РАБОЧАЯ ТЕТРАДЬ ПО
ПРОГРАММИРОВАНИЮ НА ЯЗЫКЕ

# PASCAL

ДЛЯ ОБУЧАЮЩИХСЯ 8-9 КЛАССОВ

# ПРЕДИСЛОВИЕ

Рабочая тетрадь по программированию на языке Pascal предназначена для обучающихся 8 - 9 классов и построена в соответствии с темами, посвященными программированию на языке Pascal, в учебниках по информатике Босовой Л.Л. Содержание тетради направлено на закрепление, систематизацию, воспроизведение и практическое применение изучаемого материала.

Тетрадь содержит 7 разделов, названия которых совпадают с названиями разделов в учебниках по информатике. Каждый раздел состоит из двух частей: теоретической и практической.

В теоретической части размещена главная информация по теме: важные определения, понятия, описания и правила. В практической части представлены различные задания для закрепления пройденного материала.

В конце тетради расположен Словарь, который содержит информацию обо всех используемых в курсе программирования 8 - 9 классов служебных словах, типах данных, функциях и операторах. Информация в Словаре включает в себя пояснение к слову и в некоторых случаях пример использования.

# УСЛОВНЫЕ ОБОЗНАЧЕНИЯ

 - теоретическая часть;

 - практическая часть.

# ОГЛАВЛЕНИЕ

# 1. ОБЩИЕ СВЕДЕНИЯ О ЯЗЫКЕ ПРОГРАММИРОВАНИЯ PASCAL

## 1.1 Алфавит и словарь языка

 **Алфавит языка -** набор допустимых символов, которые можно использовать для записи программы.

- латинские прописные и строчные буквы (A, B, C, …, X, Y, Z, a, b, c, …, x, y, z);
- арабские цифры (0, 1, 2, 3, 4, 5, 6, 7, 8, 9);
- специальные символы( _ , . ( ) [ ] { } + - * / и т.д.)

**Неделимые элементы**

| | |
|---|---|
| **:=** | оператор присваивания |
| **>=**<br>**<=**<br>**<>** | ≥<br>≤<br>≠ |
| **(***<br>**)** | начало комментария<br>конец комментария |

 **Задания**

1. Выберите правильные имена

▫ q24 ▫ Ш ▫ d ▫ oshibka

▫ wu_kvadrat ▫ cena s nalogom ▫ flower&picture

2. Воспользуйтесь словарем в конце тетради и заполните таблицу.

| Служебное слово, функция или оператор | Назначение |
|---|---|
| **program** | |
| **begin** | |
| **end** | |
| **mod** | |
| **div** | |
| **/** | |

3. Запишите математическое выражение в виде, верном для языка Pascal (символы для математических операций найдите в словаре в конце тетради).

| **Математическое выражение** | **На языке Pascal** |
|---|---|
| $2x$ | `2 * x` |
| $x: 6$ | `x / 6` |
| $x + 4y$ | |
| $10 - 4x$ | |
| $2x - \frac{3}{y}$ | |
| $\frac{x+y}{x-y}$ | |
| $2 + \frac{x+y}{x-y}$ | |

4. Заполните таблицу математических функций языка Pascal.

| **Математическая функция** | **Функция языка Pascal** |
|---|---|
| Модуль числа | |
| Синус | |
| Косинус | |
| Квадрат числа | |
| Квадратный корень | |

5. Запишите математическое выражение в виде, верном для языка Pascal.

| **Математическое выражение** | **На языке Pascal** |
|---|---|
| $sin\, x + cos\, x$ | |
| $\|x - 5\| - \|x + 2\| - 71$ | |
| $x^2 + 4x - sin\, x$ | |

| | |
|---|---|
| $sin\,2x\ + \sqrt{57y} \cdot \lvert x - 1 \rvert$ | |
| $sin^2 x + cos^2 x$ | |
| $\sqrt{(p-a)^2 + (p-b)^2}$ | |
| $sin\,x\ - 10x\rvert$ | |

## 1.2 Типы данных, используемые в языке Pascal

Типы в Pascal подразделяются на простые, строковые, структурированные, типы указателей, процедурные и классовые типы. К **простым** относятся целые и вещественные типы, логический, символьный, перечислимый и диапазонный тип. **Структурированные** типы образованы массивами, записями, множествами и файлами.

**Основные типы данных в Pascal**

**`integer`** — целочисленный: `1, 2, 3, 0, -45, …`

**`real`** — с плавающей запятой (вещественные числа): `1.0, 15.67, -0.12754, …`

**`string`** — строковый: `'abcd', 'hello', 'mmmm', …`

**`char`** — символьный: `'a', 'b', 'c', …, 'б', 'в', 'г', …, '1', '2', '4.56', …`

**`boolean`** — логический тип данных (принимает два значения: **`true`** или **`false`**).

*Ознакомьтесь с типами данных в словаре, представленном в конце тетради)*

**Задания**

1. Сопоставьте значения с подходящими типами данных.

| Значение | Тип |
|---|---|
| 1 | |
| 1.0 | |
| 5 | integer |
| 4.7 | |
| -5 | |
| -78.47 | real |
| 17 | |
| 0 | |

2. Сопоставьте значения с подходящими типами данных.

| Значение | Тип |
|---|---|
| 'a' | |
| 'aa' | |
| 'abc' | string |
| 'char' | |
| 'привет' | |
| '65' | char |
| 'x' | |
| '9' | |

3. Сопоставьте значения с подходящими типами данных.

| Значение | Тип |
|---|---|
| 1 | integer |
| 'Be strong!' | |
| false | real |
| '1' | |
| 1.783 | string |
| 'g' | |
| -1 | char |
| true | |
| 'Не сдавайся' | boolean |
| -1.0 | |

4. Выберите подходящий тип для значения (типы данных представлены в словаре в конце тетради)

| Значение | Тип |
|---|---|
| 2 | |
| 13.5 | |
| 'a' | |
| 'строка' | |
| false | |
| 0.645 | |
| 'c' | |

| | |
|---|---|
| 'char' | |
| 125 | |
| 'true' | |

5. Определите тип аргумента для оператора

| Оператор | Тип аргумента |
|---|---|
| **+** | |
| **-** | |
| **/** | |
| ***** | |
| **div** | |
| **mod** | |

## 1.3 Структура программы в языке Pascal

 В программе, записанной на языке Pascal, можно выделить:

1. блок программы
2. блок описания переменных и констант
3. программный блок

| | |
|---|---|
| **program** <имя программы>; | **заголовок программы** |
| **const** <список постоянных значений>;<br>**var** <описание используемых переменных>; | **блок описания переменных и констант** |
| **begin**<br>  <оператор 1>;<br>  <оператор 2>;<br>  ...<br>  <оператор n><br>**end.** | **блок операторов (программный блок)** |

Переменные могут быть описаны в разделе описаний, а также непосредственно внутри любого блока **begin/end.**

Раздел описания переменных начинается с ключевого слова **var**, после которого следуют элементы описания вида

```
список имен: тип;
```

или

```
имя: тип := выражение;
```

или

```
имя: тип = выражение; // для совместимости с Delphi
```

или

```
имя := выражение;
```

Имена в списке перечисляются через запятую. Например:

```
var
  a,b,c: integer;
  d: real := 3.7;
  s := 'PascalABC forever';
  al := new List<integer>;
  p1 := 1;
```

**Задание**

1. Из каких основных элементов состоит программа на языке Pascal?
   - ▢ Заголовок программы, блок описания переменных и программный блок
   - ▢ Заголовок программы, блок описания действий
   - ▢ Заголовок программы, блок управления и блок помощи
2. Установите соответствие между разделами программы и служебными словами с которых они начинаются.

| | |
|---|---|
| Заголовок программы | **var** |
| Раздел описания переменных | **program** |
| Программный блок | **begin** |

3. После какого знака указывается тип переменных одного типа?
   - ▢ После дефиса
   - ▢ После точки с запятой
   - ▢ После двоеточия
4. Какими словами начинается и заканчивается программный блок?
   - ▢ Начинается со слова **begin**, заканчивается словом **finish**
   - ▢ Начинается со слова **start**, заканчивается словом **end** с точкой
   - ▢ Начинается со слова **begin**, заканчивается словом **end** с точкой
5. Подпишите блоки программ

| № | Программа | Блок |
|---|---|---|
| 1. | `program skolkobukv;` | |
| | `var s:string;`<br>`  r:real;`<br>`  i,j,n:integer;` | |

| | | |
|---|---|---|
| | begin<br>r:=0;<br>readln(s);<br>for i:=1 to length(s) do begin<br>n:=0;<br>for j:=1 to length(s) do begin<br>if s[i]=s[j] then inc(n);<br>end;<br>r:=r+1/n;<br>end;<br>writeln('количество различных букв = ', r:1:0);<br>end. | |
| 2. | program prostiechisla; | |
| | const LIMIT = 500;<br>var i,j,lim : word; | |
| | begin<br>writeln;<br>for i:=1 to LIMIT do begin<br>j:=2; lim:=round(sqrt(i));<br>while (i mod j <> 0) and (j <= lim) do inc( j );<br>if (j > lim) then write( i,' ' );<br>end;<br>end. | |
| 3. | program summamassiva; | |
| | const LIMIT = 500;<br>var i,j,lim : word;<br>s:longint;<br>i:integer; | |
| | begin<br>writeln('введите 10 элементов массива');<br>s:=0;<br>for i:=1 to 10 do begin<br>readln( a[i] );<br>s:=s+a[i];<br>end;<br>writeln( 'Сумма элементов массива = ', s );<br>end. | |

## 1.4 Оператор присваивания

 Оператор присваивания имеет вид:

**переменная := выражение;**

Символ **:=** называется *знаком присваивания*. Тип выражения должен совпадать с типом переменной.

Оператор присваивания заменяет текущее значение переменной значением выражения.

Например:

```
i := i + 1; // увеличивает значение переменной i на 1
```

**Задания**

1. Какая из записей оператора присваивания на языке Pascal правильная?

   ▢ `переменная = значение;` ▢ `переменная := значение;`

   ▢ `переменная : значение;` ▢ `значение := переменная;`

2. Как называется оператор, который придаёт ячейке оперативной памяти, которой является переменная, определённое значение.

   ▢ Условный оператор ▢ Оператор цикла

   ▢ Оператор присваивания ▢ Оператор подачи

3. Заполните пропуски.

   Процесс выполнения операторов присваивания:

   ```
   a := 17;
   n := 34;
   x := a * n
   ```

   При выполнении оператора **`a := 17`** в ячейку оперативной памяти компьютера с именем **`a`** заносится значение _____; при выполнении оператора **`n := 34`** в ячейку оперативной памяти компьютера с именем **`n`** заносится значение _____. При выполнении оператора **`x := a*n`** значения ячеек оперативной памяти с именами **`a`** и **`n`** переносятся в процессор, где над ними выполняется операция умножения. Полученный результат заносится в ячейку оперативной памяти с именем _____.

