



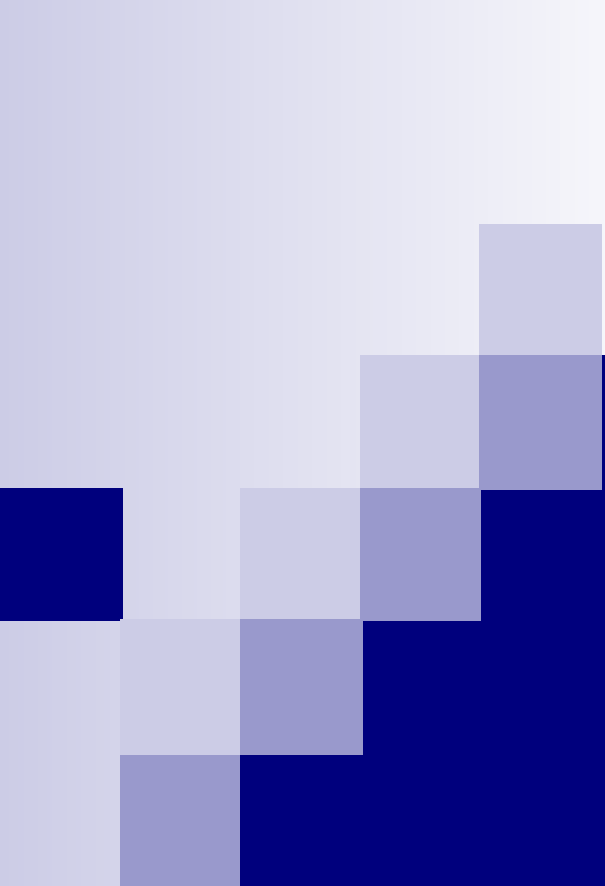
ЦИКЛЫ

Алтайский государственный университет
Факультет математики и ИТ
Кафедра информатики
Барнаул 2015

Лекция 6

■ План

- Пара заданий для самопроверки
- Операторы цикла
 - Оператор с переменной (for)
 - Оператор с пред-условием
 - Последовательности
 - Оператор с пост-условием
 - Прерывание циклов



Пара заданий для самопроверки

- Задание для самопроверки 1
- Задание для самопроверки 2

Задание для самопроверки 1

- Что будет выведено если пользователь введет 13?

```
#include <stdio.h>

void main() {
    int number;
    printf("введи число: ");
    scanf("%d", &number);
    if (number = 0)
        printf("равно 0\n");
    else
        printf("не равно 0\n");
}
```

Частая ошибка:
= ВМЕСТО ==

Задание для самопроверки 2

- Что будет выведено если пользователь введет 2 и 3?

```
#include <stdio.h>

void main() {
    int a, b, max=0;
    printf("введи a и b: ");
    scanf("%d%d", &a, &b);
    if (a > b)
        if (a > 0) max = a;
    else
        max = b;
    printf("max=%d\n", max);
}
```

Частая ошибка!
else – всегда
альтернатива к
ближайшему if

Цикл с переменной

- Цикл с переменной (`for`)
- Цикл с пред-условием (`while`)
- Последовательности
- Цикл с пост-условием (`do...while`)
- Прерывание цикла

Циклы

Цикл – это многократное выполнение одинаковой последовательности действий.

