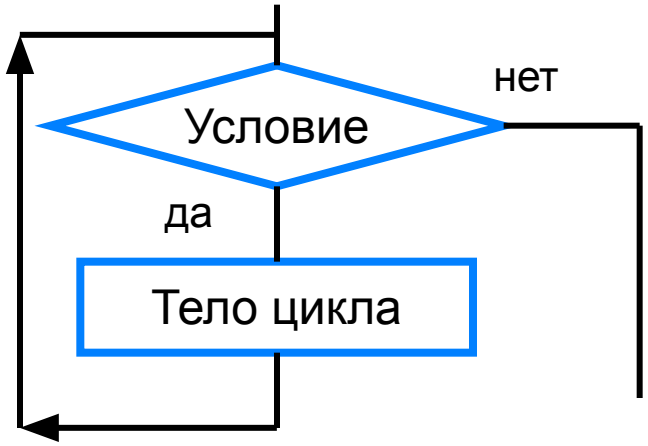
# ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ

# Программирование циклов с заданным условием продолжения работы



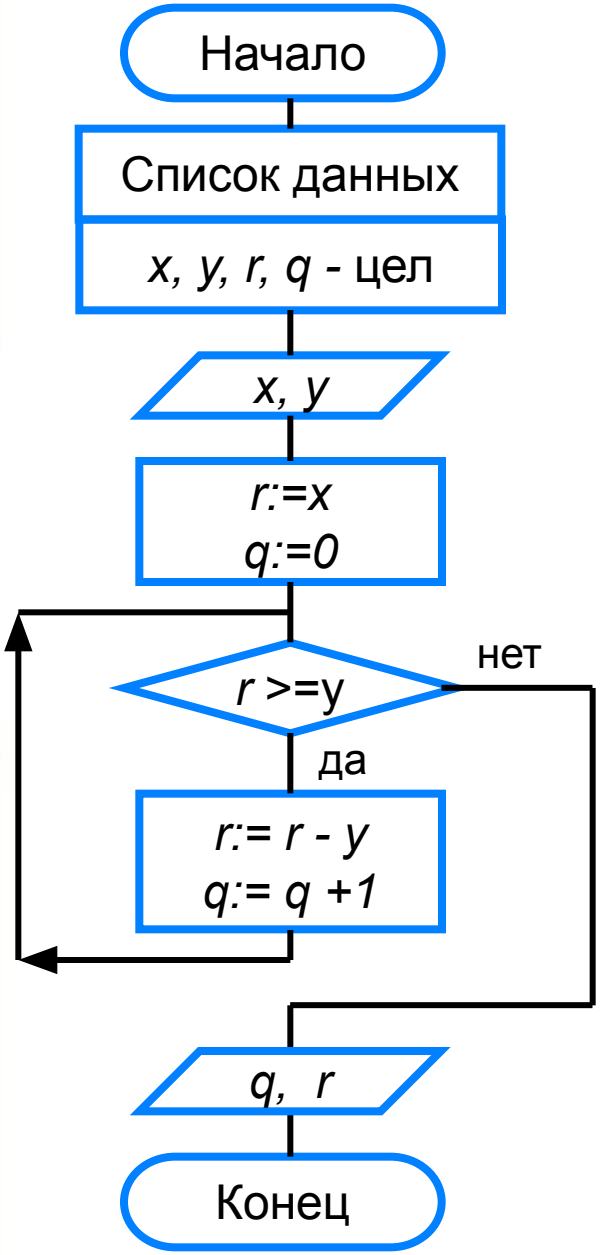

Общий вид оператора:

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

Здесь:

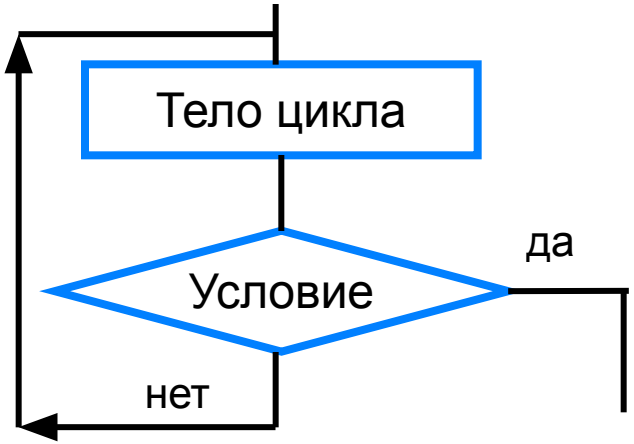
<**условие**> - логическое выражение; пока оно истинно, выполняется тело цикла;

