

## ЭЛЕМЕНТЫ ЯЗЫКА ТУРБО ПАСКАЛЬ

### Структура программы:

```
program my_prog;  
uses crt;  
модулей}  
label m1;  
const n=10;  
type mytype=set of char;  
переменных}  
var a:integer; b:boolean;  
    d:real; c:char; i:2..5; m:mytype;  
-----  
функций}  
begin  
    {тело программы}  
end.
```

{Заголовок программы}  
{Список используемых

{Описание меток}  
{Описание констант}  
{Описание типов

{Описание переменных}

{Описание процедур и

{Раздел операторов}

}  
Раздел  
описани  
й

## Основные операторы языка Паскаль

**Оператор присваивания** **Имя переменной:= выражение;**

При этом тип переменной и тип выражения должны быть одинаковыми.

Например, a:=a+2; b:=true; c:='\*'; d:=4.5;

### Операторы ввода

**read (список переменных);**

**readln (список переменных);**

read(a,d);

readln(a,d);

read(a); read(d);

readln(a); readln(d);

### Операторы вывода

**write (список выражений);**

**writeln (список выражений);**

write(a,d);

writeln(a,d);

write(a:5, d:8:3);

writeln(a); writeln(d);

writeln('введите данные');

writeln('получен результат ', X);

**Clrscr** – очистка экрана

**Readkey; readln** –задержка экрана

*Пример 1. Рассмотрим пример программы, вычисляющей значение выражения  $x^3 + |x^2 - 13x + 5| - 11$ , при заданном  $x$ .*

```
Program example_1;  
Uses crt;  
Var x, y: integer;  
Begin  
  Clrscr;  
  Writeln('Введите x: ');  
  Readln(x);  
  y:= sqr(x)*x + abs(sqr(x) - 13*x + 5) - 11;  
  Writeln('значение данного выражения равно ',y);  
  Readkey;  
End.
```



# УПРАВЛЯЮЩИЕ СТРУКТУРЫ ТУРБО ПАСКАЛЯ

**Условный оператор** **If** <условие> **then** <оператор1> **else** <оператор2>;

Пример. Даны целые числа  $a$ ,  $b$ ,  $c$ . Если  $a \leq b \leq c$ , то все числа заменить квадратами, если  $a > b > c$ , то каждое число заменить наименьшим из них, в противном случае сменить знак каждого числа.

## Оператор выбора

```
case <ключ выбора> of  
  список констант 1: оператор 1;  
  список констант 2: оператор 2;  
  .....  
  список констант N: оператор N;  
else <оператор>;  
end;
```

Пример Составим программу, которая в зависимости от номера месяца печатает количество дней в нем.

## Операторы повторений (операторы циклов)

1. цикл с параметром

2. цикл с предусловием

3. цикл с постусловием

1. **for** <параметр> := <нач\_знач> **to** <кон\_знач> **do** <оператор>;

**for** <параметр> := <нач\_знач> **downto** <кон\_знач> **do** <оператор>;

2. **While** <условие> **do** <оператор>;

3. **Repeat** <тело цикла> **until** <условие>;



## *ПРОЦЕДУРЫ И ФУНКЦИИ В ТУРБО – ПАСКАЛЕ*

Подпрограмма – это часть программы, оформленная в виде отдельной синтаксической конструкции и снабженная именем.

```
Procedure <имя процедуры> (<список формальных параметров>);  
    <раздел описаний>  
begin  
    <тело процедуры>  
end;
```

**Глобальные переменные** – это переменные, объявленные в описании основной части программы и действующие в любой ее части.

**Локальные переменные** – те, которые объявлены в подпрограмме (процедуре или функции) и действующие лишь в ней.

**Фактические параметры** – это переменные, которые передаются процедуре при обращении к ней.

**Формальные параметры** – это переменные фиктивно присутствующие в процедуре и определяющие тип и место подстановки фактических параметров, над которыми производятся действия.

*Число и тип формальных и фактический параметров должны совпадать с точностью до их следования.*

**Передача данных из программы в процедуру и наоборот может быть организована двумя способами:**

через глобальные переменные

через аргументы процедуры (формальные параметры)

***Описание формальных параметров:***

список имен переменных: тип;    или    Var список имен переменных: тип;



параметры – значения (входные параметры)



параметры – переменные (выходные параметры).

