**Министерство образования Республики Беларусь**
**Учреждение образования**
**“БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ”**

**Кафедра “Вычислительные методы и программирование”**

**А.А. Бурцев, В.П. Шестакович**

# ПРОГРАММИРОВАНИЕ

## МЕТОДИЧЕСКОЕ ПОСОБИЕ К ВЫПОЛНЕНИЮ КОНТРОЛЬНЫХ ЗАДАНИЙ

**ДЛЯ СТУДЕНТОВ ВСЕХ СПЕЦИАЛЬНОСТЕЙ БГУИР ЗАОЧНОЙ ФОРМЫ ОБУЧЕНИЯ**

**В 2-Х ЧАСТЯХ**

**ЧАСТЬ 1**

## ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ ТУРБО ПАСКАЛЬ

**Минск 2004**

УДК 681.3.06 (075.8)
ББК 32.973 я73

Авторы - составители: А.А. Бурцев, В.П. Шестакович

П 78 **Программирование:** Методическое пособие к выполнению контрольных заданий для студ. всех спец. БГУИР заочной формы обуч.: В 2 ч. Ч.1: Основы программирования на языке Турбо Паскаль/Сост. А.А. Бурцев, В.П. Шестакович. – Мн.:БГУИР, 2004.-59с.
ISBN 985-444-462-7 (ч. 1)

В работе приводятся краткие теоретические сведения и методические указания по программированию на языке Турбо Паскаль. Содержатся контрольные задания (7 задач по 30 вариантов) по следующим темам: линейные, разветвляющиеся и циклические алгоритмы, массивы, подпрограммы, строки и записи. Включены тексты программ-примеров.

**УДК 681.3.06 (075.8)**
**ББК 32.973 я73**



**ISBN 985-444-462-7 (ч.1)**
**ISBN 985-444-463-5**

**Содержание**

# ЗАДАНИЕ 1. ЛИНЕЙНЫЙ ВЫЧИСЛИТЕЛЬНЫЙ ПРОЦЕСС

*Цель работы:* изучить основы языка Паскаль, операторы ввода-вывода.

## Алфавит и классификация данных языка Паскаль

Язык Паскаль оперирует следующим набором символов:

1) прописные и строчные буквы латинского алфавита

A, B, C, D,...,U, V, W, X, Y, Z;

a, b, c, d,...,u, v, w, x, y, z;

2) десятичные цифры от 0 до 9;

3) _ - символ “подчеркивание”,

4) специальные символы

| | | | | | |
|---|---|---|---|---|---|
| + | плюс | {} | фигурные скобки | : | двоеточие |
| - | минус | [ ] | квадратные скобки | ; | точка с запятой |
| * | звездочка | ( ) | круглые скобки | ' | апостроф |
| / | дробная черта | # | номер | @ | коммерческое а |
| = | равно | | пробел | $ | знак доллара |
| > | больше | . | точка | ^ | тильда |
| < | меньше | , | запятая | | |

Комбинации специальных символов могут образовывать составные символы:

| | | | |
|---|---|---|---|
| := | присваивание | <= | меньше или равно |
| .. | диапазон значений | >= | больше или равно |
| | | < > | не равно |

(. .) альтернатива квадратным скобкам

(* *) альтернатива фигурным скобкам

5) ключевые слова (зарезервированные). Например: Begin, End;

6) стандартные идентификаторы. Например: Sin, Cos;

7) идентификаторы пользователя.

## Данные

В зависимости от способа хранения и обработки в ЭВМ данные можно разбить на две группы: константы и переменные.

*Константы* - это те данные, значения которых известны заранее и в процессе выполнения программы не изменяются.

В языке Паскаль используются следующие виды констант:

1) числовые константы целые:

| Тип | Диапазон | Требуемая память, байт |
|---|---|---|
| Byte | 0…255 | 1 |
| ShortInt | -128…127 | 1 |
| Integer | -32 768…32 767 | 2 |
| Word | 0…65 535 | 2 |
| LongInt | -2147483648…2147483647 | 4 |

2) числовые константы вещественные (Real);
3) логические (или булевские) константы (Boolean);
4) символьные (или литерные) константы (Char).

*Целые константы* - это целые числа без десятичной точки, которым может предшествовать знак "-" или “+”.

*Пример*:

286; -17; +1995;

Можно использовать целую константу в *шестнадцатеричном* виде. Шестнадцатеричное число состоит из шестнадцатеричных цифр ( 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F ), которым предшествует знак доллара $ ( код символа 36 )

*Пример*:

$3A4F; $100A

*Вещественные константы* могут быть представлены в двух видах: с фиксированной и плавающей точкой. *Константа с фиксированной точкой* - это число, содержащее точку, разделяющую целую и дробную часть (наличие целой и дробной части обязательно).

*Пример*:

-39.013; 0.256;

*Константа с плавающей точкой* - это число, представленное с десятичным порядком: mEp (без пробелов).

Здесь m - мантисса (как целые, так и вещественные числа с фиксированной точкой);

E - признак записи числа с десятичным порядком;

p - порядок числа (только целые числа).

*Пример*:

-7.78E-3; 4.9E5; -0.785E02;

3.14 --> 3.14E+00
0.314E+01
31.4E-01

*Логические константы* могут принимать только одно из двух значений: True (истина) и False (ложь).

*Символьные константы* - это последовательность символов, заключенная в апострофы.

*Пример*:

'B';  '+3ac';  'БГУИР';

В Паскале для определения констант служит ключевое (зарезервированное) слово Const.

Форма описания констант:

Const <идентификатор> = <значение константы>;

Здесь *идентификатор* (имя) должен начинаться с буквы и может содержать буквы (латинские), цифры и знак подчеркивания. Длина имени до 126 символов, но различаются имена по первым 63 символам.

*Пример*:

```
Const
   Min      = 1;
   Max      = 150;
  Argument = 55.0;
```

*Переменная* - это именованный объект, который в процессе выполнения программы может принимать различные значения. Типы переменных можно представить двумя группами: простые и сложные. *К сложным типам* относятся: массивы, строки символов, записи, множества и файлы. *Простые типы* состоят из двух групп: скалярные и пользовательские. К скалярным относятся: целые, вещественные, символьные и логические, а к пользовательским - интервальные и перечисляемые. Переменные скалярного типа могут принимать только значения, совпадающие с константами соответствующих типов.

Форма описания переменных:

Var <идентификатор 1,...> : тип;

*Пример:*

```
Var
   A, B: Integer:
  Sum, Min : Real;
  C: Char;
```

*Замечания:*

1. Для переменных логического типа значение False < True (в операциях сравнения).
2. Переменная символьного типа может принимать значение только одного символа (в отличие от символьной константы).

**Пользовательские типы переменных**

К ним относятся переменные перечисляемого и интервального типов. *Переменная типа «перечисление»* задается перечислением значений, которые она может принимать.

Форма описания этих переменных:

Type <имя типа> = (список значений);
Var <идентификатор 1,...> : <имя типа>;

или

Var <идентификатор> : (список значений);

*Пример:*

```
Type Sezon = (Zima, Vesna, Leto, Osen);
Var S1,S2 : Sezon;
```

или

```
Var S1,S2 : (Zima, Vesna, Leto, Osen);
```

Здесь S1, S2 - переменные типа «перечисление», которые могут принимать любое из заданных значений.

Следует отметить, что описание типа перечисляемой переменной одновременно вводит упорядочение ее значений. Так, для данного примера

Zima < Vesna < Leto < Osen (в операциях сравнения).

Для переменных *интервального* типа указывается некоторое подмножество значений, которые они могут принимать.

Форма описания этих переменных:

Type <имя типа> = <константа 1>..<константа 2>;
Var <идентификатор 1,...> : <имя типа>;

или

Var <идентификатор 1,...> : <константа 1>..<константа 2>;

Здесь <константа 1>, <константа 2> - соответственно константы, определяющие левую и правую границы значений, которые может принимать интервальная переменная. Значение первой константы должно быть обязательно меньше значения второй. Эти константы могут быть целого, символьного или перечисляемого типов.

*Пример:*

```
Type Dni = 1..31;
Var D1, D2 : Dni;
```

В этом примере переменные D1 и D2 имеют тип Dni и могут принимать любые значения из диапазона 1..31. Выход из диапазона вызывает программное прерывание.

Можно определять интервальный тип и более универсальным способом, задав границы диапазона не значениями констант, а их именами.

*Пример:*

```
Const  Min=1; Max=31;
Type  Dni = Min..Max;
Var  D1, D2 : Dni;
```

## Арифметические выражения

*Арифметические выражения* строятся из числовых констант, переменных, стандартных функций и операций над ними. Для обозначения операций используются символы: + сложение, - вычитание, * умножение и / деление.

В арифметическом выражении принят следующий приоритет операций:

1) вычисление значений стандартных функций;
2) умножение и деление;
3) сложение и вычитание.

Порядок выполнения операций изменяется с помощью скобок.

## Стандартные функции

Отметим, что в тригонометрических функциях аргумент должен быть задан только в радианах (см. таблицу ).

| Вызов функции | Тип аргумента | Тип значения | Назначение функции |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| Abs(x) | Целый/вещественный | Как у аргумента | Абсолютное значение x |
| Pi | Целый/вещественный | Вещественный | Значение числа Pi |
| Sin(x) | Вещественный | Вещественный | Синус x (радиан) |
| Cos(x) | Вещественный | Вещественный | Косинус x (радиан) |
| Arctan(x) | Вещественный | Вещественный | Арктангенс x (радиан) |
| Sqrt(x) | Целый/вещественный | Как у аргумента | Квадратный корень из x, x>0 |
| Sqr(x) | Целый/вещественный | Как у аргумента | Значение квадрата x |
| Exp(x) | Вещественный | Вещественный | Значение E в степени x |
| Ln(x) | Вещественный | Вещественный | Натуральный логарифм x, x>0 |
| Trunc(x) | Целый | LongInt | Целая часть значения x |
| Frac(x) | Вещественный | Вещественный | Дробная часть значения x |
| Int(x) | Вещественный | Вещественный | Целая часть значения x |
| Round(x) | Вещественный | LongInt | “Правильное” округление x до ближайшего целого |

Окончание таблицы

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| Random | Вещественный | Вещественный | Следующее число из диап.0<=…<1 |
| Random(x) | Word | Word | Следующее число из диап.0<=…<x |
| Odd(x) | Целый | Логический | Возвращает True, если x–нечетное (x–целое) |
| Succ(x) | Целый | Логический | Возвращает след. за x значение в перечисляемом типе |
| Pred(x) | Целый | Логический | Возвращает предыдущее значение x в перечисляемом типе |

| | | | |
|---|---|---|---|
| Chr(x) | Целый(Byte) | Символьный | Возвращает символ ASCII кода x |
| Ord(x) | Символьный | Целый(Byte) | Возвращает ASCII код символа x |
| Inc(x) | Целый | Целый | Увеличивает значение x на 1 |
| Dec(x) | Целый | Целый | Уменьшает значение x на 1 |
| Inc(x, n) | Целый | Целый | Увеличивает значение x на N |
| Dec(x, n) | Целый | Целый | Уменьшает значение x на N |
| A Div B | Целочисленное деление A на B. Возвращает целую часть частного, дробная часть отбрасывается | | |
| A Mod B | Восстанавливает остаток, полученный при выполнении целочисленного деления. A и B должны быть целого типа | | |

*Пример:*

A = 11; B = 5, тогда A Div B дает 2
A Mod B дает 1
A = 2; B = 3, тогда A Div B дает 0

В Паскале заданы стандартные функции для вычисления трех тригонометрических функций. Для вычисления остальных необходимо использовать известные математические соотношения, например:

ArcSin x = ArcTg (x /Sqrt(1 - x*x));
ArcCos x = Pi/2 - ArcSin x;
ArcCtg x = Pi/2 - ArcTg x;

Для вычисления логарифма с основанием **a** используется соотношение

Log x = Ln x / Ln a;

Возведение **x** в степень **a** осуществляется с использованием стандартных функций:

Exp(a * Ln(x)).

Но таким образом нельзя возвести в целую степень отрицательное число. Это можно сделать с использованием операторов цикла.

## Оператор присваивания

Общий вид оператора:

Имя := Выражение;

Здесь имя - имя переменной, выражение - арифметическое или логическое выражение.

*Пример:*

```
Y := Abs(x) -3.5;
Min := M Div N;
```

В операторах присваивания переменная и выражение должны иметь один и тот же тип, а для переменных интервального типа - одно и то же подмножество значений. Нельзя присваивать целочисленным переменным выражение типа

Real. Однако разрешается присваивать переменной типа Real выражение целочисленного типа.

