

Алгоритмы циклической структуры, программирование на языке Pascal

Часть 2

8 «Б» и 10 «Б» классы

The logo for the Pascal programming language, featuring the word "Pascal" in a bold, black, sans-serif font. Above the text is a thin, dark blue arc that starts above the 'P' and ends above the 'l', resembling a stylized 'P' or a protective shield.

Pascal

Цикл (повтор) – это...

такая форма организации действий, при которой одна и та же последовательность действий повторяется несколько раз (или ни разу) до тех пор, пока выполняется некоторое условие.

Циклы бывают:

- *Циклы со счетчиком (ДЛЯ), в которых тело цикла выполняется определенное количество раз;*
- *Циклы с условием (ПОКА и ДО), в которых тело цикла выполняется до тех пор, пока выполняется условие.*

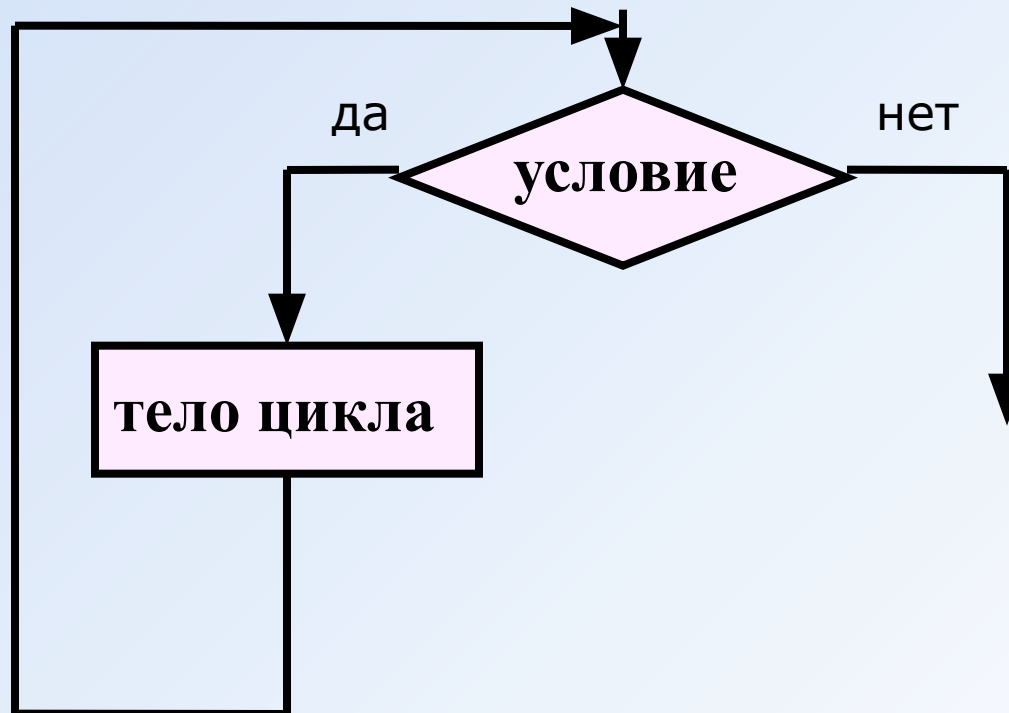
Цикл с неизвестным числом шагов



- *Циклы с условием (ПОКА и ДО), в которых тело цикла выполняется до тех пор, пока выполняется условие.*

Цикл с предусловием

- пока условие истинно, выполняется тело цикла



Цикл с предусловием

ПОКА

ВЫПОЛНЯТЬ

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

Особенности:

- МОЖНО

```
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

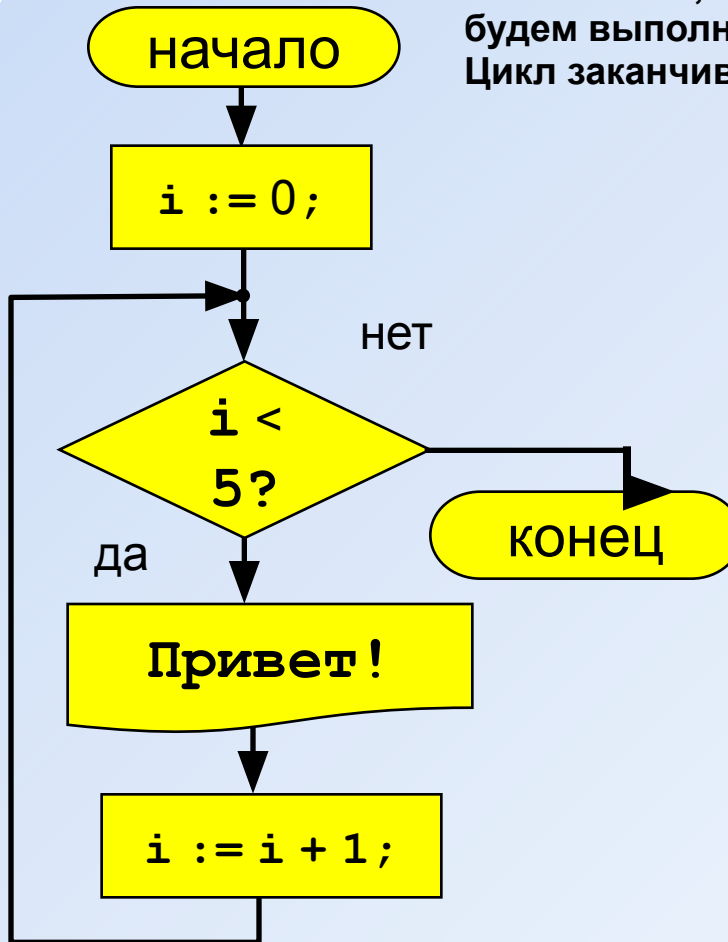
```
a := 4; b := 6;  
while a > b do  
    a := a - b;
```

- если условие никогда не станет ложным, программа закливается


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

Вывести на экран «Привет!» 5 раз

i – считает сколько раз напечатали «Привет!»
Сначала $i=0$, затем проверяется условие $i < 5$, *пока* оно выполняется будем выполнять: печатать «Привет!» и значение i увеличивать на 1.
Цикл заканчивается как только i примет значение 5.



i	Действия	Экран
0	Еще ничего не напечатано. Проверка условия. Печать <i>Привет!</i> $i=0+1=1$	<i>Привет!</i>
1	Проверка условия. Печать <i>Привет!</i> $i=1+1=2$	<i>Привет! Привет!</i>
2	Проверка условия. Печать <i>Привет!</i> $i=2+1=3$	<i>Привет! Привет! Привет!</i>
3	Проверка условия. Печать <i>Привет!</i> $i=3+1=4$	<i>Привет! Привет! Привет! Привет!</i>
4	Проверка условия. Печать <i>Привет!</i> $i=4+1=5$	<i>Привет! Привет! Привет! Привет! Привет!</i>
5	Проверка условия. Выход из цикла, конец.	



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

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

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;
```

