

**Реализация циклических
алгоритмов в языке Паскаль.
Примеры.**

Цикл с параметром («для»)

Цикл с параметром в Паскале имеет две формы:

- 1) *for* $\langle \text{параметр} \rangle := \langle \text{НЗ} \rangle$ *to* $\langle \text{КЗ} \rangle$ *do* $\langle \text{ТЦ} \rangle$;
- 2) *for* $\langle \text{параметр} \rangle := \langle \text{НЗ} \rangle$ *downto* $\langle \text{КЗ} \rangle$ *do* $\langle \text{ТЦ} \rangle$;

Здесь $\langle \text{ТЦ} \rangle$ (тело цикла) – простой или составной оператор. Если он составной, то нужно не забывать ставить операторные скобки *begin-end*;

$\langle \text{параметр} \rangle$ - это переменная любого простого скалярного типа, кроме вещественного;

$\langle N3 \rangle$ и $\langle K3 \rangle$ - начальное и конечное значения диапазона изменения параметра, причем начальное значение может быть как меньше конечного, так и больше:

to - используется, если $N3 < K3$,

downto – используется, если $N3 > K3$.

Соответственно, при использовании служебного слова *to* при каждом следующем повторении цикла параметр изменяется по закону:

$$\langle \text{параметр} \rangle := \text{succ}(\langle \text{параметр} \rangle);$$

При использовании *downto* – по закону:

$$\langle \text{параметр} \rangle := \text{pred}(\langle \text{параметр} \rangle).$$

Очевидно, что в том случае, когда параметром цикла является переменная целого типа, значение параметра изменяется либо с шагом +1 (*to*), либо -1 (*downto*), то есть параметр выступает в качестве счетчика числа повторений цикла.

Таким образом, тело цикла выполняется столько раз, сколько значений элементов соответствующего параметру типа лежат в диапазоне (на отрезке) $\langle N3 \rangle - \langle K3 \rangle$.

Пример 1. Составить программу, последовательно выводящую на экран малые буквы латинского алфавита: сначала от начала к концу алфавита, а потом – наоборот:

Program cikl_param;

var c:char; *{с-параметр цикла}*

Begin

Writeln('латинский алфавит по возрастанию');

For c:='a' to 'z' do write(c:2);*{отводим 2 позиции на символ, чтобы между буквами был пробел}*

Writeln;*{переводим строку}*

Writeln('латинский алфавит по убыванию');

For c:='z' downto 'a' do write(c:2);

Writeln

end.

Пример 2. Составить программу построения таблицы значений функции $y=\sin(x)$ на отрезке $[a,b]$ с шагом h .

$$N = \text{trunc}((b-a)/h) + 1;$$

```
Program tab_func;  
  var x,y,a,b,h : real; i,n : integer;  
begin  
  write('введите границы отрезка и шаг: ');  
  readln(a,b,h);  
  n:= trunc((b-a)/h) + 1;
```

```
x:=a; {начальное присваивание}  
  for i:=1 to n do  
    begin {начало тела цикла}  
      y:=sin(x);{вычисление функции}  
      writeln(x:6:2, y:6:2);{вывод на экран}  
      x:=x+h; {изменение значения x}  
    end;{конец цикла}  
end.
```


Протокол работы программы:

введите границы отрезка и шаг: -2 2 0.5

-2.00 -0.91

-1.50 -1.00

-1.00 -0.84

-0.50 -0.48

0.00 0.00

0.50 0.48

1.00 0.84

1.50 1.00

2.00 0.91

Циклы с предусловием («пока») и постусловием («до»)

Цикл «пока» в языке Паскаль имеет следующий *формат*:

while <булевское выражение> *do* <ТЦ>;

здесь

<ТЦ>- тело цикла – простой или составной оператор (в случае составного не забывайте использовать операторные скобки).

Пример1. Вычислить сумму $S=1+2+3+4+\dots$. Вычисления прекратить, как только значение S станет больше заданного числа x (x вводится с клавиатуры). Дополнительно определить, сколько слагаемых просуммировано.

Тело цикла имеет вид **$S:=S+k$** .

