Методические рекомендации по дисциплине

«Программирование»

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Армавирская государственная педагогическая академия

Кафедра информатики и ИТО

Бельченко В.Е., [illegible] Е.С., [illegible] Г.Ф.

ПРАКТИКУМ
по программированию на языке Паскаль

Учебно-методическое пособие

Армавир 2014

Печатается по решению редакционно-издательского совета Армавирской государственной педагогической академии

Бельченко В.Е., Белодед Е.С., Козырева Г.Ф.
Практикум по программированию на языке Паскаль: Учебно-методическое пособие. – Армавир: РИО АГПА, 2014, 70 с.

Пособие предназначено для приобретения студентами практических навыков алгоритмизации и программирования. В пособии дана общая характеристика языка программирования Паскаль, приводятся необходимые теоретические сведения о базовых конструкциях Паскаля, рассматривается реализация стандартных алгоритмов обработки данных на Паскале. Все это позволяет рекомендовать его при изучении курсов, включающих основы программирования на языке Паскаль.

Пособие может быть использовано студентами как очной, так и заочной формы обучения для организации самостоятельной работы. В пособии дан список литературы для дополнительного изучения языка программирования Паскаль.

Рецензент:

СОДЕРЖАНИЕ

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

Для выполнения лабораторных работ необходим персональный компьютер с установленной средой программирования Turbo Pascal 7.0 или PascalABC.

Лабораторная работа предусматривает реализацию полученных студентами знаний через организацию учебной работы в среде программирования Pascal по реализации, отладке и тестированию программ на ЭВМ.

Перед выполнением лабораторной работы необходимо изучить лекционный материал по данной теме, учебные материалы лабораторного практикума, ответить на контрольные вопросы. Особенно внимательно следует рассмотреть приведенные примеры составления программ, изучить алгоритм решения задачи. Рекомендуется выполнить все примеры программ, рассмотренные в пособии по данной теме.

По каждой теме студент должен получить у преподавателя индивидуальное задание в соответствии со своим вариантом, разработать программу и реализовать ее в среде программирования. Каждая программа должна быть протестирована при различных исходных данных. Исходные данные должны подбираться таким образом, чтобы при работе с ними программа прошла все основные пути алгоритма, поскольку на каждом из путей могут быть свои ошибки. Необходимо проанализировать полученные результаты, убедиться в их правильности результатов и соответствия условию задачи.

Во время выполнения заданий в учебной аудитории студент может консультироваться с преподавателем, определять наиболее эффективные методы решения поставленных задач. Если какая-то часть задания остается невыполненной, студент может продолжить её выполнение во время внеаудиторной самостоятельной работы.

Каждая лабораторная работа должна быть оформлена и защищена. Защита производится перед выполнением очередной лабораторной работы.

Отчет оформляется в рукописном или печатном виде и должен содержать:

- тему лабораторной работы
- условие задачи
- текст программы с указанием исходных данных и результата.
- результаты тестирования программы.

ВАРИАНТЫ ЗАДАНИЙ

№ вари-анта	Лабораторные работы																	
	I			II			III			IV		V		VI		VII		
1	1	6	13	1а	7	19	1	11	19	1	11	1	11	1	16	1	16
2	2	7	14	1б	2	20	2	12	20	2	12	2	12	2	17	2	17
3	3	8	15	1в	3	21	3	13	21	3	13	3	13	3	18	3	18
4	4	9	16	1г	4	22	4	14	22	4	14	4	14	4	19	4	19
5	5	10	17	2	8	23	5	15	23	5	15	5	15	5	20	5	20
6	1	11	18	3	9	24	6	16	24	6	16	6	16	6	21	6	21
7	2	12	19	4	10	25	7	17	25	7	17	7	17	7	22	7	22
8	3	6	20	5	11	26	8	18	26	8	18	8	18	8	23	8	23
9	4	7	21	6	12	27	9	19	27	9	19	9	19	9	24	9	24
10	5	8	22	7	13	28	10	13	28	10	20	10	20	10	25	10	25
11	1	9	23	2	14	29	1	14	29	1	16	1	16	11	26	11	26
12	2	10	24	3	15	30	2	15	30	2	17	2	17	12	27	12	27
13	3	11	25	4	16	22	3	16	19	3	18	3	18	13	28	13	28
14	4	12	13	5	17	23	4	17	20	4	19	4	19	14	29	14	29
15	5	6	14	6	18	24	5	18	1	5	20	5	20	15	30	15	30

ОБЩИЕ СВЕДЕНИЯ О ЯЗЫКЕ ПРОГРАММИРОВАНИЯ ПАСКАЛЬ

Информацию в компьютере обрабатывает процессор, следовательно, алгоритм должен быть записан на языке, понятном для процессора, т.е. на машинном языке, представляющем последовательности нулей и единиц.

На заре компьютерной эры, в 50-е годы 20 в., программы писались на машинном языке и представляли собой очень длинные последовательности нулей и единиц. Составление и отладка таких программ было чрезвычайно трудоемким делом.

В 60-70-е г. для облегчения труда программистов начали создаваться языки программирования высокого уровня, формальные языки, кодирующие алгоритмы в привычном для человека виде (в виде предложений). Такие языки строились на основе использования определенного алфавита и строгих правил построения предложений (синтаксиса).

Наиболее широко распространенным типом языков программирования высокого уровня являются процедурные языки. В таких языках широко используются управляющие конструкции (операторы), которые позволяют закодировать различные алгоритмические структуры (линейную, ветвление, цикл).

Одним из первых процедурных языков программирования был Basic, созданный в 1964 г. Другим широко распространенным языком программирования алгоритмического типа является Паскаль.

Язык программирования Паскаль (назван в честь выдающегося французского математика и философа Блеза Паскаля (1623 — 1662)), разработан в 1968 — 1971 гг. Н.Виртом. Язык Паскаль, созданный первоначально для обучения программированию как систематической дисциплине, скоро стал широко использоваться для разработки программных средств в профессиональном программировании.

Благодаря своей компактности, удачному первоначальному описанию Паскаль оказался достаточно легким для изучения. Язык программирования Паскаль отражает фундаментальные и наиболее важные концепции (идеи) алгоритмов в очевидной и легко воспринимаемой форме, что предоставляет программисту средства, помогающие проектировать программы. Язык Паскаль позволяет четко реализовать идеи *структурного программирования* и *структурной организации данных*. Язык Паскаль сыграл большую роль в развитии методов аналитического доказательства правильности программ и позволил реально перейти от методов отладки программ к системам автоматической проверки правильности программ.

Применение языка Паскаль значительно подняло "планку" надежности разрабатываемых программ за счет требований Паскаля к описанию используемых в программе переменных, проверки согласованности программы при компиляции без ее выполнения.

Основными элементами языка Паскаль являются:

1) латинские буквы a, b, c …, z, A, B, C … Z (регистр не учитывается);
2) цифры 0 … 9;
3) спец. Символы *, - , (,), <, > и.т.д.;
4) служебные слова begin, end, if …;
5) идентификаторы – имена переменных, констант, процедур, функций.

Имена должны начинаться с латинской буквы, за которой могут следовать буквы и цифры (A, A23, Sum, MyProgram).

Любая программа на Паскале представляет собой текстовый файл с собственным именем и расширением pas (Primer.pas, Summa.pas).

Программа на языке Паскаль состоит из заголовка, разделов описаний и раздела операторов.

Заголовок (Program …)

Раздел описаний

a) описание внешних модулей (Uses)
b) описание типов (Type)
c) описание констант (Const)
d) описание переменных (Var)
e) описание меток (Label)
f) описание процедур и функций (Procedure… Function …)

Раздел операторов

Begin

…

End.

Основные типы данных в Паскале:

Название типа	Обозначение	Описание	Примеры значений
Целый	INTEGER	целые числа из диапазона 32768...32767	125, -678, 0
целый	BYTE	Целые числа от 0 до 255	0, 42, 255
целый	LONGINT	Целые числа из диапазона 2147483648... 2147483647	0, 2345, -3456, 656435
вещественный	REAL	Действительные числа из диапазона $2.9*10^{-39}$ ... $1.7*10^{+38}$	-0.123, 23.65, 18, 3.287654E-01,
логический	BOOLEAN	Логические значения true(истина) или false(ложь)	true, false
символьный	CHAR	Символы	‘y’, ‘5’,’*’
строковый	STRING	строка, содержащая от 0 до 255 символов	‘PASCAL’, ’a+b’

Стандартные операции и функции

Функция, операция	Назначение	Тип переменных	Тип результата операции, функции
A+B	Сумма	REAL, INTEGER	REAL, INTEGER
A-B	Разность	REAL, INTEGER	REAL, INTEGER

A*B	Произведение	REAL, INTEGER	REAL, INTEGER
A/B	Частное	REAL, INTEGER	REAL
ABS(x)	Абсолютное значение	REAL, INTEGER	REAL, INTEGER
SQR(x)	Вычисление x^2	REAL, INTEGER	REAL,INTEGER
sin(x)	Нахождение sin x	REAL,INTEGER	REAL
cos (x)	Вычисление cos x	REAL, INTEGER	REAL
Arctan(x)	Вычисление arctan x	REAL, INTEGER	REAL
Exp(x)	Вычисление экспоненты e^x	REAL, INTEGER	REAL
Ln(x)	Вычисление ln x	REAL, INTEGER	REAL
SQRT(x)	Вычисление $\sqrt{x}$	REAL, INTEGER	REAL
A div B	Нахождение целой части при делении A на B	INTEGER	INTEGER
A mod B	Нахождение остатка при делении A на B	INTEGER	INTEGER
TRUNC(x)	Нахождение целой части x	REAL, INTEGER	INTEGER
ROUND(x)	Округление x в сторону ближайшего целого	REAL, INTEGER	INTEGER
PI	Зарезервированная константа число π		REAL

Примеры записи математических выражений

$\frac{a^2+\sqrt{x}}{\sin x+\cos x}$ = (sqr(a)+sqrt(x))/(siin(x)+cos(x))

$\frac{e^x+e^{-x}}{2}$ = (exp(x)+exp(-x))/2

$\frac{tgx+ctgx}{\ln x}$ = (sin(x)/cos(x)+cos(x)/sin(x))/ln(x)

ЛАБОРАТОРНЫЕ РАБОТЫ

1. КОМАНДЫ ПРИСВАИВАНИЯ, ВВОДА И ВЫВОДА

Краткие теоретические сведения

При выполнении программы происходит обработка данных. Данные в программировании называют *величинами*. Величины, значения которых могут изменяться в процессе выполнения программы, называют *переменными***.** Значение каждой переменной хранится в определенном участке памяти компьютера.

Каждая переменная характеризуется именем, типом и значением.

Имя переменной (идентификатор) всегда должно начинаться с латинской буквы, после которой могут следовать несколько латинских букв, цифр либо символ подчеркивания «_», записанных без пробелов. Например: A, B1, sum, Name, Pr_3.

Тип переменной определяет диапазон допустимых значений.

Все переменные, используемые в программе, должны быть описаны в разделе описаний. Например:

```
Var
 A, B: integer;
 X:real;
 Text: string;
```

Переменная не имеет какого-либо конкретного значения до тех пор, пока компьютеру не будет дано точное предписание, поместить что-либо определенное в соответствующую ячейку памяти.

На Паскале такого рода предписание обычно выражается *командой присваивания*, имеющей вид:

имя_переменной:=выражение_или_значение

Оператор присваивания **(:=)** предписывает выполнить выражение, заданное в его правой части, и присвоить результат переменной, идентификатор которой расположен в левой части. Переменная и выражение должны быть совместимы по типу.

Оператор присваивания выполняется следующим образом: сначала вычисляется выражение в правой части присваивания, а затем его значение присваивается переменной, указанной в левой части оператора.

Например, для оператора

Rezult:=A div B;

сначала выполняется целочисленное деление значения переменной *A* на значение переменной *B*, а затем результат присваивается переменной *Rezult*.

Примеры применения оператора присваивания:

```
A:=22;
B:=A+6;
X:=a/10;
text:='Privet!';
```

Пример, демонстрирующий работу команды присваивания.

```
program primer;
    var a:integer;
begin
```

```
a:=5;        {переменной А присваивается значение 5 – исходное значение}
writeln('a=',a);        {вывод на экран монитора значения переменной а}
a:=2*a;                 {значение переменной А увеличивается в 2 раза}
writeln('a=',a);   {вывод на экран монитора промежуточного значения переменной а}
a:=a+1;      {значение переменной А увеличивается на 1}
writeln('a=',a);     {вывод на экран монитора значения переменной а - результат}
end.
```

В результате выполнения программы на экране в «окне вывода» появится следующая информация об изменении значений переменной А:

a=5
a=10
a=11

Важно помнить: в результате выполнения команды присваивания предыдущее значение переменной стирается.

Для выполнения операций ввода-вывода служат четыре процедуры: *Read, Readln, Write, WriteLn.*

Процедура чтения Read обеспечивает ввод числовых данных, символов, строк и т.д. для последующей их обработки программой.

Формат процедуры Read: **Read (x1, x2, …, xn);**

где x1, x2, …, xn- переменные допустимых типов данных.

Значения x1, x2, …, xn набираются минимум через один пробел на клавиатуре и высвечиваются на экране. После набора данных для одной процедуры Read нажимается клавиша ввода Enter.

Значения переменных должны вводится в строгом соответствии с синтаксисом языка Паскаль. Если соответствие нарушено (например, x1 имеет тип Integer, а при вводе набирается значение типа Char), то возникают ошибки ввода-вывода. Сообщение об ошибке имеет вид: *I/O error XX*, где *XX* - код ошибки.

Процедура чтения ReadLn аналогична процедуре Read, единственное отличие заключается в том, что ее выполнения курсор автоматически перейдет на новую строку.

Процедура записи Write производит вывод числовых данных, символов, строк, булевских значений.

Формат процедуры Write: **Write (<список вывода>);**

где <список вывода>- последовательность переменных, констант, математических выражений, перечисляемых через запятую.

Процедура записи WriteLn аналогична процедуре Write, единственное отличие заключается в том, что после вывода последнего в списке значения для одной процедуры WriteLn данные для следующей процедуры WriteLn будут выводиться с начала новой строки.

Процедуры вывода допускают использование указания о ширине поля, отводимого под значение в явном виде:

WRITE (Y:m:n,X:k:l,...);

WRITELN (Y:m:n:,X:k: I,...);

где m и k- количество позиций, отведенных под запись значения переменных Y и X соответственно;

n и I - количество позиций, отведенных под запись дробной части чисел Y и X.

Пример 1. Найти сумму двух вещественных чисел.

