программирование циклов

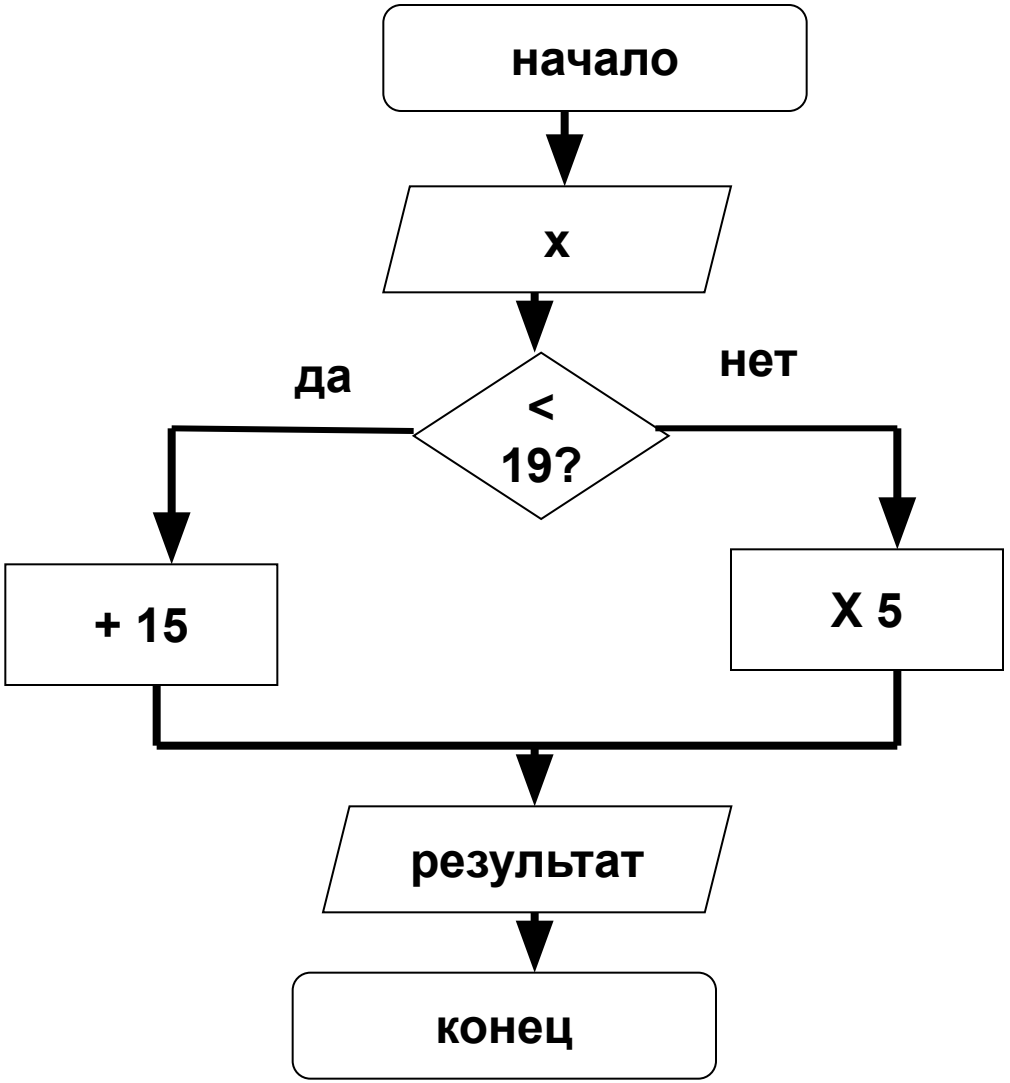
Урок в 9 классе

Найди ошибки:

```
Program new ;
  Uses crt;
Var a, b, c : integer ;
Begin
clrscr ;
Readln(a,b);
C:=a*a+b*b ;
Writeln(c);
End.
```

