ЦИКЛЫ

Цикл - многократное выполнение некоторых операторов, входящих в тело цикла.

- Циклы бывают **арифметические** и **итерационные**

- **Арифметический цикл** – это такой цикл, число повторений которого известно заранее. В Pascal такой цикл обычно реализуется с помощью оператора **for**.
- **Итерационный цикл** – это такой цикл, число повторений которого заранее неизвестно и выход из цикла производится в случае выполнения или невыполнения какого-то условия. В Pascal такие циклы обычно реализуются с помощью операторов **while** и **repeat**

Циклические операторы

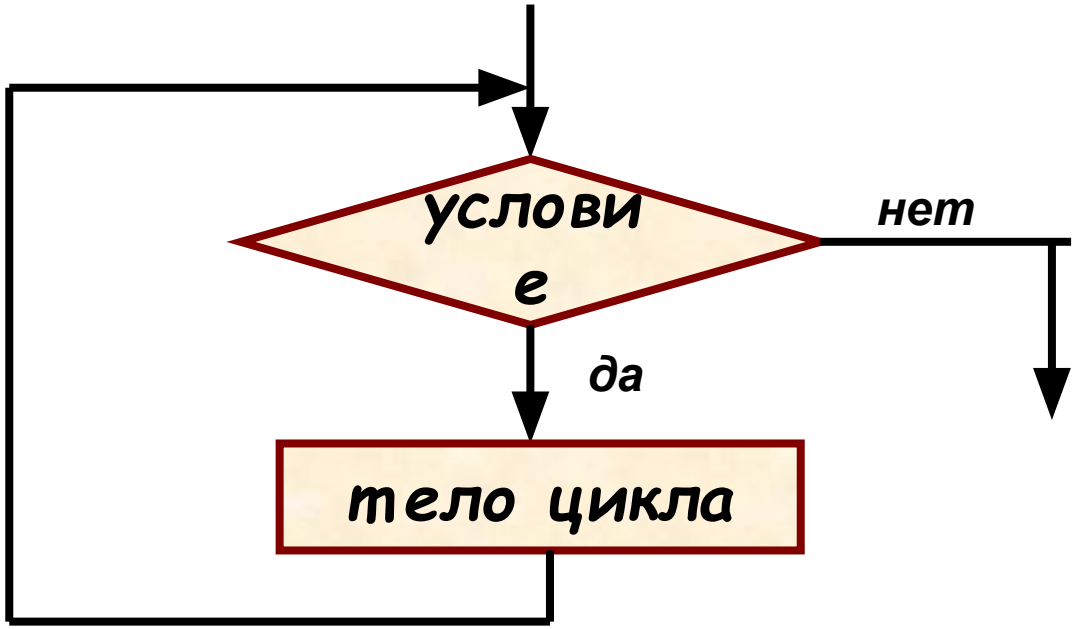
В языке Pascal имеются три оператора цикла:

- Цикл с параметром **For** (цикл на заданное число повторений);
- Цикл с предусловием **While** (цикл **ПОКА** — с предусловием);
- Цикл с постусловием **Repeat** (цикл **ДО** — с постусловием).

*Если **число повторений известно**, то лучше воспользоваться оператором цикла **с параметром**.*

Цикл с предусловием **WHILE**

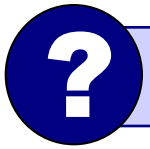
Цикл **While** сначала проверяет **условие**, и только если оно **истинно**, выполняет тело цикла.




```
While {условие} do  {оператор};
```

Циклы

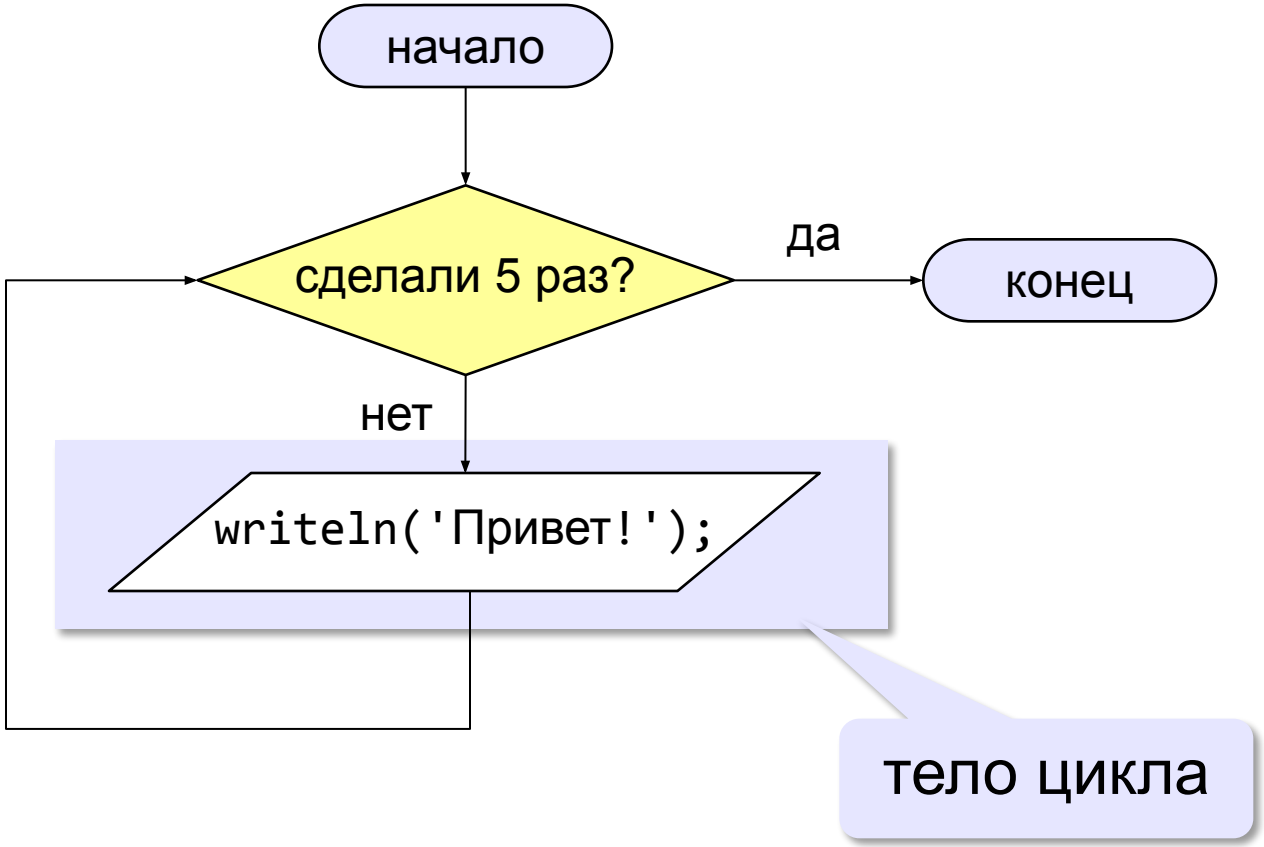
```
program qq;
begin
 writeln('Привет');
 writeln('Привет');
 writeln('Привет');
 writeln('Привет');
 writeln('Привет');
end.
```