**Вызов процедуры в теле программы:**

<имя процедуры> (список фактических параметров);

*Пример.* *Четырехугольник задан четырьмя своими сторонами  $a$ ,  $b$ ,  $c$ ,  $d$ , и диагональю  $f$ . С помощью процедуры вычисления площади треугольника по трем сторонам, вычислить площадь заданного четырехугольника.*

Функция – это подпрограмма, предназначенная для того, чтобы вычислять только одно значение.

Также функции отличаются от процедур:

- Заголовком;
- В теле функции обязательно должен присутствовать оператор присваивания, где в левой части стоит имя функции, а в правой – ее значение. Иначе, значение не будет определено;
- Обращением к функции не оператор, а выражение.

```
Function <имя >( <список формальных параметров> ): <тип результата>;  
    <раздел описаний>  
begin  
    <тело функции>  
    <имя>:=<значение>;  
end;
```

*Пример. Четырехугольник задан четырьмя своими сторонами  $a$ ,  $b$ ,  $c$ ,  $d$ , и диагональю  $f$ . С помощью функции вычисления площади треугольника по трем сторонам, вычислить площадь заданного четырехугольника.*



## Рекурсия в Паскале

Алгоритм называется **рекурсивным**, если в качестве вспомогательного алгоритма (подпрограммы) он использует самого себя.

$$n! = 1 * 2 * 3 * \dots * n$$

$$n! = \begin{cases} 1, & \text{если } n=1 \\ n * (n-1)!, & \text{если } n > 1 \end{cases}$$

Процедуры и функции, использующие вызовы самих себя, называют рекурсивными (прямая рекурсия).

*Пример.* Рассмотрим пример рекурсивной функции вычисления  $x^n$ , где  $n$  - натуральное число.

$$\text{Вспользуемся известным фактом: } x^n = \begin{cases} 1, & \text{при } n = 0 \\ x^{n-1} * x, & \text{при } n \geq 1 \end{cases}$$

```
function deg(x, n: integer): longint;  
begin  
  if n = 0 then deg:=1  
  else deg:=deg(x, n-1)*x;  
end;
```



## СОСТАВНЫЕ СТРУКТУРЫ В ТУРБО ПАСКАЛЕ

**Массив** – Это последовательность состоящая из фиксированного числа однотипных элементов, каждый из которых имеет свой индекс (номер).

**Строка** – это последовательность символов.

**Множество** – неупорядоченная совокупность отличных друг от друга однотипных элементов.

**Запись** — это последовательность, состоящая из фиксированного числа величин разных типов, называемых *полями* или *компонентами записи*.

Под **файлом** понимается либо именованная область внешней памяти ПК, либо логическое устройство – потенциальный источник или приемник информации.

Любой файл имеет три характерные особенности:

- у него есть имя, что дает программе возможность работать одновременно с несколькими файлами
- он содержит компоненты одного типа. Типом компонентов может быть любой тип, кроме файлов.
- длина вновь создаваемого файла никак не оговаривается при его объявлении и ограничивается только емкостью устройств внешней памяти.



*Пример. Даны два одномерных массива A и B. Найти их скалярное произведение.*

```
program scal_proiz;
uses crt;
const n=5; {n – число элементов массива}
type mas=array [1..n] of integer;
var a,b: mas; i:integer;
{----- Процедура ввода элементов массива-----}
procedure input(var c:mas);
begin
writeln('Введите ',n,' элементов массива');
for i:=1 to n do
read(c[i]);
end;
{-----Функция нахождения скалярного произведения двух массивов -----}
function sp(x,y:mas):longint;
var s:longint;
begin
s:=0;
for i:=1 to n do
s:=s+x[i]*y[i];
sp:=s;
end;
BEGIN
clrscr;
input(a); input(b);
writeln ('scal proizv mas = ',sp(a,b));
readkey;
END.
```

**Пример.** Дан двумерный массив случайных чисел. Найти максимальный элемент

```
program poisk_max;  
uses crt;  
var a: array [1..30,1..30] of integer; i, j,max,m,n:integer;
```

```
BEGIN clrscr;  
writeln('Введите размерность массива n*m');  
Readln(n,m); randomize;  
for i:=1 to n do begin  
for j:=1 to m do begin  
A[i,j]:=random(21)-10;  
Write(a[i,j]:4); end;  
Writeln; end;
```

```
Max:=-10;  
for i:=1 to n do begin  
for j:=1 to m do begin  
If a[i,j]>max then max:=a[i,j];  
Writeln('max=', max:4);
```

```
readkey;  
END.
```



