

# Программирование на языке Бейсик

**Тема . Циклы**

# Циклы

---

**Цикл** – это многократное выполнение одинаковой последовательности действий.

- цикл с **известным** числом шагов
- цикл с **неизвестным** числом шагов (цикл с условием)

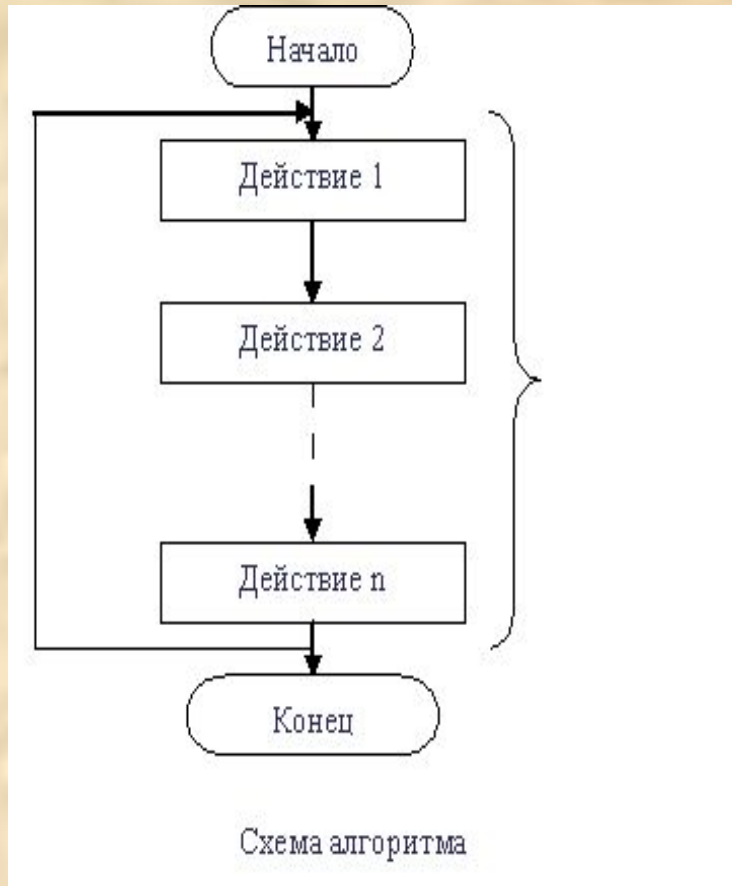
**Задача.** Вывести на экран квадраты и кубы целых чисел от 1 до 8 (от **a** до **b**).

**Особенность:** одинаковые действия выполняются 8 раз.



Можно ли решить известными методами?