Что плохо?

Циклы

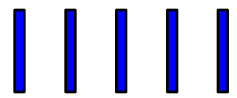
Блок-схема:



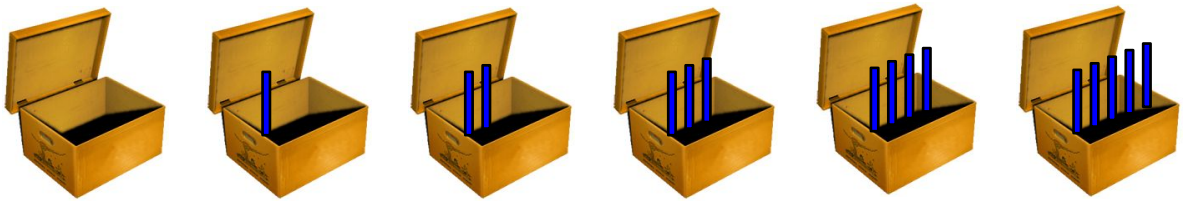
Циклы

```
program Privet;
begin
  { сделай 5 раз }
     writeln('Привет');
end.
```

Как отсчитать ровно 5 раз?

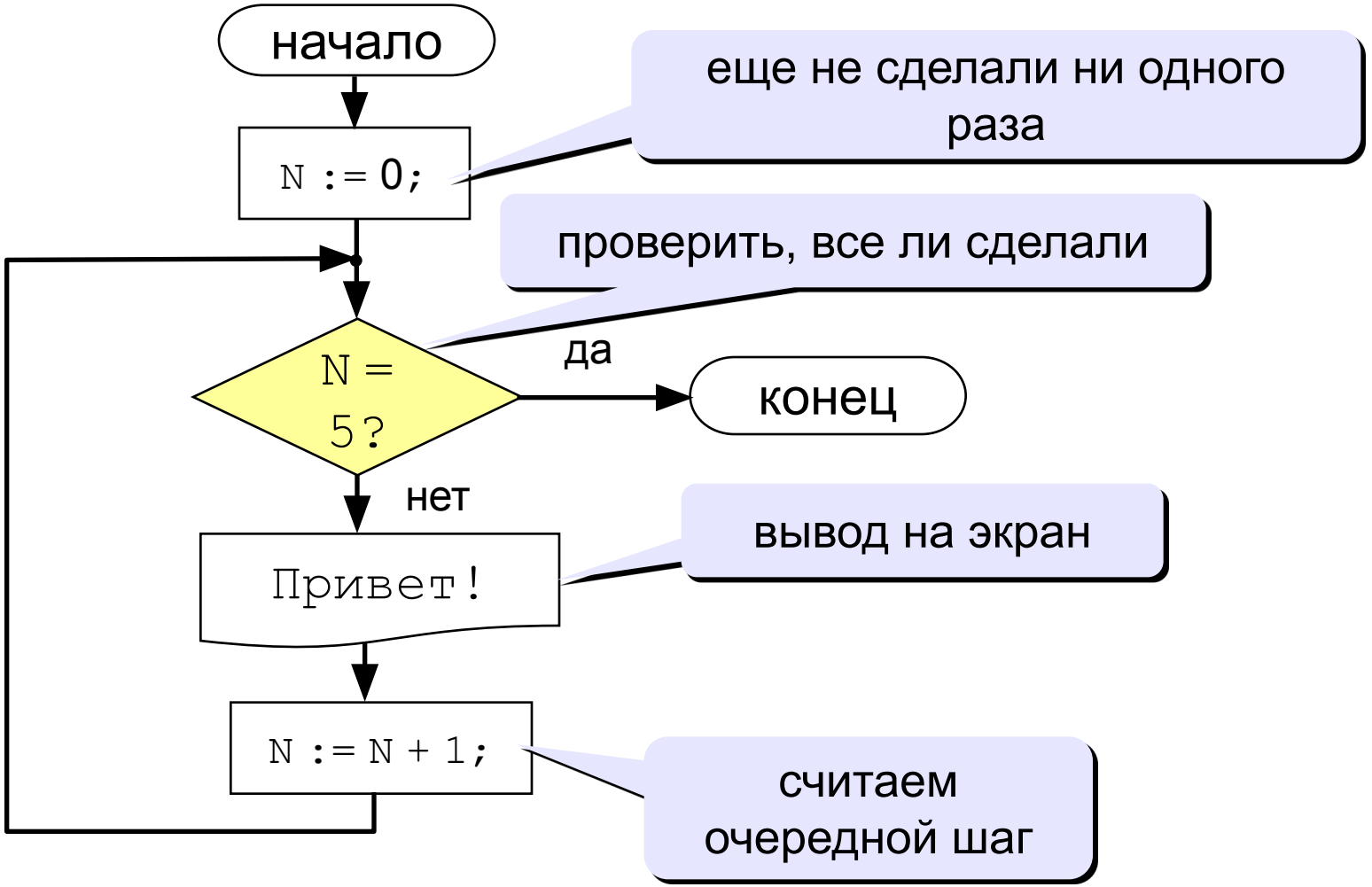


Как запоминать, сколько раз уже сделали?



```
N := N + 1;
```

Алгоритм


начало
N := 0;
еще не сделали ни одного раза
проверить, все ли сделали
N = 5?
да
конец
нет
Привет!
вывод на экран
N := N + 1;
считаем очередной шаг

Цикл с предусловием

```
program Privet2;
var N: integer;
begin
  N:=0;
  while ( N<>5 ) do begin
    writeln('Привет!');
    N:=N+1;
  end
end.
```

? Как изменить число шагов?

? Как ввести число шагов с клавиатуры?

Ввод количества

```
program Privet2;
var N , K : integer;
begin
  N:=0;
  writeln('Введите число
 шагов:');
  read(K);
  while ( N<>  ) do begin
    writeln('Привет!');
    N:=N+1;
  end
end.
```

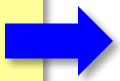
Цикл с предусловием

Вместо знаков вопроса добавьте числа и операторы так, чтобы цикл выполнился ровно 5 раз:

```
program Privet3;
var N: integer;
begin
  N:=5;
  while ( N<>  0  ) do begin
    writeln('Привет!');
    N:= N - 1;
  end
end.
```

Что получим?

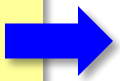
```
program Primer1;
var N: integer;
begin
   N:=1;
   while ( N<=5 ) do begin
      writeln(N);
      N:=N+1;
   end
end.
```



```
1
2
3
4
5
```

Что получим?

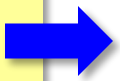