4. Какие из операторов присваивания не будет работать в программе с разделом описания переменных:

   ```
   var s, n: real;
   d: integer;
   str: string;?
   ```

   ▢ `s:=n;` ▢ `str:=3;` ▢ `d:=n;` ▢ `n:=3,5;`

5. Переменная **`x`** объявлена в программе следующим образом:

   ```
   var x: integer
   ```

   Определите, какие операторы присваивания сработают без ошибок:

   ```
   1) x := 10;     2) x := 'a';     3) x := -4;     4) x := 1.4;
   5) x := '10';   6) x := 7+2;     7) x := 4*23;   8) x := 8/2;
   ```

6. Если бы в предыдущем задании переменная **`x`** была бы объявлена следующим образом:

   ```
   var x: real,
   ```

   то какие ещё варианты ответов были бы верны?

   Ответ: _____

7. Определите тип переменной **`x`** для оставшихся вариантов ответа:

   _____

   _____

8. Определите значение переменной **a** после выполнения данного алгоритма:

```
a := 5;
b := 6;
b := 5 + a * b;
a := b – 6 * a;
```

**Ответ: _____**

9. Определите значение переменной a после выполнения алгоритма:

```
a := –12;
b := 14 – a / 2;
b := (b – a) / 8;
a := b * 2 + 6;
```

**Ответ: _____**

# 2. ОРГАНИЗАЦИЯ ВВОДА И ВЫВОДА ДАННЫХ

## 2.1 Вывод данных

 Вывод данных на экран и в файл в языке программирования Pascal осуществляется с помощью процедур

**write(A1,A2,...AK)**

**writeln(A1,A2,...AK)**

Первый из этих операторов производит вывод значений переменных A1, A2,...,AK в строку экрана. Второй оператор, в отличие от первого, не только производит вывод данных на экран, но и делает переход к началу следующей экранной строки. Если процедура writeln используется без параметров, то она просто производит пропуск строки и переход к началу следующей строки.

Переменные, составляющие список вывода, могут относиться к целому, действительному, символьному или булевскому типам. В качестве элемента списка вывода кроме имен переменных могут использоваться выражения и строки.

 **Задания**

1. При записи оператора вывода на языке Pascal используется служебное слово:
   ▫ `read` ▫ `begin` ▫ `var` ▫ `write`
2. Оператор `write ('fox');` выводит на экран _____.
3. Запишите результат выполнения оператора вывода `write (3,13,29,200);`

   _____

4. Сопоставьте оператор и результат его выполнения

| | |
|---|---|
| `write (a)` | вывод на экран буквы "`a`" |
| `write ('a')` | вывод на экран текста "`a=`", а затем значение переменной "`a`" |
| `write ('a=', a)` | вывод значения переменной "`a`" |

5. Какой тип имеет переменная g, если после выполнения оператора `write (g)` на экран было выведено следующее: `1250`?
   ▫ `boolean` ▫ `shortint` ▫ `нет верного ответа` ▫ `integer`
6. Что означает окончание "ln", добавленное к оператору write?
   ▫ Каждое значение выводится в отдельной строке.
   ▫ По окончании работы оператора курсор переместиться на следующую строку.
   ▫ Каждый символ выводится в отдельной строке

▫ Выводится только первое из указанных значений

7. Дана программа.

```
begin
      write ('dog ');
      write ('Переведите!');
end.
```

На экране данные выведены следующим образом:

▫ Переведите!  ▫ dog  ▫ dog Переведите!  ▫ dog Переведите!

8. Дано: **writeln (s:6:2);**

Если **s=112,8109**, то на экране появится _____.

9. Дана программа. Введите данную программу в Pascal.

```
var a,b,p:real;
begin
      a:= 6;
      b:= 12;
      p:=2*(a+b);
      writeln('p=',p:4:0)
end.
```

Полученный результат: _____.

## 2.2 Ввод данных с клавиатуры

Ввод данных на экран и в файл в языке программирования Pascal осуществляется с помощью процедур

**read(A1,A2,...AK)**

**readln(A1,A2,...AK)**

Процедура производит чтение К значений исходных данных и присваивает эти значения переменным А1, А2, ..., АК.

При вводе исходных данных происходит преобразование из внешней формы представления во внутреннюю, определяемую типом переменных. Переменные, образующие список ввода, могут принадлежать либо к целому, либо к действительному, либо к символьному типам. Чтение исходных данных логического типа в языке Pascal недопустимо.

Значения исходных данных могут отделяться друг от друга пробелами и нажатием клавиш табуляции или Enter.

**Не допускается разделение вводимых чисел запятыми!**

**Задания**

1. Оператор организации ввода данных с клавиатуры в языке Pascal записывается с использованием служебного слова:

   ▢ readln ▢ write ▢ begin ▢ var

2. Как называется оператор, считывающий данные с клавиатуры в оперативную память компьютера?

   ▢ оператор присваивания ▢ оператор вывода

   ▢ условный оператор ▢ оператор ввода

3. Запишите оператор, обеспечивающий во время работы программы ввод значения переменной summa.

   _____

4. Целочисленным переменным i, j, k нужно присвоить соответственно значения 10, 20 и 30. Запишите оператор ввода, соответствующий входному потоку.

   а) 20 10 30

   ***readln(j, i, k);_____***

   б) 30 20 10

   _____

   в) 10 30 20

   _____

5. Дан фрагмент программы:

```
read (a);
read (b);
c:= a+b;
write(a, b);
write(c)
```

   Упростите его, сократив число операторов ввода и вывода.

6. Дан код программы.

```
var x,y,sum, umn: integer;
begin
      write ('x=');
      readln (x);
      write ('y=');
```

```
        readln (y);
        sum:= x+y;
        umn:= x*y;
        writeln(sum);
        writeln(umn);
        readln;
end.
```

Впишите в таблицу результаты работы для заданных x и y. *(Задание можно выполнить в среде программирования на языке Pascal)*

| x | y | sum | umn |
|---|---|---|---|
| 27 | 40 | | |

# 3. ЛИНЕЙНЫЕ АЛГОРИТМЫ

 **Линейным** называется алгоритм, в котором команды выполняются последовательно друг за другом.

 **Задания**

1. Приведен неполный код программы для вычисления значения функции **y** от переменной **x**. Каждая строка кода прокомментирована. Дополните код, определив недостающие строки по комментариям.

```
var x,y: integer;        //объявление целых переменных x и y
_____________________    //начало блока операторов
    write('x=');              //вывод строки 'x='
    _____________________     //ввод значения переменной x с клавиатуры
    y := x*x + 5*x - 12;//вычисление значения функции
    writeln('y=',y);          //вывод найденного значения
end.                          //конец блока операторов
```

2. Приведен неполный код программы для вычисления длины гипотенузы прямоугольного треугольника по двум введенным катетам. Каждая строка кода прокомментирована. Дополните код, определив недостающие строки по комментариям.

```
var a,b,c: real;  //объявление действительных переменных a, b, c для двух
                  //катетов и гипотенузы
begin       //начало блока операторов
    write('a=');  //вывод строки 'a='
    _____________________     //ввод значения переменной a с клавиатуры
    _____________________     //вывод строки 'b='
    readln(b);                //ввод значения переменной b с клавиатуры
    c := sqrt(a*a + ____________);  //вычисление гипотенузы
    writeln('c=',c:1:2);            //вывод найденного значения гипотенузы
_____________________                     //конец блока операторов
```

3. Приведен код программы для вычисления значения функции **y** от переменной **x**. Некоторые строки кода прокомментированы. Допишите недостающие комментарии.

```
var                       //блок объявления переменных
      x, y: real; //объявление действительных переменных x, y
begin                     //_____________________________________________
      write('x = ');      //_____________________________________________
      readln(x);          //_____________________________________________
      y := 3*x*x - 4*x + 2;   //вычисление значения функции
      writeln('y = ',y);//вывод результата
end.                      //_____________________________________________
```

4. Приведен код программы для определения уравнения прямой, проходящей через две точки. Некоторые строки кода прокомментированы. Допишите недостающие комментарии.

```
var         //блок объявления переменных
      x1,y1,x2,y2: real;//объявление действительных переменных x1,y1,x2,y2
      k, b: real;       //объявление _____________________________________
begin                         //_____________________________________________
      write('A(x1;y1): ');    //_____________________________________________
      readln(x1, y1);         //_____________________________________________
      write('B(x2;y2): ');    //вывод строки 'B(x2;y2): '
      readln(x2, y2);         //ввод значений переменных x2 и y2
      k := (y1 - y2) / (x1 - x2);   //вычисление значения переменной k
      b := y2 - k * x2;       //_____________________________________
      writeln('y = ',k:0:2,'x + ',b:0:2);//вывод результата в виде y=kx+b
end.                          //_____________________________________________
```

5. При выполнении кода программы был получен результат:

```
Сумма = 12Произведение = 60
```

Исправьте код программы так, чтобы результат выполнения был следующим (*вывод в две строки, а не в одну*):

```
Сумма = 12
Произведение = 60
```

Код программы:

```
var x, y, k, z, p: integer;
begin
      writeln('Вычисление суммы и произведения трех чисел');
      write('Введите три целых числа через пробел');
      readln(x,y,k);
      z := x + y + k;
      p := x * y * k;
      write('Сумма = ',z);
      write('Произведение = ',p);
end.
```

6. При выполнении кода программы возникли следующие ошибки:

| Строка | Описание |
|---|---|
| 4 | Встречено 'write', а ожидалось ';' |
| 7 | Встречено 'a', а ожидалось ';' |

Найдите в коде причины возникновения этих ошибок и исправьте их.

*Подсказка: Pascal сообщает номер строки, при выполнении которой была обнаружена ошибка, но она совсем не обязательно в той строке, на которую указывает Pascal. Возможно, что проблема в конце предыдущей строки.*

```
var a,v,s:integer;
begin
```

```
        writeln('Вычисление объема и площади поверхности куба')
        write('Введите длину ребра куба');
        readln(a);
        v := a * a * a;
        s := 6a * a;
        write('Объем куба = ',v);
        write('Площадь поверхности = ',s);
end.
```

# 4. РАЗВЕТВЛЯЮЩИЕСЯ АЛГОРИТМЫ

## 4.1. Условный оператор

 Условный оператор имеет полную и краткую формы.

*Полная форма* условного оператора выглядит следующим образом:

```
if условие then оператор1
else оператор2
```

В качестве условия указывается некоторое логическое выражение. Если условие оказывается истинным, то выполняется оператор1, в противном случае выполняется оператор2.

*Краткая форма* условного оператора имеет вид:

```
if условие then оператор
```

Если условие оказывается истинным, то выполняется оператор, в противном случае происходит переход к следующему оператору программы.

В случае конструкции вида

```
if условие1 then
  if условие2 then оператор1
  else оператор2
```

**else** всегда относится к ближайшему предыдущему оператору **if**, для которого ветка **else** еще не указана. Если в предыдущем примере требуется, чтобы **else** относилась к первому оператору **if**, то необходимо использовать составной оператор:

```
if условие1 then
begin
  if условие2 then оператор1
end
else оператор2
```

Например:

```
if a<b then
  min := a
else min := b;
```

 **Задания**

1. Является ли условным оператором следующая последовательность символов?

