Циклические алгоритмы и программы

Циклом *называется многократно повторяющийся фрагмент алгоритма или программы.*

Те действия, которые повторяются, называются **телом цикла (ТЦ).**

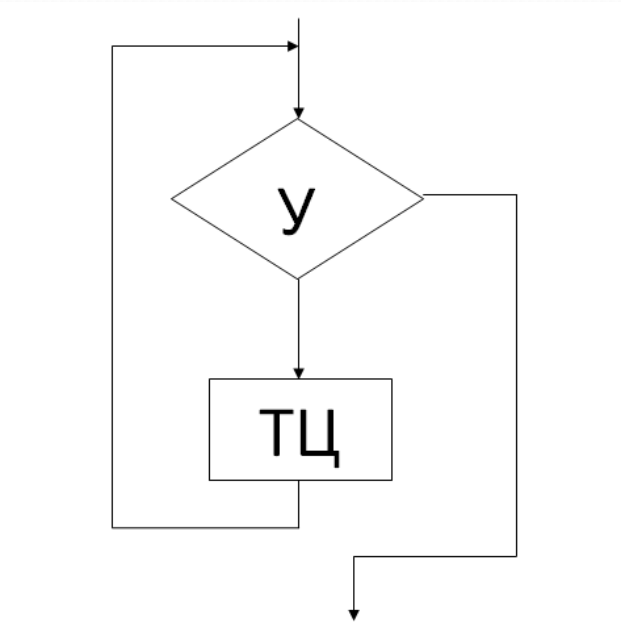
В программировании различают три типа циклов:

1) с предусловием («пока»);

2) с постусловием («до»);

3) с параметром («для»).

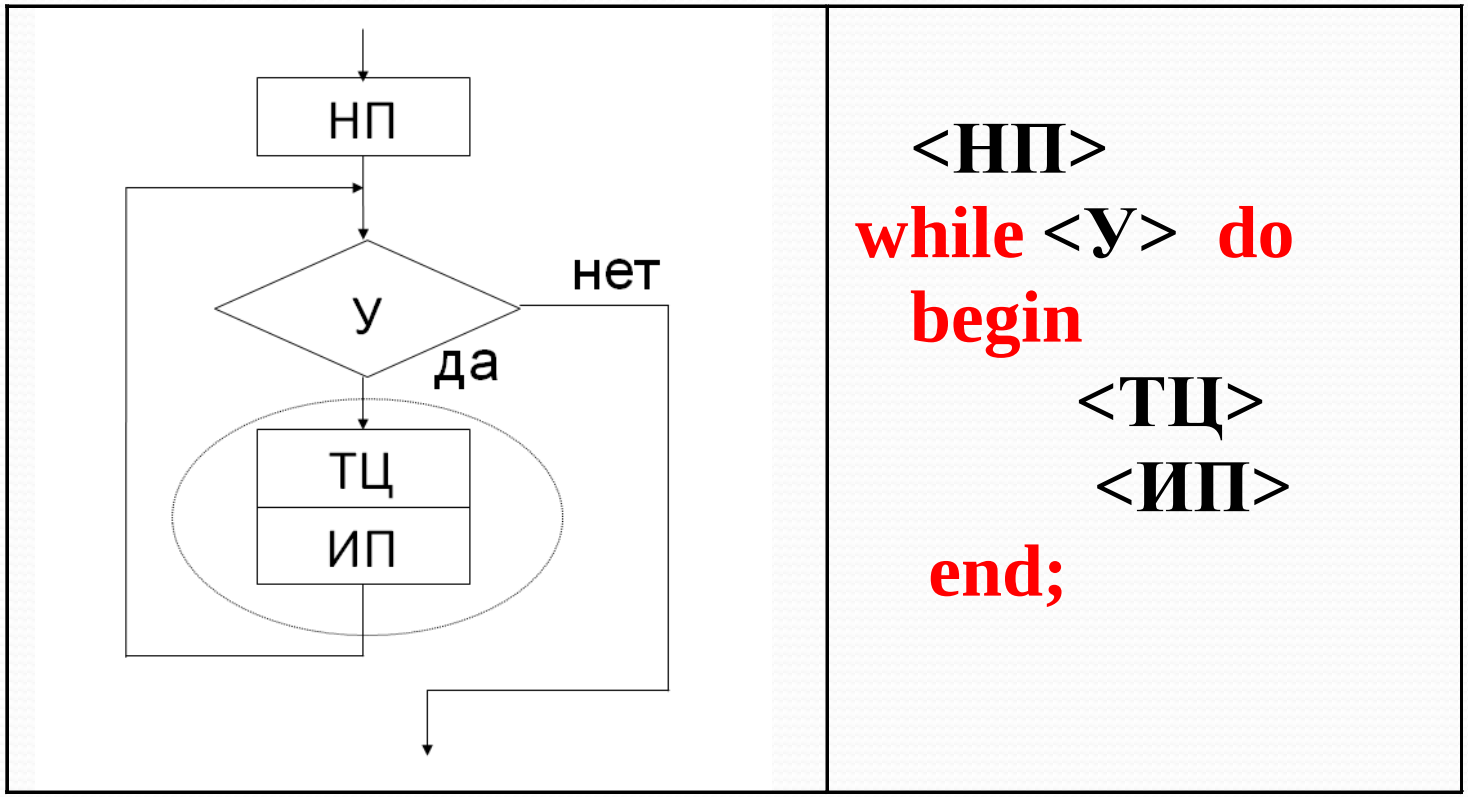
1) цикл с предусловием («пока»);