```
program Primer2;
var N: integer;
begin
   N:=1;
   while ( N<=5 ) do begin
      writeln(N);
      N:=N+2;
   end
end.
```



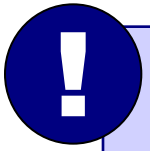
```
1
3
5
```

Что получим?

```
program Primer3;
var N: integer;
begin
   N:=2;
   while ( N<>5 ) do begin
      writeln(N);
      N:=N+2;
   end
end.
```



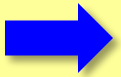
```
2
4
6
8
10
12
14
16
...
```



Условие цикла никогда не станет ложным – это зацикливание!

Что получим?

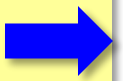
```
program Primer4;
var N: integer;
begin
   N:=1;
   while ( N<=5 ) do begin
      writeln(N*N*N);
      N:=N+1;
   end
end.
```



```
1
8
27
64
125
```

Что получим?

```
program Primer5;
var N: integer;
begin
   N:=5;
   while ( N>=1 ) do begin
      writeln(N*N*N);
      N:=N-1;
   end
end.
```



```
125
64
27
8
1
```

Задание 1

«3»: Ввести натуральное число вывести квадраты и кубы всех чисел от 1 до этого числа.

Пример:

```
Введите натуральное число:
3
1: 1 1
2: 4 8
3: 9 27
```

«4»: Ввести два целых числа a и b (a ≤ b) и вывести квадраты все чисел от a до b.

Пример:

```
Введите два числа:
4 5
4*4=16
5*5=25
```

Задание 1

«5»: Ввести два целых числа a и b (a ≤ b) и вывести сумму квадратов всех чисел от a до b.

Пример:

```
Введите два числа:
4 10
Сумма квадратов 371
```

Цикл с неизвестным числом шагов

Задача: Ввести целое число (<2000000) и определить число цифр в нем.