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

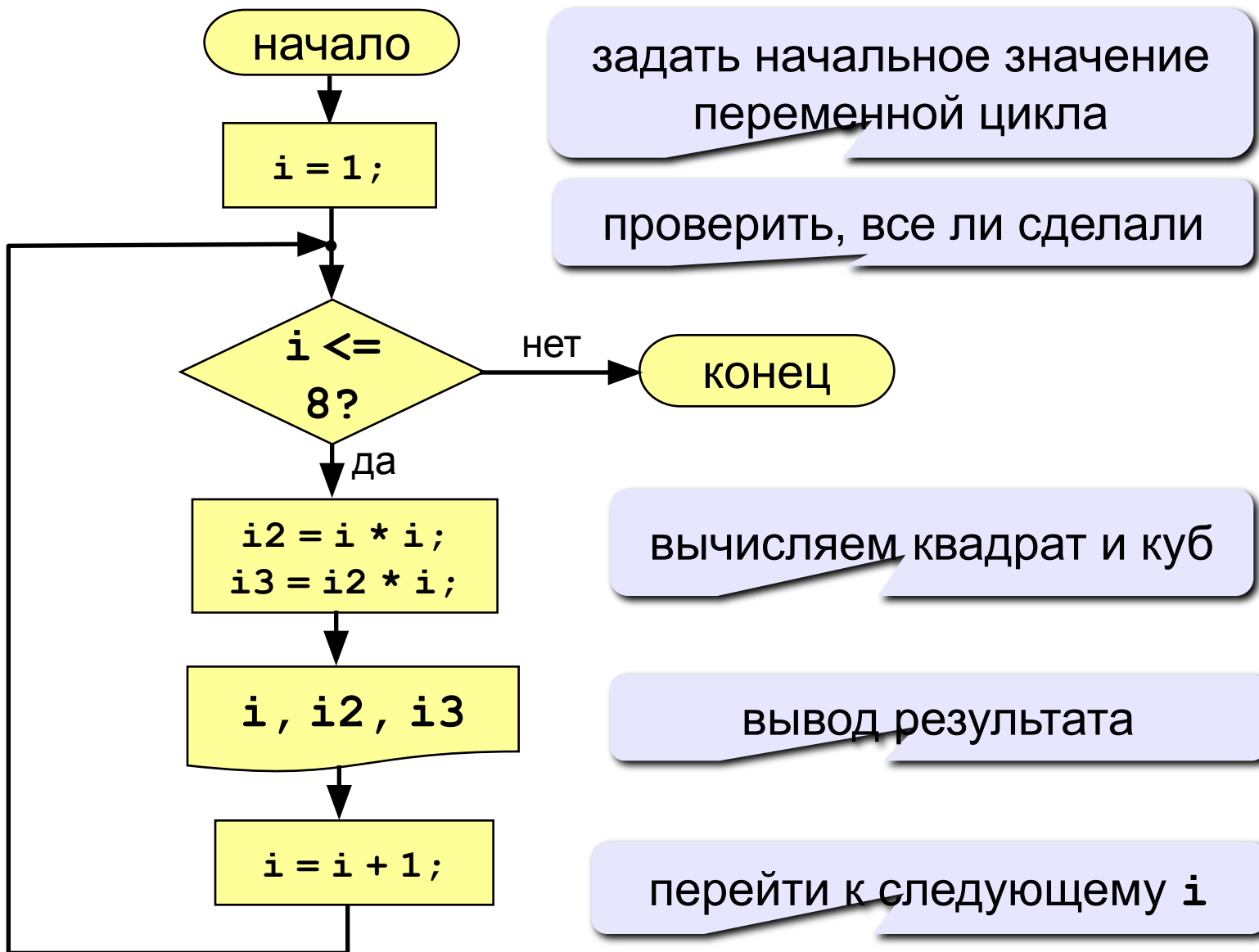
Задача. Вывести на экран квадраты и кубы целых чисел от 1 до 8 (от **a** до **b**).

Особенность: одинаковые действия выполняются 8 раз.

