строке}
If i mod 5 = 0 then writeln; {условие перехода на новую строку}
  end; .
```

Вывод элементов двумерного массива C(4 × 5) в виде матрицы выполняется построчно. Используются два цикла по аналогии с вводом:

```
For i := 1 to 4 do {i – номер строки}
begin
  For j := 1 to 5 do {j – номер столбца}
  begin
  Write (C[ i, j]:5:2); {вывод элементов i-той строки}
  end; {for по j}
Writeln; {переход на новую строку}
  end; . {for по i}
```

Поиск максимального или минимального элемента массива

При поиске максимального элемента массива за максимальный принимается первый элемент, затем выполняется сравнение каждого последующего элемента с ним. Если в процессе последовательного перебора обнаруживается элемент больше, чем принятый за максимальный элемент, то его значение становится максимальным и используется для сравнения с последующими элементами. Пример программного кода поиска максимального элемента массива C(4 × 5):

```
Var
C: array [1..4, 1..5] of real;
max: real; i, j, imax, jmax: byte;
BEGIN
max := C[1,1];
For i := 1 to 4 do
  For j := 1 to 5 do
  begin
  If C[i, j] > max then begin
                          max := C[i, j];
                          imax := i; {сохранение индексов
                          jmax := j; максимального элемента}
                          end {if }
  end; {for}
Write ('Максимальный элемент – C[',imax : 2, ',', jmax:2, ' ] ' );
Writeln(' равен', max:5:2);
END.
```

При поиске минимального элемента массива алгоритм аналогичный: за минимальный принимается первый элемент массива (min := C[1,1] ;), а затем в процессе перебора элементов определяется элемент меньше минимального, который используется как минимальный для сравнения последующих элементов с ним по условию:

```
          If C[i, j] < min then begin min := C[i, j];
imin := i; jmin := j; {сохранение индексов}
end {if }.
```

Перестановка элементов массива

Перестановка местами элементов массива осуществляется с помощью дополнительной переменной того же типа, что и базовый тип массива, в которой временно сохраняется один из за-

меняемых элементов. Фрагмент программного кода перестановки максимального и минимального элементов массива C(4 × 5):

```
{значение максимального элемента сохраняется в t}
      t := C[imax , jmax];
{замена значения максимального элемента значением минимального}
      C[imax , jmax] := C[imin , jmin];
{замена значения минимального элемента значением максимального из t}
      C[imin , jmin] = t;.
```

Преобразование двумерного массива в одномерный и наоборот

Преобразование двумерного массива A[i,j] в одномерный B[i] реализуется вычислением индексов одномерного массива. Фрагмент программного кода такого преобразования:

```
For i := 1 to N
  For j := 1 to M
  {вычисление индекса элемента одномерного массива}
  B[(i – 1) * M + j] := A[i,j];
```

Здесь N – число строк, M – число столбцов двумерного массива A[i,j].

Обратное преобразование одномерного массива в двумерный выполняется аналогично:

```
For i := 1 to N
  For j := 1 to M
A[I,j] := B[(i – 1) * M + j];.
```

Задание на лабораторную работу

Разработать алгоритм и составить программу для одного из вариантов задания:

1. Программа выполняет ввод целочисленных элементов массива C(4,2), контрольный вывод матрицы на дисплей, поиск строки с максимальной суммой элементов, вывод этой строки и суммы ее элементов.

2. Программа выполняет ввод вещественных элементов массива A(3,3) по строкам, контрольный вывод массива на дисплей в виде матрицы, определяет максимальный и минимальный элементы массива. Минимальный элемент умножается на максимальный, максимальный делится на минимальный, после чего на дисплей выводится измененная матрица.

3. Программа запрашивает размерность массива вещественных чисел A(m, n), где $m \leq 5$ – число строк, $n \leq 5$ – число столбцов, обеспечивает ввод элементов матрицы по столбцам, суммирование элементов первой строки и первого столбца раздельно и вместе, вывод матрицы и значений сумм на дисплей.

4. Программа выполняет ввод элементов массива A(4×4) вещественных чисел по столбцам и вывод их на дисплей в виде матрицы, определение максимального элемент массива. Затем максимальный элемент меняется местами с первым элементом, а диагональные элементы заменяются нулями. Измененную матрицу вывести на дисплей.

5. Программа определяет максимальный элемент в матрице D(4,4), делит элементы главной диагонали исходной матрицы на найденный элемент и умножает отрицательные элементы на 10. Количество строк и столбцов, элементы матрицы вводятся с клавиатуры, на дисплей выводятся исходная и преобразованная матрицы.

6. Программа выполняет ввод с клавиатуры вещественных элементов матрицы A(3,3), их контрольный вывод в виде матрицы, вычисление суммы всех элементов матрицы и диагональных элементов матрицы, вывод сумм на дисплей.

7. Программа определяет минимальный элемент в матрице D(3,3) и умножает диагональные элементы исходной матрицы на найденный элемент. Количество строк и столбцов, элементы матрицы вводятся с клавиатуры, на дисплей выводится исходная и преобразованная матрицы.

8. Программа выполняет ввод с клавиатуры вещественных элементов матрицы A(5,5), вычисляет сумму элементов, стоящих на главной диагонали и ниже ее. Вывести исходную матрицу, эти элементы и их сумму на дисплей.

9. Программа выполняет ввод с клавиатуры вещественных элементов матрицы C(3,4), контрольный вывод матрицы на дисплей, поиск минимального элемента второй строки и меняет его местами с диагональным элементом второй строки.

10. Программа выполняет ввод с клавиатуры вещественных элементов матрицы A(4,3), их контрольный вывод в виде матрицы, вычисление количества и суммы положительных элементов матрицы и вывод количества и суммы на дисплей.

11. Программа выполняет ввод элементов массива A(5,5) по строкам, определяет максимальный и минимальный элементы массива и выводит эти элементы на экран с указанием их индексов. Минимальный элемент умножается на 10 и меняется местами с максимальным. На дисплей выводятся исходная и измененная матрицы.

12. Программа выполняет ввод по строкам вещественных значений элементов матрицы A[2,4] и матрицы B[2,4], определение максимальных элементов матриц A и B, вывод этих элементов на экран с указанием их индексов. Произвести замену местами максимальных элементов матриц. Вывести измененные массивы на дисплей.

13. Программа выполняет ввод с клавиатуры вещественных элементов матрицы A(4,4), вычисляет сумму элементов, стоящих на главной диагонали и выше ее и выводит исходную матрицу, эти элементы и их сумму на дисплей.

14. Программа выполняет ввод по строкам вещественных значений элементов матрицы A(4,4), определяет максимальный и минимальный элементы матрицы и переставляет их местами, а отрицательные элементы заменяет нулями. Ввод элементов матрицы с клавиатуры, вывод первоначальной и измененной матрицы – на дисплей.

15. Программа запрашивает размерность массива вещественных чисел B(m, n), где $m \leq 5$ – число строк, $n \leq 5$ – число столбцов, выполняет ввод элементов матрицы по столбцам и находит суммы элементов первой строки и диагональных элементов. Исходная матрица и найденные суммы выводятся на дисплей.

Лабораторная работа № 2

ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ЗАПИСЕЙ

Цель работы: освоить типовые алгоритмы использования записей.

Вводная информация

Записи и их объявление в программе

Запись – это структурированный тип данных, состоящий из фиксированного числа компонентов, называемых полями. В одном поле могут храниться данные только одного базового типа, а в разных полях могут храниться данные разных типов.

Формат объявления переменной типа «запись» в Турбо Паскале:

```
Record
ИмяПоля1: БазовыйТипДанныхПоля1;
ИмяПоля2: БазовыйТипДанныхПоля2;
    .     .     .     .     .
ИмяПоля N: БазовыйТипДанныхПоляN
End;
```

Если тип данных нескольких полей совпадает, то имена полей можно представить списком и указать один тип для всех:

ИмяПоля1, ИмяПоля2, ИмяПоля3: БазовыйТипДанных;

Тип «запись» позволяет объединить данные одного или нескольких различных типов, например сведения о фамилии, имени и отчестве студента, номере группы, годе окончания школы, годе поступления, среднем балле аттестата и т. д., в одну смысловую группу:

```
Var
Stud: Record
        FIO: string [50];
        NGr: string [10];
        GodOk, GodPost: integer;
        SrBall: real
        End;
```

Второй вариант объявления записей – с использованием пользовательского типа:

```
Type Zap = record
        FIO: string [50];
        NGr: string [20];
        GodOk, GodPost: integer;
        SrBall: real
        End;
Var
     Stud: Zap;
```

Записи с вариантами структуры

В Турбо Паскале можно использовать записи, которые, в отличие от рассмотренных ранее записей с фиксированной структурой, могут содержать произвольное число вариантов структуры. Такие записи называются записями с вариантами и могут содержать как необязательный фиксированный набор полей, так и расположенную в конце записи одну вариантную часть.

Вариантная часть образуется с помощью оператора **case**, который содержит специальное поле признака. Значение этого поля сравнивается со значением констант выбора вариантов полей и определяет, какой из вариантов должен быть использован в данный момент. Типы поля признака и констант выбора должны быть совместимыми.

Формат объявления записи, содержащей фиксированную и вариантную части:

```
Type Varzap = record
              ИмяПоля1: ТипДанныхПоля1; {поля фиксированной
              ИмяПоля2: ТипДанныхПоля2; части}
              Case ПолеПризнака: ТипПоляПризнака of
              КонстантыВыбора1: (ВариантПоля1: Тип ПоляВарианта1);
              КонстантыВыбора2: (ВариантПоля2: Тип ПоляВарианта2);
              . . . . . .
              КонстантыВыбораN: (ВариантПоляN: Тип ПоляВариантаN)
              end; {record }
```

Имена полей вариантов и их типы заключаются в круглые скобки, даже если поле пустое. В операторе **Case** вариантной части отсутствует закрывающее **end** и не применяется **else**.

Пример объявления записи, содержащей варианты по значению поля признака Prizn:

```
Type RezIzm = record
      Obzn : string[15];        {Обозначение заготовки}
      Dln: byte;                {Длина заготовки}
      Shr: byte;                {Ширина заготовки}
      Case Prizn: char of       {Признак единицы измерения}
      'd', 'D': (inh: word);    {в дюймах}
      'c', 'C': (cm: word);     {в сантиметрах}
      'm','M': (metr: real)     {в метрах}
      end; {record}
Var Razmery: RezIzm;
```

В любой момент выполнения программы в такой записи будет активно только одно из трех полей вариантной части, у которого значение одной из констант выбора этого поля совпадает с заданным в программе значением признака Prizn.

Запись может иметь только одну вариантную часть, которая помещается в конце записи после фиксированной части.

Поле вариантной части может быть пустым, но наличие круглых скобок для пустого поля обязательно.

Использование записей в программах

Над записями в целом допускаются только операции присваивания вида

```
Z1 := Z2;
```

при условии, что Z1 и Z2 – записи с идентичной структурой.

Значение компонента записи (поля) используется в выражениях так же, как и обычная переменная. Над содержимым поля записи можно выполнять любые действия, допустимые для данных его типа. В операциях присваивания, ввода-вывода и в качестве операндов выражений используется составное имя поля записи: сначала указывается имя записи-переменной, а затем через точку – имя поля. Примеры использования составного имени поля:

```
Stud. FIO := ‘Петров С.В.‘;
Read (Stud1. SrBall);
Writeln (‘Средний балл Stud1 = ‘, Stud1. SrBall);
If Stud1. GodPost = 2014 then writeln (‘Это студент первого курса‘);.
```

Полем записи может быть другая запись. Например, в запись Stud можно добавить поле Ozn, которое тоже является записью и содержит три поля с экзаменационными оценками студента в зимнюю сессию:

```
Ozn: record
        Ekz1, Ekz 2, Ekz 3: byte;
        End;
```

Тогда для ввода оценки за первый экзамен необходимо использовать составное имя, содержащее имена двух записей:

```
Writen (‘Введите оценку за первый экзамен‘);
Read (Stud. Ozn. Ekz1);.
```

Для устранения этого неудобства работы с длинными составными именами полей в АЯ Паскаль используется оператор присоединения, формат которого:

With ИмяПеременнойТипаЗапись **Do** оператор;.

Если один раз указать переменную типа запись в операторе **With**, то в последующем можно использовать только имена полей этой записи без указания перед ними имени записи. Оператор **With** может быть вложенным (уровень вложения – не более 9), например:

```
With Stud Do
   With Ozn Do begin
                Writen (‘Введите оценку за первый экзамен ‘);
                Readln (Ekz1)
                end;
```

Другой возможный способ записи операторов присоединения

With Stud, Ozn **Do begin** …

эквивалентен предыдущему.

Можно использовать массивы из записей, которые позволяют создавать простейшие базы данных. Например, базу данных Group, содержащую сведения о 25 студентах группы, можно описать следующим образом:

```
Type Stud = Record
            FIO: string [50];
            NGr: string [10];
            GodOk, GodPost: integer;
            Ozn: record
                 Ekz1, Ekz 2, Ekz 3: byte;
                 End;
            SrBall: real
            End;
Var
Group: array [1..25] of Zap;
```

В этом случае составное имя поля Ekz1 будет иметь вид:

Group [i]. Ozn. Ekz1.

Для заполнения полей массива записей и организации поиска по заданному значению поля удобно использовать циклы с параметром. Фрагмент программного кода заполнения полей массива Group(25) имеет вид:

```
With Group Do begin
With Ozn Do begin
For i := 1 to 25 begin {i – порядковый номер в группе}
     Writen (‘FIO ‘, i, ‘ -го студента ?‘);
     Readln (FIO);
     Writen (‘№ группы ‘, i, ‘ -го студента?‘);
     Readln (NGr);
     Writen (‘Год окончания школы ‘, i, ‘ -го студента ?‘);
```