Используемые переменные: x, y –вводимые числа (исходные данные), z – их сумма (результат)

```
Program  primer;
var
  X, Y, Z: Real;
begin
  Writeln(‘Введите два  числа X и Y:');  {вывод строки  подсказки}
  Readln(X,Y);                           {ввод чисел  x и  y}
  Z := X + Y;                            {вычисление суммы  x и  y}
  Writeln(' X + Y=’, Z);                 {вывод результата}
end.
```

Результат

Введите два числа X и Y:

15 22

X + Y=37

Пример 2.

Составить программу расчета значения функции.

$Z = |\cos x^4 - 3 \operatorname{tg} x^2| + 0.8 \sin yx^2 + 10$ при любых значениях *x* и *y*.

Результат вывести в виде: при x= и y=… z=…

Используемые переменные: **x, y** -аргументы, **z** – значение функции

```
Program pr1;
    Var   x,y,z: real;
Begin
  writeln('введите X Y');                          {вывод строки  подсказки}
  readln (x,y);                                    {ввод аргументов  x и  y}
  z:=abs(cos(sqr(x)*sqr(x)-3*sin(sqr(x))/cos(sqr(x))))+0.8*sin(y*sqr(x))+10;
  writeln('при x=',x:8:2,' y=',y:8:2,' z=',z:8:2);  {вывод результата}
End.
```

Результат

```
введите X Y
1 2
при x=1.00 y=2.00 z=11.59
```

Пример 3.

Вводится вещественное число *a*. Не пользуясь никакими арифметическими операциями, кроме сложения, получить $7a$ за четыре операции.

Используемые переменные: **a** –вводимое число,

b, c, d – вспомогательные переменные

```
Program pr2;
Var    a,b,c,d:real;
Begin
  write('введите a ');
  readln (a);                    {ввод исходного числа}
  b:=a+a;                        {2a}
  c:=b+b;                        {4a}
  d:=b+c;                        {6a}
  a:=d+a;                        {7a}
  writeln('7a=',a:8:2);          {вывод результата}
End.
```

Результат

```
введите a 2
7a=   14.00
```

Пример 4.

Найти площадь круга и длину окружности.

Используемые переменные: **r** - радиус, **d** – длина окружности,
s – площадь круга

```
Program pr3;
Var     d,r,s:real;
Begin
  write('введите радиус окружности ');
  readln (r);                              {ввод радиуса}
  d:= 2*Pi*r;                              {вычисление длины окружности}
  s:=Pi*sqr(r);                            { вычисление площади круга}
  writeln('длина окружности= ',d:4:2);     {вывод результата}
  writeln('площадь окружности= ',s:4:2);
End.
```

Результат

```
введите радиус окружности 5
длина окружности=31.42
площадь окружности=78.54
```

Пример 5.

Вычисление суммы цифр введенного натурального двузначного числа.

Используемые переменные: **n** - двузначное число, **a, b** – цифры числа

```
Program  pr4;
Var   n, a, b: integer;
Begin
       write('n=   ');    readln(n);        {ввод исходного двузначного числа}
       a:=n div 10;                         {1-я цифра}
       b:=n mod 10;                         {2-я цифра}
      writeln('сумма = ', a+b);             {вывод результата}
End.
```

Результат

n=48

сумма=12

Пример 6.

Введенное натуральное 4-значное число изменить так, чтобы 2 и 3 цифры поменялись местами.

Четырехзначное число N можно представить в виде суммы разрядных слагаемых: N=n1*1000+n2*100+n3*10+n4, где n1, n2, n3, n4 – цифры соответствующих разрядов. Например, 3562=3*1000+5*100+6*10+2

Чтобы во введенном числе N поменять цифры местами, нужно выделить каждую цифру и записать число в виде N=n1*1000+n3*100+n2*10+n4

Используемые переменные: N – вводимое четырехзначное число,
n1, n2, n3, n4 – цифры

```
Program pr5;
Var  N, n1, n2, n3, n4:integer;
Begin
  write('введите n ');
  readln (n);                          {ввод исходного 4-значного числа}
   n1=N div 1000;                      {1-я цифра числа}
   n2:=N div 100 mod 10;               {2-я цифра числа }
   n3:=N div 10 mod 10;                {3-я цифра числа }
   n4:=N mod 10;                       {4-я цифра числа}
   n:= n1*1000+n3*100+n2*10+n4;        {получение числа в виде суммы разрядных
                                        слагаемых}
  writeln('результат ', n);            {вывод результата}
End.
```

Результат:

введите n 1234

результат 1324

Пример 7.

Обмен значениями переменных X и Y.

Для того, чтобы переменные X и Y поменялись своими значениями, можно использовать вспомогательную переменную, например, T. Вспомогательная переменная нужна для того, чтобы сохранить временно значение переменной X. После этого в переменную X можно занести значение переменной Y, а Y - присвоить значение X.

Используемые переменные: **X, Y** – вводимые числа,
T – вспомогательная переменная

```
Program  pr6;
Var    X, Y, T: integer;
begin
      write('Введите X   Y   ');
      readln(X, Y);                    {ввод исходных чисел}
      T:=X;
      X:=Y;
```

```
    Y:=T;
    writeln('X=', X, 'Y=',Y);   {вывод результата}
end.
```

Результат:

```
Введите X   Y   3   7
X=7 Y=3
```

Контрольные вопросы

1. Понятие переменной.
2. Команда присваивания. Формат, примеры.
3. Общие сведения о вводе-выводе данных.
4. Процедуры ввода данных. Read. Формат, примеры.
5. Процедура ReadLn. Формат, примеры.
6. Процедуры вывода данных. Write, WriteLn. Форматы, примеры.

Задания для самостоятельной работы

1. Составить программу расчета значения функции
   $Z = |3\ e^x + 3 - 2\ \ln xy| + 1{,}8x^2 + 1$ при любых значениях x и y. Результат вывести в виде: при x= … и y=… z=…
2. Составить программу расчета значения функции
   $Z = tg\ x - |2\ \sin 2y + 7.8\ \cos x| + 10$ при любых значениях x и y. Результат вывести в виде: при x= … и y=… z=…
3. Составить программу расчета значения функции
   $Z = (x - 2\ ctg\ 2y)/|8x - 5\ arctg\ y|$ при любых значениях x и y. Результат вывести в виде: при x= … и y=… z=…
4. Даны две целые переменные a, b. Составить программу обмена значениями этих переменных не используя дополнительных переменных
5. Вводится вещественное число a. Не пользуясь никакими арифметическими операциями, кроме умножения, получить a^4 за две операции.
6. Вводится вещественное число a. Не пользуясь никакими арифметическими операциями, кроме умножения, получить a^6 за три операции.
7. Вводится вещественное число a. Не пользуясь никакими арифметическими операциями, кроме умножения, получить a^7 за четыре операции.
8. Вводится вещественное число a. Не пользуясь никакими арифметическими операциями, кроме умножения, получить a8 за три операции.
9. Вводится вещественное число a. Не пользуясь никакими арифметическими операциями, кроме умножения, получить a^9 за четыре операции.
10. Вводится вещественное число a. Не пользуясь никакими арифметическими операциями, кроме умножения, получить a^{10} за четыре операции.
11. Найти сумму цифр введенного 4-значного числа.
12. Определить сумму квадратов цифр введенного 3-значного числа.
13. Введено 3-значное число. Вывести число в зеркальном отображении.
14. Введено 3-значное число. Вывести число в зеркальном отображении
15. Определить сумму квадратов цифр введенного 3-значного числа.

16. Вводится 4-значное число. Из его цифр получить два двузначных числа. Первое состоит из 1 и 3 цифр исходного числа, второе – из 2-й и 4-й. Например, 3765 -> 36 и 75
17. Вводятся два 3-значных числа. Получить 6-значное число, состоящее из цифр исходных чисел. Например, 265 и 145 -> 265145
18. Вводится 3-значное число. Из его цифр получить два двузначных числа. Первое состоит из 1 и 3 цифр исходного числа, второе – из 2-й и 3-й. Например, 765 -> 75 и 65
19. Вводятся два числа: 2-значное и 3-значное. Получить 5-значное число, состоящее из цифр исходных чисел. Например, 25 и 137 -> 25137
20. Составить программу, которая переводит значение температуры из шкалы Цельсия в шкалу Фаренгейта по формуле TF = 1.8TC + 32
21. Составить программу, которая переводит значение веса из фунтов в килограммы (1 фунт=409,5 г).
22. Составить программу, которая преобразует введенное с клавиатуры дробное число в денежный формат. Например, число 45.7 должно быть преобразовано к виду 45 руб. 70 коп.
23. Составить программу для перевода суммы из долларов в рубли. Вводится текущий курс доллара и сумма в долларах. Результат должен выводиться в денежном формате, например, 345 руб. 50 коп.
24. Составить программу для вычисления величины дохода по вкладу. Вводятся величина вклада, процентная ставка(в процентах годовых) и время хранения (в днях).
25. Написать программу для вычисления стоимости поездки на дачу (туда и обратно. Исходные данные: расстояние до дачи (в км), количество бензина, которое потребляет автомобиль на каждые 100 км, цена 1 л бензина.

2. ОПЕРАТОРЫ ВЕТВЛЕНИЯ

Краткие теоретические сведения

Операторы ветвления предназначены для выбора к исполнению одного из возможных действий (операторов) в зависимости от некоторого условия (при этом одно из действий может быть пустым, т. е. отсутствовать). В качестве условий выбора используется значение логического выражения.

В Паскале имеются два оператора ветвления: условный оператор *if* и оператор множественного выбора *case.*

Условный оператор if

Формат оператора :

if <условие> then <оператор1> else <оператор2>;

В переводе с английского языка данные форматы можно определить как:

ЕСЛИ<*условие*>ТО<*оператор1*>ИНАЧЕ<*оператор2*>

Выполнение условного оператора начинается с вычисления значения логического выражения, записанного в условии. Значением логического выражения является истина (True) или ложь (False). Если условие истинно, то выполняется <оператор1>, в противном случае - <оператор2>. После служебных слов Then и Else могут выпол-

няться несколько операторов. Как объяснить, какие операторы должны быть выполнены? В этом случае на помощь приходят операторные скобки Begin и End, которые отмечают соответственно начало и конец требуемой группы операторов.

Примечание. После **<оператора1>**, то есть перед служебным словом **Else**, символ ";" (точка с запятой) **не ставится!!!**

Условные конструкции могут содержать не одно, а несколько условий. Для объединения этих условий в одном условном операторе применяются следующие логические операции: NOT (отрицание), AND ("и"), OR ("или"), XOR (исключающее "или"). Обозначения и результаты этих операций рассмотрены на 14 шаге. Приведем пример: запишем на языке Pascal формулу:

$$y=\begin{cases} x-12, \text{ при } 0<=x<=14, \\ x^2, \text{ при } x>14. \end{cases}$$

Соответствующий ей условный оператор будет выглядеть следующим образом:

```
If (x>=0) and (x<=14) Then y:=x-12   Else y:=x*x;
```

Примечания.

1. Отношения, стоящие слева и справа от знака логической операции, должны быть заключены в скобки, поскольку логические операции имеют более высокий приоритет.
2. Принят следующий приоритет операций:
   NOT;
   AND, *, DIV, MOD;
   OR, XOR, +, -;
   операции отношения.

В некоторых случаях оператор после служебного слова Else может отсутствовать. Тогда условный оператор будет выглядеть так:

If <условие> Then <оператор>;

В качестве операторов, располагающихся после служебных слов Then и Else, могут выступать другие условные операторы.

Пример 1.

Вычислить частное двух целых чисел. В связи с тем, что делить на нуль нельзя, организовать контроль ввода данных.

Для контроля вводимых значений делителя используем оператор условного перехода *if ... then ... else.*

```
program Pr1;
var
A, B : integer;
Rezult: real;
Begin
Write('Введите значение делимого A: ');   Read(A) ;
    Write('Введите значение делителя B: ');  Read(B) ;
    if B=0          {Контроль ввода числа B}
          then Writeln('На нуль делить нельзя')      {Условие выполнено}
```

```
        else        {Условие не выполнено}
            begin
                Rezult := A / B;
                Writeln('Частное чисел ',A,' и ' ,B, ' = ', Rezult);
            end;
    End.
```

Пример 2.

Выбор минимального из трех введенных целых чисел.

Для того, чтобы выбрать минимальное из трех введенных чисел, нужно выбрать наименьшее сначала из двух и запомнить в переменную m, а затем значение m сравнить с третьим числом.

Используемые переменные: **a, b, c** – вводимые числа, **m** – минимальное из них

```
Program pr2;
Var   a, b, c, m:  integer;
BEGIN
   Write('a, b, c='); readln(a, b, c);      {ввод исходных чисел}
   if  a<b  then  m:=a  else  m:=b;   {выбор минимального из a и b}
   if  c<m  then  m:=c;               {сравнение с третьим числом}
   writeln('Min=',  m);               {вывод результата}
END.
```

Результат:

1 случай:

```
a, b, c= 15   6    9
Min=6
```

2 случай:

```
a, b, c= 1   45   4
Min=1
```

3 случай:

```
a, b, c= 25   16    9
Min=9
```

Пример 3.

Решение квадратного уравнения вида $ax^2 + bx + c = 0$.

Используемые переменные: **a, b, c** – коэффициенты квадратного уравнения, **d** - дискриминант, **x1, x2** – корни уравнения

```
Program pr3;
Var  a, b, c, d, x1, x2:   real;
Begin
   Write('a, b, c='); readln(a, b, c);   {вводим коэффициенты квадратного
```

```
                                        уравнения}
    d:=sqr(b)-4*a*c;                    {вычисляем дискриминант}
    if  d>0  then  begin                {если дискриминант положительный, то
                                        вычисляем два корня}
                   x1:=(-b+sqrt(d))/(2*a);   writeln('x1=',  x1:6:2);
                   x2:=(-b-sqrt(d))/(2*a);   writeln('x2=',  x2:6:2)
                   end
           else  if  d=0  then  begin { иначе, если дискриминант равен 0, то
                                        вычисляем один корень}
                                x1:=-b/(2*a);   writeln('x=',  x1:6:2)
                                end
                         else  writeln('Корней  нет'); {иначе выводим сообщение
                                                        'Корней нет'}
End.
```

Результат:

1 случай:

```
a, b, c= 1   -2    1
x=  1.00
```

2 случай:

```
a, b, c= 1   -6    8
x1=  4.00
x2=  2.00
```

3 случай:

```
a, b, c= 5   1    2
Корней  нет
```

Пример 4.

Определить, есть ли в записи трехзначного числа хотя бы одна нечетная цифра.