<**оператор**> - простой или составной оператор, с помощью которого записано тело цикла.




```
program n_14;
 var  x, y, q, r: integer;
begin
 writeln ('Частное и остаток');
 write ('Введите делимое x>>');
 readln (x);
 write ('Введите делитель y>>');
 read (y);
 r:=x;
 q:=0;
 while r>=y do
begin
  r:=r-y;
  q:=q+1
end;
 writeln ('Частное q=', q);
 writeln ('Остаток r=', r)
end.
```

# Программирование циклов с заданным условием окончания работы




Общий вид оператора:

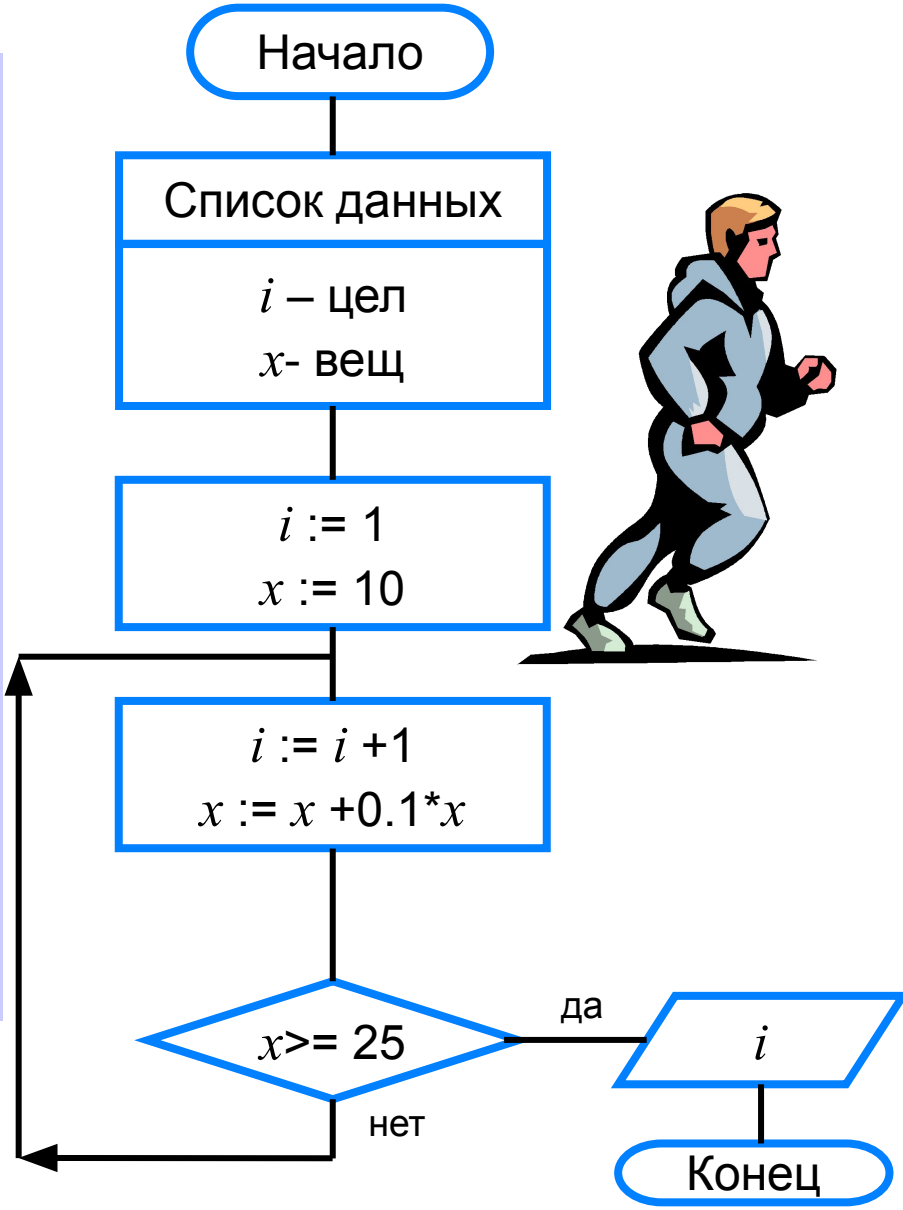
**repeat** <**оператор1; оператор2; …; >** **until** <**условие**>