Разминка

- Выполните счёт по блок-схеме для чисел: 11, 18, 21

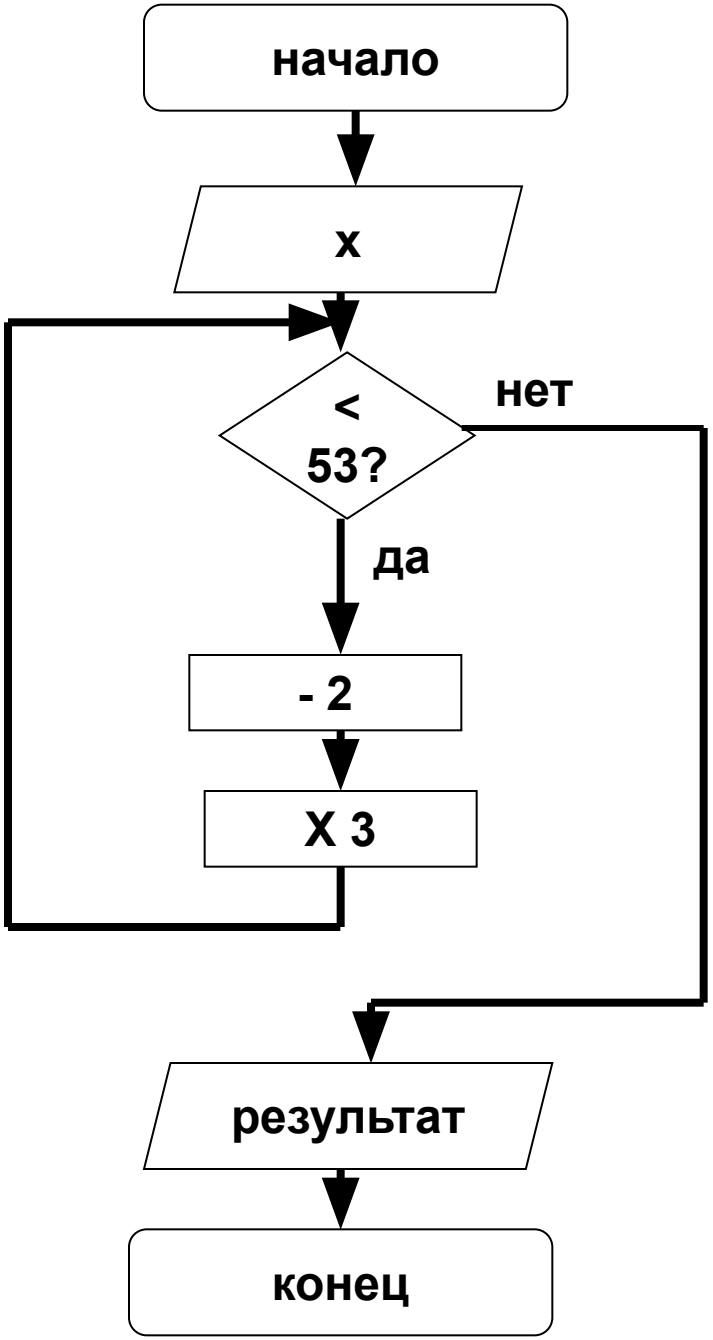


X = 11 18 21

P = **26** **33** **105**

- Выполните счёт по блок-схеме для чисел : 5, 9, 12

X =	5	9	12
P =	**57**	**57**	**84**



Циклические алгоритмы

- *Цикл* представляет собой последовательность операторов, которая выполняется *неоднократно.*
- В языке программирования Turbo Pascal имеется три разновидности цикла:

1. Цикл с постусловием (repeat)
2. Цикл с предусловием (while)
3. Цикл со счётчиком (For)