Используемые переменные: **a**- вводимое трехзначное число,

a1, a2, a3 – его цифры

```
Program pr4;
Var  a, a1, a2, a3:  integer;
Begin
    Write('a=');  readln(a);        {ввод исходного числа}
    a1:= a div 100;                 {1-я цифра}
    a2:= (a div 10) mod 10;         {2-я цифра}
    a3:= a mod 10;                  {3-я цифра}
    if  (a1 mod 2=1) or (a2 mod 2=1) or (a3 mod 2=1)     {Если 1-я цифра нечетная,
                 then writeln('Yes')  else writeln('No');  или 2-я нечетная, или
                                                        3-я нечетная, то выводим
                                                        'Yes', иначе выводим 'No'}
End.
```

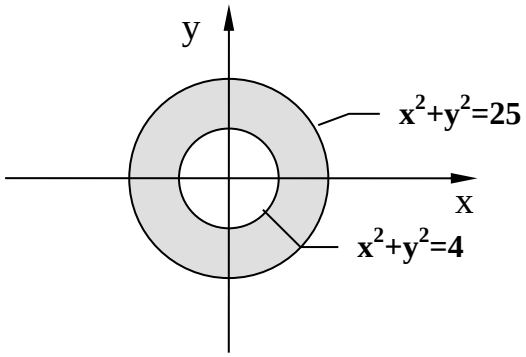
Результат:

1 случай:
a=418
Yes

2 случай:
a=246
No

Пример 5.

Вводятся координаты точки. Определить попадает ли точка в заштрихованную область или нет.



Решение

Точка будет принадлежать заштрихованной области, если она лежит **внутри** большого круга ($x^2+y^2<25$) **И,** одновременно, **за** пределами малого круга ($x^2+y^2>4$).

```
Program pr5;
Var  x,y: real;
Begin
  Write('введите x y ');
  readln (x,y);                          {ввод координат точки}
  if (sqr(x)+sqr(y)<25) and (sqr(x)+sqr(y)>4) {проверка условия принадлежности}
     then writeln('точка попадает в заштрихованную область')
     else writeln('точка не попадает в заштрихованную область');
end.
```

Результат:

1 случай:
введите x y 2 2
точка попадает в заштрихованную область

2 случай:
введите x y 2 5
точка не попадает в заштрихованную область

Пример 6.

Введено трехзначное число. Изменить число, увеличив все четные цифры на 2, а цифру 8 заменить на 0.

Решение

Для решения задачи необходимо разбить его на цифры. Каждую цифру следует изменить следующим образом: если цифра четная и меньше 8, то увеличить ее на 2, иначе, если цифра 8, то заменить ее на 0.

Используемые переменные: **n** – вводимое число, **a,b,c** – его цифры

```
Program pr6;
Var n,a,b,c:Integer;
begin
  write('введите n');     readln (n);                    {ввод числа}
  a:=n div 100;                                          {1-я цифра}
  b:=n div 10 mod 10;                                    {2-я цифра}
  c:=n mod 10;                                           {3-я цифра}
                    {проверяем каждую цифру и изменяем ее}
  if (a mod 2=0) and (a<8) then a:=a+2 else if a=8 then a:=0;
  if (b mod 2=0) and (b<8) then b:=b+2 else if b=8 then b:=0;
  if (c mod 2=0) and (c<8) then c:=c+2 else if c=8 then c:=0;
 n:=a*1000+b*100+c*10+d;    {формируем число из измененных цифр}
  writeln('n= ',n);                          {вывод результата}
end.
```

Результат:
введите n 1824
результат 1046

Оператор множественного выбора case

Если один оператор *if* может обеспечить выбор из двух альтернатив, то оператор выбора *case* позволяет сделать выбор из произвольного числа имеющихся вариантов. Он состоит из выражения, называемого *селектором* (*selection — выбор альтернативы*), и *списка параметров*, каждому из которых предшествует список *констант выбора* (список может состоять и из одной константы).

Формат записи оператора case:

```
case <выражение-селектор> of
      <список1>: <оператор1; >
      <список2>: <оператор2; >
      …
      <списокN>: <операторN>
else <оператор>
end;
```

Оператор *case* работает следующим образом. Сначала вычисляется значение выражения-селектора, затем обеспечивается реализация того оператора, константа выбора которого равна текущему значению селектора. Если ни одна из констант не равна текущему значению селектора, выполняется оператор, стоящий за словом *else.* Если слово *else* отсутствует, активизируется оператор, находящийся за словом *end*, т. е. первый оператор за границей *case.*

Пример 7.

Составить программу, которая выводит меню и выполняет указанные действия с заданными целыми числами:

1 - произведение двух чисел
2 - частное двух чисел
3 - сумма двух чисел

Используемые переменные: **a,b** – вводимые числа, **n** – номер операции

```
program pr7;
var a,b,n: integer;
begin
  writeln('1 – произведение двух чисел');
  writeln('2 – частное двух чисел');
  writeln('3 – сумма двух чисел');
  write('введите номер операции ');
  readln(n);
  write('введите два числа');    readln(a,b);
  case n of
    1: writeln('произведение=',a*b);
    2: writeln('частное=',a/b:4:2);
    3: writeln('сумма=',a+b);
    else writeln('неправильный  номер операции');
  end;
end.
```

Результат:

```
1 – произведение двух чисел
2 – частное двух чисел
3 – сумма двух чисел
введите номер операции 2
введите два числа 3 5
частное=0.60
```

Пример 7.

Ввести первую букву времени года и вывести соответствующее название времени года на русском языке.

```
Program pr7;
  Var  N: char;
Begin
          writeln ('введи первую букву времени года ');    readln (N);
            case  N  of
                 'з' , 'З' : writeln ('зима');
                 'в', 'В' : writeln ('весна');
                 'л', 'Л' : writeln('лето');
                 'о', 'О' : writeln('осень')
                 else writeln('Нет такого времени года');
           end;
```

End.

Результат:
введи первую букву времени года о
осень

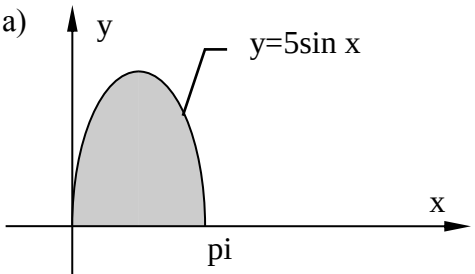
Контрольные вопросы

1. Условные операторы. Оператор if. Формат, описание. Основные правила использования. Особенности использования вложенного оператора if.
2. Примеры использования оператора if.
3. Оператор выбора case. Формат, описание.
4. Правила использования оператора case. Примеры использования.

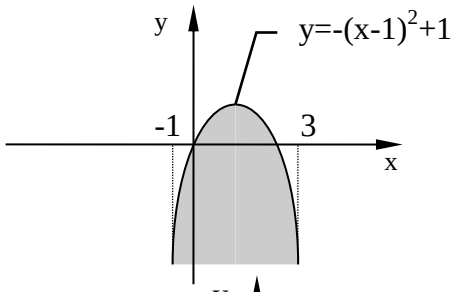
Задания для самостоятельной работы

1. Вводятся координаты точки. Определить попадает ли точка в заштрихованную область или нет.

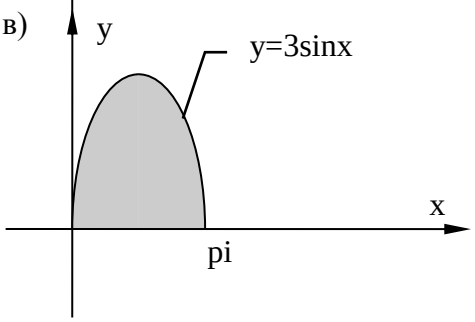
а)



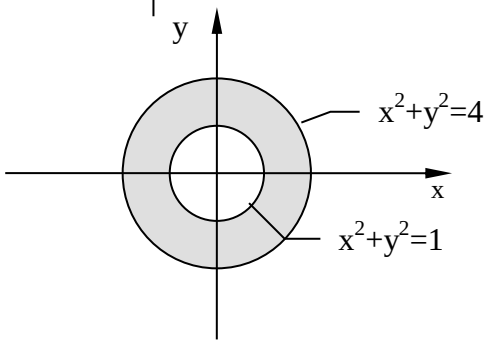
б)



в)



г)



2. Вычислить $y=\begin{cases}(X+1)^2+\ln x, & \text{если } x>0\\ |x|, & \text{если } x\le 0\end{cases}$

3. Вычислить $y=\begin{cases}X^2, & \text{если } x<0\\ X+2\sin X, & \text{если } 0\le x<20\\ 1/x, & \text{если } x\ge 20\end{cases}$

4. Введено трехзначное число. Найти сумму четных цифр.
5. Введено четырехзначное число. Содержится ли в записи этого числа цифра 7?

6. Введено трехзначное число. Если в записи числа встречается цифра 5, то записать число в зеркальном отображении.
7. Введено трехзначное число. Если сумма его цифр нечетна, то увеличить число вдвое.
8. Введено четырехзначное число. Найти сумму цифр, кратных 3-м.
9. Вводятся X и Y. Если хотя бы одно из этих чисел положительно, то найти их произведение. Иначе – найти их сумму.
10. Вводятся X и Y. Если X больше Y, то произвести их обмен.
11. Из чисел A, B, C, D выбрать максимальное.
12. Вводятся X и Y. Заменить большее из этих чисел разностью большего и меньшего.
13. Сколько среди заданных чисел A, B, C, D нечетных.
14. Вводятся A, B, C, D. Найти среднее арифметическое максимального и минимального.
15. Вводятся числа A, B, C. Упорядочить их по убыванию.
16. Из чисел A, B, C, D выбрать минимальное.
17. Определить, какие из чисел A, B, C, D. принадлежат интервалу (-10, 15).
18. Определить, есть ли среди цифр заданного трехзначного числа одинаковые.
19. Составить программу, определяющую является ли билет с 6-значным номером счастливым (счастливым является билет, у которого сумма первых 3 десятичных цифр равна сумме 3 последних).
20. Вводится трехзначное число. Определить, является ли оно числом Армстронга. Число Армстронга — натуральное число, которое равно сумме своих цифр, возведённых в степень, равную количеству его цифр. Например, число 153 — число Армстронга, потому что $1^3 + 5^3 + 3^3 = 153$
21. Вводятся положительные a, b, c d. Определить, можно ли прямоугольник со сторонами a, b поместить внутри прямоугольника со сторонами c, d так, чтобы каждая из сторон первого прямоугольника была параллельна или перпендикулярна каждой стороне второго треугольника.
22. Составить программу, проверяющую знание таблицы умножения. В программе выбираются случайным образом целые числа X ($1 \le X \le 9$) и Y ($1 \le Y \le 9$) и предлагается пользователю ввести ответ. Результат выполнения программы: ‘Правильно’ или ‘Неправильно’.
23. Вводятся числа A, B, C. Упорядочить их по возрастанию.
24. Составить программу, которая по введенной начальной букве выводит название цветов радуги (красный, оранжевый, желтый, зеленый, голубой, синий, фиолетовый).
25. Составить программу, которая по введенному порядковому номеру выводит название дня недели.
26. Составить программу, которая позволяет ввести номер месяца и вывести его название.
27. По числу текущего месяца определить день недели.
28. Напишите программу, которая по номеру дня недели - целому числу от 1 до 7 - выдает в качестве результата количество занятий в вашей группе в соответствующий день.
29. В зависимости от стажа работы педагогам введена надбавка в размере: для работающих от 5 до 10 лет–10%; для работающих от 10 до 15 лет–15%; для рабо-

тающих свыше 15 лет–20%. Составьте программу, которая по заданному стажу работы и размеру оклада определит размер заработной платы.

30. Составить программу, которая после введенного с клавиатуры числа (в диапазоне от 1 до 999), обозначающего денежную величину, дописывает слово «рубль» в правильной форме. Например, 5 рублей, 21 рубль, 173 рубля.

3. ЦИКЛЫ

Краткие теоретические сведения

Если в программе возникает необходимость неоднократно выполнить некоторые операторы, то используются *операторы повтора (цикла)*. В языке Паскаль различают три вида операторов цикла: *while, repeat, for*. Они используются для организации циклов различных типов. Выражение, управляющее повторениями, должно иметь булевский тип.

Если число повторений оператора (составного оператора) заранее неизвестно, а задано лишь условие его повторения (или окончания), используются операторы *while, repeat*. Оператор *for* используется, если число повторений заранее известно.

Оператор цикла for

В случаях, когда число повторений может быть заранее известно, для организации циклической обработки информации применяется оператор повтора *for*. Часто этот оператор повтора называют *оператором цикла с параметром,* так как число повторений задается переменной, называемой *параметром цикла*, или управляющей переменной. Оператор повтора for состоит из *заголовка* и *тела цикла*.

Он может быть представлен в двух форматах:

for <параметр цикла> := <S1> to <S2> do <оператор>;

for <параметр цикла> := <S1> downto <S2> do <оператор>;

где *Sl* и *S2* — выражения, определяющие соответственно начальное и конечное значения параметра цикла;

for ... do — заголовок цикла;

<оператор> — тело цикла.

Значение управляющей переменной изменяется на +1 (в случае to) или –1 (в случае downto).

Тело цикла может быть простым или составным оператором. Оператор *for* обеспечивает выполнение тела цикла до тех пор, пока не будут перебраны все значения параметра цикла от начального до конечного.

Пример 1.

Вывести квадраты первых десяти натуральных чисел.

Решение

Используемые переменные: i –натуральные числа, x– их квадраты

```
Program pr1;
Var  i, x:  integer;
```

```
 begin
      for i:=1 to 10 do begin        {перебираем натуральные числа от 1 до 10}
          x:=sqr(i);                 {возводим очередное число в квадрат}
          write(x, '  ');             {выводим полученное значение}
      end;
end.
```

Результат

1 4 9 16 25 36 49 64 81 100

Пример 2.

Найти сумму 1 + 1/3 + 1/5 +...(N слагаемых).

```
Program pr2;
Var   I, N:  integer;
      S:  real;
begin
        Write('N='); Readln(N);   {вводим количество слагаемых}
        S:=0;                      {обнуляем сумму}
      {выполняем цикл N раз, добавляя к сумме по одному слагаемому}
      For I:=1 to N do
          S:=S+1/(2*I-1);
      Writeln('S=',S:5:2);  {выводим  результат с двумя десятичными знаками}
end.
```

Результат:

N=4

S= 1.68

Пример 3.

Дано натуральное число n. Вычислить $3^1+3^2+\ldots...+3^n$

```
program pr3;
var i,n:integer;
    s:real;
begin
  write('введите n ');
  readln(n);
  s:=0;
  for i:=1 to n do
    s:=s+ exp(i*ln(3));
  writeln('сумма=',s:4:0);
  readln;
end.
```

Результат:

введите n 5

сумма= 363

Пример 4.

