

Севастопольский государственный университет  
Кафедра информационных систем

Курс лекций по дисциплине

**«АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ»**

(АиТ)

Лектор: Бондарев Владимир Николаевич

## Лекция 8

# Структурное программирование: следование, ветвление, циклы

# Следование и составной оператор

Конструкция «следование» реализуется в Паскале с помощью составного оператора, который объединяет в себе несколько операторов, выполняющихся в порядке следования.

**<составной оператор> ::= begin <оператор> { ; <оператор> } end**

**В Паскале точка с запятой используется как разделитель между операторами, т.е. она не входит в оператор!**

**Обратите внимание**, что раздел операторов программы – один составной оператор.

**Пример:**

**begin**

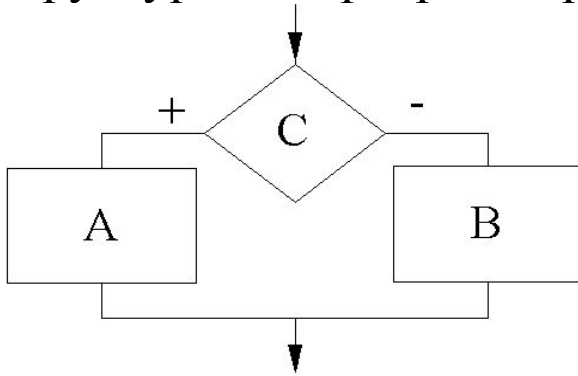
**sum:=3+5;**

**writeln('Sum=', sum) ; {Пустой оператор!}**

**end.**

# Условный оператор

**Условный оператор** языка Паскаль реализует управляющую конструкцию структурного программирования «**если-то-иначе**»:



**если** условие C **то**

действие A

**иначе** действие B

**<условный оператор> ::= if <логическое выражение> then <оператор1>  
[else <оператор2 >]**

Если требуется **проверить несколько условий**, то используют либо логические операции, либо вложение условных операторов.

**Примеры:**

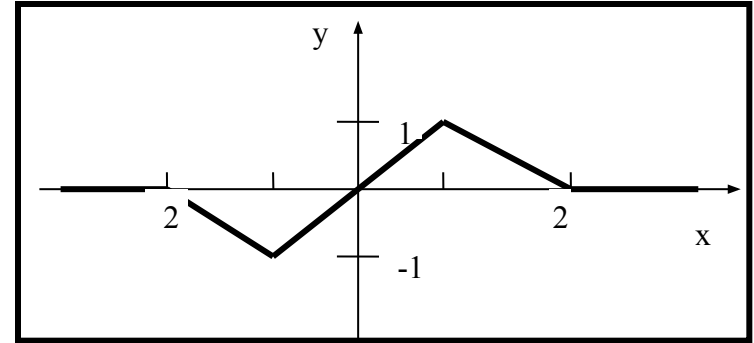
**if x > y then n := x else n := y;**

**if a >= b then begin r:=a+b; c:=k+d end  
else begin r:=a-b; c:=d-k end;**

**if (a<b) and ((a>d) or (a=0)) then b:=b+1 else begin b:=b+a; a:=0 end;**

## Пример вычисления функции

$$y = \begin{cases} 0, & x < -2 \\ -x - 2, & -2 \leq x < -1 \\ x, & -1 \leq x < 1 \\ -x + 2, & 1 \leq x < 2 \\ 0, & x \geq 2 \end{cases}$$



```
program clcfuntion1;  
{с помощью лог. операций}  
var x, y : real;  
begin  
  writeln('Введите x');  
  readln(x);  
  if x < -2 then y:= 0;  
  if (-2 <= x) and (x < -1) then y:= -x-2;  
  if (-1 <= x) and (x < 1) then y:= x;  
  if (1 <= x) and (x < 2) then y:= -x+2;  
  if x >= 2 then y:= 0;  
  writeln('Для x=', x:6:2, 'y=', y:6:2);  
end.
```

```
program clcfuntion2;  
{вложенные условные операторы }  
var x, y : real;  
begin  
  writeln('Введите x');  
  readln(x);  
  if x < -2 then y:= 0  
  else if x < -1 then y:= -x-2  
    else if x < 1 then y:= x  
      else if x < 2 then y:= -x+2  
  writeln('Для x=', x:6:2, 'y=', y:6:2);  
end.
```

# Оператор выбора (варианта)

Данный оператор позволяет выбрать одну из n ветвей алгоритма

**case** <ординальное выражение> **of**

<список меток варианта 1> : <оператор1>;

<список меток варианта 2> : <оператор2>;

<список меток варианта 3> : <оператор3>;

...

<список меток варианта n> : <оператор n>;

**end**

<список меток варианта > ::= <константа> {, <константа>}

**Примеры.**

**case i of**

**0: y := x;**

**1: y := abs (x);**

**2: y := exp (x);**

**3: y :=sin(x)**

**end;**

**case ch of**

**'a', 'A': name := 'эй';**

**'b', 'B': name := 'би';**

**'d', 'D': name:= 'ди';**

**end;**

# Оператор перехода

**Оператор перехода** – оператор, указывающий, что дальнейшая работа должна продолжаться в другой части текста программы, а именно с того места, где стоит метка.