До начала цикла следует выполнить начальное присваивание **$k:=1$** , а перед каждым новым повторением цикла изменять ее по закону **$k:=k+1$** .

Условие окончания цикла: **$S > x$** , а соответственно условие продолжения - **$S \leq x$** (взаимно противоположное условие):

```
Program pr1_cikl_poка;
```

```
  Var x,S,k : integer;
```

```
Begin
```

```
  Write('введите x '); readln(x);
```

```
  S:=0; {сумму всегда обнуляем перед циклом}
```

```
  k:=1; {первое слагаемое}
```

```
  while S<=x do
```

```
    begin
```

```
      S:=S+k;
```

```
      k:=k+1
```

```
    end;
```

```
    writeln('сумма = ',S, 'число слагаемых ',k-1);
```

```
end.
```

Пример 2. Составить программу, подсчитывающую сумму цифр некоторого натурального числа N .

Количество цифр в числе неизвестно, поэтому в теле цикла мы будем выполнять следующие действия:

- 1. выделять** из числа крайнюю правую цифру при помощи операции **$c := N \bmod 10$** ;
- 2. добавлять** её к сумме: **$S := S + c$** ;
- 3. «отбрасывать»** эту цифру при помощи оператора **$N := N \operatorname{div} 10$** .

Продолжать этот процесс будем до тех пор, пока число N не станет равным 0.

```
Program sum_cifr;  
  Var N:longint; c,s:byte;  
Begin  
  Write('Введите число ');  
  Readln(N);  
  S:=0;{начальное присваивание для суммы}  
  While N<>0 do  
    Begin  
      C:=N mod 10;  
      S:=s+c;  
      N:=N div 10  
    End;  
  Writeln('сумма цифр равна ',s)  
End.
```

Цикл «до» в языке Паскаль имеет следующий **формат:**

repeat

<тело цикла>

until <булевское выражение>;

здесь ключевое слово ***repeat*** переводится – повторять, ***until***-до:

Тело цикла повторяется до тех пор, пока булевское выражение ложно. Как только оно станет истинным – цикл закончится.

В отличии от цикла «пока» в этом цикле тело цикла выполняется хотя бы один раз всегда.

В цикле «пока» после ключевого слова *while* записывается условие продолжения цикла, а в цикле «до» после ключевого слова *until* записывается условие окончания цикла.

Пример 3. В качестве примера проиллюстрируем, как рассмотренный выше **пример 1** реализуется с помощью цикла «до», внося по ходу программы операторы-комментарии:

```
Program pr1_cikl_do;  
  Var x,S,k : integer;  
Begin  
  Write('введите x '); readln(x);  
  S:=0;{сумму всегда обнуляем перед циклом}  
  k:=0; {подготовка слагаемых}  
  repeat {начинаем цикл}  
    k:=k+1;{увеличиваем слагаемое на 1}  
    S:=S+k;{добавляем к сумме}  
  until S>x ;{условие окончания цикла}  
  writeln('сумма = ',S, ' число слагаемых ',k);  
end.
```

Пример 4. Дано действительное число x . Составить программу, вычисляющую произведение:

$$P = x * (x + 0.2) * (x + 0.4) * (x + 0.6) * \dots * (x + 1.8) * (x + 2)$$

Для накапливания (вычисления) произведений в теле цикла в общем случае используем оператор вида:

$P := P * \langle \text{очередной сомножитель} \rangle$

В данном примере: **$P := P * (x + a)$** , где a – переменная величина, которая для первого сомножителя равна 0, для второго – 0.2, для третьего – 0.4 и т.д.. То есть она изменяется по закону: **$a := a + 0.2$** , конечное значение равно 2.

Начальные присваивания: для произведения **$P := 1$** , для управляющей переменной **$a := 0$** . Условие окончания цикла: **$a > 2$** .

```
program proizv_cikl_do;  
  Var a,P,x: real;  
begin  
  Write('введите x='); readln(x);  
  P:=1; a:=0;  
  Repeat  
    P:=P*(x+a);  
    a:=a+0.2  
  until a>2;  
  writeln(P, a)  
end.
```