1.1 Операторы цикла(повторения)

В языке Паскаль существуют 3 оператора цикла.

1) ***Оператор цикла с параметром*** используется для организации цикла с известным числом повторений цикла (оператор для).

for <имя переменной>**:=** выражение1 ***to*** выражение2 ***do*** *оператор;*

или несколько операторов в цикле

for **c:=** A ***to*** B ***do*** *оператор(серия команд);*

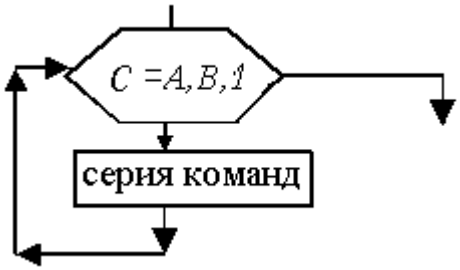


Рис1.

Отметим, что в цикле «for» переменная цикла С увеличивается всегда на 1 сама и автоматически от значения С=А до С=В.

Замечание.

1. Внутри цикла нельзя изменять значения A,B i пераменную цыкла C.

2. В операторе *for* не допускается ихменение переменной цикла на величину, которая не равна 1.

```
 for <имя переменной>:= выражение1 to выражение2 do
begin
    оператор1;
    оператор2;
 end;
```

Например, (здесь значение переменной увеличивается на единицу 4 раза).

```
s:=0;
for i:=1 to 4 do  s:=s+1; {счет от 1 до 4)
```

Переменная цикла должна быть только целого типа. Это оператор в модификации «Вниз к» имеет вид:

for i:=*a1* ***downto*** *a2* ***do*** оператор;

```
s:=0; for i:=4 downto 1 do s:=s+1; {счет от 1 до 4)
```

В этом случае при каждом новом выполнении оператора значение переменной цикла i уменьшается на 1.

Досрочный выход из цикла может быть осуществлен с помощью команды break(только из цикла) exit (выход из программы)**.**

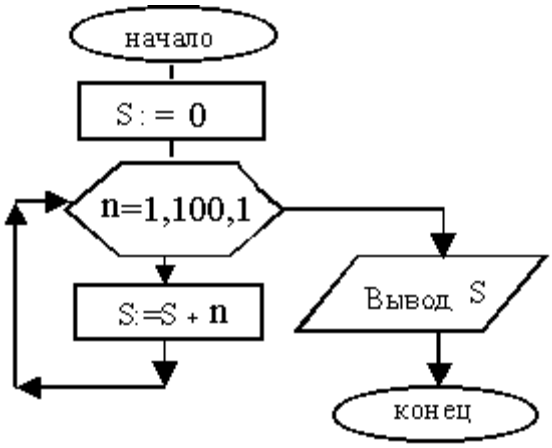
Пример. Дано натуральное число *n*. Найти наименьший простой сомножитель.

```
program somn;
var i,n:integer;
BEGIN
  write ('введите число ');     readln (n);
  for i:=2 to n do     begin
        if n mod i = 0          then begin
                          write('наименьший сомножитель',i);
                        exit;       end;
    end;
END.
```

Пример 2. Вычислить сумму натуральных чисел от 1 до 100.

$S = 0$;
1-й шаг: $S = 1$;
2-й шаг: $S = 1 + 2 = 3$ — **S:=S+2**;
3-й шаг: S:=1+2+3=6 — **S:=S+3**;
4-й шаг: $S = 1 + 2 + 3 + 4 = 10$ — **S:=S+4**;
……..
100-й шаг: **S:=S+100**.

Итак, получим общую **формулу суммы** S:=S+n (начало с 0), где $n \in [1; 100]$ и это целое число.




```
program sum;
  const p=100;
  var n:integer; s:real;
BEGIN
   S:=0;
   for  n:=1 to p   do S:=S+n;
   writeln('s=', s:7:2);
END.
```

Пример 2. Вычислить 10!. **См. блок-схему Рис.1.**

```
program sum;
  const p=10;
  var n:integer; F:real;
BEGIN
   F:=1;
   for  n:=1 to p   do F:=F*n;
   writeln('F=', F:7:2);
END.
```

формула произведения ф:=ф*n (начало с 1)