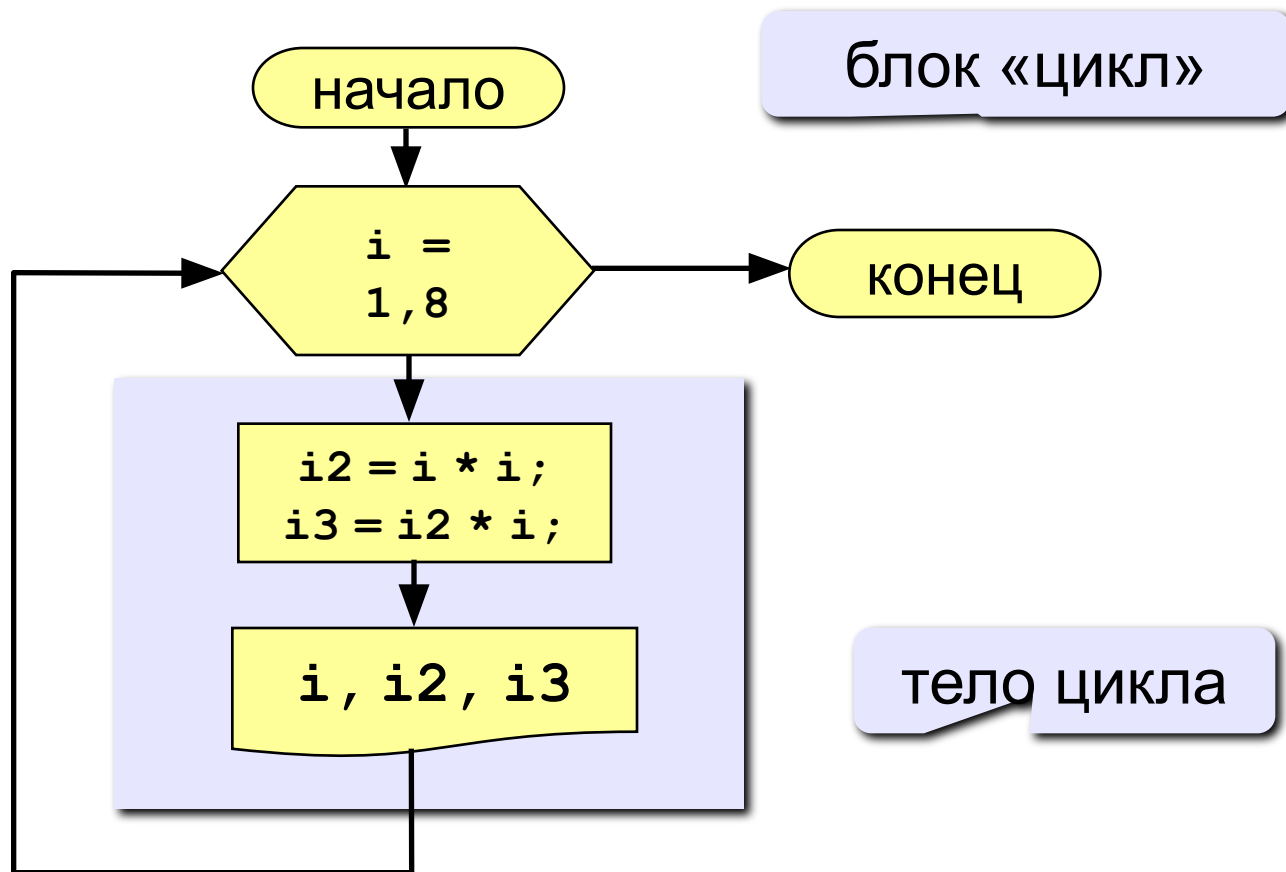


Можно ли решить известными методами?

Алгоритм



Алгоритм (с блоком «цикл»)



Программа

```
void main()
```

```
{
```

```
    int i, i2, i3
```

начальное
значение

заголовок
цикла

конечное
значение

ЦИКЛ

переменная цикла

```
    for (i=1; i<=8; i++)
```

изменение после
каждого шага:

i=i+1

начало цикла

```
{
```

```
    i2 = i*i;
```

```
    i3 = i2*i;
```

```
    printf("%4d %4d %4d\n", i, i2, i3);
```

тело цикла

```
}
```

конец цикла

ровные
столбики

```
}
```

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; i >= 1; i -- )  
{  
    i2 = i*i;  
    i3 = i2*i;  
    printf("%4d %4d %4d\n", i, i2, i3);  
}
```

Цикл с переменной

```
for (начальные значения;  
     условие продолжения цикла;  
     изменение на каждом шаге)  
{  
    /* тело цикла */  
}
```

Примеры:

```
for (a = 2; a < b; a += 2) { ... }
```

```
for (a = 2, b = 4; a < b; a += 2) { ... }
```

```
for (a = 1; c < d; x++) { ... }
```

```
for (; c < d; x++) { ... }
```

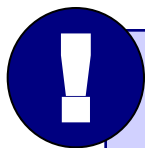
```
for (; c < d; ) { ... }
```

Цикл с переменной

Особенности:

- **условие** проверяется в начале очередного шага цикла, если оно ложно, цикл не выполняется;
- **изменения** (третья часть в заголовке) выполняются в конце очередного шага цикла;
- если **условие** никогда не станет ложным, цикл может продолжаться бесконечно (**зацикливание**)

```
for (i=1; i<8; i++) { i--; }
```



Не рекомендуется менять переменную цикла в теле цикла!

- если в теле цикла один оператор, **скобки** `}` можно не ставить:

```
for (i=1; i<8; i++) a+=b;
```

Цикл с переменной

Особенности:

- после выполнения цикла **во многих системах** устанавливается первое значение переменной цикла, при котором нарушено условие:

```
for (i=1; i<=8; i++)  
    printf("Привет");  
printf("i=%d", i);
```

i=9

```
for (i=8; i>=1; i--)  
    printf("Привет");  
printf("i=%d", i);
```

i=0

Сколько раз выполняется цикл?

```
a = 1;  
for (i=1; i<4; i++) a++;
```

3 раза
a = 4

```
a = 1;  
for (i=1; i<4; i++) a = a+i;
```

3 раза
a = 7

```
a = 1; b = 2;  
for (i=3; i >= 1; i--) a += b;
```

3 раза
a = 7

```
a = 1;  
for (i=1; i >= 3; i--) a = a+1;
```

0 раз
a = 1

```
a = 1;  
for (i=1; i <= 4; i--) a++;
```

зацикливание

Цикл с предусловием

- Цикл `while`
- Взаимозаменяемость циклов `for` и `while`

Цикл с неизвестным числом шагов

Пример: Отпилить полено от бревна. Сколько раз надо сделать движения пилой?

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

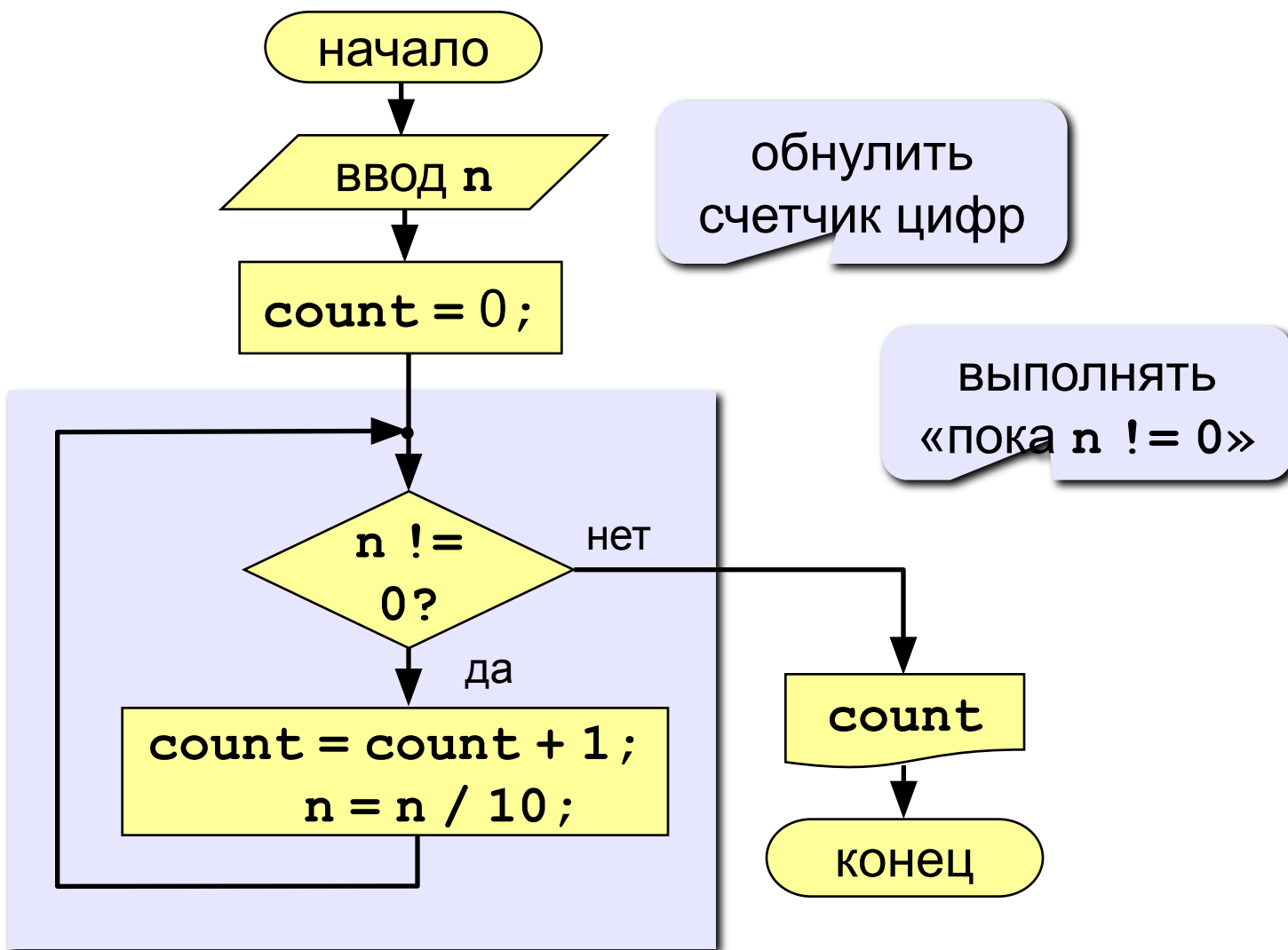
Идея решения: Отсекаем последовательно последнюю цифру, увеличиваем счетчик.

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 $n \neq 0$ ».

Алгоритм



Программа