# Схема простейшего цикла

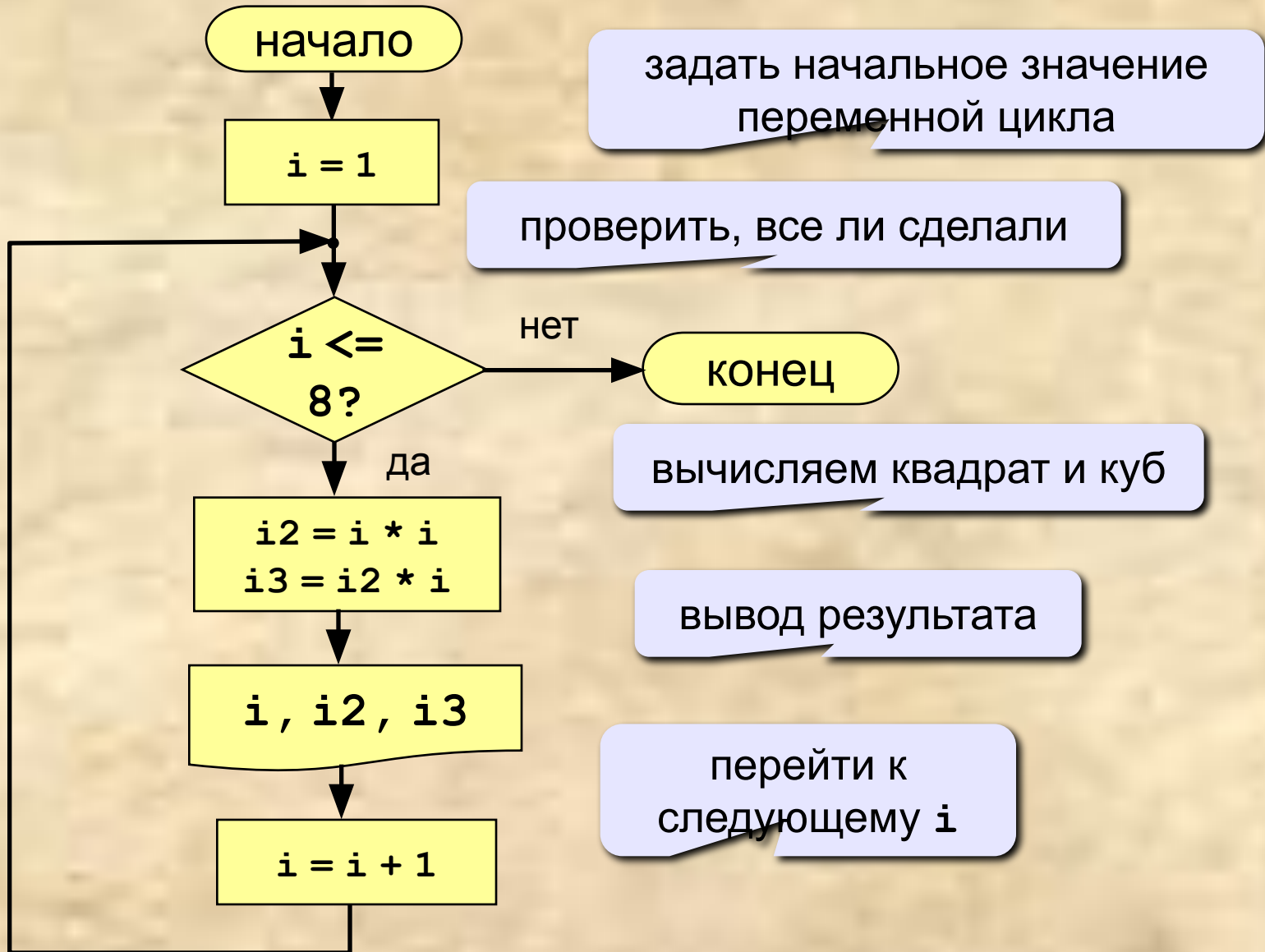


Любой цикл характеризуется тремя стадиями:

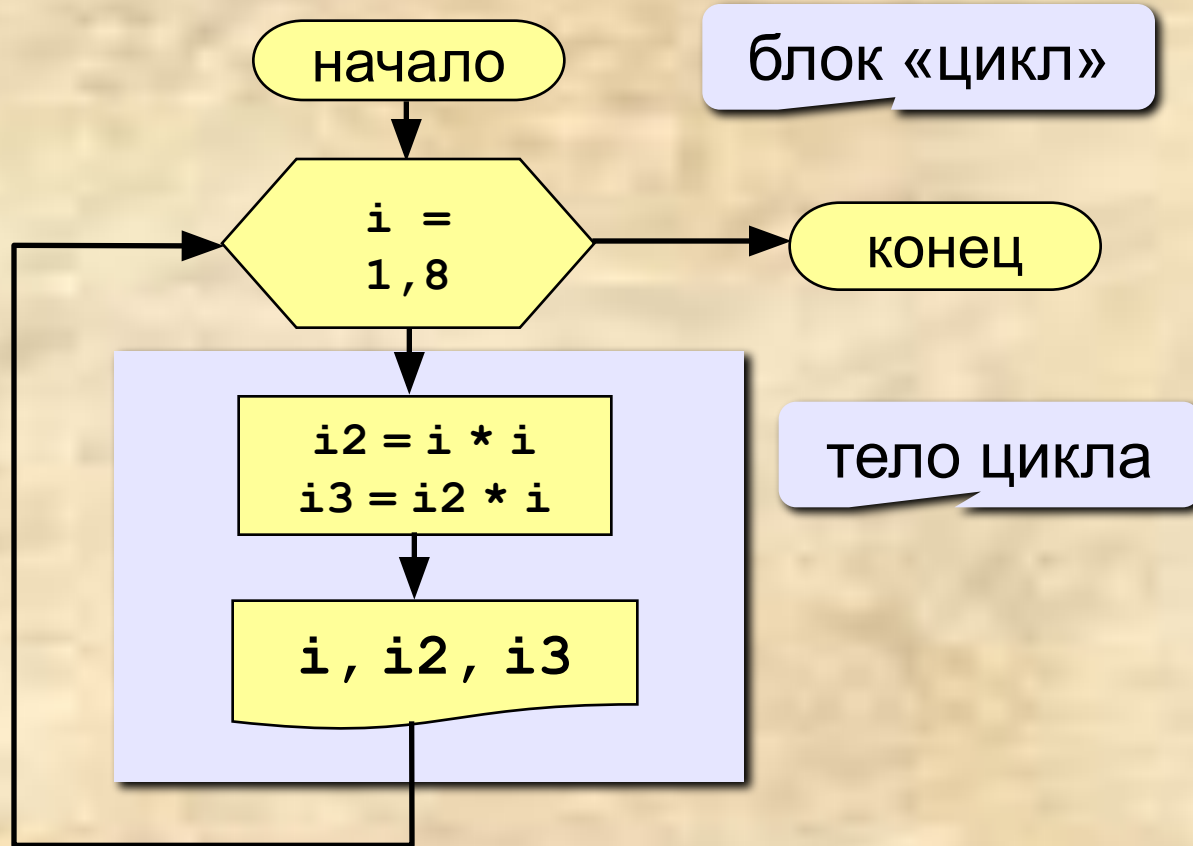
- началом цикла;
- телом цикла;
- концом цикла.

