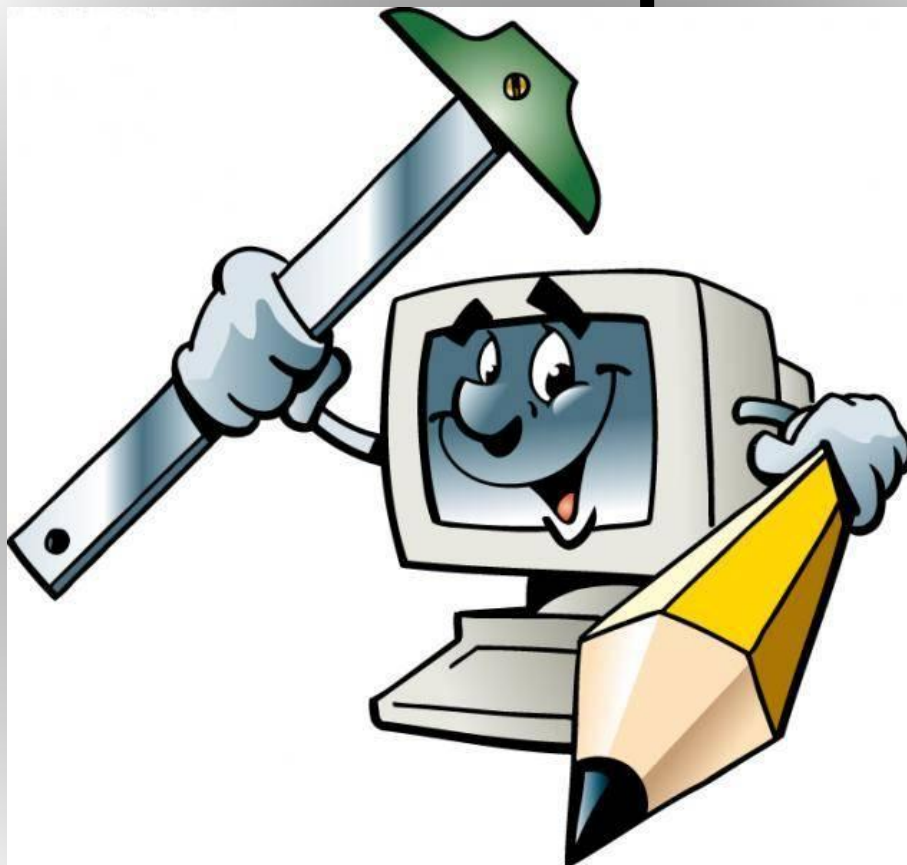


ПОДРОГРАММЫ И ФУНКЦИИ



Учитель информатики
Минина Ольга Владимировна

Подпрограммы.

Подпрограмма — последовательность операторов языка, предназначенных для решения некоторой подзадачи. В Паскале подпрограммы могут оформляться в виде **функций** и **процедур**.



```
PROGRAM main ;< Основная программа>
```

```
TYPE mas=array [1..20] of integer;
```

```
VAR A:mas;
```

```
i:byte;
```

Заголовок процедуры или функции

Раздел описания переменных

BEGIN {начало подпрограммы}

Тело подпрограммы

END ; { конец подпрограммы}

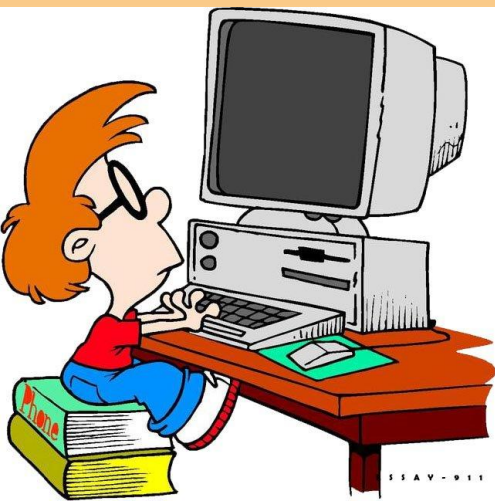
BEGIN {начало основной программы}

Тело основной программы

END. {конец основной программы}



В теле основной программы объявленные **переменные**, для которых отводится память на весь период выполнения программы, называются **глобальными**. Их можно использовать в любой подпрограмме.



