

Лекция 7

Программирование циклических алгоритмов

Циклы

Цикл – это многократное выполнение одинаковых действий.

Виды циклов

По числу повторов:

- цикл с **известным** числом шагов (*арифметический*)
- цикл с **неизвестным** числом шагов (цикл с условием, *итерационный*)

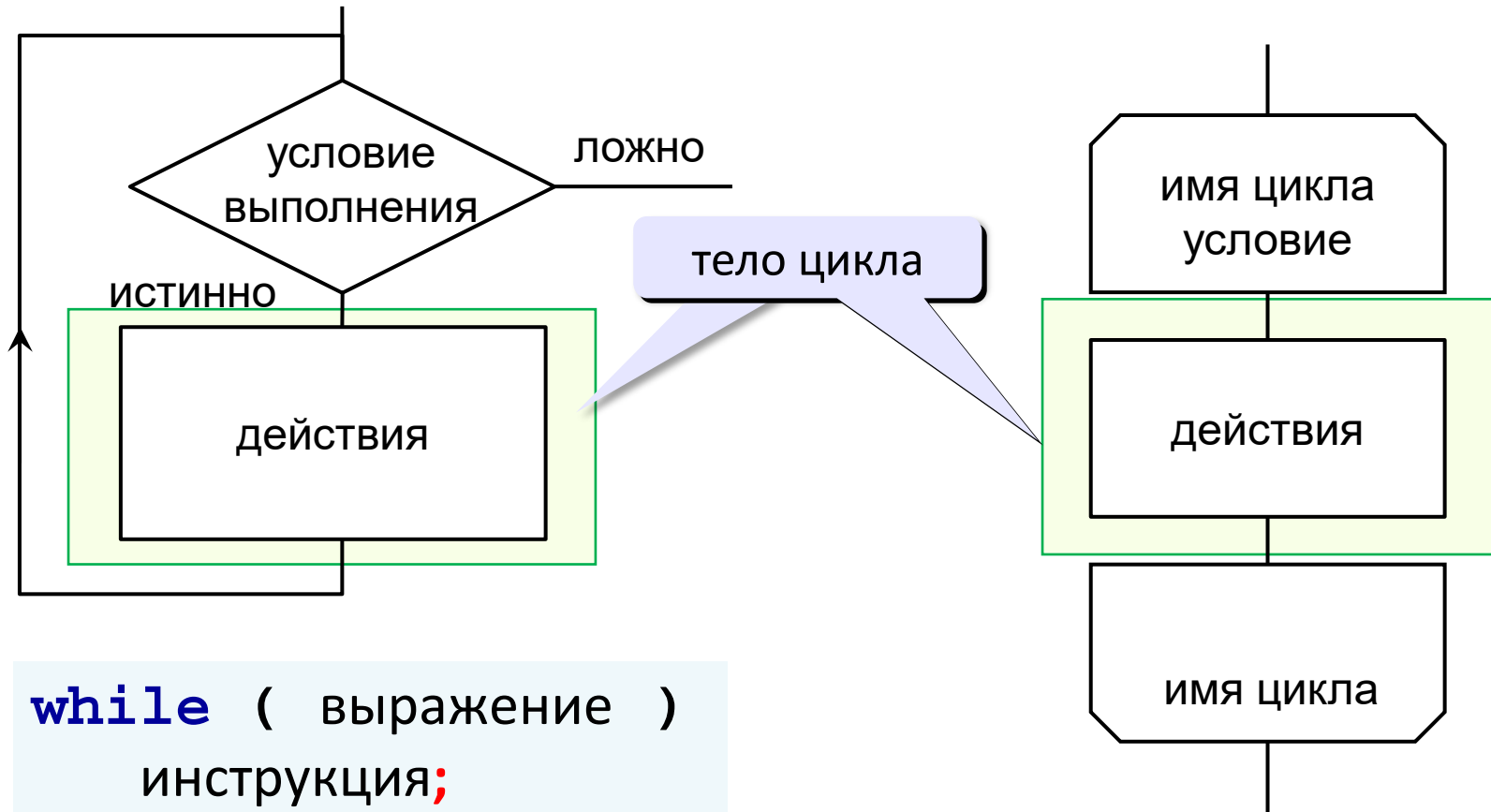
По расположению условия:

- цикл с **предусловием**
- цикл с **постусловием**
- цикл с выходом из середины тела цикла



В Си все циклы итерационные

Цикл с предусловием



```
while ( выражение )  
    инструкция;
```

```
for ( выражение ; выражение ; выражение )  
    инструкция;
```

Инструкция *while*

```
while ( выражение, определяющее условие )
```

```
{
```

```
    /* действия */
```

```
}
```

заголовок
цикла

тело цикла

Особенности:

- *выражение* может быть **любым**
- *выражение* пересчитывается каждый раз при входе в цикл
- если значение *выражения* на входе в цикл равно нулю, тело цикла не выполняется ни разу
- если тело цикла – одна инструкция, { } не нужны

Сколько раз выполняется цикл?

```
a = 4; b = 6;  
while ( a < b ) a ++;
```

2 раза
a = 6

```
a = 4; b = 6;  
while ( a < b ) a += b;
```

1 раз
a = 10

```
a = 4; b = 6;  
while ( a > b ) a ++;
```

0 раз
a = 4

```
a = 4; b = 6;  
while ( a < b ) b = a - b;
```

1 раз
b = -2

```
a = 4; b = 6;  
while ( a < b ) a --;
```

зависит от типа
переменной a

Арифметический цикл

Задача. Вычислить среднее арифметическое значение десяти чисел, введенных пользователем.

Решение «в лоб»: взять 10 различных переменных, считать в них значения, вычислить среднее.

```
#include <stdio.h>
int main()
{
    double a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, average;
    printf ("Введите 10 чисел\n");
    scanf ("%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf",
           &a1, &a2, &a3, &a4, &a5, &a6, &a7, &a8, &a9, &a10);
    average = (a1+a2+a3+a4+a5+a6+a7+a8+a9+a10) / 10;
    printf ("Среднее = %f\n", average);
    return 0;
}
```

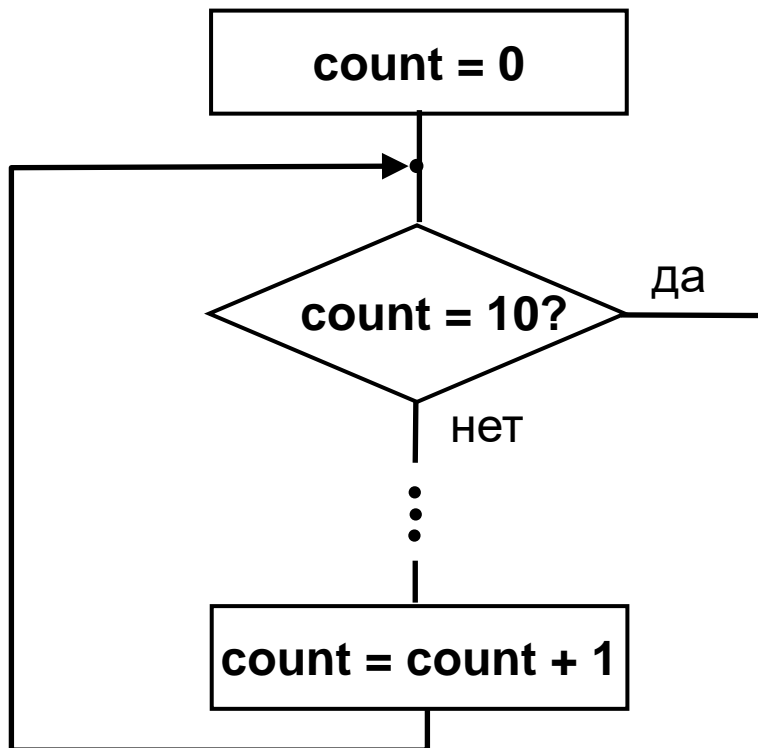


А если нужно обработать 1000 чисел
или количество чисел тоже вводится?

Арифметический цикл

Идея: считывать значения по одному и прибавлять к общей сумме.