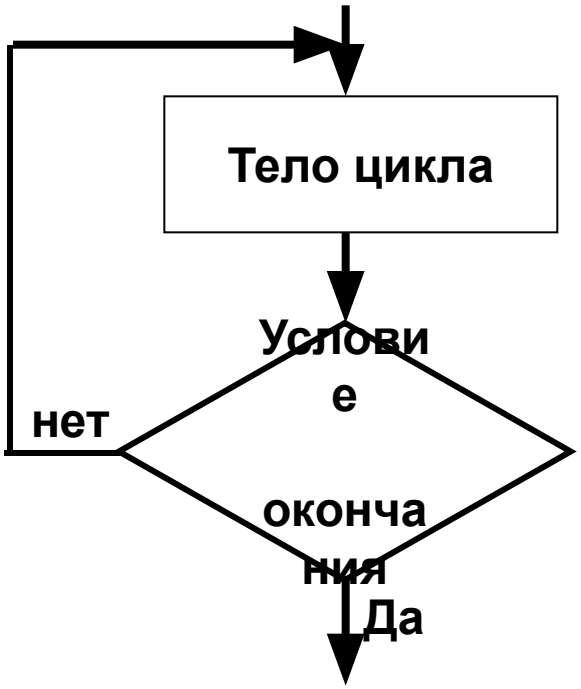
Запомните!

- Подавляющее большинство задач с циклами можно решить разными способами, используя при этом любой из трёх операторов цикла;
- В некоторых случаях предпочтительнее использовать какой-то один из операторов;
- Самым универсальным из всех операторов цикла считается *while*, поэтому в случае затруднений с выбором можно отдать предпочтение ему;
- Цикл *for* обеспечивает удобную запись циклов с *заранее известным числом повторений;*
- При неумелом использовании циклов любого типа возможна ситуация, когда компьютер не сможет нормально закончить цикл (в таком случае говорят, что программа «зациклилась»). При работе в среде Turbo Pascal для выхода из подобной ситуации используется комбинация клавиш <Ctrl>+<Break>.

Оператор REPEAT

- Оператор повтора repeat состоит из заголовка (repeat), тела и условия окончания (until). Ключевые слова repeat, until обозначают «повторяй» и «пока» соответственно.

Repeat
{инструкции}
Until Условие выхода из цикла



- **Число повторений** операторов (инструкций) цикла **repeat** заранее неизвестно и определяется в ходе программы;
- После слова **until** записывается условие ***завершения*** цикла
- ***Условие*** – это выражение логического типа: простое или сложное логическое выражение
- цикл **repeat** удобно использовать в тех случаях, когда тело цикла гарантированно *должно выполниться хотя бы один раз;*

Примеры:

Определить число n, при котором сумма квадратов натурального ряда чисел от 1 до n не превысит величину K, введенную с клавиатуры. Т. е.

S >= K, где S= $\sum_{i=1}^{n} i^2 = 1^2 + 2^2 + \ldots + n^2$

```
repeat
WriteLn('Введите
   положительное
   число');
ReadLn(x);
until x>0;
```

```
program sum_sq;  {Сумма квадратов
    натурального ряда }
    uses crt;
    var   k, s, n : integer;
    begin
    clrscr ;
    writeln( 'Введите K' );
    readln (k);
    s:=0;    n:=1;
    repeat
      s :=s+n*n;
      n := n+1;
    until  s > k;
    writeln  ('N= ', n : 3,  '   s= ' ,  s : 5 );
    readln;
    end.
```