**<оператор перехода> ::= goto <метка>**

Правила использования меток:

- 1) метка должна быть описана в разделе меток;
- 2) метка должна стоять только перед **одним** оператором;
- 3) переходы внутрь сложного оператора извне запрещены.

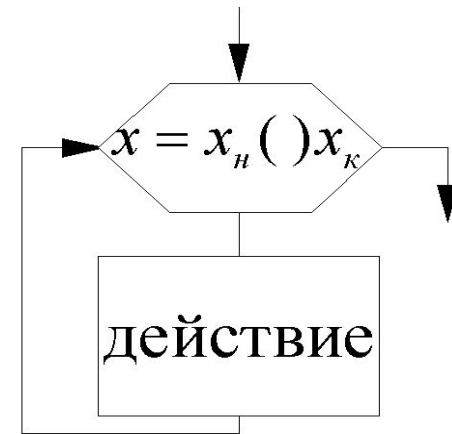
**Применение оператора перехода на метку оправдано в случае необходимости выхода из глубоко вложенных циклов.**

```
label 3, 5;  
begin  
  ...  
  goto 5;  
3: writeln('hello');  
5: a:=c+1;  
  ...
```

# Оператор цикла с параметром

Такой оператор предусматривает повторное выполнение некоторого оператора для каждого очередного значения переменной (параметра) цикла и соответствует конструкции псевдоязыка:

**для**  $x := x_{нач}$  **до**  $x_{кон}$  **повторять** действие;



В языке Паскаль оператор определяется следующим образом:

**for** <переменная> := <выражение 1> **to** <выражение2> **do**<оператор>

1. **Переменная** цикла должна относиться к **ординальному типу** и быть описана в том же блоке, где появляется сам оператор цикла.
2. **Начальное и конечное значения**, определяемые выражениями 1 и 2, вычисляются при входе в цикл единожды и должны относиться к **ординальному типу, совместимому с переменной цикла**. Если начальное значение больше конечного, то цикл выполняться **не будет**.
3. Очередное **значение переменной цикла определяется автоматически** с помощью функции **succ(x)**. Внутри цикла переменная цикла изменяться не должна. Значение переменной цикла при выходе из цикла остается неопределенным.



# Примеры на цикл с параметром

```
program stepen;
```

```
{Программа возводит число «а» в степень «n»,  
последовательно умножая  $a*a...*a$ , всего n раз}
```

```
var i, n : integer;
```

```
    a, p : real;
```

```
begin
```

```
  writeln('Введите основание "а" и степень "n" ');
```

```
  readln (a, n);
```

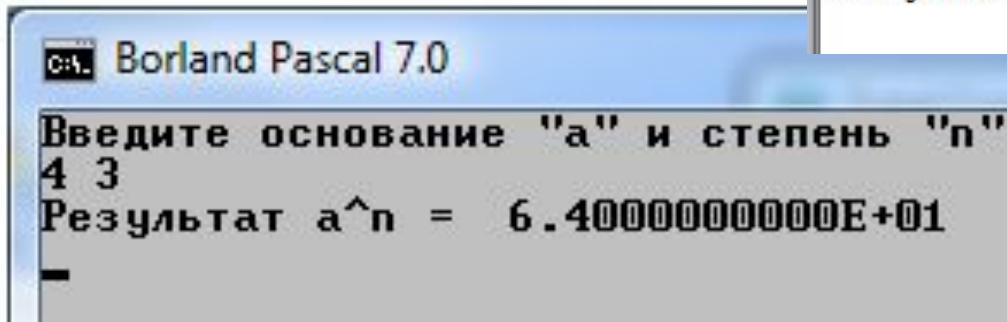
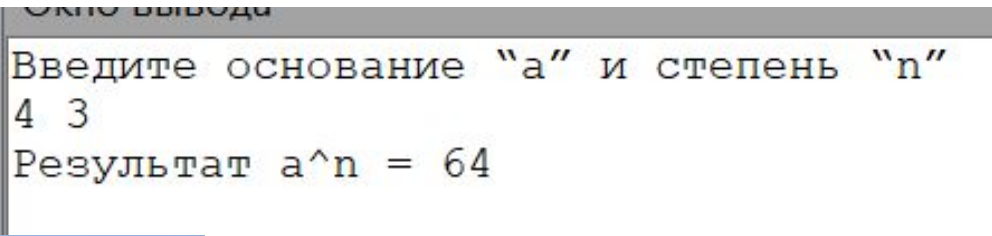
```
  p := 1;           {Начальное значение произведения!}
```

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

```
      p := p * a;
```

```
  writeln ('Результат  $a^n =$  ', p);
```

```
end.
```



# Примеры на цикл с параметром

В языке Паскаль имеется вариант цикла с параметром, в котором переменная цикла меняется от большего значения к меньшему:

**for** <переменная> := <выражение 1> **downto** <выражение2> **do**  
<оператор>

**program** **summa**;

{Вычисление суммы ряда  $H = 1 + 1/2 + 1/3 + \dots + 1/N$ }

**var**

**i, n : integer;**

**h : real;**

**begin**

**writeln**('Введите N');

**readln** (n);

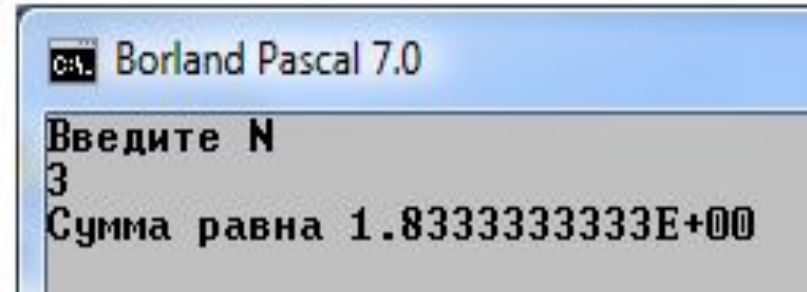
**h := 0;**

**for i := n downto 1 do**

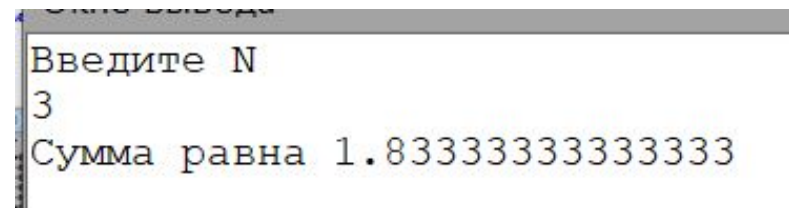
**h := h + 1/i;**

**writeln**('Сумма равна ', h)

**end.**



```
Borland Pascal 7.0
Введите N
3
Сумма равна 1.8333333333333333E+00
```



```
Введите N
3
Сумма равна 1.8333333333333333
```

# Примеры на цикл с параметром

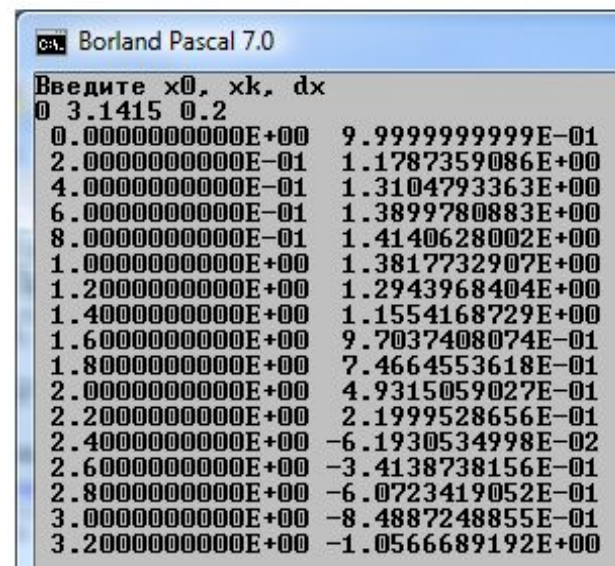
Программа табулирования функции:

$$y := \sin(x) + \cos(x)$$

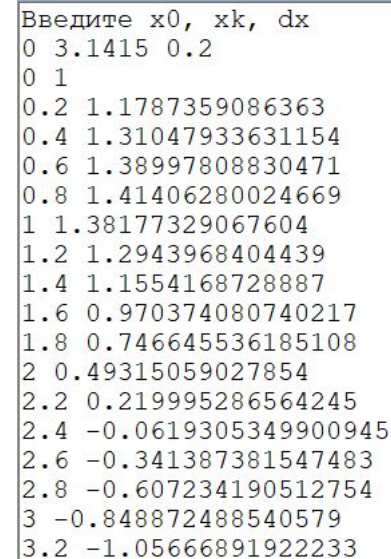
$x \in [x_0, x_k]$  – интервал изменения  $x$

$N = \text{целая часть } [(x_k - x_0)/dx] + 1$  – число повторений

```
program TabFunction;
var x0, xk, dx, y, x : real;
    i, n : integer;
begin
  writeln('Введите x0, xk, dx');
  readln(x0, xk, dx);
  n := trunc((xk-x0)/dx) + 1;
  x := x0;
  for i := 0 to n do
    begin
      y := sin(x) + cos(x);
      writeln(x, ' ', y);
      x:=x+dx
    end
  end.
```



```
Borland Pascal 7.0
Введите x0, xk, dx
0 3.1415 0.2
0.0000000000E+00  9.9999999999E-01
2.0000000000E-01  1.1787359086E+00
4.0000000000E-01  1.3104793363E+00
6.0000000000E-01  1.3899780883E+00
8.0000000000E-01  1.4140628002E+00
1.0000000000E+00  1.3817732907E+00
1.2000000000E+00  1.2943968404E+00
1.4000000000E+00  1.1554168729E+00
1.6000000000E+00  9.7037408074E-01
1.8000000000E+00  7.4664553618E-01
2.0000000000E+00  4.9315059027E-01
2.2000000000E+00  2.1999528656E-01
2.4000000000E+00 -6.1930534998E-02
2.6000000000E+00 -3.4138738156E-01
2.8000000000E+00 -6.0723419052E-01
3.0000000000E+00 -8.4887248855E-01
3.2000000000E+00 -1.0566689192E+00
```