Для переменных, которые объявляются в разделе описания подпрограммы, память отводится только на период выполнения подпрограммы, поэтому они называются **локальными**.

Функция – именованная часть программы, предназначенная для выполнения определенных действий вычислительного характера. Результатом работы функции может быть одно единственное данное любого скалярного типа или типа STRING. Обращение к функции (вызов функции на выполнение) осуществляется путем указания ее имени в виде операнда, в каком либо выражении.

Описание функции имеет следующую структуру:

<Заголовок функции >;

<Раздел описания>;

BEGIN

<Раздел операторов>;

END;

Формат заголовка имеет следующий вид:

Function <ИМЯ> (<список формальных параметров>):<тип>;

ИМЯ – это уникальный в программе идентификатор, по которому производится обращение к данной функции.

Список формальных параметров представляет перечень идентификаторов, являющихся аргументами данной функции.

ПРИМЕР

Function HGY (a:real; b:integer):real;

Раздел описаний включает в себя описание тех же объектов, что и раздел описания основной программы. В разделе операторов функции обязательно должен присутствовать хотя бы один оператор присвоения, в левой части которого стоит имя данной функции, при этом хотя бы один такой оператор должен обязательно выполняться при обращении к функции.

Обращение к функции из основной программы осуществляется по ее имени с указанием фактических параметров, т. е. данных, к которым применяется данная функция при данном обращении.

Формат обращения:
<ИМЯ>(<список фактических параметров>).

Типом результата в функциях может быть любой из стандартных типов Турбо Паскаля кроме файловых типов.

Пример:

**Программа вычисления членов ряда Фибоначчи
(вариант 2).**

```
Function Fibon (n:integer): integer;  
  {функция вычисления и печати чисел Фибоначчи}  
  Var fn, fn1, fn2, k:integer;  
  Begin  
    Fn1:=1;  
    Fn:=0;  
    For k:=1 to n do  
      Begin  
        Fn2:= Fn1;  
        Fn1:= Fn;  
        Fn:= Fn1 + Fn2;  
      End;  
      Fibo = Fn; {}  
    End;  
  {основная программа}  
  Var n: integer;  
  Begin  
    Write ('введите число членов ряда Фибоначчи:'); readln(n);  
    For i:= 1 to n do Writeln(Fibo(i)); {вызов функции}  
  End.
```

Процедуры.

Процедуры – это именованная часть программы, предназначенная для выполнения определенных действий (необязательно вычислительного характера).

Формат заголовка процедуры:

Procedure<ИМЯ> (список формальных параметров);

Обращение к процедуре из основной программы:

<ИМЯ>(список фактических параметров);

По способу передачи параметры в Турбо Паскале делятся на три типа:

- параметры-значения,
- параметры-переменные,
- параметры-константы.

Параметры-значения

Параметры, с помощью которых информация поступает в тело программы, называются **параметрами – значениями**.

Пример:

```
Procedure pro (a,b:integer);
```

При вызове процедур и функций формальным параметрам-значениям выделяется новое место в памяти и присваиваются значения фактических параметров. При этом на месте фактических параметров могут стоять выражения. Совместимость типов определяется возможностями присваивания.

После выполнения подпрограммы место формальных параметров освобождается. Изменение формальных параметров не сказывается на значении фактических. Заголовок процедуры с параметрами-значениями имеет вид:

Procedure MyProc1(par1,par2 : type1; par3,par4 : type2);

Список параметров задается через запятую, параметры формально определяют количество переменных, которое необходимо для обмена информацией, и их тип.

Параметры – переменные.

Формальный параметр, значение которого надо вернуть из процедуры в основную программу, называется **параметр – переменная**.