Найти сумму ряда: $S = x - x^2/4 + x^3/9 - x^4/16...$ (n слагаемых).

```
Program Pr4;
Var   i, n, z: integer;
        p, s: real;
begin
      writeln ('введите n');  readln ( n );
      writeln('введите x');  readln( x );
      z:=1;  p:=x;  s:=0;
      for i:=1 to n do begin
              s:= s + z*p/sqr(i);  p:=p*x;   z:=-z
      end;
      writeln ('S=', S :6:2);
      readln
end.
```

Пример 5.

Вычислить произведение:

$(1 + \sin 2)*(2 + \sin 3)*...$ (n сомножителей).

Найти сумму ряда: $S = x - x^2/4 + x^3/9 - x^4/16...$ (n слагаемых).

```
Program  Pr5;
Var  n, i:  integer;
       p: real;
begin
      write('n=');   readln(n);   p:=1;
      for   i:=1   to   n  do
            p:=p* (i + sin(i+1));
      writeln('p=', p:6:3);
end.
```

Результат:

n=5

p=131.645

Пример 6.

Подсчитать количество двузначных чисел, у которых сумма цифр нечетна.

```
Program Pr6;
Var   n, a, b, k:   integer;
begin
      k:=0;   for   n:=10  to  99  do  begin
                  a:= n div 10;  b:=n mod 10;
```

```
            if  (a+b) mod 2=1  then  k:=k+1;
        end;
    writeln('k=',k)
end.
```

Оператор цикла Repeat

Оператор повтора *repeat* состоит из заголовка repeat, тела и условия окончания until.

Формат записи:

repeat
<оператор;>
…
<оператор>
until <условие окончания цикла>;

Операторы, заключенные между словами *repeat и until,* являются телом цикла. Вначале выполняется тело цикла, затем проверяется условие выхода из цикла. Именно поэтому цикл, организованный с помощью оператора *repeat*, в любом случае *выполнится хотя бы один раз*. Если результат булевского выражения равен *False,* то тело цикла активизируется еще раз; если результат *True*, происходит выход из цикла.

Цикл *repeat* часто называют *циклом с постусловием*, или *циклом "ДО",* так как он прекращает выполняться, как только значение выражения условия, записанного после слова *until*, равно *True (истина).*

При программировании операторов тела цикла следует обеспечить влияние, по крайней мере, одного из операторов тела цикла на значение условия, иначе цикл будет выполняться бесконечно.

Пример 7.

Протабулировать функцию y:=sin(x)*x интервале [-π/2,π/2] с шагом π/10.

```
Program pr7;
Var x, y:  real;
begin
    x:=-pi/2;
    repeat
        y:=sin(x)*x;
        writeln('x=',x:8:2,'  y=',y:8:2);
        x:=x+pi/10
     until x>pi/2;
end.
```

Пример 8. Вычислить сумму $S = \frac{i}{(i+1)^2}$ с заданной точностью E=0.0001.

```
Program pr8;
Var S, e, slag:  real;
    k: integer;
begin
     e:=0.0001; s:=0; k:=1;
```

```
 slag:= k/sqr(k+1);
repeat
  s:=s+slag;
  k:=k+1;
  slag:=k/sqr(k+1);
until slag<e;
write ('s=', s:6:2);
end.
```

Оператор цикла while

Оператор *while (пока)* часто называют *оператором цикла с предусловием* за то, что проверка условия выполнения тела цикла производится в самом начале оператора.

Формат записи:

while <условие продолжения повторений> do
<тело цикла>;

Условие - логическое выражение, *тело цикла* - простой или составной оператор.

Перед каждым выполнением тела цикла вычисляется значение выражения условия. Если результат равен *True,* тело цикла выполняется и снова вычисляется выражение условия. Если результат равен *False*, происходят выход из цикла и переход к первому после *while* оператору.

Цикл *while* может не выполниться ни разу, если при первой проверке условие оказалось ложным (*False).*

Пример 9.

Задана арифметическая прогрессия -21; - 16;… Определить номер первого положительного члена прогрессии.

a – очередной член прогрессии, n – его порядковый номер

```
Program Pr9;
var
   a, n: integer;
begin
   a:= -21;   n:=1;              {задаем начальные значения}
   while a<=0 do  begin         {пока очередной член прогрессии меньше либо ра
          a:=a+5;  n:=n+1;       равен 0, вычисляем следующий член прогрессии и end;
и его порядковый номер}
    writeln('n=', n);           {выводим номер 1-го положительного члена
                                  прогрессии}
end.
```

Результат:

n=6

Пример 10. Первоначальный вклад составил S рублей. Через сколько лет сумма вклада более, чем в 2 раза превысит первоначальный вклад, если годовой процент составляет x%.

Например, S=1000 р, x=10%

1 год S=1000+1000*10/100=1100
2 год S=1100+1100*10/100=1210

```
Program pr10;
Var  s, sum, x:  real;
n: integer;
 Begin
      Write(‘S=’); Readln(S);
      Write(‘x=’); Readln(x);
      Sum:=2*s; n:=0;
      While S<=Sum do begin
          s:=s+s*x/100;
          n:=n+1;
      end;
      write('через  ', n, ‘ лет’);
end.
```

Вложенные циклы

Циклическая конструкция может содержать в теле цикла другой цикл. Такие конструкции называют *вложенными* циклами. Глубина вложения и тип циклов может быть различными.

Например, a) For . . . do – внешний цикл
While . . . do – внутренний цикл (в данном случае глубина = 2)

b) While . . . do
For . . . do
Repeat
. . .
Until (глубина =3)

Рассмотрим механизм работы вложенных циклов на примере:

```
For a:=1 to 2 do
  For b:=1 to 3 do
    Writeln(‘a=’, a, ‘ b=’, b);
```

a=1 a=1 a=1	b=1 b=2 b=3	выполняется внутренний цикл по b при a=1
a=2 a=2 a=2	b=1 b=2 b=3	выполняется внутренний цикл по b при a=2

Параметр внешнего цикла не меняет свое значение до тех пор, пока внутренний цикл не завершит свою работу.

Пример 11. Вывести на экран таблицу Пифагора.

внутренний цикл (по j)

1 2 3 4 5 6 7 8 9

1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81

```
Program pr11;
var
   i, j: integer;
Begin
   for i:=1 to 9 do
   begin
     for j:= 1 to 9 do
       write(i*j:4);
     writeln;
    end;
end.
```

Внутренний цикл по j формирует очередную (i-ю) строку таблицы. Writeln – перевод курсора на новую строку.

Пример 12. Вводятся k натуральных чисел. Найти сумму цифр каждого из них.

Повторить k раз	1. Ввести число N 2. Найти сумму его цифр 3. Вывести результат

```
Program primer;
var
   n, s, a, i, k: integer;
begin
   write (‘k=’); readln(k);
   for i:=1 to k do
   begin
       Write(‘n=’); readln (n); s:=0;
        Repeat
           a:=n mod 10;
           s:=s+a;
```

```
        n:=n div 10;
      until n=0;
      writeln ('s=',s);
    end;

end.
```

Контрольные вопросы

1. Операторы цикла. Общая характеристика.
2. Оператор цикла for. Форматы записи, описание работы цикла, ограничения использования параметра цикла.
3. Примеры использования оператора for. Различие to и downto.
4. Оператор цикла Repeat. Формат записи, особенности использования.
5. Примеры программ с использованием оператора repeat.
6. Оператор цикла while. Формат записи, описание работы цикла.
7. Механизм работы вложенных циклов

Задания для самостоятельной работы

1. Дано натуральное число n. Вычислить n! (n!=1*2*3*…*n)
2. Вычислить: sin x+sin 2x+sin 3x +….(n слагаемых).
3. Дано натуральное число n. Найти сумму S=2/5 + 2/9 + 2/13 +… . (n слагаемых)
4. Дано натуральное число n. Найти сумму S=1+1/3 + 3/5 + 5/7 +… . (n слагаемых)
5. Вычислить: cos x+ 2cos 2x+ 3cos 3x +….(n слагаемых)
6. Вычислить: x+ 2x+ 3x +….(n слагаемых)
7. Вычислить: 2x+ 4x+ 6x +….(n слагаемых)
8. Дано натуральное число n. Найти сумму S=1- 1/5 + 1/9 - 1/13 +… . (n слагаемых)
9. Вычислить: 1*2+2*3+...+n*(n+1).
10. Найти произведение: P = (1 + x)·(3+2x)·(5+3x)·... . (n множителей)
11. Найти значение суммы ряда $\frac{1}{2^2}+\frac{1}{4^2}+\dots$ с точностью $\varepsilon=10^{-4}$.
12. Задана арифметическая прогрессия. 7,6; 6,3; … . Сколько членов прогрессии нужно сложить, чтобы полученная сумма стала <0.
13. Задана арифметическая прогрессия 2; 2,8; … . Сколько членов прогрессии нужно сложить, чтобы полученная сумма стала >20.
14. Задана арифметическая прогрессия. 7,1; 5,3; … . Сколько членов прогрессии нужно сложить, чтобы полученная сумма стала <0.
15. Подсчитать количество двузначных чисел, кратных 3.
16. Подсчитать сумму двузначных чисел, сумма цифр которых не превышает 10.
17. Протабулировать функцию $y=x^3-1$ на интервале [-1,3] с шагом 0.2.
18. Протабулировать функцию y= sin x – cos x на интервале $[-\pi,\pi]$ с шагом $\pi/10$.
19. Протабулировать функцию y= cos(x)*x на интервале $[-\pi,\pi]$ с шагом $\pi/10$.
20. Вы положили с S рублей в банк под p% ежегодного прироста. Определить: а) когда сумма вклада утроится? б) какой будет сумма вклада после 10 лет?
21. Готовясь к соревнованиям, лыжник в первый день побежал 10 км, затем каждый день увеличивал расстояние на 10%. Сколько километров пробежал он за неделю тренировок? На какой день он пробежал больше 15 км?

22. Малое предприятие в первый день работы выпустило P единиц товарной продукции. Каждый последующий день оно выпускало продукции на Q единиц больше, чем в предыдущий. Сколько дней потребуется предприятию, чтобы общее количество выпущенной продукции за все время работы впервые превысило запланированный объем T?
23. Составить программу, которая находит и выводит на печать все четырехзначные числа abcd, для которых выполняются следующие условия:
a, b, c, d - разные цифры и ab-cd=a+b+c+d. Здесь запись ab означает, что число составлено из цифр a и b.
24. Числа Фибоначчи определяются как a(0)=1, a(1)=1, a(i)=a(i-1)+a(i-2). Найти: а) N-ое число Фибоначчи. б) сумму первых N чисел Фибоначчи.
25. Определить, является ли число n простым.
26. Определить, является ли число n совершенным. Совершенное число́ — натуральное число, равное сумме всех своих собственных делителей (т. е. всех положительных делителей, отличных от само́го числа). Например, 6=1+2+3
27. Найти наименьшее общее кратное (НОК) двух натуральных чисел.
28. Вводится K пар натуральных чисел. Найти НОД каждой пары.
29. Вводится N натуральных чисел. Найти среднее арифметическое цифр каждого из них.
30. Вводится N натуральных чисел. Найти количество четных цифр в каждом из них.

4. ФУНКЦИИ ПОЛЬЗОВАТЕЛЯ

Краткие теоретические сведения

При решении достаточно большой задачи её рекомендуется разбивать на отдельные смысловые части (подпрограммы), программировать их отдельно, а затем объединять в единую программу. Использование подпрограмм считается хорошим стилем программирования. Подпрограмму можно выполнять в различных местах программы для различных исходных данных.

Любая программа может содержать несколько подпрограмм, каждая из которых может, в свою очередь, содержать обращения к другим подпрограммам. Для простоты изложения ограничимся подпрограммами, которые не содержат внутри себя обращений к другим подпрограммам, и их вызов осуществляется из основной программы.

В языке программирования Pascal имеется два вида подпрограмм: **процедуры (Procedure)** и **функции (Function).**

Процедуры и функции – это относительно самостоятельные части программы, имеющие собственное имя. Они описываются перед основной программой, а вызываются из основной программы.

Функция, определенная пользователем, состоит из *заголовка* и тела *функции*.

Заголовок содержит зарезервированное слово *function,* идентификатор (имя) функции, заключенный в круглые скобки, необязательный список формальных параметров и тип возвращаемого функцией значения. Тело функции представляет собой локальный блок, по структуре аналогичный программе. В целом структура функции, определенной пользователем имеет вид:

```
Function  <имя_функции>(<формальные параметры>) : <тип результата>;
    Label <метки>;
   Const <константы>;          раздел объявления меток, констант, типов данных,
   Type <типы данных>;         переменных (может отсутствовать)
   Var <переменные>;
Begin
  <операторы, составляющие тело функции>;
End;
```

Для того, чтобы значение функции было определено и передано в основную программу, в теле функции обязательно должен быть хотя бы один оператор присваивания вида:

<имя_функции>:=<значение>.

Вызов функции, определенной пользователем, осуществляется так же, как и любой стандартной функции Паскаля. Каждый аргумент должен соответствовать формальным параметрам, указанным в заголовке, и иметь тот же тип.

Функция возвращает результат через свое имя, поэтому имя функции ставится в правой части оператора присваивания.

Пример 1.

Создать функцию нахождения максимального из двух чисел

```
Program Pr1;
Var
   a,b,m:real;
Function Max(a,b:real):real;
begin
      if a>b                                          {если a  больше b}
           then Max:=a                          {то функция принимает значение a}
          else Max:=b;                        {иначе функция принимает значение b}
end;
BEGIN
 write('a,b='); readln(a,b);                    {ввод исходных данных}
 m:=Max(a,b);                                        {вызов функции}
 write('Max=',m:5:1);                             {вывод результата}
END.
```

Результат:

```
a,b=14  9
Max= 14.0
```

Пример 2.

Составить функцию для вычисления x^n, где x,n – целые числа

Степень принимает отрицательное значение, если основание степени X<0 и показатель степени –нечетное число.

```
Program Pr2;
Var
```

```
    x,n:integer;
Function stepen(x,n:integer):real;
begin
     if (n mod 2 <> 0) and (x<0) then stepen:=-exp(n*ln(abs(x)))
                                 else stepen:=exp(n*ln(abs(x)));
end;
BEGIN
    write('x,n='); readln(x,n);                    {ввод исходных данных}
    write('stepen=',stepen(x,n):6:2);              {вызов функции и вывод ее значения}
END.
```

Результат:

1 случай
x,n=2 3
stepen= 8.00
2 случай
x,n=-2 3
stepen= -8.00
3 случай
x,n=2 -2
stepen= 0.25
4 случай
x,n=-2 -2
stepen= 0.25

Пример 3.