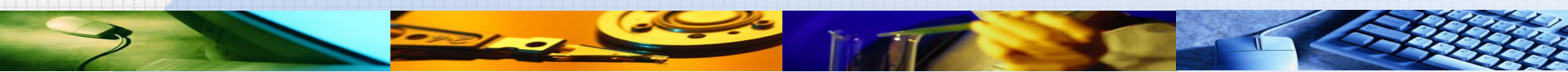
```
for i:=a downto b do
  begin
    {тело цикла}
  end;
```

```
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**.

Ввести целое число и найти сумму его цифр.



Пример: Введите целое число: **1234**

Сумма цифр числа 1234 равна 10.

Надо определить цифру и прибавить её к предыдущей сумме.

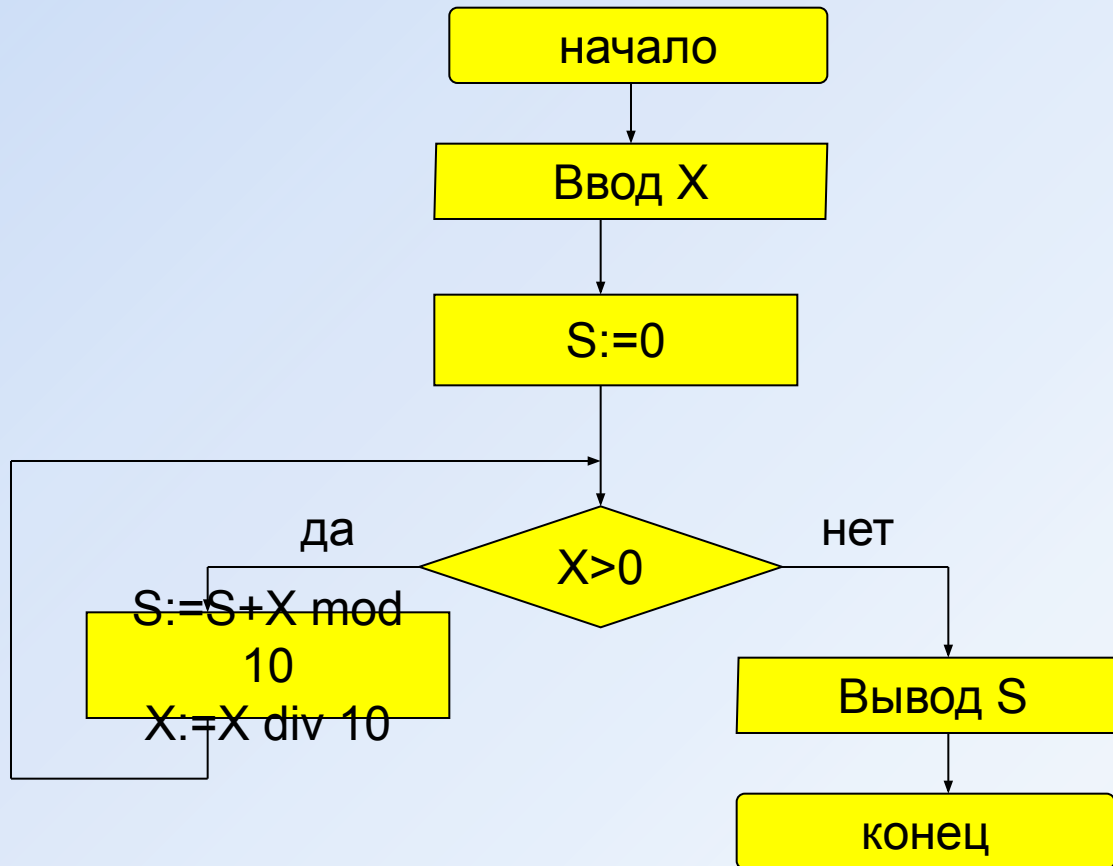
Цифра – остаток от деления числа X на 10.

Число X – целая часть от деления числа X на 10.

Действия выполняем до тех пор пока X не станет ≤ 0

Цифра	Сумма
	0
4	$0+4=4$
3	$4+3=7$
2	$7+2=9$
1	$9+1=10$

начало



Программа