Блок-схема	Паскаль
	**while <У> do <ТЦ>;**

Здесь **У –** условие (булевское выражение),

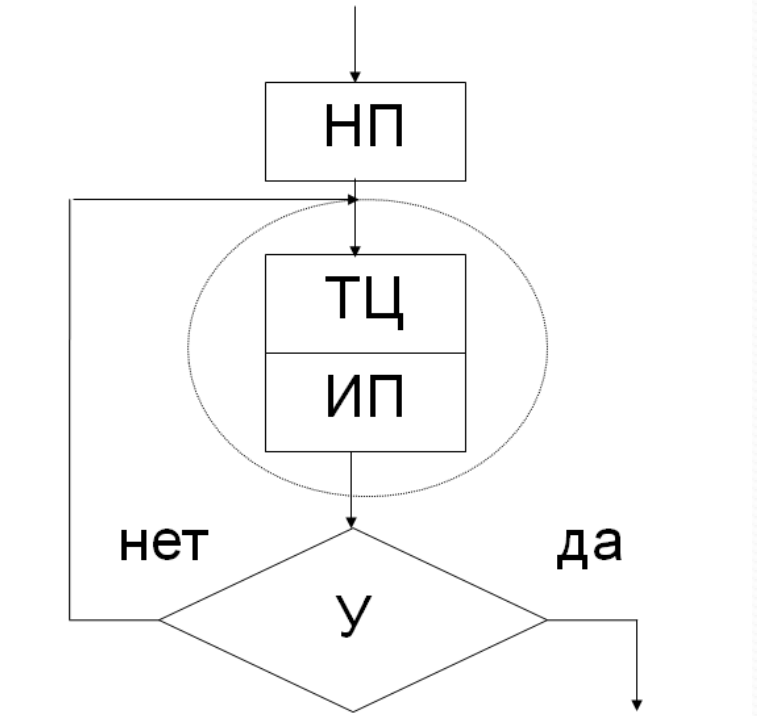
ТЦ –тело цикла.

1. блок начальных присваиваний (**НП**);
2. блок изменения управляющей переменной цикла (**ИП**);