Создать функцию нахождения наибольшего общего делителя двух натуральных чисел.

```
Program Pr3;
Var
 a,b:integer;
Function NOD(a,b:integer):integer;
begin
 while a<>b do                                  {пока числа не равны}
    if a>b then a:=a-b {заменяем большее из них разностью большего и меньшего}
           else b:=b-a;
 NOD:=a;
end;
BEGIN
 write('a,b='); readln(a,b);        {ввод исходных данных}
 write('NOD=', NOD(a,b));           {вызов функции и вывод ее значения}
END.
```

Результат:

a,b=24 16

NOD=8

Пример 4.

Два треугольника длинами своих сторон. Определить, площадь какого из них больше (создать функцию для вычисления площади треугольника по длинам его сторон).

Для решения задачи используем формулу Герона $S = \sqrt{p \cdot (p-x) \cdot (p-y) \cdot (p-z)}$, где x, y, z – стороны треугольника, p – полупериметр)

```
Program pr4;
var
  a1, b1, c1, s1, a2, b2, c2, s2: real;
            {функция вычисления площади треугольника со сторонами x,y,z}
Function PlTr(x ,y, z: real): real;
Var
     p: real;
Begin
  p:= (x + y + z)/2;                                  {полупериметр треугольника}
  PlTr:=Sqrt(p*(p-x)*(p-y)*(p-z));
End;
BEGIN                  {основная программа}
  {ввод исходных данных}
 Write('Стороны 1-го треугольника:');  Readln(a1, b1, c1);
 Write('Стороны 2-го треугольника:');  Readln(a2, b2, c2);
 S1:=PlTr(a1, b1, c1);{вызов функции для нахождения площади 1-го треугольника}
 S2:=PlTr(a2, b2, c2);{вызов функции для нахождения площади 2-го треугольника}
  if S1>S2 then Writeln('S1>S2')
          else if S1<S2 then Writeln('S1<S2')
                        else Writeln('S1=S2');
END.
```

Результат:

```
Стороны 1-го треугольника:3   4   5
Стороны 1-го треугольника:6  7   8
S1<S2
```

Пример 5.

Вычислить, используя функцию нахождения знаменателя:

$\frac{1}{2!}+\frac{2}{3!}+\frac{3}{4!}+...$(n слагаемых)

Решение:

```
Program pr5;
Var     s: real;
     i, n: integer;
Function Fakt(x: integer): real;          {функция вычисления факториала}
 Var
```

```
        i: integer; f: real;
begin
        f:=1;
        for i:=1 to x do f:=f*i;
        Fakt:=f;
end;
BEGIN
        write('n= ');       Readln(n);       {вводим количество слагаемых суммы}
        s:=0;
        For   i:=1  to  n  do                              {перебираем  i от 1 до n}
              s:=s+i/Fakt(i+1);          {добавляем к сумме очередную дробь, вызывая}
                                                 {  функцию для вычисления знаменателя}
        writeln('s=',s:6:4);                                  {вывод результата}
END.
```

Результат:
n=3
s=0.9583

Пример 6.

Вычислить, используя функцию нахождения знаменателя:

$$\frac{5}{1+2}+\frac{10}{1+2+3}+\frac{15}{1+2+3+4}+\ldots$$ (n слагаемых)

```
Program pr6;
Var     s: real;
      i, n: integer;
{ функция вычисления суммы первых m натуральных чисел}
Function Sum(m:integer ):integer;
Var
      k, S:  integer;
begin
      S:=0;
      for k:=1 to m do
                S:=S+k;
     Sum:=S;
end;

BEGIN
        write('n= ');       Readln(n);          {вводим количество слагаемых суммы}
        s:=0;
        for   i:=1  to  n  do                              {перебираем  i от 1 до n}
             s:=s+i*5/sum(i+1);       {добавляем к сумме очередную дробь, вызывая}
                                              {   функцию для вычисления знаменателя}
        writeln('s=',s:6:4)                                   {вывод результата}
 END.
```

Результат:
n=3
s=4.8333

Контрольные вопросы

1. Структура функции, определенной пользователем.
2. Как осуществляется вызов функции из основной программы?
3. Каким образом функция передает результат в основную программу?

Задания для самостоятельной работы

1. Найти минимальное из A, B, C, создав функцию выбора минимального из двух произвольных чисел.
2. Найти максимальное из чисел A, B, C, D, создав процедуру выбора максимального из 2-х произвольных чисел.
3. Сократить дробь вида a/b (a, b – вводимые натуральные числа), создав функцию для нахождения наибольшего общего делителя двух натуральных чисел.
4. Найти наибольший общий делитель чисел A, B, C, создав функцию для нахождения НОД двух натуральных чисел.
5. Два прямоугольных треугольника заданы своими катетами. Определить, у какого из них площадь больше (создать функцию для вычисления площади прямоугольного треугольника по его катетам).
6. Два прямоугольника заданы координатами вершин. Определить, площадь какого из них больше (использовать функцию нахождения площади прямоугольника).
7. Два прямоугольных треугольника заданы своими катетами. Определить, у какого из них периметр больше (создать функцию для вычисления периметра прямоугольного треугольника по его катетам).
8. Вводятся 2 натуральных числа. Найти среднее арифметическое цифр каждого из них (создать функцию для нахождения среднего арифметического цифр произвольного натурального числа).
9. Вводятся 3 натуральных числа. Найти сумму цифр каждого из них (создать функцию для нахождения суммы цифр произвольного натурального числа).
10. Вычислить $3^{-2} + 4^2 + 5^{-2}$, создав функцию для вычисления степени.
11. Вычислить $(-3^2 + 2^4)/4^{-2}$, создав функцию для вычисления степени.
12. Найти (a! + b!)/(a+b)!, создав функцию для вычисления факториала произвольного натурального числа.
13. Найти m! + (m+n)!, создав функцию для вычисления факториала произвольного натурального числа.
14. Вычислить, используя функцию нахождения знаменателя:
$\frac{1}{1+2} + \frac{3}{1+2+3} + \frac{5}{1+2+3+4} + \ldots$ (15 слагаемых)
15. Вычислить, используя функцию нахождения знаменателя:
$\frac{1}{2!} + \frac{4}{3!} + \frac{9}{4!} + \ldots$ (n слагаемых)
16. Вычислить, используя функцию нахождения знаменателя:
$\frac{2}{3!} + \frac{4}{5!} + \frac{6}{7!} + \ldots$ (n слаг.)
17. Вычислить, используя функцию нахождения знаменателя:

$\frac{2}{1+2}+\frac{4}{1+2+3}+\frac{6}{1+2+3+4}+...$ (15 слаг)

18.Вычислить, используя функцию нахождения знаменателя:

$\frac{1}{3!}+\frac{2}{4!}+\frac{3}{5!}+...$ (n слагаемых)

19.Вычислить, используя функцию нахождения знаменателя:

$\frac{2}{3!}+\frac{4}{5!}+\frac{6}{7!}+...$ (n слаг.)

20.Вычислить, используя функцию нахождения знаменателя:

$\frac{1}{1+3}+\frac{2}{1+3+5}+\frac{3}{1+3+5+7}+...$ (15 слаг)

5. ПРОЦЕДУРЫ ПОЛЬЗОВАТЕЛЯ

Краткие теоретические сведения

Процедура – это относительно самостоятельная часть программы, имеющая собственное имя. Процедура описывается перед основной программой.

Процедура возвращает результат через параметры, поэтому процедура является оператором.

Структура процедуры:

```
Procedure <имя_процедуры>(<формальные параметры>);   -заголовок процедуры
   Label <метки>;
   Const <константы>;        раздел объявления меток, констант, типов данных,
   Type <типы данных>;       переменных (может отсутствовать)
   Var <переменные>;
Begin
  <операторы >                -  тело процедуры
End;
```

Вызов процедуры осуществляется из основной программы указанием ее имени. Различают формальные и фактические параметры. Параметры, указанные в заголовке процедуры называются *формальными*. Они определяют, как будет происходить обработка конкретных значений этих параметров. *Фактические* параметры указываются при обращении к подпрограмме. При вызове подпрограммы значения фактических параметров ***автоматически*** передаются в переменные, являющиеся формальными параметрами. После чего начинается выполнение подпрограммы. Количество и тип формальных и фактических параметров должны совпадать.

Для передачи исходных данных в процедуру используются *параметры-значения*. Фактические параметры-значения могут быть константами, переменными, выражениями. Полученный в процедуре результат передается в основную программу с помощью *параметров-переменных*. В заголовке описания процедуры перед параметрами-переменными пишется Var.

Пример 1.

Создать процедуру вычисления среднего арифметического и среднего квадратичного натуральных чисел a и b.

```
Program pr1;
Var
  a, b: integer;
  SrAr, SrKv: real;  {SrAr- среднее арифметическое, SrKv – среднее квадратичное}
Procedure Sredn(a,b:integer; Var SrAr, SrKv: real);
{a,b – входные данные (параметры-значения}
 {SrAr, SrKv – результат выполнения процедуры (параметры-переменные)}
Begin                          {тело процедуры}
  SrAr:=(a+b)/2;
  SrKv:=SQRT(a*b);
End;
BEGIN                                  {основная программа}
       Write('a='); Readln(a);          {ввод a}
        Write('b='); Readln(b);         {ввод b}
       {вызов процедуры и передача фактических параметров a,b }
        Sredn(a, b, SrAr, SrKv);
         {вывод результата, полученного в процедуре}
       Writeln('Ср. арифм.=', SrAr:6:2, '  Ср. квадр.е =', SrKv:6:2);
END.
```

Результат:

```
a=2
b=8
Ср. арифм.=  5.00  Ср. квадр.=  4.00
```

Пример 2.

Создать процедуру для вычисления периметра и площади прямоугольного треугольника по его катетам a, b.

```
Program pr2;
Var
    a, b, p, s: real;                  {  a, b- катеты, p - периметр, s - площадь }
Procedure Treug(a,b: real; Var p, s: real);
 {a, b- входные данные(параметры-значения),}
{p, s-выходные данные(параметры-переменные)}
Begin
    p:= a + b + Sgrt(a*a + b*b);   {периметр треугольника с катетами a, b}
    s:=  a*b/2;                    { площадь треугольника с катетами a, b }
End;
Begin              {основная программа}
Write('a, b =');Readln(a, b);                          {ввод исходных данных}
   Treug(a, b,  p, s);            {вызов процедуры и передача исходных данных a, b}
   Writeln('p=', p:4:1, '    s=', s:4:1);  {вывод результата, полученного из процедуры}
End.
```

Результат:
a, b= 3 4
p=12.0 s= 6.0

Пример 3.

Создать процедуру, определяющую возможность построения треугольника со сторонами x, y, z. Из чисел a,b,c,d выбрать тройки таких, которые позволяют построить треугольник.

Для решения задачи используется «правило треугольника»: каждая сторона треугольника меньше суммы двух других сторон.

```
Program pr3;
Var
    a,b,c,d: integer;
Procedure Treug(x, y, z:integer);
Begin
    if (x<y+z) and (y<x+z) and (z<x+y)
            then writeln (‘Треугольник со сторонами  ‘,  x,  ‘,’, y,  ‘,’,  z,  ’можно построить’)
           else writeln(‘Треугольник со сторонами ’  x,  ‘,’,  y,  ‘,’,  z,  ‘нельзя построить ‘)
End;
Begin                                                    {основная программа}
Write('Введите a b c d: '); Readln(a,b,c,d);      {ввод исходных данных}
{вызываем процедуру и проверяем возможность построения треугольника со сторонами a,b,c}
     Treug(a,b,c);
{вызываем процедуру и проверяем возможность построения треугольника со сторонами a,b,d}
      Treug(a,b,d);
{вызываем процедуру и проверяем возможность построения треугольника со сторонами a,c,d}
      Treug(a,c,d);
{вызываем процедуру и проверяем возможность построения треугольника со сторонами b,c,d}
       Treug(b,c,d);
end.
```

Результат:
Введите a b c d: 1 3 4 5
Треугольник со сторонами 1,3,4 нельзя построить
Треугольник со сторонами 1,3,5 нельзя построить
Треугольник со сторонами 1,4,5 нельзя построить
Треугольник со сторонами 3,4,5 можно построить

Пример 4.

Упорядочить по возрастанию числа A, B, C, создав процедуру обмена значениями 2-х переменных.

```
Program pr4;
Var
  A, B, C: real;
Procedure Obmen(Var X,Y:real);
{X,Y являются и входными, и выходными  данными}
Var
    T: real;              { T –вспомогательная переменная}
Begin
     T:=X;
     X:=Y;
     Y:=T;
End;
begin            {основная программа}
 Write('A, B, C ='); Readln(A,B,C);                    {ввод исходных данных}
 If A>B then Obmen(A,B);            {если A больше B, то производим их обмен}
 If A>C then Obmen(A,C);            {если A больше C, то производим их обмен}
 If B>C then Obmen(B,C);            {если B больше C, то производим их обмен}
 Writeln('A=', A:5:2,'  B=', B:5:2,'  C=', C:5:2);          {вывод результата}
end.
```

Результат:

```
A, B, C = 6  18  3
A=  3.00  B=  6.00   C= 18.00
```

Пример 5.

Создать процедуру для вывода первых N членов арифметической прогрессии, заданной значением первого члена **a** и разностью **d**. Вывести первые 7 членов прогрессии 2, 5, … (a=2, d=3) и первые 10 членов прогрессии 20, 19, ….(a=20, d= -1)

```
Program pr5;
Procedure progres(a,d,n:integer);
Var
     i: integer;              {i – номер очередного члена прогрессии }
Begin
  for i:=1 to n do                              {перебираем i от 1 до n}
    begin
       write(a,' ');            {выводим очередной член прогрессии}
       a:=a+d;                  {вычисляем следующий член прогрессии}
    end;
End;
begin              {основная программа}
 {вызов процедуры для вывода 7 членов арифметической прогрессии 2, 5,…}
 progres(2,3,7);
  writeln;
```

```
   {вызов процедуры для вывода 10 членов прогрессии 20,19,…}
    progres(20,-1,10);
end.
```

Результат:

```
2  5  8  11  14  17  20
20  19  18  17  16  15  14  13  12  11
```

Пример 6

Выяснить, какие натуральные числа от 2 до 10 являются простыми, а какие - составными (создать процедуру, определяющую простым или составным является данное число).