Прыклад 2. Составим программу вычисления значения выражения:

$$\sum_{H=1}^{P} \frac{\sin(H!) + e^{\frac{1}{H}}}{\cos((H+2)!) + (2H)!}.$$

```
program sum1;
  const  p=10;
  var n:integer; s,f,f1,f2,t:real;
  BEGIN
  s:=0; f:=1; f2:=1
  for n:=1 to p do
```

```
  begin
    f:=f*n;       f1:=f*(n+1)*(n+2);     f2:=f2*(2*n-1)*(2*n);
    t:=(sin(f)+exp(1/n))/(cos(f1)+f2);
    s:=s+t;
  end;
  wrinteln('s= ',s:8:3);
END.
```

Общая формула суммы S:=S+t (начало с 0), **где t –слагаемое, формула которого определяется программистом**

Общая формула произведения F:=F*p (начало с 1), **где p –множитель, формула которого определяется программистом**

Циклические алгоритмы могут иметь сложную структуру. Если внутри цикла создан другой цикл, то такой цикл называют кратным (вложенным). Максимальное число вложенных циклов зависит от конкретного языка программирования. В случае кратных циклов недопустимо использование одной переменной более чем в одном цикле для его формирования**.**

Пример. $C = \sum_{a=1}^{5} \sum_{p=1}^{3} a * p$.

a	*p*	*c*
		0
1	1	1
1	2	1+2=3
1	3	3+3=6
2	1	6+2=8
2	2	8+4=12
2	3	12+6=18
3	1	18+3=21
3	2	21+6=27
3	3	27+9=36
4	1	36+4=40
4	2	40+8=48
4	3	48+12=60
5	1	60+5=65
5	2	65+10=75
5	3	75+15=90

```
s:=0;
for a:= 1 to 5 do
  for p:=1 to  3 do
            c:=c+a*p;
```

Рассмотрим пример работы операторов цикла и опрераторов модуля *crt*.

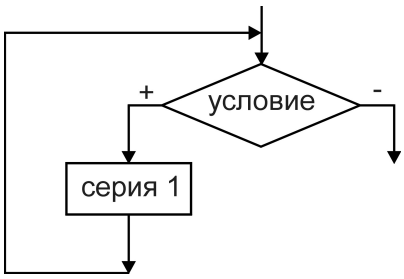
Пример. *Вывод по диагонали 15 слов информатика*

```
 for i:=1 to 15 do
     begin        gotoxy(i,i);          writeln('информатика');
     end;
```

```
program aa;
uses crt;
var i:byte;
BEGIN
  clrscr;    write('informatita');
  repeat
     for i:=2 to 22 do
```

```
        begin
            insline;            delay(90);
        end;
      gotoxy(1,1);
      for i:=2 to 22 do
        begin
            delline;            delay(90);
        end;
      delay(100);
  until keypressed;
END.
```

2) ***Оператор цикла с предусловием (цикл пока).*** Этот оператор имеет вид:
while **условие**
do оператор;.



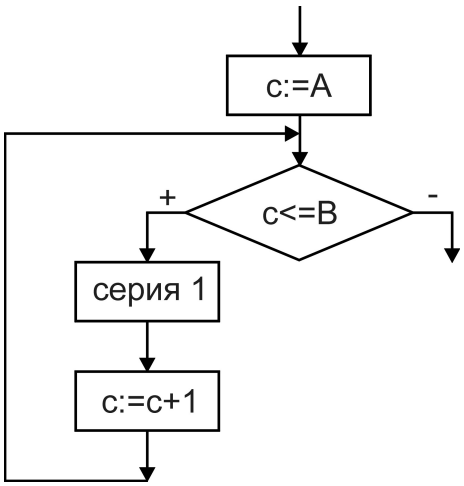
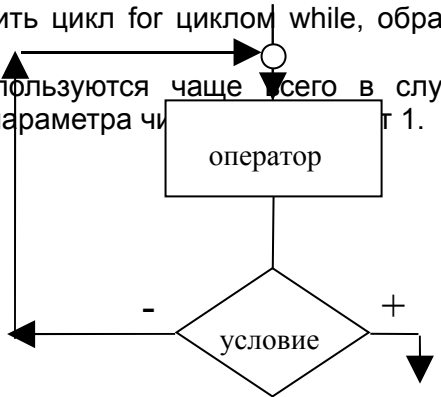


Рис 2. *Рис.3.*

while и *do* – служебные слова, условие – логическое (булево) выражение. Выполняется следующим образом: сначала вычисляется значение булева выражения. Если это значение истинно (*true*), то выполняется оператор, следующий за служебным словом *do* (операторы могут быть заключены в операторные скобки *begin...end*), и снова происходит возврат к вычислению значения булева выражения. Так повторяется, пока значение булева выражения не станет ложным (*false*). Выполнение оператора, следующего за словом *do*, прекращается, поэтому такой оператор называется оператором с предусловием. Это означает также, что оператор, следующий за служебным словом *do*, может быть и не выполнен ни разу, если при первом же вычислении значение булева выражения будет ложным.