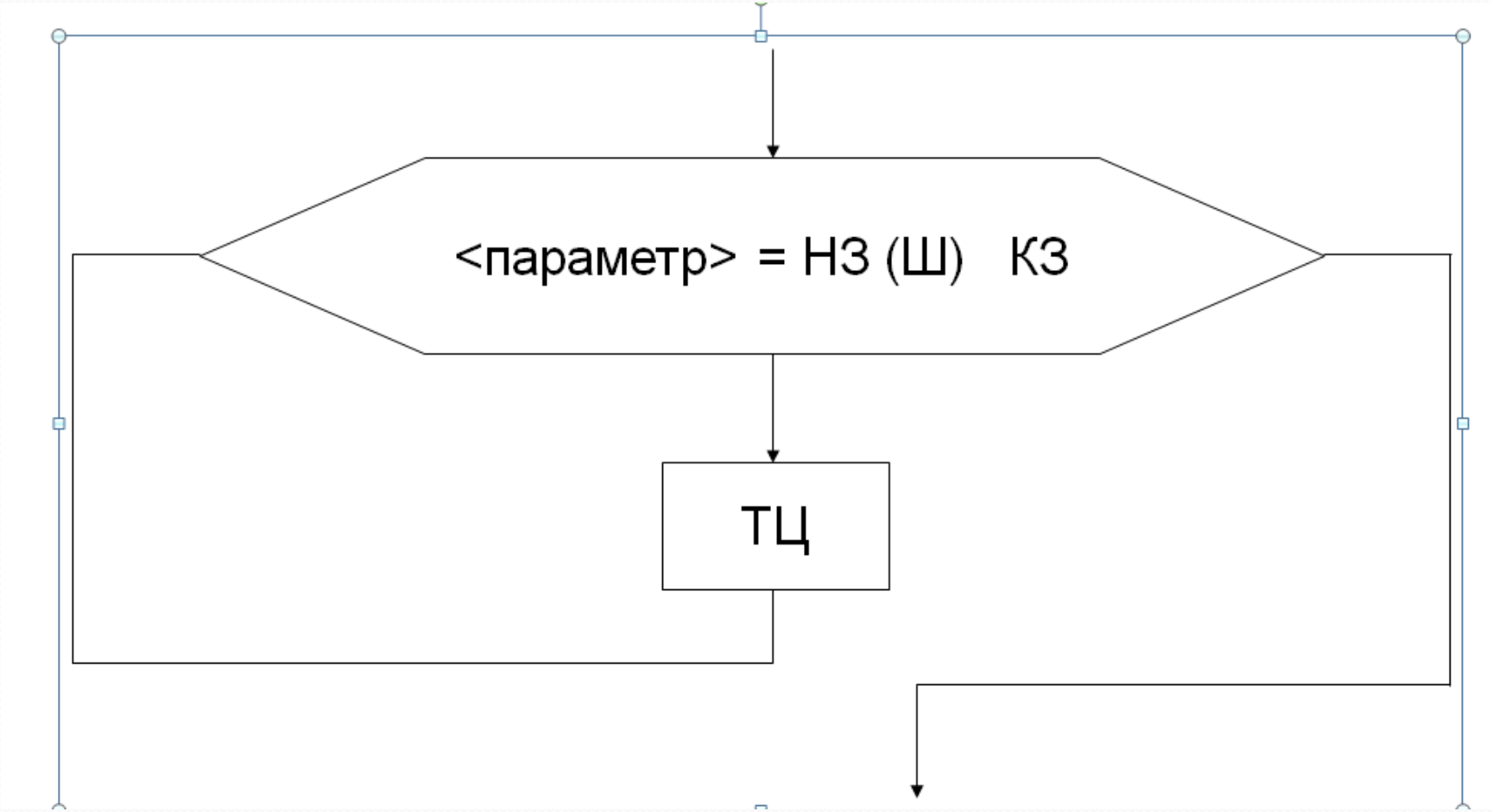



```
<НП>
while <У>  do
  begin
        <ТЦ>
         <ИП>
   end;
```

2) цикл с постусловием («до»);

Блок-схема	Паскаль
	**<НП>** **repeat** **<ТЦ>** **<ИП>** **until <У> ;**

3) цикл с параметром («для»)



Алгоритм выполнения цикла "для":

Обозначим р-параметр

1. **р:= НЗ**;
2. **р внутри отрезка [НЗ, КЗ] ?**

если **"да",** то к **п.3**, если **"нет"**, то **конец цикла;**

3. выполняется **ТЦ**
4. **р:= р+шаг**;
5. **К п.2;**

Замечания

1)Следует обратить внимание на связь между значениями параметра и количеством повторений тела цикла

n :=целая часть**(abs(КЗ - НЗ))/шаг+1.**

2) В блоке модификации (изменения) параметра цикла по сути дела объединены блоки **НП, ИП и У** циклов с потсусловием и предусловием (автоматически реализуется механизм изменения управляющей переменной и работы цикла). Поэтому существует правило: **в теле цикла параметр цикла изменять нельзя!**

Реализация циклических алгоритмов в языке Паскаль. Примеры.

Цикл с параметром («для»)

Цикл с параметром в Паскале имеет две формы:

1) ***for*** *<параметр>:= <НЗ>* ***to*** *<КЗ>* ***do*** *<ТЦ>;*
2) ***for*** *<параметр>:= <НЗ>* ***downto*** *<КЗ>* ***do*** *<ТЦ>;*

Здесь *<ТЦ>* (тело цикла) – простой или составной оператор. Если он составной, то нужно не забывать ставить операторные скобки ***begin-end;***

<параметр> - это переменная любого простого скалярного типа, кроме вещественного;

<НЗ> и *<КЗ>* - начальное и конечное значения диапазона изменения параметра, причем начальное значение может быть как меньше конечного, так и больше:

to **-** используется, если *НЗ< КЗ,*

downto **–** используется, если *НЗ> КЗ.*

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

<*параметр*>:= succ(<*параметр*>);

При использовании ***downto*** **–** по закону:

<*параметр*>:= pred(<*параметр*>).

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) <*НЗ*> - <*КЗ*>.

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; {c-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=trunc((b-a)/n) + 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+… . 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<=x*** (взаимно противоположное условие):

```
Program pr1_cikl_poka;
   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 mod 10;**
2. **добавлять** её к сумме: **S:=S+c;**
3. **«отбрасывать»** эту цифру при помощи оператора **N:=N 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)*…*(x+1.8)*(x+2)

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

P:=P*<очередной сомножитель>

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.
```