

# Программирование на языке Си

1. [Введение](#)
2. [Переменные](#)
3. [Ветвления](#)
4. [Сложные условия](#)
5. [Циклы](#)
6. [Циклы с переменной](#)
7. [Оператор выбора](#)
8. [Отладка программ](#)
9. [Графика](#)
10. [Графики функций](#)
11. [Процедуры](#)
12. [Анимация](#)
13. [Функции](#)
14. [Случайные числа](#)

# Программирование на языке Си

## Тема 1. Введение

# Алгоритм

---

**Алгоритм** – это четко определенный план действий для исполнителя.

## Свойства алгоритма

- **дискретность**: состоит из отдельных шагов (команд)
- **понятность**: должен включать только команды, известные исполнителю (входящие в СКИ)
- **определенность**: при одинаковых исходных данных всегда выдает один и тот же результат
- **конечность**: заканчивается за конечное число шагов
- **массовость**: может применяться многократно при различных исходных данных
- **корректность**: дает верное решение при любых допустимых исходных данных

# Программа

---

**Программа** – это

- алгоритм, записанный на каком-либо языке программирования
- набор команд для компьютера

**Команда** – это описание действий, которые должен выполнить компьютер.

- откуда взять исходные данные?
- что нужно с ними сделать?
- куда поместить результат?

# Языки программирования

---

- **Машинно-ориентированные (низкого уровня)** - каждая команда соответствует одной команде процессора (ассемблер)
- **Языки высокого уровня** – приближены к естественному (английскому) языку, легче воспринимаются человеком, **не зависят от конкретного компьютера**
  - *для обучения*: Бейсик, ЛОГО, Паскаль
  - *профессиональные*: Си, Фортран, Паскаль
  - *для задач искусственного интеллекта*: Пролог, ЛИСП
  - *для Интернета*: JavaScript, Java, Perl, PHP, ASP

# Язык Си

---

**1972-1974** – Б. Керниган, Д. Ритчи

- ⊕ • высокая скорость работы программ
- много возможностей
- стал основой многих современных языков (*C++*, *C#*, *Javascript*, *Java*, *ActionScript*, *PHP*)
- ⊖ • много шансов сделать ошибку, которая не обнаруживается автоматически

# Простейшая программа

главная (основная) программа  
всегда имеет имя *main*

```
main()
```

начало  
программы

«тело»  
программы  
(основная  
часть)

```
{
```

```
}
```

конец  
программы



Что делает эта программа?

# Что происходит дальше?

текст программы на Си или Си++

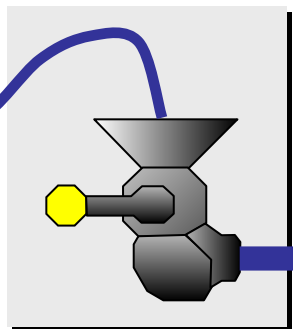
`first.cpp`

транслятор

`first.o`

```
main()
{
}
```

исходный файл



```
ЪBzЦ2?|ё3БKa
n/36ШпlC+И-
Ц3_5MyPЧ6
s6bd^:/@:лЖ1_
```

объектный файл

стандартные  
функции

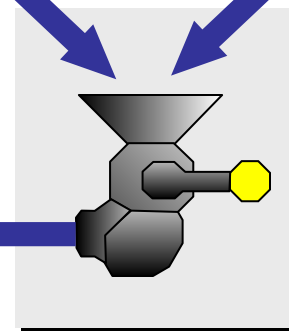


- по исходному файлу можно восстановить остальные
- исполняемый файл можно запустить

`first.exe`

```
MZPo:€Pэ_e3"!_
`кп,Цb€-Цр1
G_БАС,
_Ощяхя9жФ
```

исполняемый файл



редактор  
связей  
(компоновка)



# Вывод текста на экран

*include* = ВКЛЮЧИТЬ

```
#include <stdio.h>
main ()
{
printf ("Привет! " ) ;
}
```

файл *stdio.h*:  
описание  
стандартных  
функций ввода  
и вывода

ВЫЗОВ СТАНДАРТНОЙ  
функции  
*printf* = *print format*  
(форматный вывод)

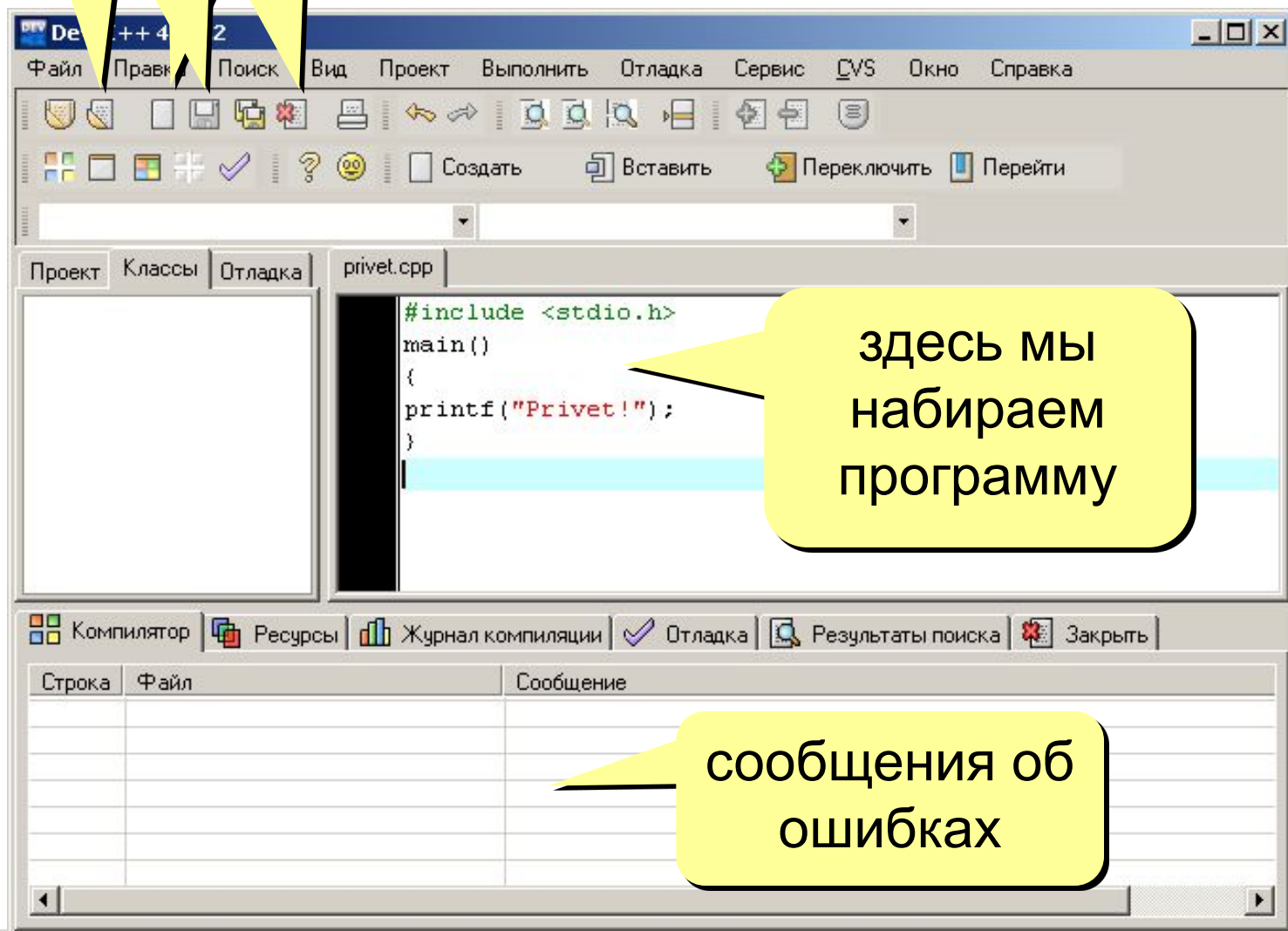
ЭТОТ ТЕКСТ  
будет на  
экране

# Как начать работу?



Dev-C++

Открыть | Сохранить | Закрывать



## Оболочка Dev C ++ 4.9

---



**IDE** = *Integrated Development Environment*

интегрированная среда разработки:

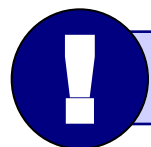
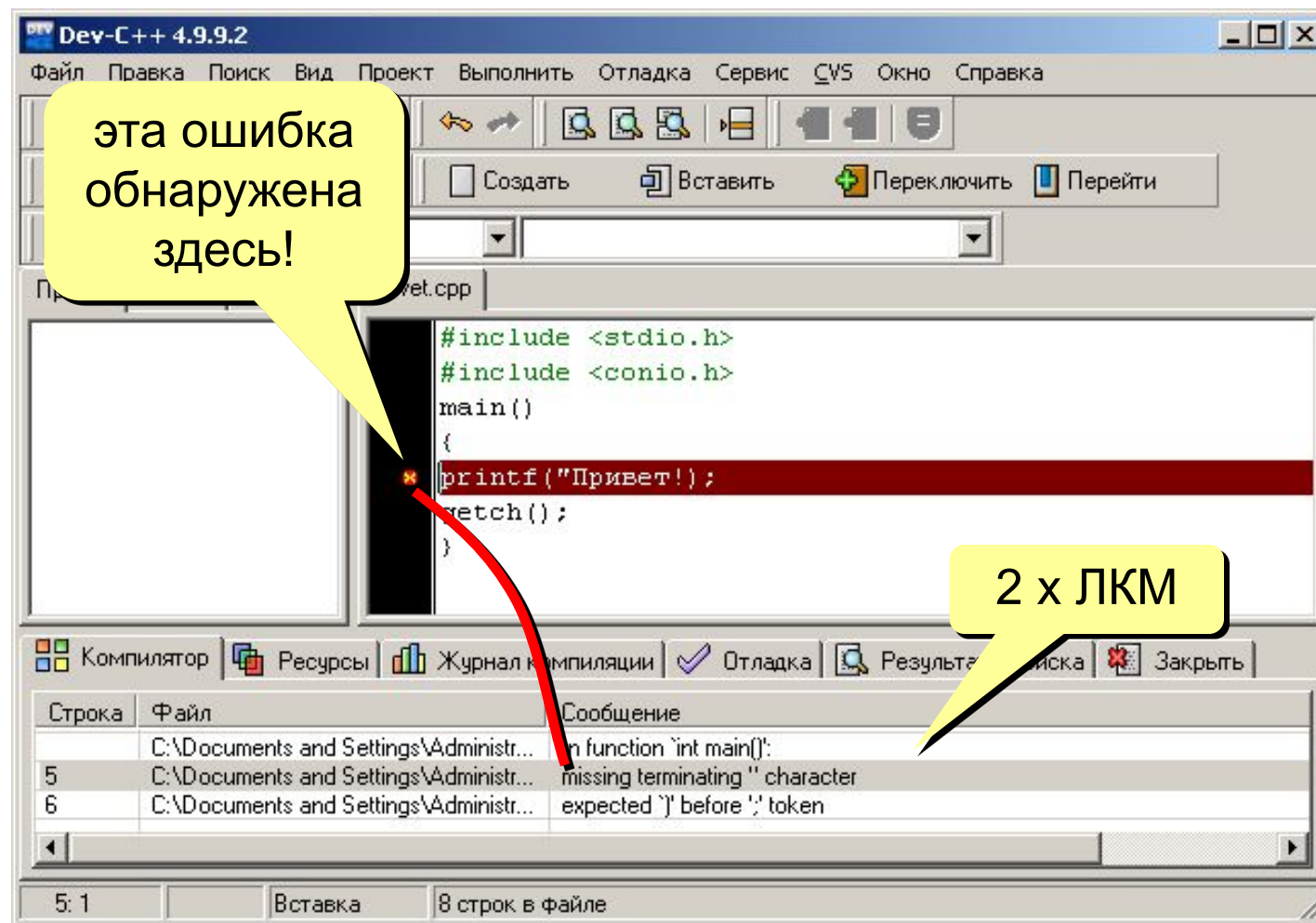
- **текстовый редактор** для создания и редактирования текстов программ
- **транслятор** для перевода текстов программ на Си и Си++ в команды процессора
- **КОМПОНОВЩИК** для создания исполняемого файла (EXE-файла), подключаются стандартные функции
- **отладчик** для поиска ошибок в программах

# Управление клавишами

---

Новый файл (Создать)	<b>Ctrl+N</b>	
Открыть файл	<b>Ctrl+O</b>	
Сохранить файл	<b>Ctrl+S</b>	
Закреть окно с программой	<b>Ctrl-F4</b>	
Запуск программы	<b>F9</b>	
Отменить	<b>Ctrl-Z</b>	
Восстановить отмененное	<b>Shift-Ctrl-Z</b>	

# Где ошибки?



**Ошибка может быть в конце предыдущей строки!**

# Наиболее «популярные» ошибки

---

**xxx.h: No such file or directory**

не найден заголовочный файл 'xxx.h' (неверно указано его имя, он удален или т.п.)

**'xxx' undeclared (first use this function)**

функция или переменная 'xxx' неизвестна

**missing terminating " character**

не закрыты кавычки "

**expected ;**

нет точки с запятой в конце оператора **в предыдущей строке**

**expected }**

не закрыта фигурная скобка

# Ждем нажатия любой клавиши

```
#include <stdio.h>
#include <conio.h>
main ()
{
printf("Привет!"); // вывод на экран
getch(); /* ждать нажатия клавиши */
}
```

файл **conio.h**: описание функций для работы с клавиатурой и монитором

комментарий до конца строки

ждать нажатия на любую клавишу

комментарий между /\* и \*/

# Переход на новую строку

---

```
#include <stdio.h>
#include <conio.h>
main ()
{
printf ("Привет, \n Вася! " );
getch ();
}
```

последовательность  
`\n` (код 10)  
переход на новую строку

на экране:

```
Привет ,
Вася!
```



# Задания

---

**«4»:** Вывести на экран текст "лесенкой"

Вася

пошел

гулять

**«5»:** Вывести на экран рисунок из букв

```
Ж
ЖЖЖ
ЖЖЖЖЖ
ЖЖЖЖЖЖЖ
НН НН
ZZZZZ
```

# Программирование на языке Си

## Тема 2. Переменные

# Что такое переменная?

---

**Переменная** – это ячейка в памяти компьютера, которая имеет имя и хранит некоторое значение.

- Значение переменной может меняться во время выполнения программы.
- При записи в ячейку нового значения старое стирается.

## Типы переменных

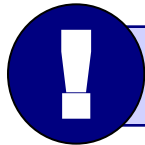
- **int** – целое число (4 байта)
- **float** – вещественное число, *floating point* (4 байта)
- **char** – символ, *character* (1 байт)

# Имена переменных

---

## Могут включать

- латинские буквы (A-Z, a-z)
- знак подчеркивания \_
- цифры 0-9



**Имя не может начинаться с цифры!**

## НЕ могут включать

- русские буквы
- пробелы
- скобки, знаки +, =, !, ? и др.

## Какие имена правильные?

**AXby R&B 4Wheel Вася "PesBarbos"**  
**TU154 [QuQu] \_ABBA A+B**

# Объявление переменных

**Объявить переменную** = определить ее имя, тип, начальное значение, и выделить ей место в памяти.

```
main ()
{
  i
  f
  int Tu104, I186=23, Yak42;
  float x=4.56, y, z;
  char c, c2='A', m;
}
```

целая переменная a

целая и дробная части отделяются точкой

вещественные переменные

целые переменные Tu104, I186 и Yak42

вещественные переменные x, y и z  
x = 4,56

символьные переменные c, c2 и m  
c2 = 'A'



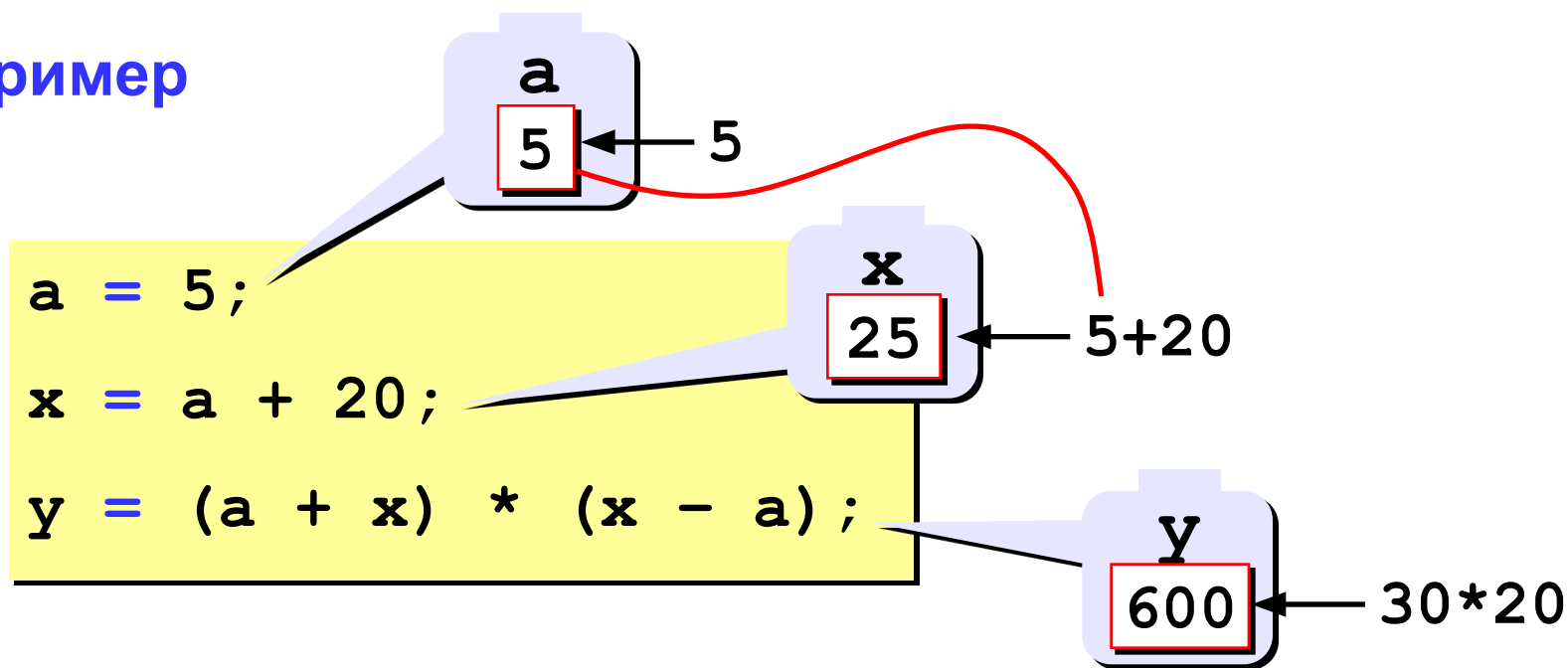
**Если начальное значение не задано, в этой ячейке находится «мусор»!**

# Оператор присваивания

**Оператор** – это команда языка программирования высокого уровня.

**Оператор присваивания** служит для изменения значения переменной.

## Пример



# Оператор присваивания

Общая структура:

куда записать

что

```
имя переменной = выражение;
```

Арифметическое выражение может включать

- константы (постоянные)
- имена переменных
- знаки арифметических операций:

+ - \* / %

умножение

деление

остаток от  
деления

- вызовы функций
- круглые скобки ( )



Для чего служат  
круглые скобки?

# Сложение двух чисел

**Задача.** Ввести два целых числа и вывести на экран их сумму.

**Простейшее решение:**

```
#include <stdio.h>
#include <conio.h>
main ()
{
    int a, b, c;
    printf("Введите два целых числа: ");
    scanf ("%d%d", &a, &b);
    c = a + b;
    printf ("%d", c);
    getch ();
}
```

подсказка для  
ввода

ВВОД ДВУХ  
ЧИСЕЛ С  
КЛАВИАТУРЫ

ВЫВОД РЕЗУЛЬТАТА



# Ввод чисел с клавиатуры

**scanf** –  
форматный ввод

формат ввода

адреса ячеек, куда  
записать введенные  
числа

```
scanf ("%d%d", &a, &b);
```

**Формат** – символьная строка, которая показывает, какие числа вводятся (выводятся).

**%d** – целое число

**%f** – вещественное число

**%c** – 1 символ

**%s** – символьная строка

**&a** – адрес  
переменной **a**

7652

12

**a** – значение  
переменной **a**

ждать ввода с клавиатуры двух  
целых чисел (через пробел или  
*Enter*), первое из них записать в  
переменную **a**, второе – в **b**

# Что неправильно?

```
int a, b;
```

```
scanf ("%d", a);
```

```
scanf ("%d", &a, &b);
```

```
scanf ("%d%d", &a);
```

убрать пробел

```
scanf ("%d %d", &a, &b);
```

```
scanf ("%f%f", &a, &b);
```

&a

%d%d

&a, &b

%d%d

# Вывод чисел на экран

здесь вывести  
целое число

это число взять  
из ячейки **c**

```
printf ("%d", c);
```

```
printf ("Результат: %d", c);
```

```
printf ("%d+%d=%d", a, b, c);
```

формат вывода

список значений

```
printf ("%d+%d=%d", a, b, a+b);
```

арифметическое  
выражение

# Вывод целых чисел

```
int x = 1234;  
printf ("%d", x);
```

или "%i"

1234

МИНИМАЛЬНОЕ  
ЧИСЛО ПОЗИЦИЙ

или "%9i"

```
printf ("%9d", x);
```

1234

всего 9 позиций

5

4

# Вывод вещественных чисел

```
float x = 123.4567;  
printf ("%f", x);
```

123.456700

минимальное число  
позиций, **6 цифр** в  
дробной части

```
printf ("%9.3f",  
x);
```

123.456

всего 9 позиций,  
**3 цифры** в дробной  
части

```
printf ("%e", x);
```

1.234560e+02

стандартный вид:  
 $1,23456 \cdot 10^2$

```
printf ("%10.2e", x);
```

1.23e+02

всего 10 позиций,  
**2 цифры** в дробной  
части мантиссы

# Полное решение

```
#include <stdio.h>
#include <conio.h>
main()
{
    int a, b, c;
    printf("Введите два целых числа\n");
    scanf("%d%d", &a, &b);
    c = a + b;
    printf("%d+%d=%d", a, b, c);
    getch();
}
```