Число называется *простым*, если оно не имеет делителей кроме 1 и самого себя. Если у него есть другие делители, то число – *составное*.

```
Program pr6;
Var
  i: integer;
Procedure prostoe(x:integer);
Var
    k,n: integer;    {n -  возможные делители числа,  k – количество делителей}
Begin
    k:=0;
    for n:=2 to x div 2 do    {перебираем числа от 2 до x div 2  для поиска делителей}
          if x mod n = 0                          {если n –делитель числа x}
                then k:=k+1;                    {увеличиваем количество делителей на 1}
    if k=0                                        {если у числа нет делителей}
         then writeln (x,' -  prostoe')          {то число простое}
        else writeln (x,' -  sostavnoe');        {иначе число составное}
End;
begin                                    {основная программа}
      for i:=2 to 10 do                                  {перебираем числа от 2 до 10}
           prostoe(i);              {вызываем процедуру для проверки очередного числа}
end.
```

Результат:

```
2 -  prostoe
3 -  prostoe
4 -  sostavnoe
5 -  prostoe
6 -  sostavnoe
7 -  prostoe
8 -  sostavnoe
9 -  sostavnoe
10 -  sostavnoe
```

Контрольные вопросы

1. Описание процедуры. Общая структура.
2. Как осуществляется вызов процедуры из основной программы?
3. Каким образом процедура передает результат в основную программу?
4. Что такое формальные и фактические параметры?
5. Какая взаимосвязь существует между формальными и фактическими параметрами?
6. Какие виды параметров могут быть указаны при описании процедуры или функции в ее заголовке?
7. Для чего используются параметры-значения?
8. Для чего используются параметры-переменные?

Задания для самостоятельной работы

1. Создать процедуру для вычисления периметра и площади квадрата по длине его стороны.
2. Создать процедуру для вычисления объема и площади поверхности куба по длине его ребра.
3. Создать процедуру для вычисления длины окружности и площади круга по заданному значению радиуса. ($L= 2\pi R$, $S=\pi R^2$)
4. Создать процедуру для вычисления периметра и площади прямоугольника по длинам его сторон.
5. Создать процедуру для вычисления периметра и площади треугольника по длинам его сторон ($S = \sqrt{p \cdot (p-a) \cdot (p-b) \cdot (p-c)}$, a, b, c – стороны треугольника, p – полупериметр)
6. Создать процедуру для нахождения корней квадратного уравнения по его коэффициентам **a,b,c.**
7. Вводятся A, B, C, D. Поменять местами A и C, B и D, создав процедуру обмена значениями 2-х переменных.
8. Создать процедуру для вывода первых N членов арифметической прогрессии, заданной первым членом **a** и разностью **d**. С помощью этой процедуры вывести 8 первых членов прогрессии 1, 4, … и 10 первых членов прогрессии 24, 22, ….
9. Создать процедуру для вывода первых N членов арифметической прогрессии, заданной первым членом **a** и разностью **d**. С помощью этой процедуры вывести 5 первых членов прогрессии 3, 7, … и 7 первых членов прогрессии 14, 11, ….
10. Создать процедуру для вывода первых N членов арифметической прогрессии, заданной первым членом **a** и разностью **d**. С помощью этой процедуры вывести 7 первых членов прогрессии 12, 10, … и 9 первых членов прогрессии 2, 5, ….
11. Создать процедуру для вывода и нахождения суммы первых N членов арифметической прогрессии, заданной формулой $\mathbf{a_n=2n+1.}$
12. Создать процедуру для вывода и нахождения суммы первых N членов арифметической прогрессии, заданной формулой $\mathbf{a_n=3n-2.}$
13. Создать процедуру для вывода и нахождения суммы первых N членов арифметической прогрессии, заданной формулой $\mathbf{a_n=5n-3.}$
14. Определить, какие из целых чисел от -3 до 8 являются решениями неравенства **(x-5)(x+1)>0** (создать процедуру, определяющую является или не является данное целое число решением неравенства (x-5)(x+1)>0)

15. Определить, какие из целых чисел от -10 до 2 являются решениями неравенства **(x+4)(x+1)<0** (создать процедуру, определяющую является или не является данное целое число решением неравенства (x+4)(x+1)<0)
16. Создать процедуру для вывода и подсчета суммы последовательных целых чисел от **a** до **b**. С помощью этой процедуры вывести и найти сумму целых чисел: 1) от 10 до 20; 2) от -5 до 15.
17. Создать процедуру для вывода таблицы значений функции **y=sin 3x + 1** для x от **a** до **b** с шагом **h**. Вывести две таблицы: а) для x от 1 до 2 с шагом 0.1 и б) для x от 5 до 10 с шагом 0.5
18. Создать процедуру для вывода таблицы значений функции **y=tg x + 1/x** для x от **a** до **b** с шагом **h**. Вывести две таблицы: а) для x от 2 до 4 с шагом 0.2 и б) для x от 1 до 10 с шагом 0.5
19. Создать процедуру для вывода таблицы значений функции y= $\mathbf{3x^2 + 1}$ для x от **a** до **b** с шагом **h**. Вывести две таблицы: а) для x от 1 до 2 с шагом 0.1 и б) для x от 2 до 5 с шагом 0.5
20. Создать процедуру для вывода таблицы значений функции $\mathbf{y=sin\ (x + 1)^2}$ для x от **a** до **b** с шагом **h**. Вывести две таблицы: а) для x от 2 до 3 с шагом 0.1 и б) для x от -2 до 3 с шагом 0.5

6. МАССИВЫ

Краткие теоретические сведения

С понятием "массив" приходится сталкиваться при решении научно-технических и экономических задач обработки совокупностей большого количества значений. В общем случае ***массив*** - это структурированный тип данных, состоящий из фиксированного числа элементов, имеющих один и тот же тип.

Тип элементов массива называется базовым. Число элементов массива фиксируется при описании и в процессе выполнения программы не меняется. Доступ к элементам массива осуществляется путем индексирования элементов массива. Тип индекса определяет границы изменения индекса. Для описания массива предназначено словосочетание ***array of*** (массив из).

Описание типа

Type <имя типа>=array [<тип индекса>] of <тип компонента>;
Var <идентификатор, …> : <имя типа>;

Описание может быть и без представления типа в разделе type:

Var <идентификатор>:array[<тип индекса>] of <тип компонента>;

Примеры описаний массивов:

```
Type
    Vektor =array [1..7] of integer;
    Massiv =array [1..20] of real;
```

```
Var
    A,B:vector; X: massiv;
```
То же в другом формате:
```
    Var   A, B : array [1..7] of integer;
          X: array [1..20] of real;
```
Если в описании массива задан один индекс, массив называется *одномерным*, если два индекса - *двумерным*, если n индексов — *n-мерным массивом*. Одномерный массив соответствует понятию линейной таблицы (вектора), двумерный - понятию прямоугольной таблицы (матрицы, набору векторов). Размерность ограничена только объемом памяти конкретного компьютера.

Описать двумерный массив можно двумя способами:
```
Var
  A: Array[1..20] of Array [1..30] of  Integer;
```
или
```
Var
  A : Array [1..20,1..30] Of Integer;
```

В обоих случаях описан двумерный массив, соответствующий таблице, состоящей из 20 строк и 30 столбцов. Отдельный элемент двумерного массива адресуется двумя индексами. Например, элемент, находящаяся в 5-й строке и 6-м столбце будет обозначаться A[5,6].

Элементы массива располагаются в памяти последовательно. Элементы с меньшими значениями индекса хранятся в более низких адресах памяти. Многомерные массивы располагаются таким образом, что самый правый индекс возрастает самым первым.

Например, если имеется массив:

A:array[1..5,1..5] of integer;

то в памяти элементы массива будут размещены по возрастанию адресов:

A[1,1] A[1,2] … A[1,5] A[2,1] A[2,2] … A[5,5]

Над всем массивом можно выполнять действия A=B, A<>B, A:=B.

Присваивать можно только массивы одинаковых типов. Обращение к отдельному элементу массива производится при помощи указания имени всего массива и в квадратных скобках – индекса конкретного элемента. Например: X[10] - элемент массива X с индексом 10.

Пример 1.

Сформировать массив из 20 целых чисел, вводимых с клавиатуры, и вывести элементы массива в обратном порядке.

```
Program PR1;
Var
    A : Array [1..20] Of Integer;    I : Integer;
BEGIN
    For I:=1 To 20 Do       {в  цикле перебираем индексы элементов массива}
        Readln(A[I]);       { вводим очередной элемент массива  с клавиатуры }
    For I:=20 Downto 1 Do       {Распечатываем массив в обратном порядке}
        Write(A[I],'   ')
END.
```

Для учебных целей удобнее использовать массивы, сформированные с помощью генератора случайных чисел. В языке Паскаль случайные числа, равномерно расположенными в интервале от 0 до 1, формирует функция **Random**. Для получения целых случайных чисел от 0 до N следует использовать функцию **Random(N+1).** Чтобы выбрать целое случайное число из интервала **[a, b]** можно применить выражение **Random(b-a+1)+a.**

Пример 2.

Сформировать и распечатать массив из 20 целых случайных чисел от 10 до 100. Найти сумму элементов массива.

```
Program pr3;
Var
    A : Array [1..20] Of Integer;  I, S : Integer;
BEGIN
    S:=0;  {обнуляем значение суммы}
    For I:=1 To N Do  begin
            A[I]:= Random(101)-50;     {формирование элементов массива}
            Write(A[I],'   ');                  {вывод  массива}
            S:=S+A[I];                  {суммирование элементов массива}
    end
END.
```

Пример 3.

В массиве хранятся цены на 10 видов мороженого. С помощью датчика случайных чисел заполнить массив целыми значениями, лежащими в диапазоне от 20 до 80 включительно. Определить порядковый номер самого дорогого мороженого.

```
Program pr3;
Var A:array[1..10] of integer;
   i, n, m: integer;  {m- максимальный элемент, n- его порядковый номер}
BEGIN
    for i:=1 to 10 do begin         {формирование и вывод элементов массива}
        a[i]:=random(18)+3;     write(a[i],' ');
    end; writeln;
    m:=a[1]; n:=1;      {выбираем 1-й  элемент в качестве максимального }
    for i:=2 to 10 do  {перебираем все элементы массива, начиная со 2-го}
      if  a[i]>m then begin m:=a[i]; n:=i end; {запоминаем  наибольший и его индекс}
    write('n=',n)
 END.
```

Пример 4.

В массиве хранится возраст 15 человек. С помощью датчика случайных чисел заполнить массив целыми значениями, лежащими в диапазоне от 16 до 30 включительно. Найти количество человек моложе 25 лет.

```
Program pr4;
```

```
Var A:array[1..15] of integer;
   i, k : integer;     {k - количество человек моложе 25 лет }
BEGIN
    for i:=1 to 15 do begin     {формирование и вывод элементов массива}
       a[i]:=random(15)+16;         write(a[i],' ');
    end;
    writeln;  k:=0;
    for i:=1 to 15 do
        if  a[i]<25 then k:=k+1; {подсчет  количества человек моложе 25 лет }
    write('k=',k)
END.
```

Пример 5.

Массив A[1..9] заполняется числами, вводимыми с клавиатуры. Найти среднее арифметическое элементов, которые больше 12 и меньше 25.

```
Program pr5;
Var A: array[1..9] of integer;
     i, k, s: integer;
BEGIN
    for i:=1 to 9 do begin   {формирование   массива}
        write('a[',i,']='); readln(a[i])
    end;
    writeln; k:=0; s:=0;
    for i:=1 to 9 do
         if  (a[i]<25) and (a[i]>12) then begin k:=k+1;  s:=s+ a[i]; end;
    if k>0  then writeln('srednee=',s/k:4:2) else writeln('Искомых элементов нет');
    readln;
END.
```

Пример 6.

Задан массив A[1..10]. Сформировать два массива, включая в первый четные элементы исходного массива, а во второй - нечетные.

```
Program Pr9;
Const n=10;
Var    a, x, y: array[1..n] of integer;
       i, k, m: integer; {k- количество элементов массива X,  m- количество элементов массива Y}
BEGIN
   randomize;
   writeln('Массив A:');
   for  i:=1  to  n  do   begin     {формирование и вывод элементов массива A}
            a[i] := random(51); Write(a[i]:4);
   end;
   writeln;    k:=0;  m:=0;
   for  i:=1  to  n  do
```

```
    {если очередной элемент массива A четный, то заносим его в массив X, иначе - в массив Y}
        if  a[i] mod 2=0  then  begin  k:=k+1;  x[k]:=a[i];  end
                                else  begin  m:=m+1;  y[m]:=a[i]; end;
   writeln('Массив X');  for i:=1 to k do  write(x[i]:4);        {вывод массива X}
   writeln;  writeln('Массив Y');   for i:=1 to m do  write(y[i]:4);    {вывод массива Y}
END.
```

Пример 7.

Найти сумму элементов целочисленной матрицы размером N*M.

```
Program  pr7;
   Const  n=2;  m=5;
Var  Mt: array [1..n,1..m]  of integer;
      J, S, I: integer;
BEGIN                {ввод элементов матрицы}
Writeln (‘введи элементы матрицы по строкам’);
     For  I:=1  to  n  do
        For j:=1  to  m  do  begin
             Write ('Mt [',  I, ’,’ ,  j,  ']= ');   Readln (Mt[i,j]);
        End;
{нахождение суммы элементов матрицы}
S:=0; For  I:=1  to  n  do
          For  j:=1  to  m  do
             S:=S+Mt[i,j];
        Writeln (‘сумма = ’, S);
END.
```

Пример 8.

Сформировать и вывести на экран в виде таблицы массив A[1..4,1..6], заполнив его целыми случайными числами из интервала [25,80].

```
Program pr8;
Var a: array[1..4,1..6] of integer;
   i, j: integer;
BEGIN
    randomize;          {формирование и вывод построчно элементов массива}
    for i:=1 to 4 do begin                {I – номер строки}
        for j:=1 to 6 do begin            {j – номер столбца}
            a[i,j]:=random(56)+25; write(a[i,j],' ');
        end;
        writeln;                          {переход на новую строку}
    end;
END.
```

Пример 9.

Дана целочисленная матрица размером N*M. Все отрицательные элементы заменить нулями, а положительные – единицами. Полученную матрицу вывести в виде таблицы.

```
Program pr9;
   Const  n=3;  m=3;
   Type  massiv=array [1..n,1..m]  of integer;
   Var   Mt: massiv;
         i, j: integer;
BEGIN              {ввод элемента матрицы}
    Writeln (‘введи элемент матрицы по строкам’);
     For  I:=1  to  n  do
          For  j:=1  to  m  do  begin
               Write ('Mt [', I, j, ']= ');  readln (Mt [i, j])
          End;
     For I:=1  to  n  do          {перебираем построчно элементы массива}
          For j:=1  to  m  do
              If Mt [i, j] <=0  then  Mt [i, j] :=0  Else  Mt[i, j]:=1;
           {вывод в виде таблицы}
     for I:=1  to n do   begin
          for j:=1  to  m  do  write (Mt [I, j]);
          writeln        {переходим на новую строку}
     end;
END.
```

Пример 10.