*Пример 1. Подсчитать количество пробелов в данной строке.*

```
Program probel;  
Uses crt;  
Var s: string[50];  
    K, i: integer;  
Begin  
Clrscr;  
Writeln('Введите строку');  
Readln(s);  
  For i:=1 to length(s) do  
    If s[i] = ' ' then inc(k);  
Writeln('Количество пробелов = ', k);  
Readln  
End.
```

*Пример 2. Подсчитать сумму цифр в данной строке.*

```
Program summa;  
Uses crt;  
Var s: string[50];  
    K, i, n, l: integer;  
Begin  
Clrscr;  
Writeln('Введите строку');  
Readln(s);  
  For i:=1 to length(s) do  
    Begin  
      Val(s[i], n, l);  
      If l = 0 then k:=k + n;  
    End;  
Writeln('Сумма цифр = ', k);  
Readln  
End.
```

**Процедуры и функции для  
работы со строками**

## Стандартные процедуры для обработки строковых величин:

Процедура удаления *delete(s, k, n)* – из строки *s* удаляется *n* символов, начиная с *k*-того.

Пример: *s = 'барабан'; delete(s, 5, 2);* Результат *s = 'баран';*

Процедура вставки *insert(s1, s2, n)* – строка *s1* вставляется в строку *s2*, начиная с *n*-го символа.

Пример: *s = 'барабан'; insert('щик', s, 8);* Результат *s = 'барабанщик';*

Процедура *str(k, s)* преобразовывает число *k* в строку *s*.

Пример: *str(3456, s); s = '3456';*

Процедура *val(s, k, n)* преобразовывает строку из цифр *s* в число *k*. *N* – номер позиции первого символа строки *s*, отличного от цифры.

Пример: *val('3456', k, n); k = 3456; n = 0;*     *val('34g56', k, n); k = 0; n = 3;*

## Стандартные функции для обработки строковых величин:

Функция *concat(s1, s2, ..., sn)*, аналогична операции склеивания. Значение функции результат соединения строк *s1, s2, ..., sn*.

Пример: *s := concat('сегодня', ' ', '19 апреля');* Результат: *s = 'сегодня 19 апреля';*

Функция *copy(s, n, k)* – из строки *s* выделяет *k* символов, начиная с *n*-го символа.

Пример: *c := copy('барабан', 1, 3);* Результат: *c = "бар";*

Функция *length(s)* определяет длину строки *s*.

Пример: *s := 'барабан'; a := length(s); a = 7;*

Функция *pos(s, c)* определяет номер позиции, начиная с которой строка *s* первый раз входит в строку *c*.

Пример: *c := 'колокол'; k := pos('кол', c); x := pos('дол', c);* Результат: *k = 1; x = 0;*



## **Операции над множествами:**

‘+’ - операция объединения множеств.

‘-’ - операция дополнения (разности).

‘\*’ - операция пересечения множеств.

Операции отношения равенства множеств ( $A = B$ ), неравенства ( $A \neq B$ ), включения ( $A \subseteq B$ ,  $B \subseteq A$ ).

Логическая операция принадлежности (**in**):  $x \text{ in } A = \text{true}$ , если элемент  $x$  принадлежит множеству  $A$ . В противном случае  $x \text{ in } A = \text{false}$ .

### ***Примеры:***

$A = [1, 3, 4, 5, 8]$ ,  $B = [2, 4, 6, 8]$

$A + B = [1, 2, 3, 4, 5, 6, 8]$ ;

$A - B = [1, 3, 5]$ ;

$A * B = [4, 8]$ ;

$A \neq B$ ;

$3 \text{ in } A = \text{true}$ ;  $6 \text{ in } A = \text{false}$ ;