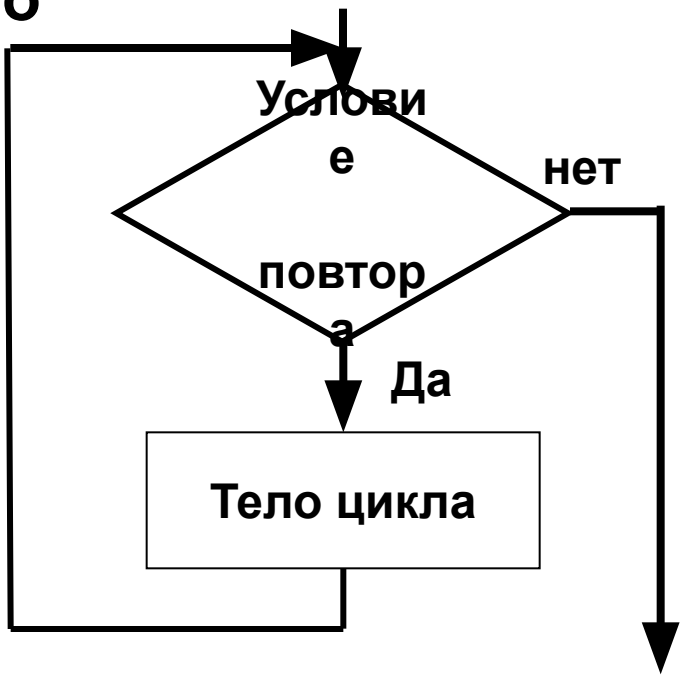
Оператор WHILE

- Оператор повтора while состоит из *заголовка* и *тела цикла.* Ключевые слова *while* и *do* обозначают «до тех пор, пока» и «выполняй» соответственно

While Условие выполнения цикла **do**

{инструкции}

End;



- ***Число повторений*** операторов (инструкций) цикла **while** определяется в ходе работы программы;
- После слова **while** записывается условие ***продолжения выполнения*** инструкций цикла. В этом отличие цикла **while** от цикла **repeat**;
- ***условие*** – это выражение логического типа, которое может принимать одно из двух значений: true или false;
- Цикл **while –** это цикл с предусловием, т.е. инструкции тела цикла *вообще могут быть не выполнены,* если проверяемое условие ложно с самого начала;

Примеры:

Пример: Надо вычислить сумму S=1+1/2+1/3+...+1/50

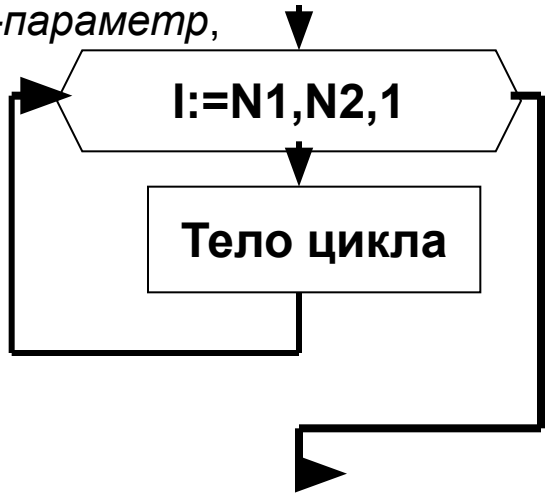
```
Program summa;
VAR S: REAL;
 N:INTEGER;
BEGIN
 S:=0; N:=1;
WHILE N<=50 DO
 BEGIN S:=S+1/N; N:=N+1;
   END;
WRITELN(' S=',S);
 END.
```

Пример: необходимо подсчитать целое кол-во отрезков длиной 1,5 м получающихся из бруска длиной 20 м, цикл подсчета будет выглядеть следующим образом:

```
Program brusok;
Var   L: real; i: integer;
Begin
i:=0;{кол-во отрезков}
L:=0;{суммарная длина
    отрезков}
while L<20 do
begin
L:=L+1.5;
i:=i+1;
 end;
write("количество отрезков ",i);
End.
```

Оператор FOR

- Этот вид оператора цикла называют ***циклом со счётчиком*** или ***цикл с параметром.*** В нём важную роль играет *переменная-параметр*, которая на каждом шаге цикла автоматически изменяет своё значение ровно *на единицу* – поэтому её и называют *счётчиком.*



- Инструкцию **for** можно реализовать двумя способами:

Вариант 1

For счётчик:=НачальноеЗначение **to** КонечноеЗначение **do**
Begin
{инструкции}
End;

Ключевые слова for, do обозначают «для», «выполняй» соответственно. Строка, содержащая for . . . do, называется *заголовком цикла,* оператор, стоящий после do образует его *тело.* Очень часто тело цикла – составной оператор. Если тело цикла представлено одиночным оператором, то begin и end не пишутся.