*Пример:*

```
Y:=Sqrt(Exp(x)) + 2 * Sqr(x) * Sin(x/2) - Exp(5*Ln(x))
```

## Процедура ввода данных

Для вызова *процедуры ввода* используются три оператора:

1) Read (список переменных);

Каждое вводимое значение набирается минимум через один пробел и последовательно присваивается переменным из списка;

2) ReadLn (список переменных);

То же, что и оператор Read, только после ввода данных происходит переход на новую строку (т.е. следующий оператор ввода будет вводить данные с новой строки);

3) ReadLn;

Происходит переход на новую строку без ввода данных.

Последовательно расположенные операторы 1, 3 эквивалентны одному оператору 2. Значения переменных вводятся с терминала и должны соответствовать типам переменных из списка ввода. В Паскале допускается вводить следующие данные: *целые, вещественные, символьные.*

С помощью операторов ввода нельзя ввести значение переменной следующих типов: *логический, перечисляемый, массив* (необходимо вводить значения отдельных элементов массива), *множество* (следует вводить значения элементов множества) и *запись* (необходимо вводить значения отдельных полей записи).

*Пример*:

```
Var A,B,C : Real;
    D,F   : Integer;
     .  .
    Read (A,B,C);
    ReadLn;
    Read (D,F);
```

Для ввода значений переменных на экране набираются числа в следующем порядке:

```
0.5   6.78   -3.974E-1
10   25
```

## Процедура вывода данных

Для вызова *процедуры вывода* используются три оператора:

1) Write (список переменных);

Выводит последовательно значения переменных из списка;

2) Writeln (список переменных);

То же, что и оператор Write, но после вывода переменных осуществляется переход на новую строку (следующий оператор вывода будет выводить данные с начала новой строки);

3) Writeln;

Осуществляется переход на новую строку без вывода данных.

Как и при выводе, последовательно расположенные операторы вида 1, 3 эквивалентны одному оператору 2.

*Допустим вывод следующих данных:* целых, вещественных, символьных, логических, символьных констант, арифметических и логических выражений.

С помощью оператора вывода *нельзя вывести:* значение переменной типа перечисление, массив (необходимо выводить значения отдельных его элементов), множество (следует выводить значения отдельных его элементов), запись (необходимо выводить значения отдельных полей).

## Форматы вывода данных

В процедурах вывода имеются две возможности выводить данные: без указания ширины поля вывода (бесформатный вывод) и с указанием ширины поля вывода (форматный вывод).

*Бесформатный вывод*

*Целые, символьные и логические* - выводятся начиная с позиции курсора.

*Пример*:

```
a=15     Wrire('A=',A);  -->   A=15
c='X'    Write('C=',C);   -->   C=X
d=True   Write('D=',D);   -->   D=TRUE
```

*Вещественные* - выводятся в поле шириной 17 позиций в формате с плавающей точкой. Дробная часть мантиссы содержит 10 цифр.

*Пример:*

```
a=125.286  Write('A=',A);  -->  A= 1.2528600000E+02
b=-2.281e1 Write('B=',B);  -->  B=-2.2810000000E+01
```

Форматный вывод

Общий вид P : M;

Здесь - P - имя переменной; M - ширина поля вывода в позициях.

*Целые* - выводятся в правые крайние позиции поля шириной M.

*Пример:*

```
a=17          Write ('A=',A:5);  -->  A=   17
b=3456        Write ('B=',B:5);  -->  B= 3456
```

*Вещественные* - выводятся в крайние правые позиции поля шириной M в формате с плавающей точкой. Минимальная ширина поля равна 8, в противном случае она игнорируется.

*Пример:*

```
a=134.25        Write('A=',a:11);  --> A= 1.3425E+02
b=-134.25       Write('B=',b:11);  --> B=-1.3425E+02
```

В случае если форматный вывод имеет вид P:M:N, где N - число позиций дробной части, то значение переменной P выводится в виде числа с фиксированной точкой.

*Пример:*

```
r=3.1743   Write(' R=',R:5:2);   -->  7 R= 3.17
```

**Структура программы**

Программа на языке Паскаль состоит из заголовка и блока:

```
(* Заголовок *)
Program имя программы;
(* Блок *)
Uses                    - раздел подключаемых модулей;
Label                   - раздел меток;
Const                   - раздел констант;
Type                    - раздел типов;
Var                     - раздел переменных;
Procedure, Function     - раздел процедур и функций;
Begin
оператор 1;
  .  .  .               - раздел операторов;
оператор n
End.
```

Блок программы состоит из *семи разделов*. Раздел операторов должен присутствовать всегда, остальные разделы могут отсутствовать.

*Раздел Uses* - с его помощью подключаются библиотечные модули из стандартного набора Турбо Паскаля или написанные самим пользователем. Если он присутствует, то должен стоять перед прочими разделами. Кроме того, слово Uses может появиться в программе только один раз.

*Пример:*

```
Uses Crt, Printer, My_Lib;
```

*Раздел меток* (Label). Любой выполняемый оператор может иметь метку. Метки могут обозначаться целыми числами в диапазоне от 0 до 9999 или идентификаторами. Все метки должны быть описаны в разделе Label. Разделов Label может быть несколько, и стоять они могут где угодно до начала основного блока.

Общая форма записи:

Label список меток;

*Пример:*

Label 25, M1, K750, A;

В программе метка отделяется от оператора двоеточием и следующим за ним пробелом.

*Пример:*

A1: X:=X*Sin(Y);

*Раздел констант* (Const). Если в программе используются константы, имеющие достаточно громоздкую запись, либо сменные константы (для разных вариантов программы), то их целесообразно описать в разделе Const, а в программе использовать только имена констант. Раздел Const может располагаться в любом месте, и их может быть несколько или не быть вообще.

*Раздел типов* (Type). Является необязательным разделом. В нем описываются типы переменных, отличающиеся от стандартных, т.е. перечисляемые, интервальные, массивы, записи и т.д. В этом разделе могут быть использованы константы из блока Const. Если это так, то раздел Type может быть расположен где угодно, но не выше соответствующего раздела Const. Если же описания типов ни с чем не связаны, то они могут быть помещены в любом месте между другими разделами, но выше того места, где будут использованы.

*Раздел переменных* (Var) формально тоже необязателен и может отсутствовать. Реально он объявляется и содержит список глобальных переменных программы и их типов. Разделов Var может быть несколько, но переменные в них не должны повторяться.

Если в программе описываются *процедуры или функции*, то их определение должно предшествовать основному разделу (разделу операторов).

*Раздел операторов* - это собственно программа, использующая все, что было описано и объявлено. Он обязательно начинается словом Begin и заканчивается End с точкой. После точки, завершающей основной раздел, любой текст игнорируется.

Операторы языка Паскаль не привязаны к определенной позиции строки. В одной строке можно указывать несколько описаний и операторов. Исполняемые операторы отделяются друг от друга ";". Точка с запятой не ставится после Begin и перед End, так как они являются не операторами, а операторными скобками.

В операторах цикла точка с запятой не ставится после While, Repeat, Do и перед Until.

В условных операторах ";" не ставится после Then и перед Else.

Допускается перенос с одной строки на другую частей операторов и описаний, но без разделения ключевых слов.

*Комментарий* - это пояснительный текст, который можно записать в любом месте программы, где разрешен пробел. Текст комментария ограничен

символами или (* *) и может содержать любые комбинации латинских и русских букв, цифр и других символов алфавита.

В ограничителях (* *) пробелы между скобкой и звездочкой запрещены. Ограничения на длину комментария нет. Ограничители и (* *) удобно использовать при отладке программы.

*Пример* программирования линейного алгоритма. Составить программу для вычисления следующего выражения:

```
Program Lin;
  Var A, B, C : Real;
      W, X : Real;
  Begin
      A := 1.03;
        B := 2.91e-03;     { Ввод исходных данных }
         X := 5.27e-2;
      W := Exp(7 * Ln(A));
        W := Abs(W);
           W := Ln(W) / Ln(10);
      C := Exp(3 * Ln(X));
        C := ArcTan(C);
           C := C * Pi * A / Sqrt(Abs(A + X));
              C := C * (B - Cos(A / B));
      W := W + C;
      Writeln('Исходные данные к задаче:');
      Writeln('a= ',A : 4 : 2,'  b=',B : 9,'  x=',X : 9);
      Writeln;
      Write('Результат: w=',W : 13)
  End.                                    {Ответ: W=9.008913E-02}
```

**Контрольные вопросы**

1. Что такое алгоритм и как он обычно представляется?
2. Что включает в себя алфавит языка Турбо Паскаль?
3. Какие типы констант и переменных существуют в языке Турбо Паскаль?
4. Что такое арифметическое выражение и в каком порядке в нем выполняются вычисления?
5. Какие правила существуют при возведении в степень?
6. Что такое стандартные функции Турбо Паскаля?
7. Что такое арифметический оператор присваивания? Его назначение и правила выполнения арифметических выражений.
8. Каково назначение операторов ввода-вывода?

## Задача 1. Варианты

Составить программу для вычисления выражения.

1.1. $w=2^{-x}\sqrt{x+\sqrt[4]{|y|}}*\sqrt{e^{x-1/\sin x}}$.

1.2. $w=y^{\sqrt[3]{|x|}}+\cos^3(y-3)*\dfrac{|x-y|}{e^{|x-y|}+\dfrac{x}{2}}*\left(1+\dfrac{\sin^2 z}{\sqrt{x+y}}\right)$.

1.3. $w=2^{y^x}+\left(3^x\right)^y-\dfrac{y\left(arctg\ z-\dfrac{p}{6}\right)}{|x|+\dfrac{1}{y^2+1}}$.

1.4. $w=\dfrac{\sqrt{|x-1|}-\sqrt[3]{|y|}}{1+\dfrac{x^2}{2}+\dfrac{y^2}{4}}+x*\left(arctg\quad z+e^{-(x+3)}\right)$.

1.5. $w=\dfrac{\sqrt[4]{y+\sqrt[3]{x-1}}}{|x-y|*\left(\sin^2 z+tg\quad z\right)}$.

1.6. $w=\dfrac{y^{x+1}}{\sqrt[3]{|y-2|}+3}+\dfrac{x+\dfrac{y}{2}}{2*|x+y|}*(x+1)^{-\frac{1}{\sin z}}$.

1.7. $w=\dfrac{x^{y+1}+e^{y-1}}{1+x*|y-tg\quad z|}*(1+|y-x|)+\dfrac{|y-x|^2}{2}-\dfrac{|y-x|^3}{3}$.

1.8. $w=\dfrac{1+x+\dfrac{x^2}{2}+\dfrac{x^3}{3}+\dfrac{x^4}{4}}{x*\left(\sin arctg\quad z-\cos^2 y\right)}$.

1.9. $w=(1+y)*\dfrac{x+\dfrac{y}{x^2+4}}{y^{x-2}+\dfrac{1}{x^2+4}}-\dfrac{1+\cos(y-2)}{\dfrac{x}{2}+\sin^2 z}$.

1.10. $w=y+\dfrac{x}{y+\dfrac{x^2}{y+\dfrac{x^3}{y}}}*\left(1+tg^2\dfrac{z}{2}\right)^{\sqrt{|y|+6}}$.

1.11. $w=\lg\left(\sqrt{e^{x-y}}+x^{|y|}+z\right)*\left(x-\dfrac{x^3}{3}+\dfrac{x^5}{5}\right)$.