## Протокол:

Введите два целых числа

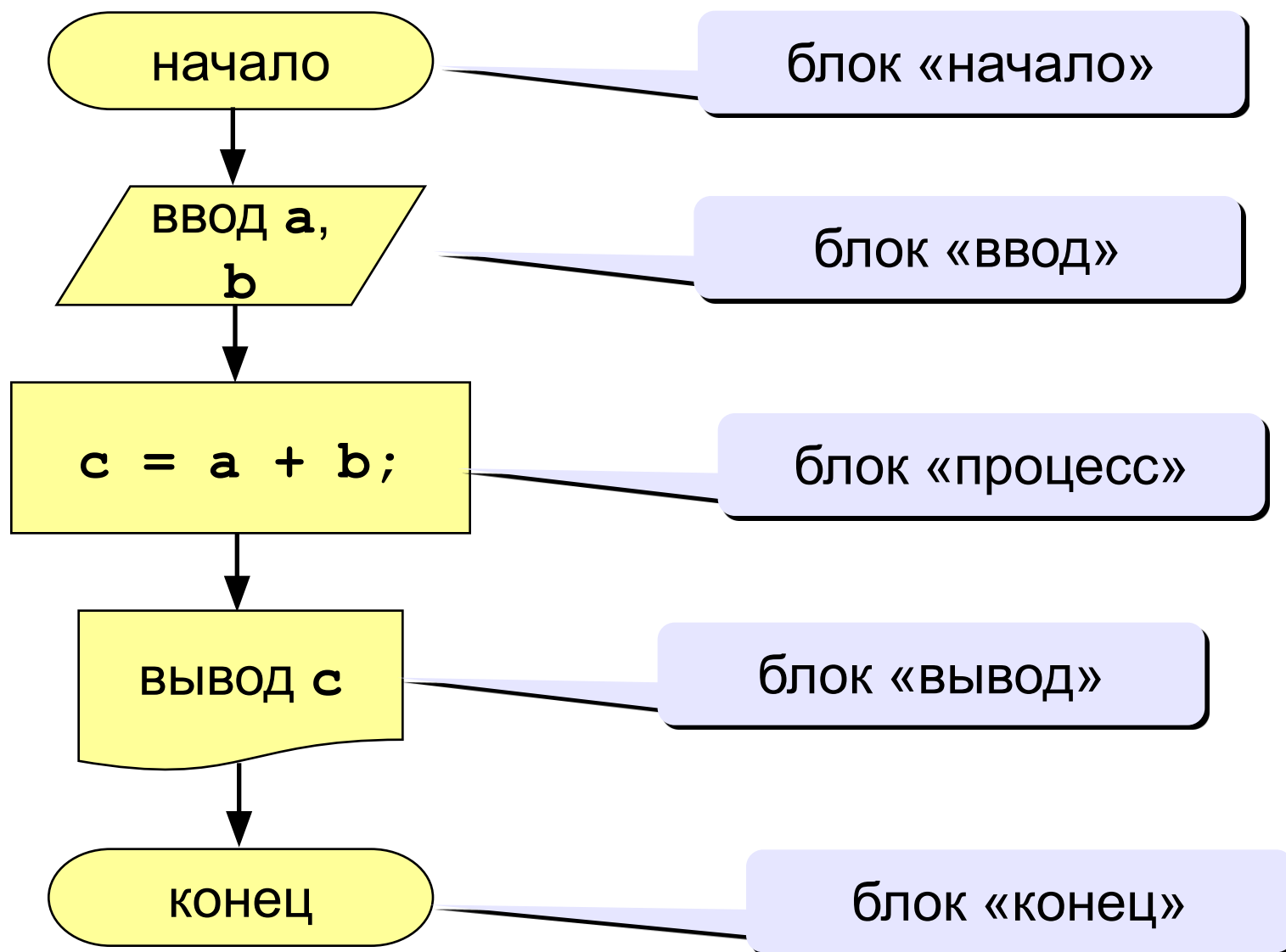
25 30

25+30=55

ЭТО ВЫВОДИТ  
КОМПЬЮТЕР

ЭТО ВВОДИТ  
ПОЛЬЗОВАТЕЛЬ

# Блок-схема линейного алгоритма



# Задания

---

**«3»:** Ввести три числа, найти их сумму.

**Пример:**

Введите три числа:

4 5 7

$$4+5+7=16$$

**«4»:** Ввести три числа, найти их сумму и произведение.

**Пример:**

Введите три числа:

4 5 7

$$4+5+7=16$$

$$4*5*7=140$$



# Задания

---

**«5»:** Ввести три числа, найти их сумму, произведение и среднее арифметическое.

**Пример:**

Введите три числа:

4 5 7

$$4+5+7=16$$

$$4*5*7=140$$

$$(4+5+7)/3=5.33$$

# Какие операторы неправильные?

```
main ()
{
    int a, b;
    float x, y;
    a = 5;
    10 = x;
    y = 7,8;
    b = 2.5;
    x = 2*(a + y);
    a = b + x;
}
```

имя переменной  
должно быть слева  
от знака =

целая и дробная часть  
отделяются точкой

при записи вещественного  
значения в целую  
переменную **дробная  
часть будет отброшена**

# Особенность деления в Си

**!** При делении целых чисел остаток отбрасывается!

```
main ()
{
  int a = 7;
  float x;
  x = a / 4;
  x = 4 / a;
  x = float(a) / 4;
  x = 1.*a / 4;
}
```

1

0

1.75

1.75

# Сокращенная запись операций в Си

полная запись	сокращенная запись
<code>a = a + 1;</code> <small>инкремент</small>	<code>a++;</code>
<code>a = a + b;</code>	<code>a += b;</code>
<code>a = a - 1;</code> <small>декремент</small>	<code>a--;</code>
<code>a = a - b;</code>	<code>a -= b;</code>
<code>a = a * b;</code>	<code>a *= b;</code>
<code>a = a / b;</code>	<code>a /= b;</code>
<code>a = a % b;</code>	<code>a %= b;</code>

## Порядок выполнения операций

- 1) вычисление выражений в скобках
- 2) умножение, деление и **%**(остаток от деления) слева направо
- 3) сложение и вычитание слева направо

1 2 4 5 3 6

```
z = (5*a+c) / a * (b-c) / b;
```

$$x = \frac{5c^2 - d(a+b)}{(c+d)(d-2a)}$$

$$z = \frac{5a+c}{ab} (b-c)$$

2 3 5 4 1 10 6 9 8 7

```
x = (5*c*c-d*(a+b)) / ((c+d)*(d-2*a))
```

# Ручная прокрутка программы

```
main ()
{
    int a, b;
    a = 5;
    b = a + 2;
    a = (a + 2) * (b - 3);
    b = a / 5;
    a = a % b;
    a++;
    b = (a + 14) % 7;
}
```

a	b
?	?
5	
	7
28	
	5
3	
4	
	4

# Вывод на экран

```
int a = 1, b = 3;  
printf("%d+%d=%d", a, b, a+b);
```

формат  
вывода

список  
вывода

- элементы списка разделяются запятыми
- форматы вывода начинаются с %
- выражения (элементы без кавычек) вычисляются и выводится их результат



Что будет выведено?

1+3=4

# Что будет выведено?

```
int a = 1, b = 3;  
printf("a+%d=a+b", b);
```

a+3=a+b

```
int a = 1, b = 3;  
printf("%d=F(%d)", a, b);
```

1=F(3)

```
int a = 1, b = 3;  
printf("a=F(%d);", b);
```

a=F(3);

```
int a = 1, b = 3;  
printf("%d>%d!", a+b, b);
```

4>3!

```
int a = 1, b = 3;  
printf("F(%d)=X(%d)", b, a);
```

F(3)=X(1)



# Как записать оператор вывода?

```
int a = 1, b = 3  
printf("X(%d)=%d", b, a);
```

X(3)=1

```
int a = 1, b = 3  
printf("%d=%d+%d", a+b, a, b);
```

4=1+3

```
int a = 1, b = 3  
printf("f(%d)>f(%d)", a, b);
```

f(1)>f(3)

```
int a = 1, b = 3  
printf("<%d<>%d>", a, b);
```

<1<>3>

```
int a = 1, b = 3  
printf("%d+%d=?", a, b);
```

1+3=?

# Программирование на языке Си

## Тема 3. Ветвления

# Разветвляющиеся алгоритмы

---

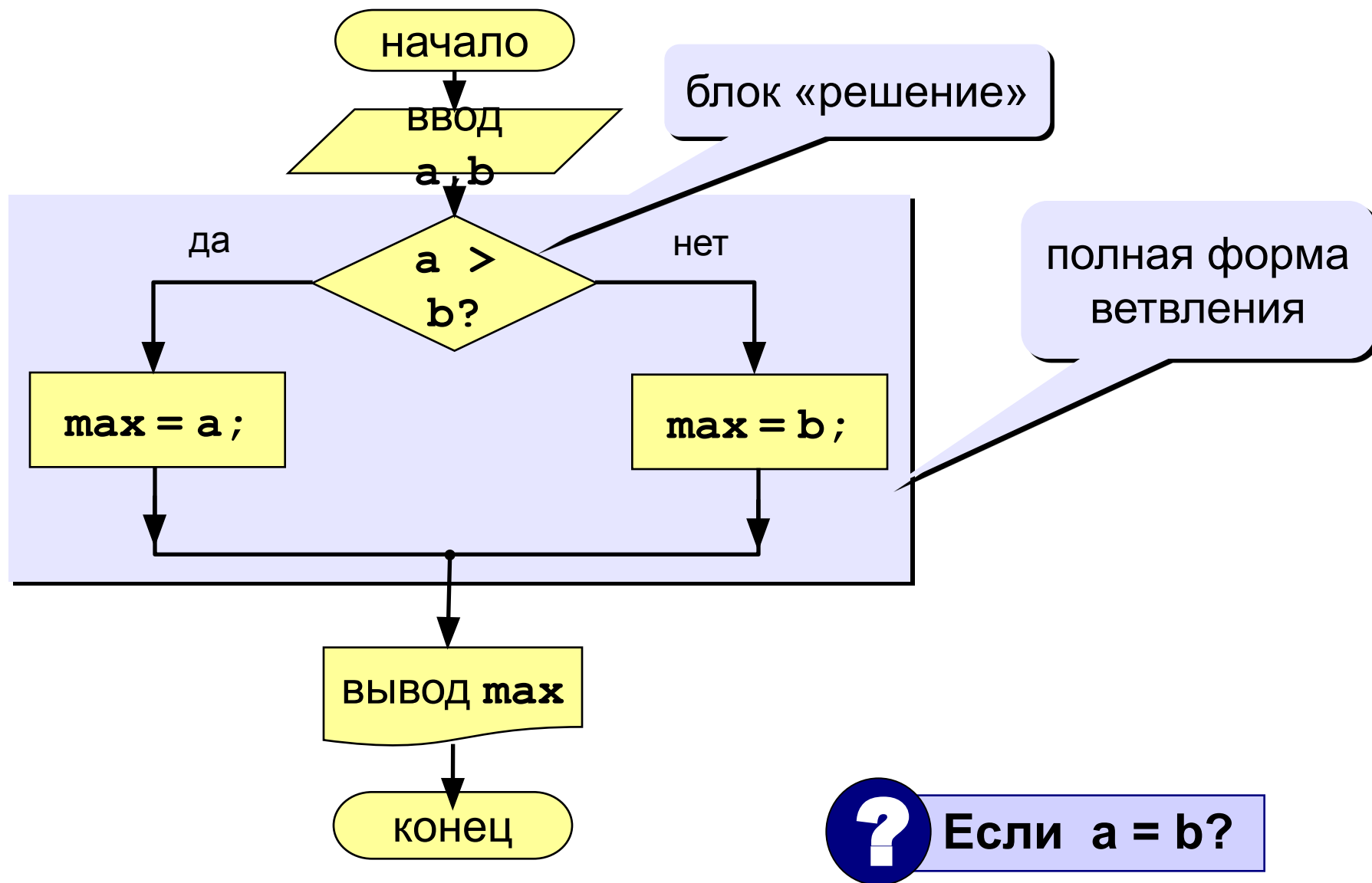
**Задача.** Ввести два целых числа и вывести на экран наибольшее из них.

**Идея решения:** надо вывести на экран первое число, если оно больше второго, или второе, если оно больше первого.

**Особенность:** действия исполнителя зависят от некоторых условий (*если ... иначе ...*).

Алгоритмы, в которых последовательность шагов зависит от выполнения некоторых условий, называются **разветвляющимися.**

# Вариант 1. Блок-схема



# Вариант 1. Программа

```
main ()
{
    int a, b, max;
    printf("Введите два целых числа\n");
    scanf("%d%d", &a, &b );

    if (a > b) {
        max = a;
    }
    else {
        max = b;
    }

    printf("Наибольшее число %d", max);
}
```

полная форма  
условного  
оператора

# Условный оператор

```
if ( условие )  
  {  
    // что делать, если условие верно  
  }  
else  
  {  
    // что делать, если условие неверно  
  }
```

## Особенности:

- вторая часть (***else*** ...) может отсутствовать (неполная форма)
- если в блоке один оператор, можно убрать { }

# Что неправильно?

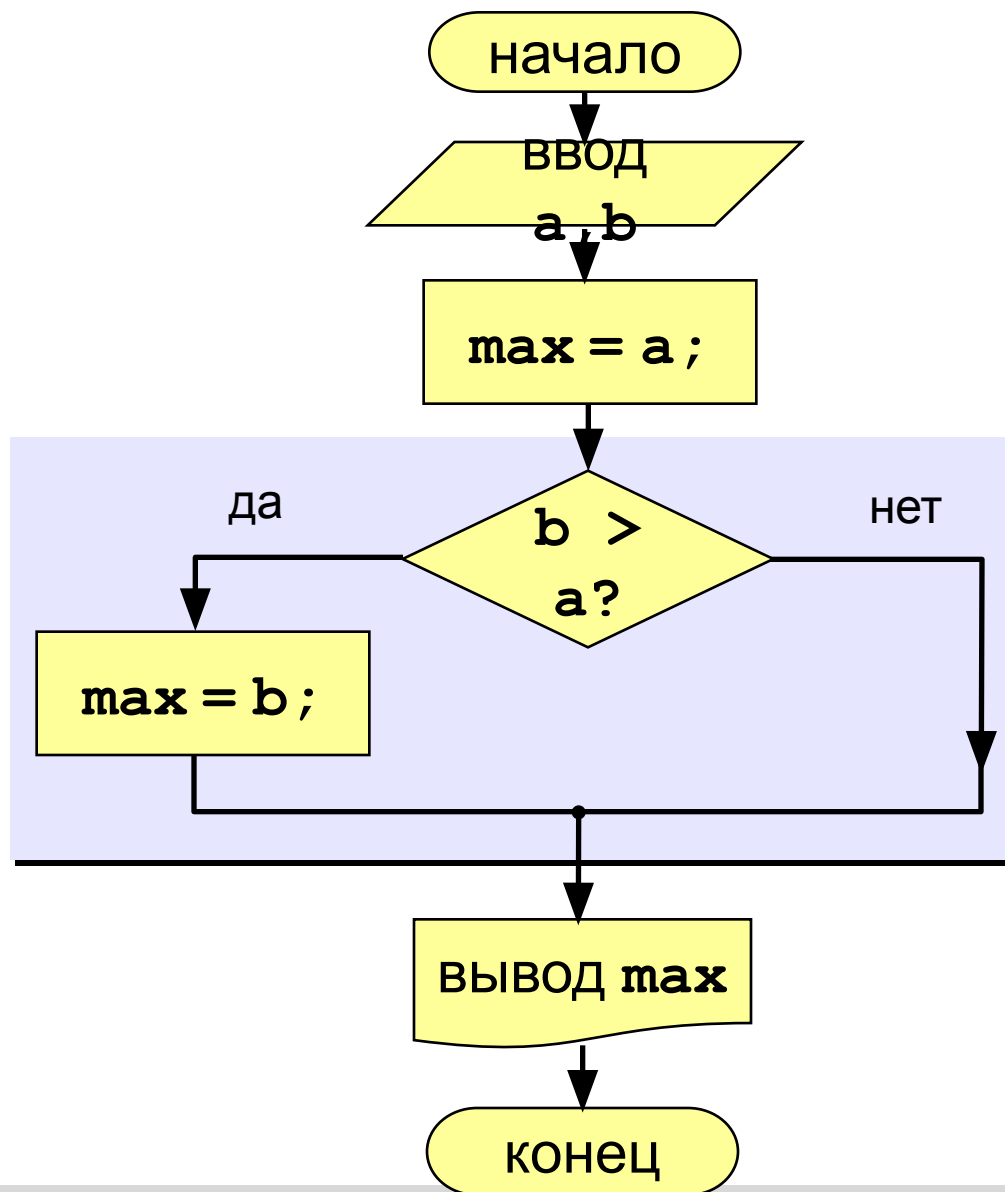
```
if ( a > b ) {  
    a = b;  
}  
else  
    b = a;
```

```
if ( a > b ) {  
    a = b; }  
else  
    b = a;
```

```
if ( a > b ) a = b;  
else  
    b = a;
```

```
if ( a > b ) {  
    a = b;  
    c = 2*a; }  
else  
    b = a;
```

## Вариант 2. Блок-схема



неполная форма  
ветвления



## Вариант 2. Программа

```
main ()
{
    int a, b, max;
    printf("Введите два целых числа\n");
    scanf("%d%d", &a, &b );
    max = a;
    if (b > a)
        max = b;
    printf("Наибольшее число %d", max);
}
```

неполная форма  
условного  
оператора

## Вариант 2Б. Программа

```
main ()
{
    int a, b, max;
    printf("Введите два целых числа\n");
    scanf("%d%d", &a, &b );
    max = b;
    if ( a > b )
        max = a;
    printf("Наибольшее число %d", max);
}
```

# Задания

---

**«3»:** Ввести два числа и вывести их в порядке возрастания.

*Пример:*

Введите два числа:

15 9

Ответ: 9 15

**«4»:** Ввести три числа и найти наибольшее из них.

*Пример:*

Введите три числа:

4 15 9

Наибольшее число 15

# Задания

---

**«5»:** Ввести пять чисел и найти наибольшее из них.

**Пример:**

Введите пять чисел:

**4      15      9      56      4**

Наибольшее число **56**

# Программирование на языке Си

## Тема 4. Сложные условия

## Сложные условия

---

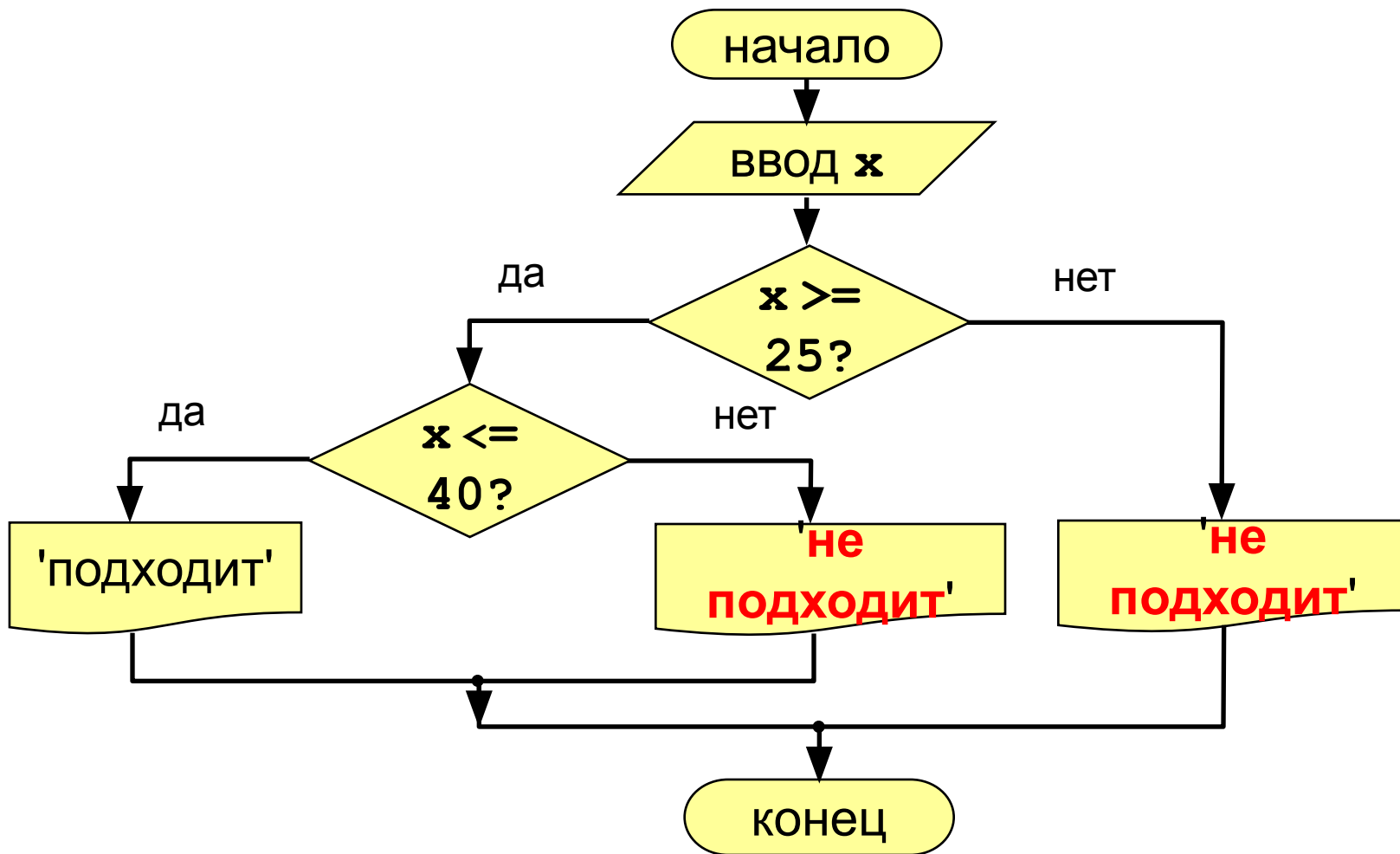
**Задача.** Фирма набирает сотрудников от 25 до 40 лет включительно. Ввести возраст человека и определить, подходит ли он фирме (вывести ответ «подходит» или «не подходит»).

**Особенность:** надо проверить, выполняются ли два условия одновременно.



Можно ли решить известными методами?

