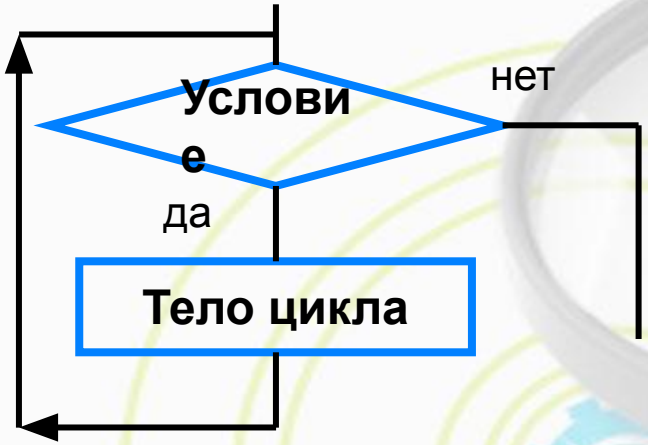
Тема урока:

Циклы с условием

Оператор цикла с предусловием



Формат оператора:

while <условие> do <оператор>

или:

while <условие> do
begin
операторы циклической части программы
end;

Здесь **while** *(пока)* и **do** *(выполнить)* --- служебные слова.

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

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

Оператор цикла с предусловием

```
while <условие> do
begin
операторы циклической части программы
end;
```

Выполнение оператора:

1). Вычисляется логическое выражение и проверяется условие.

2). Пока значение логического выражения **истинно** выполняются операторы циклической части.

3). Как только оно становится **ложным**, происходит **выход из цикла**. Если с самого начала значение логического выражения ложно, то операторы циклической части не выполняются ни разу.

*Возможен случай, когда в циклической части стоит оператор перехода (**goto**), передающий управление за пределы цикла. В такой ситуации цикл может завершится до его естественного окончания.*

Задача1 Мой богатый дядюшка подарил мне один доллар в мой первый день рождения. В каждый следующий день рождения он удваивал свой подарок и прибавлял к нему столько долларов, сколько лет мне исполнилось. Рассмотрим программу, указывающую, к какому дню рождения подарок превысит 100$.

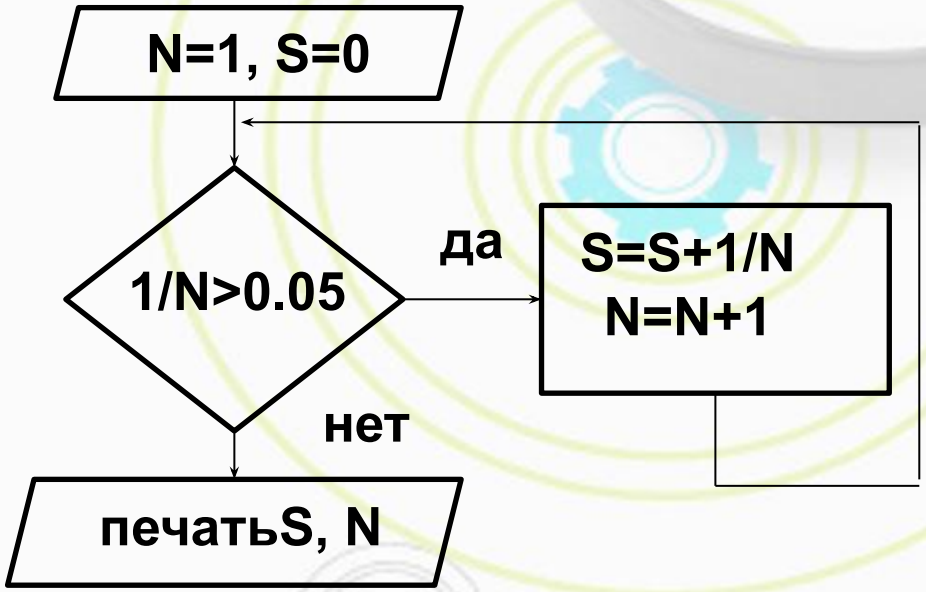
```
program PODAROK;
  var i, S: integer;
begin
     i:=1;  S:=1;
     while S<=100 do
        begin
        i:=i+1;
        S:=S*2+i
        end;
     writeln ('сумма=', S)
     writeln ('количество лет =', i)
end.
```

Задача2 Перед вами программа вычисления суммы элементов ряда:

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \ldots\ldots\ldots$$

Вычисления заканчиваются, когда очередное слагаемое становится меньше 0.05

Блок-схема:




```
Program SUMMA;
var S: real;  N: integer;
begin
S:=0;  N:=1;
    while 1/N>0.05 do
    begin
     S:=S+1/N;
     N:=N+1
     end;
 writeln ('Сумма = ', S:10:9)
end.
```

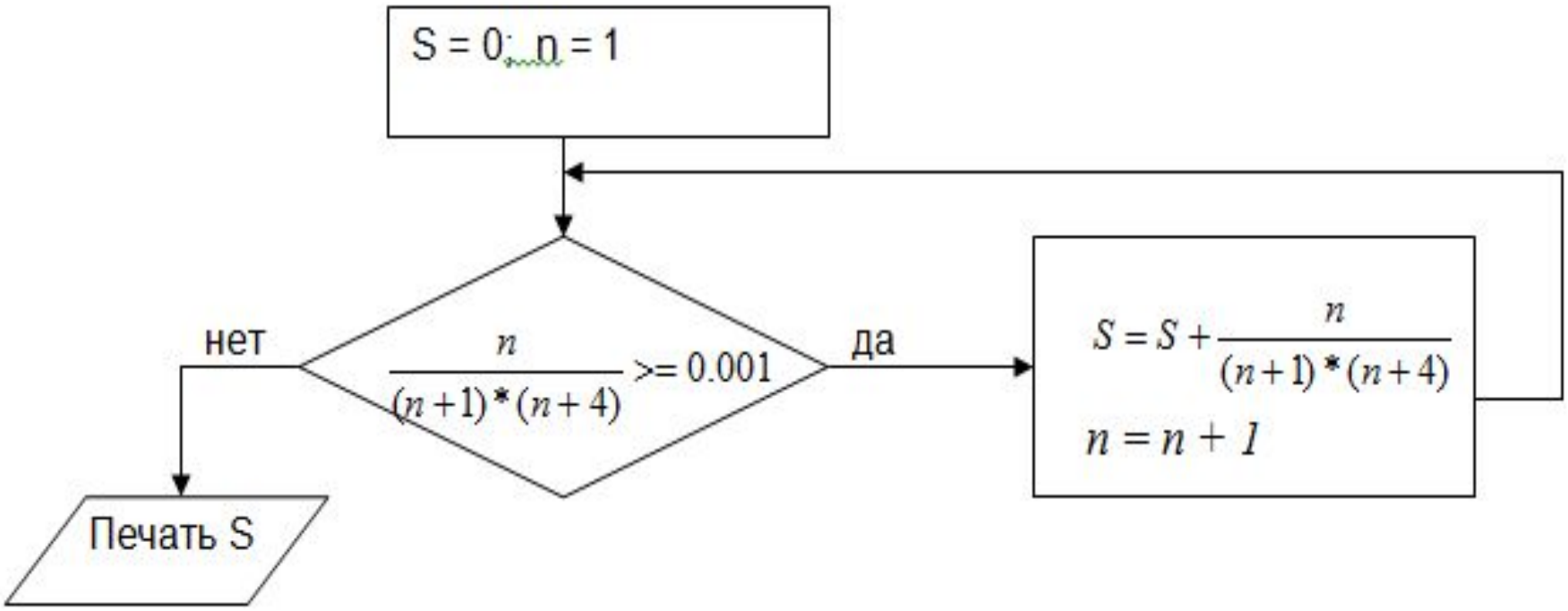
Задача3 Проанализируем алгоритм вычисления суммы ряда:

$$S = \frac{1}{2*5} + \frac{2}{3*6} + \ldots\ldots + \frac{n}{(n+1)*(n+4)}$$

представленный в виде блок-схемы.

Вычисления будут закончены, когда очередное слагаемое станет меньше числа **0,001**.

Блок-схема:



Задача3 Перед вами программа вычисления суммы ряда: $S = \frac{1}{2*5} + \frac{2}{3*6} + \ldots\ldots + \frac{n}{(n+1)*(n+4)}$

Вычисления заканчиваются тогда, когда очередное слагаемое становится меньше числа **0,001**.

```
Program SUMMA;
var s: real; n: integer;
begin
s:=0;  n:=1;
   while n /((n+1)*(n+4)) >=0.001 do
      begin
       S := S + n / ((n+1)*(n+4));
       n := n + 1
       end;
 writeln ('Сумма = ',s);
end.
```

Задача4 Рассмотрим алгоритм вычисления факториала: **10! = 1·2·3·4·5·6·7·8·9·10** с помощью программы:

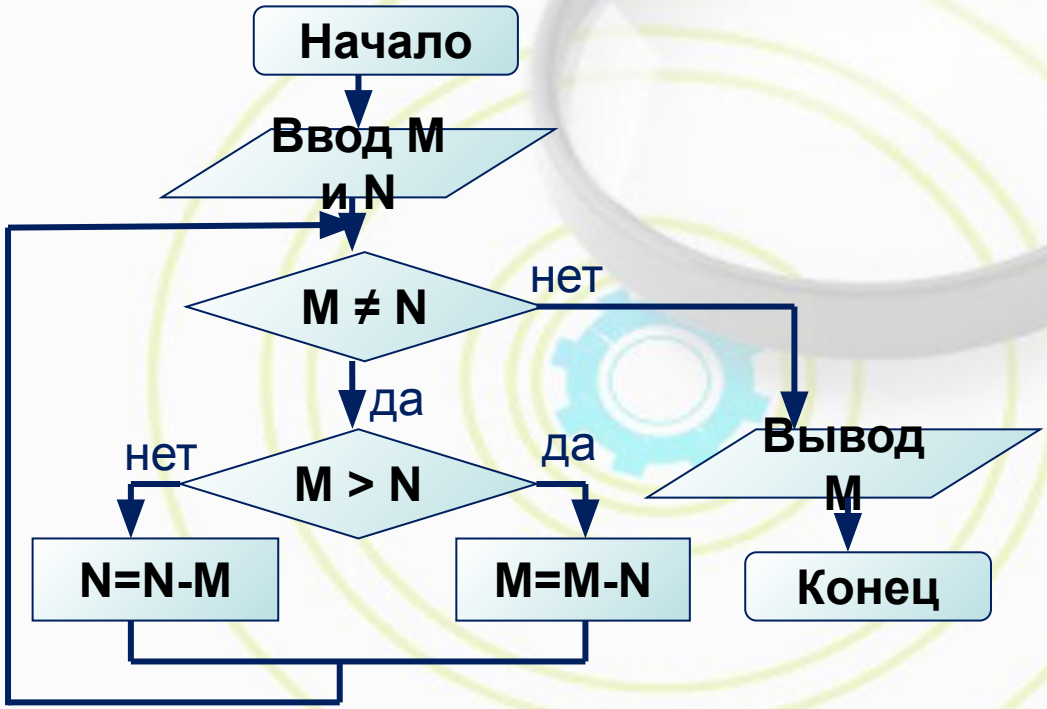
```
program fact;
VAR Factorial, N: Integer;
BEGIN
Factorial := 1;          {стартовое значение факториала =0! }
N := 1;                  {стартовое значение для условия цикла }
WHILE N<=10 DO           {заголовок цикла, условие }
   begin                 {начало тела цикла }
  Factorial := Factorial*N;  {вычисление факториала N! }
  N := N + 1             {N должно меняться в цикле}
  end;                   {конец тела цикла }
WriteLn('10!= ',Factorial);  {вывод результата}
END.
```

Задача5 Наша задача усложняется. Нужно вычислить сумму: **S=1+1/2!+1/3!+...+1/N!** (при целом N>0).

```
Program SUMMA;
var s, f: real;  i, n: integer;
begin
  write ('Введи N= ');  readln(n);
 s:=1;  f:=1;   i:= 2;
 while i<= n do
    begin
     f:=f*i;
     s:=s+1/f
     end;
 writeln ('Сумма = ',s:10:9);
end.
```

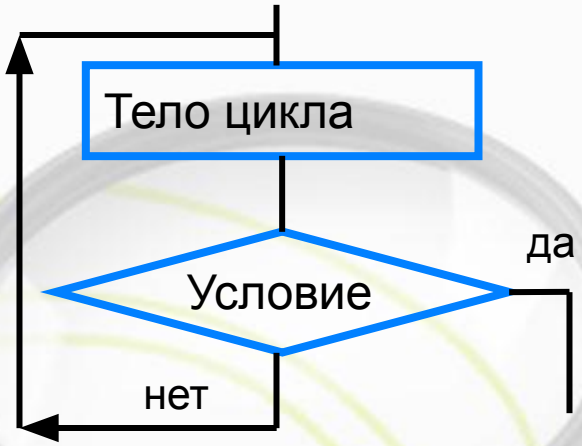
В этой задаче в одном цикле объединено нахождение суммы и вычисление факториала очередного слагаемого.