1.12. $w=\dfrac{2\cos\left(x-\dfrac{p}{6}\right)}{\dfrac{1}{2}+\sin^2 y}*\left(1+\dfrac{z^2}{3-\dfrac{z^2}{5}}\right).$

1.13. $w=\dfrac{\sqrt[3]{8+|x-y|^2+1}}{x^2+y^2+2}-e^{|x-y|}*\left(tg^2 z+1\right)^x.$

1.14. $w=\dfrac{1+\sin^2(x+y)}{\left|x-\dfrac{2y}{1+x^2y^2}\right|}*x^{|y|}+\cos^2\left(arctg\dfrac{1}{z}\right).$

1.15. $w=\left|\cos x-\cos y\right|^{\left(1+2\sin^2 y\right)}\left(1+z+\dfrac{z^2}{2}+\dfrac{z^3}{3}+\dfrac{z^4}{4}\right).$

1.16. $w=\ln\left(y^{-\sqrt{|x|}}\right)*\left(x-\dfrac{y}{2}\right)+\sin^2 arctgz.$

1.17. $w=\sqrt{10*\left(\sqrt[3]{x}+x^{y+2}\right)}*\left((\arcsin z)^2-|x-y|\right).$

1.18. $w=5\text{arctgx}-\dfrac{1}{4}\arccos x*\dfrac{x+3|x-y|+x^2}{|x-y|^z+x^2}.$

1.19. $w=\dfrac{e^{|x-y|}*|x-y|^{x+y}}{arctg\quad x+arctg\quad z}+\sqrt[3]{x^6+\ln^2 y}.$

1.20. $w=\left|x^{\frac{y}{x}}-\sqrt[3]{\dfrac{y}{x}}\right|+(y+x)*\dfrac{\cos y-\dfrac{z}{y-x}}{1+(y+x)^2}.$

1.21. $w=\dfrac{x+\dfrac{y}{5+\sqrt{x}}}{|y-x|+\sqrt{x}}*e^{n-1}+\arcsin z.$

1.22. $w=y^x+\sqrt[3]{|x|+|y|}*y-\dfrac{z^3*\sin y}{y+\dfrac{z^3}{y-z^3}}.$

1.23. $w=\dfrac{\sqrt[3]{x+\sqrt[4]{|y|}}}{\sqrt{|y|}*e^{-(y-z\cos x)}}.$

1.24. $w=\dfrac{1}{2}*\left(x^{|y-x|}+y^{|x+y|}*\lg\left(\sqrt[3]{z}+\sqrt{x}+\dfrac{2}{\sin z}\right)\right).$

1.25. $w=\left(2+y^2\right)*\dfrac{x+\dfrac{y}{2}}{y^2+\dfrac{1}{1+y^2}}-\sqrt{\sin^2 arctg\left(z\right)+\left|\cos y\right|}.$

1.26. $w=z^{\frac{x+y}{2}}-\sqrt[3]{\dfrac{x-1}{\left|y\right|+1}}*\sin\left(2\arccos z\right).$

1.27. $w=\dfrac{a+b^2}{e^{2x}-\sin\dfrac{x}{2}}-7.2e^x\ln x^2.$

1.28. $w=a+\dfrac{bx}{2.3-1.5*10^{-3}bx^4}+\dfrac{x^3\left(b-1\right)}{\ln\left(\left|x^3-a\right|\right)}.$

1.29. $w=\sqrt{e^x+2\sin x^2}+a^3\left(\dfrac{b}{a+b}\right)^{\frac{2}{3}}.$

1.30. $w=\lg\left|a^7\right|+arctg\quad x^3*\dfrac{p\,a}{\sqrt{\left|a+x\right|}}*\left(b-\cos\dfrac{a}{b}\right).$

## *ЗАДАНИЕ 2. ВЕТВЯЩИЙСЯ ВЫЧИСЛИТЕЛЬНЫЙ ПРОЦЕСС*

*Цель работы:* изучить приемы составления схем алгоритмов ветвящихся вычислительных процессов, операторы безусловной и условной передач управления.

### Логические выражения

*Логические выражения* строятся из логических констант и переменных, операций отношения и логических операций. В операциях отношения могут участвовать арифметические и логические выражения, а также символьные данные. Результатом логического выражения является значение TRUE (истина) или FALSE (ложь).

*Операции отношения* : <, >, =, <=, >=, < >.

*Логические операции*:

Not --> НЕ - логическое отрицание;
And --> И - логическое умножение;
Or --> ИЛИ - логическое сложение.

*В логических выражениях* действия выполняются слева направо с соблюдением следующего старшинства:

1) Not;

2) *, /, Div, Mod, And;
3) +, -, Or;
4) операции отношения.

## Составной оператор

*Составной оператор* - это объединение нескольких операторов в одну группу.

Общий вид оператора :

```
Begin
   оператор 1;
   оператор 2;
     . . .
   оператор n
End;
```

В этой конструкции ключевые слова Begin и End выполняют роль операторных скобок - открывающей и закрывающей. Составной оператор можно вставлять в любое место программы, где допускается один оператор.

*Пример:*

```
Begin
  Y:=Sin(a+b);
  Y:=y - Cos(a-b)
End;
```

В свою очередь, любой из операторов составного оператора также может быть составным. После Begin и перед End ";" можно не ставить.

Нельзя извне составного оператора передавать управление внутрь него (оператором GoTo).

## Оператор безусловного перехода

Как правило, операторы программы выполняются подряд, один за другим. Если в каком-либо месте программы надо перейти не к следующему оператору, а к некоторому другому, то в этом месте помещают оператор перехода, в котором указывается, к выполнению какого оператора следует перейти. Чтобы указать это, необходимо пометить оператор, т.е. написать перед ним метку, отделив ее двоеточием от оператора.

Общий вид: GoTo метка;

*Пример:*

```
    GoTo 50;
25: A:=10;
. . .
50: Y:=X + Pi;
```

После оператора GoTo должен следовать оператор, помеченный меткой (иначе он никогда не будет выполнен).

## Оператор условного перехода IF

Имеет две отличающиеся конструкции:

1) If логическое выражение Then оператор 1;

2) If логическое выражение Then оператор 1
Else оператор 2;

Здесь If (если), Then (тогда), Else (иначе) - ключевые слова.

Перед Else ";" не ставится. Оператор 1, 2 - это простые или составные операторы.

*Работа операторов.* Если значение логического выражения "истина", то выполняется оператор 1, а затем оператор, следующий за If. Если значение логического выражения "ложь", то выполняется оператор, следующий за If, в первой конструкции или оператор, следующий за Else, во второй конструкции.

В качестве операторов 1, 2 могут использоваться другие операторы If.

*Пример*:

Записать оператор If для следующей алгебраической схемы:

$$X = \begin{cases} 5 & при \quad a = b \quad и \quad c < d; \\ 10 & при \quad a = b \quad и \quad c >= d; \\ 15 & при \quad a <> d. \end{cases}$$

```
If a = b Then If c < d Then x:=5
                       Else x:=10
         Else x:= 15;
```

В примере "**;**" ставится в конце оператора. Этот же оператор можно записать с использованием трех операторов If:

```
If  (a = b)  And  (c < d)   Then x:=5;
If  (a = b)  And  (c >= d)  Then x:=10;
If   a <> b                 Then x:=15;
```

В примере наличие скобок в логических выражениях операторов If является обязательным, так как операции сравнения имеют более низкий приоритет, чем логические операции.

## Пустой оператор

*Пустой оператор* никаких действий не задает и никак в программе не изображается. Признаком, по которому его можно обнаружить, служит метка перед ним или наличие символа ";" на том месте, где по правилам должен быть какой-либо оператор.

Пустой *помеченный* оператор употребляется в основном для того, чтобы на него можно было передать управление для выхода из какой-либо более

сложной конструкции программы, например: условного оператора, оператора цикла, процедуры и т.п.

*Пример*:

Пусть требуется написать таблицу функций y=x*sin(x) для x, изменяющегося на интервале [0,P1] с шагом 0,1.

```
   X:=0;
10: If X > P1  Then  GoTo 20;
    Y:=X*Sin(x);
    Writeln('X=',x:4:2,' ':3,'Y=',y:10);
     X:=X + 0.1;
    GoTo 10;
20:  ;
```

**Оператор выбора CASE**

Является обобщением оператора If и позволяет сделать выбор из произвольного числа имеющихся вариантов. Он состоит из выражения, называемого селектором, и списка параметров, каждому из которых предшествует список констант выбора. Как и в If, здесь может присутствовать ELSE, имеющее тот же смысл.

Общий вид: CASE <выражение селектор> OF
<список 1>: <оператор 1>;
. . .
<список n>: <оператор n>;

**Else <оператор>**

END;

*Работа оператора*. Сначала вычисляется значение выражения-селектора, затем выполняется тот оператор, константа выбора которого равна значению выражения-селектора. Если ни одна из констант не равна значению селектора, то выполняется оператор, следующий за ELSE. Если ELSE отсутствует, то выполняется оператор, следующий за END. Селектор должен относиться к целочисленному, булевскому, литерному или пользовательскому типу. Вещественные и строковые типы в качестве селектора запрещены.

*Пример:*

```
  CASE  K + 2  OF
        7: A:=k=10;
      8,9: A:=k+20;
 10..100: A:=k+30
    ELSE A:=100
  END;
```

*Пример:*
Составить программу вычисления выражения:

$$Z = \begin{cases} \sin x * x + \cos(sqrt(y)) & при \quad x > y; \\ \ln(\exp(3/2 * \ln(y)) - x * x * x) - \exp(abs(x - y)) & при \quad 1 < x <= y; \end{cases}$$

```
Program Razv;
  Const  X=1.75;
         Y=0.21e2;
  Var  Z1,Z2 : Real;

  Begin
    Z1:=Sin(X*X) + Cos(Sqrt(Y));
      Z2:=Ln(Exp(2/3 * Ln(Y)) - Sqr(X)*X)-Exp(Abs(X-Y));
    If X > Y Then Write('Выполнилось 1-е условие Z=',Z1:10);
    If (1<X) And (X<=Y) Then
                    Write('Выполнилось 2-е условие Z=',Z2:10);
    If X<=1 Then Write ('Результат вычисления неопределен.')
  End.
```

**Контрольные вопросы**

1. Какой вычислительный процесс называется ветвящимся?
2. Как работает оператор условия If ?
3. Как работает оператор выбора Case ?
4. Какого типа может быть выражение-селектор в операторе Case ?

**Задача 2. Варианты**

Составить программу для вычисления составной функции. Самостоятельно выбрать необходимое количество исходных данных для того, чтобы в программе выполнялись все возможные ветви алгоритма. Перед выводом полученного результата программа должна сообщать о ветви, при прохождении которой он получен.

2.1. $f=\begin{cases} \sin^2(5k + 3m\ln|k|), & если \quad 0 \le k < m; \\ \cos^2(5k + 3m\ln|k|), & если \quad k \ge \dfrac{m}{2}. \end{cases}$

2.2. $f=\begin{cases}\ln\left(\left|2j-3e^2q\right|\right), & \text{если } |j|<5|q|;\\ \ln\left(\left|2j^2-3q\right|\right), & \text{если } 5|q|<|j|\le 7.5|q|.\end{cases}$

2.3. $f=\begin{cases}\sqrt{\left|2k_1-5k_2^2\right|}*e^{k_1+k_2}, & \text{если } 0<k_1\cdot k_2\le 1;\\ \sqrt{\left|2k_1^2+5k_2\right|}*e^{k_1-k_2}, & \text{если } k_1\cdot k_2>1.\end{cases}$

2.4. $f=\begin{cases}\dfrac{4r+3m}{r^3+m^2}*\sin^2 m^3, & \text{если } 0.5<|r|<|m|+0.5;\\ \sqrt{|r-m|}*\cos^3 r^2, & \text{если } |r|>|m|+0.5.\end{cases}$

2.5. $f=\begin{cases}arctg\left(5m^2t+7mt^2\right), & \text{если } m^2+t^2>0.5;\\ \arcsin\left(5m^2t+7mt^2\right), & \text{если } 0.1<m^2+t^2\le 0.5.\end{cases}$

2.6. $f=\begin{cases}\sin^2\left(p\,n_1+e^{n_2}\right), & \text{если } p\le n_1+n_2<5;\\ \sin^2\left(p\,n_2+n_1\right), & \text{если } n_1+n_2\ge 5.\end{cases}$

2.7. $f=\begin{cases}\sqrt{|3m-5r|}*e^{\frac{m}{r}}, & \text{если } r\le m<2r;\\ \sqrt{|3m+5r|}*e^{\frac{r}{m}}, & \text{если } m>2r.\end{cases}$

2.8. $f=\begin{cases}tg^2(c-2k), & \text{если } |c+k|>2;\\ \ln\left(|c-2k|\right)-\sin\dfrac{c}{2k}, & \text{если } 0.5<|c+k|\le 2.\end{cases}$

2.9. $f=\begin{cases}\dfrac{m_1-2m_2}{m_1^2+2m_2^2}, & \text{если } 0.1<|m_1-2m_2|\le 1;\\ 2(m_1-m_2)*e^{\frac{m_1}{m_2}-1}, & \text{если } |m_1-2m_2|>1.\end{cases}$

2.10. $f=\begin{cases}\sqrt{\left|se^2-ne^{-2}\right|}, & \text{если } \dfrac{|n|}{2}<s\le|n|;\\ \sqrt{|s-n|}*\sin^3(s+n), & \text{если } s>|n|.\end{cases}$

2.11. $f=\begin{cases}z^3-\ln\left(|p|+|z|\right), & \text{если } 0<p\le z+1;\\ \ln\left(|p-z|\right)+\cos^2 p, & \text{если } p>z+1.\end{cases}$

2.12. $f=\begin{cases}\sqrt{\left|x*e^{\sin x}+t*e^{-2x}\right|}, & \text{если } 3t\le x<10t;\\ \sqrt{|x+t|}*e^{\cos x}, & \text{если } x\ge 10t.\end{cases}$