   `if x>19 then y:=x+19 else y:=x-19` ▢ Да ▢ Нет

2. Дан условный оператор: `if (a<16) then y:=a else y:=a+9.`

   Условием является

   ▢ `y:=a` ▢ `a<16` ▢ `y:=a+9` ▢ все ответы верные

3. Условный оператор: `if (x>2) then y:=x+7 else y:=x-27.`

Выбери действие, которое будет выполняться в случае истинности условия:

▢ x>2 ▢ y:=x+7 ▢ y:=x−27 ▢ нет верного ответа

4. Дана программа, которая выводит “неотрицательное”, если введенное с клавиатуры число больше или равно нулю; “отрицательное”, если - меньше нуля. Заполните пропуски.

```
var x: real;
begin
      write('Введите число = ');
      readln(_____);
      if ___ > ___ then writeln('неотрицательное')
      else writeln(_____________________);
end.
```

5. Выберите фрагмент, который необходимо вставить в пропуск для решения задачи.

1) Дано целое число. Если оно является положительным, то прибавить к нему 5; в противном случае не изменять его. Вывести полученное число.

```
var a: integer;
begin
      writeln('Введите a');
      readln(a);
      _____________________
      writeln(a);
end.
```

▢ `if a > 0 then a:=a;` ▢ `if a > 0 then a:=a+5;` ▢ `if a < 0 then a:=a+5;`

2) Дано целое число. Если оно является положительным, то прибавить к нему 1; если отрицательным, то вычесть из него 3; если нулевым, то заменить его на 17. Вывести полученное число.

```
var a: integer;
begin
      writeln('Введите число a');
      readln(a);

      ____________________________
      writeln(a);
end.
```

▢
```
if a > 0 then
if a=0 then a:=17 else
a:=a-3
else a:=a+1;
```

▢
```
if a >= 0 then
if a=0 then a:=17 else
a:=a+1
else a:=a-3;
```

▢
```
if a >= 0 then
if a=0 then a:=17 else
a:=a-3
else a:=a+1;
```

6. Дан код программы. В ней пользователь вводит число. Если введенное число четное, то программа делит его на два, а если нечетное, то прибавляет единицу и делит результат на два.

```
var n:integer;
begin
  writeln('Введите число ',);
  readln(n);
  if n mod 2 = 0 then n=n div 2
  else n:=(n+1) div 2;
```

```
  write(n)
end.
```

При выполнении программы возникли следующие ошибки.

| Строка | Описание |
|---|---|
| 3 | Встречено ')', а ожидалось выражение |
| 5 | Встречено '=', а ожидалось ';' |

Исправьте ошибки.

## 4.2. Составной оператор

 Составной оператор предназначен для объединения нескольких операторов в один. Он имеет вид:

```
begin
  операторы
end
```

Операторы отделяются один от другого символом ";". Ключевые слова **begin** и **end**, окаймляющие операторы, называются *операторными скобками*.

Например:

```
s := 0;
p := 1;
for var i:=1 to 10 do
begin
  p := p * i;
  s := s + p
end
```

Перед **end** также может ставиться ";". В этом случае считается, что последним оператором перед end является пустой оператор, не выполняющий никаких действий.

 **Задания**

1. Как называют служебные слова, между которыми заключается программный блок?
   - ▫ Начало и конец
   - ▫ Фигурные скобки
   - ▫ Логические скобки
   - ▫ Операторные скобки
2. Дополните программу решения линейного уравнения. Коэффициенты a, b вводятся с клавиатуры. На экране выводится информация о значении корня линейного уравнения.

```
var a,b,x:real;
begin
  writeln('Решение линейного уравнения');
  _______('Введите коэффициенты a и b через пробел');
  readln(__,__);                                    //ввод значений коэффициентов
if a<>0 _____                                       //условный оператор
    begin
      __:=-b/a;                                     //вычисление корня
      writeln('Корень уравнения __=',____);         //вывод результата
    _______                                   //конец составного оператора
    else if b<>0 then writeln ('Корней нет')
            else _____________(' x - любое число');    //вывод результата
end.
```

3. Дополните программу решения квадратного уравнения. Коэффициенты a, b и c вводятся с клавиатуры. На экране выводится информация о значении корней квадратного уравнения.

```
var a, b, c, d, x, x1, x2: real;     //описание переменных
begin
  writeln ('Решение квадратного уравнения');
  write ('Введите коэффициенты a, b, c: ');
  _________ (a, b, c);                      //ввод значений коэффициентов
  d:=_____________;                         //вычисление дискриминанта
  if d<0 then __________ ('Корней нет');    //вывод результата
  if d=0 then
           begin x:=-b/2*__;                //вычисление единственного корня
           _________ ('Корень уравнения x=', x:9:3)//вывод результата
           end;
  if d__0 then                              //если дискриминант больше нуля
           _________                        //начало составного оператора
             x1:=(-b+sqrt(d))/(2*a); //вычисление первого корня
             ___:=(-b-sqrt(___))/(2*a);     //вычисление второго корня
             writeln ('Корни уравнения:');
             writeln ('x1=', x1:9:3);       //вывод результата
             writeln ('x2=', x2:9:3) //вывод результата
           end
end.
```

4. Ниже представлен код программы, которая решает следующую задачу: даны действительные числа a, b (a≠b); меньшее из этих двух чисел необходимо заменить их суммой, а большее - их произведением.

```
var a,b,sa,sb: integer;
begin
    writeln('Введите два числа');
    readln(a,b);
    sa:=a;
    sb:=b;
    if a>b then
            begin
            b:=sa+sb;
            a:=sa*sb;
            end;
      else
            begin
            a:=sa+sb;
            b:=sa*sb;
            end;
    writeln ('первоначальные числа  a=',sa,'  b=',sb);
    writeln ('полученные  числа  a=,a,'  b=',b);
end.
```

При выполнении кода программы возникли следующие ошибки:

| Строка | Описание |
|---|---|
| 12 | Встречено 'else', а ожидалось ';' |
| 18 | Встречено 'b', а ожидалось ')' |

Найдите в коде причины возникновения этих ошибок и исправьте их.

5. Придумайте тест для проверки работы программы из задания 3 и запишите его в таблицу.

   *Тестирование программы - важный этап процесса программирования. Для этого этапа необходимо придумать разные варианты входных данных и рассчитать для них результаты, которые должны получиться в программе.*

| № | Входные данные (a, b) | Результат |
|---|---|---|
| 1 | *a=20, b=13* | *Произведение: 260, Сумма: 33* |
| 2 | | |
| 3 | | |

6. Придумайте тест для проверки работы программы из задания 2 и запишите его в таблицу.

| № | Входные данные (a, b, c) | Результат |
|---|---|---|
| 1 | *a=1, b=1, c=1* | *Корней нет* |
| 2 | | |
| 3 | | |
| 4 | | |

# 5. ЦИКЛЫ

## 5.1. Циклы с заданным условием продолжения работы

### while

 Цикл с заданным условием продолжения работы иначе еще называется **циклом с предусловием** или циклом **while**.

Оператор цикла **while** имеет следующую форму:

```
while условие do
  оператор
```

Условие представляет собой выражение логического типа, а оператор после **do** называется телом цикла. Перед каждой итерацией цикла условие вычисляется, и если оно истинно, то выполняется тело цикла, в противном случае происходит выход из цикла.

Если условие всегда оказывается истинным, то может произойти зацикливание:

```
while 2>1 do
  write(1);
```

**Пример.** Ввод числа с клавиатуры в заданном диапазоне. Пользователь вводит число. Если число меньше 10 и больше 20, программа запрашивает число снова. Если было введено число большее 10 и меньшее 20, оно выводится на экран.

```
var n: real;                          //объявление переменной для числа
begin
n := -1;                                    //присваиваем начальное значение
while not((n > 10) and (n < 20)) do         //пока число попадает в диапазон
      begin                                 //начало тела цикла
      writeln('Введите число большее 10 и меньшее 20');//вывод приглашения
      readln(n);                            //ввод значения
      end;                                  //конец тела цикла
writeln('Вы ввели число ', n);        //вывод сообщения
end.
```

 **Задания**

1. Выберите верный способ описания цикла **while**.

| | | |
|---|---|---|
| ▢ `while i > 0 do`<br>`  writeln(i*i);`<br>`  i := i - 5;` | ▢ `while i > 0 begin`<br>`  writeln(i*i);`<br>`  i := i - 5;`<br>`  end;` | ▢ `while i > 0 do begin`<br>`  writeln(i*i);`<br>`  i := i - 5;`<br>`  end;` |

2. Дан код программы, которая позволяет отгадать целое число, которое "загадал" компьютер в определенном диапазоне. Проанализируйте код и заполните пропуски.

```
var a,b: integer;
begin
    randomize;
    a := random(100);
    while a <> b do begin
        write('Введите число: ');
        readln(b);
        if b > a then writeln('Много')
        else
            if b < a then writeln('Мало')
            else
                writeln('Угадал');
    end;
end.
```

Псевдослучайное число записывается в переменную ___.

Пользователь вводит с клавиатуры значение, которое записывается в переменную ___.

В программе используется цикл с *условием окончания работы / условием продолжения работы / заданным числом повторений* (подчеркнуть).

Условие выполнения цикла: _____.

Условие в первом условном операторе: _____; условие во втором условном операторе: _____.

Условие, при котором на экране появится слово 'Угадал': _____.

3. Приведен неполный код программы, которая позволяет составить таблицу значений функции $y = 5 - \frac{x^2}{2}$ на отрезке [-5; 5] с шагом 0.5. Заполните пропуски, опираясь на комментарии.

```
var _________________              //объявление вещественных переменных x и y
begin
      _________________            //присвоение переменной x значения -5
      writeln(' x      y  ');
      _________________ do begin   //цикл: пока x не больше 5
      y:=_______________           //вычисление значения функции и запись в y
      writeln(x:4:1,' | ',y:5:2);
      x:=_______________           //увеличение значения переменной x на 0.5
      end;
end.
```

4. Необходимо подсчитать количество четных и нечетных цифр в числе. Заполните пропуски в тексте, чтобы составить алгоритм своих действий для решения данной задачи на языке Pascal.

**Описание переменных:**

| Зачем нужна переменная | Имя переменной | Тип |
|---|---|---|
| для введенного числа | `a` | целый |
| для количества четных цифр | `chet` | _____ |
| для количества _____ цифр | `nechet` | _____ |

**Алгоритм решения задачи:**

1) Если число делится без остатка на 2, значит последняя цифра _____*(четная/нечетная)*, и мы должны увеличить переменную `chet` на 1. Иначе - последняя цифра _____*(четная/нечетная)*, и мы должны увеличить переменную _____*(a/chet/nechet)* на 1.
2) Избавляемся от последней цифры в числе. Для этого поделим число на 10 без остатка. Для этого используем операцию _____ *(div/mod)*.