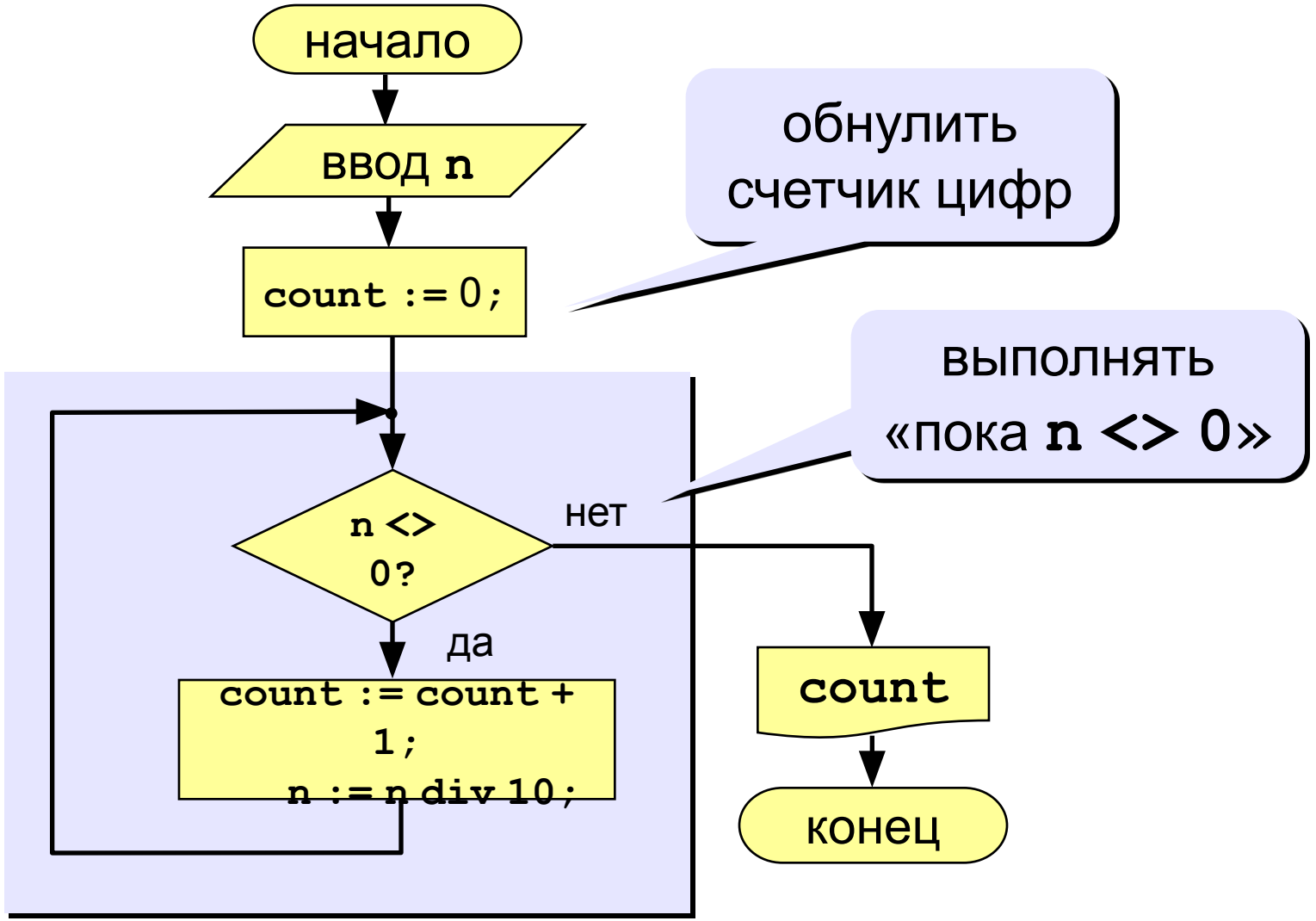
Идея решения: Отсекаем последовательно последнюю цифру, увеличиваем счетчик.

n	count
123	0
12	1
1	2
0	3

Проблема: Неизвестно, сколько шагов надо сделать.

Решение: Надо остановиться, когда **n = 0**, т.е. надо делать «пока **n <> 0**».

Алгоритм


начало
ввод n
count := 0;
обнулить счетчик цифр
выполнять «пока n <> 0»
n <> 0?
нет
да
count := count + 1;
n := n div 10;
count
конец

Программа

```
program qq;
var n, count, n1: integer;
begin
  writeln('Введите целое число');
  read(n); n1 := n;
  count := 0;
  while n <> 0 do begin
    count := count + 1;
    n := n div 10;
  end;
  writeln('В числе ', n1, ' нашли ',
          count, ' цифр');
end.
```

выполнять «пока n <> 0»

? Что плохо?

Цикл с условием

```
while <условие> do begin
   {тело цикла}
end;
```

Особенности:

- можно использовать сложные условия:

```
while (a < b) and (b < c) do begin
   {тело цикла}
end;
```

- если в теле цикла только один оператор, слова **begin** и **end** можно не писать:

```
while a < b do
   a := a + 1;
```

Цикл с предусловием

Особенности:

- условие пересчитывается **каждый раз** при входе в цикл
- если условие на входе в цикл ложно, цикл не выполняется ни разу

```
a := 4; b := 6;
while a > b do
   a := a - b;
```

- если условие никогда не станет ложным, программа **зацикливается**

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

Сколько раз выполняется цикл?

```
a := 4; b := 6;
while a < b do a := a + 1;
```

2 раза
a = 6

```
a := 4; b := 6;
while a < b do a := a + b;
```

1 раз
a = 10

```
a := 4; b := 6;
while a > b do a := a + 1;
```

0 раз
a = 4

```
a := 4; b := 6;
while a < b do b := a - b;
```

1 раз
b = -2

```
a := 4; b := 6;
while a < b do a := a - 1;
```

зацикливание

Задание 2

«3»: Ввести целое число и определить, верно ли, что в нём ровно 3 цифры.

Пример:

```
Введите число:      Введите число:
123                 1234
Да.                 Нет.
```

«4»: Ввести целое число и найти сумму его цифр.