2.13. $f=\begin{cases} \dfrac{arctg\,7k-5p}{2\sin k^2+3p^2}, & \text{если } k>|p|; \\ |k-p|*arcctg\,2k, & \text{если } 0.1|p|<k\le|p|. \end{cases}$

2.14. $f=\begin{cases} e^{-|m+r|}+\lg|m|, & \text{если } r>-2m; \\ e^{|m+r|}-\lg|m|, & \text{если } -2.5m<r\le-2m. \end{cases}$

2.15. $f=\begin{cases} e^{-\frac{|a|+|b|}{2}}*ctga, & \text{если } a\cdot b>0.5; \\ |a+b^2|*ctg\,b, & \text{если } 0.4<a\cdot b\le 0.5. \end{cases}$

2.16. $f=\begin{cases} e^{2arcctg\,u}, & \text{если } 0\le n+u\le 2; \\ e^{3arcctg\,n}, & \text{если } n+u>2. \end{cases}$

2.17. $f=\begin{cases} \ln\left(|arctg\ (m)|\right), & \text{если } 1<m<n; \\ \ln\left(|arctg\ (m+n)|\right), & \text{если } m\ge n. \end{cases}$

2.18. $f=\begin{cases} \arcsin\left(|p\,n_1+n_2|\right), & \text{если } 1\le n_1+n_2<p; \\ \arcsin\left(|n_1+p\,n_2|\right), & \text{если } n_1+n_2<1. \end{cases}$

2.19. $f=\begin{cases} \arcsin\left(|5k+2m|\right), & \text{если } 0\le|k|<|m|; \\ \arccos\left(|2k+5m|\right), & \text{если } |k|\ge m. \end{cases}$

2.20. $f=\begin{cases} \sqrt{|2i_1e^{-i_1}-3i_2e^{-i_2}|}, & \text{если } 0<i_1\cdot i_2<1; \\ \sqrt{|\sin^2 i_1+\cos^2 i_2|}, & \text{если } i_1\cdot i_2\ge 1. \end{cases}$

2.21. $f=\begin{cases} \sin\left(p\,n_1+e^{-n_2}\right), & \text{если } 1\le|n_1+n_2|\le p; \\ \cos\left(p\,n_2+e^{-n_1}\right), & \text{если } |n_1+n_2|<1. \end{cases}$

2.22. $f=\begin{cases} arctg\ \left(\ln\left(|f|+|g|\right)\right), & \text{если } 1\le f\cdot g<3; \\ arcctg\ \left(|f+g|\right), & \text{если } f\cdot g<1. \end{cases}$

2.23. $f=\begin{cases} \ln\left(|\cos^2(m\cdot x)|\right), & \text{если } |m+x|>2; \\ \ln\left(|\sin^2(m+x)|\right), & \text{если } 0.5<|m+x|\le 2. \end{cases}$

2.24. $f=\begin{cases} \sin\left(m\cdot\cos^2 x\right), & \text{если } 3\le m+x\le p; \\ \sin\left(m\cdot\cos x^2\right), & \text{если } m+x<3. \end{cases}$

2.25. $f=\begin{cases} -p + arcctg\ (|x-a|), \text{ если } 0 < x \le a+1; \\ \sqrt{a}\cdot \sin\ (a+e^{-x}), \text{ если } x > a+1. \end{cases}$

2.26. $f=\begin{cases} \frac{p}{x} + \ln\ (|arctg\ \ a|), \text{ если } \frac{a}{2} < x \le a; \\ p\,x - arctg\ (\ln\ |a|), \text{ если } x > a. \end{cases}$

2.27. $f=\begin{cases} \sin^2(p + ba^2), \text{ если } 1 \le a < b; \\ a^4 - e^{|a^2-b^2|}, \text{ если } a < 1. \end{cases}$

2.28. $f=\begin{cases} \lg\ (|p|+|n|), \text{ если } 1 < p \le n+0.5; \\ \lg\ (|p-n|), \text{ если } p > n+0.5. \end{cases}$

2.29. $f=\begin{cases} a^b + b^a, \text{ если } 0 < b < a; \\ \sqrt{|e^b + a|}, \text{ если } b < 0. \end{cases}$

2.30. $f=\begin{cases} \sin\ (\cos\ (\sqrt{a^2+b^2})), \text{ если } 1 < a\cdot b < p; \\ \cos\ (\sin\ (\lg\ (a^2+b^2))), \text{ если } a\cdot b \ge p. \end{cases}$

## ЗАДАНИЕ 3. ЦИКЛИЧЕСКИЕ ВЫЧИСЛИТЕЛЬНЫЕ ПРОЦЕССЫ

*Цель работы:* изучить приемы составления схем алгоритмов циклических вычислительных процессов, операторы For, Repeat, While.

В языке Паскаль имеются три вида операторов цикла:
1) While - оператор цикла с предварительным условием;
2) Repeat - оператор цикла с последующим условием;
3) For - оператор цикла с параметром.

### Оператор цикла While

Общий вид оператора:
While логическое выражение Do оператор;

Здесь оператор - простой или составной оператор.

*Работа оператора*. Предварительно проверяется значение логического выражения. Пока оно *истинно*, выполняется оператор тела цикла (после Do). Как только оно становится *ложным*, происходит выход из цикла. Если с самого начала значение логического выражения *ложно*, то оператор не выполняется ни

разу. Если логическое выражение никогда не принимает значение *ложно*, то происходит зацикливание.

*Пример*:

Для вычисления B в 9-й степени при B < 0, когда Exp(9*Ln(B)) недопустим, можно использовать следующий цикл:

```
  K:=1; P:=1;
While K <= 9 Do
   Begin
      P:=P*B;
      K:=K+1
   End;
```

*Пример:*

Составить программу вычисления и вывода на печать таблицы значений функции z=a*exp(b*x - c*x*x) при X <= X <= X с шагом dX. Здесь a=-105; b=-3,62e-2; c=1,1; X =2,65; X =5,55; dX=0,15.

```
Program Cikl_1;
  Var A,B,C: Real;
      X,Xn,Xk,dX : Real;
      Z: Real;
  Begin
     Read(A,B,C,Xn,Xk,dX);
     X:=Xn;
     While X <= Xk Do
        Begin
           Z:=A*Exp(B*X - C*X*X);
           Writeln(' X=',X:5:2,' ':5,'Z=',Z:10);
           X:=X+dX
        End
   End.
```

**Оператор цикла Repeat**

Общий вид оператора:

```
Repeat
   тело цикла
Until  логическое выражение;
```

*Работа оператора*. Выполняются операторы циклической части, проверяется значение логического выражения: если оно *ложно*, то вновь выполняются операторы циклической части; если же оно *истинно*, то цикл заканчивается. Если значение логического выражения *истинно с самого начала*, то операторы циклической части выполняются один раз. Если же логическое выражение *никогда не принимает значение «истинно»*, то

операторы тела цикла выполняются бесконечное число раз, т.е. происходит зацикливание.

Нижняя граница операторов тела цикла четко обозначается словом Until, поэтому нет необходимости заключать эти операторы в операторные скобки Begin- End. В то же время наличие дополнительных операторных скобок не является ошибкой.

*Пример:*

Вычислить B в 9-й степени при B<0 с использованием оператора цикла Repeat.

```
K:=1; P:=1;
Repeat
  P:=P*B;
  K:=K+1
Until K > 9;
```

*Пример*:

Составить программу для вычисления и вывода на печать таблицы значений функции y=sin(a*x)*sqrt(x) при x=5,6,...,25; a=15.27e-2.

```
Program Cikl_2;
  Var A,Y : Real;
      X : Byte;
 Begin
   A:=15.27e-2;
    X:=5;
   Repeat
     Y:=Sin(A*X) * Sqrt(X);
     WriteLn(' X=',x,'  Y=',y:10);
     X:=X+1
   Until X > 25
 End.
```

Операторы тела цикла будут выполняться до тех пор, пока X не станет больше 25.

**Оператор цикла For**

Общий вид оператора: For K:=N1 To N2 Do оператор;

Здесь K - переменная цикла;

N1,N2 - начальное и конечное значения переменной цикла.

В качестве переменной цикла K можно использовать только простую переменную, а в качестве N1, N2 могут использоваться выражения (за исключением вещественного типа Real).

Параметры K, N1, N2 должны быть одного и того же скалярного типа (целого, символьного, интервального типа или типа «перечисление»), но не Real. Переменная цикла K принимает последовательные значения данного типа от N1 до N2.

Если K, N1, N2 - данные целого типа, то шаг изменения переменной цикла всегда равен единице измерения.

*Пример*:

Вычислить B в 9-й степени при B<0 с помощью оператора For:

```
P:=1;
For K:=1 To 9 Do P:=P*B;
```

*Пример*:

Найти минимальное значение функции y=cos(a*x*x)+sin(b*x) для X=5,6,7,...,25;

a=105; b=-2.38.

```
Program Cikl_3;
   Const   A=0.105; B=-2.38;
   Var     Y,Ymin : Real;
           X      : Integer;
   Begin
     X:=5;
    Ymin:=Cos(A*X*X) + Sin(B*X);
    For X:=6 To 25 Do
      Begin
        Y:=Cos(A*X*X) + Sin(B*X);
        If Y < Ymin Then Ymin:=Y
      End;
    Write('Ymin=',Ymin:10)
   End.
```

Если необходимо записать цикл по убывающим значениям параметра K от N2 до N1, то используется оператор

```
For K:=N2 DownTo N1 Do оператор;
```

В этом случае параметр K принимает последовательно убывающие значения данного типа от N2 до N1.

Если K, N1, N2 - данные целого типа, то шаг изменения переменной цикла всегда равен -1.

*Пример*:

Возвести B в 9-ю степень при B<0, используя оператор цикла For с убывающим значением управляющей переменной.

```
P:=1;   For K:=9 DownTo 1 Do P:=P*B;
```

*Замечания:*

1. Внутри цикла нельзя изменять ни начальное, ни конечное значения (N1, N2) переменной цикла K, а также само значение K.

2. Если в возрастающем цикле N1 > N2, то цикл не выполняется ни разу. Аналогично - для убывающего цикла с DownTo, если N2 < N1.

3. После завершения цикла значение переменной цикла K становится неопределенным, за исключением выхода из цикла с помощью оператора GoTo.

4. Во всех трех операторах цикла (While, Repeat, For) среди операторов циклической части можно использовать операторы условного и безусловного перехода If, GoTo. Но нельзя передавать управление извне цикла внутрь цикла.

*Пример*: Составить программу вычисления произведения

$$V = \prod_{k=1}^{m} \frac{m^{(k/2)}}{2*(m-1)} \quad при \quad m = 8.$$

```
Program Cikl_4;
   Var K,M : Byte;
         V     : Real;
  Begin
       M:=8;
       V:=1.0;
       For K:=1 To M Do
          V:=V * Exp(K/2*Ln(M))/(2*(M-1));
      Write('V=',V:10)
    End.
```

*Пример:* Составить программу вычисления суммы

$$S = \sum_{i=1}^{n} \frac{\sin(a*i)}{i}, \quad где \quad a = 0.1; \quad n = 10.$$

```
Program Cikl_5;
   Uses Crt;
   Var K,N : Byte;
        A,S : Real;
  Begin
     ClrScr;
     Read(N,A);
     S:=0;
     For K:=1 To N Do S:=S + Sin(A*K) / K;
     Write('S=', S:10)
  End.
```

**Контрольные вопросы**

1. Какой вычислительный процесс называется циклическим ?
2. Как работает оператор For ?
3. Как работает оператор While ?
4. Как работает оператор Repeat ?

## Задача 3. Варианты

Составить программу вычисления таблицы значений суммы S(x) и функции Y(x) в произвольном диапазоне [a,b] изменения аргумента x с произвольным шагом h. Значения a,b, и h вводятся с клавиатуры.