```
program qq;  
var x, S: integer;  
begin  
  read (x);  
  S:=0;  
  while x > 0 do  
    begin  
      S := S + X mod 10;  
      X := X div 10;  
    end;  
  writeln(S);  
end.
```

Последовательности

Примеры:

- 1, 2, 3, 4, 5, ...

$$a_n = n$$

$$a_1 = 1, a_{n+1} = a_n + 1$$

- 1, 2, 4, 7, 11, 16, ...

$$a_1 = 1, a_{n+1} = a_n + n$$

- 1, 2, 4, 8, 16, 32, ...

$$a_n = 2^{n-1}$$

$$a_1 = 1, a_{n+1} = 2a_n$$

- $\frac{1}{2}, \frac{1}{2}, \frac{3}{8}, \frac{1}{4}, \frac{5}{32}, \dots$

$$\frac{1}{2}, \frac{2}{4}, \frac{3}{8}, \frac{4}{16}, \frac{5}{32}, \dots$$

$$a_n = \frac{b_n}{c_n}$$

$$b_1 = 1, b_{n+1} = b_n + 1$$

$$c_1 = 2, c_{n+1} = 2c_n$$

Найти сумму всех элементов
последовательности,

$$1, \frac{1}{2}, \frac{2}{4}, \frac{3}{8}, \frac{4}{16}, \frac{5}{32}, \dots$$

которые по модулю больше 0,001:

Ход решения:

$$S = 1 - \frac{1}{2} + \frac{2}{4} - \frac{3}{8} + \frac{4}{16} - \frac{5}{32} + \dots$$

Элемент последовательности (начиная с №2):

$$a = z \frac{b}{c}$$

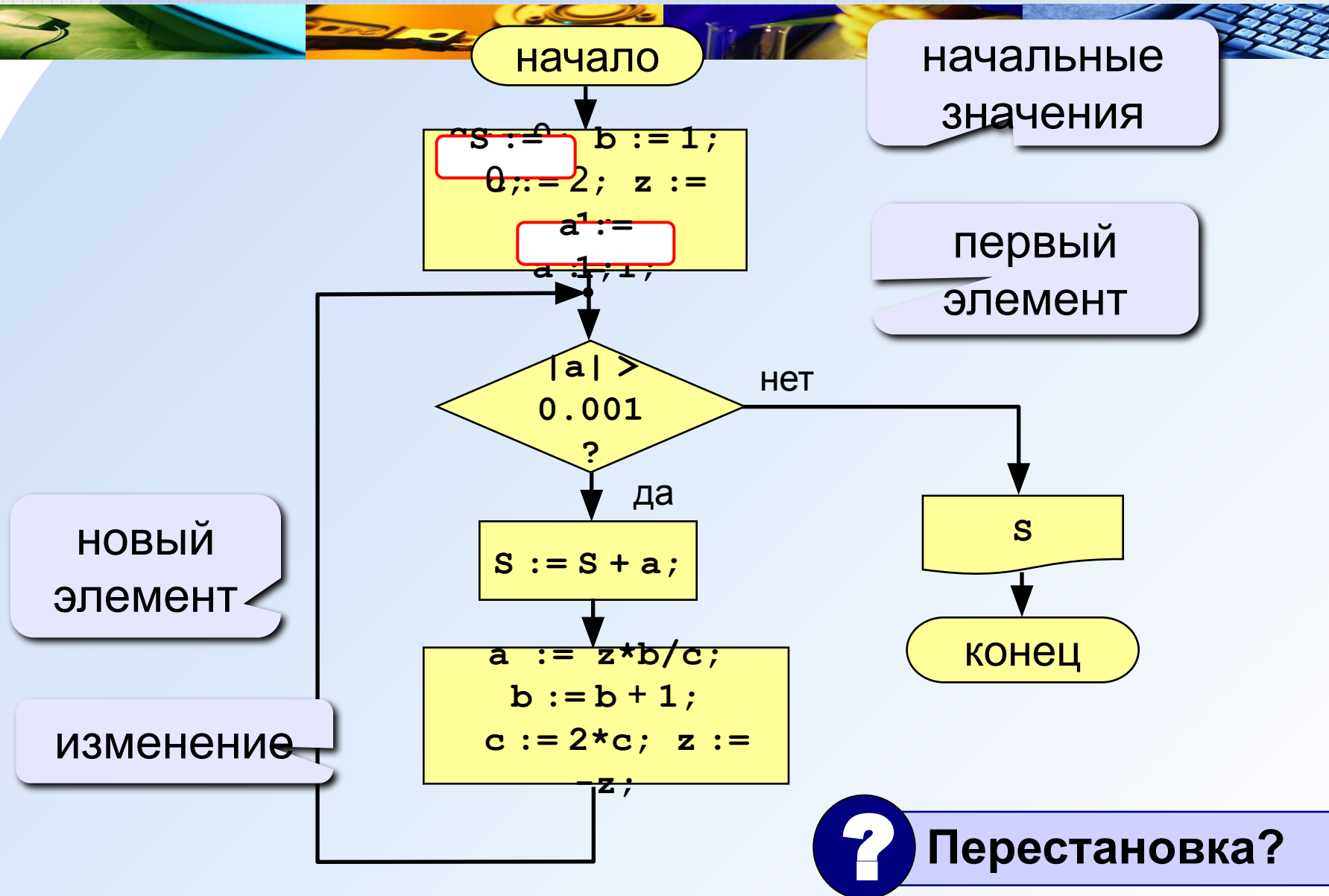
n	1	2	3	4	5	...
b	1	2	3	4	5	...
c	2	4	8	16	32	...
z	-1	1	-1	1	-1	...

`b := b+1;`

`c := 2*c;`

`z := -z;`

Алгоритм



Программа