```
      Readln (GodOk);
      Writen (‘Год поступления в ВУЗ ‘, i, ‘ -го студента ?‘);
      Readln (GodPost);
      Writen (‘Средний балл аттестата ‘, i, ‘ -го студента ?‘);
      Readln (SrBall);
      Writen (‘Оценка за первый экзамен ‘, i, ‘ -го студента ?‘);
      Readln (Ekz1);
      Writen (‘Оценка за второй экзамен ‘, i, ‘ -го студента ?‘);
      Readln (Ekz2);
      Writen (‘Оценка за третий экзамен ‘, i, ‘ -го студента ?‘);
      Readln (Ekz3);
      end {for i}
    end {with Ozn}
  end; {with Group}.
```

Задание на лабораторную работу

Разработать алгоритм и составить программу для одного из вариантов задания:

1. Создать массив записей, каждый элемент которого содержит фамилию студента, оценки за четыре экзамена последней сессии, год окончания школы, сведения об изучаемом иностранном языке (английский, немецкий или испанский). Определить студентов, которые окончили школу в прошлом году и имеют оценку по иностранному языку не ниже 5. На дисплей вывести фамилии студентов и их средние баллы за сессию, изучаемый иностранный язык и оценку по иностранному языку. Если удовлетворяющие условиям поиска данные не найдены, вывести сообщение «Таких данных нет».

2. Создать массив записей, каждый элемент которого содержит фамилию студента, оценки по четырем экзаменам последней сессии, сведения о проживании его в общежитии (да – нет) и количестве пропущенных часов занятий за семестр. Программа запрашивает количество студентов в группе

и обеспечивает ввод данных каждого студента. Определить студентов, которые не проживают в общежитии и имеют количество часов пропусков занятий больше заданного. На дисплей выводятся фамилии студентов, сведения о количестве часов пропусков занятий и их средние баллы за сессию. Если удовлетворяющие условиям поиска данные не найдены, вывести сообщение «Таких данных нет».

3. Создать массив записей, аналогичных по структуре с вариантом 2 задания. Определить студентов, которые поступили в текущем году и имеют средний балл больше 6,5. На дисплей выводятся фамилии студентов, их оценки и год окончания школы. Если удовлетворяющие условиям поиска данные не найдены, вывести сообщение «Таких данных нет».

4. Создать массив записей, аналогичных по структуре с вариантом 2 задания. Определить студентов, которые проживают в общежитии и имеют оценку за первый экзамен не менее 5. На дисплей выводятся фамилии студентов, сведения о количестве часов пропусков занятий и их оценки. Если удовлетворяющие условиям поиска данные не найдены, вывести сообщение «Таких данных нет».

5. Создать массив записей, аналогичных по структуре с вариантом 2 задания. Определить студентов, не имеющих пропусков занятий, фамилия которых начинается с заданной буквы. На дисплей выводятся фамилии студентов, их средние баллы за сессию и сведения о проживании в общежитии. Если удовлетворяющие условиям поиска данные не найдены, вывести сообщение «Таких данных нет».

6. Создать массив записей, каждый элемент которого содержит сведения об автомобилях на стоянке и их владельцах и представляет собой запись вида

```
Type Zap = record
Marka: string[20];
GodV: integer;
Color: string[8];
```

```
NReg: string[12];
NPl: byte;
Vlad: record
        FIO: string[30];
        Adr: string[30];
        end;
end;
```

Здесь Marka – марка автомобиля; GodV – год выпуска; Color – цвет; NReg – госномер; NPl – номер места на стоянке; Vlad – сведения о владельце (FIO – фамилия и инициалы; Adr – адрес).

Определить по фамилии владельца и адресу марку автомобиля, его год выпуска, госномер и номер места на стоянке. Если удовлетворяющие условиям поиска данные не найдены, вывести сообщение «Таких данных нет».

7. Создать массив записей по варианту 6. Определить номера мест на стоянке, на которых хранятся автомобили заданной марки с годом выпуска не позже заданного. На дисплей вывести номера мест на стоянке, марку автомобиля, год выпуска и фамилию владельца. Если удовлетворяющие условиям поиска данные не найдены, вывести сообщение «Таких данных нет».

8. Создать массив записей по варианту 6. Определить фамилии владельцев, имеющих автомобили заданной марки и цвета. На дисплей вывести марку автомобиля, его цвет, фамилии владельцев, госномера и номера мест на стоянке. Если удовлетворяющие условиям поиска данные не найдены, вывести сообщение «Таких данных нет».

9. Создать массив записей по варианту 6. Определить фамилии и адреса по марке автомобиля и заданному году выпуска. На дисплей вывести номера мест на стоянке, цвет автомобиля и сведения о владельцах. Если удовлетворяющие условиям поиска данные не найдены, вывести сообщение «Таких данных нет».

10. Создать массив записей по варианту 6. Определить номера мест на стоянке, на которых хранятся автомобили заданного цвета не старше заданного года выпуска. На дисплей вывести номера мест на стоянке, марку автомобиля и адреса владельцев. Если удовлетворяющие условиям поиска данные не найдены, вывести сообщение «Таких данных нет».

11. Создать массив записей, каждый элемент которого содержит результаты анализа образцов, полученных при разведочном бурении, и представляет собой запись вида:

```
Type Zap = record
N: byte;
D: string[10];
Pl: string[10];
F: string[20];
Case h: integer of
      h1: (konz1: real);
      h2: (konz2: real);
      h3: (konz3: real);
end; { Zap }
```

Здесь N – номер образца; D – дата отбора образца; Pl – название площадки; h – глубина отбора в метрах; konz – содержание полезного ископаемого в долях единицы; F – фамилия лаборанта, проводившего анализ. Определить по номеру образца и глубине отбора название площадки и содержание полезного ископаемого в образце, вывести результаты поиска на дисплей. Если удовлетворяющие условиям поиска данные не найдены, вывести сообщение «Таких данных нет».

12. Создать массив записей, аналогичных по структуре заданным в варианте 11. Определить по глубине отбора образцов и фамилии лаборанта даты отбора и номера образцов, анализ которых он проводил, и содержание полезного ископаемого в них, вывести результаты поиска на дисплей. Если

удовлетворяющие условиям поиска данные не найдены, вывести сообщение «Таких данных нет».

13. Создать массив записей, аналогичных по структуре заданным в варианте 11. Определить по дате и глубине отбора образцов их номер и содержание полезного ископаемого, вывести результаты поиска на дисплей. Если удовлетворяющие условиям поиска данные не найдены, вывести сообщение «Таких данных нет».

14. Создать массив записей, аналогичных по структуре заданным в варианте 11. Определить по номеру образца и глубине его отбора дату отбора и содержание полезного ископаемого, вывести результаты поиска на дисплей. Если удовлетворяющие условиям поиска данные не найдены, вывести сообщение «Таких данных нет».

15. Создать массив записей, аналогичных по структуре заданным в варианте 11. Определить по названию площадки, номеру образца и глубине его отбора содержание полезного ископаемого и фамилию лаборанта, вывести результаты поиска на дисплей. Если удовлетворяющие условиям поиска данные не найдены, вывести сообщение «Таких данных нет».

Лабораторная работа № 3

ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ПОДПРОГРАММ

Цель работы: освоить типовые алгоритмы использования подпрограмм.

Вводная информация

Подпрограммы в алгоритмическом языке Турбо Паскаль

Подпрограммой называется логически связанная поименованная группа операторов, реализующая заданную последовательность действий. Запускать подпрограмму на выполнение можно любое количество раз, в различных местах головной программы и с разными значениями используемых в ней переменных путем вызова ее по имени.

В Турбо Паскале используются два вида подпрограмм: подпрограммы-функции (далее по тексту – функции) и подпрограммы-процедуры (далее – процедуры).

Процедуры и функции АЯ Турбо Паскаль подразделяются на стандартные (встроенные) и созданные пользователем.

Стандартные процедуры и функции определены разработчиком языка и входят в его состав. Они вызываются по зарезервированному имени без предварительного их описания в головной программе.

Процедуры и функции, созданные пользователем, должны быть предварительно объявлены в головной программе. Обычно они помещаются в разделе описаний головной программы до начала ее раздела операторов.

Оба вида подпрограмм состоят из заголовка и программного блока, который называется телом подпрограммы. Структура подпрограмм отличается от структуры обычной программы

заголовками и точкой с запятой вместо точки после завершающего **end**. Обычно заголовок подпрограммы содержит в круглых скобках перечень используемых формальных параметров с указанием их типов, которые обеспечивают обмен значениями между вызывающей программой и подпрограммой. Если обмен данными с вызывающей программой алгоритмом подпрограммы не предусмотрен, формальные параметры в заголовке не указываются. Формальные параметры отделяются друг от друга точкой с запятой с указанием их типа.

Раздел описаний подпрограммы по аналогии с обычной программой может содержать свои внутренние (локальные) разделы описания меток, констант, типов и переменных, а также, если в теле подпрограммы используются другие (вложенные) подпрограммы, раздел описания этих подпрограмм. Вложенные процедуры или функции могут вызываться только внутри охватывающей их подпрограммы и недоступны головной программе.

Константы, типы и переменные, объявленные в подпрограмме, являются локальными, то есть действуют только в этой подпрограмме и недоступны для раздела операторов головной программы и других подпрограмм.

Использование функций

Функция – это отдельная программа, предназначенная для вычисления и возврата в головную программу только одного результата, который присваивается ее имени. Поэтому в заголовке подпрограммы-функции, кроме ее имени и перечня используемых формальных параметров с указанием их типов, указывается тип имени функции.

Описание функции имеет вид:

Function Имя (перечень формальных параметров): ТипИмени;

раздел описаний функции;

begin
операторы тела функции;
end;.

При обращении к функции формальные параметры заменяются фактическими параметрами вызывающей программы, которые встают на место формальных и над ними выполняются предусмотренные алгоритмом тела функции действия.

Формальные параметры являются локальными переменными функции и имеют одинаковый статус с локальными переменными, объявленными в ее разделе описания переменных.

Для обращения к функции пользователя используется оператор вида

Имя (список фактических параметров);.

Входные фактические параметры, передаваемые в функцию при ее вызове, записываются в списке через запятую без указания их типа. Количество фактических параметров, их тип и порядок следования должны соответствовать указанным в заголовке функции формальным параметрам.

Фактические параметры являются аргументами функции при вычислениях. Имя функции с аргументами используется в выражениях как операнд. Поэтому этот вид подпрограмм очень удобно использовать для создания пользователем своего набора функций в дополнение к стандартным.

Пример объявления и использования подпрограммы-функции в программе вычисления пяти значений выражения

$$Z = \frac{p \cdot F(x,y) - t \cdot F(x,y)}{1 + p^2 \cdot F(x,y)},$$

в которое входят константы p и t и функция $F(x,y) = (x+y)^{-3} - (x-2y)^3 + 3xy.$