Задача 6 Рассмотрим программу нахождения наибольшего общего делителя (**NOD**) двух целых чисел: **M** и **N** по алгоритму Евклида.




```
program Evklid;
var M, N, NOD: integer;
begin
writeln ('Введите два числа');
 readln (M, N);
       while M<>N do
        if M>N
        then M:=M-N
        else N:=N-M;
NOD:=M;
writeln 'НОД = ',NOD);
end.
```

Оператор цикла с последующим условием



Формат оператора:

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

Здесь:

repeat (***повторить***) и **until** (***до тех пор пока***) -- служебные слова.

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

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

Оператор цикла с предварительным условием

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

Выполнение оператора:

1). Операторы циклической части (**оператор1; оператор2; …;**) выполняются один раз, затем проверяется логическое условие.

2). Если логическое условие **ложно**, то снова повторно выполняются операторы циклической части.

3). Условием прекращения циклических вычислений является **истинное значение** логического выражения после **UNTIL**.

Следует подчеркнуть, что нижняя граница операторов циклической части четко обозначена словом **UNTIL**, поэтому нет надобности заключать операторы циклической части в скобки вида **BEGIN --- END**.

В то же время и дополнительное наличие скобок не является ошибкой.

Задача7 Рассмотрим программу вычисления и печати элементов последовательности, каждый из которых вычисляется по формуле: $X_n = \frac{2n-1}{2n}$

где n – порядковый номер элемента. Вычисления продолжаются пока $|x_{n+1} - x_n| > 0.001$.

Программа:

```
Program progr;
var A, B: real; n: integer;
begin
n:=1;   A := (2*n – 1)/(2*n);
n:=n+1; B = (2*n – 1)/(2*n)
repeat
    writeln (‘элемент=', A, ‘ его номер=‘, n-1);
    A := (2*n – 1)/(2*n);
    n:=n+1;
    B = (2*n – 1)/(2*n);
until  ABS (A - B) <  0.001;
readln;
end.
```

Различные варианты программирования циклического алгоритма

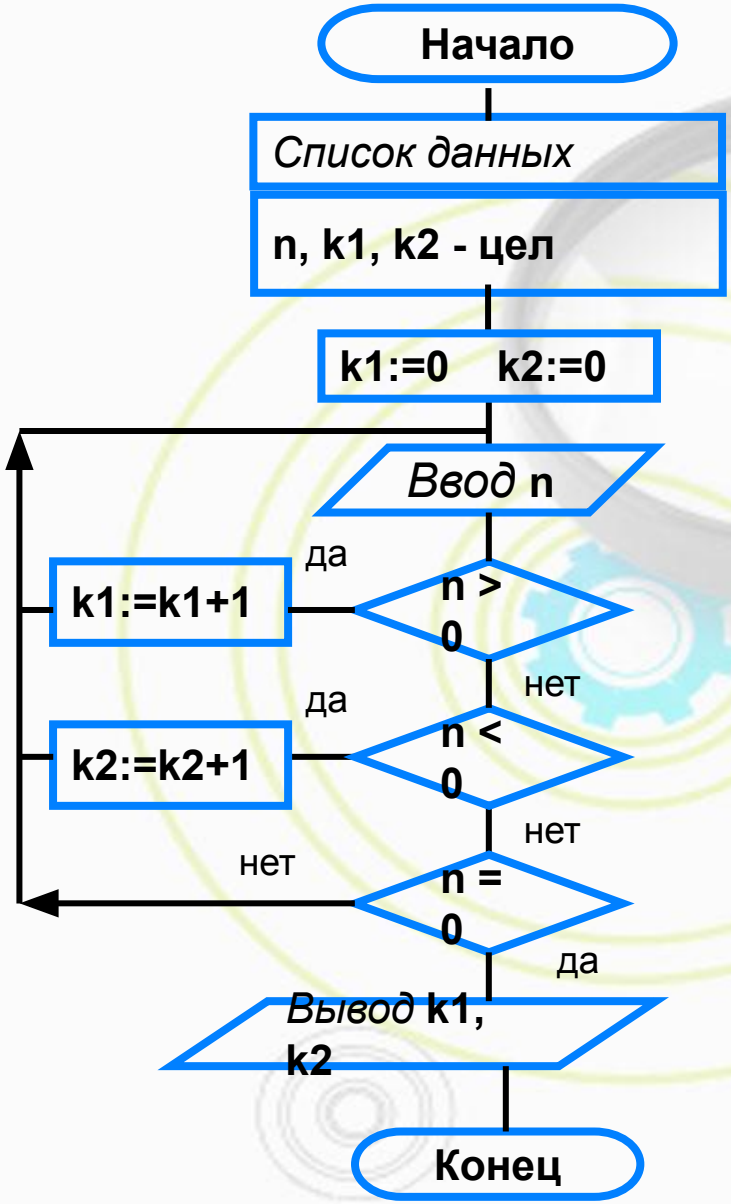
Для решения одной и той же задачи могут быть созданы разные программы, например, для следующей задачи:

Составить программу, которая организует 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** (***повторить***)

Подсчет количества положительных и отрицательных чисел.