3.1. $s(x)=\sum_{k=o}^{n}\frac{\ln^k 3}{k!}*x^k,\ \ y(x)=3^x;\ [-3;3].$

3.2. $s(x)=\sum_{k=0}^{n}\frac{k^2+1}{k!}*\left(\frac{x}{2}\right)^k,\ \ y(x)=\left(\frac{x^2}{4}+\frac{x}{2}+1\right)*e^{\frac{x}{2}};\ [-3;3].$

3.3. $s(x)=\sum_{k=0}^{n}(-1)^k*\frac{x^{2k+1}}{(2k+1)!},\ y(x)=\sin x;\ [-3;3].$

3.4. $s(x)=\sum_{k=1}^{n}(-1)^{k+1}*\frac{\sin kx}{k},\ \ y(x)=\frac{x}{2};\ [-1;1].$

3.5. $s(x)=\sum_{k=0}^{n}\frac{x^k}{k!},\ \ y(x)=e^x;\ [-3;\ 3].$

3.6. $s(x)=\sum_{k=0}^{n}\frac{\cos\frac{kp}{4}}{k!}*x^k,\ \ y(x)=e^{x\cos\frac{p}{4}}*\cos\left(x\cdot\sin\frac{p}{4}\right);\ [-3;\ 3].$

3.7. $s(x)=\sum_{k=0}^{n}(-1)^k*\frac{x^{2k}}{(2k)!},\ \ y(x)=\cos x;\ [-3;\ 3].$

3.8. $s(x)=\sum_{k=1}^{n}x^k*\sin\frac{kp}{4},\ \ y(x)=\frac{x\sin\frac{p}{4}}{1-2x\cos\frac{p}{4}+x^2};\ [0;\ 0{,}9].$

3.9. $s(x)=\sum_{k=1}^{n}(-1)^k*\frac{(2x)^{2k}}{(2k)!},\ \ y(x)=2\left(\cos^2 x-1\right);\ [-3;\ 3].$

3.10. $s(x)=\sum_{k=0}^{n}\frac{\cos kx}{k!},\ \ y(x)=e^{\cos x}*\cos(\sin x);\ [-3;\ 3].$

3.11. $s(x)=\sum_{k=0}^{n}\frac{2k+1}{k!}*x^{2k},\ \ y(x)=\left(1+2x^2\right)*e^{x^2};\ [-3;\ 3].$

3.12. $s(x)=\sum_{k=1}^{n}\frac{x^k*\cos\frac{kp}{3}}{k},\ \ y(x)=-\frac{1}{2}\ln\left(1-2x\cos\frac{p}{3}+x^2\right);\ [0;\ 0{,}9].$

3.13. $s(x)=\sum_{k=0}^{n}\frac{1}{2k+1}*\left(\frac{x-1}{x+1}\right)^{2k+1},\ \ y(x)=\frac{1}{2}\ln x;\ [0{,}1;\ 3\ ].$

3.14. $S(x)=\sum_{k=1}^{n}(-1)^k * \frac{\cos kx}{k^2}, \quad y(x)=\frac{1}{4}\left(x^2-\frac{p^2}{3}\right); \ [-3;\ 3].$

3.15. $s(x)=\sum_{k=1}^{n}(-1)^{k+1} * \frac{x^{2k+1}}{4k^2-1}, \quad y(x)=\frac{1+x^2}{2} * arctg\, x-\frac{x}{2}; \ [-1;\ 1].$

3.16. $s(x)=\sum_{k=1}^{n}\frac{\sin(2k-1)*x}{2k-1}, \quad y(x)=\frac{p}{4}; \ [0{,}1;\ 2{,}5].$

3.17. $s(x)=\sum_{k=0}^{n}\frac{x^{2k}}{(2k)!}, \quad y(x)=\frac{e^x+e^{-x}}{2}; \ [-3;\ 3].$

3.18. $s(x)=\sum_{k=0}^{n}\frac{\cos kx}{k!}, \quad y(x)=e^{\cos x} * \cos(\sin x); \ [-3;\ 3].$

3.19. $s(x)=\sum_{k=0}^{n}\frac{(2x)^k}{k!}, \quad y(x)=e^{2x}; \ [-3;\ 3].$

3.20. $s(x)=\sum_{k=0}^{n}\frac{k^2+1}{k!} * \left(\frac{x}{2}\right)^k, \quad y(x)=\left(\frac{x^2}{4}+\frac{x}{2}+1\right) * e^{\frac{x}{2}}; \ [-3;\ 3].$

3.21. $s(x)=\sum_{k=0}^{n}(-1)^k * \frac{x^{2k+1}}{2k+1}, \quad y(x)=arctg\, x; \ [-1;\ 1].$

3.22. $s(x)=\sum_{k=0}^{n}(-1)^k * \frac{2k^2+1}{(2k)!} * x^{2k}, \quad y(x)=\left(1-\frac{x^2}{2}\right)\cos x-\frac{x}{2}\sin x; \ [-3;\ 3].$

3.23. $s(x)=\sum_{k=1}^{n}(-1)^k * \frac{(2x)^{2k}}{(2k)!}, \quad y(x)=2*\left(\cos^2 x-1\right); \ [-3;\ 3].$

3.24. $s(x)=\sum_{k=1}^{n}(-1)^k * \frac{(1+x)^{2k}}{k}, \quad y(x)=\ln\frac{1}{2+2x+x^2}; \ [-2;\ -0{,}1].$

3.25. $s(x)=\sum_{k=0}^{n}\frac{x^{2k+1}}{(2k+1)!}, \quad y(x)=\frac{e^x-e^{-x}}{2}; \ [-3;\ 3].$

3.26. $s(x)=\sum_{k=1}^{n}\frac{k^2}{(2k+1)!} * x^k, \quad y(x)=\frac{1}{4} * \left(\frac{x+1}{\sqrt{x}} * sh\sqrt{x}-ch\sqrt{x}\right); \ [0{,}1;\ 10].$

3.27. $s(x)=\sum_{k=1}^{n}x^k * \cos\frac{kp}{4}, \quad y(x)=\frac{x*\cos\frac{p}{4}-x^2}{1-2x*\cos\frac{p}{4}+x^2}; \ [-0{,}9;\ 0{,}9].$

3.28. $s(x)=\sum_{k=1}^{n}k*(k+2)*x^k, \quad y(x)=\frac{x*(3-x)}{(1-x)^3}; \ [-0{,}8;\ 0{,}8].$

3.29. $s(x)=\sum_{k=1}^{n}\frac{\cos(2k-1)x}{(2k-1)^2}$, $y(x)=\frac{p^2}{8}-\frac{p}{4}*|x|$; $[0;\ 3]$.

3.30. $s(x)=\sum_{k=1}^{n}(-1)^{k+1}*\frac{x^{2k}}{2k(2k-1)}$, $y(x)=x*arctgx-\ln\sqrt{1+x^2}$; $[-1;\ 1]$.

## *ЗАДАНИЕ 4. МАССИВЫ*

*Цель работы:* изучить приемы составления программ с использованием массивов.

*Массив* - это упорядоченная совокупность конечного числа данных одного типа. Массивы относятся к сложным типам данных. Имена массивов образуются так же, как и имена простых переменных. Каждый элемент массива определяется своим индексом, по которому к нему осуществляется доступ.

Возможны два способа описания массивов:

Type имя типа = Array[t ,t ,...,t ] Of базовый тип элементов;

Var имя массива : имя типа;

или

Var имя массива:Array[t ,t ,...,t ] Of базовый тип элементов;

Здесь t , t ,...,t - типы индексов массива ( любой скалярный тип, кроме real ).

Количество индексов и определяет размерность массива. В качестве индексов массива могут быть использованы любые выражения скалярного типа, кроме Real.

*Пример*:

В программе необходимо описать двухмерный массив целых чисел:

4 0 7

2 1 5

Описание этого массива в соответствии с первым способом выглядит следующим образом:

```
Type mas = Array[1..2,1..3] Of Byte;
Var M : mas;
```

Для второго способа имеем:

```
Var M : Array[1..2,1..3] Of Byte;
```

Для обращения в программе к элементу во второй строке и третьем столбце используется запись M [2,3].

В качестве индексов массива могут использоваться выражения, частным случаем которых является константа или переменная, например:

```
Min := A[2*i+1,3];
Sum := Sum - C[k];
Rez := B[4] + 5.2;
```

*Пример* описания трехмерного массива:

```
Var A : Array[char,boolean,1..10 ] of Real;
```

В этом случае при использовании в программе элемента A [i,j,k] индексы i,j,k должны быть следующих типов: i- символьного, j- логического, k - интервального.

Для ввода и вывода числовых значений элементов массива используются циклы. Например, цикл

```
For k := 1 To 13 Do Read(C[k]);
```

организует ввод 13 значений элементов массива C, а цикл

```
For k := 1 To 13 Do Write(C[k],' ');
```

вывод этих элементов.

Начальные значения элементов массива могут быть введены сразу с описанием массива как типизированная константа в разделе Const. При этом могут применяться также две формы описания:

```
Type имя типа = Array[тип индекса] Of базовый тип элементов;
Const имя константы : имя типа = (список констант);
```

или

```
Const имя константы : Array[тип индекса] Of базовый тип элементов =
(список констант);
```

При описании типизированной константы типа "многомерный массив" константы каждой строки заключаются в отдельные скобки. Например, описание двухмерного массива со следующими начальными значениями:

```
 2   3  15   4
 7  18  23   5
 1   9  11  10
32  16   6   4
```

будет выглядеть следующим образом:

```
Const M : Array[1..4,1..4] Of Byte = ((  2,  3, 15,  4),
                                      (  7, 18, 23,  5),
                                      (  1,  9, 11, 10),
                                      (32, 16,  6,  4));
```

В языке Паскаль допускается использование массива в целом только в операторе присваивания вида:

```
M1 := M2;
```

где M1, M2 - два однотипных массива одинаковой размерности.

*Пример:* В квадратной матрице n-го порядка найти и вывести на экран строку, сумма элементов которой максимальна.

```
Program matrica;
  Const Nmax = 20;
  Var   A : Array [1..nmax, 1..nmax ] Of Real;
        i,j,N,Nstr : Byte;
        S,Smax : Real;
```

```
Begin
   Writeln ('Ввести порядок матрицы ( не более ',nmax,' )');
    Readln (n);
    For i:=1 To N Do
       Begin
          Writeln ('Введите элементы ',i,'-й строки ');
           For j:=1 To N Do Read (A[i,j]);
           S:=0;
           For j:=1 To N Do S:=S + A[i,j];
           If i=1 Then
                    Begin
                       Nstr:=1;
                       Smax:=S;
                    End
                Else if S > Smax Then
                                        Begin
                                          Nstr:=i;
                                          Smax:=s;
                                        End;
          End;
      Writeln ( 'Номер строки : ',nstr );
       For j:=1 To N Do Write (A[nstr,j]:10:3);
       Writeln;
  End.
```

**Контрольные вопросы**

1. Приведите определение массива.
2. Что такое размер и размерность массива?
3. Как описывается массив в Паскаль-программе?
4. Как описать типизированную константу типа «многомерный массив»?
5. Какой тип данных можно использовать в качестве индексов элементов массивов?

**Задача 4. Варианты**

4.1. Задан двухмерный массив целых чисел A размером N на N. Найти сумму элементов, расположенных на главной диагонали.

4.2. Задан двухмерный массив целых чисел A размером N на N. Найти произведение элементов, расположенных на главной диагонали.

4.3. Задан двухмерный массив целых чисел A размером N на M. Найти максимальный элемент и поменять его местами с элементом A[1,1].

4.4. Задан двухмерный массив целых чисел A размером N на M. Найти минимальный элемент и поменять его с элементом A[1,1].

4.5. Задан двухмерный массив целых чисел A размером N на M. Найти максимальный элемент и поменять его с элементом A[N,M].

4.6. Задан двухмерный массив целых чисел A размером N на M. Найти минимальный элемент и поменять его с элементом A[N,M].

4.7. Задан двухмерный массив целых чисел A размером N на M, состоящий из положительных и отрицательных чисел. Найти количество отрицательных и положительных элементов массива и подсчитать их сумму.

4.8. Задан двухмерный массив целых чисел A размером N на M, состоящий из нулей и единиц. Найти количество нулей и единиц в этом массиве.

4.9. Задан двухмерный массив целых чисел A размером N на M. Найти число элементов A[i,j] > T и сумму этих элементов.

4.10. Задан двухмерный массив целых чисел A размером N на M. Найти число элементов A[i,j] > T и произведение этих элементов.

4.11. Задан двухмерный массив целых чисел A размером N на M. Найти число элементов A[i,j] < T и сумму этих элементов.

4.12. Задан двухмерный массив целых чисел A размером N на M. Найти число элементов A[i,j] < T и произведение этих элементов.

4.13. Задан двухмерный массив целых чисел A размером N на M. Найти число элементов A[i,j] = T и сумму этих элементов.

4.14. Задан двухмерный массив целых чисел A размером N на M. Найти число элементов A[i,j] = T и произведение этих элементов.

4.15. Задан двухмерный массив целых чисел A размером N на M. Сформировать одномерный массив B[K], состоящий из отрицательных элементов массива A, и найти их сумму.

4.16. Задан двухмерный массив целых чисел A размером N на M. Сформировать одномерный массив B[K], состоящий из положительных элементов массива A, и найти их сумму.

4.17 Задан двухмерный массив целых чисел A размером N на M. Сформировать одномерный массив B[K], состоящий из элементов массива A, больших значения T, и найти их сумму.