```
Program funk;
const
   p = 2.3; t = 1.2;
var
   x, y, z: real; i: byte;
function Fn (a, b: real;): real; {Заголовок функции Fn}
   var q, w: real; {Локальные переменные функции}
   begin
   q := Exp (–3* ln(a + b));
   w := Exp (3*ln(a – 2*b))
   Fn := q – w +3* a* b; {Присваивание результата имени функции}
   end;
BEGIN
for i:=1 to 5 do
            begin
      write (‘Введите значения x и y ‘);
      readln (x, y);
      z := (p* Fn(x,y) – t* Fn(x,y))/(1+Sqr(p)*Fn(x,y));
      writeln (‘ x=‘, x:5:2, ‘ y= ‘, y:5:2, ‘ z= ‘, z:6:2);
      end {for}
END.
```

Использование процедур

Процедура – это отдельная поименованная программа, аналогичная по структуре функции, которая может либо не возвращать в головную программу никаких результатов, либо возвращать один или несколько результатов. Описание процедуры имеет вид:

Procedure ИмяПроцедуры (перечень формальных параметров);

внутренние разделы описаний процедуры;

begin
операторы тела процедуры;
end;.

Формальные параметры в круглых скобках в заголовке процедуры отделяются друг от друга точкой с запятой с указанием их типа. По назначению они разделяются на входные (параметры-значения) и выходные (параметры-переменные). Входные параметры передают свои значения в подпрограмму при ее вызове. Они могут изменять свои значения в подпрограмме, но эти изменения будут локальными и не изменят стартовое значение входных параметров в головной программе (точнее – в операторе вызова подпрограммы).

Выходным формальным параметрам в процедуре присваиваются возвращаемые в вызвавшую программу результаты вычислений. В перечне формальных параметров они записываются после служебного слова **var**. Это означает, что процедура может изменять значение этих переданных ей параметров-переменных.

Для вызова процедуры используется оператор вида

ИмяПроцедуры (список фактических параметров);

Фактические параметры в списке записываются через запятую без указания их типа. Должно соблюдаться соответствие между формальными и фактическими параметрами по количеству, типу данных и порядку их следования.

В качестве передаваемых в процедуру входных фактических параметров могут использоваться константы, переменные и выражения. Выходными фактическими параметрами могут быть только переменные.

При использовании в качестве параметров процедуры структурированных типов данных (массивы, записи, множества) необходимо описать в головной программе тип пользо-

вателя для этих данных. Объявленный тип пользователя используется в перечне формальных параметров заголовка процедуры.

Когда процедура отработает, управление передается оператору головной программы, следующему за оператором вызова процедуры.

Пример использования процедуры в программе, обеспечивающей ввод элементов матриц M1 (3 × 3) и M2 (3 × 3), поиск и вывод значений максимальных элементов этих матриц с указанием их индексов:

```
Program proc1;
Const N = 3;
Type Mass = array [1..N, 1..N] of real;
Var
M1, M2: Mass;
Imax, Jmax, i, j: byte;
Max: real;
Procedure Maks (B: Mass; R: byte; var Mxelem: real; kmax,
lmax: byte);
var p, t: byte;
begin
Mxelem := B[1, 1];
For p := 1 to R do
   For t := 1 to R do
      If B[p, t] > Mxelem then begin
                                Mxelem := B[p, t];
                                kmax := p; lmax := t;
                                end; {If}
   end; {procedure}
BEGIN
For i := 1 to N do {Ввод элементов матриц}
For j :=1 to N do begin
        Write ('Введите M1( ', i, ',', j, ') '); Readln (M1[i, j]);
```