5. Приведен неполный код программы, которая позволяет подсчитать количество четных и нечетных цифр числа. Заполните пропуски, опираясь на комментарии и алгоритм из задания 3.

```
var   _____________________________;      //объявление переменных
begin
      readln(a);
      chet := 0;                          //присвоение начального значения
      nechet := ____;                     //переменным chet и nechet
      while __________ do begin          //цикл: пока a больше нуля
            if (__________) = 0 then      //если остаток от деления a на 2 - ноль
            chet := _______________       //увеличиваем chet на 1
            else                          //иначе
            ____________________________; //увеличиваем nechet на 1
      a := _____________;                 //избавляемся от последней цифры числа
      end;
      writeln('Четных: ', chet);
      writeln('Нечетных: ', nechet);
end.
```

6. Приведен код программы, которая позволяет составить таблицу значений функции $y = x^2 - 4x + 1$ на отрезке [-1; 1] с шагом 0.1. Напишите комментарии.

```
var x,y:real;                         //_____________________________________
begin
```

```
        x:=-1;                          //_____________________________________
        writeln(' x      y  ');
        while x<=1 do begin             //_____________________________________
        y:=x*x-4*x+1;                   //_____________________________________
        writeln(x:4:1,' | ',y:5:2);
        x:=x+0.1;                       //_____________________________________
        end;
 end.
```

7. Дан код программы для вычисления степени числа.

```
var: n, step, i, res: integer;
begin
    write ('Число: ');
    readln (n);
    write ('Степень: ')
    readln (step);
    res := 1;
    i := 0;
    while i < abs(step) do begin
        res := res * n;
        i = i + 1;
    end;
    writeln (res);
end.
```

При его выполнении возникли следующие ошибки:

| Строка | Описание |
|---|---|
| 1 | Встречено ':', а ожидался идентификатор |
| 6 | Встречено 'readln', а ожидалось ';' |
| 11 | Встречено '=', а ожидалось ';' |

Найдите в коде причины ошибок и исправьте их.

8. Придумайте тест для программы из задания 6 и заполните таблицу.

| № | Входные данные (`n, step`) | Результат |
|---|---|---|
| 1 | *n=6, step=3* | *216* |
| 2 | | |
| 3 | | |
| 4 | | |

| | | |
|---|---|---|
| 5 | | |

## 5.2. Программирование циклов с заданным условием окончания работы

### repeat

 Цикл с заданным условием окончания работы иначе еще называется **циклом с постусловием** или циклом **repeat**.

Оператор цикла **repeat** имеет следующую форму:

```
repeat
  операторы
until условие
```

В отличие от цикла **while**, условие вычисляется после очередной итерации цикла, и если оно истинно, то происходит выход из цикла. Таким образом, операторы, образующие тело цикла оператора repeat, выполняются по крайней мере один раз.

Обычно оператор **repeat** используют в ситуациях, где условие нельзя проверить, не выполнив тело цикла. Например:

```
repeat
  read(x);
until x=0;
```

Если условие всегда оказывается ложным, то может произойти зацикливание:

```
repeat
  write(1);
until 2=1;
```

 **Задания**

1. Какой цикл называют циклом с предусловием?

   ▫ Repeat ▫ While ▫ For

2. Какой цикл называют циклом с условием окончания работы?

   ▫ Repeat ▫ While ▫ For

3. Сколько раз будет выполнено тело цикла в следующем фрагменте программы?

```
a:=1;
b:=2;
repeat
    a:=a+1;
    b:=b+2;
until a+b>10;
```

   Ответ: _____

4. По приведенному примеру перепишите фрагмент программы так, чтобы вместо цикла **while** получился цикл **repeat**.

*Пример:*

Цикл **while**

```
a:=2;
b:=0;
while a+b<7 do
begin
  a:=a+1;
  b:=b+a;
end;
```

Цикл **repeat**

```
a:=2;
b:=0;
repeat
  a:=a+1;
  b:=b+a;
until a+b>=7
```

Задание:

Цикл **while**

```
a:=2;

b:=3;

while a+b < 10 do

begin

  a:=a+1;

  b:=b+2;

end;
```

Цикл **repeat**

5. Дан код программы, в которой происходит вывод цифр введенного пользователем числа в столбик по разрядам (единицы, десятки, сотни и т.д.).

   Пример. Введено число 2 572.

   Результат работы:

   ```
   1 - 2
   10 - 7
   100 - 5
   1000 - 2
   ```

   Прокомментируйте код как можно подробнее.

```
var n,i: integer;                  //_____
begin                              //_____
  i:=1;                            //_____
  writeln ('Введите число n: ');   //_____
  readln(n);                       //_____
  repeat                           //_____
      writeln(i,' - ', n mod 10);  //_____
```

```
      n:= n div 10;                 //_____________________________________
      i:=i*10;                      //_____________________________________
  until n = 0;                      //_____________________________________
end.                                //_____________________________________
```

6. Заполните пропуски и составьте программу, которая будет находить сумму цифр числа.

   *В переменную **n** с клавиатуры вводится число. Сумма цифр записывается в переменную **sum**.*

```
var n, sum: integer;
begin
      ________('Введите число n: ');    //вывод строки с приглашением на ввод
      ________(n);                      //ввод значения в переменную n
      sum:=0;
      ________                          //начало цикла
            sum:= sum + (n mod 10);
            n:= n ________ 10;          //отбрасываем последнюю цифру числа n
      until n = 0;
      _________________________         //выводим результат
end.
```

7. Дан код программы, которая находит произведение цифр введенного числа.

```
var n, prod: integer;
begin
      writeln ('Введите число n: ');
      readln(n);
      prod:=1;
      repeat
            prod:= prod * (n mod 10);
            n:= n div 10;
      until n = 0;
      wrile(prod)
end
```

   При запуске программы возникли следующие ошибки.

| Строка | Описание |
|---|---|
| 10 | Неизвестное имя 'wrile' |
| 11 | Ожидалось '.' |

   Исправьте код программы.

8. Составьте тест для программ из заданий 6 и 7 и заполните таблицу.

| № | Входные данные (n) | Результат работы программы 6 | Результат работы программы 7 |
|---|---|---|---|
| 1 | *n=15637* | *sum = 22* | *prod = 630* |
| 2 | | | |

| | | | |
|---|---|---|---|
| 3 | | | |
| 4 | | | |
| 5 | | | |

## 5.3. Программирование циклов с заданным числом повторений

## for

 Оператор цикла **for** имеет одну из двух форм:

```
for переменная := начальное значение to конечное значение do оператор
```

или

```
for переменная := начальное значение downto конечное значение do оператор
```

Текст от слова **for** до слова **do** включительно называется *заголовком цикла*, а оператор после **do** - *телом цикла*. Переменная после слова **for** называется параметром цикла. Для первой формы цикла с ключевым словом **to** параметр цикла изменяется от начального значения до конечного значения, увеличиваясь всякий раз на единицу, а для второй формы ключевым словом **downto** - уменьшаясь на единицу. Для каждого значения переменной-параметра выполняется тело цикла. Однократное повторение тела цикла называется итерацией цикла. Значение параметра цикла после завершения цикла считается неопределенным.

Если для цикла **for** ... **to** начальное значение переменной цикла больше конечного значения или для цикла **for** ... **downto** начальное значение переменной цикла меньше конечного значения, то тело цикла не выполнится ни разу.

 **Задания**

1. В каких строках код написан верно?

   ▫ `for i:=1 to 10 do`  ▫ `for i:=1 downto 10 do`

   ▫ `for i:=10 downto 1 do`  ▫ `for i:=10 to 1 do`

2. Дано: `for i:=1 to 56 do`

   При последнем выполнении тела цикла **i=_____.**

3. Сколько раз будет выполнен цикл `for n:=1 to 35 do`?

   Ответ: _____.

4. Дана программа на языке Pascal. Вывод на экран квадратов чисел от 1 до 46.

   1) Заполните пропуски  2)Прокомментируйте код

```
var ____: integer;      //_____________________________________
begin                   //_____________________________________
```

```
      for n:=1 to ____ do      //_____
      writeln(sqr(____));      //_____
end.                           //_____
```

5. Запишите программу на языке Pascal. Вывести на экран числа от 3 до 18 в обратном порядке.

   1) Заполните пропуски  2)Прокомментируйте код

```
var i: integer;                //_____
begin                          //_____
      for i:=____ downto ____ do//_____
      writeln(____);           //_____
end.                           //_____
```

6. Дан код программы, выводящей кубы чисел от 12 до 4.

```
var i:integer;
begin
      for m:=12 downto 4 do
      writeln (m*m*m);
end.
```

   При выполнении программы возникли ошибки:

| Строка | Описание |
|---|---|
| 3 | Неизвестное имя 'm' |
| 4 | Неизвестное имя 'm' |

   Найдите и устраните причину возникновения таких ошибок.

7. Вычислите n! («эн-факториал») — произведение всех натуральных чисел от 1 до n (n!=1·2·3·…· n). Дополните код программы, опираясь на комментарии. Допишите комментарии, где они пропущены.

```
var  ___,___,___: __________;   //объявление переменных s,i,n целого типа
begin                           //_____
  _________(n);                 //ввод с клавиатуры значения n
  s := 1;                       //_____
  for i := __ to __ do          //цикл от 1 до n
    s := s * i;                 //_____
  ___________________           //вывод результата s на экран
end.                            //_____
```

# 6. ОДНОМЕРНЫЕ МАССИВЫ ЦЕЛЫХ ЧИСЕЛ

**Массив —** это пронумерованная последовательность величин одного типа, обозначаемая одним именем. Номер элемента массива называется **индексом**.

## 6.1. Описание массива

Массивы задают свой размер непосредственно в типе. Память под такие массивы выделяется сразу при описании. В квадратных скобках через две точки указывается индекс первого элемента и индекс последнего элемента. Количество элементов в массиве определяется именно этими значениями.

Массив объявляется следующим образом:

```
var имя_массива: array [индекс_начала..индекса_конца] of тип_элементов
```

Пример:
```
var a: array[1..10] of integer;    //объявлен массив a, состоящий из десяти целых чисел.
```

**Задания**

1. Для описания массивов в языке Pascal используется ключевое слово:
   ▢ **record** ▢ **array** ▢ **string** ▢ **integer**
2. Индексом в массиве называется:
   ▢ Значение максимального элемента ▢ Порядковый номер элемента в массиве
   ▢ Разрядность массива ▢ Количество элементов в массиве
3. Может ли массив содержать одновременно целые и вещественные значения?
   Ответ: _____
4. Какая из записей описывает массив правильно?
   а)
   ▢ `var a[1..10] of array` ▢ `var a:array[1..10] of integer`
   ▢ `var array[1..10] of integer` ▢ `var a of array: integer`
   б)
   ▢ `D[1..5]:array of integer;` ▢ `D: array[1..5] of real;`
   ▢ `D: array[1...5] of real;` ▢ `Array D: [1..5] of real;`
5. Как задается количество элементов массива?
   ▢ Номером и значением первого элемента ▢ Номером последнего элемента
   ▢ Значением первого и последнего элемента ▢ Индексами первого и последнего элемента

6. Массив данных имеет

   ▢ общее имя, один тип, равные значения ▢ общее имя, один тип, разные значения

   ▢ один тип, разные имена, разные значения ▢ общее имя, разные типы, разные значения

7. В какой строке НЕправильное описание массива?

   ▢ `var ch: array[1..10] of integer;` ▢ `var stih: array[1,5..7] of real;`

   ▢ `var STL23: array[2..108] of real;` ▢ `var a: array[10..20] of integer;`

8. Какое количество данных может находиться в данном массиве

   `var ch: array[3..15] of integer;`?