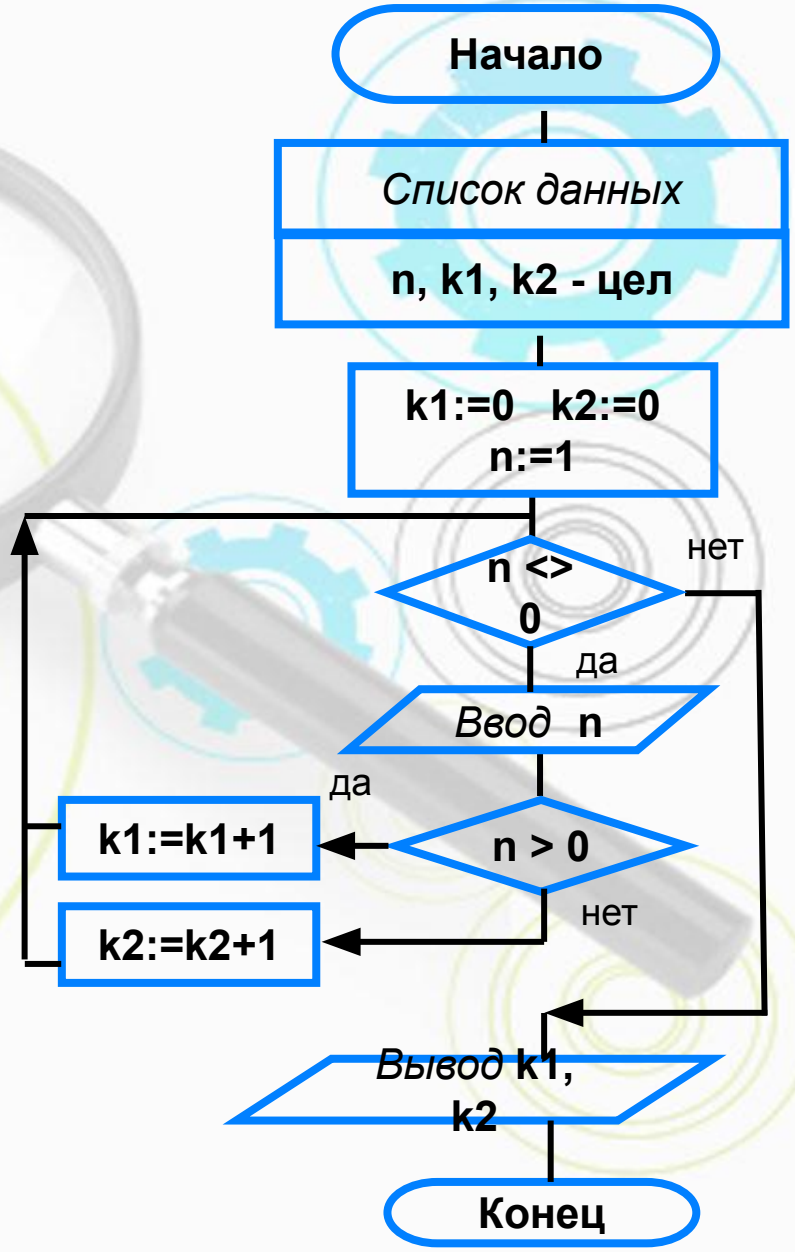

```
program n_17;
  var n, k1, k2: integer;
begin
  k1:=0;   k2:=0;
        repeat
          write ('Введите целое число>>');
          readln (n);
          if n>0 then k1:=k1+1;
          if n<0 then k2:=k2+1;
        until n=0;
writeln ('положительных чисел – ', k1);
  writeln ('отрицательных чисел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



Задание на дом:

1. **Выучить конспект урока.**

2. **Задача.** Составьте программу , которая напечатает элементы, каждый из которых вычисляется по формуле:

$$a = \frac{n}{(n+1)*(n+4)}$$

где n - порядковый номер слагаемого.
Вывод закончить, когда очередной элемент станет меньше числа 0,001.