```
        Write ('Введите M2( ', i, ',', j, ') '); Readln (M2[i, j]);
        end; {for }
  {Поиск максимальных элементов с использованием процедуры}
  Maks (M1, N, Max, Imax, Jmax);   {Обращение к процедуре}
  Write ('Максимальный элемент первого массива – ');
  Writeln (' M1 (', Imax, ',', Jmax, ') = ', Max: 5:2);
  Maks (M2, N, Max, Imax, Jmax);   {Обращение к процедуре}
  Write ('Максимальный элемент второго массива – ');
  Writeln (' M2 (', Imax, ',', Jmax, ') = ', Max: 5:2);
  END.
```

Использование глобальных переменных

Переменные, описанные в головной программе, являются *глобальными* по отношению ко всем подпрограммам этой программы и могут использоваться в теле любой из них. При этом изменение значений глобальных переменных в подпрограмме приводит к изменению их значений в головной программе. Непосредственное изменение глобальных переменных – второй (кроме использования параметров-переменных) способ возвращения результата работы подпрограммы в головную программу.

Пример использования глобальных переменных в программе, которая запрашивает ввод начальных целых значений x и y и количество повторных вычислений z, вычисляет результат деления суммы квадратов x и y на их разность и выводит на дисплей исходные данные и результат. Вычисления выполняются сначала для начальных значений x и y, а затем при каждом повторном вычислении значение x увеличивается на единицу, а значение y – на 2.