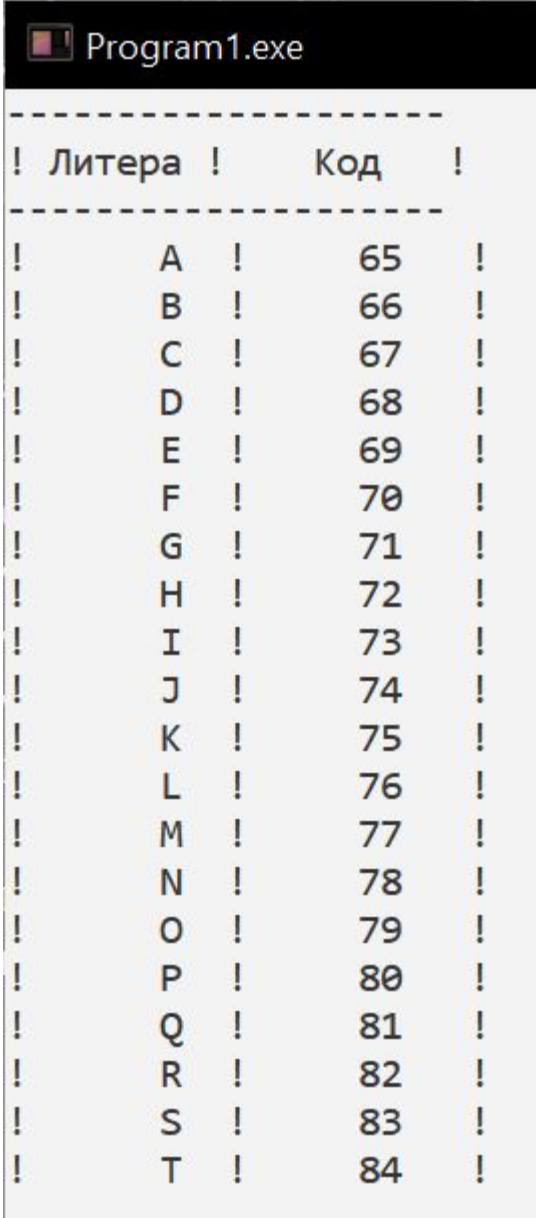


```
Введите x0, xk, dx
0 3.1415 0.2
0 1
0.2 1.1787359086363
0.4 1.31047933631154
0.6 1.38997808830471
0.8 1.41406280024669
1 1.38177329067604
1.2 1.2943968404439
1.4 1.1554168728887
1.6 0.970374080740217
1.8 0.746645536185108
2 0.49315059027854
2.2 0.219995286564245
2.4 -0.0619305349900945
2.6 -0.341387381547483
2.8 -0.607234190512754
3 -0.848872488540579
3.2 -1.05666891922233
```

# Примеры на цикл с параметром

Требуется напечатать коды литер от А до Т.

```
program KodCharacters;  
uses crt;  
var c : char;  
begin  
  clrscr;  
  writeln('-----');  
  writeln('! Литера ! ', ' Код  !' );  
  writeln('-----');  
  for c:= 'A' to 'T' do  
    writeln('! ', c:6, ' ! ', ord(c):6, ' ! ');  
  readkey  
end.
```



```
Program1.exe  
-----  
! Литера !      Код  !  
-----  
!      A  !      65  !  
!      B  !      66  !  
!      C  !      67  !  
!      D  !      68  !  
!      E  !      69  !  
!      F  !      70  !  
!      G  !      71  !  
!      H  !      72  !  
!      I  !      73  !  
!      J  !      74  !  
!      K  !      75  !  
!      L  !      76  !  
!      M  !      77  !  
!      N  !      78  !  
!      O  !      79  !  
!      P  !      80  !  
!      Q  !      81  !  
!      R  !      82  !  
!      S  !      83  !  
!      T  !      84  !
```

# Цикл с постусловием

Этот цикл соответствует циклу, управляемому условием «до»:

**повторять**

действие А

**до** условие С;

На Паскале данный цикл записывается в виде:

**repeat**

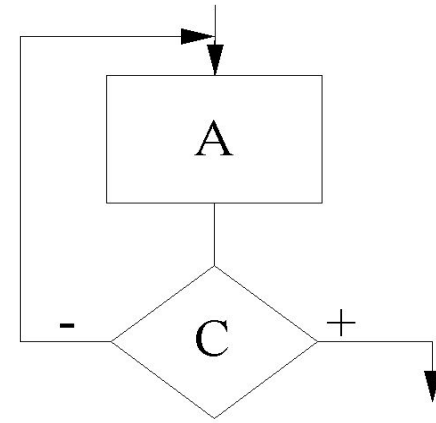
<оператор 1>;

<оператор 2>;

...