Здесь:

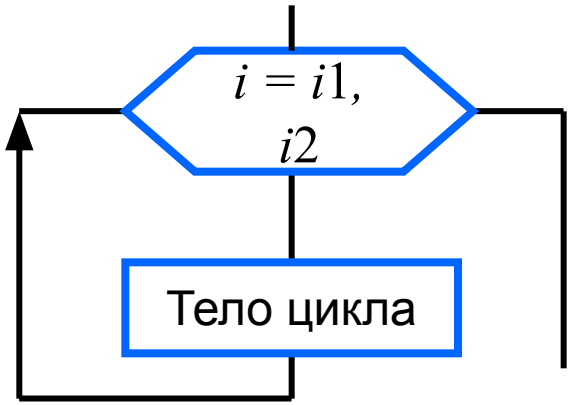
<**оператор1**>; <**оператор2**>; … - операторы, образующие тело цикла;

<**условие**> - логическое выражение; если оно ложно, то выполняется тело цикла.

```
program n_15;
  var i: integer; x: real;
begin
  writeln ('График тренировок');
  i:=1;
  x:=10;
  repeat
  i:=i+1;
  x:=x+0.1*x;
  until x>=25;
  writeln ('Начиная с ', i, '-го  дня спортсмен будет пробегать 25 км')
end.
```



# Программирование циклов с заданным числом повторений




Общий вид оператора:

**for** <параметр>:=<начальное_значение>

**to** <конечное_значение> **do** <оператор>

Здесь:

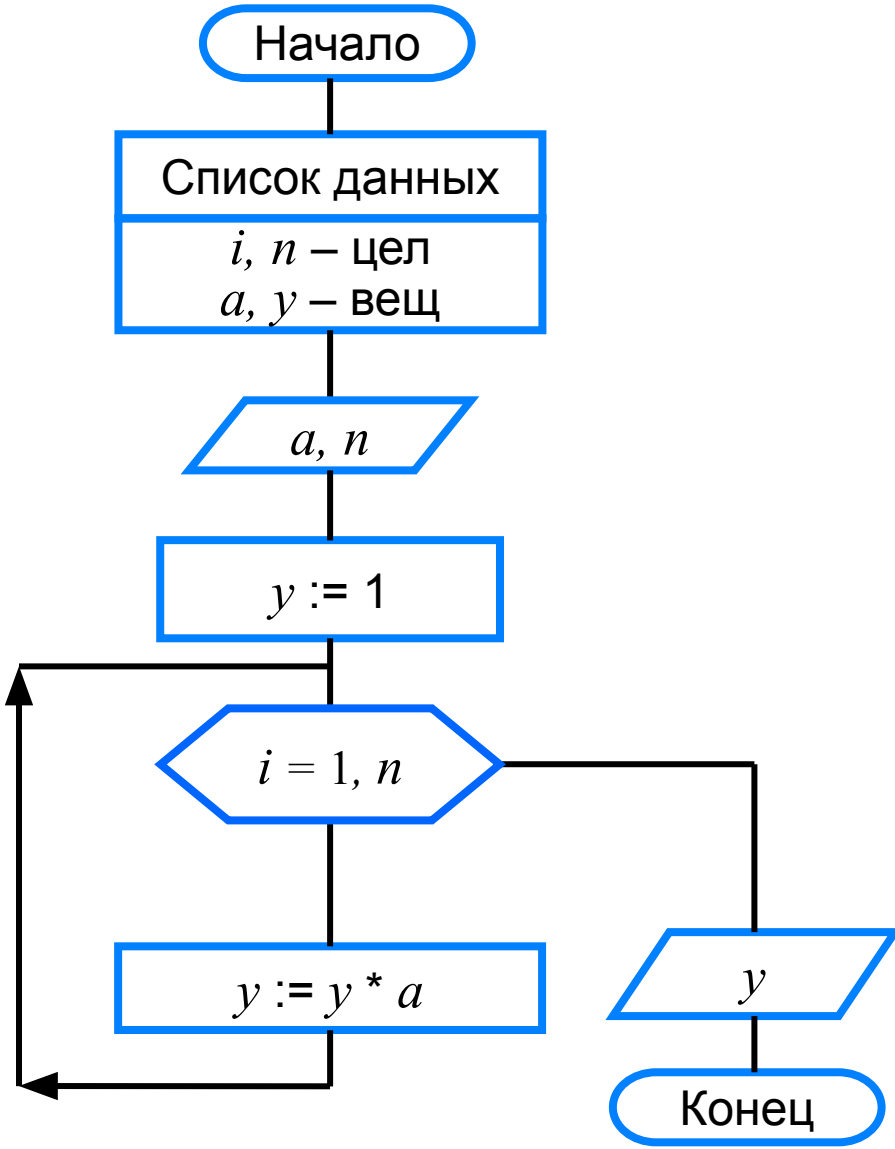
<параметр> - переменная целого типа;