В массиве A[1..5,1..5] найти сумму элементов главной диагонали, кратных 5.

```
Program p10;
Var a: array[1..5,1..5] of integer;
      i, j, s: integer;
BEGIN
    randomize;     for  i:=1 to 5 do  begin     {формируем и выводим таблицу}
                          for   j:=1 to 5 do begin
                               a[i,j]:=random(80);   write(a[i,j],' ');
                           end;
                          writeln;
                       end;
       s:=0;         for i:=1 to 5 do   {перебираем построчно элементы массива}
                          for j:=1 to 5 do
{если элемент находится на главной диагонали и кратен 5, то суммируем }
                               if (i=j) and (a[i,j] mod 5=0) then s:=s+a[i,j];
                 writeln('s=',s)
END.
```

Пример 11

В двумерном массиве хранится информация о количестве студентов в каждой из трех групп каждого курса с первого по пятый (в 1-ой строке – информация о первом курсе, во 2-ой строке – о втором курсе и т. д.). Найти численность самой большой группы на 3 курсе. На каком курсе больше студентов, на первом или на пятом?

```
Program pr11;
Var   a: array[1..5,1..3] of integer;
```

```
      i, j, m, s1, s5: integer;
BEGIN
   randomize;              {формируем и выводим таблицу}
   for i:=1 to 5 do
          for j:=1 to 3 do begin
              a[i,j]:=random(10)+20;     write(a[i,j],' ');
          end;
         writeln
   end;
   m:=a[3,1];                        {находим наибольший элемент в 3-ей строке}
   for i:=1 to 3 do
      if a[3, i]>m then m:=a[3,i];
 writeln('численность самой большой группы на третьем курсе ', m);
    {находим сумму элементов 1-й строки  и сумму элементов 5-й строки}
 for j:=1 to 3 do  begin
       s1:=s1+a[1,i];
       s5:=s5+a[5,i]
end;
{сравниваем количество студентов на 1-м и на 5-м курсах}
if s1>s5 then writeln('студентов больше на 1 курсе')
          else  if s1<s5 then writeln('студентов больше на 5 курсе')
                        else writeln('количество студентов на 5-м и на 1-м курсах равно')
END.
```

Контрольные вопросы

1. Массивы. Основные понятия и определения.
2. Формат записи массивов. Описание одномерного и двумерного массивов.
3. Действия над массивами. Действия над элементами массивов.
4. Примеры описания и ввода-вывода линейного и двумерного массивов.

Задания для самостоятельной работы

1. Сформировать массив из 15 целых чисел, выбранных случайным образом из интервала [10, 90]. Поменять местами первый и минимальный элементы.

2. Задан одномерный массив A[1..20]. Найти минимальный элемент среди элементов массива с n-го по k-й (n и k вводятся с клавиатуры)

3. В заданном массиве поменять местами наибольший и наименьший элементы.

4. Задан одномерный массив A[1..20]. Найти сумму максимального и минимального элементов.

5. В массиве хранятся сведения об общей стоимости товаров, проданных фирмой за каждый день марта. Определить дни, в которые стоимость проданных товаров превысила среднюю ежедневную сумму продаж.

6. В массиве хранятся сведения об общей стоимости товаров, проданных фирмой за каждый день ноября. Определить дни, в которые стоимость проданных товаров превысила значение S.

7. В одномерном массиве хранятся сведения о стоимости товаров, проданных фирмой за каждый месяц года. Определить: а) месяц, в котором сумма продаж была мак-

симальна; б) номера месяцев, когда сумма продаж превысила среднюю за год; в) в каком полугодии работа фирмы была более эффективна.

8. В одномерном массиве хранится информация о зарплате 15 человек, работающих в отделе. Составить программу для определения: а) итоговой суммы по всему отделу; б) порядкового номера человека, получившего наименьшую зарплату; в) средней зарплаты по отделу.

9. В одномерном массиве хранятся сведения о стоимости товаров, проданных фирмой за каждый месяц года. Определить: а) общую сумму продаж за год; б) номера месяцев, когда сумма продаж превысила среднюю за год; в) в каком полугодии работа фирмы была более эффективна.

10. В одномерном массиве хранится информация о прибыли каждого их 8 магазинов фирмы «Луч» за месяц. Определить: а) общую прибыль фирмы; б)номера магазинов с максимальной и минимальной прибылью; в) среднее значение прибыли.

11. В одномерном массиве хранится информация о коммунальных платежах каждой из семей 20-квартирного дома за месяц. Определить: а) общую сумму платежей; б) номера квартир, которые не оплатили коммунальные услуги; в) номера квартир, платежи которых превысили заданное значение.

12. В одномерном массиве хранится информация об отчислениях на развитие предприятия за каждый месяц года. Определить: а) общую сумму отчислений; б) номера месяцев, когда отчислений не было; в) номера месяцев, когда сумма отчислений была максимальной.

13. В одномерном массиве хранится информация об отчислениях на благотворительность каждой из 15 фирм. Определить: а) общую сумму отчислений; б) номера фирм, которые перечислили сумму выше средней; в) номера фирм, перечисливших минимальную сумму.

14. В одномерном массиве хранится информация о количестве осадков, выпавших за каждый день сентября. Определить: а) в какие дни осадков не было; б) дни, когда количество осадков превысило среднее значение; в) общее количество осадков за месяц.

15. В одномерном массиве хранится информация о ценах на 20 видов товаров. Определить: а) цену самого дешевого товара и его порядковый номер; б) цену самого дорогого товара и его порядковый номер; в) номера товаров, цена которых превышает среднее значение;

16. В двухмерном массиве хранится информация о количестве учеников в каждом классе каждой параллели школы с первой по одиннадцатую (в первой строке – информация о классах первой параллели, во второй – второй параллели и т.д.). В каждой параллели школы имеются 4 класса. Определить: а) общее число учеников в параллели 5-х классов; б) самый большой по наполняемости класс в параллели 9-х классов и его порядковый номер.

17. Составить программу нахождения минимального элемента в каждом столбце и максимального в каждой строке квадратной матрицы.

18. Составить программу обмена местами максимального и минимального элементов главной диагонали матрицы.

19. Дана матрица N x N . Вывести на экран дисплея элементы той строки, сумма элементов которой максимальна.

20. Вывести на экран матрицу 6x6, элементами которой являются целые случайные числа из интервала [20,50]. Определить сумму минимальных элементов строк матрицы.

21. Вывести на экран матрицу 5x5, элементами которой являются целые случайные числа из интервала [19,49]. Найти сумму нечетных элементов каждого столбца.

22. Вывести на экран матрицу 4x6, элементами которой являются целые случайные числа из интервала [25,50]. Определить произведение максимальных элементов столбцов матрицы.

23. Вывести на экран матрицу 8x5, элементами которой являются целые случайные числа из интервала [20,40]. Определить номер строки, содержащей не менее 2-х чётных чисел.

24. Вывести на экран матрицу 3x6, элементами которой являются целые случайные числа из интервала [10,90]. Определить номер минимального элемента в каждой строке.

25. Вывести на экран матрицу 4x6. Найти сумму элементов каждой строки. Вывести результат в виде одномерного массива и найти его максимальный элемент.

26. Вывести на экран матрицу 10x8, элементами которой являются целые случайные числа из интервала [19,49]. Определить минимальный элемент в каждом столбце и выбрать из них максимальный.

27. Вывести на экран матрицу 4x8 , элементами которой являются целые случайные числа из интервала [-20,30]. Определить номер максимального элемента в каждой строке и номер минимального в каждом столбце.

28. Вывести на экран матрицу 8x5, элементами которой являются целые случайные числа из интервала [30,70]. Определить среднее арифметическое минимального и максимального элемента в каждом столбце.

29. Вывести на экран матрицу 5x5, элементами которой являются целые случайные числа из интервала [19,49]. Найти среднее арифметическое нечетных элементов каждого столбца.

30. Вывести на экран матрицу 8x5, элементами которой являются целые случайные числа из интервала [20,40]. Определить номер строки, содержащей не более 3-х чётных чисел.

7. ОБРАБОТКА СИМВОЛЬНЫХ И СТРОКОВЫХ ВЕЛИЧИН

Краткие теоретические сведения

В стандарте языка Паскаль описаны два типа переменных для литерных величин. Это - **String** и **Char**. Переменная типа Char может содержать в себе только один единственный символ, тип String предназначен для хранения строковых величин до 255 символов длиною.

При описании переменной этого типа можно указать максимальное число символов, которое можно занести в нее (это число не должно превышать 255). Например:

Var S : String[30]; *{максимальная длина строки S – 30 символов}*

St: String;]; *{максимальная длина строки St – 255 символов}*

При использовании строковой переменной, к каждому ее символу можно обратиться отдельно, указав его порядковый номер.

Пример 1.

Вывести посимвольно строку '*primer*'

```
Program pr1;
Var st: string[6];    i: integer;
BEGIN
      st:='primer';
      for i:=1 to 6 do  write(st[i], '   ');
END.
```

В результате работы данной программы на экран будут распечатаны следующие значения: ***p r i m e r***

Таким образом, первым символом *st[1]* в переменной *st* является буква *p*, вторым символом *st[2]* является *r* , третьим – *i* и т.д.

Две строковые величины можно объединять. Эта операция называется *конкатенацией* и обозначается знаком "+".

Например, результатом выполнения следующих команд:

```
R:='kadabra';
H:='abra';
S:=H+R;
```

в переменной S будет значение 'abrakadabra'.

Для конкатенации результат зависит от порядка операндов (в отличие от операции сложения). Следует помнить о том, какой максимальной длины может быть результирующая переменная, так как в случае превышения значением выражения числа, указанного после String в описании переменной, "лишние" символы в переменную не попадут.

Строковые величины можно сравнивать между собой. Сравнение строк происходит посимвольно. Большим из двух символов считается тот, код которого больше. Если равны первые символы, то сравнивается следующая пара до тех пор, пока не будет найдено различие. Если начало строк совпадает, а одна из них кончается раньше, то вторая автоматически называется большей.

Код символа в Паскале можно определить при помощи функции **Ord(C),** где C - либо непосредственно указанный символ, либо переменная символьного типа, либо один символ строковой переменной.

Есть и обратная функция, которая возвращает символ по известному коду. Это функция **Chr(N),** где N - выражение, приводящее к целому числу в интервале от 0 до 255 (возможные значения кода символа). Очевидно, что Chr(Ord(C))=C, Ord(Chr(N))=N.

Пример 2.

Вывести на экран кодовую таблицу:

```
Program Pr2;
Var   I : Byte;
BEGIN
      For I:=32 to 255 do Write(I:4, '-',Chr(I))
END.
```

Цикл в программе начинается с 32 потому, что символы с кодами от 0 до 31 являются управляющими и не имеют соответствующего графического представления.

В Паскале используются следующие процедуры и функции для обработки строковых величин:

Функция **Concat(S1[,S2,…,SN]): string** – сцепляет (объединяет) строки *S1, S2,…, SN*. Параметры, указанные в квадратных скобках, не являются обязательными.

Функция **Copy(S: string; Index, Count: integer): string** – копирует из строки *S* количество *Count* символов, начиная с символа с номером *Index*.

Функция **Length(S: string):byte** – определяет длину строки *S*.

Функция **Pos(SubS, S: string):byte** – отыскивает в строке *S* первое вхождение подстроки *SubS* и определяет номер позиции, с которой она начинается. Если подстрока не найдена, то выдается 0.

Функция **UpCase(C: char):char** – преобразует строчную латинскую букву в прописную. Любые другие символы возвращаются без преобразования.

Процедура **Delete(var S: string; Index, Count: integer***)* – удаляет количество *Count* символов из строки *S*, начиная с символа с номером *Index*.

Процедура **Insert(SubS: string; var S: string; Index: integer***)* – вставляет подстроку *SubS* в строку *S*, начиная с символа с номером *Index*.

Процедура **Str(X [:Width [:Decimals]]; Var S: string)** – преобразует число *X* в строку символов *S*. Параметры Width и Decimals задают формат преобразования (общую ширину поля и количество символов в дробной части соответственно).

Процедура **Val(S: String; var X; var Code: integer)** – преобразует строковое значение *S* в его численное представление *X* типа Real или Integer. Параметр Code содержит признак ошибки преобразования (0 – нет ошибки),

Пример 3.

Определить длину введенной пользователем строковой величины.

```
Program Pr3;
Var  S : String;
BEGIN
     Writeln('Введите последовательность символов');  Readln(S);
     Writeln('Вы ввели строку из ',Length(S), ' символов')
END.
```

Пример 4.

Задан список из 6 слов. Определить, сколько слов списка начинается на букву «п».

```
Program Pr4;
Var   s: string[20];   i,k:integer;
BEGIN
     k:=0; for i:=1 to 6 do  begin
                writeln('Введите слово');     readln(s);
                if s[1]='п' then k:=k+1;
```

```
        end;
      writeln(k);
END.
```

Пример 5.

Определить, является ли введенная строка "перевертышем".

Перевертышем называется такая строка, которая одинаково читается с начала и с конца. Например, "казак" и "потоп" - перевертыши, "канат" - не перевертыш".

Поступим следующим образом: из введенной строки сформируем другую строку из символов первой, записанных в обратном порядке, затем сравним первую строку со второй; если они окажутся равны, то ответ положительный, иначе - отрицательный. Естественно, предложенный способ решения не является единственно возможным.

```
Program Pr5;
Var
   S,B : String;     I : Byte;
BEGIN
   Writeln('Введите строку'); Readln(S);
   B:='';                {Переменной B присваиваем значение "пустая строка"}
   For I:=1 to Length(S) do              {перебираем символы строки}
        B:=S[I]+B; {Символы строки S присоединяются к переменной B слева}
   If B=S Then Writeln('Перевертыш') Else Writeln('Не перевертыш')
END.
```

Пример 6.

Найти сумму цифр введенного натурального числа.

```
Program Pr6;
Var S : String;  I,X,A,C : Integer;
BEGIN
      Writeln('Введите натуральное число');
      Readln(S);              {Число вводится в строковую переменную}
      A:=0;
      For I:=1 To Length(S) Do Begin
            Val(S[I],X,C);        {Цифра преобразуется  в число}
            A:=A+X                    {Числа суммируются}
      End;
      Writeln('Сумма цифр равна ',A)
END.
```

Пример 7.

Во введенной строке заменить все вхождения подстроки 'ABC' на ‘KLMNO’.

```
Program Pr7;
Var S : String; A : Byte;
BEGIN
  Writeln('Введите строку'); Readln(S);
  While Pos('ABC',S)<>0 Do {пока подстрока 'ABC' содержится в строке S }
  Begin
    A:= Pos('ABC',S); {определяем позицию вхождения подстроки 'ABC' в строку S}
```

```
    Delete(S,A,3);  {удаляем из  строки S 3 символа, начиная с позиции A }
    Insert('KLMNO',S,A) {вставляем в строку S на это место подстроку 'KLMNO'}
End;
Writeln(S)
END.
```

Пример 8.