<оператор n>

**until** <лог. выражение>

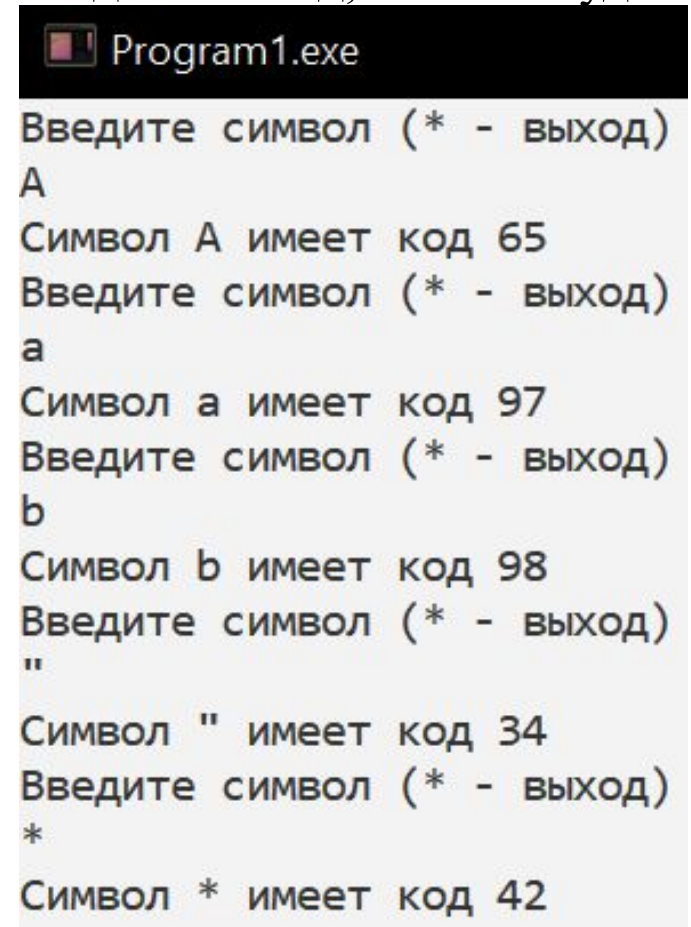


Последовательность операторов, образующих тело цикла, выполняется хотя бы один раз. После каждого выполнения последовательности операторов вычисляется логическое выражение. Повторения продолжаются до тех пор, пока выражение не получит значение true.

# Примеры на цикл с постусловием

Требуется написать программу, которая циклически выполняет ввод символа с клавиатуры и для каждого символа выводит его код, пока не будет введен символ '\*'.  
\*

```
program CalcSymbolCode;  
{Ввод и печать кодов символов,  
пока не будет введен символ '*'}  
var sym : char;  
begin  
  repeat  
    writeln('Введите символ (* - выход)');  
    readln(sym); {read – не годится!}  
    writeln ('Символ ', sym, ' имеет код ',  
            ord(sym))  
  until sym = '*';  
end.
```



```
Program1.exe  
Введите символ (* - выход)  
A  
Символ A имеет код 65  
Введите символ (* - выход)  
a  
Символ a имеет код 97  
Введите символ (* - выход)  
b  
Символ b имеет код 98  
Введите символ (* - выход)  
"  
Символ " имеет код 34  
Введите символ (* - выход)  
*  
Символ * имеет код 42
```

**read(sym)** – будет читать коды 13 и 10, связанные с нажатием Enter.

# Примеры на цикл с постусловием

Вычислить  $y = \sqrt[n]{x}$  с точностью  $\varepsilon$ .

Будем использовать рекуррентную формулу:

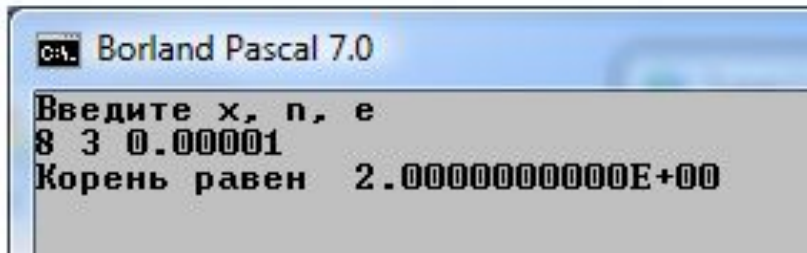
$$y_0 = x; \quad y_k = y_{k-1} + \frac{1}{n} \left( \frac{x}{y_{k-1}^{n-1}} - y_{k-1} \right)$$

**Рекуррентной** называется формула, которая выражает один элемент последовательности через значения предыдущих элементов.

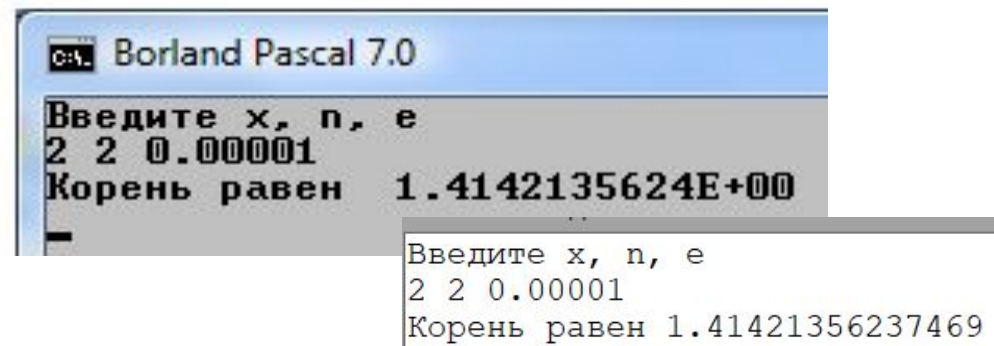
Вычисления заканчиваются когда

$$d = |y_k - y_{k-1}| < \varepsilon$$

```
program nroot;  
{вычисление корня n-ой степени}  
var      n : integer;  
        y, x, d, e : real;  
begin  
  writeln('Введите x, n, e');  
  read (x, n, e);  y:=x;  
  repeat  
    d:= (x/exp ((n - 1) * ln (y)) - y)/n;  
    y:= y + d;  
  until abs(d) < e;  
  writeln ('Корень равен ', y);  
  readln;  
end.
```



```
Borland Pascal 7.0  
Введите x, n, e  
8 3 0.00001  
Корень равен 2.000000000000E+00
```