<начальное значение> и <конечное значение> - выражения того же типа, что и параметр;

<оператор> - простой или составной оператор - тело цикла.

После каждого выполнения тела цикла происходит увеличение на единицу параметра цикла; условие выхода из цикла - превышение параметром конечного значения.

```
program n_16;
  var i,n:integer;a,y:real;
begin
  writeln ('Возведение в степень');
  write ('Введите основание a>>');
  readln (a);
  write ('Введите показатель n>>');
  readln (n);
  y:=1;
  for i:=1 to n do y:=y*a;
  writeln ('y=', y)
end.
```



# Различные варианты программирования циклического алгоритма

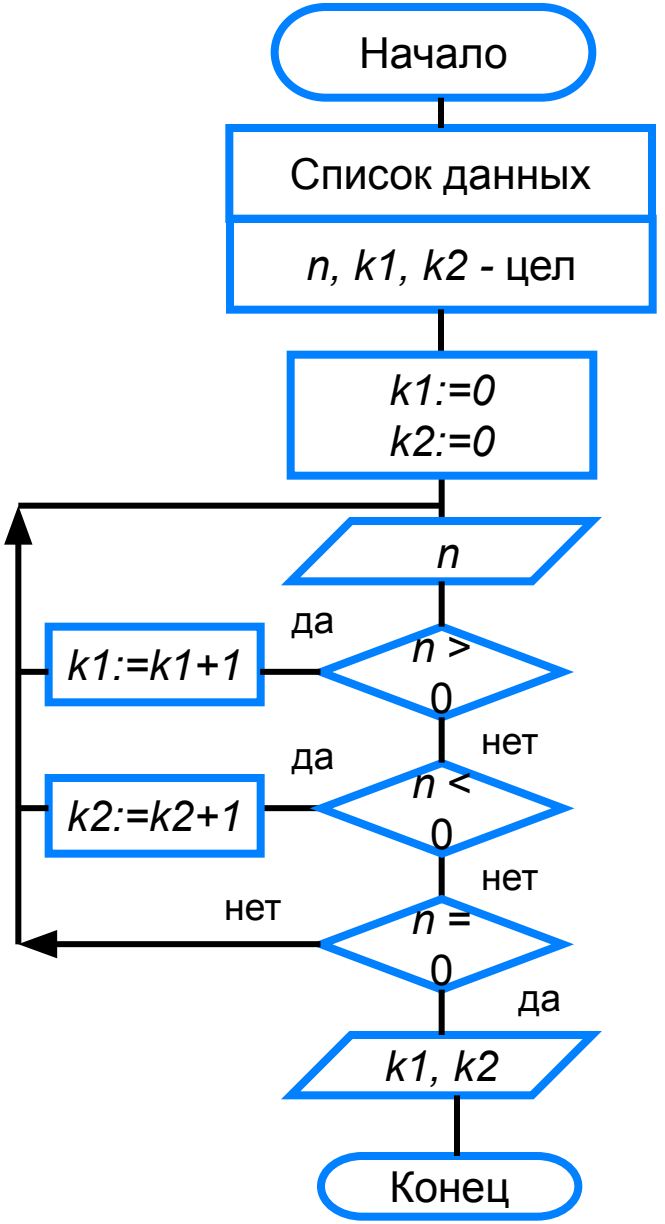
Для решения одной и той же задачи могут быть созданы разные программы.