Пример:

```
Введите целое число:
1234
Сумма цифр числа 1234 равна 10.
```

Задание 2

«5»: Ввести целое число и определить, верно ли, что в его записи есть две одинаковые цифры, стоящие рядом.

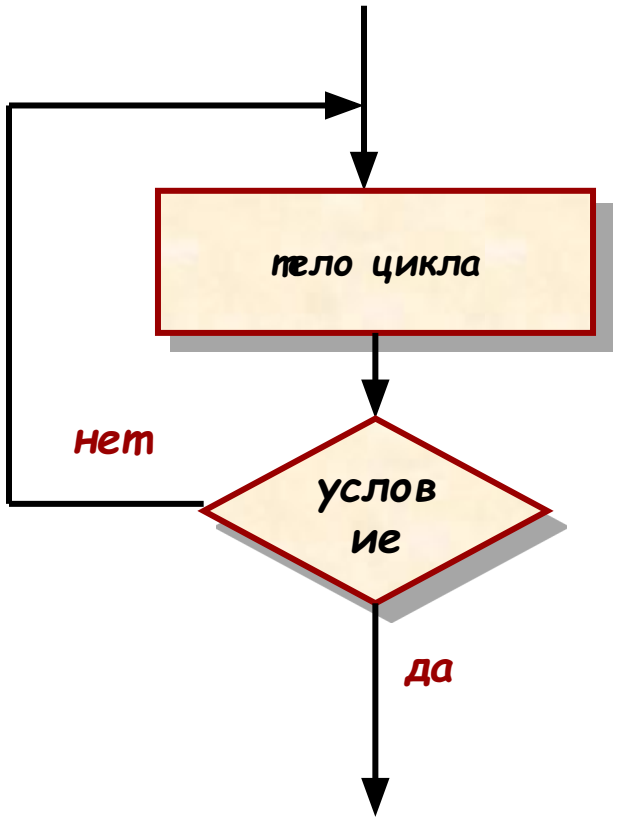
Пример:

```
Введите целое число:        Введите целое число:
1232                        1224
Нет.                        Да.
```

Цикл с постусловием Repeat («ДО ТЕХ ПОР»)

Цикл **Repeat** сначала выполняет тело цикла, а лишь затем проверяет условие.

```
Repeat
   {тело_цикла}
Until {условие};
```



- Нет необходимости в цикле **Repeat** использовать *составной оператор*, т. к. данная конструкция предусматривает выполнение не одного, а *нескольких операторов*, заключённых между словами **Repeat** и **Until**.
- Тело цикла с постусловием выполняется *хотя бы один раз*.

Цикл с постусловием

Задача: Ввести целое **положительное** число (<2000000) и определить число цифр в нем.

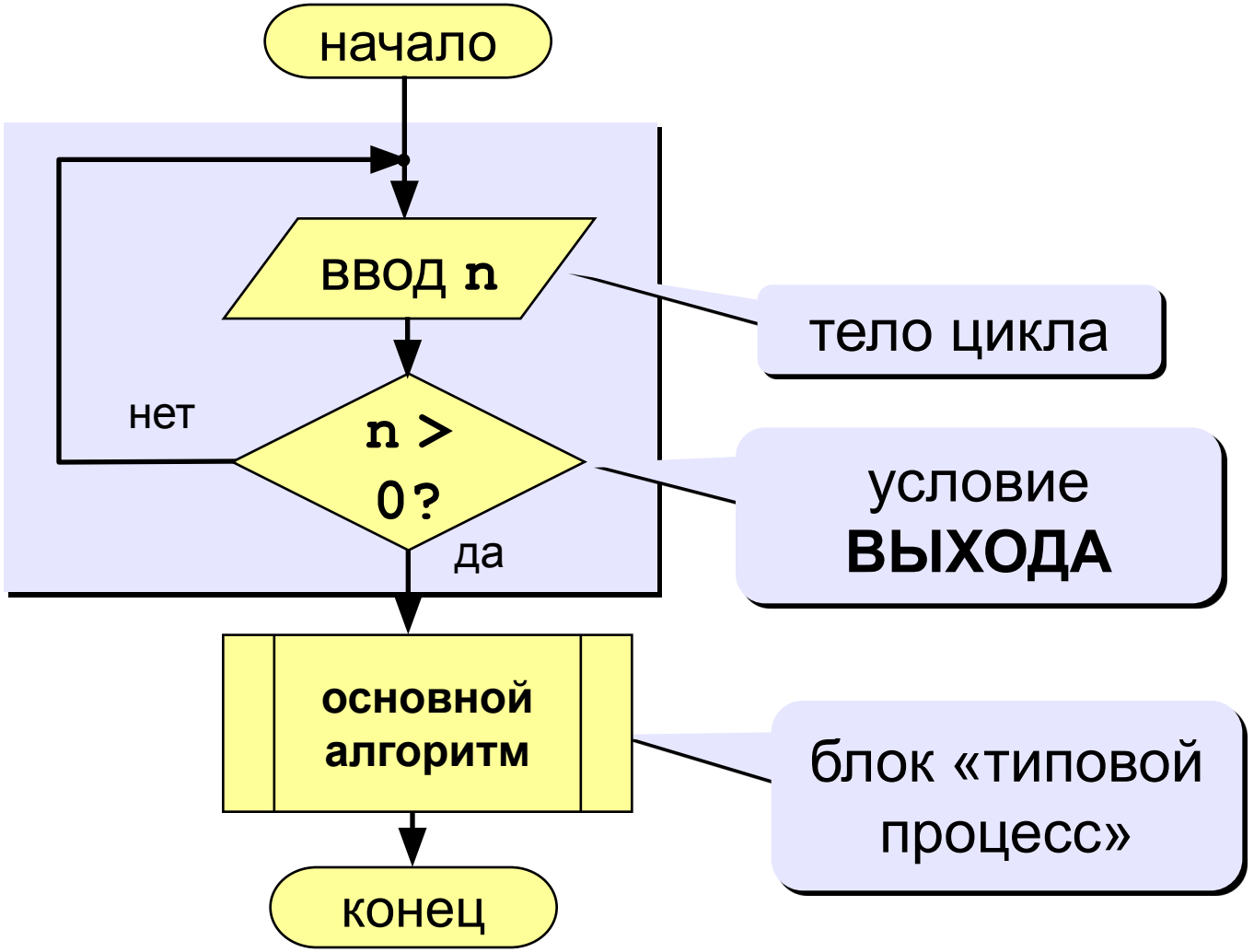
Проблема: Как не дать ввести отрицательное число или ноль?