4.18. Задан двухмерный массив целых чисел A размером N на M. Сформировать одномерный массив B[K], состоящий из элементов массива A, меньших значения T, и найти их сумму.

4.19. Задан двухмерный массив целых чисел A размером N на M. Сформировать одномерный массив B[K], состоящий из элементов массива A, равных значению T, и найти их сумму.

4.20. Задан двухмерный массив целых чисел A размером N на M, состоящий из нулей и единиц. Сформировать одномерный массив B[K], состоящий из нулей, и подсчитать их количество.

4.21. Задан двухмерный массив целых чисел A размером N на M, состоящий из нулей и единиц. Сформировать одномерный массив B[K], состоящий из единиц, и подсчитать их количество.

4.22. Задан двухмерный массив целых чисел A размером N на M. Сформировать одномерный массив B[K], состоящий из отрицательных элементов массива A, кратных 3, и найти их сумму.

4.23. Задан двухмерный массив целых чисел A размером N на M. Сформировать одномерный массив B[K], состоящий из положительных элементов массива A, кратных 3, и найти их сумму.

4.24. Задан двухмерный массив целых чисел A размером N на M. Сформировать одномерный массив B[K], состоящий из отрицательных элементов массива A в диапазоне от -3 до 0, и найти их сумму.

4.25. Задан двухмерный массив целых чисел A размером N на M, состоящий из положительных и отрицательных чисел. Найти количество отрицательных элементов этого массива и их сумму.

4.26. Задан двухмерный массив целых чисел A размером N на M, состоящий из положительных и отрицательных чисел. Найти количество положительных элементов этого массива и их сумму.

4.27. Задан двухмерный массив целых чисел A размером N на M, состоящий из положительных и отрицательных чисел. Найти количество отрицательных элементов этого массива и их произведение.

4.28. Задан двухмерный массив целых чисел A размером N на M, состоящий из положительных и отрицательных чисел. Найти количество положительных элементов этого массива, делящихся на 2, и их сумму.

4.29. Задан двухмерный массив целых чисел A размером N на M, состоящий из положительных и отрицательных чисел. Найти количество отрицательных элементов этого массива, делящихся на 2, и их сумму.

4.30. Задан двухмерный массив целых чисел A размером N на M, состоящий из положительных и отрицательных чисел. Найти количество положительных элементов этого массива, которые делятся на 2, и их произведение.

## ЗАДАНИЕ 5. ПРОЦЕДУРЫ И ФУНКЦИИ

*Цель работы:* изучить приемы составления программ с использованием подпрограмм ( локальных блоков ).

### Подпрограмма процедура

*Подпрограммой* называется именованная логически законченная группа операторов, которую можно многократно вызвать для выполнения по имени из различных мест программы.

Для организации подпрограмм используются процедуры и функции.

Описание процедуры имеет вид:

```
Procedure имя ( формальные параметры );
    раздел описаний
```

```
Begin
    раздел операторов
End;
```

и помещается в разделе описаний основной программы (глобальном блоке).

Здесь *имя* - имя процедуры. Раздел описаний, как и в основной программе, включает разделы Label, Const, Type, Var и раздел процедур и функций. Формальные параметры представляют собой список переменных с указанием их типа. Эти переменные не описываются в разделе описаний процедуры. Допускается использование процедур без формальных параметров.

Формальные параметры могут быть трех видов:

1) параметры-значения (входные параметры);
2) параметры-переменные (выходные параметры);
3) параметры процедурного типа.

Описание входных параметров имеет вид:

список переменных 1: тип 1; список переменных 2: тип 2;…

Описание выходных параметров соответственно:

Var список переменных 1: тип 1; Var список переменных 2: тип 2; ...

Вызов процедуры в основной программе производится оператором вида:

Имя процедуры (фактические параметры);

Здесь параметры представляют собой список фактических параметров, перечисленных через запятую (без указания их типа). Входными фактическими параметрами, т.е. теми, которые передаются в процедуру, могут быть константы, переменные и выражения. Выходными параметрами (которые получают значения из процедуры) могут быть только переменные.

Между формальными и фактическими параметрами должно быть соответствие по количеству параметров, порядку их следования и типу данных. Имена соответствующих параметров могут быть одинаковыми или разными.

*Пример:*

Составить программу для вычисления суммы квадратов натуральных чисел от 1 до n и оформить ее в виде процедуры.

```
Procedure Lux (N:Integer; Var Sum:Integer);
  Var i:Integer;
Begin
  Sum:=0;
  For i:=1  To  N   Do   Sum:=Sum + Sqr(i);
End;
```

Вызов процедуры в основной программе имеет вид

```
Lux ( 10,s );
```

Здесь s - переменная типа Integer.

При использовании в качестве параметров процедур данных сложного типа

( массивы, множества, записи ) в основной программе необходимо предварительно описать имя типа этих данных, которые потом указываются в списке формальных параметров процедуры.

*Пример:*

Перемножить две квадратные матрицы A и B. Результат занести в матрицу C. В основной программе описывается тип

```
Type Mat = Array [1..5, 1..5] Of Real;
```

и заголовок процедуры тогда может иметь вид

```
Procedure Umn (A,B:Mat; Var C:Mat);
```

**Подпрограмма-функция**

Подпрограмма-функция аналогична процедуре, но имеет следующие отличия.

1. Заголовок функции имеет вид

```
Function имя ( формальные параметры ): тип функции;
```

2. Функция имеет только один результат выполнения.
3. Результат обозначается именем функции, поэтому в разделе операторов функции обязательно должен присутствовать оператор присваивания, в левой части которого стоит имя этой функции.
4. Вызов функции в основной программе осуществляется непосредственно внутри выражения по ее имени с указанием фактических параметров.

*Пример:*

Оформить предыдущую задачу в виде функции:

```
Function Lux (N:Integer): Integer;
  Var s,i: Integer;
Begin
   S:=0;
   For  i:=1  To  N  Do   S:=S + Sqr(i);
   Lux:=S;
End;
```

Вызов функции в основной программе может иметь вид

```
W:=Lux (10);
```

Здесь W - переменная типа Integer.

*Замечание.* При использовании подпрограмм процедур и функций следует иметь в виду, что переменные, представленные в разделе описания основной программы (Program), действуют в разделе операторов основной программы и в любой ее подпрограмме. Эти переменные называются *глобальными*. Переменные, описанные в подпрограмме, действуют только в этой подпрограмме и в любой объявленной в ней процедуре и функции.

Такие переменные называются *локальными*. Они недоступны для операторов основной программы и других подпрограмм.

## Контрольные вопросы

1. Что называется подпрограммой?
2. Как оформляется процедура?
3. Как оформляется функция?
4. Как осуществляется вызов процедуры?
5. Как осуществляется вызов функции?

### *Задача 5. Варианты*

Составить программу, оформив вычисления в виде подпрограммы (процедуры или функции). В головной программе произвести ввод исходных данных, вызов подпрограммы и вывод результатов.

5.1. В вещественной квадратной матрице N-го порядка найти максимальный и минимальный элементы. Переставить строки, в которых они находятся. Если они находятся в одной строке, выдать об этом сообщение.

5.2. Задана одномерная матрица N-го порядка, содержащая целые числа, каждое из которых занимает 4 позиции. Подсчитать количество единиц в числах и их порядковые номера.

5.3. Дана вещественная матрица размером NxM. Переставляя ее строки и столбцы, добиться того, чтобы наибольший элемент (один из них) оказался в верхнем левом углу.

5.4. Дана вещественная матрица размером NxM. Упорядочить ее строки по возрастанию наибольших элементов в строках матрицы.

5.5. Задана квадратная матрица N-го порядка, состоящая из 0 и 1. Повернуть элементы матрицы на $90^0$ по часовой стрелке.

5.6. Задана одномерная матрица N-го порядка, содержащая нули и целые числа. Заменить нули полусуммой последующего и предыдущего чисел. Если нуль является первым или последним числом матрицы, то его соответственно заменить последующим или предыдущим числом.

5.7. Дана вещественная квадратная матрица N-го порядка, все элементы которой различны. Найти скалярное произведение строки, в которой находится наибольший элемент матрицы, и столбца с наименьшим элементом.

5.8. Определить, является ли заданная целочисленная квадратная матрица N-го порядка ортонормированной, т.е. такой, в которой скалярное произведение каждой пары различных строк равно 0, а скалярное произведение каждой строки на себя равно 1.

5.9. Определить, является ли заданная матрица N-го порядка магическим квадратом, т.е. такой, в которой сумма элементов во всех строках и столбцах одинакова.

5.10. Дана целочисленная матрица размером NxM. Найти сумму наименьших элементов ее нечетных строк и наибольших элементов ее четных строк.

5.11. Дана действительная квадратная матрица N-го порядка. Рассмотрим те элементы, которые расположены в строках, начинающихся с отрицательного элемента. Найти сумму тех из них, которые расположены ниже главной диагонали матрицы.

5.12. Дана вещественная квадратная матрица N-го порядка. Получить целочисленную квадратную матрицу, в которой элемент равен 1, если соответствующий ему элемент исходной матрицы больше элемента, расположенного на главной диагонали, и равен 0 в противном случае.

5.13. Дана квадратная целочисленная матрица N-го порядка. Упорядочить элементы в строках по возрастанию.

5.14. Дана действительная квадратная матрица N-го порядка. Рассмотрим те элементы, которые расположены в строках, начинающихся с отрицательного элемента. Найти сумму тех из них, которые расположены выше главной диагонали матрицы.

5.15. Дана действительная квадратная матрица N-го порядка. Рассмотрим те элементы, которые расположены в строках, начинающихся с отрицательного элемента. Найти сумму тех из них, которые расположены на главной диагонали матрицы.

5.16. Дана квадратная целочисленная матрица N-го порядка. Упорядочить элементы в строках по убыванию.

5.17. Дана квадратная матрица N-го порядка, содержащая целые числа, каждое из которых занимает 3 позиции. Найти сумму чисел, в среднем разряде которых содержится число 5 и которые расположены на побочной диагонали и ниже.

5.18. Дана квадратная целочисленная матрица N-го порядка. Упорядочить элементы в столбцах по возрастанию.

5.19. Дана квадратная целочисленная матрица N-го порядка. Найти минимальный элемент среди положительных и максимальный среди отрицательных и их координаты.

5.20.Задана матрица размером NxM. Получить массив B, присвоив его K-му элементу значение 0, если все элементы K-го столбца матрицы нулевые, и значение 1 в противном случае.

5.21. Дана квадратная целочисленная матрица N-го порядка. Найти суммы элементов, расположенных на линиях, параллельных главной диагонали матрицы и находящихся выше нее.

5.22. Задана матрица размером NxM. Получить массив B, присвоив его K-му элементу значение 1, если K-я строка матрицы симметрична, и значение 0 в противном случае.

5.23. Задана матрица размером NxM. Определить K - количество «особых» элементов матрицы, считая, что элемент «особый», если он больше суммы остальных элементов своего столбца.

5.24. Задана матрица размером NxM. Определить K - количество «особых» элементов матрицы, считая, что элемент «особый», если в его строке слева от него находятся элементы, меньшие его, а справа – большие.

5.25. Дана квадратная целочисленная матрица N-го порядка. Найти сумму элементов тех строк матрицы, у которых на главной диагонали расположены отрицательные элементы.

5.26. Задана символьная матрица размером NxM. Определить K – количество различных элементов матрицы (т.е. повторяющиеся элементы считать один раз).

5.27. Определить, является ли заданная квадратная матрица n-го порядка симметричной относительно побочной диагонали.

5.28. Определить, является ли заданная квадратная матрица n-го порядка симметричной относительно главной диагонали.

5.29. В матрице n-го порядка найти максимальный среди элементов, лежащих ниже побочной диагонали.

5.30. В матрице n-го порядка найти минимальный среди элементов, лежащих выше главной диагонали.

## ЗАДАНИЕ 6. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ СТРОКОВЫХ ДАННЫХ

*Цель работы*: изучить приемы работы с данными типа «строка», приобрести навыки использования строковых процедур и функций Паскаля.

*Строка* - это последовательность символов кодовой таблицы персональной ЭВМ. При использовании в выражениях строка-константа заключается в апострофы. Длина строки равняется количеству символов в этой строке и может изменяться от 0 до 255.

Для определения данных строкового типа используется идентификатор *string*, за которым следует заключенное в квадратные скобки значение

максимально допустимой длины строки данного типа. Если это значение не указывается, то по умолчанию длина строки равна 255 байт.

Переменную строкового типа можно задать через описание типа в разделе определения типов или непосредственно в разделе описания переменных.