Оператор цикла  
For...Next

# Алгоритм



# Алгоритм (с блоком «цикл»)



# Программа

переменная  
цикла

начальное значение

конечное значение

```
for i=1 to 8
  i2 = i*i
  i3 = i2*i
  print i,i2,i3
next i
end
```

# Цикл с уменьшением переменной

---

**Задача.** Вывести на экран квадраты и кубы целых чисел от 8 до 1 (в обратном порядке).

**Особенность:** переменная цикла должна уменьшаться.

**Решение:**

```
for i=8 to 1 step -1
    i2 = i*i
    i3 = i2*i
    print i,i2,i3
next i

end
```

# Цикл с переменной

---

## Увеличение переменной на 1:

```
for <переменная> = <начальное значение> to  
    <конечное значение>  
    {тело цикла}  
next <переменная>
```

## Уменьшение переменной на 1:

```
for <переменная> = <начальное значение> to  
<конечное значение> step -1  
    {тело цикла}  
next <переменная>
```



# Цикл с переменной

---

## Особенности:

Увеличение переменной более чем 1:

```
for <переменная> = <начальное значение> to  
    <конечное значение> step <изменение  
переменной>  
    {тело цикла}  
next <переменная>
```

- если конечное значение меньше начального, цикл (**to**) не выполняется ни разу (проверка условия в начале цикла, цикл с предусловием)

# Как изменить шаг?

---

**Задача.** Вывести на экран квадраты и кубы нечётных целых чисел от 1 до 9.

**Особенность:** переменная цикла должна увеличиваться на 2.

**Решение:**

```
for i:=1 to 9 step 2
    i2 = i*i
    i3 = i2*i
    print i,i2,i3
end
```

выполняется  
только для  
нечётных **i**

# Задания

---

**«4»:** Ввести  $a$  и  $b$  и вывести квадраты и кубы чисел от  $a$  до  $b$ .

**Пример:**

Введите границы интервала:

4 6

4 16 64

5 25 125

6 36 216

**«5»:** Вывести квадраты и кубы 10 чисел следующей последовательности: 1, 2, 4, 7, 11, 16, ...

**Пример:**

1 1 1

2 4 8

4 16 64

...

46 2116 97336

# Программирование на языке Бейсик

**Тема . Циклы с условием**

***Итерационный цикл*** – это цикл, для которого нельзя указать число повторений, и проверка окончания которого происходит по достижению нужного условия.

Блок-схема базовых структур.

Цикл - пока (может не выполняться ни разу)

Цикл - до (выполняется хотя бы раз)

## Цикл WHILE...WEND.

### Цикл - пока

Он применяется, когда известно условие цикла, но неизвестно число повторений цикла.