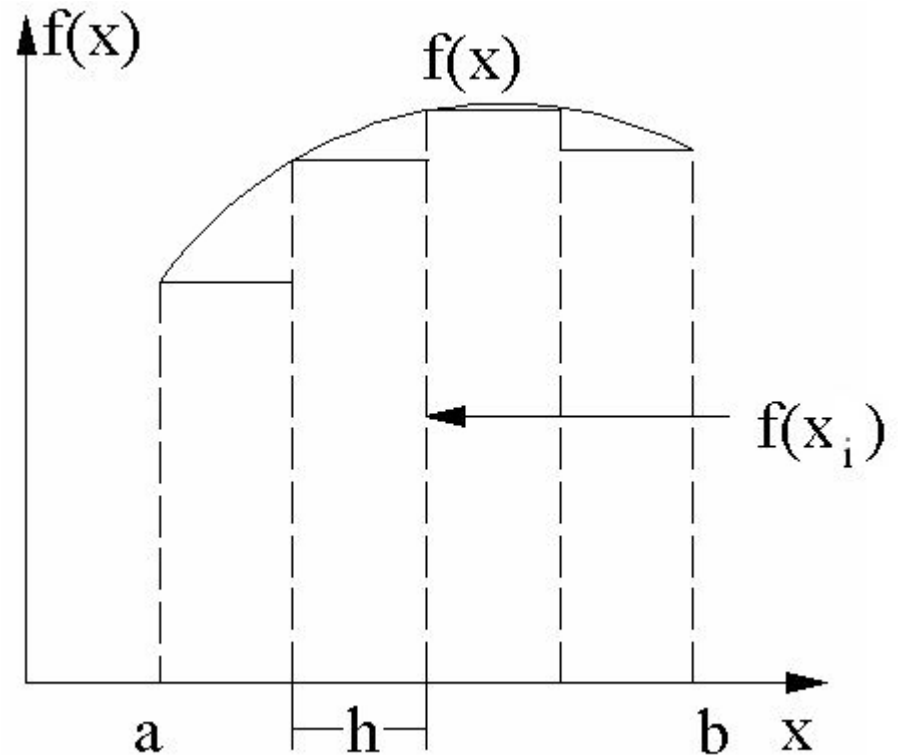
```
Borland Pascal 7.0  
Введите x, n, e  
2 2 0.00001  
Корень равен 1.4142135624E+00  
-  
Введите x, n, e  
2 2 0.00001  
Корень равен 1.41421356237469
```

# Пример на вложенные циклы

Вычисление определенного интеграла:

$$\text{Int} = \int_a^b f(x) dx \approx \sum_{i=1}^n f(x_i) \cdot h$$

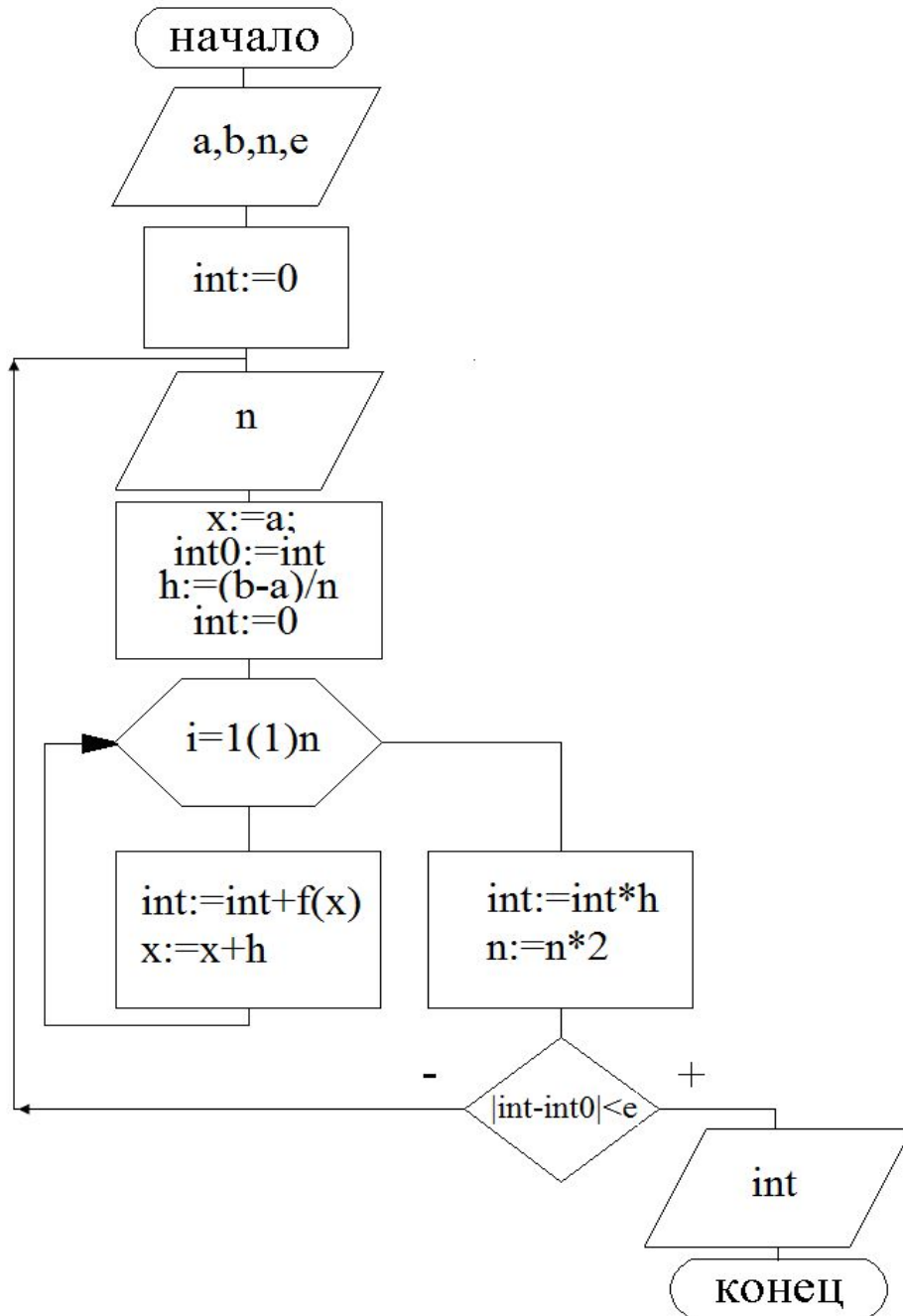
```
x:=a;  
h:=(b-a)/n;  
int:=0;  
for i:=1 to n do  
begin  
  int:=int+f(x)*h;  
  x:=x+h  
end;
```



Вычисления необходимо выполнять **с заданной точностью**. Для этого запоминают вычисленное значение **int**, например в переменной **int0**, и повторно вычисляют интеграл, увеличив число интервалов в 2 раза: **n:=n\*2**. Вычисления заканчивают, когда выполнится условие  $|\text{int}-\text{int0}| \leq e$ , где **e** – точность.



# Пример на вложенные циклы



```
program integral;
var      i, n : longint;
         a, b, e, h, x, int, int0 : real;
begin
  readln(a, b, n, e);
  int:=0;
  repeat
    writeln('n=',n); {КОНТР. ВЫВОД}
    x:=a; int0:=int;
    h:=(b-a)/n; int:=0;
    for i:=1 to n do
      begin
        int:=int+sin(x);
        x:=x+h
      end;
    int:=int*h;
    n:=n*2;
  until abs(int-int0)<=e;
  writeln('int=', int)
end.
```

# Пример на вложенные циклы

Простой тест:

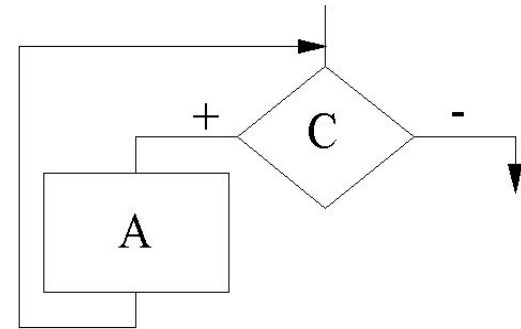
$$\text{Int} = \int_0^{\pi} \sin(x) dx = -\cos(x) \Big|_0^{\pi} = -(-1 - 1) = 2.$$

```
0 3.1415 5 0.00001
n=5
n=10
n=20
n=40
n=80
n=160
n=320
n=640
n=1280
int=1.99999887807899
```

# Цикл с предусловием

Этот цикл соответствует циклу, управляемому условием «пока»:

**пока** условие С **повторять**  
действие А;



На Паскале данный цикл записывается в виде:

**while** <условие> **do** <оператор>;

Оператор, идущий за словом **do**, выполняется нуль или более раз.

Условие (логическое выражение) выполнения цикла проверяется до выполнения оператора. Если его значение **true**, оператор выполняется.

Поскольку выражение вычисляется при каждой итерации, его следует делать насколько возможно простым.