Форма описания строковых данных:

Type имя типа=String [максимальная длина строки];
Var имя строковой переменной : имя типа;
или
Var имя строковой переменной : String [максимальная длина строки];

*Пример:*

```
Type str=String[30];
Var C1,C2:str;
    или
Var C3,C4:String[25];
```

Объем памяти в байтах, требуемый для размещения строки, равен ее максимальной длине плюс 1 байт, в котором запоминается длина данной строки.

Допустимо использование *типизированных констант строкового типа*, например:

```
Const S1:String[5]='БГУИР';
      S2:String[4]=#107#116#102#44; {ktf.}
      S3:String[10]='Да';
```

Знак # означает, что символ представляется его ASCII кодом.

Как и в массивах, к отдельным символам строки можно обратиться с помощью индексов в квадратных скобках: S1[2], S2[3]. При этом символ с нулевым индексом S1[0] содержит код, равный числу символов в строке S1.

Выражения, в которых операндами служат строковые данные, называются *строковыми*. Они состоят из строковых констант, переменных, строковых функций и знаков операций. Над строковыми данными допустимы операции присваивания, сцепления и отношения.

**Операция присваивания**

Общий вид:

Имя строковой переменной := строковое выражение;

*Пример*:

```
Var S1 : String[2];
    S2 : String[3];
      .  .  .
S1 := 'No';  S2 := 'Yes';
S1 := S2;
```

Если длина строкового выражения превышает максимальную длину строковой переменной, то все лишние символы справа отбрасываются. Так, значение S1 в приведенном примере станет равным ‘Ye’.

Отметим, что ввод и вывод значений строковых переменных с помощью операторов Read и Write осуществляются без заключения их в апострофы. Так, если в предыдущем примере вместо оператора S2:=’Yes’ мы воспользуемся оператором Read(S2), то на экране монитора необходимо набрать Yes, начиная с первой позиции.

### Операция сцепления

Применяется для сцепления нескольких строк в одну результирующую строку. Для обозначения операции сцепления используется знак **“+”** . Длина результирующей строки не должна превышать 255.

Например:

```
Program String_01;
Var C : Char;
    S1:String[3];
    S2:String[7];
Begin
    Read(S1,C);
    S2:=S1+C+’Yes’;
    Write(‘S2=’,S2);
End.
```

Пусть была введена строка No i. Тогда после выполнения этой программы получим результат: S2=No i Yes.

### Операции отношения

Над строковыми данными допустимы следующие операции отношения:

**=, <>, >, <, >=, <=**

Эти операции имеют приоритет более низкий, чем операция сцепления, т.е. вначале всегда выполняются все операции сцепления, если они присутствуют, и лишь потом реализуются операции отношения. Сравнение строк с помощью операций отношения производится слева направо до первого несовпадающего символа. Строка считается больше, если в ней первый несовпадающий символ имеет больший номер в стандартной кодовой таблице.

Результат выполнения операций отношения над строковыми операндами всегда относится к булевскому типу и принимает значение *True*, если выражение истинно, и *False*, если выражение ложно.

Строки считаются равными, если они полностью совпадают по текущей, а не по объявленной длине и содержат одни и те же символы.

**Встроенные стандартные процедуры для обработки строк**

Delete (Var S:String; Poz, L:Integer) видоизменяет строку S, стирая L символов, начиная с символа с номером Poz.

*Пример*:

```
Var S:String[10];
  .  .  .
  S:=’строка’;
  Delete(S,2,4);    {S=’са’}.
```

После стирания подстроки ее оставшиеся части как бы склеиваются. Если Poz = 0 или превышает длину строки S, то строка не изменится. Также не изменит длину строки значение L=0. При L больше остатка строки, будет удалена подстрока от Poz и до конца. Это используют для “подрезания” строк до заданной величины.

Insert (S1:String; Var S:String; Poz:Integer) вставляет подстроку S1 в строку S, начиная с позиции Poz.

*Пример*:

```
Var S1,S: String[40];
  .  .  .
  S:=’Начало-конец‘;
  Insert(‘середина-‘, S, 8);    { Имеем S=’Начало-середина-конец‘}.
```

Str (X [:Width [:dec]]; Var S:String) служит для преобразования числовых значений в строковые. Для целых значений можно задать только Width, для вещественных – либо оба поля (формат с фиксированной точкой), либо одно – Width (экспоненциальная форма). Если число имеет меньше знаков, чем дано в поле, то оно выравнивается по правому краю, пустое место заполняется пробелами. Можно задать Width отрицательным, в этом случае выравнивание происходит по левому краю, а излишек как бы стирается.

*Пример*:

```
Var S:String;
  .  .  .
  Str(4.53:8:2, S);          {S=’    4.53’}
  Str(4.53:-8:2, S);         {S=’4.53’}
  Str(4.53:8:0, S);          {S=’       5’}
  Str(1.234567:6:4, S);    {S=’1.2346’}.
```

Val (S:String; Var V; Var ErrCode:Integer) преобразует числовые значения, записанные в строке S, в числовую переменную V. Если преобразование возможно, то переменная ErrCode равна нулю, в противном случае она содержит номер символа в строке S, на котором процедура застопорилась. Тип V должен соответствовать содержимому строки S.

**Встроенные стандартные функции для обработки строк**

Length(S:String) : Byte возвращает текущую длину строки S. Результат имеет целочисленный тип.

*Пример*:

```
Var L:Byte;
  .  .  .
  L:=Length(S);
```

Concat(S1,S2,…,SN:String) : String выполняет слияние строк S1,S2,…,SN в том порядке, в каком они указаны в списке параметров.

*Пример*:

```
Var Ssum:String[50];
  .  .  .
  Ssum:=Concat(S1,S2,S3);
```

Если сумма длин строк в Concat превысит объявленную длину строки в левой части оператора присваивания, то излишек будет отсечен. Следует помнить, что вместо Concat можно пользоваться операцией сцепления. Например, Ssum:=S1+S2+S3;

Copy(S:String; Poz, L:Length) : String позволяет выделить из строки S последовательность из L символов, начиная с позиции Poz. Если Poz>Length(S), то функция вернет пустую строку, а если L больше, чем число символов от Poz до конца строки S, то вернется остаток строки S от Poz до конца.

*Пример*:

```
Var Ssum:String;
  .  .  .
  Ssum:=Copy('ABC***123', 4, 3);     {Ssum='***'}
  Ssum:=Copy('ABC', 4, 3);           {Ssum=' '}
  Ssum:=Copy('ABC***123', 4, 11);    {Ssum='***123']
```

Pos(S1, S:String) : Byte возвращает номер символа в строке S, с которого начинается включение в S подстроки S1. Если S не содержит в себе S1, то функция вернет 0. Недостатком функции Poz является то, что она возвращает ближайшую стартовую позицию S1 в S от начала строки, т.е. вызов

```
Var P:Byte;
  .  .  .
  P:=Poz('abc', 'Nom abcabcabcfcd');
```

завершит свою работу, вернув значение 5, хотя есть еще и 8, и 11.

UpCase(C:Ch) : Char преобразует строчную букву латинского алфавита в прописную, возвращая все остальные, в том числе и буквы русского алфавита, в исходном виде.

Pred(C:Char) : Char выдает предшествующий C символ.

Succ(C:Char) : Char выдает следующий за C символ.

Chr(X:Byte) : Char возвращает символ, код которого равен X.

Ord(C:Char) : Byte возвращает число, равное коду символа C. Учитывая, что текущая длина строки S содержится в S[0], она может быть определена следующим образом:

```
Var S:String;
   L:Byte;
   .  .  .
   Read(S);
   L:=Ord(S[0]);
```

*Пример*:

Дана последовательность из 45 символов. Подсчитать в этой последовательности количество вопросительных и восклицательных знаков.

```
Program String 02;
Type  St = String[45];
Var    A : st;
       i,K : Byte;
Begin
   Writeln('ВВЕДИТЕ СТРОКУ СИМВОЛОВ:');
   Readln(a);
   K:=0;
For i:=1 to Length(a) Do
   if (a[i]='!') Or  (a[i]='?') Then K:=K+1;
   Writeln('КОЛИЧЕСТВО СИМВОЛОВ ! и ? В СТРОКЕ= ',k);
End.
```

## Контрольные вопросы

1. Как описываются строковые данные?
2. Чему равна максимальная длина строковой переменной?
3. Какие операции допустимы над строковыми данными, их приоритет, какие выражения называются строковыми?
4. Какие стандартные процедуры и функции существуют в Паскале для работы со строковыми данными?

## Задача 6. Варианты

6.1. Для каждого символа заданного текста указать, сколько раз он встречается в тексте. Сообщение об одном символе должно печататься не более одного раза.

6.2. Для встречающихся в заданном тексте пар рядом расположенных символов указать, сколько раз встречается каждое из таких двухбуквенных сочетаний.

6.3. Отредактировать предложение, удаляя из него лишние пробелы, оставляя по одному пробелу между словами.

6.4. В заданном предложении указать слово, в котором доля гласных (A,E,I,O) максимальна.

6.5. Проверить, имеется ли в заданном тексте баланс открывающих и закрывающих скобок, имея в виду, что балансом, например, будет комбинация ((…)), в то время как комбинация )..(..)..( балансом не является.

6.6. Для каждого слова заданного предложения указать долю согласных. Определить слово, в котором доля согласных максимальна.

6.7. Найти самое длинное симметричное слово заданного предложения, например АККА, и указать номер позиции, с которого оно начинается.

6.8. Отредактировать заданное предложение, заменяя многоточия точкой, а вместо точки ставить восклицательный знак.

6.9. В заданном предложении найти самое короткое и самое длинное слова и указать позиции, с которых они начинаются.

6.10. Из заданного текста предложения выбрать и напечатать только те символы, которые встречаются в нем только один раз (в том порядке, в котором они встречаются в тексте).

6.11. В заданном тексте заменить последовательность символов X(i) на A[k] и подсчитать число произведенных замен.

6.12. Задана строка символов, состоящая из букв, цифр, запятых, точек, знаков **“+”** и **“-“**, открывающей и закрывающей скобок. Выделить подстроку, состоящую из цифр, соответствующих целому числу (т.е. начинается со знака “+” или “–“ и внутри подстроки нет букв и точки).

6.13. Задана строка символов, состоящая из букв, цифр, запятых, точек, знаков **“+”** и **“-“**, открывающей и закрывающей скобок. Выделить подстроку, соответствующую вещественному числу с фиксированной точкой (т.е. ту, которая начинается с символа “+” или “–“ и внутри которой есть десятичная точка).

6.14. Задана строка символов, состоящая из букв, цифр, запятых, точек, знаков **“+”** и **“-“**, открывающей и закрывающей скобок. Выделить подстроку, соответствующую вещественному числу с плавающей точкой (т.е. в виде –0.00E-00).

6.15. Удалить из заданного текста символы “пробелы” и подсчитать количество удаленных символов и длину сформированного текста.

6.16. В тексте предложения заменить символ “пробел” на символ “запятая”. Конечные символы удалить, не заменяя на запятые. Если в тексте встречается несколько символов “пробел” подряд, то вместо них поставить одну запятую. Определить длину исходного и преобразованного предложений.

6.17. В заданном тексте предложения расставить слова по алфавиту в соответствии с его первой буквой.

6.18. В заданном тексте предложения заменить строчные буквы на прописные и подсчитать количество произведенных замен.

6.19. Задан текст, состоящий из произвольной последовательности буквенных символов. Упорядочить их в алфавитном порядке, при этом повторяющиеся символы должны быть удалены.

6.20. В заданном предложении поменять порядок следования слов на обратный.

6.21. Ввести текст, содержащий от 1 до 4 цифровых символов, отображающих целые числа от 1 до 2000. На печать вывести введенные символы и их представление в римской системе счисления.

6.22. В заданном предложении, состоящем из нескольких слов, поменять местами второе и четвертое слово.

6.23. В заданном тексте в каждом нечетном слове поменять местами первые два буквенных символа и заменить их на заглавные.

6.24. В заданном тексте в каждом четном слове заменить все строчные буквенные символы на прописные, а каждое нечетное слово заключить в круглые скобки.

6.25. В заданном тексте подсчитать количество четырехбуквенных слов и каждое четное из них заменить на сочетание “SsSs”.

6.26. Дана строка символов, состоящая из произвольного текста на английском языке, слова отделены пробелами. После каждого согласного символа вставить сочетание “Ff”.

6.27. Дана строка символов, состоящая из произвольного текста на английском языке, слова отделены пробелами. Перед каждым символом, обозначающим гласный, вставить сочетание “kK”.

6.28. Дана строка символов, состоящая из произвольного текста на английском языке, слова отделены пробелами. Каждую букву заменить на следующую в алфавите, при этом “z” меняется на“a”.