Проблема: Как считать число повторов? Нужен счетчик.



Текст программы. Инструкция *while*

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    double number, sum = 0;
```

```
    int count = 0;
```

```
    printf ("Введите 10 чисел\n");
```

```
    while (count != 10)
```

```
    {
```

```
        scanf ("%lf", &number);
```

```
        sum += number;
```

```
        count++;
```

```
    }
```

```
    printf ("Среднее = %f\n", sum / 10);
```

```
    return 0;
```

```
}
```

Начальное значение суммы

Начальное значение
счетчика повторов

Условие выполнения цикла

Приращение счетчика

Текст программы. Инструкция *for*

```
#include <stdio.h>
int main()
{
    double number, sum = 0;
    int count;
    printf ("Введите 10 чисел\n");
    for ( count = 0 ; count != 10 ; count++ )
    {
        scanf ("%lf", &number);
        sum += number;
    }
    printf ("Среднее = %f\n", sum / 10);
    return 0;
}
```

Начальное значение
счетчика повторов

Приращение счетчика

Условие выполнения цикла

Инструкция *for*

Инициализация
(установка начальных значений)

Модификация (изменение значений переменных, влияющих на условие выполнения)

for (**выражение_1**; **выражение_2**; **выражение_3**)

{
/* действия */
}

тело цикла

заголовок
цикла

Условие выполнения
тела цикла

Инструкция *for*

Особенности:

- *for* – инструкция **цикла с предусловием**
- *выражения* могут быть **любыми**
- *выражение_1* вычисляется только один раз
- *выражение_2* пересчитывается каждый раз при входе в цикл
- если значение *выражения_2* на входе в цикл равно нулю, тело цикла не выполняется ни разу
- если тело цикла – одна инструкция, { } не нужны

Инструкция *for*

Особенности:

- любое из *выражений* может отсутствовать, **;** ставятся всегда

```
double number, sum = 0;  
int count = 0;  
printf ("Введите 10 чисел\n");  
for ( ; count != 10 ; )  
{  
    scanf ("%lf", &number);  
    sum += number;  
    count++;  
}
```

Начальное значение
счетчика повторов

Инициализация
не требуется

Модификация не
требуется

Счетчик изменяется в
теле цикла

```
for ( a = 5 ; ; a += 2 )  
    инструкция;
```

Отсутствие выражения == ИСТИНА
Цикл **бесконечный**

Цикл с неизвестным числом повторов

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

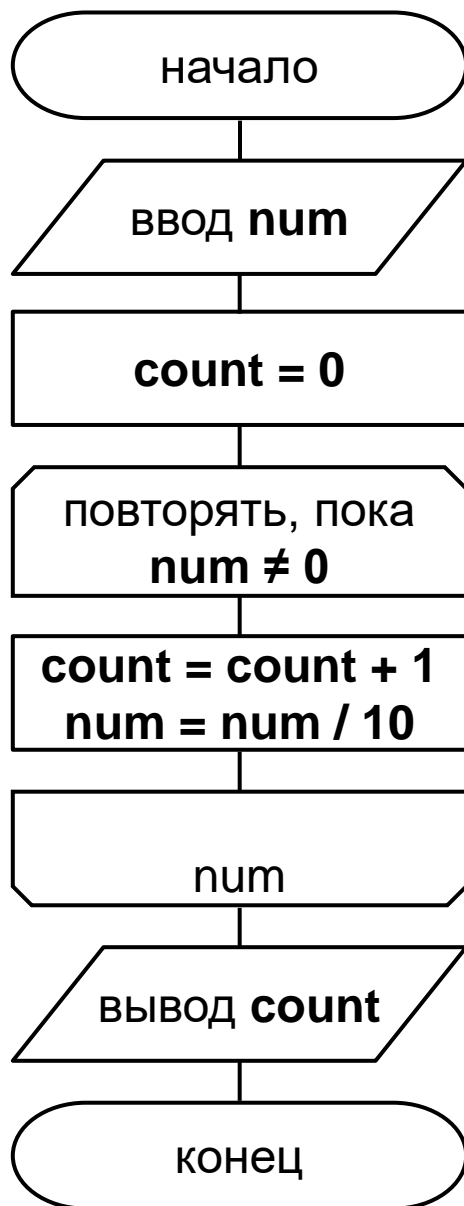
Идея решения: Отсекаем последовательно последнюю цифру, увеличиваем счетчик.

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 $n \neq 0$ ».

Схема и текст программы