   ▢ 15 целых чисел ▢ 12 целых чисел ▢ 13 целых чисел ▢ 12 действительных чисел

## 6.2. Заполнение массива

 Заполнение массива происходит поэлементно (обращаемся отдельно к каждому элементу массива) с помощью цикла **for**, количество повторений которого равно количеству элементов массива. Чтобы обратиться к элементу массива, нужно указать имя массива и в квадратных скобках - индекс элемента.

```
a[3] - обращаемся к третьему элементу массива a
mass[i] - обращаемся к i-тому элементу массива mass
```

Массив можно заполнять вручную, присваивая каждому элементу массива значение, введенное с клавиатуры (оператор **read**):

```
for i:=1 to 10 do readln(a[i]);
```

можно заполнить массив случайными числами, подключив генератор случайных чисел **randomize**:

```
randomize;
for i:=1 to 10 do a[i]:=random(10);
```

а также, массив может заполняться значениями, вычисляемыми по какой-нибудь формуле:

```
for i:=1 to 10 do a[i]:=i*i-2*i+1;
```

 **Задания**

1. Для заполнения массива данными с помощью генерации случайных чисел используется команда:

   ▢ **writeln** ▢ **readln** ▢ **randomize** ▢ **clrscr**

2. Для заполнения массива путем ввода чисел с клавиатуры мы используется оператор

   ▢ **write** ▢ **repeat** ▢ **until** ▢ **read**

3. В записи **Mass[5]=3.6** число 5 обозначает:

   ▢ имя массива ▢ порядковый номер элемента в массиве

   ▢ значение ячейки массива ▢ имя ячейки массива

4. В записи `d[4]=2.5`, 2.5 обозначает

   ▢ имя массива ▢ обозначение типа

   ▢ значение элемента массива ▢ имя ячейки

5. Каким способом осуществляется ввод элементов массива

```
randomize;
writeln('Введите количество элементов массива');
readln(n);
for i:=1 to n do
begin
      a[i]:=random(50);
      writeln('a['i']=)',a[i]);
end;
```

   ▢ с помощью генератора случайных чисел ▢ с клавиатуры

   ▢ присвоением вычисляемых значений ▢ нет верного ответа

6. Каким способом осуществляется ввод элементов массива

```
for i:=1 to 10 do
begin
      a[i]:=i*i/i+2;
      writeln('a(',i,')=',a[i]);
end;
```

   ▢ с помощью генератора случайных чисел ▢ с клавиатуры

   ▢ присвоением вычисляемых значений ▢ нет верного ответа

7. Заполните таблицу.

| Функция | Диапазон случайных чисел |
|---|---|
| `random(10);` | от 0 до 10 |
| `random(100);` | |
| `random;` | |
| `random(10)-9;` | |
| `random(101)-50;` | |

## 6.3. Вывод массива

 Процедура **`write`** выводит статический массив, заключая элементы в квадратные скобки и разделяя их запятыми:

```
writeln(a); // [1,2,3,4,5]
```

Также элементы массива можно вывести с помощью цикла в столбик:

```
for i:=1 to 10 do
```

```
  writeln(a[i]);
```

или в строку через пробел:

```
for i:=1 to 10 do
  write(a[i], ' ');
```

**Задания**

1. Задан одномерный массив a, состоящий из целых чисел. Найдите элемент массива **a[i]**.

   а) **a=[1, 5, 9, 13]** **a[2]=_____**

   б) **a=[4, 8, 13, 25, 12]** **a[3]=_____**

   в) **a=[0,1,2,3,4,5,6,7,8,9]** **a[7]=_____**

2. Что выполняет следующий фрагмент программы:

```
for i:=1 to 10 do
write (a[ i ], ' ');
```

   ▫ выводит 10 значений массива на экран в столбик

   ▫ производит ввод 10 элементов в массив

   ▫ выводит 10 значений массива на экран в строчку

   ▫ выполняет проверку значений элементов массива

3. Дан одномерный массив `a=[1,2,3,4,5]`. Сопоставьте фрагменты кода и результаты его выполнения.

| **Фрагмент кода** | **Результат** |
|---|---|
| `for i:=1 to 5 do write(a[i], ' ');` | `[1,2,3,4,5]` |
| `for i:=1 to 5 do writeln(a[i], ')');` | `1,2,3,4,5,` |
| `writeln(a);` | `1)`<br>`2)`<br>`3)`<br>`4)`<br>`5)` |
| `for i:=1 to 5 do write(a[i], ',');` | `1 2 3 4 5` |
| `for i:=1 to 5 do writeln(a[i]);` | `1`<br>`2`<br>`3`<br>`4`<br>`5` |

4. Дополните код так, чтобы одномерный массив был заполнен квадратами первых десяти натуральных чисел, выведите массив и добавьте комментарии.

```
var_____________________________________    //описание массива из 10 целых чисел
i: integer;                                 //_____________________________________
begin
      for i:=1 to 10 do  a[i]:=_______;     //_____________________________________
      ____________________                  //вывод массива через запятую
end.
```

5. Дан код программы, в которой массив заполняется значениями функции $y = x - 3$ в диапазоне $[-3; 3]$, а затем выводится в столбик в виде: $(x, y)$.

```
var x,y:array[1..7] of integer;
i: integer;
begin
for i=1 to 7 do
  begin
  x[i]:=i-4
  y[i]:=x[i]-3;
  writeln((',x[i],',',y(i),')');
  end
end.
```

При выполнении этой программы возникли следующие ошибки.

| Строка | Описание |
|---|---|
| 4 | Встречено '=', а ожидалось ':' |
| 7 | Встречено 'y', а ожидалось ';' |
| 8 | Неожиданный символ ''' |
| 8 | Встречено ']', а ожидалось ')' |

Найдите и исправьте эти ошибки.

*Обратите внимание на две последние ошибки. Pascal сообщает об ошибке и описывает ее в соответствии со своей логикой, которая не всегда совпадает с намерениями программиста.*

## 6.4. Вычисление суммы элементов массива

Одной из основных задач при работе с массивами является нахождение суммы элементов. Для решения этой задачи воспользуемся циклом **for** и дополнительной переменной, которую назовем **sum**. Начальное значение переменной для нахождения суммы нескольких элементов всегда равно нулю! *(Начальное значение переменной для нахождения произведения нескольких элементов всегда равно 1!)*

**Пример.** Нахождение суммы элементов массива из 10 чисел, который заполнен случайными числами в диапазоне от 0 до 10.

```
var a: array[1..10] of integer;          //описание массива
sum, i: integer;                         //переменные для суммы и счетчика цикла
begin
  randomize;                             //подключение генератора случайных чисел
  sum:=0;                                //присвоение начального значения сумме
  for i:=1 to 10 do
    a[i]:=random(11);                    //заполнение случ.числами от 0 до 10
  writeln('Массив: ',a);                 //вывод массива в строку
  for i:=1 to 10 do                      //цикл для поиска суммы массива
    sum:=sum + a[i];                     //добавление к сумме текущего элемента
  writeln('Сумма элементов: ', sum);     //вывод результата
end.
```

Вероятный результат выполнения:

```
Массив: [3,0,9,5,2,0,3,6,7,1]
Сумма элементов: 36
```

**Задания**

1. С помощью какого цикла удобно вычислять сумму элементов массива?

   ▫ Repeat ▫ While ▫ For

2. Какое начальное значение нужно присвоить переменной для записи суммы элементов массива?

   Ответ: _____

3. Какое начальное значение нужно присвоить переменной для записи произведения элементов массива?

   Ответ: _____

4. Какой фрагмент кода для вычисления суммы элементов массива верный?

   ▫ `for i:=1 to 10 do sum:=a[i];`

   ▫ `for i:=1 to 10 do sum:=sum + a[i];`

   ▫ `for i:=1 to 10 do sum + a[i];`

5. Дополните код программы, которая находит сумму элементов с третьего по седьмой массива из 10 действительных чисел. Массив заполняется случайным образом числами от нуля до 10.

```
var a: _____;                            //описание массива
_____;                                   //переменные для суммы и счетчика цикла
begin
  _____;                                 //подключение генератора случайных чисел
  _____;                                 //присвоение начального значения сумме
  for i:=1 to 10 do
  _____;                                 //заполнение случ.числами от 0 до 10
  _____;                                 //вывод массива в строку
```

```
  for i:=3 to 7 do                        //цикл для поиска суммы массива
  _____________________________;          //добавление к сумме текущего элемента
  writeln('Сумма элементов: ', sum);      //вывод результата
end.
```

6. Дан код программы для нахождения суммы элементов массива из 10 чисел, который заполнен случайными числами в диапазоне от -10 до 10. Прокомментируйте код как можно подробнее.

```
var a: array[1..10] of integer;           //_____________________________________
sum, i: integer;                          //_____________________________________
begin                                     //_____________________________________/
  randomize;                              /_____________________________________
  sum:=0;                                 //_____________________________________
  for i:=1 to 10 do                       //_____________________________________
    a[i]:=random(21)-10;                  //_____________________________________
  writeln('Массив: ',a);                  //_____________________________________
  for i:=1 to 10 do                       //_____________________________________
    sum:=sum + a[i];                      //_____________________________________
  writeln('Сумма элементов: ', sum);      //_____________________________________
end.                                      //_____________________________________
```

7. Дан код программы, вычисляющей сумму элементов массива, заполненного случайными числами.

```
var a: array[1..5] of integer;
sum, i: integer;
begin
  randomize;
  sum:=0;
  for i:=1 to 10 do
    a:=random(10);
  writeln('Массив: ',a);
  for i:=1 to 10 do
    sum:=sum + a;
  writeln('Сумма элементов: ', sum);
end.
```

При выполнении этой программы возникли следующие ошибки.

| Строка | Описание |
|---|---|
| 7 | Нельзя преобразовать тип integer к array [1..5] of integer |
| 10 | Нельзя преобразовать тип IEnumerable<integer> к integer |

К тому же программист планировал заполнить массив числами из диапазона [-10;10], но неверно указал диапазон в коде. Найдите и исправьте эти ошибки.

## 6.5. Последовательный поиск в массиве

 Еще одна задача при работе с массивами - поиск и/или подсчет элементов с определенным значением. Чаще всего в таких программах используется цикл `for`.