На рис 3. Показано, как можно заменить цикл for циклом while, обратное можно сделать не всегда.

Операторы цикла while и repeat используются чаще всего в случаях, когда неизвестно число повторений или изпенение параметра ци т 1.



3***) Оператор цикла с постусловием***.
repeat

оператор
***until* условие**;

Этот оператор выполняется следующим образом: сначала выполняется оператор, следующий за служебным словом *repeat*, затем вычисляется значение булева выражения, образующего условие. Если значение булевого выражения ложно, то происходит возврат к выполнению оператора, следующего за служебным словом *repeat*, и снова вычисляется булево выражение. Так повторяется, пока значение булева выражения остается ложным. Как только оно станет истинным, выполнение оператора цикла прекращается. В отличие от оператора *while...do* оператор, следующий за служебным словом *repeat* будет обязательно выполнен хотя бы один раз независимо от значения булева выражения.

```
repeat
  read (b);      if b<>0 then m:=m+b;
until b=0;.
```

Если в операторе *while...do* используется составной оператор, то он записывается с использованием служебных слов *begin* и *end*.

Вычисления значения функцуии на отрезке от –10 до 0 с шагом 2.

```
x:=-10
while x<=0 do     { после do точка с запятой не ставится}
  begin
     y:=sqr(x);
      writeln('x= ',x:7:2,'     y=',y:7:2);
     x:=x+2;
  end;
```

Если такой же составной оператор используется в операторе *repeat*, то в его записи служебные слова *begin* и *end* могут быть опущены.

```
x:=-10;
repeat
  y:=sqr(x);
  writeln('x= ',x:7:2,'     y=',y:7:2);
  x:=x+2;
  until x>0;.
```

Чтобы операторы цикла while и repeat выполнялись конечное число раз и не произошло зацикливания при построении цикла, необходимо предусмотреть, чтобы среди выполняемых операторов обязательно был оператор, выполнение которого влияло бы на значение булева выражения. В этих примерах это оператор *x:=x+2*.

Для досрочного выхода из цикла, определяемого одним из операторов *while*, *repeat* могут быть использованы операторы *break* или *exit*.

Пример. Сумма цифр двузначного числа равна 6. Если к этому числу прибавить 18, то получиться число, записанно теми же цифрами, но в обратном порядке. Найдем это двузначное число:

```
program sls;
        var a,b,n:integer;
BEGIN
        writeln('вариант 1');
        for a:=1 to 6 do
                for b:=0 to 5 do
                        if (a+b=6) and (a*10+b+18=b*10+a) then n:=a*10+b;
        writeln(n);
```

```
        writeln('вариант 2');
        a:=1;
        while a<=6 do
begin
                b:=0;
   while b<=5 do begin
                    if (a+b=6) and (a*10+b+18=b*10+a) then n:=a*10+b;
                    b:=b+1
                  end;
        a:=a+1
        end;
        writeln(n);
        writeln('вариант 3');
        a:=1;
        repeat
                b:=0;
                repeat
                    if (a+b=6) and (a*10+b+18=b*10+a) then n:=a*10+b;
                    b:=b+1
                until b>5;
                a:=a+1
        until a>6;
        writeln(n)
END.
```

Пример. Составим программу проверки знаний таблицы умножения:

```
program tablica;
 uses crt;
        const k=10;      var a,b,c,d,i:integer;
BEGIN
        randomize; clrscr;
        for i:=1 to k do
                begin
                    a:=random(8)+2;   b:=random(8)+2;
                  c:=a*b;
                    writeln('Чему равно   ',  a, '*',  b,  ' ? ');
                    write('Введите ответ    '); readln(d);
                    if c=d then writeln ('Правильно')
                            else writeln('Ошибка');
                    writeln;
                end;END.
```