# Вариант 1. Алгоритм



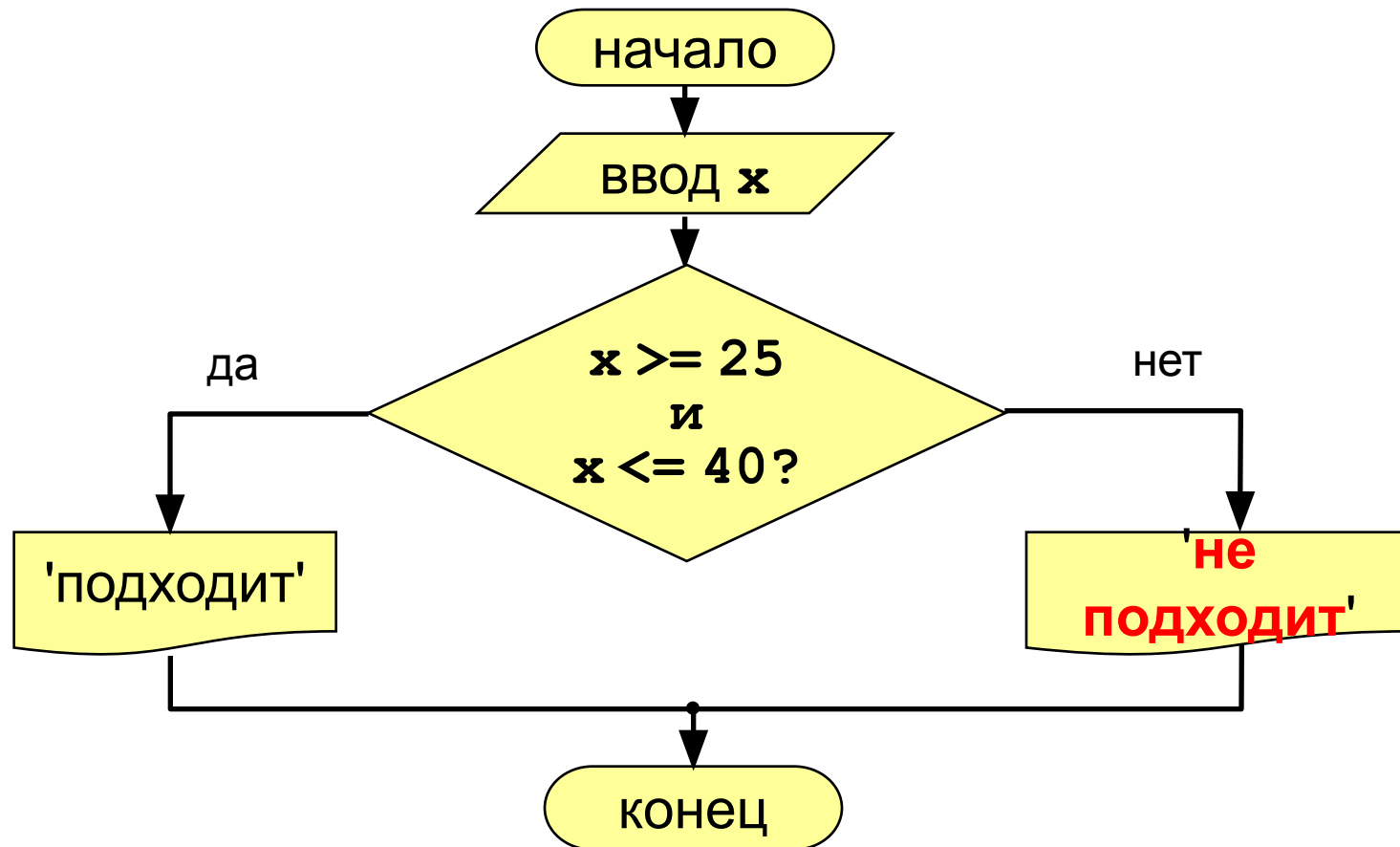
# Вариант 1. Программа

```
main()
{
    int x;
    printf("Введите возраст\n");
    scanf("%d", &x);

    if (x >= 25)
    {
        if (x <= 40)
            printf("Подходит");
        else printf("Не подходит");
    }
    else
        printf("Не подходит");
}
```



# Вариант 2. Алгоритм



## Вариант 2. Программа

```
main ()
{
  int x;
  printf("Введите возраст\n");
  scanf("%d", &x);
  if ( x >= 25 && x <= 40 )
    printf("Подходит");
  else printf("Не подходит");
}
```

СЛОЖНОЕ  
УСЛОВИЕ

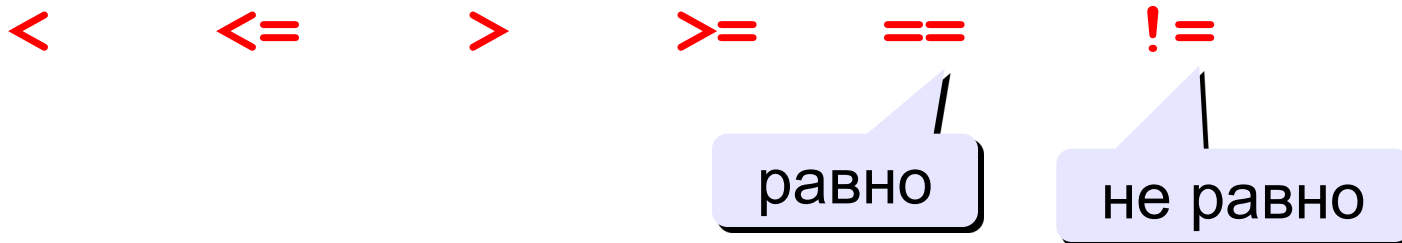
# Сложные условия

---

**Сложное условие** – это условие, состоящее из нескольких простых условий (отношений), связанных с помощью **логических операций**:

- ! – НЕ (*not*, отрицание, инверсия)
- && – И (*and*, логическое умножение, конъюнкция, одновременное выполнение условий)
- || – ИЛИ (*or*, логическое сложение, дизъюнкция, выполнение хотя бы одного из условий)

## Простые условия (отношения)



# Сложные условия

---

## Порядок выполнения сложных условий:

- выражения в скобках
- ! (НЕ, отрицание)
- <, <=, >, >=
- ==, !=
- && (И)
- || (ИЛИ)

## Пример:

```
if ( 2      1      6      3      5      4
    ! (a > b) || c != d && b == a )
{
    ...
}
```

# Сложные условия

Истинно или ложно при  $a = 2$ ;  $b = 3$ ;  $c = 4$ ;

$!(a > b)$

1

$a < b \ \&\& \ b < c$

1

$!(a \geq b) \ || \ c == d$

1

$a < c \ || \ b < c \ \&\& \ b < a$

1

$a > b \ || \ !(b < c)$

0

Для каких значений  $x$  истинны условия:

$x < 6 \ \&\& \ x < 10$

$x < 6 \ \&\& \ x > 10$

$x > 6 \ \&\& \ x < 10$

$x > 6 \ \&\& \ x > 10$

$x < 6 \ || \ x < 10$

$x < 6 \ || \ x > 10$

$x > 6 \ || \ x < 10$

$x > 6 \ || \ x > 10$

$(-\infty, 6)$	$x < 6$
$\emptyset$	
$(6, 10)$	
$(10, \infty)$	$x > 10$
$(-\infty, 10)$	$x < 10$
$(-\infty, 6) \cup (10, \infty)$	
$(-\infty, \infty)$	
$(6, \infty)$	$x > 6$

# Задания

---

**«3»:** Ввести три числа и определить, верно ли, что они вводились в порядке возрастания.

*Пример:*

Введите три числа:

4 5 17

да

**«4»:** Ввести номер месяца и вывести название времени года.

*Пример:*

Введите номер месяца:

4

весна

# Задания

---

**«5»:** Ввести возраст человека (от 1 до 150 лет) и вывести его вместе с последующим словом «год», «года» или «лет».

**Пример:**

Введите возраст:

**24**

Вам 24 года

Введите возраст:

**57**

Вам 57 лет

# Программирование на языке Си

## Тема 5. Циклы



# Циклы

---

**Цикл** – это многократное выполнение одинаковых действий.

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

**Задача.** Вывести на экран 5 раз слово «Привет».

**Особенность:** одинаковые действия выполняются 5 раз.



Можно ли решить известными методами?

# ЦИКЛЫ

---

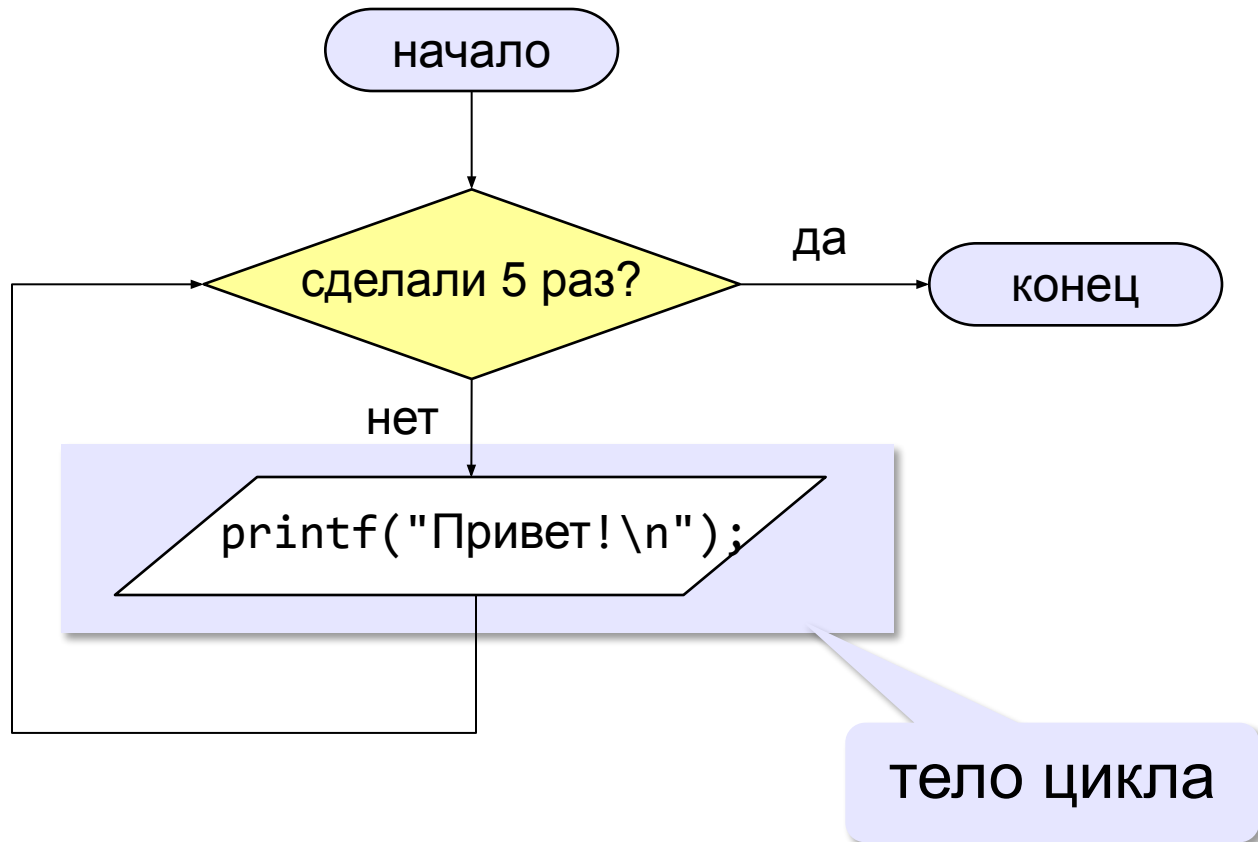
```
#include <stdio.h>
main ()
{
    printf ("Привет! \n" );
    printf ("Привет! \n" );
    printf ("Привет! \n" );
    printf ("Привет! \n" );
    printf ("Привет! \n" );
}
```



Что плохо?

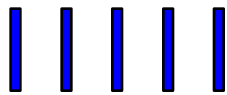
# Циклы


Блок-схема:

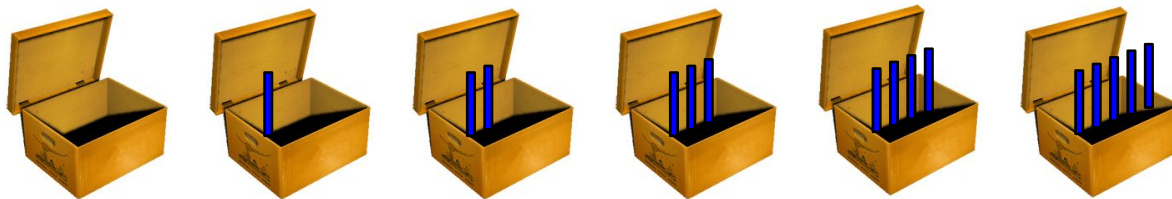


# Циклы

 Как отсчитать ровно 5 раз?

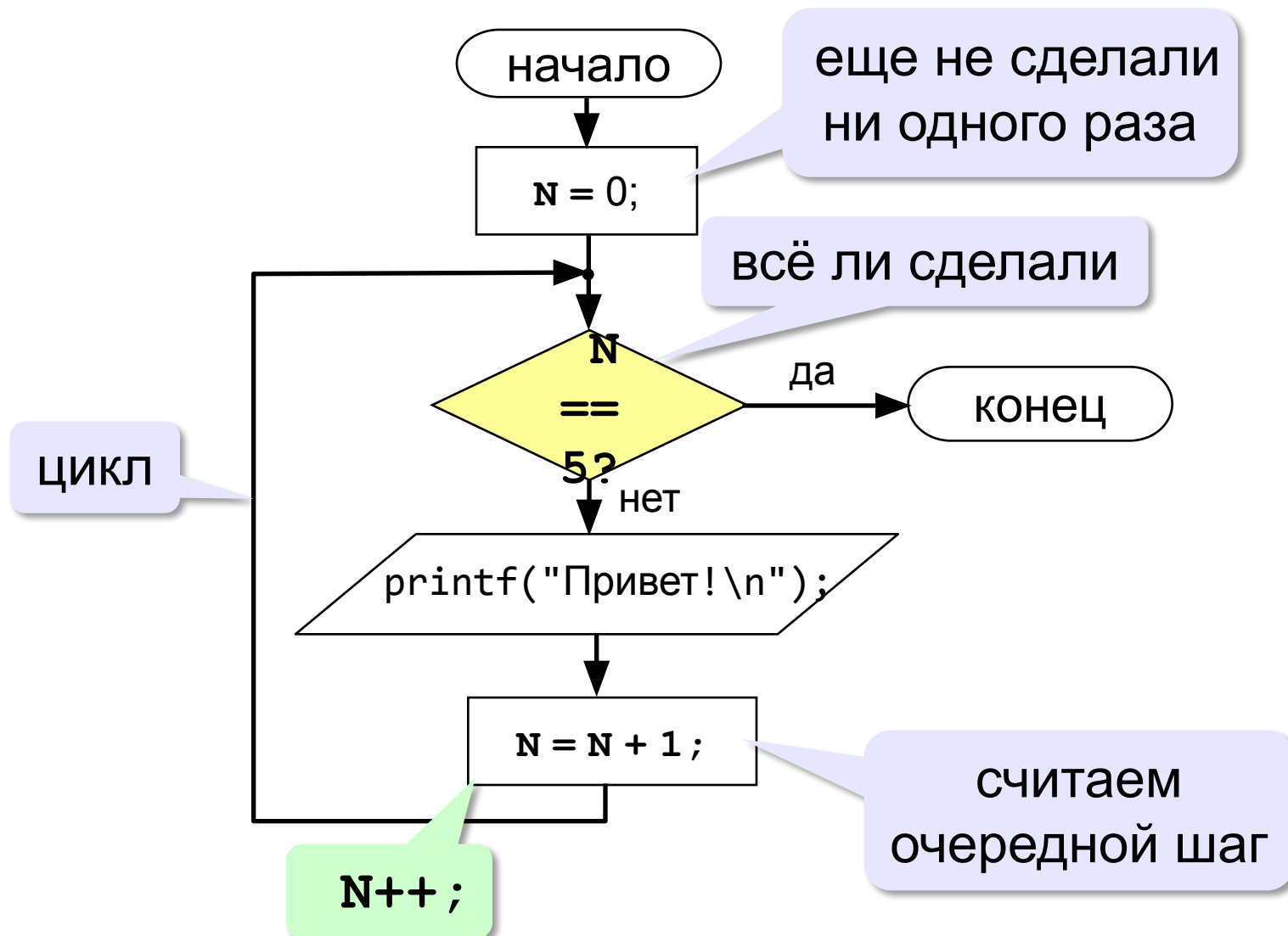


 Как запоминать, сколько раз уже сделали?



$N := N + 1$

# Циклы



# Циклы с условием

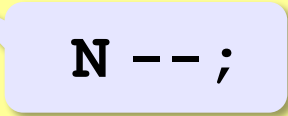
---

```
main ()
{
    int N;
    N = 0;
    while ( N != 5 )
    {
        printf ("Привет! \n" );
        N ++;
    }
}
```

## Цикл с условием

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

```
main ()
{
    int N;
    N = 5;
    while ( N != 0 )
    {
        printf ("Привет! \n" );
        N = N - 1;
    }
}
```



# Что получим?

```
// Пример 1
main()
{
    int N;
    N = 1;
    while ( N <= 5 )
    {
        printf("%d\n", N);
        N ++;
    }
}
```



1  
2  
3  
4  
5



# Что получим?

```
// Пример 2
main()
{
    int N;
    N = 1;
    while ( N <= 5 )
    {
        printf("%d\n", N);
        N = N + 2;
    }
}
```

N += 2;



1  
3  
5

# Что получим?

```
// Пример 3
main()
{
    int N;
    N = 2;
    while ( N != 5 )
    {
        printf("%d\n", N);
        N += 2;
    }
}
```



2  
4  
6  
8  
10  
12  
14  
16  
...



Условие цикла никогда не станет ложным – это закливание!

# Что получим?

```
// Пример 4
main()
{
    int N;
    N = 1;
    while ( N != 5 )
    {
        printf("%d\n", N*N*N);
        N = N + 1;
    }
}
```



1  
8  
27  
64  
125

# Что получим?

```
// Пример 4
main()
{
    int N;
    N = 5;
    while ( N >= 1 )
    {
        printf("%d\n", N*N*N);
        N = N - 1;
    }
}
```



125  
64  
27  
8  
1

## Задания

---

**«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  $b$  ( $a \leq b$ ) и вывести квадраты все чисел от  $a$  до  $b$ .

*Пример:*

Введите два числа:

4 5

4\*4=16

5\*5=25

# Задания

---

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

*Пример:*

Введите два числа:

**4 10**

Сумма квадратов 371

## Цикл с неизвестным числом шагов

---

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

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

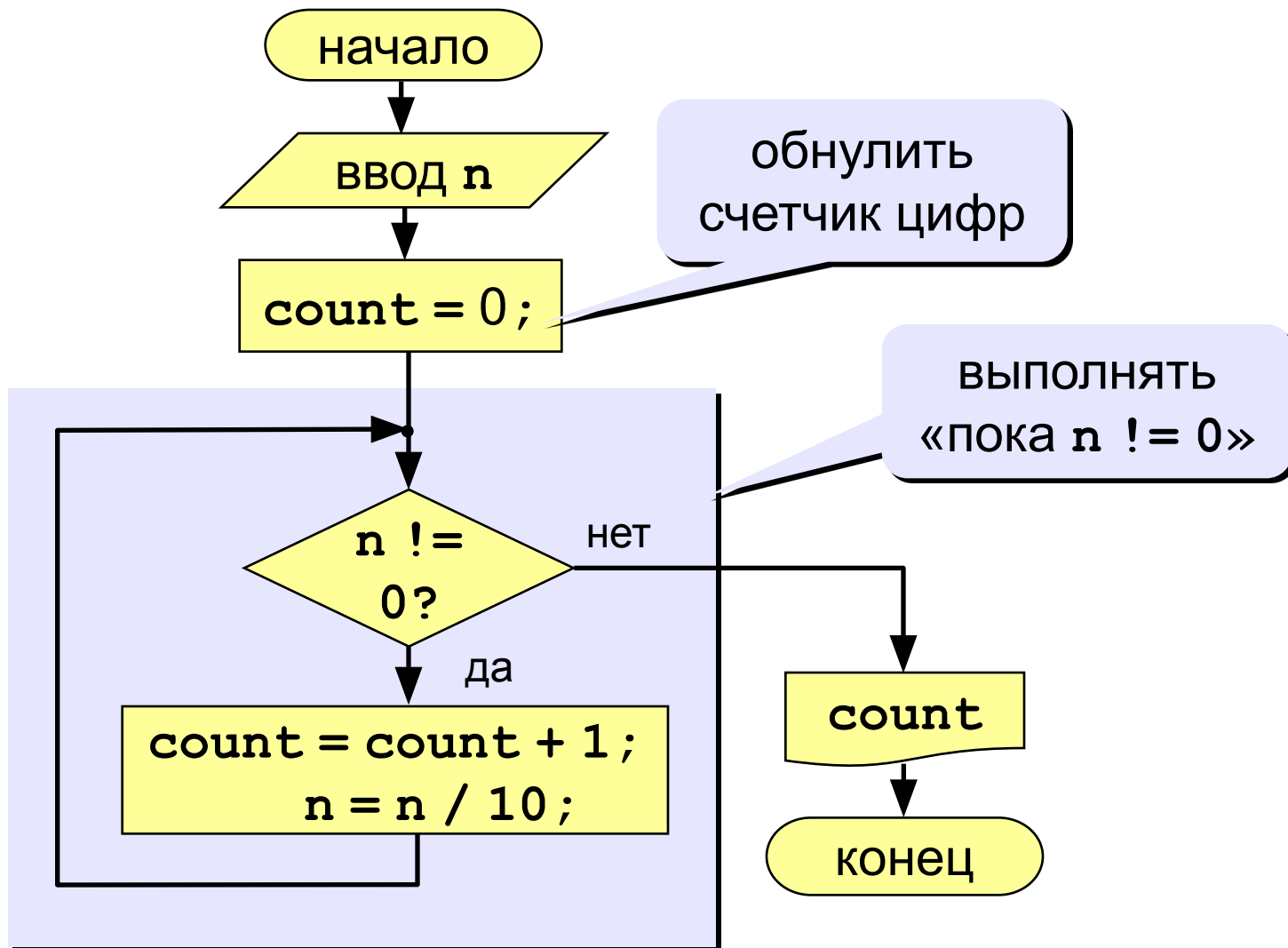
**Идея решения:** Отсекаем последовательно последнюю цифру, увеличиваем счетчик.

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

```
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**

## Задания

---

**«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.

# Задания

---

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

*Пример:*

Введите целое число:

**1232**

Нет.

Введите целое число:

**1224**

Да.

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

*Пример:*

Введите целое число:

**1234**

Нет.

Введите целое число:

**1242**

Да.

## Задания-2

---

**«3»:** Ввести целое число и определить, верно ли, что в нём ровно 1 цифра «9».

*Пример:*

Введите число:

**193**

Да .

Введите число:

**1994**

Нет .

**«4»:** Ввести целое число и определить, верно ли, что все его цифры четные.

*Пример:*

Введите число:

**2684**

Да .

Введите число:

**2994**

Нет .

## Задания-2

---

**«5»:** Ввести целое число и определить, верно ли, что все его цифры расположены в порядке возрастания.

*Пример:*

Введите целое число:

1238

Да.

Введите целое число:

1274

Нет.

**«6»:** Ввести целое число и «перевернуть» его, так чтобы первая цифра стала последней и т.д.

*Пример:*

Введите целое число:

1234

4321

Введите целое число:

782

287



# Вычисление НОД

**НОД** = наибольший общий делитель двух натуральных чисел – это наибольшее число, на которое оба исходных числа делятся без остатка.

## Перебор:

1. Записать в переменную **k** минимальное из двух чисел.
2. Если **a** и **b** без остатка делятся на **k**, то стоп.
3. Уменьшить **k** на 1.
4. Перейти к шагу 2.



Где будет НОД?



Почему алгоритм обязательно закончится?

это цикл с условием!

# Алгоритм Евклида

**Надо:** вычислить наибольший общий делитель (НОД) чисел  $a$  и  $b$ .

Заменяем большее из двух чисел **разностью** большего и меньшего до тех пор, пока они не станут равны. Это и есть НОД.

$$\begin{aligned}\text{НОД}(a, b) &= \text{НОД}(a-b, b) \\ &= \text{НОД}(a, b-a)\end{aligned}$$



Евклид  
(365-300 до. н. э.)

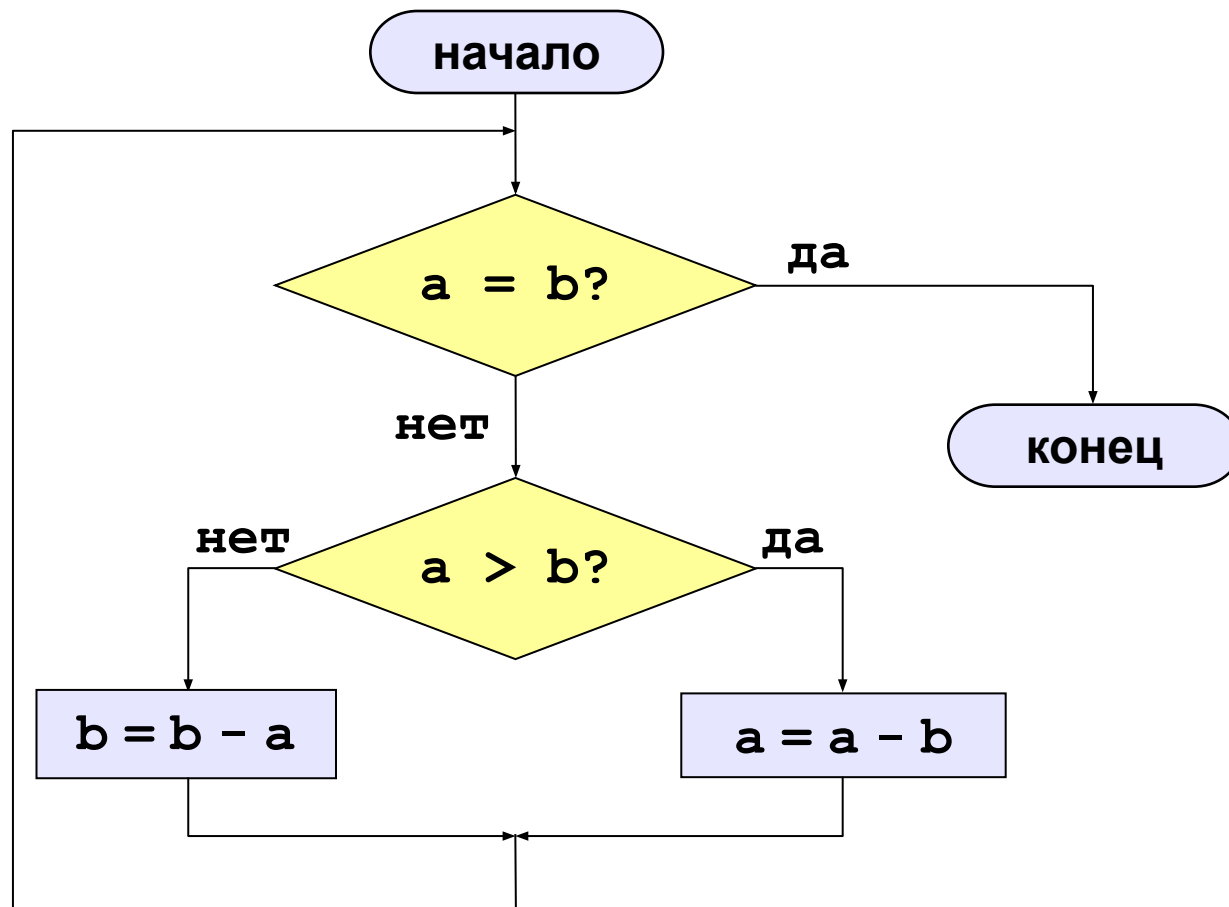
## Пример:

$$\begin{aligned}\text{НОД}(14, 21) &= \text{НОД}(14, 21-14) = \text{НОД}(14, 7) \\ &= \text{НОД}(7, 7) = 7\end{aligned}$$

⊖ много шагов при большой разнице чисел:

$$\text{НОД}(1998, 2) = \text{НОД}(1996, 2) = \dots = 2$$

# Блок-схема алгоритма



# Алгоритм Евклида