Решение: Если вводится неверное число, вернуться назад к вводу данных (цикл!).

Особенность: Один раз тело цикла надо сделать в любом случае => проверку условия цикла надо делать в конце цикла (цикл с **постусловием**).

Цикл с постусловием – это цикл, в котором проверка условия выполняется в конце цикла.

Цикл с постусловием: алгоритм



Программа

```
program qq;
var n: integer;
begin
   repeat
     writeln('Введите положительное число');
     read(n);
   until n > 0;
    ... { основной алгоритм }
end.
```

условие **ВЫХОДА**

Особенности:

- тело цикла всегда выполняется хотя бы один раз
- после слова **until** ("до тех пор, пока не…") ставится условие **ВЫХОДА** из цикла

Сколько раз выполняется цикл?

```
a := 4; b := 6;
repeat a := a + 1; until a > b;
```

3 раза
a = 7

```
a := 4; b := 6;
repeat a := a + b; until a > b;
```

1 раз
a = 10

```
a := 4; b := 6;
repeat a := a + b; until a < b;
```

зацикливание

```
a := 4; b := 6;
repeat b := a - b; until a < b;
```

2 раза
b = 6

```
a := 4; b := 6;
repeat a := a + 2; until a < b;
```

зацикливание

Задание 3 (с защитой от неверного ввода)

«4»: Ввести натуральное число и определить, верно ли, что сумма его цифр равна 10.

Пример:

```
Введите число >= 0:
-234
Нужно положительное число.
Введите число >= 0:
1234
Да
```

```
Введите число >= 0:
1233
Нет
```

«5»: Ввести натуральное число и определить, какие цифры встречаются несколько раз.

Пример:

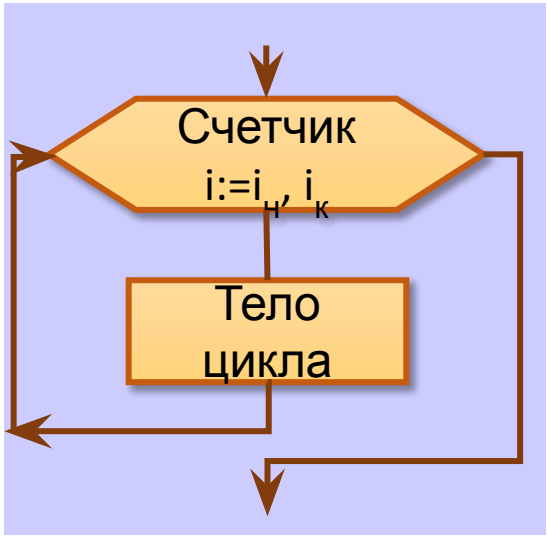
```
Введите число >= 0:
2323
Повторяются: 2, 3
```

```
Введите число >= 0:
1234
Нет повторов.
```

Цикл с параметром (счетчиком) FOR

Цикл на заданное число повторений с *возрастающим* или *убывающим* значением параметра.

```
for счетчик := начальное to конечное do
                значение      значение
begin
  тело цикла
end
```



Принцип работы: Сначала счетчику цикла присваивается начальное значение. Если это значение не больше конечного значения, то выполняется тело цикла. Затем значение счетчика увеличивается на 1 и опять сравнивается с конечным значением. Если оно по-прежнему не больше конечного значения, то оператор выполняется еще раз и так далее.

Замечание: если тело цикла состоит из одного оператора, то begin и end можно опустить

Цикл с параметром (счетчиком) FOR

```
for счетчик := начальное значение downto конечное значение do
begin
  тело цикла
end
```