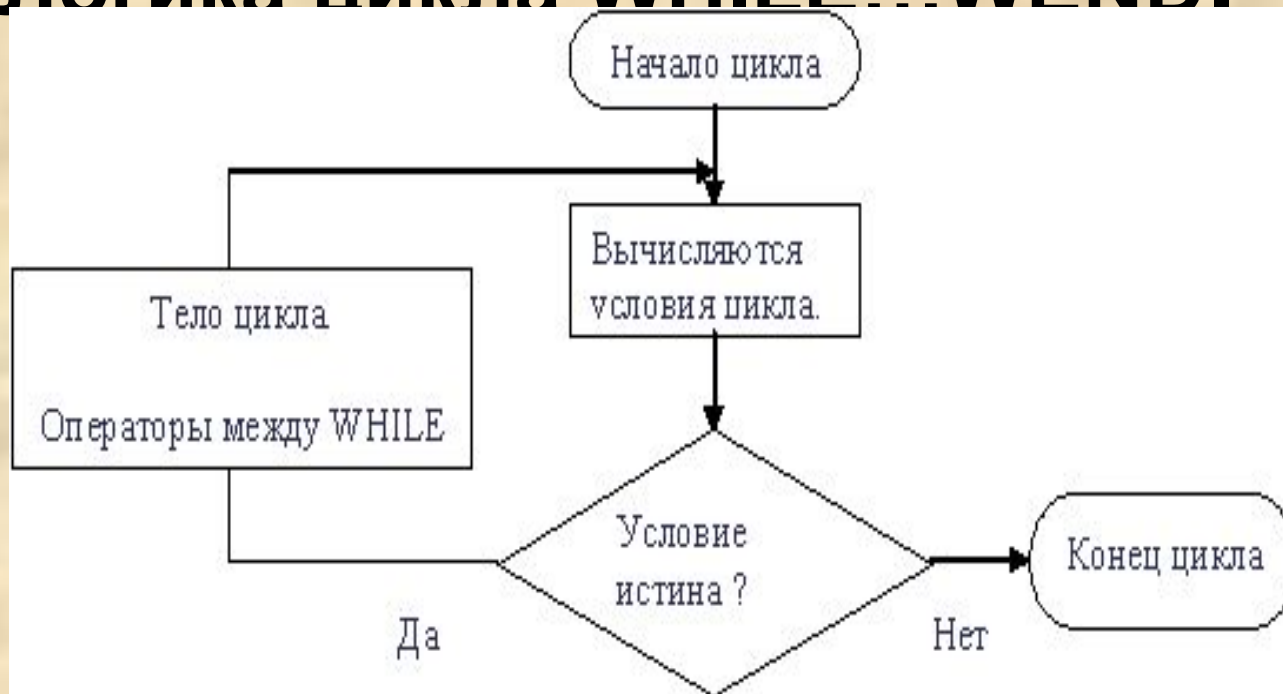
*While <условие продолжения цикла>*

*<операторы тела цикла>*

*Wend*

Цикл повторяется до тех пор, пока условие, записанное в заголовке WHILE, остается истинным.

# Логика цикла WHILE...WEND.



Работает он следующим образом. Сначала записывается оператор `while` с условием, при котором цикл выполняется. Ключевое слово `wend` аналогично по своему назначению слову `next`, т. е. является последней строкой цикла. Всякий раз, доходя до `wend`, компьютер проверяет, выполняется ли условие, указанное оператором `while`. Если оно не выполняется, то программа переходит к исполнению операторов, следующих за `wend`. Если же выполняется, то цикл повторяется снова.

# Цикл с условием

---

```
while <условие>
  {тело цикла}
wend
```

## Особенности:

- можно использовать сложные условия. В этом случае используются логические операции **and**, **or** или **not**:

```
while (a<b) and (b<c)
  {тело цикла}
wend
```

- условие пересчитывается **каждый** раз при входе в цикл



# Цикл с условием

---

## Особенности:

- если условие на входе в цикл ложно, цикл не выполняется ни разу

```
a = 4: b = 6
while a > b
    a = a - b
wend
```

- Если условие никогда не станет ложным, программа зацикливается

```
a = 4: b = 6
while a < b do
    d = a + b
wend
```

# Замена for на while и наоборот

```
for i=1 to 10
  {тело цикла}
Next i
```

```
i = 1
while i <= 10
  {тело цикла}
  i = i + 1
wend
```

```
for i=a to b step -1
  {тело цикла}
Next i
```

```
i = a
while i >= b
  {тело цикла}
  i = i - 1
wend
```

Замена цикла **for** на **while** возможна **всегда**.

Замена **while** на **for** возможна только тогда, когда можно заранее **рассчитать число шагов цикла**.

**Задача.** Вывести на экран квадраты и кубы  
целых чисел от 1 до 8 (от **a** до **b**).

```
for i=1 to 8
  i2 = i*i
  i3 = i2*i
  print i,i2,i3
next i
```

С помощью while  
... wend

С помощью  
for...next

```
i = 1
while i <= 8
  i2 = i*i
  i3 = i2*i
  print i,i2,i3
  i = i + 1
wend
```

Пример . В подъезде N ступенек. Сколько шагов будет сделано, если шагать через 3 ступеньки.

```
' KS - количество ступенек
' KH - количество шагов
INPUT "Введите количество ступенек";N
KS=0
KH=0
WHILE KS<=N
  KS=KS+3
  KH=KH+1
WEND
PRINT"Количество шагов=";KH
```

**Пример 3. Составить программу** суммирования всех натуральных чисел (начиная с 1) до момента, пока сумма не превысит 1000. На экран выводится количество слагаемых в сумме и значение суммы.

```
REM
S = 0
K = 0
WHILE S <= 1000
K = K + 1
S = S + K
WEND
PRINT K ; S
END
```

Строка  $K = K + 1$  в теле цикла называется *счетчиком цикла*, поскольку она дает количество повторений цикла. При выполнении данной программы последнее повторение оказывается лишним, поэтому на печать выводятся результаты предыдущего прохода цикла ( $K-1$  и  $S-K$ ).