```
program qq;  
var b, c, z: integer;  
    S, a: real;  
begin  
    S := 0; z := -1;  
    b := 1; c := 2; a := 1;  
    while abs(a) > 0.001 do begin  
        S := S + a;  
        a := z * b / c;  
        z := - z;  
        b := b + 1;  
        c := c * 2;  
    end;  
    writeln('S =', S:10:3);  
end.
```

начальные
значения

увеличение
суммы

расчет элемента
последовательности

переход к
следующему
слагаемому

Задания

На «4»: Найти сумму элементов последовательности с точностью 0,001:

$$S = 1 + \frac{2}{3 \cdot 3} - \frac{4}{5 \cdot 9} + \frac{6}{7 \cdot 27} - \frac{8}{9 \cdot 81} + \dots$$

Ответ:

$$S = 1.157$$

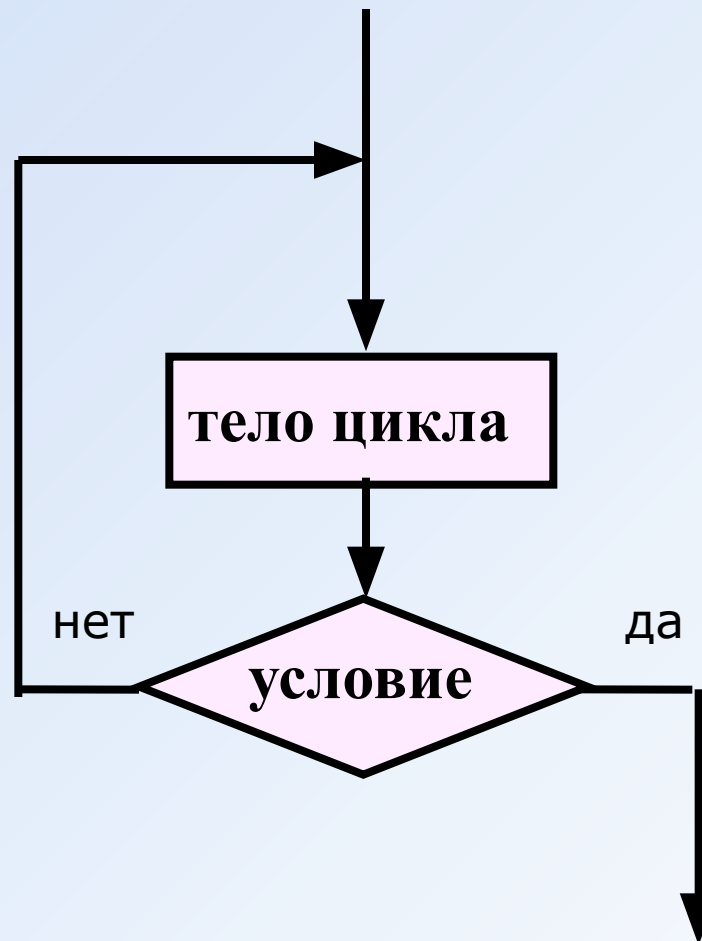
На «5»: Найти сумму элементов последовательности с точностью 0,001:

$$\text{Ответ: } S = 1 + \frac{2}{2 \cdot 3} - \frac{4}{3 \cdot 9} + \frac{6}{5 \cdot 27} - \frac{8}{8 \cdot 81} + \frac{10}{13 \cdot 243} - \dots$$

$$S = 1.220$$

Цикл с постусловием

- повторяется выполнение тела цикла до того, как условие станет истинным



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



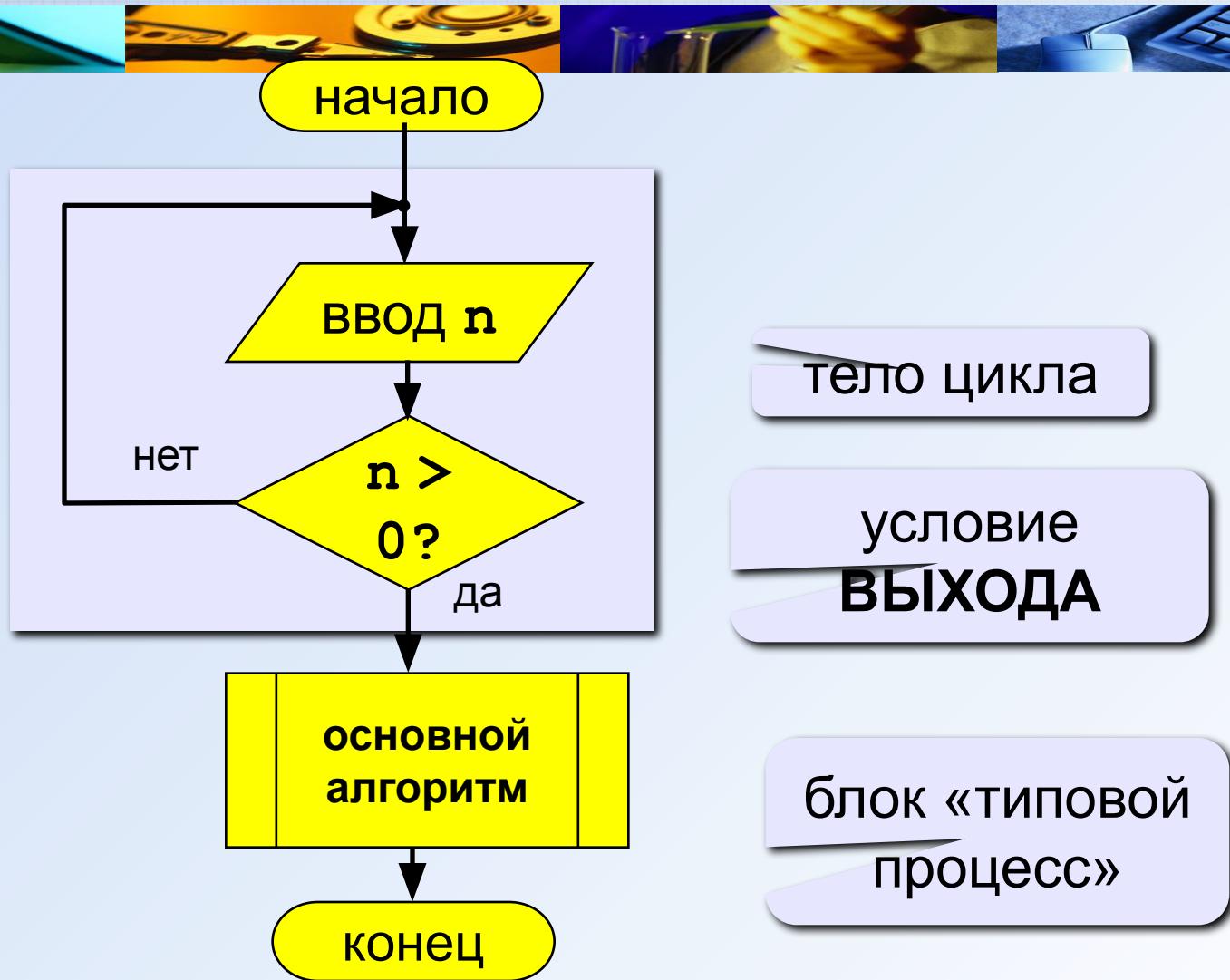
Проблема: Как не дать ввести отрицательное число или ноль?

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.

Алгоритм



Программа

```
program qq;  
var n: integer;  
begin  
  repeat  
    writeln('Введите положительное число');  
    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

Задания (с защитой от неверного ввода)

На«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

На«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.