```
#include <stdio.h>
int main()
{
    int num, count = 0;
    printf ("Введите целое число\n");
    scanf ("%d", &num);
    while ( num )
    {
        count++;
        num /= 10;
    }
    printf ("В числе %d цифр\n", count);
    return 0;
}
```

инструкция **while**

тот же цикл,
инструкция **for**

```
for (; num ; num /= 10 )
    count++;
```

Прерывание цикла и сложные условия

Задача: Ввести натуральное число и определить, является ли оно простым (простые числа не имеют других делителей, кроме самих себя и 1).

Проблема: Как понять, что число ни на что не делится?

Решение: Попробовать поделить число на все возможные делители от 2 до корня из этого числа. Если найдется делитель, на который данное число поделится без остатка, значит, число составное, цикл можно прервать.

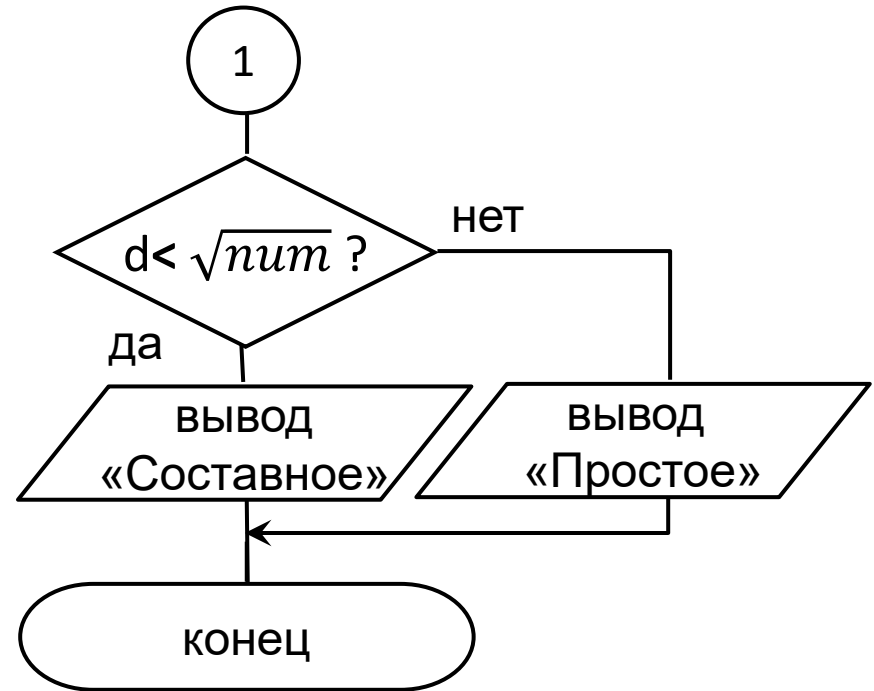
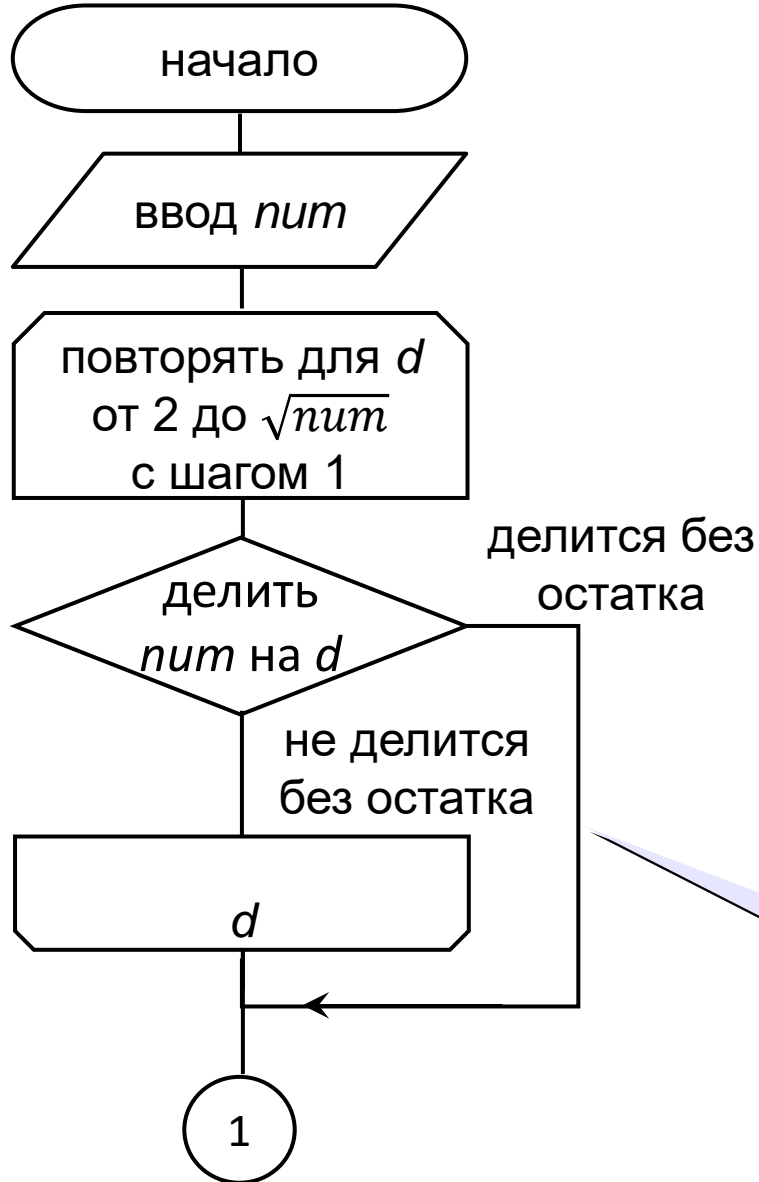


Почему именно до корня?



Можно ли выполнить сравнение, не извлекая корень?

Прерывание цикла и сложные условия



прерывание цикла (break)

Прерывание цикла и сложные условия