**Пример 1. Поиск минимального (максимального) элемента в массиве.**

Эта задача является основной задачей на поиск в массиве. Чтобы ее выполнить, во-первых, нужно завести переменную, в которую будет записано минимальное (максимальное) значение, назовем ее `min` (`max`). Запишем в эту переменную значение первого элемента массива, так как предполагаем, что он является минимальным (максимальным). Далее последовательно сравниваем все следующие элементы массива с `min` (`max`). Если возникает ситуация, что какой-то элемент меньше (больше), чем `min` (`max`), то запишем в `min` (`max`) значение этого элемента, а следующие элементы массива будем сравнивать с ним. *В коде ниже реализован поиск минимального значения массива. Поиск максимального осуществите самостоятельно.*

```
var
  i,min: integer;
  a: array[1..10] of integer;
begin
  randomize;
  for i:=1 to 10 do a[i]:=random(11)-5;//заполнение массива числами от -5 до 5
  min:=a[1];
  for i:=2 to 10 do
if a[i]<min then min:=a[i];//если элемент массива меньше минимума, записываем его в min
  writeln('Минимальный элемент равен: ',min)
end.
```

**Пример 2**. **Проверка, есть ли в массиве целых чисел, который заполняется случайным образом, число `0`.**

Для выполнения проверки заведем переменную `zero` логического типа. Ее начальное значение будет `false` - то есть ложь. Таким образом мы предполагаем, что в массиве нет нулей. Если какой-то из элементов массива всё-таки равен нулю, заменим значение переменной `zero` на `true`, то есть истина.

```
var
  i: integer;
  a: array[1..10] of integer;
  zero:=false;                 //переменная логического типа. начальное значение - ложь
begin
  randomize;
  for i:=1 to 10 do a[i]:=random(11)-5;    //заполнение массива числами от -5 до 5
```

```
  for i:=1 to 10 do
      if a[i]=0 then zero:=true;//если элемент массива равен нулю, меняем false на true
  if zero=true then writeln('В массиве есть 0')
      else writeln('Нуля нет');
end.
```

**Пример 3. Поиск номера элемента массива со значением `0`.**

Для выполнения этой задачи заведем переменную `n` целого типа. Ее начальное значение будет равно какому-нибудь числу, которое не используется для индексов в массив, например `-1`. Поиск нуля будем осуществлять так же, как в предыдущем примере, но сейчас при успехе будем изменять переменную `n` так, чтобы в ней был записан индекс элемента со значением `0`.

```
var
  i,n: integer;
  a: array[1..10] of integer;
begin
  randomize;
  for i:=1 to 10 do a[i]:=random(11)-5;//заполнение массива числами от -5 до 5
  n:=-1;
  for i:=1 to 10 do if a[i]=0
      then n:=i;  //если элемент массива равен нулю, записываем в n индекс i
  if n=-1 then writeln('Нуля нет')
else writeln('Ноль под номером ', n);
end.
```

***Обратите внимание!*** *Если в массиве несколько нулей, то в переменную n будет записан только индекс последнего. Это объясняется тем, что в цикле происходит перезапись переменной n при каждом успехе.*

**Пример 4. Подсчет количества элементов массива, значение которых больше нуля.**

Когда в задачах программирования необходимо посчитать количество каких-то элементов, нужно завести переменную, которая будет выполнять функцию счетчика. Пусть это будет переменная `kol`. Ее начальное значение равно нулю, так как мы предполагаем, что положительных элементов вообще нет в массиве. Если элемент массива удовлетворяет условию, то мы увеличим значение переменной `kol` на 1.

```
var
  i,kol: integer;                    //переменные для цикла и подсчитывания (счетчик)
  a: array[1..10] of integer;        //описание массива
begin
  randomize;
  for i:=1 to 10 do a[i]:=random(11)-5;//заполнение массива числами от -5 до 5
  kol:=0;                            //начальное значение
```

```
  for i:=1 to 10 do
      if a[i]>0 then kol:=kol+1;//если элемент массива больше 0, увеличиваем kol на 1
  writeln('Количество положительных чисел: ',kol)
end.
```

**Задания**

1. Массив a[1]=12, a[2]=3, a[3]=-5, a[4]=-6, a[5]=4, a[6]=9, a[7]=0, a[8]=8. Определите значение переменной kol после выполнения следующего фрагмента программы:

   ```
   kol:=0;
   for i:=4 to 8 do
   if a[i]<0 then
   kol:=kol+1;
   ```

   Ответ: _____

2. Массив r[1]=12, r[2]=3, r[3]=-5, r[4]=-6, r[5]=4, r[6]=9, r[7]=0, r[8]=8. Определите значение переменной kol после выполнения следующего фрагмента программы:

   ```
   kol:=0;
   for i:=1 to 8 do
   if r[i]>-2 then
   kol:=kol+1;
   ```

   Ответ: _____

3. Массив m[1]=12, m[2]=3, m[3]=5, m[4]=6, m[5]=4, m[6]=9, m[7]=12, m[8]=8. Определите значение переменной min после выполнения следующего фрагмента программы:

   ```
   min:=0
   for i:=1 to 8 do
   if r[i]<min then
   min:=r[i];
   ```

   Ответ: _____

4. Массив a[1]=12, a[2]=3, a[3]=-5, a[4]=-6, a[5]=4, a[6]=9, a[7]=0, a[8]=8. Определите значение переменной k после выполнения следующего фрагмента программы:

   ```
   k:=0;
   for i:=1 to 8 do
   if r[i]>5 then
   k:=k+i;
   ```

   Ответ: _____

5. Что выполняет следующий фрагмент программы?

   ```
   min:=a[1];
   for i:=1 to 5 do
     if a[i]< min then
     min:=a[i];
   writeln(min);
   ```

- ▢ находит значение минимального элемента массива
- ▢ сравнивает первый элемент массива с минимальным элементом
- ▢ находит индекс минимального элемента массива
- ▢ делает значения элементов массива минимальными

6. Дан код программы, в которой находится количество элементов массива, которые меньше нуля. Массив из 100 элементов заполняется случайными числами в диапазоне (-50, 50). Прокомментируйте код как можно подробнее.

```
var a: array[1..100] of integer;      //_____________________________________
kol, i: integer;                      //_____________________________________
begin                                 //_____________________________________
  randomize;                          //_____________________________________
  kol:=0;                             //_____________________________________
  for i:=1 to 100 do                  //_____________________________________
    a[i]:=random(101)-50;             //_____________________________________
  for i:=1 to 100 do                  //_____________________________________
    if a[i]<0 then kol:=kol + 1;      //_____________________________________
  writeln('Количество отрицательных   //_____________________________________
элементов: ', kol);                   //_____________________________________
end.                                  //_____________________________________
```

7. Дополните код программы, в которой находится количество чисел, которые равны введенному числу. Массив состоит из 50 элементов, заполняется случайным образом числами в диапазоне (0, 7).

```
var _____________________________;     //описание массива из 50 элементов
kol, i, n: integer;                    //объявление переменных
begin
  writeln('Введите число от 0 до 7: '); //приглашение на ввод
  _____________________________;       //ввод числа n
  _____________________________;       //подключение генератора случ.чисел
  kol:=0;                              //присваивание начального значения
  for i:=1 to 50 do
    a[i]:=random(___);                 //заполнение массива случ.числами
  _____________________________;       //вывод массива в строку
  for i:=1 to 50 do
    if a[i]=n                          //сравнение элементов массива с n
    then _____________________;        //увеличение kol, если число равно n
  writeln('Количество ', n, ': ', kol); //вывод результата
end.
```

8. Дан код программы, в которой находится сумма всех положительных элементов массива из 10 случайных чисел в диапазоне (-10,10).

```
var a: array[1..10] of integer;
sum, i: integer;
begin
  randomize;
  sum:=0;
  for i:=0 to 10 do
    a[i]:=random(21)-10;
  writeln('Массив: ',a);
  for i:=1 to 10 do
    if a[i]>0;
    then sum:=sum + a[i];
  writeln('Сумма положительных элементов: ' sum);
end.
```

При запуске программы произошли следующие ошибки

| Строка | Описание |
|---|---|
| 7 | Ошибка времени выполнения: Индекс находился вне границ массива. |
| 10 | Встречено ';', а ожидалось then |
| 12 | Встречено 'sum', а ожидалось ';' |

Исправьте ошибки в коде.

9. Перепишите код программы из примера 1, так, чтобы программа выполняла поиск максимального значения в массиве целых чисел.

## 6.6. Сортировка в массиве

 При работе с массивами данных нередко возникает задача их сортировки по возрастанию или убыванию, т.е. упорядочивания. Это значит, что элементы массива нужно расположить строго по порядку. В отсортированном массиве легче производить поиск элементов.

В случае сортировки *по возрастанию/по убыванию* следующий элемент должен быть больше / меньше предыдущего. Также существую такие понятия, как сортировка по *невозрастанию / неубыванию*: следующий элемент должен быть не больше /не меньше предыдущего.

Для сортировки массивов используются разные методы.

Проще всего в программировании осуществляется сортировка выбором.

**Сортировка выбором** (например, по убыванию) осуществляется следующим образом:

1. в массиве выбирается максимальный элемент;
2. максимальный и первый элементы меняются местами (первый элемент считается отсортированным);
3. в неотсортированной части массива снова выбирается максимальный элемент; он меняется местами с первым неотсортированным элементом массива;
4. действия, описанные в п. 3, повторяются с неотсортированными элементами массива до тех пор, пока не останется один неотсортированный элемент (его значение будет минимальным).

*Чтобы поменять местами значения двух переменных нужно воспользоваться третьей переменной-буфером, например, для того, чтобы поменять местами значения i-го и j-го элементов массива, нужно выполнить следующее:*

```
buf:=a[i];          //записываем значение i-го элемента в пустой буфер
a[i]:=a[j];         //теперь спокойно записываем значение j-го элемента в i-тый
a[j]:=buf;          //записываем в j-тый элемент значение из буфера
```

**Пример. Сортировка массива по возрастанию.**

```
var                                          //блок объявления переменных
  i,j,buf,imin: integer;                     //i,j для циклов, imin для индекса мин.
  a: array[1..10] of integer;                //элемента, buf для обмена, массив a
begin                                        //начало блока операторов
  randomize;                                 //подключение генератора случайных чисел
  for i:=1 to 10 do a[i]:=random(21)-10;     //заполнение массива случайными числами
  writeln(a);                                //в диапазоне (-10;10), вывод массива
  for j:=1 to 9 do                           //первый цикл для прохода по массиву
    begin                                    //начало тела цикла
      imin:=j;                               //присваиваем индексу минимума значение j
      for i:=j+1 to 10 do                    //второй цикл для поиска минимума
        if a[i]<a[imin] then imin:=i;        //если элемент меньше, то меняем индекс
      buf:=a[j];                             //обмен значениями
      a[j]:=a[imin];
      a[imin]:=buf;
      writeln(a);                            //вывод массива
    end;                                     //конец тела цикла
end.                                         //конец блока операторов
```

**Задания**

1. Сортировка (упорядочение) массива -…
   - ▢ Удержание объекта в существующем состоянии
   - ▢ Словесная информационная модель
   - ▢ Перераспределение значений его элементов в определенном порядке
2. Как называют порядок, при котором значение каждого следующего элемента не меньше значения предыдущего элемента?
   - ▢ по возрастанию ▢ по убыванию ▢ по невозрастанию ▢ по неубыванию
3. Цель сортировки массива:
   - ▢ Перераспределить значения ▢ Работать с базами данных
   - ▢ Облегчить последующий поиск элементов

4. Дополните программу, которая сортирует массив **по убыванию** методом выбора, добавьте недостающие комментарии.

```
var                                    //_____________________________________
  i,j,buf,______: integer;             //i,j для циклов, imax для индекса макс.
  a: array[1..10] of integer;          //элемента, buf для обмена, массив a
begin                                  //_____________________________________
  randomize;                           //подключение генератора случайных чисел
  for i:=1 to 10 do a[i]:=random(21)-10; //_____________________________________
  writeln(a);                          //_____________________________________
  for j:=1 to 9 do                     //первый цикл для прохода по массиву
    begin                              //начало тела цикла
      ________:=j;                     //присваиваем индексу максимума значение j
      for i:=j+1 to 10 do              //второй цикл для поиска максимума
        if _____________ then _______; //если элемент больше, то меняем индекс
      buf:=a[j];                       //обмен значениями
      a[j]:=a[_______];                //_____________________________________
      a[_______]:=buf;                 //_____________________________________
      writeln(a);                      //_____________________________________
    end                                //конец тела цикла
end.                                   //_____________________________________
```

5. Дан код программы, в которой генерируется массив из двадцати случайных чисел в диапазоне [0;50] и затем сортируется по неубыванию.

```
var
  i,buf,imin: integer;
  a: array[1..20] of integer;
begin
  randomize;
  for i:=1 to 20 do a[i]:=random(51);
  writeln(a)
  for j:=1 to 19 do
    begin
      imin:=j;
      for i:=j+1 to 20 do
        if a[i]<a[imin] then imin:=i;
      buf:=a[j];
      a[j]:=a[imix];
      a[imin]:=buf;
    end
  writeln(a)
end.
```

При запуске произошли следующие ошибки.

| Строка | Описание |
|---|---|
| 8 | Встречено 'for', а ожидалось ';' |
| 8 | Неизвестное имя 'j' |
| 14 | Неизвестное имя 'imix' |
| 17 | Встречено 'writeln', а ожидалось ';' |

К тому же программист решил изменить диапазон случайных чисел на [1;50].