```
while ( a != b )  
{  
    if ( a > b )  
        a = a - b;  
    else b = b - a;  
}
```



Где будет НОД? Как его вывести?



Как вывести НОД в формате  $\text{НОД}(14,21) = 7$ ?



А без дополнительных переменных?

# Модифицированный алгоритм Евклида

Заменяем большее из двух чисел **остатком от деления** большего на меньшее до тех пор, пока меньшее не станет равно нулю. Тогда большее — это НОД.

$$\begin{aligned}\text{НОД}(a, b) &= \text{НОД}(a \% b, b) \\ &= \text{НОД}(a, b \% a)\end{aligned}$$

**Пример:**

$$\text{НОД}(14, 21) = \text{НОД}(14, 7) = \text{НОД}(0, 7) =$$

**Еще <sup>7</sup> один вариант:**

$$\text{НОД}(2 \cdot a, 2 \cdot b) = 2 \cdot \text{НОД}(a, b)$$

$$\text{НОД}(2 \cdot a, b) = \text{НОД}(a, b) \quad | \quad \text{при нечетном } b$$

# Алгоритм Евклида

---

«3»: Составить программу для вычисления НОД с помощью алгоритма Евклида.

«4»: Составить программу для вычисления НОД с помощью **модифицированного** алгоритма Евклида и заполнить таблицу:

a	64168	358853	6365133	17905514	549868978
b	82678	691042	11494962	23108855	298294835
НОД (a, b)					

# Алгоритм Евклида

---

**«5»:** Выполнить задание на «4» и подсчитать число шагов алгоритма для каждого случая.

<b>a</b>	64168	358853	6365133	17905514	549868978
<b>b</b>	82678	691042	11494962	23108855	298294835
<b>НОД (a , b)</b>					
<b>шагов</b>					

# 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):**

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

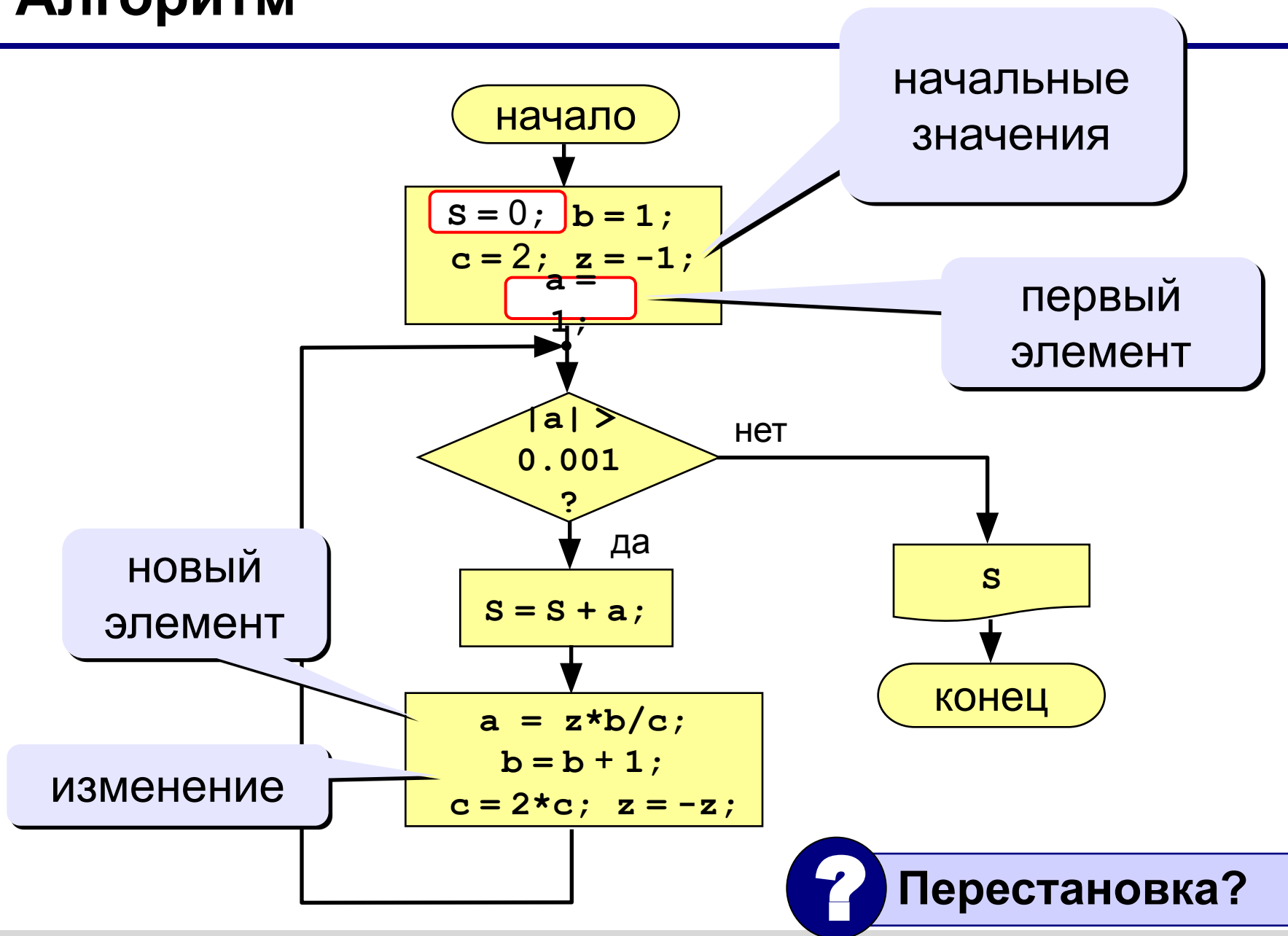
<b>n</b>	1	2	3	4	5	...
<b>b</b>	1	2	3	4	5	...
<b>c</b>	2	4	8	16	32	...
<b>z</b>	-1	1	-1	1	-1	...

**b = b+1;**

**c = 2\*c;**

**z = -z;**

# Алгоритм



# Программа

```

#include <math.h>
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 действительного числа

увеличение суммы

расчет элемента последовательности

**? Что плохо?**

## Задания

---

**«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:**

$$S = 1.157$$

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

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

**Ответ:**

$$S = 1.220$$

# Цикл с постусловием

---

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

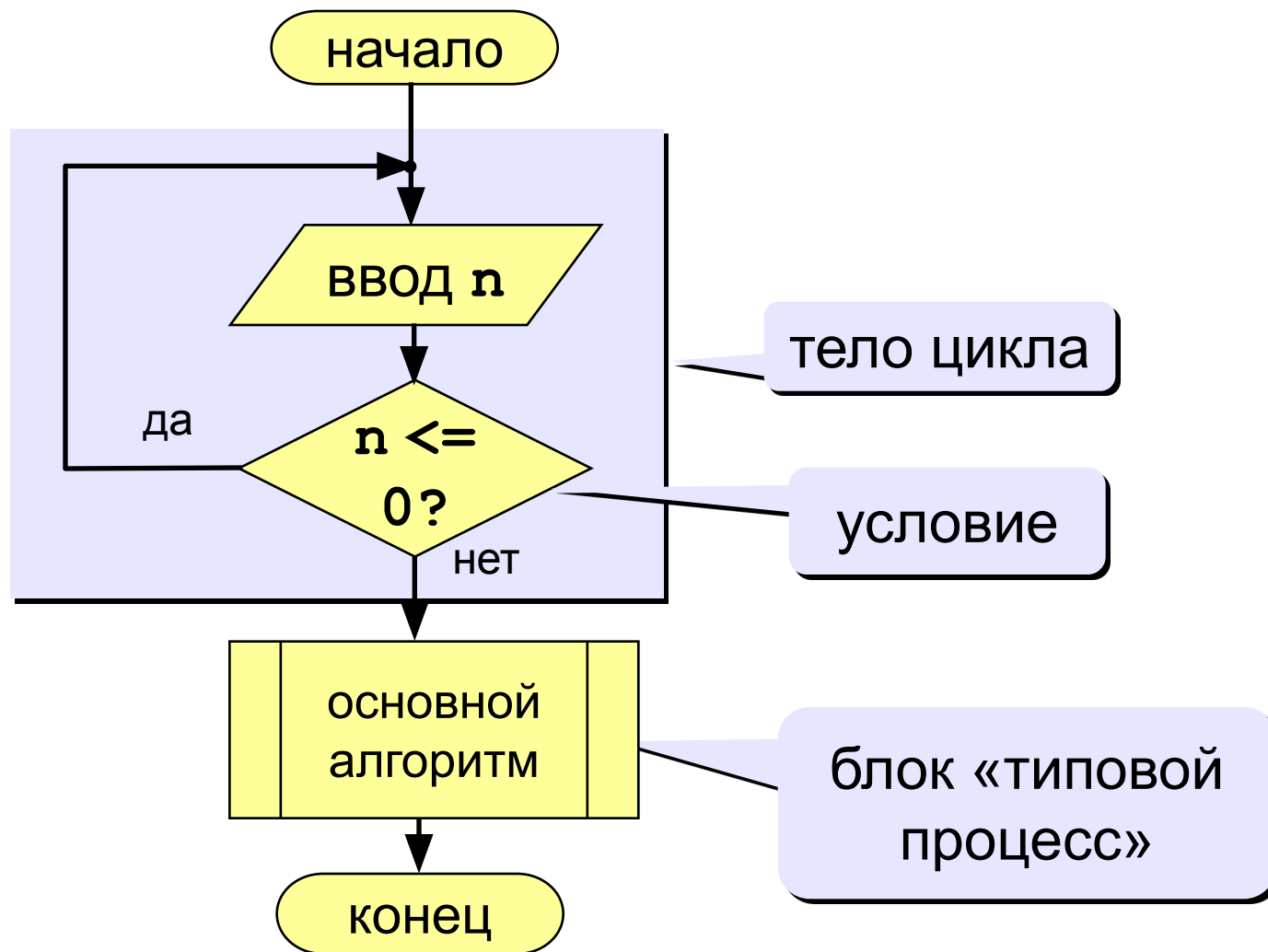
**Проблема:** Как не дать ввести отрицательное число или ноль?

**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



# Программа

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



## 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**



## Задания (с защитой от неверного ввода)

---

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

Пример:

Введите число  $\geq 0$ :

**-234**

Нужно положительное число.

Введите число  $\geq 0$ :

**1234**

Да

Введите число  $\geq 0$ :

**1233**

Нет

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

Пример:

Введите число  $\geq 0$ :

**2323**

Повторяются: 2, 3

Введите число  $\geq 0$ :

**1234**

Нет повторов.

# Программирование на языке Си

## Тема 6. Циклы с переменной

# Цикл с переменной

---

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

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

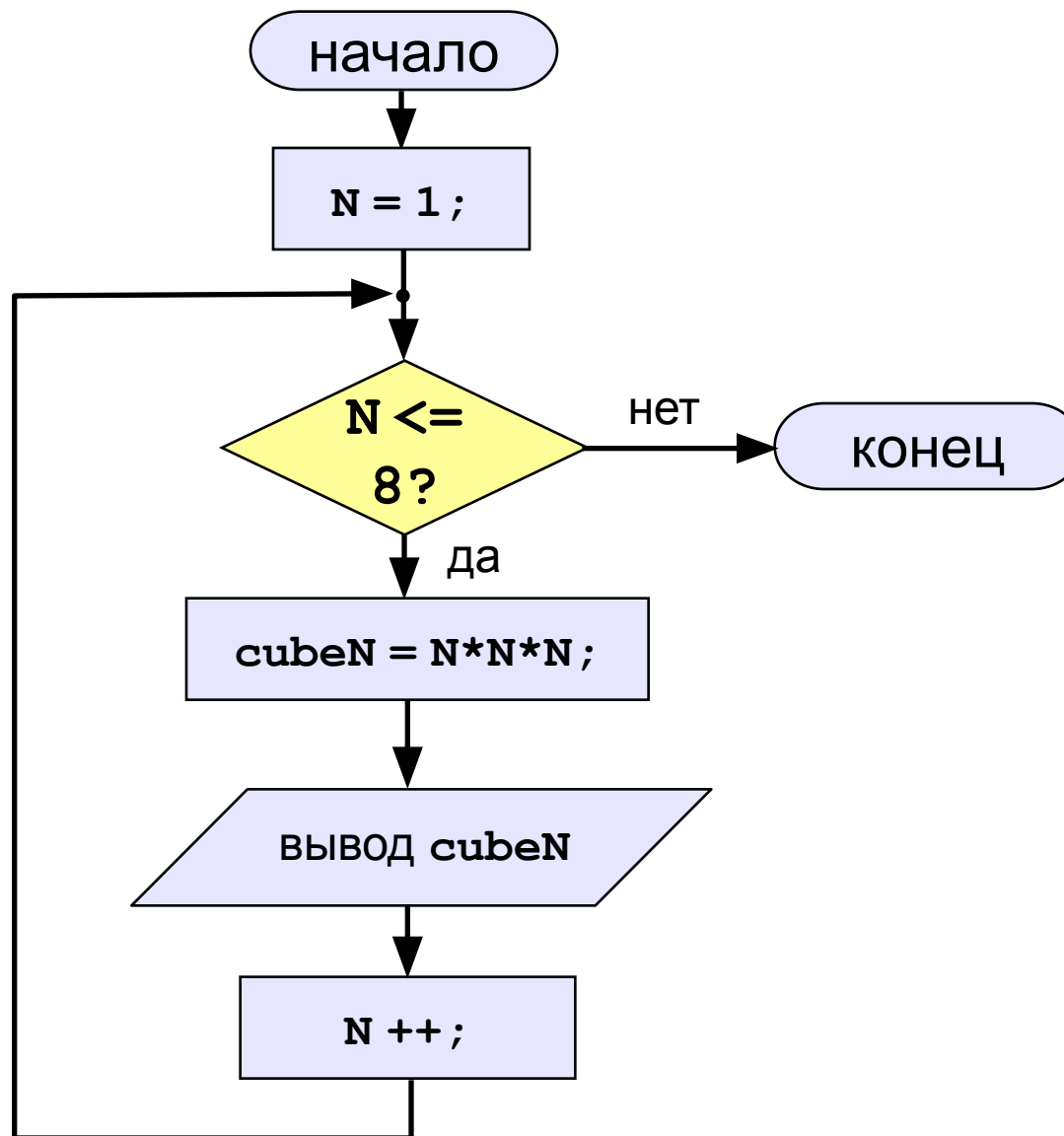
**Задача.** Вывести на экран кубы целых чисел от 1 до 8 (от **a** до **b**).

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



**Можно ли решить известными методами?**

# Алгоритм



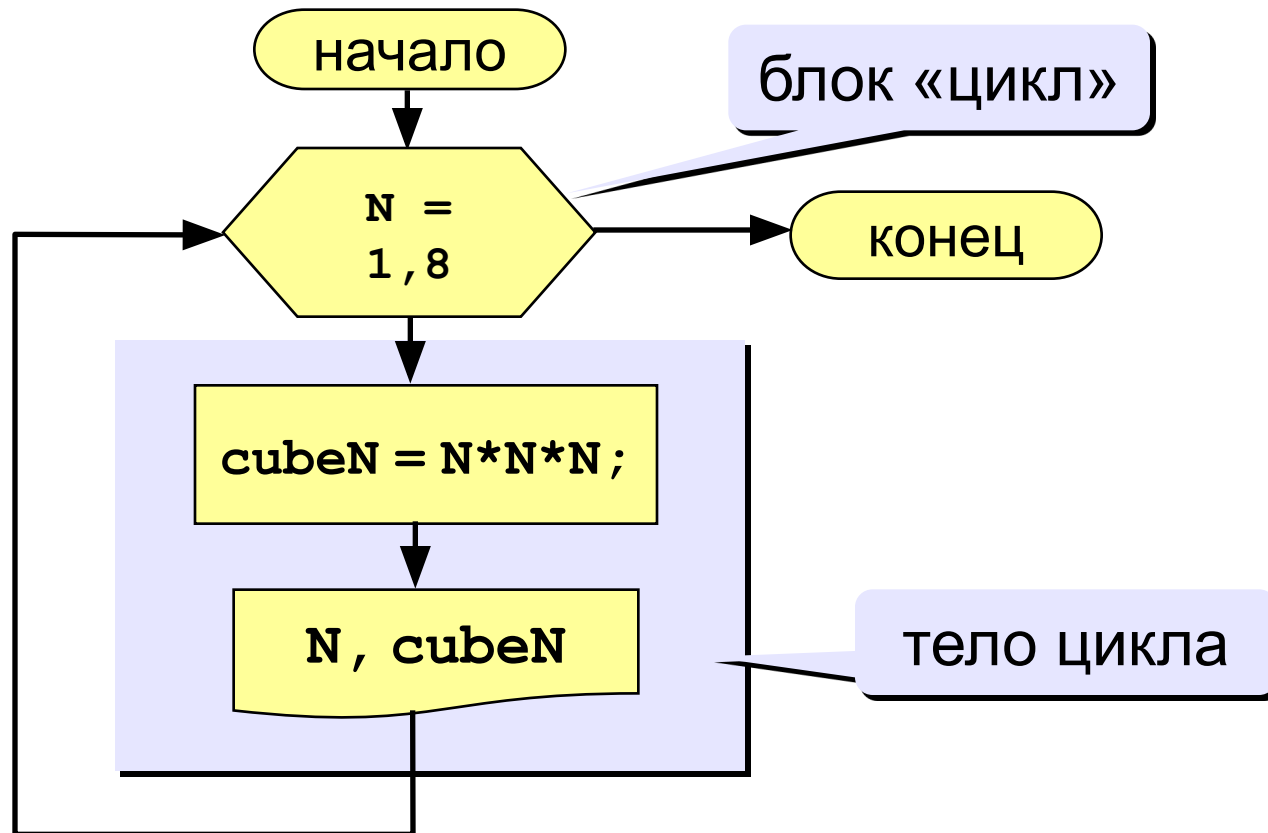
# Цикл с переменной

Задача: вывести кубы натуральных чисел от 1 до 8.

```
main ()
{
    int N, cubeN;
    N = 1;
    while ( N <= 8 ) {
        cubeN = N*N*N;
        printf ("%4d\n", cubeN);
        N++;
    }
}
```

3 действия с N

# Алгоритм (с блоком «цикл»)



# Программа

```
main ()
```

```
{
```

```
int N, cubeN;
```

переменная цикла

начальное  
значение  
заголовка  
цикла

конечное  
значение

ЦИКЛ

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

начало цикла

работает, пока это

изменение на  
каждом шаге:

i=: тело цикла

```
cubeN = N*N*N;
```

```
printf ("%4d %4d\n", N, cubeN);
```

```
}
```

конец цикла

ровные  
столбики

```
}
```

# 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; N >= 1; N -- )  
{  
    cubeN = N*N*N;  
    printf("%4d %4d\n", N, cubeN);  
}
```



# 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++;
```

a = 4

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

a = 7

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

a = 7

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

a = 1

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

# Задания

---

**«3»:** Ввести натуральное число  $N$  и вывести числа от  $N$  до 1 (через одно) в порядке убывания.

*Пример:*

Введите натуральное число:

**8**

Ответ: 8 6 4 2

## Задания

---

**«4»:** Ввести два целых числа  $a$  и  $b$  ( $a \leq b$ ) и вывести кубы всех чисел от  $a$  до  $b$ .

*Пример:*

Введите два числа:

4 6

$$4 * 4 * 4 = 64$$

$$5 * 5 * 5 = 125$$

$$6 * 6 * 6 = 216$$

**«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

# Задания

---

**«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  $b$  ( $a \leq b$ ) и вывести квадраты все чисел от  $a$  до  $b$ .

*Пример:*

Введите два числа:

4 5

4\*4=16

5\*5=25



# Задания

---

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

*Пример:*

Введите два числа:

**4 10**

Сумма квадратов 371

# Программирование на языке Си

## Тема 7. Оператор выбора

# Оператор выбора

---

**Задача:** Ввести номер месяца и вывести количество дней в этом месяце.

**Решение:** Число дней по месяцам:

**28 дней** – 2 (февраль)

**30 дней** – 4 (апрель), 6 (июнь), 9 (сентябрь), 11 (ноябрь)

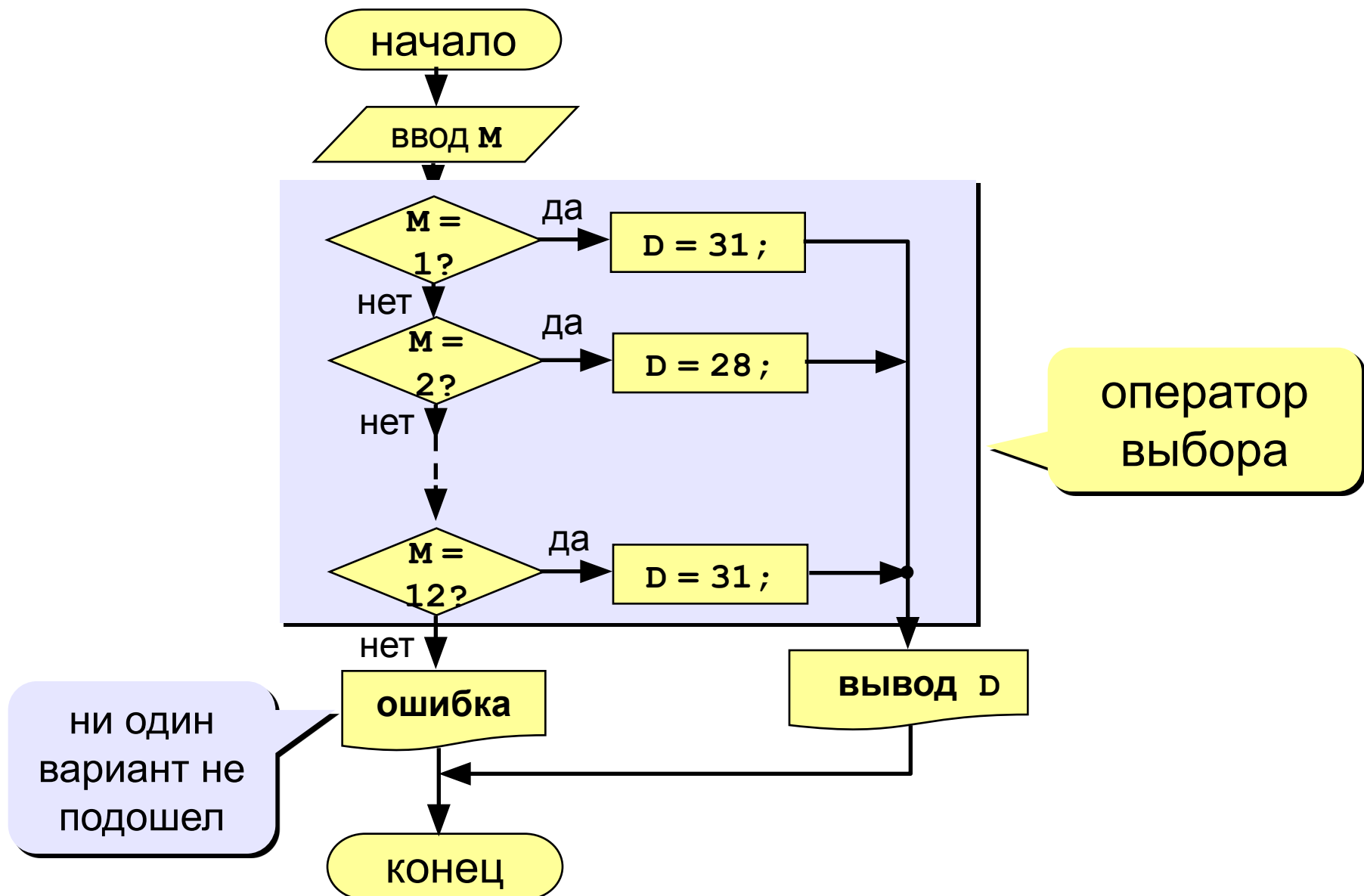
**31 день** – 1 (январь), 3 (март), 5 (май), 7 (июль),  
8 (август), 10 (октябрь), 12 (декабрь)

**Особенность:** Выбор не из двух, а из нескольких вариантов в зависимости от номера месяца.



**Можно ли решить известными методами?**

# Алгоритм



# Программа

```
main()
{
    int M, D;
    printf("Введите номер месяца:\n");
    scanf("%d", &M);

    switch ( M ) {
        case 2:  D = 28; break;
        case 4: case 6: case 9: case 11:
                D = 30; break;
        case 1: case 3: case 5: case 7:
        case 8: case 10: case 12:
                D = 31; break;
        default: D = -1;
    }

    if (D > 0)
        printf("В этом месяце %d дней.", D);
    else printf("Неверный номер месяца");
}
```

ВЫЙТИ ИЗ  
switch

НИ ОДИН  
вариант не  
подошел

# Оператор выбора

**Задача:** Ввести букву и вывести название животного на эту букву.

**Особенность:** выбор по символьной величине.

```
main()
{
    char c;
    printf("Введите первую букву названия животного:\n");
    scanf("%c", &c);

    switch ( c ) {
        case 'a': printf("Антилопа"); break;
        case 'б': printf("Бизон"); break;
        case 'в': printf("Волк"); break;
        default:  printf("Я не знаю!");
    }
}
```



Что будет, если везде убрать break?