**Пример.** Даны две строки найти их общие элементы

```
Program obch;  
Uses crt;  
Var s1,s2: string[50]; a,b,c:set of char;  
    K: char; i: integer;  
Begin  
Clrscr; a:=[]; b:=[]; c:=[];  
Writeln('Введите первую строку');  
Readln(s1);  
Writeln('Введите вторую строку');  
Readln(s2);  
  For i:=1 to length(s1) do a:=a+[s1[i]];  
  For i:=1 to length(s2) do b:=b+[s2[i]];  
  c:=a*b;  
Writeln('общие элементы двух строк ');  
For k:='a' to 'z' do  
If k in c then write(k:2);  
Readln;  
End.
```





```

uses crt;
type student = record
    fio: string[20];
    gr: 11..56;
    ball: array[1..4] of 2..5;
end;
var base: array[1..1000] of student;
    n, i, j: integer;
Begin clrscr;
write('Введите количество студентов: '); readln(n);
for i:=1 to n do begin
    write('Введите фамилию ', i, '-го студента: '); readln(base[i].fio);
    write('Введите группу ', i, '-го студента: '); readln(base[i].gr);
    writeln('Введите его оценки по 4 предметам: ');
    for j:=1 to 4 do readln(base[i].ball[j]); end;
writeln('Успевающие 35 группы:');
for i:=1 to n do
with base[i] do
begin
if (gr = 35) and ((ball[1] + ball[2] + ball[3] + ball[4]) / 4 >= 4) then
begin
write(fio, ' ');
for j:=1 to 4 do write(ball[j]:3);
writeln;
end;
end; readkey
end.

```

*Сформировать базу данных о студентах математического факультета. Распечатать все сведения о студентах 35 группы со средним баллом  $\geq 4$ .*



## Типизированные файлы

**Пример 1.** Создать и сохранить в файле 'x.dat' последовательность целых чисел от 10 до 20;

```
program ex2;
uses crt;
var fi:file of integer ;
    i:integer;
Begin clrscr;  assign(fi,'x.dat'); rewrite(fi);
    for i:=10 to 20 do write(fi,i);
close(fi);
end.
```

**Пример 2.** Считать первые пять компонент из файла 'x.dat' и вывести на экран квадраты этих значений.

```
program ex3;
uses crt;
var fi:file of integer ;
    I,k:integer;
Begin clrscr;
assign(fi,'x.dat'); reset(fi);
for k:=1 to filesize(fi) do
    begin read(fi,i); write (i*i,' '); end;
close(fi); readkey;
end.
```

### Процедуры и функции

**filesize**(f) —возвращает текущее число компонент открытого файла;

**filepos**(f) —возвращает номер текущей позиции маркера;

**seek**(f,N) —устанавливает маркер на позицию N;

## Текстовые файлы

**Пример 1.** Дан текстовый файл 'text.txt'. Найти количество строк в нем и дописать в конец файла.

```
program ex2;
uses crt;
var t:text ; s,strk:string;
    k:integer;
Begin clrscr;  assign(t, 'c:\text.txt'); reset(t);
    while not eof(t) do begin readln(t,s); k:=k+1; end;
Str(k,strk)
Append(t);
Write(t,strk);
close(fi);
end.
```

### Процедуры и функции

**eof**(f)—возвращает TRUE, если найден конец файла;

**eoln**(f)—возвращает TRUE, если найден конец строки.



## Работа с графикой

Графические возможности реализованы с помощью стандартного модуля **Graph.tpu**. Подключение модуля к программе выполняется директивой **uses graph**.

Процедура инициализации графического режима имеет три аргумента:

**Initgraph**(<драйвер>, <режим>, '<путь к драйверу>')

и может быть выполнена так:

```
uses graph;
```

```
var gd, gm: integer; {переменные gd и gm определяют драйвер и режим}
```

```
begin
```

```
gd:=vga; gm:=vgahi;
```

```
initgraph(gd,gm,'d:\tp7');
```

```
.....
```

Первые две команды можно заменить одной: **gd:=detect** с целью автоматического распознавания драйвера и установления режима максимального разрешения для данной машины.

Процедура **closegraph** освобождает память от драйвера и устанавливает режим работы экрана, который был до инициализации графики.

*Пример. Построение графика функции*

**Program** grafik;

uses crt,graph;

var k,u,,gm,gd:integer;

x,y:real;

**BEGIN** gd:=detect;

    initgraph(gd,gm,' '); setlinestyle(0,0,3);

setcolor(9); line(320,10,320,400);line(10,240,620,240);

x:=-12; y:=cos(x);

u:=320+round(20\*x); t:=240-round(20\*y);

setcolor(12); moveto(u,t);

for k:=1 to 240 do

**begin**

    x:=x+0.1; y:=cos(x);

    u:=320+round(20\*x); t:=240-round(20\*y);

    lineto(u,t);

**end;**

setcolor(14);

setttextstyle(0,0,1);outtextxy(330,10,'y'); outtextxy(610,245,'x');

setttextstyle(0,0,2);

outtextxy(90,430,'y=cos(x)');

repeat until keypressed;

closegraph

**END.**



**Модуль** – программная единица, текст которой компилируется независимо (автономно). Она включает определения констант, типов данных, переменных, процедур и функций, доступных для использования в вызывающих программах. Однако внутренняя структура модуля скрыта от пользователя.