# Пример на цикл с предусловием

Требуется вычислить  $\cos(x)$  с заданной точностью. Воспользуемся представлением функции  $\cos(x)$  в виде **суммы элементов степенного ряда**:

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots = \sum_{n=0}^{\infty} t_n(x)$$

Выведем **рекуррентную** формулу для элемента ряда:

$$t_n(x) = (-1)^n \frac{x^{2n}}{2n!}$$

$$t_{n-1} = (-1)^{n-1} \frac{x^{2(n-1)}}{[2(n-1)]!}$$

$$t_n = t_{n-1} \cdot \varphi_n$$

$$\varphi_n = \frac{t_n}{t_{n-1}} = \frac{(-1)x^2(2n-2)!}{(2n)!} = -\frac{x^2}{(2n-1) \cdot 2n}$$

Вычисления прекращаются, когда  $|t_n| \leq \varepsilon$

# Пример на цикл с предусловием

```
program mycos;
```

```
{вычисление  $\cos(x)$  с заданной точностью "e"}
```

```
var
```

```
  n:integer;
```

```
  s, x, t, e, f: real;
```

```
begin
```

```
  writeln('Введите x и точность e');
```

```
  readln(x, e);
```

```
  s := 0;
```

```
  t := 1;
```

```
  n := 0;
```

```
  while abs (t) > e do
```

```
  begin
```

```
    s:=s+t;
```

```
    n := n + 1;
```

```
    f := -sqr (x)/((2 * n - 1) * 2 * n);
```

```
    t := t * f;
```

```
  end;
```

```
  writeln('x=', x:10:8, ' s=', s:10:8, ' cos(x)=', cos(x) :10:8);
```

```
end.
```

```
Введите x и точность e
```

```
3.1415 0.0001
```

```
x=3.14150000 s=-1.00000416 cos(x)=-1.00000000
```

```
Введите x и точность e
```

```
1.57 0.000001
```

```
x=1.57000000 s=0.00079586 cos(x)=0.00079633
```

# Прерывания циклов

В некоторых случаях необходимо завершить цикл раньше, чем выполнится условие завершения цикла, или прервать очередную итерацию цикла и перейти к следующей итерации.

**Это всегда можно сделать, усложнив условия завершения цикла или добавив дополнительные операторы ветвления в тело цикла.**

Для этого можно использовать процедуры **break** и **continue**.

**break** – завершает выполнение цикла, внутри которого записана.

**continue** – выполняет переход к следующей итерации цикла.

Пусть требуется вычислить **гиперболический косинус** с заданной точностью:

$$ch(x) = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \dots = \sum_{n=0}^{\infty} t_n(x)$$

Рекуррентная формула для элемента ряда:  $t_n = t_{n-1} \cdot \varphi_n$

$$\varphi_n = \frac{t_n}{t_{n-1}} = \frac{x^2 (2n-2)!}{(2n)!} = \frac{x^2}{(2n-1) \cdot 2n}$$

# Прерывания циклов

Для вычисления суммы ряда можно использовать **циклический алгоритм**, по аналогии с предыдущим примером. Однако в циклах такого рода есть опасность, что он никогда не завершится, как из-за возможных ошибок вычислений, так и из-за ограниченной сходимости ряда. В данном случае значения функции при увеличении абсолютного значения аргумента  $x$  **очень сильно возрастают** и могут переполнить разрядную сетку, т.к.

$$ch(x) = (e^x + e^{-x}) / 2$$

Поэтому для надежности программы необходимо предусмотреть **аварийный выход** из программы с выдачей соответствующего сообщения по достижении некоторого максимально возможного числа итераций.

Для этого введем в программе константу **MaxIter = 200**. Если  $n$  будет превышать это значение, то будем прерывать выполнение цикла с помощью процедуры **break** и выводить аварийное сообщение.

```

program hypercos;
{вычисление ch(x) с точностью "e"}
const MaxIter=200;
var  n:integer; done: boolean;
      s, x, t, e, f : real;
begin
  writeln('Введите x и точность e');
  readln(x, e);
  s := 1; t := 1; n := 1; done:=true;
  while abs (t) > e do
  begin
    f := sqr (x)/((2 * n - 1) * 2 * n);
    t := t * f; s:=s+t; n := n + 1;
    if n>MaxIter then begin
      writeln('Ряд расходится');
      done:=false; break;
    end
  end;
  if done then
    writeln('x=', x, '#13#10', 'ch(x)=', s, ' n= ', n-1);
  end.

```

```

Введите x и точность e
85 0.000001
x=85
ch(x)=4.11150635731146E+36 n= 121

```

```

Введите x и точность e
1000 0.000001
Ряд расходится

```

Пример использования **continue**:

```

if n<=MaxIter then continue;
writeln('Ряд расходится');
done:=false; break;

```



**uses Crt;**

**var**

**C: Char;**

**begin**

**Writeln('Please press a key');**

**C := Readkey;**

**Writeln(' You pressed ', C, ', whose ASCII value is ', Ord(C), '.');**

**end.**

```
Please press a key  
You pressed f, whose ASCII value is 102.
```

```
Please press a key  
You pressed A, whose ASCII value is 65.
```

```
Please press a key  
You pressed ←, whose ASCII value is 27.
```

```
uses Crt;  
var C: Char;  
begin  
  while true do  
    begin  
      Writeln('Please press a key (* - exit)');  
      C := Readkey;  
      if c = '*' then break;  
      Writeln(' You pressed ', C, ', whose ASCII value is ', Ord(C), '.');  
    end;  
  end.  
end.
```

```
uses Crt;  
var C: Char;  
begin  
  repeat  
    Writeln('Please press a key (* - exit)');  
    C := Readkey;  
    if c = '*' then break;  
    Writeln(' You pressed ', C, ', whose ASCII value is ', Ord(C), '.');  
  until false;  
end.
```