Вариант 2

For Cчётчик:=НачальноеЗначение **downto** КонечноеЗначение **do**
Begin
{инструкции}
End;

Инструкции между **begin** и **end** выполняются столько раз, сколько определяет выражение [(НачальноеЗначение – КонечноеЗначение)+1]

Примеры:

Пример: Надо вычислить сумму S=1+1/2+1/3+...+1/50

```
Program summa;
VAR S: REAL;
 I: INTEGER;
BEGIN
S:=0;
FOR I:=1 TO 50 DO
   S:=S+1/I;
WRITELN(' S=',S);
   END.
```

Пример: Необходимо возвести 2 в 5-ую степень

```
Program stepen;
VAR f, I: INTEGER;
BEGIN
f:=1;
for i:=1 to 5 do
begin
f:=f*2;
end;
WRITELN (‘2^5 =  ‘,F);
End.
```

Примеры

- Например, выполнение цикла-фрагмента программы:

For i:=14 **downto** 10 **do** write(i:3);

Выведет на экран последовательность чисел в виде:

14 13 12 11 10

Если переменная-счётчик имеет символьный char тип, то оператор

For ch:='a' **to** 'e' **do** write (ch:2);

Выведет на экран последовательность букв в виде:

a b c d e

Оператор

For ch:='e' **to** 'a' **downto** write (ch:2);

Выведет на экран последовательность букв в виде:

e d c b a

Правила

- Оператор for используется для организации циклов с *фиксированным,* заранее известным числом повторений;
- Количество повторений определяется *начальным* и *конечным* значением *переменной-счётчика*.
- Переменная-счётчик должна быть порядкового типа: чаще integer, реже – char, boolean и т.д.
- Начальное и конечное значение должны быть константами и должны принадлежать к одному и тому же типу.
- Параметр цикла for может изменяться(увеличиваться или уменьшаться) каждый раз при выполнении тела цикла *только на единицу.* Если нужен другой шаг изменения параметра, предпочтительнее циклы repeat или while

1. Постановка задачи

- Дано N кубиков, на которых написаны разные буквы. Сколько различных N-буквенных слов можно составить из этих кубиков (слова не обязательно должны иметь смысл)?
- Искомую величину обозначим буквой F.

Тогда постановка задачи выглядит так:

Дано: N

Найти: F

Математическая формализация

- Получим расчётную формулу на конкретном примере. Пусть имеются два кубика с буквами «И» и «К». Тогда можно составить только два слова ИК и КИ.
- Добавим третью букву А, тогда получим слова:

ИКА, КИА, ИАК, АКИ, КАИ, АИК (6 слов) и т.д.

Количество различных комбинаций из N букв равно:

N! (N – факториал)

Примеры:

1!=1

2!=1*2

3! = 1*2*3

4! = 1*2*3*4

5! = 1*2*3*4*5 и т.д.

N! = 1*2*3*4 … *N

Алгоритм «Слова»

```
Алг СЛОВА
Цел F, N, R
Нач ввод N
     F:=1
     R:=1
   пока  R<=N , повторять
   нц
     F:=F*R
     R:=R+1
   кц
   вывод F
кон
```

Цикл с предусловием

Программа

```
Program words;
Var F, N, R : integer ;
Begin
Write (‘введите число букв’);
Readln(N) ;
F:=1;
R:=1;
While R<=N do
Begin
F:=F*R ;
R:=R+1
End ;
Write(‘Из’ , N , ‘букв можно составить  ‘ , F ,  ‘ слов’)
End.
```