В процедуру в момент обращения передается не значение фактического параметра, а его адрес расположения в области данных основной программы, т. е. фактический параметр на время работы процедуры получает другое имя – имя соответствующего формального параметра. Само значение фактического параметра никуда не копируется и не перемещается. После выхода из процедуры это данное опять получает свое прежнее имя. Таким образом, изменение параметра – переменной в процедуре одновременно является изменением и соответствующего фактического параметра.

Заголовок процедуры с параметрами-переменными имеет вид:

```
Procedure MyProc2(var par1,par2 : type1; var par3,par4 : type2);
```

При вызове процедур и функций формальные параметры-переменные занимают то же самое место в памяти, что и соответствующие им фактические параметры. Таким образом, дополнительное место в памяти не выделяется и изменения формального параметра приводят к изменениям фактического.

Вывод : С помощью параметров – значений происходит передача в процедуру исходных данных для работы, а результаты работы процедуры возвращаются в основную программу через параметры – переменные.

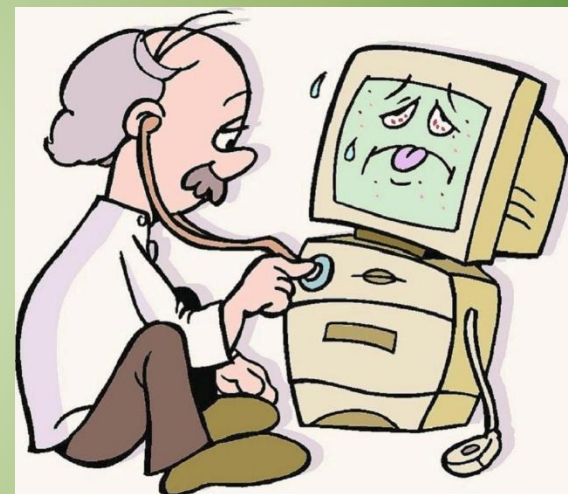
Пример:
Программа вычисления членов ряда Фибоначчи
(вариант 1).

```
Procedure fibon (n:integer);  
  {процедура вычисления и печати чисел Фибоначчи}  
  Var fn, fn1, fn2, k:integer;  
  Begin  
    Fn1:=1;  
    Fn:=0;  
    For k:=1 to n do  
      Begin  
        Fn1:= Fn2;  
        Fn1:= Fn;  
        Fn:= Fn1 + Fn2;  
        Writeln (Fn);  
      End;  
    End;  
  {основная программа}  
  Var n: integer;  
  Begin  
    Write ('введите число членов ряда Фибоначчи:'); readln(n);  
    Fibon(n); {вызов процедуры}  
  End.
```



Пример:
Программа вычисления членов ряда Фибоначчи
(вариант 2).

```
Function Fibon (n:integer): integer;  
  {функция вычисления и печати чисел Фибоначчи}  
  Var fn, fn1, fn2, k:integer;  
  Begin  
    Fn1:=1;  
    Fn:=0;  
    For k:=1 to n do  
      Begin  
        Fn2:= Fn1;  
        Fn1:= Fn;  
        Fn:= Fn1 + Fn2;  
      End;  
      Fibo = Fn; {}  
    End;  
    {основная программа}  
    Var n: integer;  
    Begin  
      Write ('введите число членов ряда Фибоначчи:'); readln(n);  
      For i:= 1 to n do Writeln(Fibo(i)); {вызов функции}  
    End.
```



Задачи

1. Вычислить значения выражения $S = \frac{a^m + b^n}{(a + b)^{m+n}}$ при задаваемых с клавиатуры вещественных значениях a и b и целых m и n .
2. Ввести с клавиатуры три матрицы R, P, M и вычислить их произведение $N = R * P * M$.
3. Написать процедуру, которая выводит на экран строку, состоящую из звездочек. Длина строки (количество звездочек) является параметром функции.
4. Написать процедуру, которая вычисляет объем и площадь поверхности параллелепипеда.