Исправьте код программы.

# 7. ЗАПИСЬ ВСПОМОГАТЕЛЬНЫХ АЛГОРИТМОВ НА ЯЗЫКЕ PASCAL

 Вспомогательные алгоритмы в языке программирования представлены процедурами и функциями. Вспомогательный алгоритм нужен тогда, когда в разных места кода необходимо выполнить одну и ту же последовательность шагов обработки данных.

**Процедура** или **функция** представляет собой последовательность операторов, которая имеет имя, список параметров и может быть вызвана из различных частей программы. Функции, в отличие от процедур, в результате своего выполнения *возвращают значение*, которое может быть использовано в выражении. Для единообразия функции и процедуры называются **подпрограммами**.

Любая используемая в программе процедура или функция должна быть предварительно описана в разделе описаний.

Её описание располагается между разделом **`var`** и программным блоком главной программы. Если подпрограмм несколько, то их описания располагаются в произвольном порядке одно за другим. Структура описания подпрограммы аналогична структуре главной программы. Описание подпрограммы начинается с заголовка и заканчивается оператором **`end`**.

Чтобы воспользоваться предварительно описанной процедурой (функцией), нужно в коде основной программы указать ее имя со списком параметров.

 **Задания**

1. При помощи чего реализуются вспомогательные алгоритмы при программировании?
   ▢ при помощи ветвлений ▢ при помощи циклов
   ▢ при помощи следований ▢ при помощи подпрограмм
2. Что такое подпрограмма?
   ▢ алгоритм, имеющий циклическую структуру ▢ именованная последовательность команд
   ▢ алгоритм, имеющий разветвленную структуру ▢ любой участок кода
3. Какие существуют подпрограммы в языке Паскаль?
   ▢ Процедуры ▢ Методы ▢ Функции ▢ Свойства ▢ Операторы
4. В какой части программы описываются подпрограммы в языке Паскаль?
   ▢ между заголовком и разделом описания
   ▢ между разделом описания и программным блоком главной программы
   ▢ в программном блоке после **`begin`**
5. Опишите структуру любой подпрограммы.

_____

_____

_____

## 7.1. Процедуры

 **Процедура** — подпрограмма, имеющая произвольное количество входных и выходных данных.

Описание процедуры начинается с ключевого слова **procedure** и имеет вид:

```
procedure имя_процедуры(описание_параметров_значений;
      var: описание_параметров_переменных);
begin
  операторы
end;
```

В заголовке процедуры после её имени приводится перечень формальных параметров и их типов. Входные параметры, значения которых не изменяются в программе, должны быть *параметрами-значениями.* Выходные (результирующие) параметры должны быть *параметрами-переменными.* Для вызова процедуры достаточно указать её имя со списком фактических параметров. Например, заголовок процедуры вычисления наибольшего общего делителя может быть описан так:

```
procedure nod (a,b:integer; var c:integer);
```

Возможны следующие варианты вызова этой процедуры:

| | |
|---|---|
| **nod(36,15,9)** | в качестве параметров значений использованы константы |
| **nod(x,y,z)** | в качестве параметров значений использованы имена переменных |
| **nod(x+y,15,z)** | в качестве параметров значений использованы выражение, константа и имя переменной |

В любом случае, между фактическими и формальными параметрами должно быть полное соответствие по количеству, порядку следования и типу.

Пример программы с процедурой, которая находит все делители введенного числа:

```
var n: integer;              //целое число
procedure del(n: integer);   //описание процедуры del, в которую передается целое число
var i: integer;
begin                        //переменная i для перебора делителей, начиная с 1
i:=1;                        //начало цикла с условие окончания работы
repeat
begin                        //если n делится на i без остатка
if n mod i = 0               //то выводим i, как один из делителей, и добавляем пробел
then write(i, ' ');          //берем следующее натуральное число
i:=i+1
end;                         //условие окончания работы цикла: делитель больше числа
until i>n
end;
begin
write('Введите число: ');
```

```
readln(n);                    //вызов функции с переданным ей числом
del(n)
end.
```

**Задания**

1. Придумайте тест для проверки работы программы из примера (нахождение делителей числа) и запишите его в таблицу.

| № | Входные данные (n) | Результат |
|---|---|---|
| 1 | *n=6* | *1, 2, 3, 6* |
| 2 | | |
| 3 | | |
| 4 | | |
| 5 | | |

2. Что пропущено в данной записи процедуры?

```
procedure test( ..................................... var t: integer);
begin
t:=x+c;
end;
```

▢ выходные параметры ▢ промежуточные переменные

▢ название процедуры ▢ входные параметры

3. Добавьте как можно более подробные комментарии к коду программы с процедурой перевода неправильной дроби в смешанное число. Пользователь вводит два числа: числитель и знаменатель. Чтобы найти целую часть, нужно воспользоваться функцией `div`, числитель смешанной дроби находится функцией `mod`. Знаменатель остается без изменений. Полученное число вывести в виде `целая часть  числитель/знаменатель`.

```
var a,b:integer;                    //_____

procedure div_mod(a,b:integer);     //_____
var nch, ost:integer;               //_____
begin                               //_____
  nch:=a div b;                     //_____
  ost:= a mod b;                    //_____
  writeln(nch, ' ', ost,'/',b);     //_____
```

```
end;                                        //_____________________________________

begin                                       //_____________________________________
  write('Введите числитель дроби: ');       //_____________________________________
  readln(a);                                //_____________________________________
  write('Введите знаменатель дроби: ');     //_____________________________________
  readln(b);                                //_____________________________________
  div_mod(a,b);                             //_____________________________________
end.                                        //_____________________________________
```

4. Придумайте тест для работы программы из задания 3 и заполните таблицу.

| № | Входные данные (a,b) | Результат |
|---|---|---|
| 1 | *a = 44, b = 6* | *7  2/6* |
| 2 | | |
| 3 | | |
| 4 | | |
| 5 | | |

5. В практике программирования часто встречается задача поменять местами значения двух переменных, поэтому чтобы сэкономить время, можно описать процедуру, которая может быть скопирована и использована при необходимости в других программах.

   Дополните код так, чтобы процедура **obmen** меняла местами значения двух введенных чисел.

   *Обмен значениями вы рассматривали в разделе "Сортировка в массиве".*

```
var n1, n2: integer;                        //объявление двух переменных для чисел
procedure obmen (var a,b: integer);         //описание процедуры
    var _____________________               //объявление переменной-буфера
    begin                                   //начало операторов процедуры
            buf := a;
            _____________________
            _____________________
    end;                                    //конец операторов процедуры
begin                                       //начало блока операторов
    writeln ('Введите два числа: ');        //вывод строки
```

```
    readln (n1, n2);                        //ввод значений
    _____________________ (n1, n2);         //вызов процедуры obmen
    writeln (n1, ', ', n2)                  //вывод чисел в строку через запятую
end.                                        //конец блока операторов
```

6. Дополните программу так, чтобы процедура **diapazon** заполняла массив из 10 элементов случайными числами (диапазон указывает пользователь).

*В данной программе в первой строке описывается тип пользователя **myarr**, как массив из десяти целых числе. Это нужно, потому что в процедуре нельзя включать описание индексов.*

В процедуру передается два числа - начало и конец диапазона x и y, и пустой массив a, который будет заполнен в процедуре. Поэтому необходимо объявить две переменные целого типа для границ диапазона и массив.

*Если возникли трудности с описанием диапазона в функции **random**, загляните в словарь в конце тетради.*

```
type myarr = array[1..10] of integer;               //описание типа myarr
var                                                 //блок переменных
    x, y: integer;                                  //начало и конец диапазона
    a: myarr;                                       //массив типа myarr
procedure arr_rand (x,y:integer; var a: myarr);     //описание процедуры
    var i: integer;                                 //переменная для цикла
    begin
        ____________________                        //подключение генератора СЧ[1]
        for i := 1 to 10 do
            a[i] := ____________________            //заполнение массива
    end;                                            //случайными числами
begin                                               //начало блока операторов
  writeln('Введите начало и конец диапазона: ');    //вывод приглашения
  ____________________                              //ввод значений x и y
  ____________________                              //вызов процедуры
  ____________________                              //вывод массива
end.                                                //конец блока операторов
```

7. Дан код программы с процедурой sqrt_eq нахождения корней квадратного уравнения.

[1] случайных чисел

```
1     var a, b, c: real;
2
3     procedure sqrt_eq(a,b,c: real);
4     var a, d, x, x1, x2: real;
5     begin
6       writeln ('Решение квадратного уравнения');
7         if a < 0
8         then _____________________________________________
9         else _____________________________________________
10      d:=b*b-4*a*c;
11      if d<0 then writeln ('Корней нет');
12      if d=0 then
13       begin
14          x:=-b/2*a;
15          writeln ('Корень уравнения x=', x:9:3)
16       end;
17      if d>0 then
18       begin
19         x1:=(-b+sqrt(d))/(2*a);
20         x2:=(-b-sqrt(d))/(2*a);
21         writeln ('Корни уравнения:');
22         writeln ('x1=', x1:9:3);
23         writeln ('x2=', x2:9:3)
24        end
25    end;
26
27    begin
28      write ('Введите коэффициенты a, b, c: ');
29      readln (a, b, c);
30      if a=0 then
31        begin
32          ____________________________________________________________
33          writeln('Корень уравнения x=',_________);
34        end;
35      else  sqrt_eq(a,b,c,d)
36    end.
```

При ее выполнении возникли следующие ошибки.

| Строка | Описание |
|---|---|
| 4 | Повторно объявленный идентификатор a |
| 35 | Встречено 'else', а ожидался оператор |
| 35 | Неизвестное имя 'd' |

При этом программист планировал еще и обеспечить вывод информации о направлении ветвей параболы: если `a<0`, то программа должна вывести строку “График - парабола, ветви вниз” (8 строка), а если `a>0`, то “График - парабола, ветви вверх” (9 строка). А если `a=0`, то необходимо вывести строку “Это не квадратное, а линейное уравнение. График - прямая” (32 строка), и найти корень линейного уравнения (33 строка). Исправьте и дополните код программы.