```
Program proc2;
Var
```

```
    i, k: byte;
    x, y: integer; {глобальные переменные}
    Rez: real;
    procedure Skw (a, b: byte; var R: real);
    var p: real; {локальная переменная процедуры}
    begin
    {локальные действия в процедуре}
    p := a*a + b*b;
    R := p / (x - y);
    {модификация значений глобальных переменных }
    x := x + 1; y := y + 2;
      end;
  BEGIN
  Write (‘Введите целые начальные значения x и y (x > y) ‘);
Readln (x, y);
  Write (‘Введите число повторений ‘); Readln (z);
  For i := 1 to k do begin
                    Skw (x, y, Rez); {обращение к процедуре}
                    Writeln (‘ x=’, x:3, ‘ y=’, y:3, ‘Rez =’, Rez:6:2);
                    end {for}
  END.
```

При каждом обращении к процедуре значения глобальных переменных x и y изменяются в головной программе.

Задание на лабораторную работу

Разработать алгоритм и составить программу для одного из вариантов задания:

1. Вычислить выражение $Z = \dfrac{a \cdot F(x,y) + b^2 \cdot F(x,y)}{1 - a \cdot F(x,y)}$, где $a =$ $= 0{,}5$, $b = 2{,}1$. Функция имеет вид $F(x,y) = (x+y)^2 - (x-2y)^3 - b.$

2. Вычислить выражение $z = f(u,t) + 2f(u,t) + f(u,t)/3$, где u и t вводятся с клавиатуры. Функция имеет вид: $f(u,t) = $
$= \begin{cases} u - t, & \text{если } u > t, \\ u + t, & \text{иначе.} \end{cases}$

3. Вычислить выражение $p = \dfrac{n!}{m!(n-m)!}$, где $0 \le m \le n$. Вычисление факториалов – в подпрограмме-функции .

4. Вычислить выражение $w = \dfrac{\operatorname{ctg}(x+3)}{x^2 + b} + \log_5 \left| x^5 - 2b \right|$, где $b > 0$ и $x > 0$ вводятся с клавиатуры. Вычисление котангенса и логарифма – в подпрограмме-функции.

5. Вычислить выражение $z = \dfrac{x\sqrt{(x^2 + a)^3} - \operatorname{tg}(x+2)}{\log_3(ax^2 + 3)}$, где $a > 0$ и $x > 0$ вводятся с клавиатуры. Вычисление тангенса и логарифма – в подпрограмме-функции.

6. Вычислить выражение $z = \dfrac{p \cdot t}{p + t}$, где $p = \dfrac{x\sqrt[3]{x+3}}{s}$, $t = \dfrac{e^x \cos x}{s}$ и $s = 3x + x^3$. Значения p и t определить в подпрограмме-процедуре, значение s – в подпрограмме-функции.

7. Выполнить ввод целочисленных элементов матрицы M(4 × 4) и контрольный вывод матрицы на дисплей, определить в подпрограмме-процедуре и вывести на дисплей элементы матрицы, кратные двум.

8. Выполнить ввод элементов матрицы A(3 × 3) и контрольный вывод матрицы на дисплей, определить в подпрограмме-процедуре количество положительных и отрицательных элементов матрицы. Результат работы процедуры вывести на дисплей.

9. Выполнить ввод элементов матрицы B(4 × 3) и контрольный вывод матрицы на дисплей, определить в одной подпрограмме-процедуре количество отрицательных элемен-

тов и сумму диагональных элементов, вывести результаты на дисплей.

10. Выполнить ввод элементов матриц C(3 × 3) и D(3 × 3) и контрольный вывод матриц на дисплей, вычислить в подпрограмме-процедуре разность значений максимального и минимального элементов каждой матрицы с выводом результатов на дисплей.

11. Выполнить ввод элементов матрицы D(4 × 4), контрольный вывод матрицы на дисплей, определить сумму диагональных элементов матрицы в подпрограмме-функции и вычислить в подпрограмме-процедуре значение $Sr = s / k$, где s и k – сумма и количество положительных элементов матрицы, вывести результаты на дисплей.

12. Выполнить ввод с клавиатуры значений вектора X(10), контрольный вывод элементов по пять в строке, определить максимальный элемент вектора X(10) в подпрограмме-функции и значения $M = \frac{\sum x_i}{n}$, $D = \sqrt{\frac{\sum (x_i - M)^2}{n-1}}$ при n = 10 в подпрограмме-процедуре.

13. Программа три раза запрашивает ввод с клавиатуры размеров прямоугольного параллелепипеда (длина, ширина, высота), вычисляет его объем в подпрограмме-процедуре и площадь поверхности в подпрограмме-функции, выводит результаты на дисплей.

14. Программа три раза запрашивает ввод с клавиатуры значений радиуса основания и высоты цилиндра, вычисляет его объем в подпрограмме-функции, а площади боковой и полной поверхности – в подпрограмме-процедуре и выводит результаты на дисплей.

15. Вычислить сумму 15-ти первых членов ряда $a_n = n * \lg n / (\log_2 n)^n$, где $n_{нач}$ = 2, dn = 1. Вычисление десятичного логарифма выполнить в подпрограмме-функции, логарифма по основанию 2 – в подпрограмме-процедуре.

СПИСОК ЛИТЕРАТУРЫ

1. Офицеров, Д. В. Программирование в интегрированной среде Турбо-Паскаль : справочное пособие / Д. В. Офицеров, В. А. Старых. – Минск: Беларусь, 1992. – 240 с.: ил.

2. Офицеров, Д. В. Программирование на персональных ЭВМ : практикум : учебное пособие / Д. В. Офицеров, А. Б. Долгий, В. А. Старых. – Минск: Вышэйшая школа, 1993. – 256 с.

3. Бармин, В. А. Вычислительная техника и программирование: Ч. 1: Программирование в среде TURBO PASCAL 7.0 / В. А. Бармин, А. П. Киселева, В. В. Трикозенко. – Минск: БНТУ, 1998. – 116 с.

4. Фаронов, В. В. Программирование на персональных ЭВМ в среде Турбо-Паскаль / В. В. Фаронов. – Москва: Изд-во МГТУ, 1990. – 580 с.

5. Пильщиков, В. Н. Сборник упражнений по языку Паскаль : учебное пособие / В. Н. Пильщиков. – Москва: Высшая школа, 1990. – 223 с.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3

Лабораторная работа № 1.
ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ
МАССИВОВ..... 4

Лабораторная работа № 2.
ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ
ЗАПИСЕЙ..... 14

Лабораторная работа № 3.
ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ
ПОДПРОГРАММ 25

СПИСОК ЛИТЕРАТУРЫ 35

Учебное издание

ПРОГРАММИРОВАНИЕ В СРЕДЕ TURBO PASCAL 7.0

Лабораторный практикум для студентов
специальностей 1-36 10 01 «Горные машины и оборудование
(по направлениям)» и 1-36 13 01 «Технология и оборудование
торфяного производства»

В 3 частях

Часть 3

СТРУКТУРИРОВАННЫЕ ТИПЫ ДАННЫХ И ПОДПРОГРАММЫ

Составитель
ПЕТРЕНКО Станислав Михайлович

Редактор *Ю. В. Ходочинская*
Компьютерная верстка *Е. А. Беспанской*

Подписано в печать 26.10.2018. Формат 60×84 $^{1}/_{16}$. Бумага офсетная. Ризография.
Усл. печ. л. 2,09. Уч.-изд. л. 1,64. Тираж 100. Заказ 319.

Издатель и полиграфическое исполнение: Белорусский национальный технический университет.
Свидетельство о государственной регистрации издателя, изготовителя, распространителя
печатных изданий № 1/173 от 12.02.2014. Пр. Независимости, 65. 220013, г. Минск.