Во введенной строке подсчитать процент вхождения символа ‘o’ и символа ‘e’ (создать функцию для определения процентного содержания заданного символа в строке)

```
Program Pr8;
{функция, определяющая  процент вхождения символа  c в строку st}
Function proc_char(st:string;c:char):real;
Var  k, n, I:integer;   {k –количество заданных символов,   n – длина строки,
                         I -  номер очередного символа строки}
Begin
   n:=length(st);                    {вычисляем длину строки}
   k:=0; for I:=1 to n do            {перебираем символы строки}
      if st[I]=c then k:=k+1;     {если очередной символ строки равен заданному
                                    символу, то увеличиваем счетчик k}
   proc_char:=k/n*100;              {определяем процент заданных символов}
End;
Var st: string;
   p1,p2: real;
BEGIN                                {основная программа}
  write('st=');readln(st);                {вводим строку}
  p1:=proc_char(st,'o');        {определяем процентное содержание символа ‘o’}
  p2:=proc_char(st,'e');        { определяем процентное содержание символа ‘e’}
  writeln('o-', p1:2:1,'%');              {вывод результата}
   writeln('e-', p2:2:1,'%');
END.
```

Пример 9.

Задан список из 8 слов. Найти самое короткое слово в списке. Если таких слов несколько, то распечатать их в один столбец.

Решение поставленной задачи сводится к нескольким этапам: ввести список слов в виде массива строковых переменных; подсчитать длину каждой строки; определить наименьшую из длин; распечатать те строки массива, длина которых совпадает с наименьшей.

```
Program Pr9;
Var  s: array[1..8] of string[20];
     n: array[1..8] of integer;
     i, min: integer;
BEGIN
       for i:=1 to 8 do  begin
            writeln('Введите слово');     readln(s[i]);
            n[i]:=length(s[i]);
```

```
    end;
    min:=n[1];   for i:=2 to 8 do   if min>n[i] then min:=n[i];
    for i:=1 to 8 do   if n[i]=min then writeln(s[i]);
END.
```

В данной программе для обозначения слов используется массив строковых переменных *s*, для обозначений соответствующих им длин слов – целочисленный массив *n*. Наименьший элемент массива *n* хранится в виде переменной *min*.

Пример 10.

Задана строка из двух слов, разделенных пробелом. Поменять в данной строке слова местами.

```
Program Pr10;
Var s: string[40];
    s1,s2: string[20];
    i,n: integer;
BEGIN
    writeln('Введите строку');
    readln(s);
    n:=length(s);   i:=pos('  ', s);
     s1:=copy(s, 1, i);   s2:=copy(s, i+1, n-i);   s:=s2 + '  ' + s1;
     writeln(s);
END.
```

В данной программе переменная *s* служит для обозначения строки из двух слов, переменные *s1* и *s2* – для обозначения первого и второго слова строки соответственно. Алгоритм решения заключается в том, что сначала с помощью функции *Pos* выясняется, каким по счету символом в строке *s* находится пробел. Далее в строке выделяются две подстроки: одна включает все символы левее пробела (первое слово), другая – правее пробела (второе слово). В конце программы слова сцепляются в обратном порядке с использованием разделяющего их пробела.

Контрольные вопросы

1. Понятие строки. Описание строкового типа данных.
2. Представление строки в Паскале. Обращение к символу строки. Длина строки.
3. Стандартные строковые процедуры и функции в языке Паскаль.
4. Понятие строковых выражений.
5. Операции со строковыми данными.
6. Примеры программ работы со строками.

Задания для самостоятельной работы

1. В заданном тексте удалить часть текста, заключенную в скобки (вместе со скобками).
2. Сколько раз в тексте встречается заданное слово (слова разделены пробелами).
3. В тексте вставить между словами вместо одного пробела запятую и пробел.
4. Определить, какой процент слов в тексте начинается на букву К. Слова разделены пробелами.

5. Написать программу, исключающую из символьной строки все цифры.

6. Определить, содержит ли данный текст символы, отличные от букв и пробела.

7. Преобразовать введенную строку так, чтобы из нее были удалены буквы с ASCII - кодами от 70 до 75.

8. Определить, имеются ли во введенной строке следующие подряд две "4".

9. Введенную строку букв и цифр преобразовать так, чтобы после каждой цифры следовал пробел.

10. Изменить введенную строку так, чтобы каждая из цифр увеличилась на 1 (9 заменить 0).

11. Преобразовать введенную строку так, чтобы сначала были расположены цифры, потом буквы.

12. Введена строка маленьких латинских букв. Преобразовать ее, превратив маленькие буквы в большие.

13. Зашифровать введенное слово, сместив все буквы на 1 позицию (последняя становиться первой).

14. Введенную строку цифр вывести, расположив в каждой подстроке по 5 цифр.

15. Преобразовать буквы введенной строки так, чтобы их ASCII-коды увеличились на 3.

16. Введены 3 строки. Подсчитать сумму цифр, кратных 3, в каждой из них (создать функцию, подсчитывающую сумму цифр, кратных 3, в строке).

17. Даны 2 строки. Определить сумму цифр в каждой из них (создать функцию, подсчитывающую сумму цифр)

18. Создать функцию пользователя, определяющую количество символов введенной строки, ASCII-коды которых >=70.

19. Создать процедуру, позволяющую изменить введенную строку, добавив справа заданное количество заданных символов.

20. Создать процедуру, позволяющую из заданной строки удалить пробелы.

21. Создать процедуру, позволяющую записывать введенное слово в зеркальном отображении.

22. Создать процедуру, которая позволяет введенный текст разбить на строки по k символов.

23. Создать процедуру, которая позволит во введенной строке через каждые n символов вставить k пробелов.

24. Создать процедуру, которая в заданном тексте заменяет слово A1 на слово A2 (длины слов не совпадают).

25. Создать процедуру, которая во введенном слове заменяет один символ другим.

26. Создать процедуру, которая в тексте убирает лишние пробелы между словами, оставив по одному.

27. Вводятся три строки. Зашифровать каждую из них, заменив все буквы "c" на "o"(создать процедуру, заменяющую в заданной строке один символ другим)

28. Преобразовать три введенные строки, чтобы после каждой цифры следовал символ ‘!’ (создать процедуру, вставляющую пробел после каждой цифры в строке)

29. Дано предложение. Все пробелы в нем заменить на символ "_"(создать соответствующую процедуру).

30. Дано предложение. Удалить из него буквы «р» и «с»(создать процедуру, удаляющую из строки заданный символ).

Дополнительные задания

1. В старояпонском календаре был принят двенадцатилетний цикл. Годы внутри цикла носили названия животных: крысы, коровы, тигра, зайца, дракона, змеи, лошади, овцы, обезьяны, петуха, собаки, свиньи. Напишите программу, которая позволяет ввести номер года нашей эры и печатает его название по старояпонскому календарю. (1996 г. – начало очередного цикла).

2. Даны координаты (как целые от 1 до 8) двух полей шахматной доски. Определить, может ли ферзь за один ход перейти с одного из этих полей на другое. Ответ вывести в форме «Да» / «Нет»

3. Составить программу, запрашивающую с клавиатуры два натуральных числа A и B - стороны прямоугольника - и определяющую на сколько квадратов его можно разрезать, отрезая каждый раз квадрат максимальной площади с целой длиной стороны. Вывести количество и размер всех квадратов.

Например, при A = 20, B = 10 ответ: 2 со стороной 10;

при A = 20, B = 15 ответ: 1 со стороной 15, 3 со стороной 5.

4. Определить, все простые числа от 2 до заданного натурального числа N.

5. В 1626 году индейцы продали остров Манхеттен за 20 долларов. Если бы эти деньги были помещены в банк на текущий счет и ежегодный прирост составлял бы 4%, какова была бы стоимость капитала в 2016 году?

6. Занести в массив карту расположения кораблей в игре "Морской бой" и смоделировать игру.

7. Чего больше: всех возможных трехзначных чисел, записанных цифрами 2, 3, 4, 5 или всех четырехзначных чисел, записанных цифрами 1, 3, 7, 8, 9? Подсчет производить по формуле размещений:

$$A_n^k = \frac{n!}{(n-k)!}$$

8. Сколькими способами можно отобрать команду в составе 5 человек из 8 кандидатов; из 10 кандидатов; из 11 кандидатов? Подсчет количества способов отбора оформить по формуле сочетаний

$$C_n^k = \frac{n!}{k!(n-k)!}$$

9. Приписать к числу 523 три такие цифры справа, чтобы полученное число делилось на 7, 8 и 9.

10.Найти трехзначное число abc, для которого a!+b!+c! = abc.

11. Найти все трехзначные числа, которые являются полными квадратами и записываются четными цифрами.

12.Найти четырехзначное число, являющееся точным квадратом, у которого первые две цифры одинаковые и две последние тоже одинаковые.

13.Найти трехзначное число, квадрат которого оканчивается тремя одинаковыми цифрами, отличными от 0.

14.Приписать к числу 523 три такие цифры справа, чтобы полученное число делилось на 7, 8, 9.

15.Квадрат трехзначного числа оканчивается тремя цифрами, которые как раз составляют взятое число. Найти все такие числа.

16.В трехзначном числе, все цифры которого нечетны, зачеркнули среднюю цифру. Оказалось, что полученное двузначное число является делителем исходного числа. Найти все такие трехзначные числа.

17.Четырехзначное число, а также число, записанное теми же цифрами в обратном порядке, оба являются точными квадратами. Найти эти числа.

18.Приписать к числу 999 слева три такие цифры, чтобы полученное шестизначное число делилось на 13, 17 и 19.

19.Найти все натуральные числа, не превосходящие 600, у которых сумма цифр является делителем самого числа.

20.Найти все трехзначные числа, которые являются полными квадратами и записываются тремя нечетными цифрами.

21.Найти четырехзначное число, равное квадрату суммы двух двузначных чисел, образованных двумя первыми и двумя последними цифрами числа.

22.Рассмотрим произвольное натуральное число и найдем сумму его цифр, затем сумму цифр полученного числа и так далее, пока не получим однозначное число. Назовем это число цифровым корнем. Напишите программу, запрашивающую с клавиатуры натуральное число N и вычисляющую его цифровой корень.

23.Натуральное число называется совершенным, если оно равно сумме всех своих делителей, за исключением самого себя. Например, число 6 - совершенное, так как 1+2+3=6, а 8 не является совершенным (1+2+4 не равно 8). Составить программу, запрашивающую с клавиатуры натуральное число и печатающую все совершенные числа, не превышающие введенного числа.

24.Составить программу для перевода заданного натурального числа N из десятичной системы счисления в двоичную.

25.Составить программу для перевода заданного натурального числа N из десятичной системы счисления в троичную.

26.Составить программу для перевода заданного натурального числа N из десятичной системы счисления в 8-ричную.

27.Составить программу, содержащую функцию вычисления экспоненты в виде бесконечного ряда с точностью 10^{-9}.

$$\exp(x) = 1 + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^N}{N!} + \ldots$$

В основной программе организовать вычисление этого ряда для двух значений x, запрашиваемых с клавиатуры, и проверку получаемых результатов путем сравнения с системной функцией EXP(X).

28. Составить программу, содержащую функцию вычисления косинуса в виде бесконечного ряда с точностью 10^{-7}.

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots + (-1)^N \frac{x^{2N}}{(2N)!} + \ldots$$

В основной программе организовать вычисление этого ряда для двух значений x, запрашиваемых с клавиатуры, и проверку получаемых результатов путем сравнения с системной функцией COS(X).

29. Составить программу, содержащую функцию вычисления синуса в виде бесконечного ряда с точностью 10^{-8}.

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots + (-1)^N \frac{x^{2N+1}}{(2N+1)!} + ..$$

В основной программе организовать вычисление этого ряда для двух значений x, запрашиваемых с клавиатуры, и проверку получаемых результатов путем сравнения с системной функцией SIN(X).

30. Составить программу, содержащую функцию вычисления гиперболического синуса в виде бесконечного ряда с точностью 10^{-8}.

$$sh(x) = \frac{e^X - e^{-X}}{2} = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \ldots + \frac{x^{2N+1}}{(2N+1)!} + ..$$

В основной программе организовать вычисление этого ряда для двух значений x, запрашиваемых с клавиатуры, и проверку получаемых результатов путем сравнения с системной функцией (exp(X)-exp(-X))/2.

31.Составить программу расчета роста по месяцам в течение одного года банковского вклада. Программа запрашивает с защитой от неверного ввода данных следующую информацию:

- начальный размер вклада (2000…20000),
- размер процентной ставки по вкладу (1% … 3% в месяц).

Вывести таблицу роста вклада по месяцам, а также определить количества месяцев, необходимые для роста вклада в полтора раза.

ЛИТЕРАТУРА

1. Бабенко Т. А., Бельченко В.Е., Козырева Г.Ф Практикум по курсу «Основы программирования»: Учебно-методическое пособие для студентов, обучающихся по специальности "Информатика". - Армавир, 2006 г.
2. Бабенко Т. А., Бельченко В.Е., Козырева Г.Ф Практикум по курсу «Основы программирования». Часть 2. Учебно-методическое пособие для студентов, обучающихся по специальности "Информатика". - Армавир, отпечатано в типографии Симакова А.А.,2008 – 48 с.
3. Безручко, В.Т. Информатика (курс лекций): учеб. пособ. / В.Т. Безручко.- М.: ИНФРА-М, 2013.- 432 с.
4. Бабенко Т.А. и др. Сборник задач по курсу Основы программирования: уч.метод.пособие / Бабенко Т.А., Бельченко В.Е., Козырева Г.Ф. - Армавир. - Ч. 1. - 2004- 40 с.
5. Большаков В.А. Информатика [Электронный ресурс]: лабораторный практикум по программированию на Турбо-Паскале/ Большаков В.А., Воронов Г.И., Савватеева Л.А.— Электрон. текстовые данные.— СПб.: Российский государственный гидрометеорологический университет, 2002.— 190 с.— Режим доступа: http://www.iprbookshop.ru/14906.— ЭБС «IPRbooks»
6. Давыдова Н.А. Программирование [Электронный ресурс]: учебное пособие/ Давыдова Н.А., Боровская Е.В.— Электрон. текстовые данные.— М.: БИНОМ. Лаборатория знаний, 2012.— 238 с.— Режим доступа: http://www.iprbookshop.ru/6485.— ЭБС «IPRbooks»
7. Златопольский Д.М. Программирование. Типовые задачи, алгоритмы, методы [Электронный ресурс]/ Златопольский Д.М.— Электрон. текстовые данные.— М.: БИНОМ. Лаборатория знаний, 2012.— 223 с.— Режим доступа: http://www.iprbookshop.ru/12264.— ЭБС «IPRbooks»
8. Задачник PascalABC http://interacia.net/index.php/2011-02-15-18-33-42/begin40.html
9. Пособие для учащихся «Основы программирования на Паскаль АВС» http://gigabaza.ru/download/64205.html