## 7.2. Функции

**Функция** — подпрограмма, имеющая единственный результат, записываемый в ячейку памяти, имя которой совпадает с именем функции. Поэтому в блоке функции обязательно должен присутствовать оператор <имя_функции>:=<результат>.

Описание функции начинается с ключевого слова **function** и имеет вид:

```
function имя_функции(описание_входных_данных): тип_возвращаемого_значения;
begin
      операторы;
      имя_функции:=результат
end;
```

В заголовке функции после её имени приводится описание входных данных — указывается перечень формальных параметров и их типов. Там же указывается тип самой функции, т.е. тип результата.

Для вызова функции достаточно указать её имя со списком фактических параметров в любом выражении, в условиях (после слов `if, while, until`) или в операторе `write` главной программы.

**Пример.** Программа с функцией `nok` для нахождения наименьшего общего кратного двух чисел методом перебора.

Алгоритм решения:

1) Определим максимум из двух введенных чисел `max`.
2) Запустить цикл, в котором вычисляются остатки от деления чисел от `a*b` до `max`.
3) Если в цикле встретится число, которое делится на оба введенных числа без остатка, запомнить его в результат.

```
var a,b: integer;                          //описание переменных для введенных чисел

function nok(a,b: integer): integer;       //описание функции
var i, max: integer;                       //переменные для цикла и хранения максимума
begin
  if a>b then max:=a else max:=b;          //определение наибольшего из двух чисел
  for i:=a*b downto max do                 //цикл
    if (i mod a =0) and (i mod b = 0)      //проверка остатков от деления
then nok:=i;                               //запись в результат
end;

begin
  writeln('Введите два числа: ');          //приглашение на ввод чисел
  readln(a,b);                             //ввод чисел
  writeln('НОК=',nok(a,b))                 //вызов функции в выводе строки
end.
```

**Рекурсивная функция** — функция, которой реализован способ вычисления очередного значения функции через вычисление ее предшествующих значений. Проще говоря, рекурсия - это функция, которая вызывает сама себя. Чтобы рекурсия не выполнялась бесконечно, в ее описании

обязательно должно быть условие завершения, то есть вычисление результата так, чтобы функция не вызывала себя.

**Пример. Вычисление факториала числа n.**

*С понятием факториала вы уже знакомились в разделе "Циклы".*

Алгоритм решения:

1) Если число равно нулю или единице, то результат будет равен единице - это условие завершения рекурсии.
2) Если введено число n, отличное от нуля и единицы, то находим произведение n на результат выполнения функции от числа n-1 - это шаг, где функция вызывает саму себя.

```
var n: integer;

function f(n: integer): integer;              //описание функции
begin
  if (n=0) or (n=1) then f:=1                 //условие завершения рекурсии
  else f:=n*f(n-1);                           //функция вызывает сама себя
end;

begin
  writeln('Введите число: ');
  readln(n);
  writeln('n!=',f(n))
end.
```

**Задания**

1. Какой оператор должен обязательно присутствовать в операторном блоке функции?

   ▫ любой оператор, использующий ее входные параметры

   ▫ условный оператор

   ▫ оператор цикла

   ▫ оператор присваивания имени функции результата работы подпрограммы

2. Чем функции отличаются от процедур?

   ▫ Функция принимает на вход несколько параметром, а процедура -один

   ▫ Функция возвращает значение, а процедура просто выполняет набор действий

   ▫ Функция возвращается в составе другого оператора или команды, а вызов процедуры- это отдельная команда

   ▫ Результатом работы функции может быть значение только простого типа, а процедуры- любого

   ▫ Функция может возвращать данные только логического типа, а процедура - численных

3. Что пропущено в описании функции (несколько вариантов ответа)?

```
function my_function(a,b: integer);
begin
      if a > b then a:=b else b:=a;
```

```
end;
```

▢ запись результата ▢ название функции

▢ входные параметры ▢ определение типа результата (типа функции)

4. Что будет результатом выполнения следующей функции при n=4?

```
function f(n:integer): integer;
begin
      if (n=1) or (n=0) then f:=1  else f:=n*f(n-1);
end;
```

Ответ: _____

5. В какой строке программы допущена ошибка (отметьте галочкой)?

```
▢1  var a, b : integer;
▢2    function NOD (a,b:integer):integer;
▢3       begin
▢4            if a>b      then NOD := NOD(a-b,b)
▢5              else  if a<b         then NOD := NOD (a,b-a)
▢6              else NOD := a;
▢7        end;
▢8   begin
▢9     readln (a,b);
▢10     NOD (a,b,n);
▢11           writeln ('NOD =' ,n);
▢12    end.
```

6. Дан код функции, которая считает количество цифр в введенном числе. Прокомментируйте код как можно подробнее.

```
function kol(n:integer):integer;     //_____
var i:integer;                       //_____
begin                                //_____
  i:=0;                              //_____
  while n>0 do                       //_____
  begin                              //_____
    n:=n div 10;                     //_____
    i:=i+1;                          //_____
  end;                               //_____
  kol:=i;                            //_____
end;                                 //_____
```

7. Дополните код программы, функция в которой вычисляет значение функции $y = sin(x) - cos(x)$ от введенного $x$.

```
var x:real;                          //объявление переменной
```

```
function y(x:real):real;                    //описание функции
begin
  y:=_____________________________;         //вычисление значения по формуле
end;

_________                                   //начало блока операторов
  write('Введите x: ');                     //приглашение на ввод
  _____________________________;            //ввод числа x
  write('y=', y(x));                        //вывод с вызовом функции
_________.                                  //конец блока операторов
```

8. Дан код программы, функция **s** в которой вычисляет площадь круга по заданному радиусу.

```
var r:real;

function s(r:real):real;
begin
  s:=pi*r^2;
end;

begin
  write('Введите r: ');
  readln(r);
  write('S=',s(r);
end.
```

При запуске программы произошли следующие ошибки.

| Строка | Описание |
|---|---|
| 2 | Встречено '2', а ожидалось ';' |
| 11 | Встречено ';', а ожидалось ')' |

Исправьте код.

# СЛОВАРЬ

| Служебные слова | | | |
|---|---|---|---|
| **Служебное слово** | **Значение** | **Пример использования** | |
| **and** | и | `while (i mod j <> 0) and (j <= 10) do j:=j+1;` | |
| **array** | массив | `var a:array[1..10] of integer;` | |
| **begin** | начало | `begin`<br>`  p := p * i;`<br>`  s := s + p`<br>`end` | |
| **const** | константа | `const LIMIT = 500;`<br>`var i,j,lim : word;` | |
| **do** | выполнить | `for i:=1 to 10 do`<br>`begin`<br>`  p := p * i;`<br>`  s := s + p`<br>`end` | |
| **downto** | уменьшать на 1 до | `for i:=10 downto 1 do` | |
| **else** | иначе | `if a<b then min := a else min := b;` | |
| **end** | конец | `begin`<br>`  a := 1;`<br>`  b := a;`<br>`end` | |
| **for** | для | `for i:=1 to 10 do` | |
| **if** | если | `if a<b then min := a;` | |
| **of** | из | `var a:array[1..10] of integer;` | |
| **or** | или | `if (n=1) or (n=0) then f:=1  else f:=n*f(n-1);` | |
| **procedure** | процедура | `procedure del(n: integer);`<br>`var i: integer;`<br>`begin`<br>`  ...`<br>`end;` | |
| **program** | программа | `program test;` | |
| **read** | оператор ввода | `read(i, a, k)` | С клавиатуры нужно ввести три числа, значения которых будут введены в |
| **readln** | | `readln(i, a, k)` | |

| | | | |
|---|---|---|---|
| | | | оперативную память |
| **repeat** | повторять | **repeat** read(x);<br>until x=0; | |
| **then** | то | if a<b **then** min := a; | |
| **to** | увеличивая до | for i:=1 **to** 10 do | |
| **until** | до тех пор, пока | repeat read(x);<br>**until** x=0; | |
| **var** | переменная | **var** a,b: integer;<br><br>**var** r: real; | |
| **while** | пока | **while** (i mod j <> 0) and (j <= 10) do j:=j+1; | |
| **write** | оператор вывода | **write**('s=',12) | |
| **writeln** | | **writeln**('s=',12) | |

## Некоторые типы данных

| Обозначение | Название | Допустимые значения | Область памяти |
|---|---|---|---|
| **integer** | целочисленный | -32 768 .. 32 767 | 2 байта со знаком |
| **real** | вещественный | $\pm\ (2,9 \cdot 10^{-39} .. 1.7 \cdot 10^{+38})$ | 6 байтов |
| **char** | символьный | произвольный символ алфавита | 1 байт |
| **string** | строковый | последовательность символов длиной меньше 255 | 1 байт на символ |
| **boolean** | логический | **true** или **false** | 1 байт |

## Стандартные функции

| Функция | Назначение | Тип аргумента / тип результата | Пример использования |
|---|---|---|---|
| **abs(x)** | Модуль *x* | **integer, real** / такой же как у аргумента | **abs(-4.5)**<br>Результат: 4.5 |
| **cos(x)** | Косинус *x* | **real / real** | **sin(2.0)**<br>Результат: 0.909 |
| **sin(x)** | Синус *x* | | **cos(2.0)**<br>Результат: 0.416 |
| **sqr(x)** | Квадрат *x* | **integer, real** / такой же как у аргумента | **sqr(2)**<br>Результат: 4 |

| | | | |
|---|---|---|---|
| **sqrt(x)** | Квадратный корень из *x* | **integer, real / real** | **sqrt(9)**<br>Результат: 3 |
| **round(x)** | Округление *x* до ближайшего целого | **real** | **round(6.7)**<br>Результат: 7<br>**round(6.4)**<br>Результат: 6 |
| **int(x)** | Целая часть *x* | **real** | **int(6.7)**<br>Результат: 6 |
| **frac(x)** | Дробная часть *x* | **real** | **frac(6.7)**<br>Результат: 7 |
| **random** | Случайное число от 0 до 1 | **- / real** | **a:=random;**<br>Возможный результат: в переменную **a** записано значение **0.226148907200503** |
| **random(x)** | Случайное число от 0 до *x* | **integer / integer** | **random(10)**<br>Результат: число от 0 до 9<br>**random(10)+1**<br>Результат: число от 1 до 10<br>**random(11)-10**<br>Результат: число от -10 до 0<br>**random(y-x+1)+x**<br>Результат: число от x до y |

## Операторы

| Оператор | Назначение | Тип аргумента / тип результата | Пример |
|---|---|---|---|
| **+** | сложение | **real, integer /** такой же как у аргумента | **4 + 3**<br>Результат: 7 |
| **-** | вычитание | **real, integer /** такой же как у аргумента | **9 - 11**<br>Результат: -2 |
| ***** | умножение | **real, integer /** такой же как у аргумента | **2 * 6**<br>Результат: 12 |
| **/** | деление | **real / real** | **9 / 3**<br>Результат: 3 |
| **div** | деление нацело (остаток отбрасывается) | **integer / integer** | **54 div 10**<br>Результат: 5 |
| **mod** | остаток от деления | **integer / integer** | **54 mod 10**<br>Результат: 4 |