```
void main()  
{  
    int n, count, n1;  
    printf("Введите целое число\n");  
    scanf("%d", &n);  
    count = 0; n1 = n;  
    while (n != 0)  
    {  
        count ++;  
        n = n / 10;  
    }  
    printf("В числе %d нашли %d цифр", n1, count);  
}
```

ВЫПОЛНЯТЬ
«ПОКА n != 0»



Что плохо?

Цикл с условием

```
while ( условие )  
{  
    /* тело цикла */  
}
```

Особенности:

- МОЖНО ИСПОЛЬЗОВАТЬ СЛОЖНЫЕ УСЛОВИЯ:

```
while ( a < b && b < c ) { ... }
```

- если в теле цикла только один оператор, скобки {}
МОЖНО НЕ ПИСАТЬ:

```
while ( a < b ) a ++;
```

Цикл с условием

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 ) a = a - b;
```

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

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

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 --;
```

зацикливание

Замена for на while и наоборот

```
for ( i=1; i<=10; i++)  
{  
    /* тело цикла */  
}
```

```
i = 1;  
while ( i <= 10 ) {  
    /* тело цикла */  
    i ++;  
}
```

```
for ( i=a; i>=b; i-- )  
{  
    /* тело цикла */  
}
```

```
i = a;  
while ( i >= b ) {  
    /* тело цикла */  
    i --;  
}
```



В языке Си замена цикла `for` на `while` и наоборот возможна **всегда!**



Последовательности

- Способы определения
- Типичные алгоритмы
- Пример программы

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, ...

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

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

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

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

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

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

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

Последовательности

Задача: 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:

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

Элемент последовательности (начиная с №2):

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

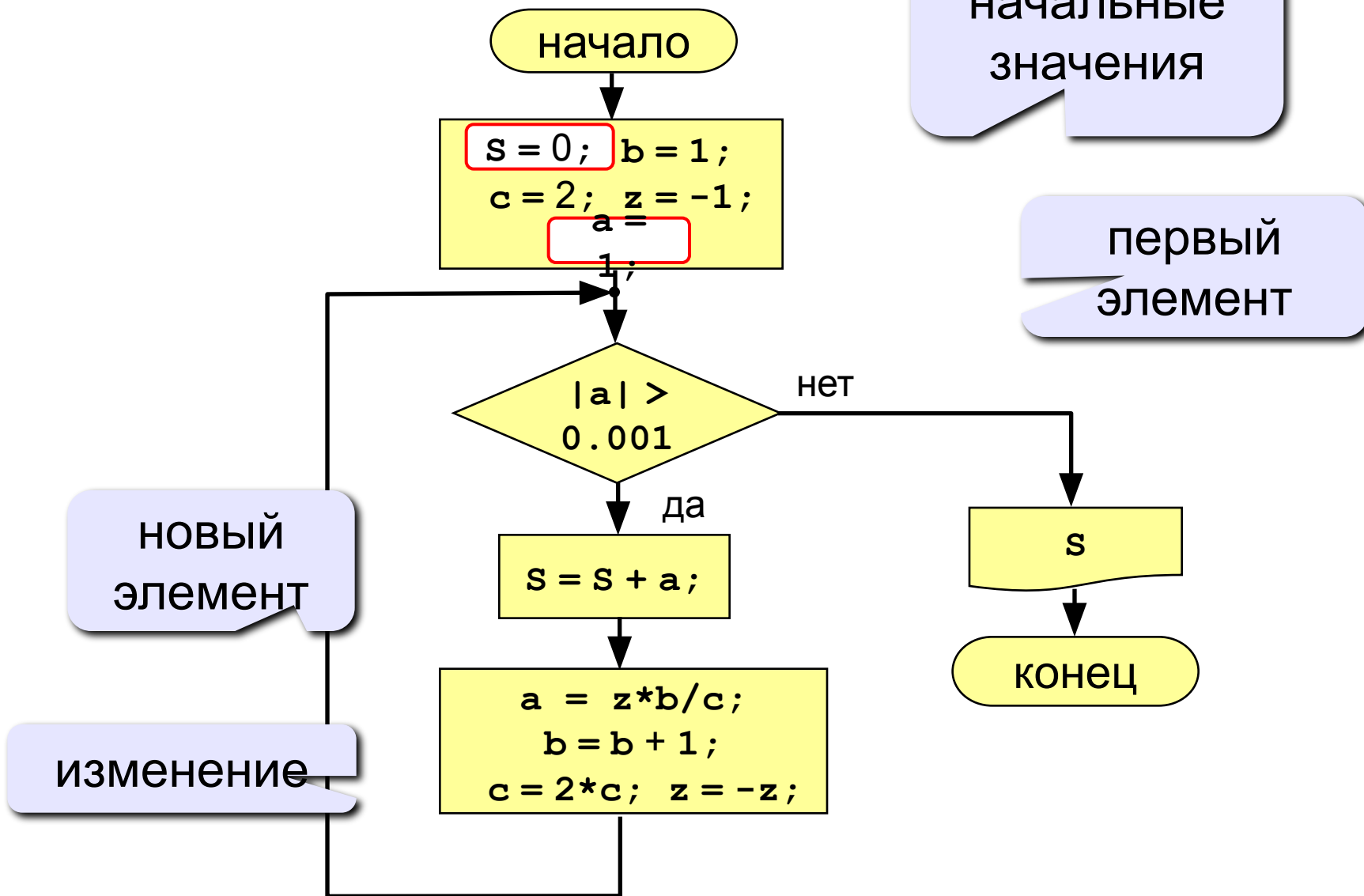
n	1	2	3	4	5	...
b	1	2	3	4	5	...
c	2	4	8	16	32	...
z	-1	1	-1	1	-1	...

$$b = b + 1;$$

$$c = 2 * c;$$

$$z = -z;$$

Алгоритм



Программа