# Оператор выбора

---

## Особенности:

- после **switch** может быть имя переменной или арифметическое выражение целого типа (**int**)

```
switch ( i+3 ) {  
    case 1: a = b; break;  
    case 2: a = c;  
}
```

или символьного типа (**char**)

- **нельзя** ставить два **одинаковых** значения:

```
switch ( x ) {  
    case 1: a = b; break;  
    case 1: a = c;  
}
```

## Задания (с защитой от неверного ввода)

---

**«4»:** Ввести номер месяца и вывести количество дней в нем, а также число ошибок при вводе.

**Пример:**

Введите номер месяца:

**-2**

Введите номер месяца:

**11**

В этом месяце 30 дней.

Вы вводили неверно 1 раз.

Введите номер месяца:

**2**

В этом месяце 28 дней.

Вы вводили неверно 0 раз.

**«5»:** Ввести номер месяца и номер дня, вывести число дней, оставшихся до Нового года.

**Пример:**

Введите номер месяца:

**12**

Введите день:

**25**

До Нового года осталось 6 дней.



# Программирование на языке Си

## Тема 8. Отладка программ

# Отладка программ

---

**Отладка** – поиск и исправление ошибок в программе.

Англ. *debugging*, *bug* = моль, жучок

## Методы:

- **трассировка** – вывод сигнальных сообщений
- **отключение части кода** (в комментариях)
- **пошаговое выполнение** – выполнить одну строчку программы и остановиться
- **точки останова** – выполнение программы останавливается при достижении отмеченных строк (переход в пошаговый режим)
- просмотр и изменение **значений переменных** в пошаговом режиме

# Трассировка

```
main()
{
    int i, X;
    printf("Введите целое число:\n");
    scanf("%d", &X);
    printf("Введено X=%d\n", X);
    for(i=1; i<10; i++)
    {
        printf("В цикле: i=%d, X=%d\n", i, X);
        ...
    }
    printf("После цикла: X=%d\n", X);
    ...
}
```

# Отключение части кода (комментарии)

```
main ()
{
  int i, X;
  printf("Введите целое число:\n");
  scanf("%d", &X);
  // X *= X + 2;
  for(i=1; i<10; i++) X *= i;
  /* while ( X > 5 ) {
     ...
  } */
  ...
}
```

комментарий до  
конца строки //

закомментированный  
блок /\* ... \*/

# Точки останова

ЛКМ или Ctrl+F5

ЭТО ТОЧКА ОСТАНОВА

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

main()
{
    char c;
    printf("Введите первую букву названия животного:\n");
}
```

**F8** – запустить и выполнить до следующей точки останова

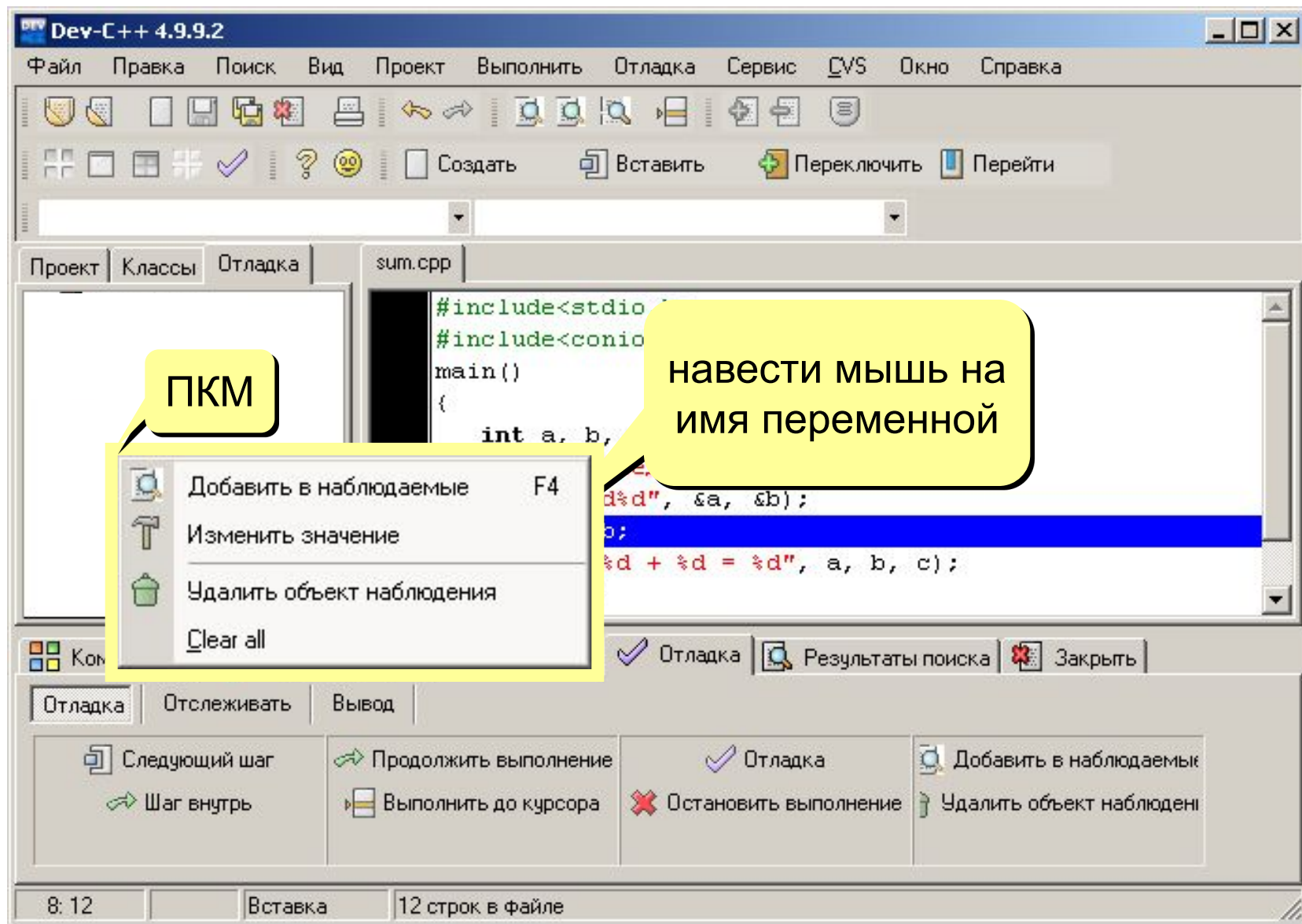
**F7** – выполнить одну строку

**Shift+F7** – войти в процедуру (функцию)

**Ctrl-F7** – выполнять дальше

**Ctrl-Alt-F2** – остановить программу

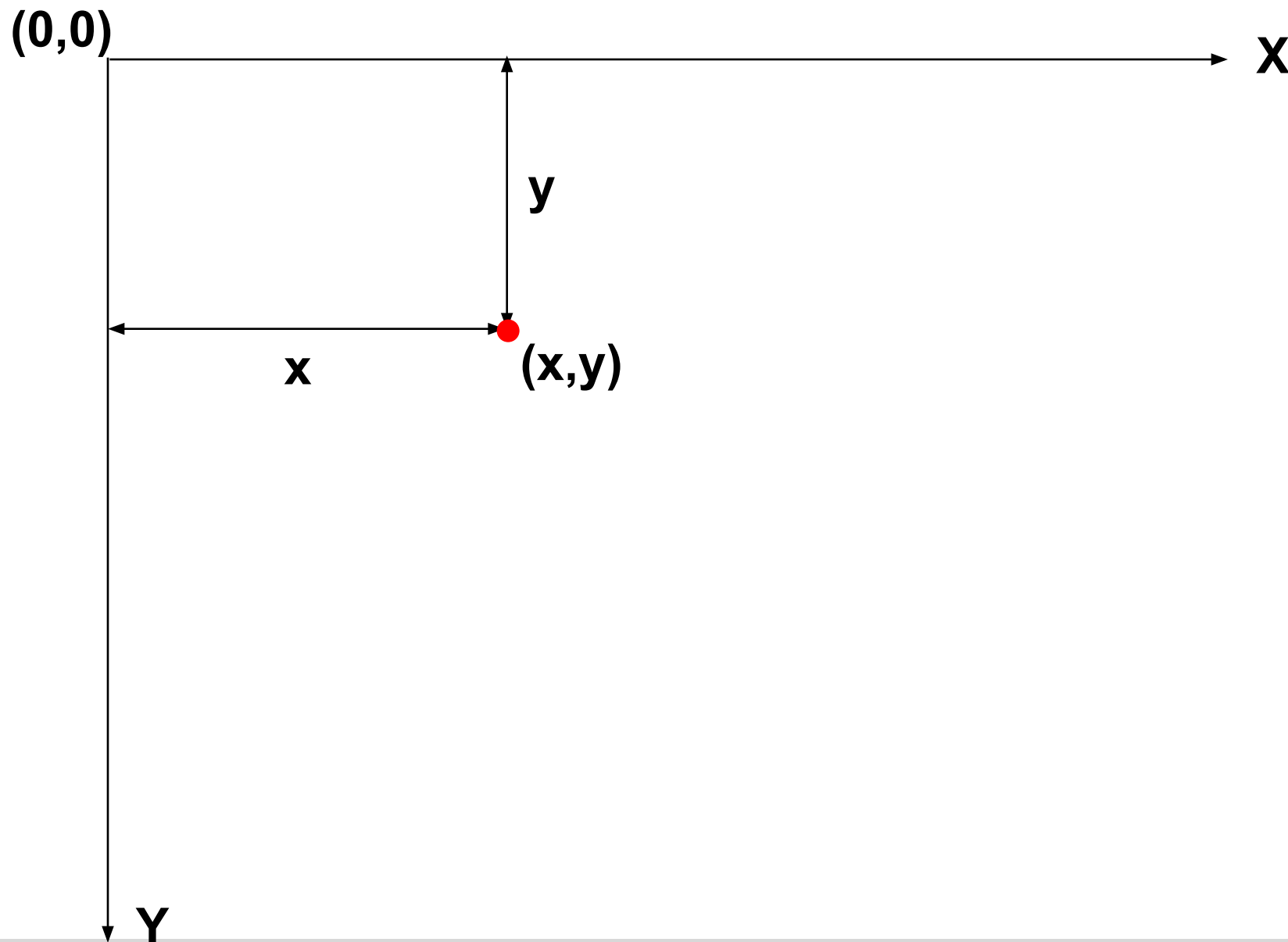
# Просмотр значений переменных



# Программирование на языке Си

## Тема 9. Графика

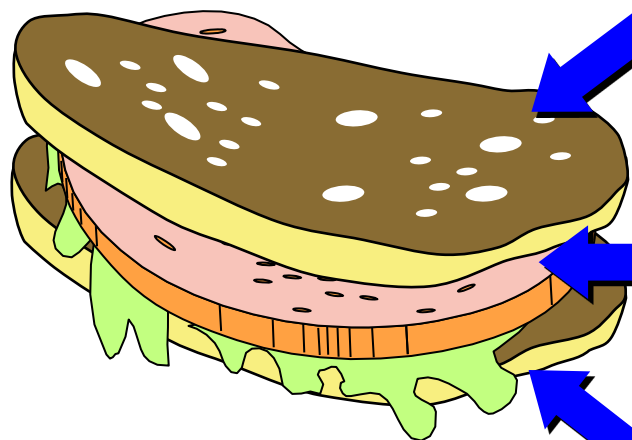
# Система координат





# Принцип сэндвича

---



открыть окно для графики

рисование в графическом  
режиме

закреть окно для графики

# Структура графической программы

```
#include <graphics.h>
```

библиотека для  
работы с графикой

```
#include <conio.h>
```

```
main()
```

```
{
```

ширина

высота

```
initwindow ( 400 , 300 );
```

открыть  
окно для  
графики

```
... // рисуем на экране
```

```
getch();
```

чтобы посмотреть результат

```
closegraph();
```

закреть  
окно

```
}
```

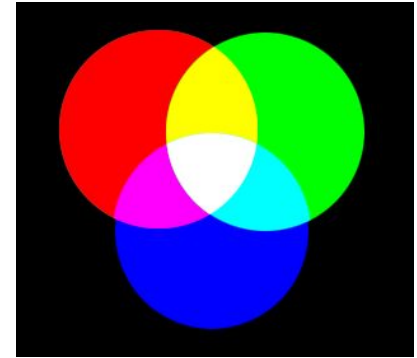
# Цвета

Код	Название
0	BLACK
1	BLUE
2	GREEN
3	CYAN
4	RED
5	MAGENTA
6	BROWN
7	LIGHTGRAY

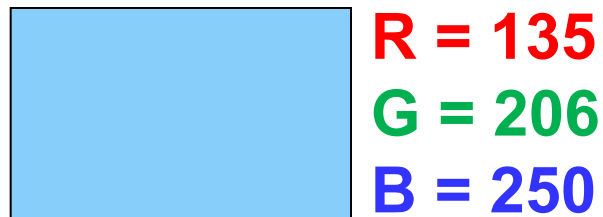
Код	Название
8	DARKGRAY
9	LIGHTBLUE
10	LIGHTGREEN
11	LIGHTCYAN
12	LIGHTRED
13	LIGHTMAGENTA
14	YELLOW
15	WHITE

# Полная палитра цветов

цвет = **R** + **G** + **B**  
*Red*            *Green*            *Blue*  
красный        зеленый            синий  
0..255           0..255            0..255



**R = 218**  
**G = 164**  
**B = 32**



**R = 135**  
**G = 206**  
**B = 250**



Сколько разных цветов?

**256·256·256 = 16 777 216 (True Color)**

# Управление цветом

## Цвет линий и текста:

*set color* = установить цвет

```
setcolor ( 12 );
```

номер  
цвета

R

G

B

```
setcolor ( COLOR(255,255,0) );
```

## Цвет и стиль заливки:

*set fill style* = установить стиль заливки

```
setfillstyle ( СТИЛЬ, ЦВЕТ );
```

0 – выключить      3..6 – наклонные линии  
1 – сплошная      7..8 – сетка  
9..11 – точечная

# Точки, отрезки и ломаные

$(x, y)$



ЦВЕТ

```
putpixel (x, y, 9);
```

$(x_1, y_1)$



$(x_2, y_2)$



```
setcolor ( 10 );  
line (x1, y1, x2, y2);
```

$(x_1, y_1)$



$(x_2, y_2)$



$(x_5, y_5)$



$(x_3, y_3)$



$(x_4, y_4)$

```
setcolor ( 12 );  
moveto (x1, y1);  
lineto (x2, y2);  
lineto (x3, y3);  
lineto (x4, y4);  
lineto (x5, y5);
```

# Прямоугольники

$(x_1, y_1)$



$(x_2, y_2)$

```
setcolor ( 9 );  
rectangle (x1, y1, x2, y2);
```

СТИЛЬ

(1 - сплошная)

ЦВЕТ

$(x_1, y_1)$



$(x_2, y_2)$

```
setfillstyle ( 1, 12 );  
bar (x1, y1, x2, y2);
```

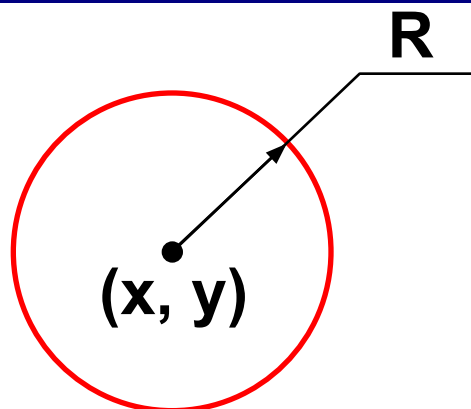
$(x_1, y_1)$



$(x_2, y_2)$

```
setfillstyle ( 1, 12 );  
bar (x1, y1, x2, y2);  
setcolor ( 9 );  
rectangle (x1, y1, x2, y2);
```

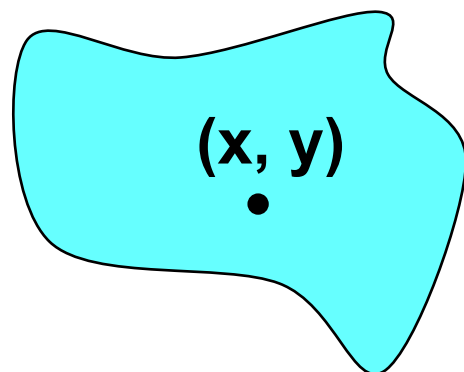
# Окружность, заливка, текст



```
setcolor ( COLOR(255,0,0) );  
circle ( x, y, R );
```

**СТИЛЬ**  
(1 - сплошная)

**ЦВЕТ**  
заливки



```
setfillstyle ( 1, 11 );  
floodfill ( x, y, 0 );
```

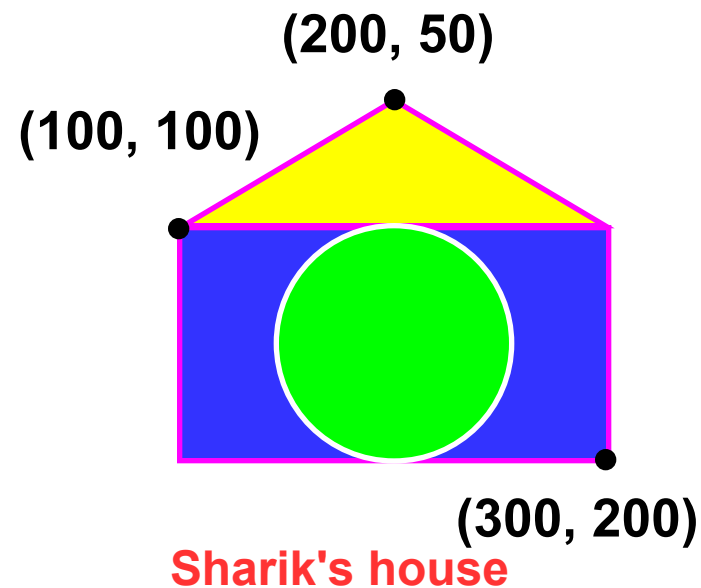
**цвет границы**

$(x, y)$   
•  
**Вася**

```
setcolor ( 9 );  
outtextxy ( x, y, "Вася" );
```



# Пример

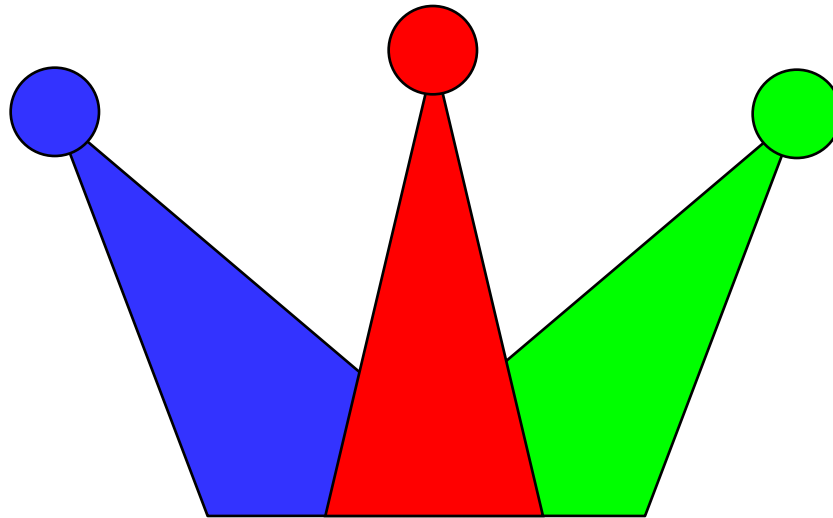


```
setfillstyle (1, 9);  
bar (100,100,300,200);  
setcolor (13);  
rectangle (100,100,300,200);  
moveto (100,100);  
lineto (200, 50);  
lineto (300,100);  
setfillstyle (1, 14);  
floodfill (200, 75, 13);  
setcolor (15);  
circle (200, 150,50);  
setfillstyle (1, 10);  
floodfill (200,150, 15);  
setcolor (12);  
outtextxy (100, 230,  
           "Sharik's house.");
```

# Задания

---

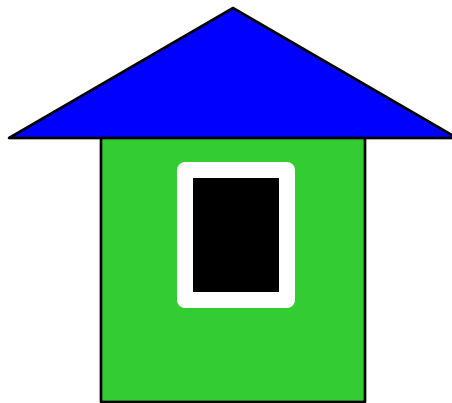
## «5»: «Корона»



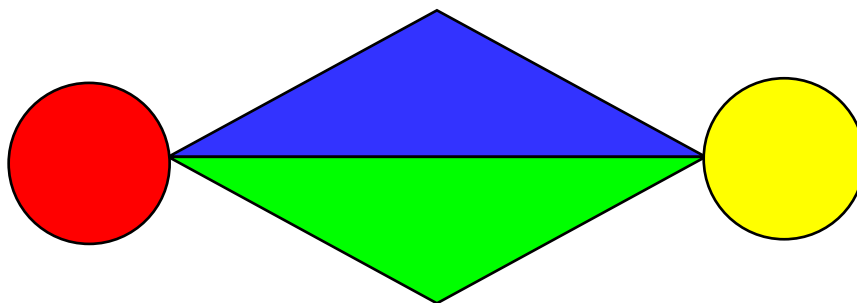
# Задания

---

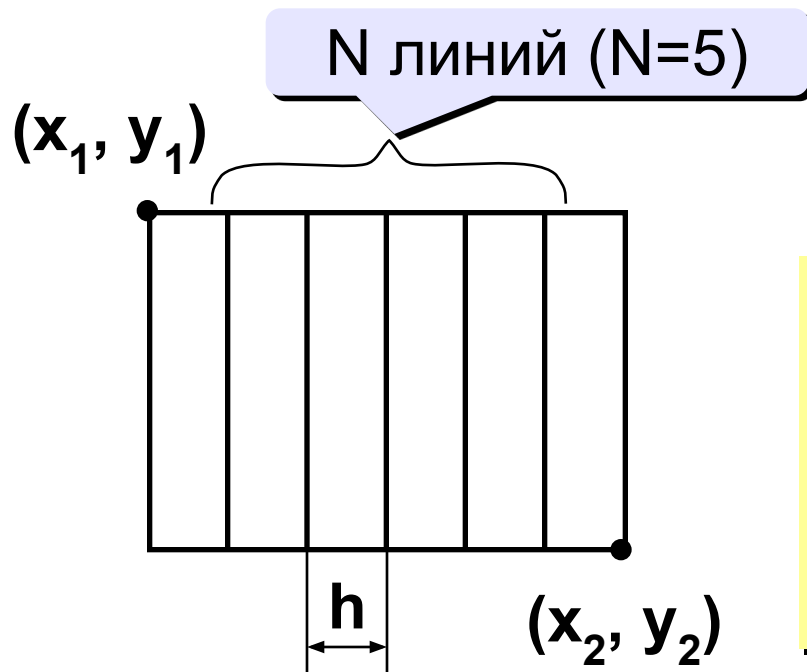
«3»: «Домик»



«4»: «Лягушка»



# Штриховка



$$h = \frac{x_2 - x_1}{N + 1}$$