```
#include <stdio.h>
int main()
{
    int num, d;
    printf ("Введите натуральное число\n");
    scanf ("%d", &num);
    for ( d = 2 ; d * d <= num ; d++ )
        if ( num % d == 0 ) break;
    if ( d * d <= num )
        printf ("Это составное число\n");
    else
        printf ("Это простое число\n");
    return 0;
}
```

условие
выполнения

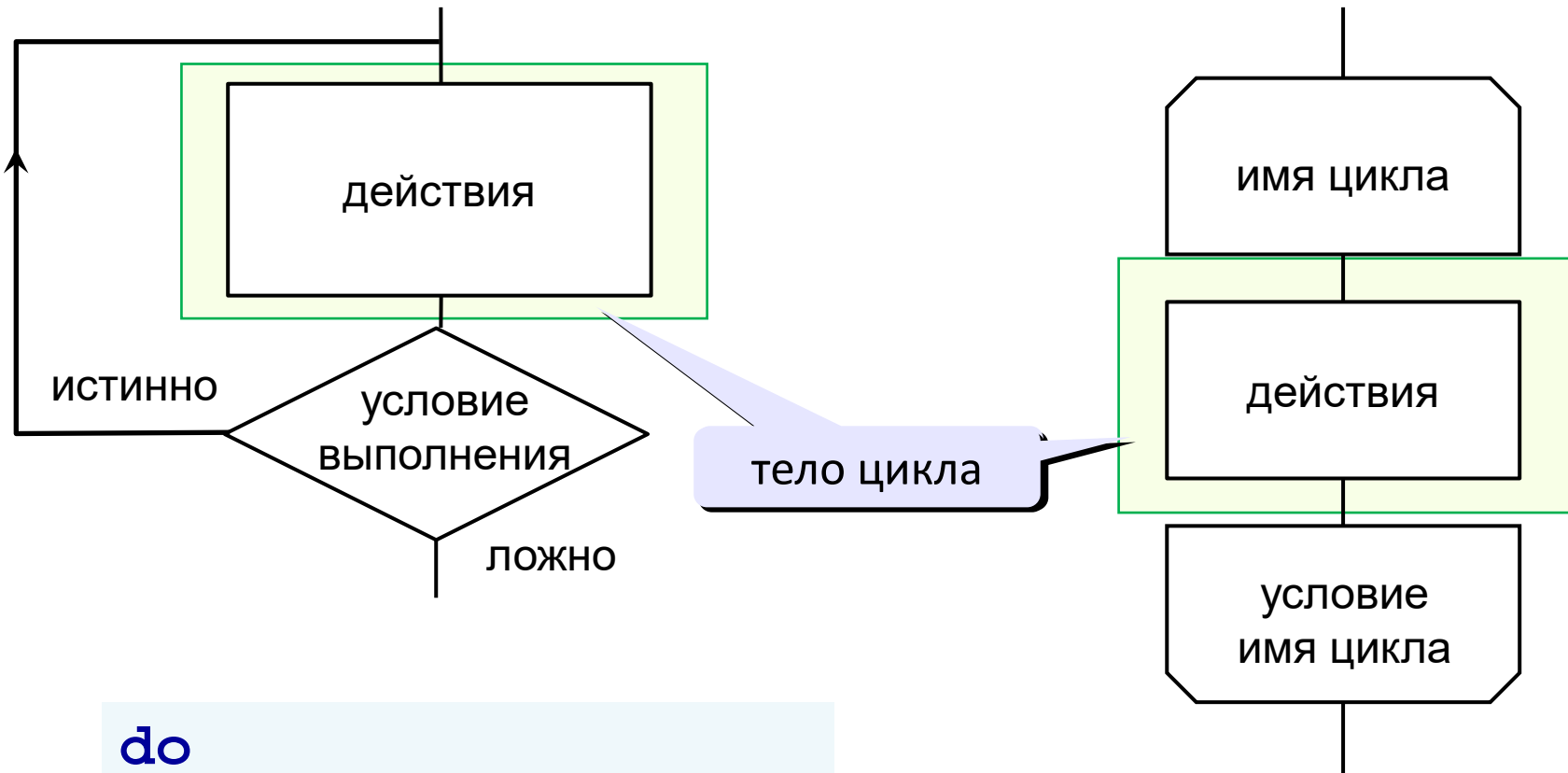
прерывание
цикла

условие
выхода

тело цикла –
пустой оператор