Организуем ввод целых чисел и подсчёт количества введённых положительных и отрицательных чисел. Ввод должен осуществляться до тех пор, пока не будет введён ноль.

В задаче в явном виде задано условие окончания работы.

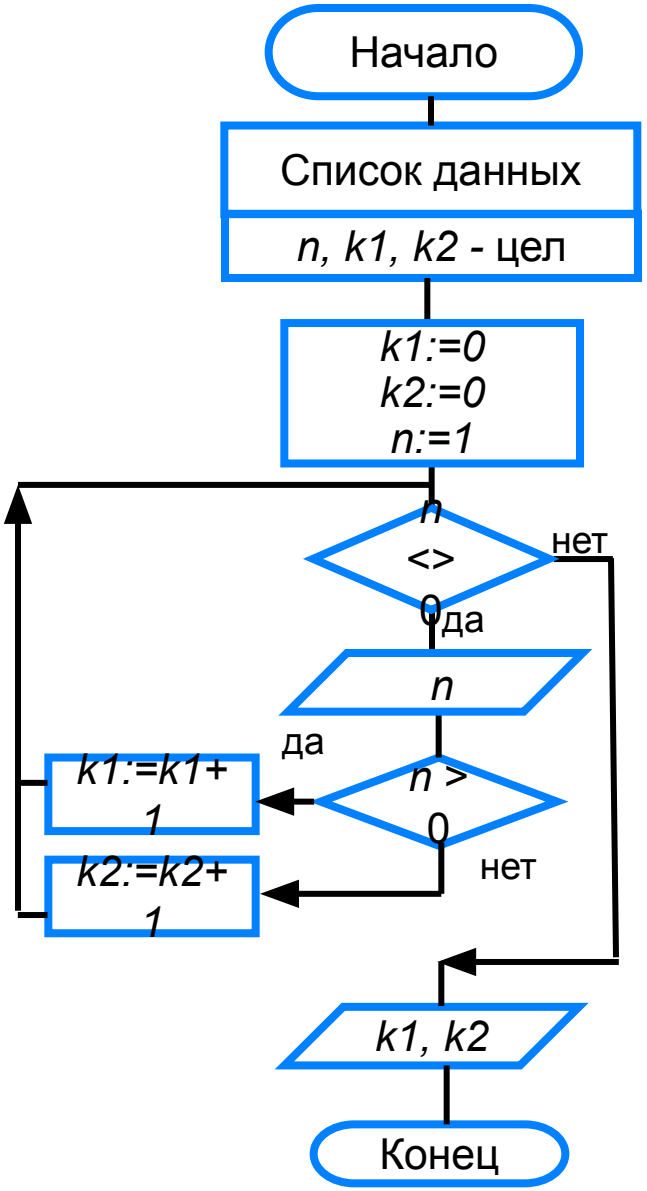
Воспользуемся оператором **repeat**.

```
program n_17;
  var n, k1, k2: integer;
begin
  k1:=0;
  k2:=0;
  repeat
    write ('Введите целое число>>');
    readln (n);
    if n>0 then k1:=k1+1;
    if n<0 then k2:=k2+1;
  until n=0;
  writeln ('Введено:');
  writeln ('положительных чисел – ', k1);
  writeln ('отрицательных чисел – ', k2)
end.
```



Ввод осуществляется до тех пор, пока не будет введён ноль.

Работа продолжается, пока $n \neq 0$.

Воспользуемся оператором **while**:

```
program n_18;
  var n, k1, k2: integer;
begin
  k1:=0;
  k2:=0;
  n:=1;
  while n<>0 do
  begin
    writeln ('Введите целое число>>');
    read (n);
    if n>0 then k1:=k1+1;
    if n<0 then k2:=k2+1;
  end;
  writeln ('Введено:');
  writeln ('положительных – ', k1);
  writeln ('отрицательных – ', k2)
end.
```