```
rectangle (x1, y1, x2, y2);
line (x1+h, y1, x1+h, y2);
line (x1+2*h, y1, x1+2*h, y2);
line (x1+3*h, y1, x1+3*h, y2);
...
```

x

x

```
rectangle (x1, y1, x2, y2);
h = (x2 - x1) / (N + 1.);
for (x = x1+h; x < x2; x += h)
    line (x, y1, x, y2);
```

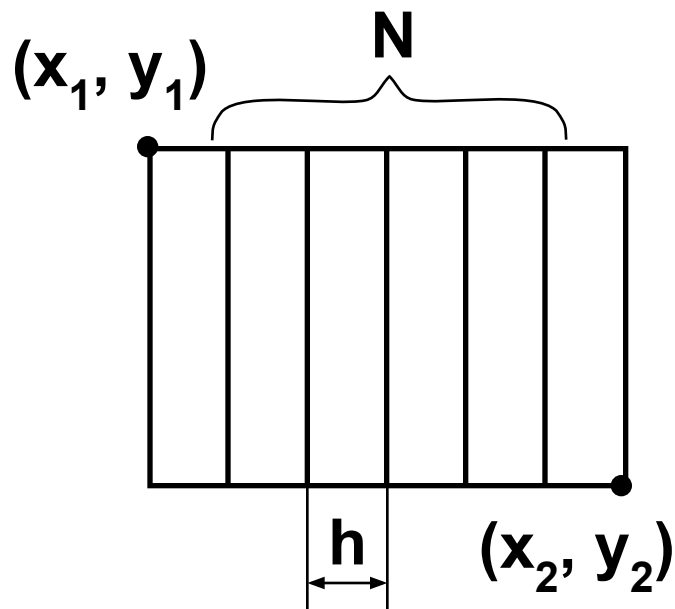
результат –  
дробное число



float x, h;

дробная часть x  
отбрасывается

# Штриховка (программа)

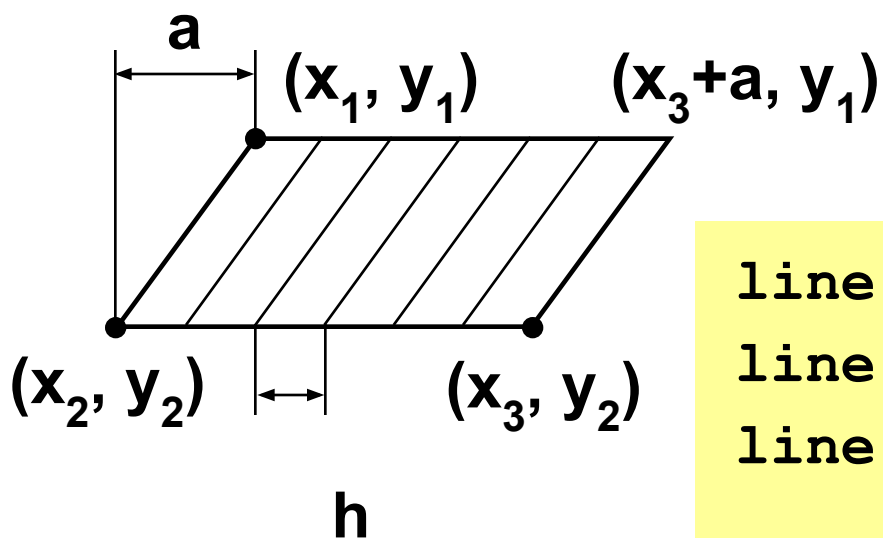


прямоугольник

штриховка

```
#include <graphics.h>
#include <conio.h>
main()
{
    int N = 10, x1 = 100,
        x2 = 300, y1 = 100,
        y2 = 200;
    float h, x;
    initwindow(800, 600);
    rectangle(x1, y1, x2, y2);
    h = (x2 - x1) / (N + 1.);
    for (x = x1 + h; x < x2; x += h)
        line(x, y1, x, y2);
    getch();
    closegraph();
}
```

# Штриховка



$$a = x_1 - x_2$$

$$h = \frac{x_3 - x_2}{N + 1}$$

```
line (x1+h, y1, x1+h-a, y2);
line (x1+2*h, y1, x1+2*h-a, y2);
line (x1+3*h, y1, x1+3*h-a, y2);
...
```

$x$

$x-a$

```
h = (x3 - x2) / (N + 1.);
```

```
a = x2 - x1;
```

```
x = x1 + h;
```

```
for (i = 1; i <= N; i++, x += h)
```

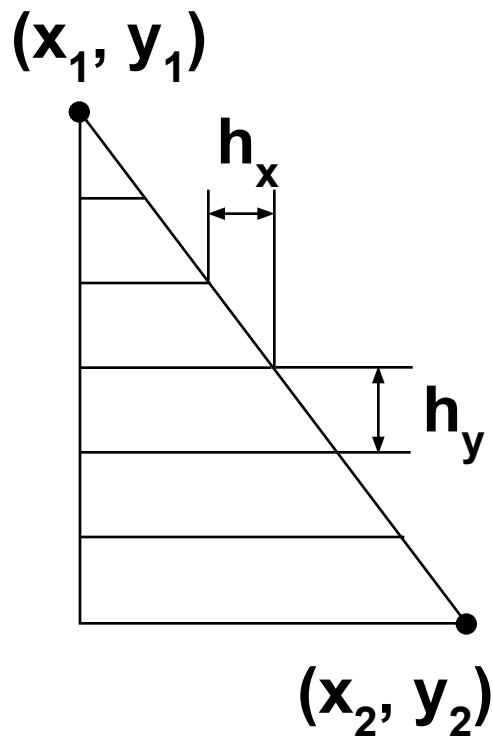
```
    line(x, y1, x-a, y2);
```

после каждого шага выполняются две команды



Плюсы и минусы?

# Штриховка



$$h_x = \frac{x_2 - x_1}{N + 1}$$

$$h_y = \frac{y_2 - y_1}{N + 1}$$

```

line ( x1, y1+hy, x1+hx, y1+hy );
line ( x1, y1+2*hy, x1+2*hx, y1+2*hy );
line ( x1, y1+3*hy, x1+3*hx, y1+3*hy );
...

```

y

x

y

```

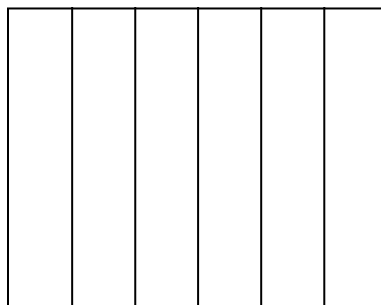
hx = (x2 - x1) / (N + 1.);
hy = (y2 - y1) / (N + 1.);
x = x1 + hx; y = y1 + hy;
for (i=1; i <= N; i++) {
    line ( x1, y, x, y );
    x += hx; y += hy;
}

```

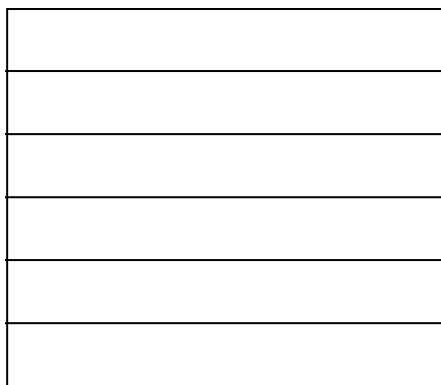
# Задания

---

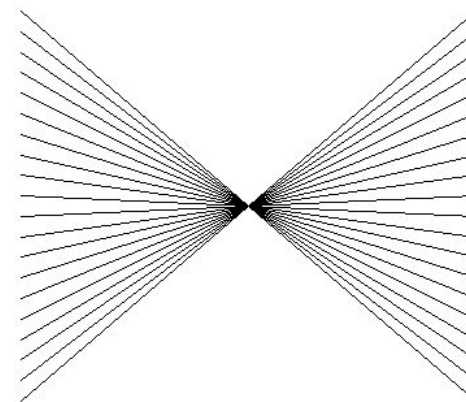
**«3»:** Ввести с клавиатуры количество линий, построить фигуру и выполнить штриховку:



**«4»:** Ввести с клавиатуры количество линий, построить фигуру и выполнить штриховку:



**или**

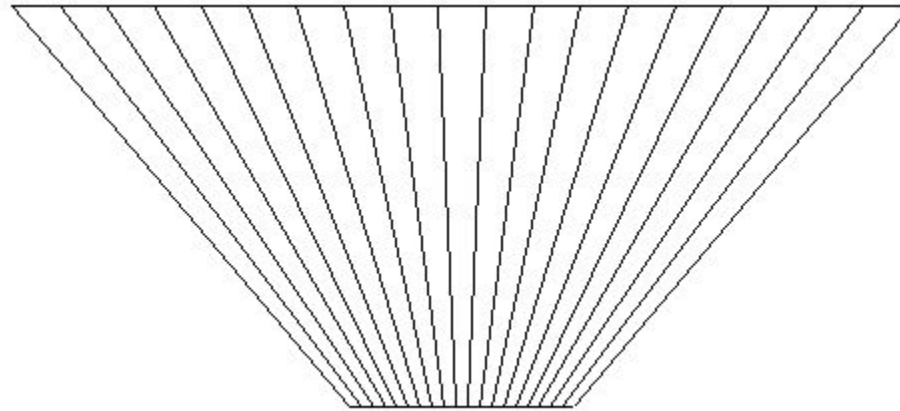




# Задание

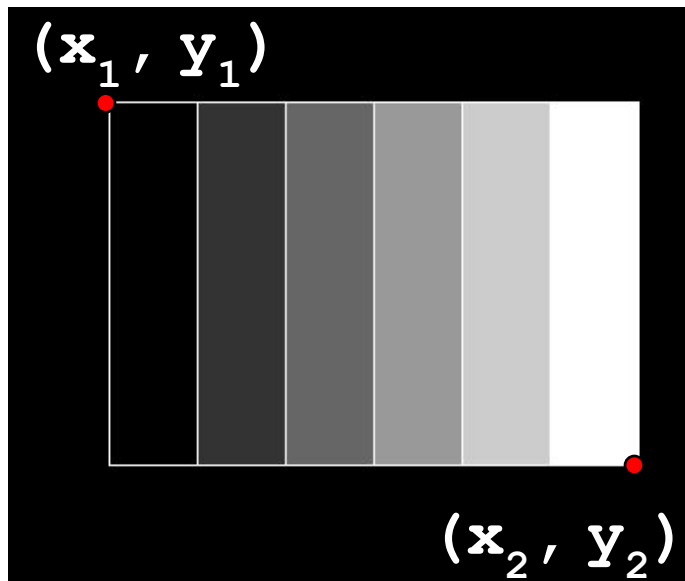
---

«5»: Ввести с клавиатуры количество линий и построить фигуру:



# Как менять цвет?

серый:  $R = G = B$



Цвет: `COLOR (c, c, c)`

Изменение  $c$ :  $0, \dots, 255$   
 $N$

Шаг изменения  $c$ :

$$h_c = \frac{255}{N}$$

```

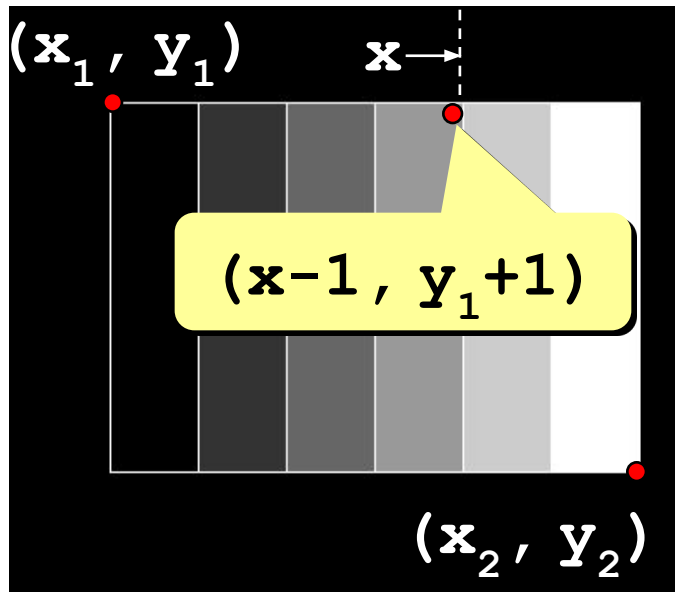
hc = 255 / N;
c = 0;
for ( i=1; i<=N+1; i++ ) {
    setfillstyle ( 1, COLOR(c,c,c) );
    floodfill( ???, ???, 15 );
    c += hc;
}

```

ЦВЕТ  
границы

# Как менять цвет?

```
setfillstyle( 1, COLOR(с,с,с) );
floodfill ( ???, ???, 15 );
```



```
hc = 255 / N;
```

```
с = 0;
```

```
x = x1 + h;
```

```
for ( i=1; i <= N+1; i++ ) {
```

```
  setfillstyle(1, COLOR(с,с,с) );
```

```
  floodfill ( x-1, y1+1, 15 );
```

```
  x += h;
```

```
  с += hc;
```

```
}
```

правая  
граница  
полосы

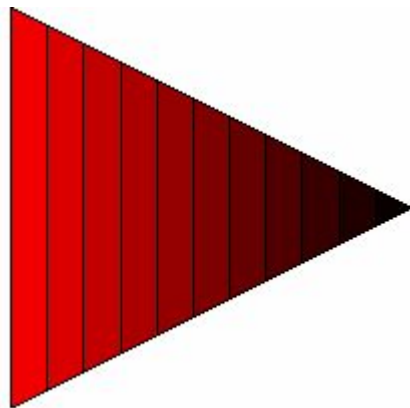


Как объединить циклы штриховки и заливки?

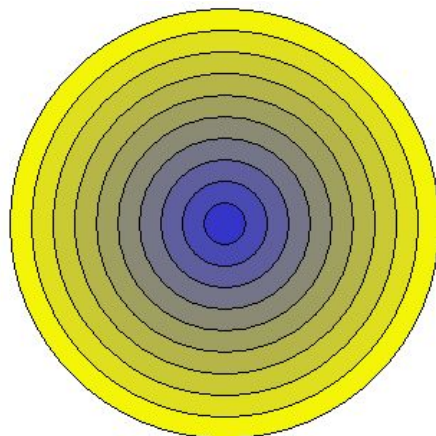
# Задания

---

**«4»:** Ввести с клавиатуры число линий штриховки и построить фигуру, залив все области разным цветом.



**«5»:** Ввести с клавиатуры число окружностей и построить фигуру, залив все области разным цветом.



# Программирование на языке Си

## **Тема 10. Графики функций** (только с 9 класса)

# Построение графиков функций

---

**Задача:** построить график функции  $y = 3 \sin(x)$  на интервале от 0 до  $2\pi$ .

**Анализ:**

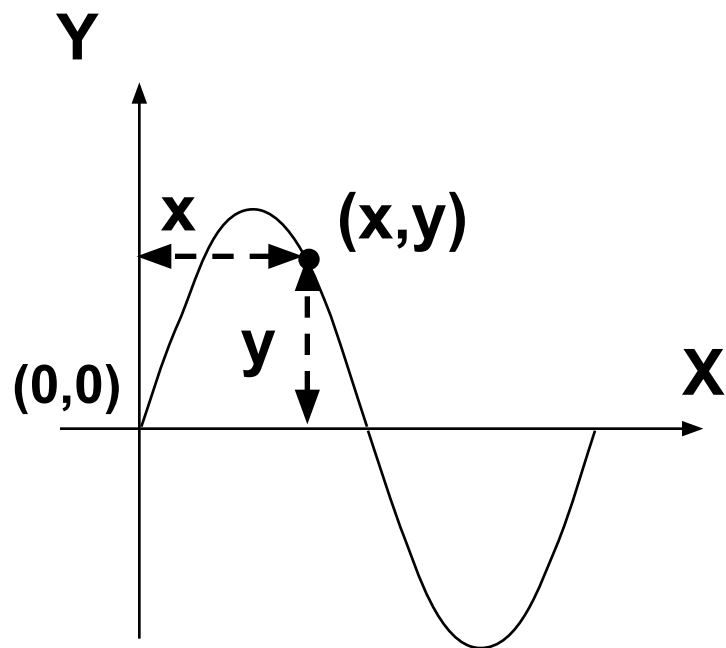
максимальное значение  $y_{\max} = 3$  при  $x = \pi/2$

минимальное значение  $y_{\min} = -3$  при  $x = 3\pi/2$

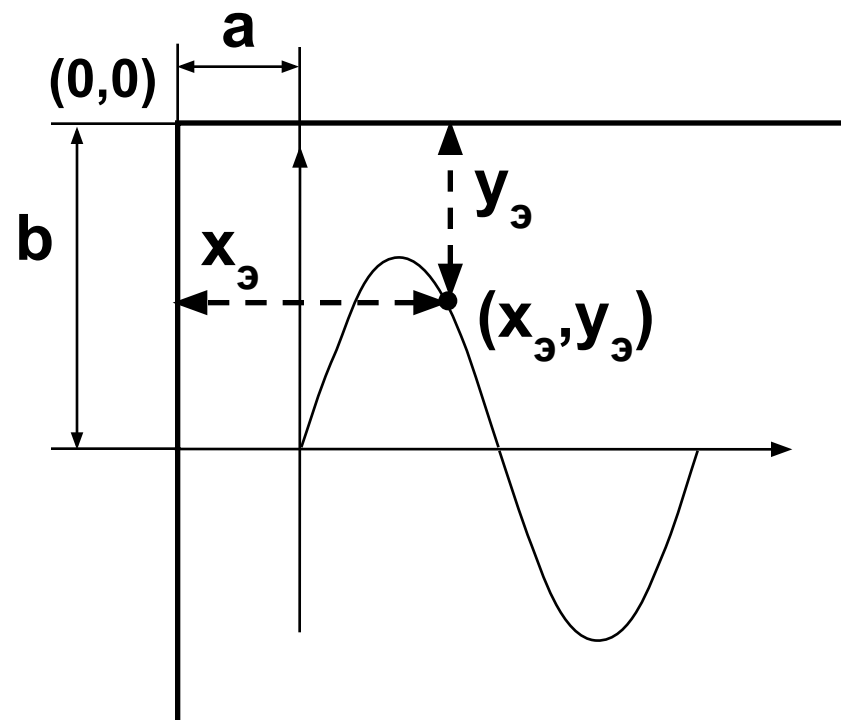
**Проблема:** функция задана в математической системе координат, строить надо на экране, указывая координаты в пикселях.

# Преобразование координат

Математическая  
система координат



Экранная система  
координат (пиксели)



**k** – масштаб (длина  
изображения единичного  
отрезка на экране)

$$x_{\text{э}} = a + kx$$

$$y_{\text{э}} = b - ky$$

# Программа