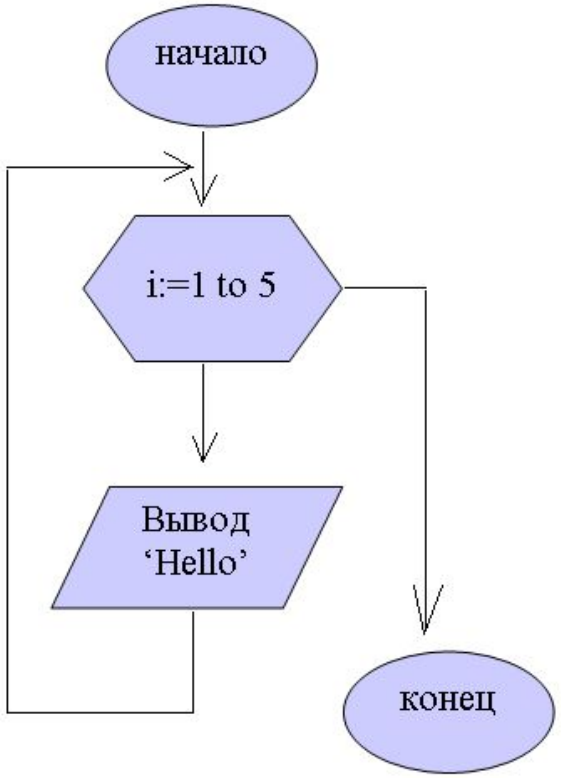
Принцип работы: как и в первом случае, пока начальное значение не меньше конечного значения, то выполняется тело цикла. Затем значение счетчика уменьшается на 1

Замечание:

- параметр — порядковый тип;
- в теле цикла нельзя менять параметр цикла;
- параметр цикла увеличивается или уменьшается на 1

Задача: Вывести на экран 5 раз «Hello».

```
Program primer 1;
var i: integer;
begin
for i:=1 to 5 do
  begin
    writeln('Hello');
  end;
end.
```



Здесь переменная i запоминает сколько раз выполнилась повторяющаяся команда (тело цикла)

Цикл с параметрами

Задача: вывести кубы натуральных чисел от 1 до 8.

```
program Cubes;
var N, cubeN: integer;
begin
  N:=
  1;
  while ( N <= 8 ) do begin
    cubeN:=N*N*N;
    writeln(cubeN);
    N:=N+1;
  end
end.
```

3 действия с N

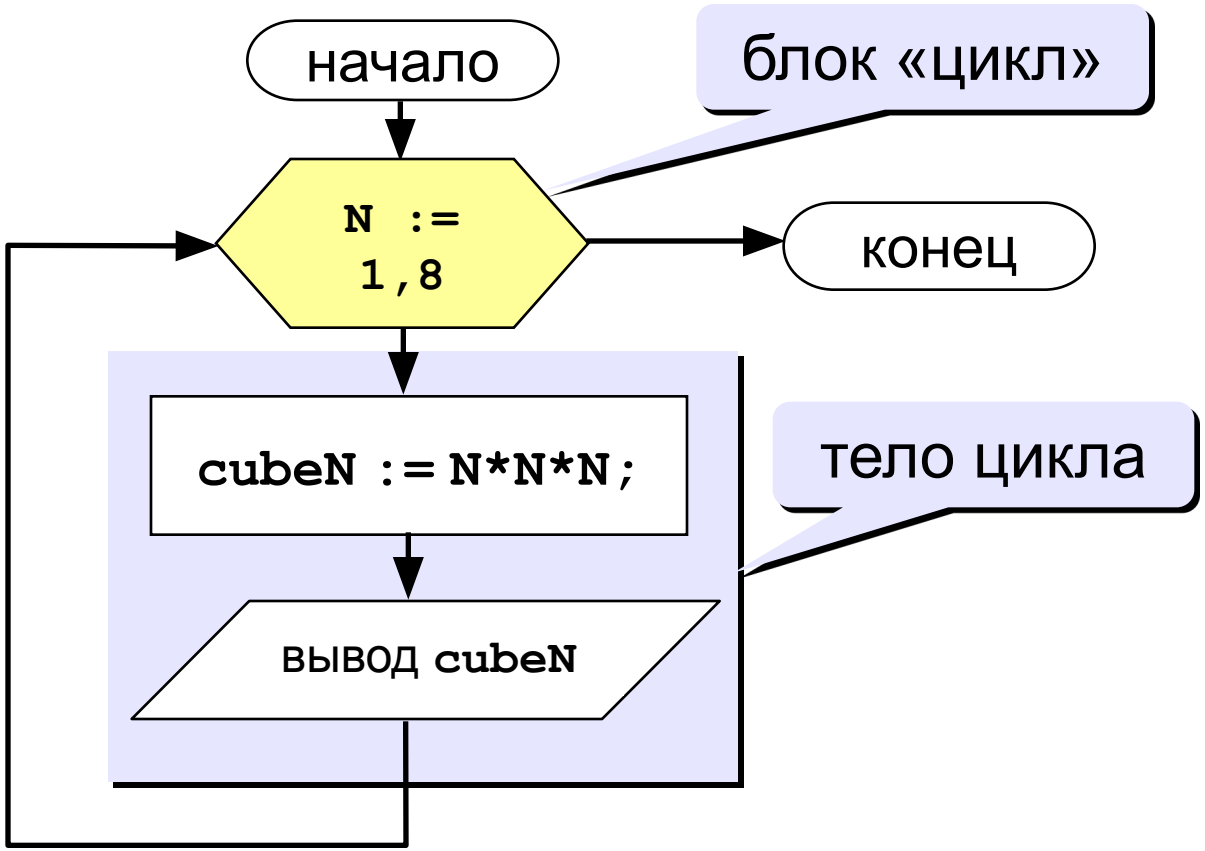
Цикл с параметрами

Задача: вывести кубы натуральных чисел от 1 до 8.

```
program Cubes2;
var N, cubeN: integer;
begin
  for N:=1 to 8 do begin
    cubeN:=N*N*N;
    writeln(cubeN)
  end
end.
```

для 1,2,3,…,8

Алгоритм (с блоком «цикл»)



Программа

```
program qq;
var N, cubeN: integer;
begin
  for N:=1 to 8 do begin
    cubeN := N*N*N;
    writeln(N:4, cubeN:4)
  end
end.
```

переменная цикла

начальное значение

конечное значение

Цикл с уменьшением переменной

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

Особенность: переменная цикла должна уменьшаться.