6.29. Дана строка символов, состоящая из произвольного текста на английском языке, слова отделены пробелами. Каждая согласная буква заменяется на следующую в алфавите (при этом “z” меняется на “a”), а каждая гласная – на предыдущую в алфавите (при этом “a” меняется на “z”).

6.30. Дана строка символов, состоящая из произвольного текста на английском языке, слова отделены пробелами. Каждую гласную букву поменять местами со следующей за ней. Если за ней следует гласная или символ “пробел”, то ее поменять местами с предыдущей.

## ЗАДАНИЕ 7. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ПЕРЕМЕННЫХ ТИПА “ЗАПИСЬ”

*Цель работы*: изучить приемы описания и работы с данными типа «запись», приобрести навыки программирования с использованием переменных типа «запись».

При решении ряда задач возникает необходимость объединения данных различных типов, которые описывают некоторые особенности какого-либо одного объекта, в одну группу. Например, при создании программы, обрабатывающей данные о студентах, необходимо объединить такие данные, как номер группы, фамилия, имя и отчество, год рождения, номер паспорта, домашний адрес и т.д. в один блок.

В языке Паскаль для группирования логически связанных данных различных типов в один блок используются записи. Запись имеет структурированный тип. *Запись* - это структура данных, состоящая из фиксированного числа элементов, которые называют *полями*. В отличие от массива поля могут иметь разные типы. Чтобы можно было ссылаться на тот или иной элемент записи, поля именуются.

Форма описания записи :

```
type имя типа = record
                  список идентификаторов полей1 : тип полей1;
                  список идентификаторов полей2 : тип полей2;
                                  . . .
                  список идентификаторов полейN : тип полейN;
                end;
var имя записи : имя типа;
```

или:

```
var имя записи : record
                  список идентификаторов полей1 : тип полей1;
                  список идентификаторов полей2 : тип полей2;
                                  . . .
                  список идентификаторов полейN : тип полейN;
                end;
```

*Пример* записи, объединяющей данные об одном студенте:

```
type student = record
                 nom      : string[8];      {номер группы}
                 fio      : string[25];     {фамилия,имя,отчество}
                 year     : word;           {год рождения}
                 pasport  : string[12];     {серия и номер паспорта}
                 address  : string[30];     {домашний адрес}
               end;
var   stud, bun : student;
```

Обращение к элементам записи производится с помощью составного имени. Составное имя имеет следующий вид :

Имя записи . имя поля

Например, для обращения к паспортным данным и адресу студента нужно воспользоваться следующими составными именами:

stud.pasport и stud.adress

Элементы записи используются в программе так же, как и обычные переменные. *Например*:

Stud.address := ’г.Минск, ул. Скорины 8, кв. 36.’;

Обращение к записи как к единому блоку допускается только в операторе присваивания. *Например*:

Bun := stud;

Запись может быть элементом других структурированных данных. Чаще всего записи используют в качестве элементов массивов. Так, список группы, состоящий из 20 студентов, может быть оформлен как массив следующим образом:

```
type student = record
                 nom      : string[8];    {номер группы}
                 fio      : string[25];   {фамилия, имя, отчество}
                 year     : word;         {год рождения}
                 pasport  : string[12];   {серия и номер паспорта}
                 address  : string[30];   {домашний адрес}
               end;
var  group : array[1..20] of student;
```

Для обращения, например, к полю fio 10-го элемента этого массива, т.е. к ФИО студента, имеющему 10-й порядковый номер, нужно использовать следующее составное имя:

group[10].fio

Обращение к полям записи по составному имени имеет достаточно громоздкий вид. Поэтому для сокращения записи составных имен в языке Паскаль предусмотрен оператор WITH, который имеет следующий формат:

With имя записи Do оператор;

где имя записи - имя переменной типа «запись», а оператор - простой или составной оператор Паскаля, в котором имена полей указываются как обычные переменные.

*Пример*:

Ввод данных о 20-м студенте массива GROUP может быть выполнен следующим образом:

```
with group[20] do
```

```
begin
  nom := ’200101’;
  fio := 'Иванов П.А.';
  year := 1972;
  pasport := 'IX-BB 520221';
  address := 'Минск, Богдановича 12-235';
end;
```

Паскаль допускает *вложение записей* друг в друга, т.е. поле в записи также может быть записью.

Например, поле fio в данных о студенте может быть разбито на три элемента: f - фамилия , i - имя, o - отчество студента. В этом случае блок описания будет иметь следующий вид:

```
type fff = record
   f : string[15];
   i : string[15];
   o : string[15]
end;
type student = record
                  nom      : string[8];    {номер группы}
                  fio      : fff;          {фамилия, имя, отчество}
                                           { вложенная запись}
                  year     : word;         {год рождения}
                  pasport  : string[12];   {серия и номер паспорта}
                  address  : string[30];   {домашний адрес}
                end;
var   group : array[1..20] of student;
```

В этом случае появляется возможность для работы, например с полем "f" (фамилия студента) или с полем "o" (отчество студента) через их составные имена:

group[20].fio.f  и соответственно group[20].fio.o

Оператор with для работы с этими полями также будет вложенным, например:

```
with group[20]  do
   with fio  do
     begin
      f := 'Иванов';
      i := 'Петр';
      o := 'Алексеевич'
     end;
```

*Пример*: Пусть имеется ведомость студентов одной группы с результатами сдачи экзамена по программированию:

| N студ. | Ф.И.О. | Оценка |
|---|---|---|
| 1 | Петров В.В. | 5 |
| 2 | Иванов С.И. | 4 |
| 3 | Чемоданов М.Т. | 3. |

Написать программу, которая определит количество студентов, сдавших экзамен на 4 и 5, и выведет их фамилии.

Программа будет выглядеть следующим образом:

```
Program zad7;
        {Блок описания данных}
type v = record
             nom : integer;
             fio    : string[25];
             ball   : integer
           end;
var
  spisok       : array[1..25] of  v;
  i, n, kol_vo : integer;
        {Начало программы}
begin
        {Ввод данных и подсчет студентов}
   kol_vo := 0;
   write('введите кол-во студ. в группе: ');
   readln(n);
   for i := 1 to n do
     begin
       writeln('вводите данные след. студента');
       with spisok[i] do
          begin
             write('номер в списке:');    readln(nom);
             write('Ф.И.О.:');            readln(fio);
             write('оценка:');            readln(ball);
             if ball >= 4 then  kol_vo := kol_vo + 1;
          end;
      end;
        {Вывод данных}
   writeln('кол-во студентов, сдавших экзамен на 4 и 5: ', kol_vo);
   writeln(' Ф.И.О. этих студентов:');
   for i := 1 to n do
     begin
        with spisok[i] do
        if ball >= 4 then writeln(fio);
```

```
        end;
    end.
```

**Контрольные вопросы**

1. Как описать переменную типа «запись»?
2. В каких случаях используются переменные типа «запись»?
3. Что такое составное имя?
4. Как можно обратиться к полю записи?
5. Каково назначение оператора with?

**Задача 7. Варианты**

*Используя записи, написать программу, которая:*

7.1 - проверяла бы, "бьет" ли карта К1 карту К2, с учетом того, что масть КМ является козырной.

```
Дано: type mast = (pica,tref,bub,cher);
          dost = (six,seven,eigth,nine,ten,valet,dama,korol,tuz);
          cart = record
                    m : mast;
                    g  : dost
                  end;
```

7.2 - печатала бы название самой высокой вершины из списка.

```
Дано:  type ver = record
                    name : string[12];
                    hight : 1000..9999
                  end;
           list = array [1..30] of ver;
```

*Используя записи, написать программу, которая формирует расписание полетов самолетов. Расписание включает в себя номер поезда, пункт назначения, время вылета, тип самолета. Программа выводит:*

7.3 - номера рейсов, время вылета, типы самолетов всех рейсов на вводимый с клавиатуры пункт назначения;

7.4 - пункт назначения, время вылета, тип самолета на вводимый с клавиатуры номер рейса;

7.5 - номера рейсов, пункты назначения, время вылета введенного с клавиатуры типа самолета.

*Используя записи, написать программу, которая заполняет анкеты студентов. Анкета включает в себя ФИО, возраст, пол, номер группы и оценки по четырем предметам. Программа :*

7.6 - выводит ФИО самого молодого студента

7.7 - выводит ФИО студента, который имеет максимальный балл по четырем предметам;

7.8 - выводит ФИО студентов женского пола;

7.9 - выводит ФИО студентов мужского пола;
7.10 - выводит средний балл студентов женского пола;
7.11 - выводит средний балл студентов мужского пола;
7.12 - выводит средний возраст студентов женского пола;
7.13 - выводит средний возраст студентов мужского пола.

*Написать программу, которая :*

7.14 - проверяет, правильно ли выставлены кости домино в ряду, т.е. равна ли правая цифра очередной кости левой цифре следующей кости.

```
Дано:  type cost_dom = record
                          left, right : 0..6
                        end;
       list = array[1..28] of cost_dom;
```

7.15 - упорядочивает записи таблицы T по возрастанию их ключей.

```
Дано:  const N = 30;
       type zap = record
                    key : integer;
                    body : array [1..9] of  'a'..'z'
                  end;
            tab = array[1..N] of zap;
```

*Используя записи, написать программу, которая формирует расписание движения поездов. Расписание включает в себя номер поезда, пункт назначения, время отправления, тип поезда ( пассажирский, скорый ), наличие вагона-ресторана. Программа выводит:*

7.16 - номера поездов, время отправления, тип поезда для всех маршрутов до вводимого с клавиатуры пункта назначения;

7.17 - пункт назначения, время отправления, тип поезда для вводимого с клавиатуры номера поезда;

7.18 - номера поездов, пункты назначения, время отправления введенного с клавиатуры типа поезда;

7.19 - номера поездов, пункты назначения, время отправления всех скорых поездов;

7.20 - номера поездов, пункты назначения, время отправления всех пассажирских поездов;

7.21 - номера поездов, пункты назначения, время отправления, тип поезда для всех поездов, имеющих вагоны-рестораны;

7.22 - номера поездов, пункты назначения, время отправления, тип поезда для всех поездов, не имеющих вагоны-рестораны.

*Используя записи, написать программу, которая формирует расписание междугородных автобусов. Расписание включает в себя номер рейса, пункт назначения, время отправления, тип автобуса, количество мест. Программа выводит:*

7.23 - номера рейсов, время отправления, тип автобуса для всех рейсов до вводимого с клавиатуры пункта назначения;

7.24 - пункт назначения, время отправления, тип автобуса для вводимого с клавиатуры номера рейса;

7.25 - номера рейсов, пункты назначения, время отправления введенного с клавиатуры типа автобуса;

7.26 - номера рейсов, пункты назначения, время отправления всех автобусов, имеющих более 20 мест.

*Используя записи, написать программу, которая формирует библиотечный каталог. Каталог включает в себя фамилию автора, название книги, издательство, год, стоимость книги. Программа выводит:*

7.27 - названия книг, издательство, год, стоимость всех книг вводимого с клавиатуры автора;

7.28 - автора книги, издательство, год, стоимость вводимого с клавиатуры названия книги;

7.29 - авторов и названия книг, издательство, стоимость всех книг, выпущенных в введенном с клавиатуры году;

7.30 - авторов и названия книг, издательство, год выпуска всех книг стоимостью более 1 тысячи рублей.

## ЛИТЕРАТУРА


1. Немнюгин С. Turbo Pascal. Программирование на языке высокого уровня.
-СПб: Питер, 2003. 544 с.
2. Немнюгин С. Turbo Pascal. Практикум. -СПб: Питер, 2003. 272 с.
3. Фаронов В. Turbo Pascal в подлиннике. –Киев: BHV, 2003. 1054 с.
4. Марченко А., Марченко. Л. Программирование в среде Turbo Pascal 7.0. -М.: Век, 2003. 464 с.
5. Сухарев М. Turbo Pascal 7.0. Теория и практика программирования. –М.: Наука и техника, 2003. 576 с.
6. Федоренко Ю. Алгоритмы и программы на Turbo Pascal. Учебный курс. -СПб: Питер, 2001. 240 с.
7. Ускова О. Программирование на языке Паскаль. Задачник. -СПб: Питер, 2002. 336 с.
8. Васильев П. Турбо Паскаль в примерах и задачах. Освой самостоятельно.
-М.: Финансы и статистика, 2002. 496 с.
9. Павловская Т. Паскаль. Программирование на языке высокого уровня. -СПб: Питер, 2003. 400 с.
10. Меженный О. Turbo Pascal. Самоучитель. -М.: Вильямс, 2003. 336 с.
11. Зеленяк О. Практикум программирования на Turbo Pascal. Задачи, алгоритмы и решения. -М.: ДиаСофт, 2002. 320 с.