```
const a = 50, b = 200, k = 50;
const float xmin = 0, xmax = 2*M_PI;
float x, y, h = 0.01;
int xe, ye, w;
w = (xmax - xmin)*k;
line(a-10, b, a+w, b);
line(a, 0, a, 2*b);
for (x = xmin; x < xmax; x += h)
{
    y = 3*sin(x);
    xe = a + k*x;
    ye = b - k*y;
    putpixel (xe, ye, 12);
}
```

2π

h – шаг изменения x

w – длина оси OX в пикселях

оси координат

обязательно

#include &lt;math.h&gt;

координаты точки на  
экране

Что плохо?



# Как соединить точки?

## Алгоритм:

Если первая точка  
перейти в точку  $(x_э, y_э)$   
иначе  
отрезок в точку  $(x_э, y_э)$

выбор  
варианта  
действий

## Программа:

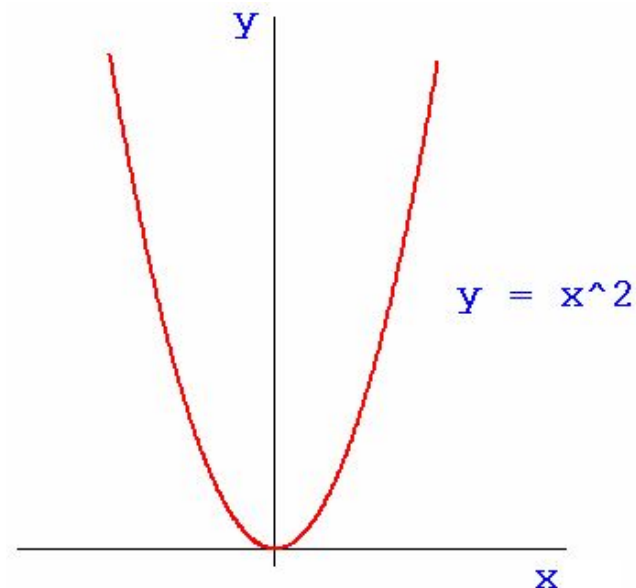
```
int first;  
...  
first = 1;  
for (x = xmin; x < xmax; x += h)  
{  
    ...  
    if ( first ) {  
        moveto(xe, ye);  
        first = 0;  
    }  
    else lineto(xe, ye);  
    ...  
}
```

переменная-  
флаг (1 или 0)

начальное значение

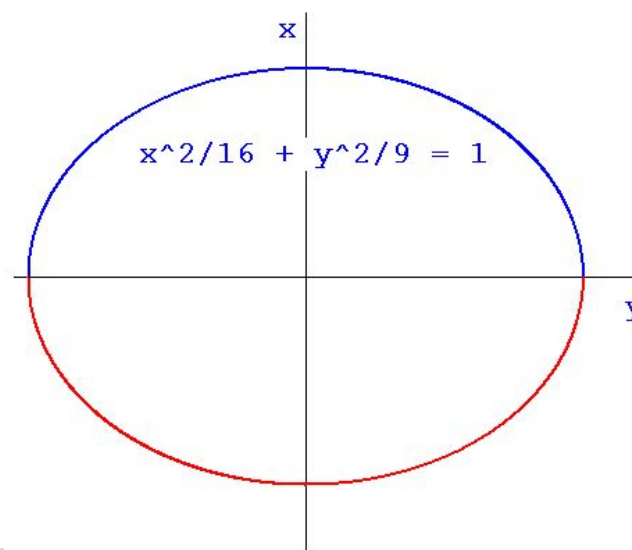
# Задания

**«4»:** Построить график функции  $y = x^2$  на интервале  $[-3,3]$ .



**«5»:** Построить график функции (эллипс)

$$\frac{x^2}{16} + \frac{y^2}{9} = 1$$



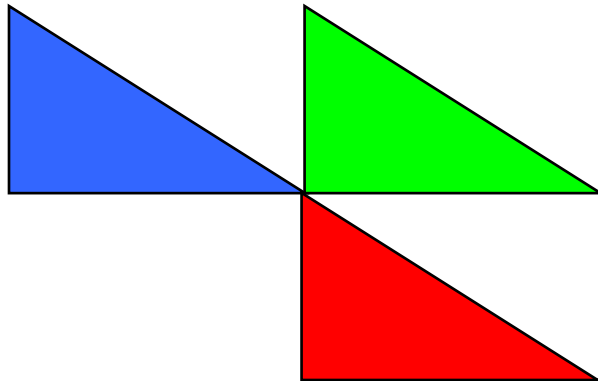
# Программирование на языке Си

## Тема 11. Процедуры

# Процедуры

---

**Задача:** Построить фигуру:



**Можно ли решить известными методами?**

**Общность:** три похожие фигуры.

**общее:** размеры, угол поворота

**отличия:** координаты, цвет



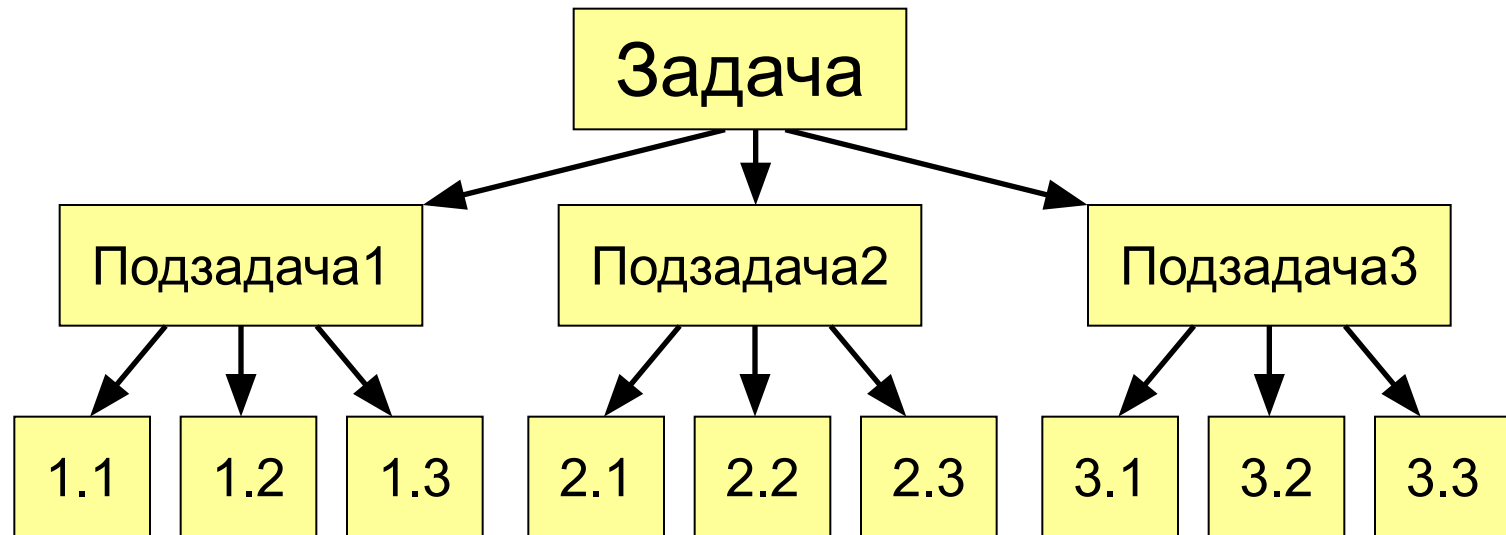
**Сколько координат надо задать?**

# Процедуры

**Процедура** – это вспомогательный алгоритм, который предназначен для выполнения некоторых действий.

## Применение:

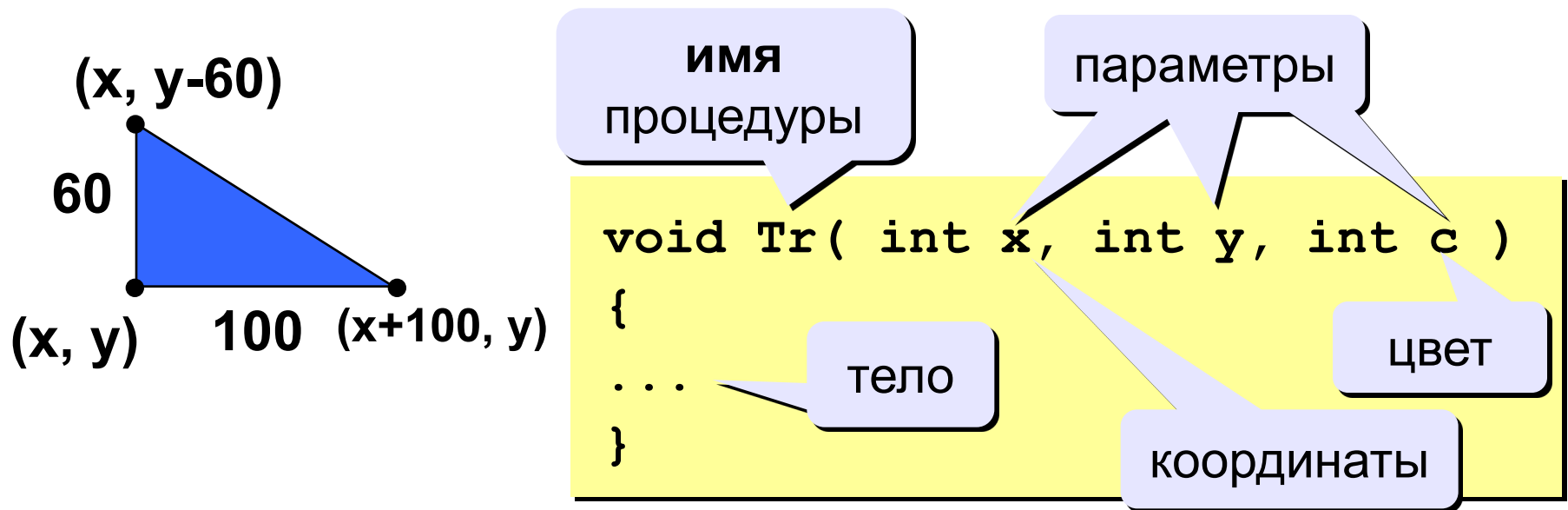
- выполнение одинаковых действий в разных местах программы
- разбивка программы (или другой процедуры) на подзадачи для лучшего восприятия



# Процедуры

## Порядок разработки:

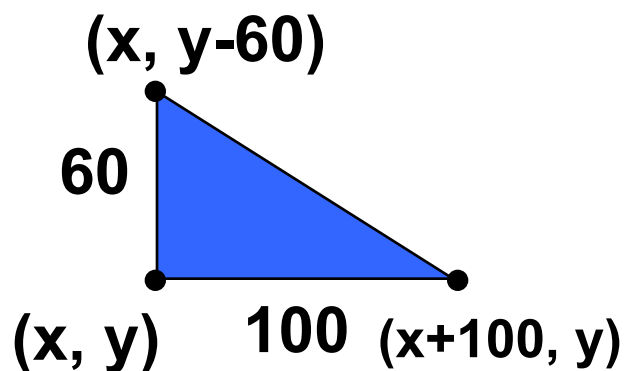
- выделить одинаковое или похожее (*три фигуры*)
- найти в них **общее** (размеры, форма, угол поворота) и **отличия** (координаты, цвет)
- отличия записать в виде неизвестных переменных, они будут **параметрами** процедуры



`void` – «пустой» (некоторые действия)

# Процедуры

формальные  
параметры

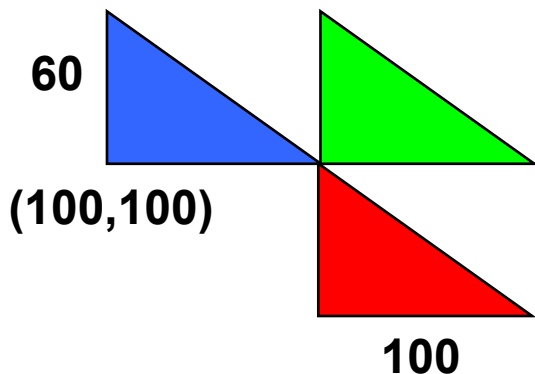


тело процедуры

```
void Tr( int x, int y, int c )  
{  
    moveto ( x, y );  
    lineto ( x, y-60 );  
    lineto ( x+100, y );  
    lineto ( x, y );  
    setfillstyle ( 1, c );  
    floodfill ( x+20, y-20, 15 );  
}
```

«**Формальные параметры**» могут изменяться, заранее неизвестны (обозначаются именами, как переменные).

# Программа



ВЫЗОВЫ  
процедуры

```
#include <conio.h>
#include <graphics.h>

void Tr( int x, int y, int c)
{
    ...
}

main()
{
    initwindow (400, 300);
    Tr (100, 100, COLOR(0,0,255));
    Tr (200, 100, COLOR(0,255,0));
    Tr (200, 160, COLOR(255,0,0));
    getch();
    closegraph();
}
```

формальные  
параметры

процедура

фактические  
параметры



# Процедуры

---

## Особенности:

- *обычно* процедуры расположены **выше** основной программы
- в заголовке процедуры перечисляются **формальные** параметры, они обозначаются именами, поскольку могут меняться

```
void Tr( int x, int y, int c )
```

- при вызове процедуры в скобках указывают **фактические** параметры (числа или арифметические выражения) **в том же порядке**

```
Tr ( 200, 100, COLOR(255,0,0) );
```

**x**

**y**

**c**

# Процедуры

## Особенности:

- для каждого формального параметра в заголовке процедуры указывают его **тип**

```
void A ( int x, float y, char z ) { ... }
```

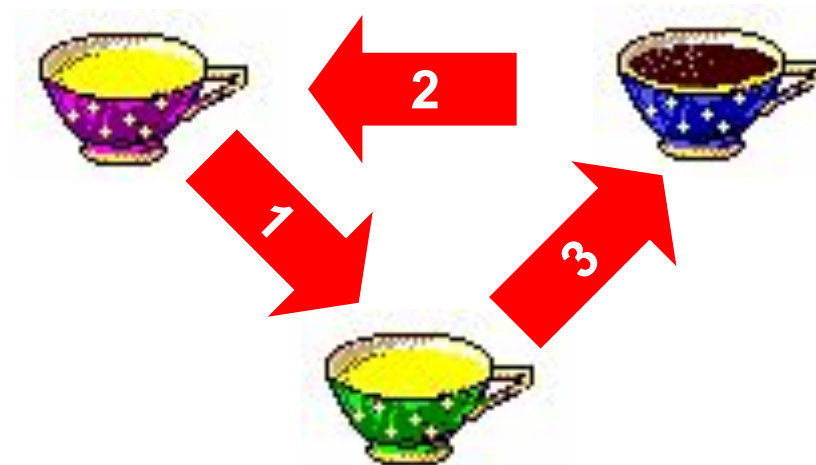
- внутри процедуры параметры используются так же, как и переменные
- в процедуре можно объявлять дополнительные **локальные переменные**, остальные процедуры не имеют к ним доступа

```
void A ( int x, float y, char z )  
{  
  int a2, bbc =  
    345;  
  ...  
}
```

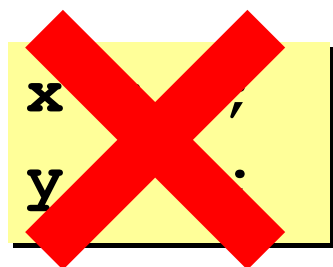
локальные  
переменные

# Как поменять местами?

**Задача:** поменять местами содержимое двух чашек.



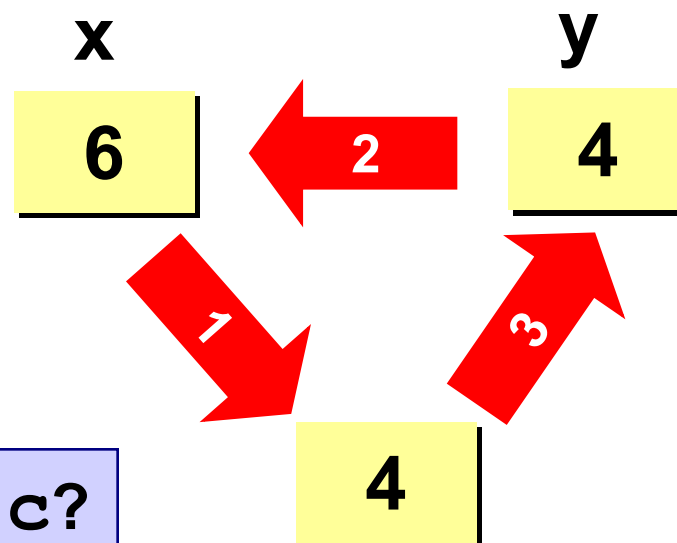
**Задача:** поменять местами содержимое двух ячеек памяти.



```

c = x;
x = y;
y = c;

```



Можно ли обойтись без c?

# Параметры-переменные

**Задача:** составить процедуру, которая меняет местами значения двух переменных.

**Особенности:** надо, чтобы изменения, сделанные в процедуре, стали известны вызывающей программе.

```
void Swap ( int a, int b )  
{  
  int c;  
  c = a; a = b, b = c;  
}
```

эта процедура  
работает с  
**КОПИЯМИ**  
параметров

```
main ()
```

```
{
```

```
  int x = 1, y = 2;
```

```
  Swap ( x, y );
```

```
  printf ( "x = %d, y = %d", x, y );
```

```
}
```

**x = 1, y = 2**

# Параметры-переменные

```
void Swap ( int &a, int &b )  
{  
    int c;  
    c = a; a = b; b = c;  
}
```

параметры могут  
изменяться

## Применение:

таким образом процедура (и функция) может возвращать несколько значений

## Запрещенные варианты вызова

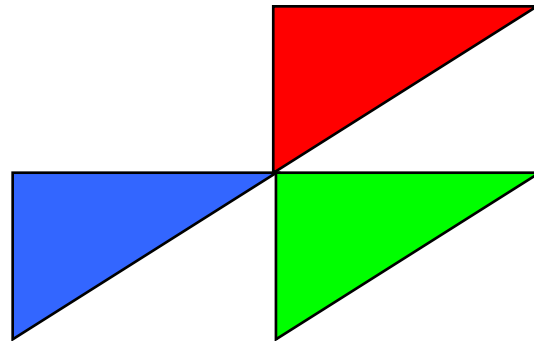
```
Swap ( 2, 3 ); // числа
```

```
Swap ( x+z, y+2 ); // выражения
```

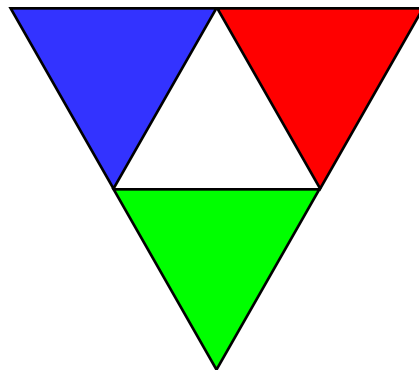
# Задания

---

«3»: Используя одну процедуру, построить фигуру.



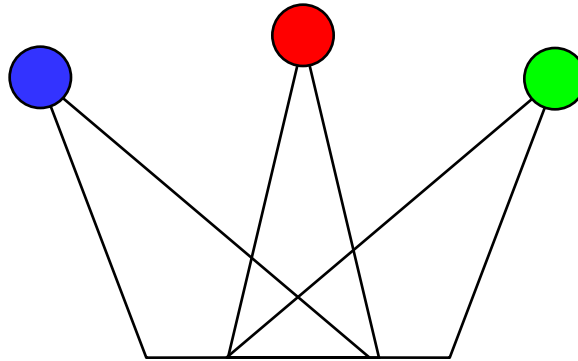
«4»: Используя одну процедуру, построить фигуру.



# Задания

---

«5»: Используя одну процедуру, построить фигуру.



# Программирование на языке Си

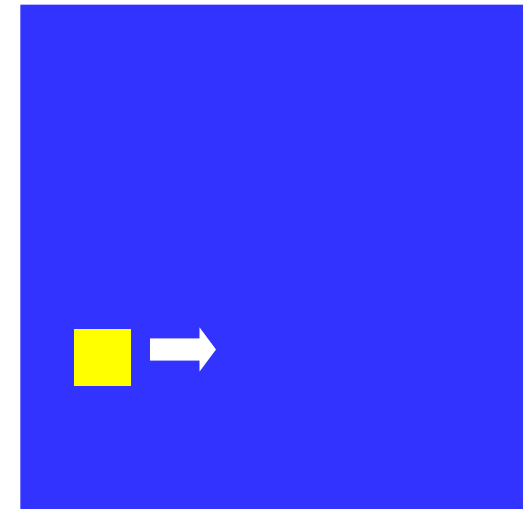
## Тема 12. Анимация



# Анимация

**Анимация** (англ. *animation*) – оживление изображения на экране.

**Задача:** внутри синего квадрата 400 на 400 пикселей слева направо движется желтый квадрат 20 на 20 пикселей. Программа останавливается, если нажата клавиша **Esc** или квадрат дошел до границы синей области.



**Проблема:** как изобразить перемещение объекта на экране?

**Привязка:** состояние объекта задается координатами **(x,y)**

**Принцип анимации:**

1. рисуем объект в точке **(x,y)**
2. задержка на несколько миллисекунд
3. стираем объект
4. изменяем координаты **(x,y)**
5. переходим к шагу 1

## Как «поймать» нажатие клавиши?

`kbhit()` – функция, определяет, было ли нажатие на (любую!) клавишу (**0** – не было, **не 0** – было).

```
if ( kbhit() )  
    printf("Нажата какая-то клавиша...");  
else printf("Нет нажатия...");
```

`if ( kbhit() != 0 )`

`getch()` – функция, которая определяет код нажатой клавиши: **27** = *Esc*, **13** = *Enter*, **32** = пробел, ...

```
if ( kbhit() ) {  
    printf("Нажата какая-то клавиша...");  
    c = getch();  
    printf("Код клавиши %d", c);  
}
```

`int c;`

# Как выйти из цикла?



Как не допустить выход за границу поля?

$x + 20 < 400$

для `kbhit()` и `getch()`

```
#include <conio.h>
main()
{
    ...
    while ( x + 20 < 400 )
    {
        if ( kbhit() )
            if ( getch() == 27 ) break;
        ...
    }
    ...
}
```

пока не вышли за границу синего квадрата

если нажата клавиша ...

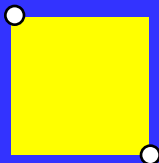
если нажата клавиша  
с кодом 27 (*Esc*),  
выйти из цикла

# Процедура (рисование и стирание)

## Идеи

- одна процедура рисует и стирает
- стереть = нарисовать цветом фона
- границу квадрата отключить (в основной программе)

(x, y)



(x+20, y+20)

цвет: желтым рисуем,  
синим стираем

```
void Draw( int x, int y, int color )  
{  
  setfillstyle ( 1, color );  
  bar ( x, y, x+20, y+20 );  
}
```

сплошная заливка  
цветом color

залитый  
прямоугольник

# Полная программа

```
#include <conio.h>
#include <graphics.h>
void Draw ( int x, int y, int color )
{
    ...
}
```

процедура

```
main()
{
    int x, y;
    initwindow (500, 500);
    setfillstyle (1, COLOR (0, 0, 255));
    bar (0, 0, 399, 399);
    x = 0; y = 240;
    /* анимация */
    closegraph ();
}
```

синий фон

начальные  
координаты

# Цикл анимации

пока не вышли из  
синего квадрата

```
while ( x + 20 < 400 )  
{  
  if ( kbhit() )  
    if ( getch() == 27 ) break;  
  Draw ( x, y, COLOR(255,255,0) );  
  delay ( 20 );  
  Draw ( x, y, COLOR(0,0,255) );  
  x ++;  
}
```

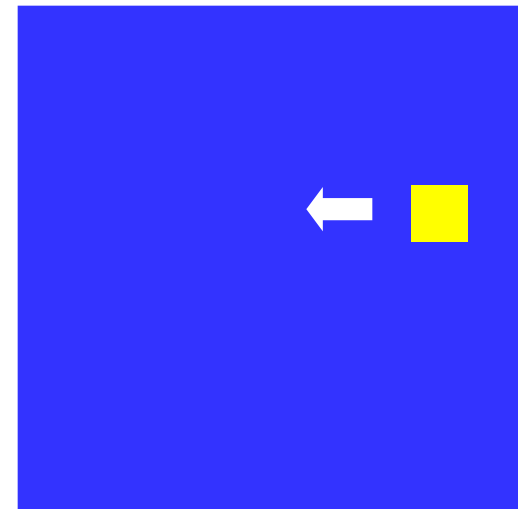
ВЫХОД ПО  
клавише *Esc*

ждем 20 мс

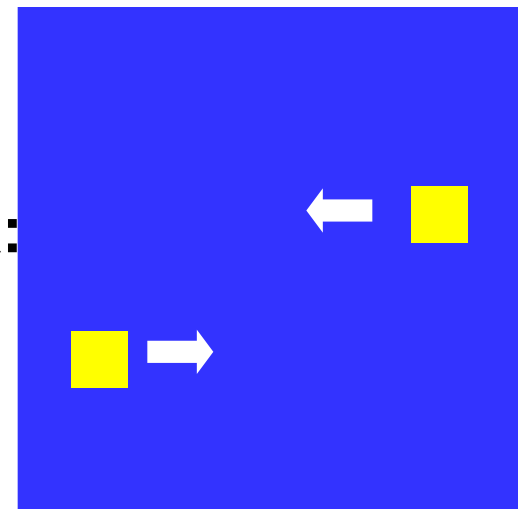
# Задания

---

**«3»:** Квадрат двигается справа налево:



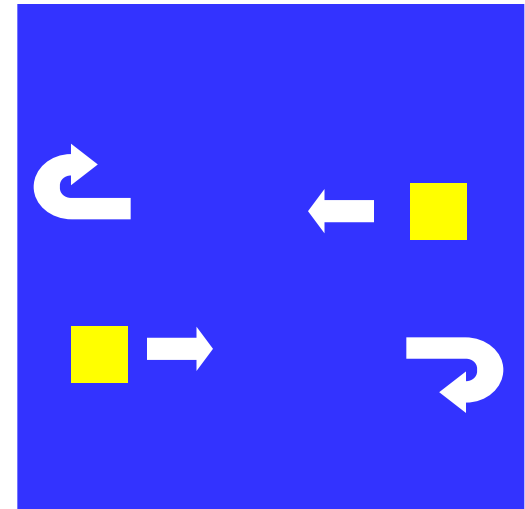
**«4»:** Два квадрата двигаются в противоположных направлениях:



# Задания

---

**«5»:** Два квадрата двигаются в противоположных направлениях и отталкиваются от стенок синего квадрата:





# Управление клавишами

**Задача:** жёлтый квадрат внутри синего квадрата управляется клавишами-стрелками. Коды клавиш:

влево – **75**

вверх – **72**

Esc – **27**

вправо – **77**

вниз – **80**

**Проблема:** как изменять направление

если было нажатие на клавишу, ...

**Решение:**

получить  
код  
клавиши

```
if ( kbhit() ) {  
    code = getch();  
    if (code == 27) break;  
    switch ( code ) {  
        case 75: x --; break;  
        case 77: x ++; break;  
        case 72: y --; break;  
        case 80: y ++;  
    }  
}
```

выход по Esc

перемещение

# Программа