Решение:

```
for N:=8 downto 1 do begin
  cubeN := N*N*N;
  writeln(N:4, cubeN:4)
end;
```

Цикл с переменной

Увеличение переменной на 1:

```
for <переменная> := <начальное значение> to
    <конечное значение> do begin
  {тело цикла}
end;
```

Уменьшение переменной на 1:

```
for <переменная> := <начальное значение>
          downto
    <конечное значение> do begin
  {тело цикла}
end;
```

Цикл с переменной

Особенности:

- переменная цикла может быть только целой (**integer**)
- шаг изменения переменной цикла всегда равен 1 (**to**) или -1 (**downto**)
- если в теле цикла только один оператор, слова **begin** и **end** можно не писать:

```
for i:=1 to 8 do
  writeln('Привет');
```

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

Сколько раз выполняется цикл?

```
a:=1;
for i:=1 to 3 do a:=a+1;
```

a = 4

```
a:=1;
for i:=3 to 1 do a:=a+1;
```

a = 1

```
a:=1;
for i:=1 downto 3 do a:=a+1;
```

a = 1

```
a:=1;
for i:=3 downto 1 do a:=a+1;
```

a = 4

Как изменить шаг?

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

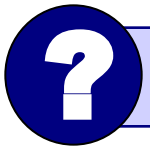
Особенность: переменная цикла должна увеличиваться на 2.

Проблема: в Паскале шаг может быть 1 или -1.

Решение:

```
for N:=1 to 9 do begin
    if N mod 2 = 1 then begin
        cubeN := N*N*N;
        writeln(N:4, cubeN:4);
    end;
end;
```

выполняется только для нечетных **i**



Что плохо?

Как изменить шаг? – II

Идея: Надо вывести всего 5 чисел, переменная **k** изменяется от 1 до 5. Начальное значение **N** равно 1, с каждым шагом цикла **N** увеличивается на 2.

Решение:

```
N := 1;
for k:=1 to 5 do begin
  cubeN := N*N*N;
  writeln(N:4, cubeN:4);
  N := N + 2;
end;
```

Как изменить шаг? – III

Идея: Надо вывести всего 5 чисел, переменная **k** изменяется от 1 до 5. **Зная k, надо рассчитать N.**

k	1	2	3	4	5
N	1	3	5	7	9

N = 2k-1

Решение:

```
for k:=1 to 5 do begin
   N := 2*k - 1;
   cubeN := N*N*N;
   writeln(N:4, cubeN:4);
 end;
```

Замена for на while и наоборот

```
for i:=1 to 10 do begin
  {тело цикла}
end;
```

```
i := 1;
while i <= 10 do begin
  {тело цикла}
  i := i + 1;
end;
```

```
for i:=a downto b do
  begin
  {тело цикла}
end;
```

```
i := a;
while i >= b do begin
  {тело цикла}
  i := i - 1;
end;
```

Замена цикла for на while возможна всегда.

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

Задание 4

«3»: Ввести два целых числа a и b (a ≤ b) и вывести кубы всех чисел от a до b.

Пример:

```
Введите два числа:
4 6
4*4*4=64
5*5*5=125
6*6*6=216
```

«4»: Ввести натуральное число A и вывести числа от A до 1 (через одно) в порядке убывания.

Пример:

```
Введите натуральное число:
8
Ответ: 8 6 4 2
```

Задание 4

«5»: Ввести целое число a и вывести сумму квадратов всех чисел от 1 до a с шагом 0.1.

Пример:

```
Введите последнее число:
3
Сумма 91.7
```

$1^2 + 1.1^2 + 1.2^2 + \ldots + a^2$

Задание 4-2

«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
```

Анализ работы трёх циклических операторов

For

```
For f:=a to b Do
   S := S + f;
```

While

```
F := a;
While f<=b do Begin
   S := S + f;
   F := F + 1;
End;
```

Repeat

```
F := a;
Repeat
   S := S + f;
   F := F + 1;
Until f>b;
```

- Выбор модели цикла, зависит лишь от удобства его использования в конкретной ситуации.
- Мы практически всегда можем вместо одного вида цикла воспользоваться другим.

- Инициализируем начальное значение.
- Наращиваем «счётчик цикла».

Зацикливание и прерывание циклов

For

```
{бесконечный цикл}
x:=0;
for i:=1 to 10 do
 begin
  x:=x+1;
  i:=1;
 end;
```

```
{прерывание цикла}
x:=0; i:=1;
while i<=10 do
 i:=i+10;
```

While

```
{бесконечный цикл}
x:=0; i:=1;
while i<=10 do
 x:=x+1;
```

```
{прерывание цикла}
x:=0;
for i:=1 to 10 do
 begin
  x:=x+1;
  i:=10;
 end;
```

Repeat

```
{бесконечный цикл}
x:=0; i:=1;
repeat
x:=x+1;
until i>=10
```

```
{прерывание цикла}
x:=0; i:=1;
repeat
i:=i+10;
until i>=10
```

Для гибкого управления циклическими операторами используются процедуры:

- **Break** — выход из цикла;
- **Continue** — завершение очередного прохода цикла.