```

#include <math.h>
void main()
{
    int x, c, z;
    float S, a, b;
    S = 0; z = -1;
    b = 1; c = 2; a = 1;
    while (fabs(a) > 0.001) {
        S += a;
        a = z * b / c;
        z = -z;
        b++;
        c *= 2;
    }
    printf ("S = %10.3f", S);
}

```

математические функции

чтобы не было
округления при
делении

модуль
абсолютного
числа

знач

увеличение
суммы

расчет элемента
последовательности



Что плохо?

Упражнения

1. 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:

$$S = 1.157$$

2. Найти сумму элементов последовательности с точностью 0,001:

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

- Цикл с пост-условием (`do...while`)
- Общая схема
- Пример

Цикл с постусловием

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

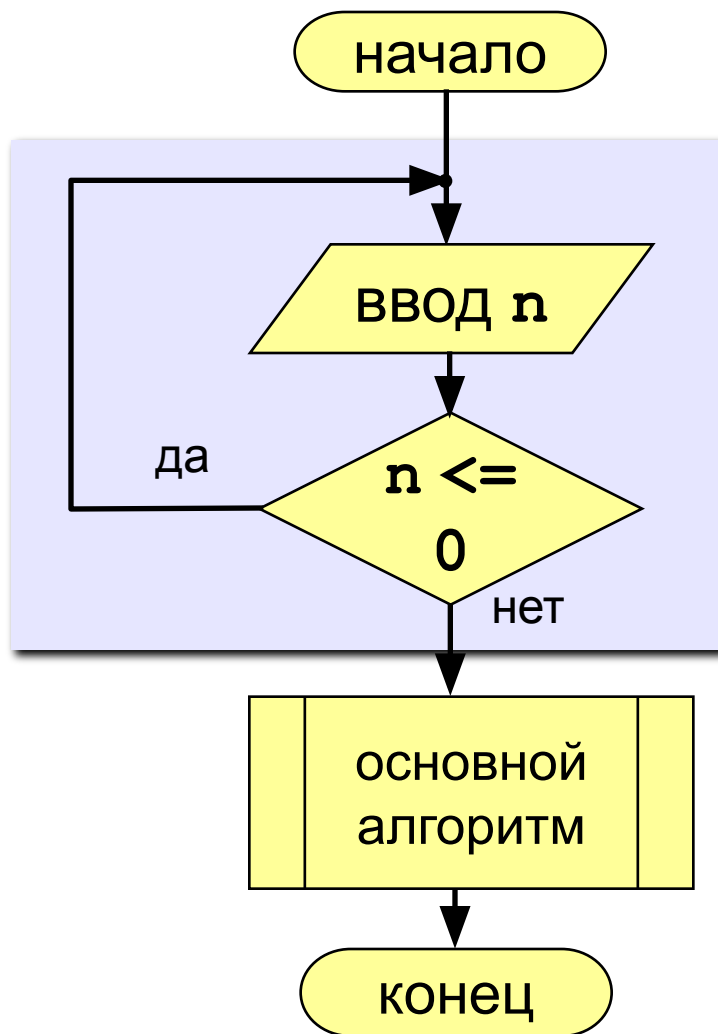
Проблема: Как не дать ввести отрицательное число или ноль?

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

Особенность: Один раз тело цикла надо сделать в любом случае \Rightarrow 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



тело цикла

условие

блок «ТИПОВОЙ
процесс»

Программа

```
void main()
{
    int n;
    do {
        printf("Введите положительное число\n");
        scanf("%d", &n);
    }
    while ( n <= 0 );
    ... /* основной алгоритм */
}
```

условие

Особенности:

- тело цикла всегда выполняется хотя бы один раз
- после слова **while** («пока...») ставится условие продолжения цикла

Сколько раз выполняется цикл?

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

3 раза

~~a = 7~~

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

1 раз

~~a = 10~~

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

зацикливание