```
for ( d = 2 ; d * d <= num && num % d ; d++ );
```

Цикл с постусловием



do

инструкция;


while (выражение);



Хотя бы один раз тело цикла обязательно выполнится

Инструкция *do ... while*

```
do
{
    /* действия */
}
while ( выражение, определяющее условие );
```



The diagram shows a light purple rounded rectangle labeled "тело цикла" (loop body) with a line pointing to the code block between the curly braces of the `do` statement.

Особенности:

- *выражение* может быть **любым**
- *выражение* определяет условие возврата к началу цикла (условие повторения цикла)
- *выражение* пересчитывается каждый раз при выходе из цикла
- тело цикла выполняется хотя бы один раз
- если тело цикла – одна инструкция, { } не нужны

Цикл с постусловием

Чаще всего цикл с постусловием применяется для **проверки** корректности **ввода**

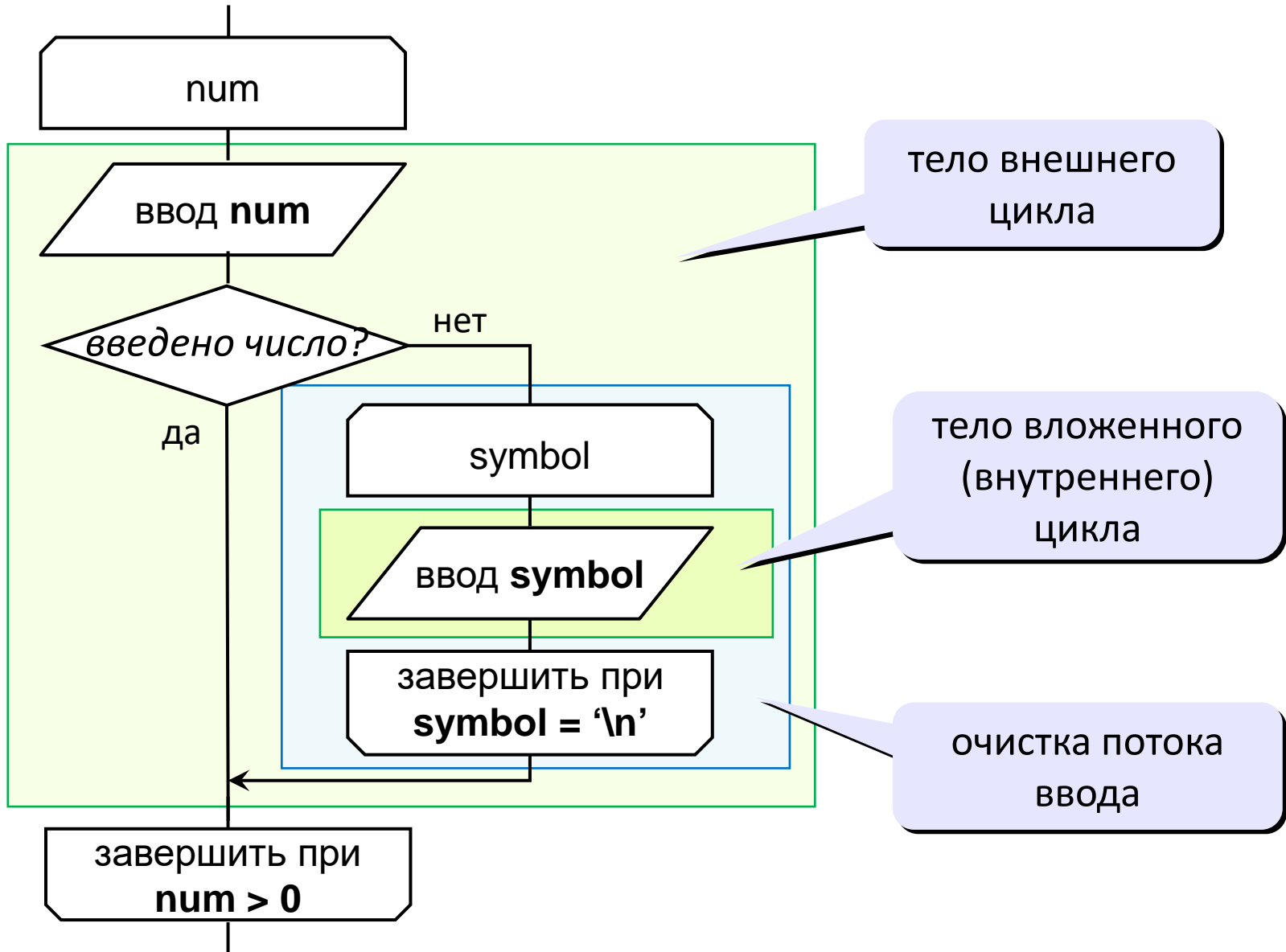
Задача: Ввести целое **положительное** число (< 20000000) и определить число цифр в нем.

Проблема: Как не дать ввести отрицательное число или ноль?

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

```
int num, count = 0;
printf ("Введите целое положительное число\n");
do
    scanf ("%d", &num);
while ( num <= 0 );
...
```

Контроль ввода. Цикл в цикле



Контроль ввода. Цикл в цикле

```
int num = 0;  
do  
{
```

внешний
цикл

начальное значение на случай,
если с первой попытки будет
введено не число

```
printf ("Введите натуральное число\n");  
if ( !scanf ("%d", &num) )  
{
```

очистка
потока
ввода

```
char c;  
do
```

вложенный
цикл

```
scanf ("%c", &c);  
while ( c != '\n' );
```

```
}  
while ( num <= 0 );
```