Напишем процедуру, рисующую снежинку, произвольного размера и цвета. Поместим эту процедуру в модуль snow.tpu

```
unit snow;
```

```
Interface {Интерфейсная часть}
```

```
uses graph;
```

```
var gd,gm:integer;
```

```
procedure show_sneg(x,y,color,razmer:integer);
```

```
Implementation {Исполняемая часть}
```

```
procedure show_sneg(x,y,color,razmer:integer);
```

```
begin
```

```
setcolor(color);
```

```
line(x-razmer,y,x+razmer,y);
```

```
line(x,y-razmer,x,y+razmer);
```

```
line(x-round(razmer/2),y-round(razmer/2),x+round(razmer/2),y+round(razmer/2));
```

```
line(x+round(razmer/2),y-round(razmer/2),x-round(razmer/2),y+round(razmer/2));
```

```
end;
```

```
begin {Иницилирующая часть}
```

```
gd:=detect; initgraph(gd,gm,' ');
```

```
end.
```

Напишем программу «снегопад»- появление снежинок случайного цвета и размера в случайном месте экрана.

```
program snows;  
uses graph,crt,snow;  
var x,y,cvet,r:integer;  
begin  
  randomize;  
  repeat  
    cvet:=random(14)+1; r:=random(20)+5;  
    x:=random(600); y:=random(400);  
    show_sneg(x,y,cvet,r); delay(2000);  
    { show_sneg(x,y,0,r);    - «стирать» снежинки с экрана }  
  until keypressed;  
  closegraph;  
end.
```



```
Program ex_3;
Uses crt;
Var a, b, c: integer;
Begin
Clrscr;
Writeln('Введите числа: ');
Readln(a, b, c);
If (a <= b) and (b <= c) then
  Begin
    a:= sqr(a); b:= sqr(b); c:= sqr(c);
  End
Else
  If (a > b) and (b > c) then
    Begin
      a:=c; b:=c;
    End
  Else
    Begin
      a:=-a; b:=-b; c:=-c;
    End;
Writeln('a =', a, ' b =', b, ' c =', c);
Readkey;
End.
```





```
Program ex_4;  
Uses crt;  
Var n: integer;  
Begin  
  Clrscr;  
  Write('Введите номер месяца: ');  
  Readln(n);  
  Case n of  
    1, 3, 5, 7, 8, 10, 12: writeln('В этом месяце 31 день');  
    4, 6, 9, 11: writeln('В этом месяце 30 дней');  
    else writeln('В этом месяце 28 дней');  
  end;  
  Readkey;  
End.
```



```
Program pl;  
Uses crt;  
Var a, b, c, d, f, s1, s2, s: real;  
Procedure treug(x, y, z: real; var v: real);  
  Var p: real;  
  Begin  
    P:=(x + y + z)/2;  
    V:=sqrt(p*(p - x)*(p - y)*(p - z));  
  End;  
BEGIN  
  Clrscr;  
  Writeln('Введите стороны четырехугольника и диагональ: ');  
  Readln(a, b, c, d, f);  
  Treug(a, b, f, s1);  
  Treug(c, d, f, s2);  
  S:=s1 + s2;  
  Write('Площадь четырехугольника = ', s:5:2);  
  Readkey;  
END.
```



```
Program pl2;  
Uses crt;  
Var a, b, c, d, f, s: real;  
Function PL_t(x, y, z: real):real;  
  Var p: real;  
  Begin  
    P:=(x + y + z)/2;  
    Pl_t:=sqrt(p*(p - x)*(p - y)*(p - z));  
  End;  
BEGIN  
  Clrscr;  
  Writeln('Введите стороны четырехугольника и диагональ: ');  
  Readln(a, b, c, d, f);  
  S:=Pl_t(a,b,f)+Pl_t(c,d,f);  
  Write('Площадь четырехугольника = ', s:5:2);  
  Readkey;  
END.
```