```
a = 4; b = 6;  
do b = a - b; while (a >= b);
```

2 раза

~~b = 6~~

```
a = 4; b = 6;  
do a += 2; while (a >= b);
```

зацикливание

Прерывание цикла

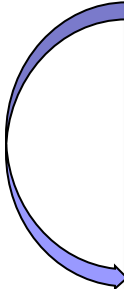
- Прерывание цикла (**break**)
- Прерывание шага цикла (**continue**)
- Прерывание вложенных циклов

Прерывание цикла

■ Оператор **break**

- **break** досрочно завершает (прерывает) цикл и передает управление на оператор, следующий за циклом

- Работает для любого цикла
 - **for**
 - **while**
 - **do ... while**



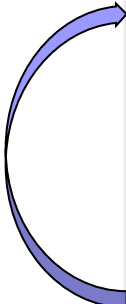
```
...  
while (<условие>) {  
    <оператор 1>;  
    <оператор 2>;  
  
    ...  
    if(...) break;  
    ...  
    <оператор N>;  
}  
<оператор>;  
...
```

Прерывание шага цикла

■ Оператор `continue`

- **`continue`** досрочно завершает шаг цикла и начинает следующий

- Работает для любого цикла
 - `for`
 - `while`
 - `do ... while`



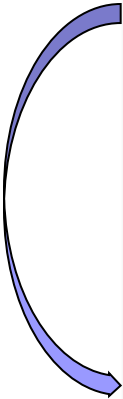
```
...  
while (<условие>) {  
    <оператор 1>;  
    <оператор 2>;  
    ...  
    if(...) continue;  
    ...  
    <оператор N>;  
}  
...
```

Прерывание вложенных циклов

■ Оператор `goto`

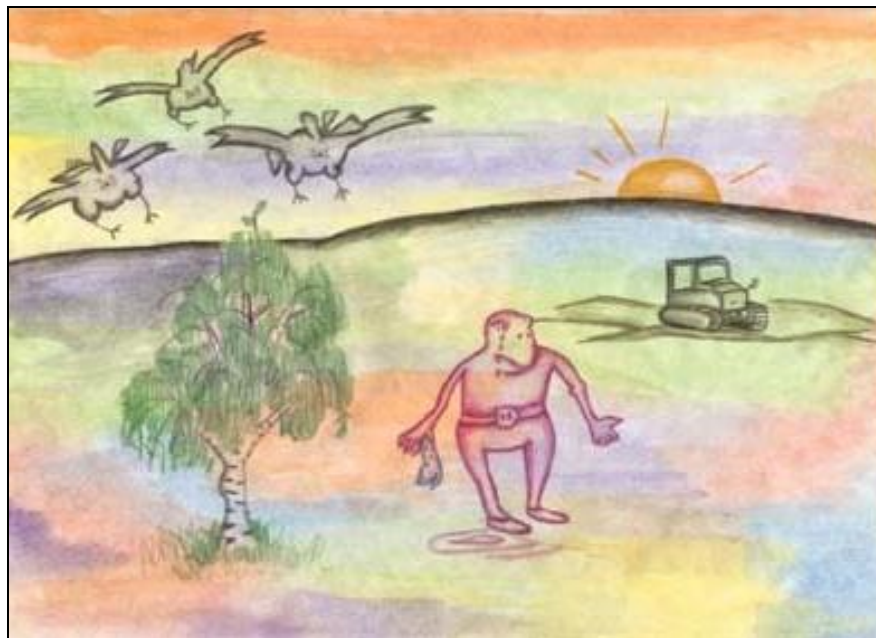
- `goto` осуществляет безусловный переход в точку программы, помеченную меткой
- Не рекомендуется использовать без настоятельной необходимости
- Нарушает принципы структурного программирования

```
...
for () {
    while (<условие>) {
        <оператор 1>;
        <оператор 2>;
        ...
        if (...) goto error;
        ...
        <оператор N>;
    }
}
error: <оператор>;
...
```



Вопросы?

- Операторы цикла
 - Цикл с переменной (`for`)
 - Цикл с пред-условием (`while`)
 - Последовательности
 - Цикл с пост-условием (`do...while`)
 - Прерывание цикла



Дубовая роща. Грачи улетели