```
void Draw (int x, int y, int color)
{
    ...
}
```

процедура

```
main()
{
    int x, y, code;
    ...
```

ОСНОВНОЙ ЦИКЛ

```
while ( 1 ) {
    Draw(x, y, COLOR(255,255,0));
    delay(20);
    Draw(x, y, COLOR(0,0,255));
    if ( kbhit() ) {
        ...
    }
}
```

обработка  
нажатия на  
клавишу



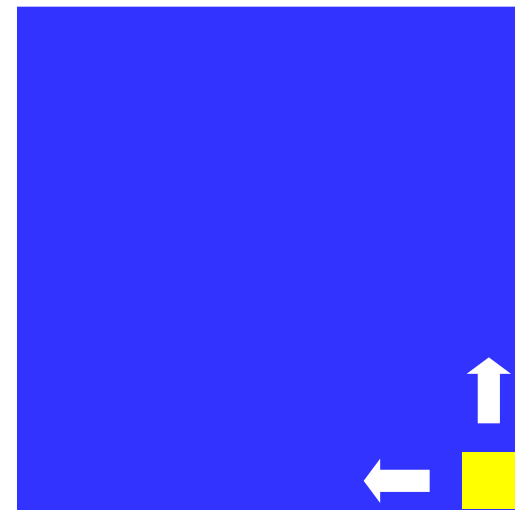
Что плохо?



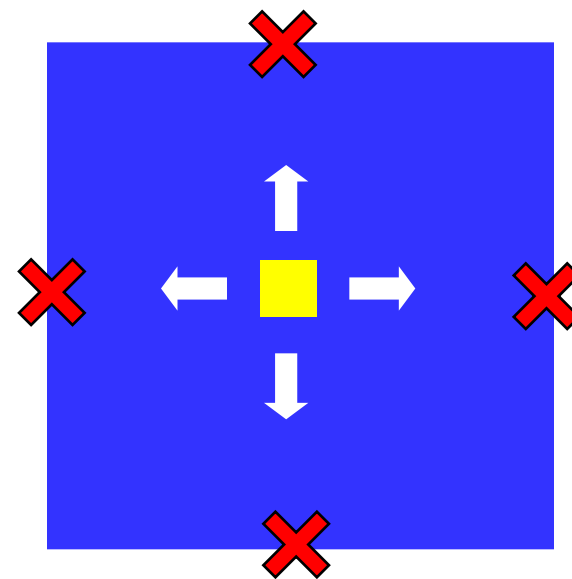
Как убрать мигание?

# Задания

**«3»:** Квадрат в самом начале стоит в правом нижнем углу, и двигается при нажатии стрелок только вверх или влево:



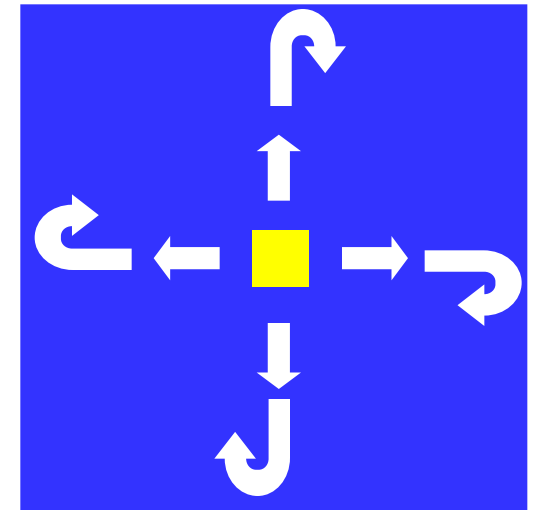
**«4»:** Квадрат двигается при нажатии стрелок, однако не может выйти за границы синего квадрата:



# Задания

---

**«5»:** Квадрат непрерывно двигается, при нажатии стрелок меняет направление и отталкивается от стенок синего квадрата:

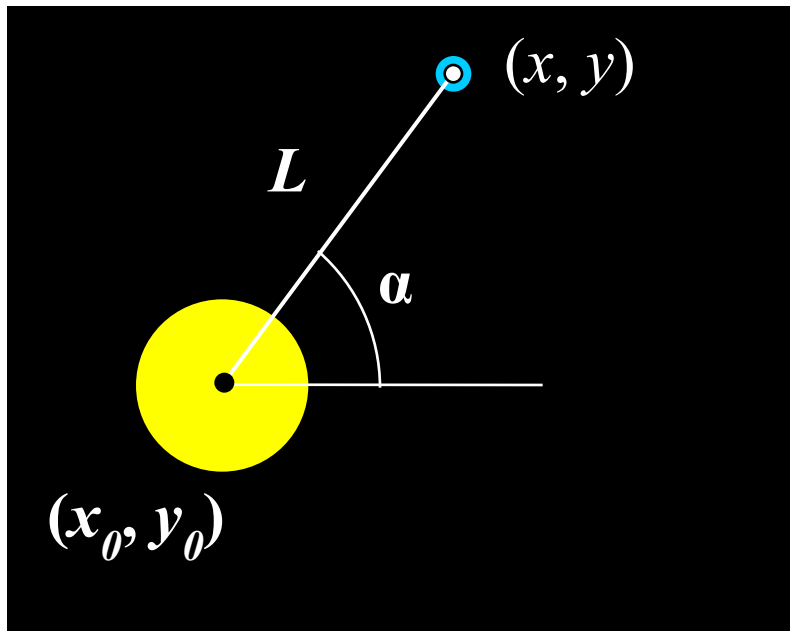


# Вращение (для 8-11 класса)

**Задача:** изобразить модель вращения Земли вокруг Солнца.

**Проблема:** движение по окружности, как изменять координаты?

**Решение:** использовать в качестве независимой переменной (менять в цикле) угол поворота  $\alpha$



$$x = x_0 + L \cdot \cos(\alpha)$$

$$y = y_0 - L \cdot \sin(\alpha)$$

# Процедура

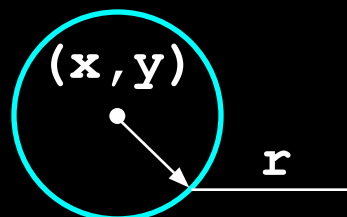
цвет: желтый – рисуем,  
черный – стираем

```
void Draw( int x, int y, int color )  
{  
  const int r = 10;  
  setcolor ( color );  
  circle ( x, y, r );  
}
```

постоянная

радиус Земли

установили  
цвет линий



# Константы и переменные

```
#include <math.h>    // математические функции

void Draw ( int x, int y, int color )
{
    ...
}

main()
{
    const int
        rSun = 60,    // радиус Солнца
        L   = 150,    // радиус орбиты Земли
        x0  = 200,    // координаты центра Солнца
        y0  = 200;

    int    x, y,      // координаты Земли
           code;      // код нажатой клавиши
    float  a, ha;     // угол поворота, шаг
    initwindow( 500, 500 );
    ...
}
```

# ОСНОВНОЙ ЦИКЛ

рисует Солнце:  
белый контур,  
желтая заливка

```
circle ( x0, y0, rSun );  
setfillstyle(1, COLOR(255,255,0));  
floodfill(x0, y0, COLOR(255,255,255));  
a = 0; // начальный угол  
ha = M_PI/180; // шаг 1° за 20 мс
```

```
while(1) {  
    x = x0 + L*cos(a);  
    y = y0 - L*sin(a);  
    Draw ( x, y, COLOR(0,255,255) );  
    delay ( 20 );  
    Draw(x, y, 0);  
    if ( kbhit() )  
        if ( 27 == getch() ) break;  
    a = a + ha;  
}
```

НОВЫЕ  
КОординаты

ждем 20 мс

ВЫХОД по Esc

поворот на ha

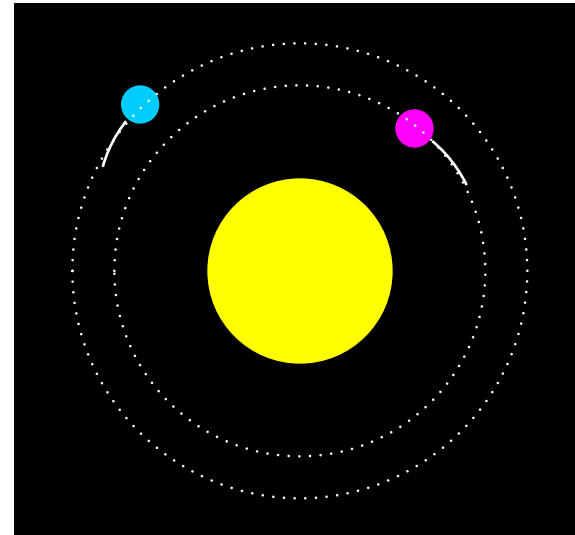
```
closegraph(); #include<math.h> // sin, cos, M_PI
```



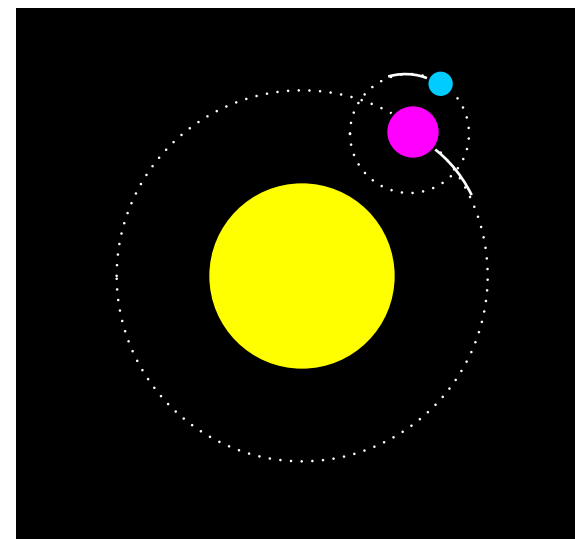


# Задания

**«4»:** Изобразить модель Солнца с двумя планетами, которые вращаются в противоположные стороны:



**«5»:** Изобразить модель системы Солнце-Земля-Луна:



# Программирование на языке Си

## Тема 13. Функции

# Функции

---

**Функция** – это вспомогательный алгоритм (подпрограмма), результатом работы которого является некоторое значение.

## Примеры:

- вычисление модуля числа,  $\sqrt{x}$
- расчет значений по сложным формулам
- ответ на вопрос (простое число или нет?)

## Зачем?

- для выполнения одинаковых расчетов в различных местах программы
- для создания общедоступных библиотек функций



В чем отличие от процедур?

# Функции

**Задача:** составить функцию, которая вычисляет наибольшее из двух значений, и привести пример ее использования

**Функция:**

тип  
результата

формальные  
параметры

return - вернуть  
результат функции

```
int Max ( int a, int b )  
{  
    if ( a > b ) return a ;  
    else      return b ;  
}
```

# Функции

---

## Особенности:

- в начале заголовка ставится **тип результата**

```
int Max ( int a, int b )
```

- формальные параметры описываются так же, как и для процедур

```
float qq ( int a, float x, char c )
```

- можно использовать параметры-переменные

```
int Vasya (int &a, int &b )
```

- функции объявляются до основной программы

# Функции

## Особенности:

- МОЖНО ОБЪЯВЛЯТЬ И ИСПОЛЬЗОВАТЬ **локальные переменные**

```
float qq ( int a, int b)
{
  float x, y;
  ...
}
```

локальные  
переменные



**Локальные переменные недоступны в основной программе и других процедурах и функциях.**

# Программа

```
int Max ( int a, int b )  
{  
    ...  
}
```

формальные  
параметры

```
main ()
```

```
{  
    int a, b, c;  
    printf ( "Введите два числа\n" );  
    scanf ( "%d%d" &a, &b );  
    c = Max ( a, b );  
    printf ( "Наибольшее число %d", c );  
}
```

фактические  
параметры

ВЫЗОВ  
функции

# Задания

---

**«4»:** Составить функцию, которая определяет сумму всех чисел от 1 до N и привести пример ее использования.

**Пример:**

Введите число:

**100**

сумма чисел от 1 до 100 = 5050

**«5»:** Составить функцию, которая определяет, сколько зерен попросил положить на N-ую клетку изобретатель шахмат (на 1-ую – 1 зерно, на 2-ую – 2 зерна, на 3-ю – 4 зерна, ...)

**Пример:**

Введите номер клетки:

**28**

На 28-ой клетке 134217728 зерен.



## Задания (вариант 2 для 9-11 класса)

---

**«4»:** Составить функцию, которая определяет наибольший общий делитель двух натуральных и привести пример ее использования.

**Пример:**

Введите два числа:

**14 21**

$\text{НОД}(14, 21) = 7$

**«5»:** Составить функцию, которая вычисляет функцию синус как сумму ряда (с точностью 0.001)

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

**x в радианах!**

**Пример:**

Введите угол в градусах:

**45**

$\sin(45) = 0.707$

# Логические функции

**Задача:** составить функцию, которая определяет, верно ли, что заданное число – простое.

## Особенности:

- ответ – **логическое** значение: «да» (1) или «нет» (0)
- результат функции можно использовать как логическую величину **в условиях** (`if`, `while`)

**Алгоритм:** считаем число делителей в интервале от 2 до N-1, если оно не равно нулю – число составное.

```
count = 0;
for (i = 2; i < N; i++)
    if (N % i == 0) count++;
if (count == 0)
    // число N простое
else // число N составное
```



Как улучшить?

# Функция: простое число или нет

---

```
int Prime ( int N )  
{  
  int count = 0, i;  
  for (i = 2; i*i <= N; i++)  
    if (N % i == 0) count ++;  
  return (count == 0);  
}
```

перебор только до  $\sqrt{N}$

```
if (count == 0) return 1;  
else           return 0;
```

# Логические функции

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

```
int Prime ( int N )  
{  
    ...  
}
```

функция

```
main()  
{
```

```
    int N;
```

```
    printf ( "Введите целое число\n" );
```

```
    scanf ( "%d", &N );
```

```
    if ( Prime ( N ) )
```

```
        printf ( "%d - простое число", N );
```

```
    else printf ( "%d - составное число", N );
```

```
}
```

# Задания

---

**«4»:** Составить функцию, которая определяет, верно ли, что сумма его цифр – четное число.

**Пример:**

Введите число:

**136**

Сумма цифр четная.

Введите число:

**245**

Сумма цифр нечетная.

**«5»:** Составить функцию, которая определяет, верно ли, что в заданном числе все цифры стоят по возрастанию.

**Пример:**

Введите число:

**258**

Верно.

Введите число:

**528**

Неверно.

# Программирование на языке Си

## Тема 14. Случайные числа

# Случайные числа

---

**Случайные явления:** везде...

- бросание монеты («орел» или «решка»)
- падение снега
- броуновское движение
- помехи при телефонной связи
- шум радиозэфира

**Случайные числа** – это такая последовательность чисел, для которой невозможно предсказать следующее даже зная все предыдущие.

**Проблема:** как получить на компьютере?

**Возможные решения:**

- использовать внешний источник шумовых помех
- с помощью математических преобразований

# Псевдослучайные числа

**Псевдослучайные числа** – это такая последовательность чисел, которая обладает свойствами случайных чисел, но каждое следующее число вычисляется по заданной формуле.

## Примеры:

1. Случайные *целые* числа  $[0, m)$  (**линейный конгруэнтный метод**)

$$x_n = (a \cdot x_{n-1} + c) \bmod m$$

$$x_n = (16807 \cdot x_{n-1} + 12345) \bmod 1073741823$$

$a, c, m$  - целые числа

$2^{30}-1$

простое число

остаток от деления

2. Случайные *вещественные* числа  $[0, 1]$

$$x_n = \left\{ (\pi + x_{n-1})^k \right\}$$

дробная часть числа

## Литература:

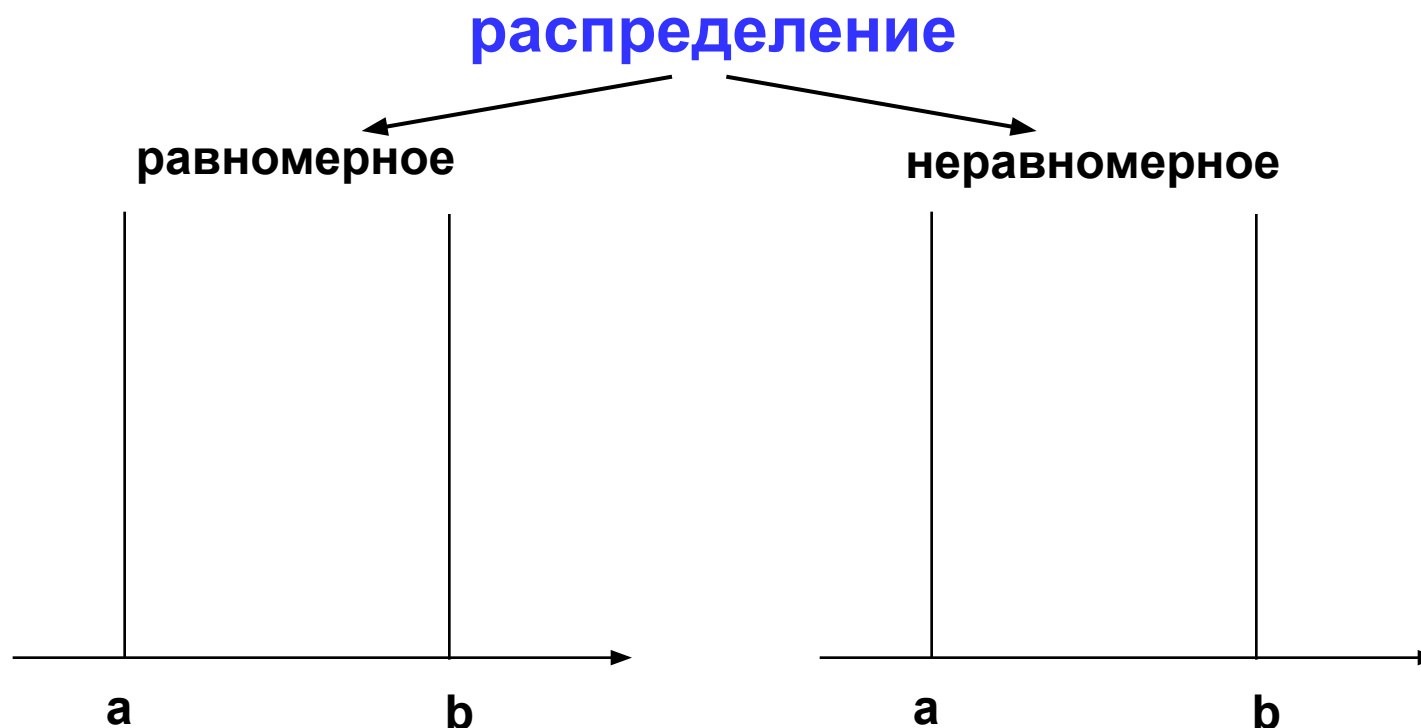
например,  $k = 5$

Д. Кнут, Искусство программирования для ЭВМ, т.2.



# Распределение случайных чисел

**Модель:** снежинки падают на отрезок  $[a,b]$

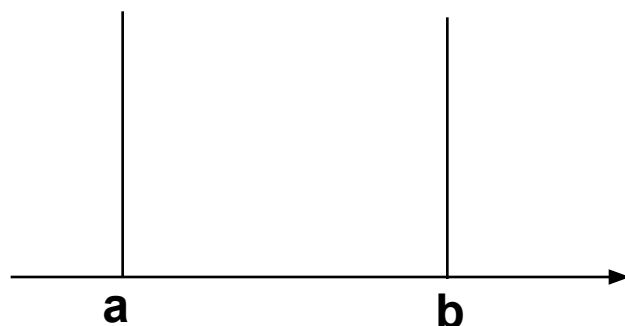


Сколько может быть разных распределений?

# Распределение случайных чисел

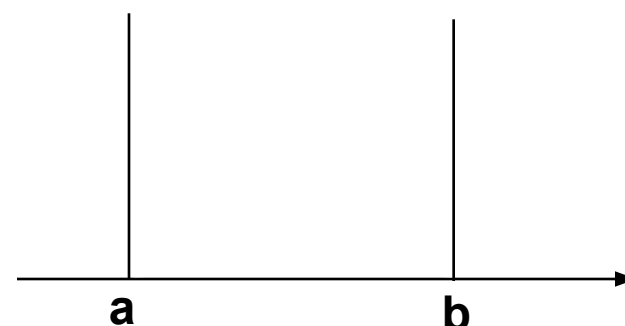
## Особенности:

- распределение – это характеристика **всей последовательности**, а не одного числа
- **равномерное** распределение одно, компьютерные датчики (псевдо)случайных чисел дают равномерное распределение
- неравномерных – много
- любое неравномерное можно получить с помощью равномерного



$$x = \frac{x_1 + x_2}{2}$$

равномерное распределение



$$x = \frac{x_1 + x_2 + \dots + x_{12}}{12}$$

равномерное распределение

# Генератор случайных чисел в Си

```
#include <stdlib.h> // случайные числа
```

**RAND\_MAX** – максимальное случайное целое число  
(обычно `RAND_MAX = 32767`)

**rand()** – случайное целое число в интервале  
[0, `RAND_MAX`]

```
int x, y;  
x = rand(); // первое число [0, RAND_MAX]  
y = rand(); // уже другое число
```

**srand(N)** – установить начальное значение  
последовательности случайных чисел N:

```
srand ( 345 ); // начнем с 345
```

# Целые числа в заданном интервале

---

Целые числа в интервале  $[0, N-1]$ :

```
int random(int N) {  
    return rand() % N;  
}
```

Примеры:

```
x = random ( 100 ) ;    // интервал [0, 99]  
x = random ( z ) ;     // интервал [0, z-1]
```

Целые числа в интервале  $[a, b]$ :

```
x = random ( z ) + a ;    // интервал [a, z-1+a]  
x = random ( b - a + 1 ) + a ; // интервал [a, b]
```

# Генератор случайных чисел в Си

---

## Вещественные числа в интервале $[0, 1]$

```
float x;
```

```
[0, RAND_MAX] = [0, 32767]
```

```
x = 1.*rand() / RAND_MAX; // интервал [0, 1]
```

## Вещественные числа в интервале $[0, z]$

```
x = 1.*z*rand() / RAND_MAX;
```

## Вещественные числа в интервале $[a, z+a]$

```
x = 1.*z*rand() / RAND_MAX + a;
```

## Вещественные числа в интервале $[a, b]$

```
x = 1.*(b-a)*rand() / RAND_MAX + a;
```

# Случайные числа

---

**Задача:** заполнить прямоугольник 400 на 300 пикселей равномерно точками случайного цвета



**Как получить случайные координаты точки?**

```
x = random ( 400 ) ;
```

```
y = random ( 300 ) ;
```

**Как добиться равномерности?**

обеспечивается автоматически при использовании функции `random`

**Как получить случайный цвет?**

```
R = random ( 256 ) ; G = random ( 256 ) ;
```

```
B = random ( 256 ) ;
```

`COLOR ( R , G , B )`

# Программа

```
#include <graphics.h>
#include <conio.h>
#include <stdlib.h>
```

функция для получения  
случайного числа от 0  
до N-1

```
int random(int N) {
    return rand() % N;
}
```

```
main()
{
    int x, y, R, G, B;
    initwindow ( 500, 500 );
    // цикл до нажатия на Esc
    closegraph();
}
```

# ОСНОВНОЙ ЦИКЛ

бесконечный  
цикл???

ВЫХОД ПО  
*Esc*

```
while ( 1 ) {  
    if ( kbhit() )  
        if ( 27 == getch() ) break;  
  
    x = random(400);  
    y = random(300);  
    R = random(256);  
    G = random(256);  
    B = random(256);  
    putpixel ( x, y, COLOR(R,G,B) );  
}
```

случайные  
координаты

случайный цвет



# Задания

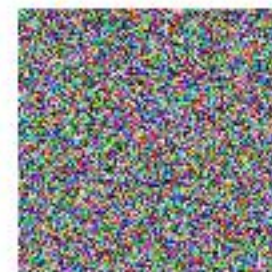
---

**«3»:** Заполнить квадрат точками случайного цвета.  
размер квадрата ввести с клавиатуры:

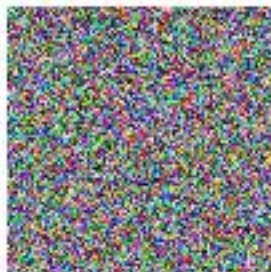
**Пример:**

Введите размер квадрата :

**150**



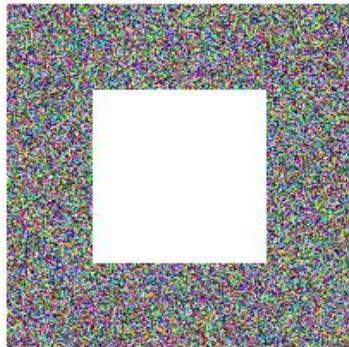
**«4»:** Заполнить область точками случайного цвета:



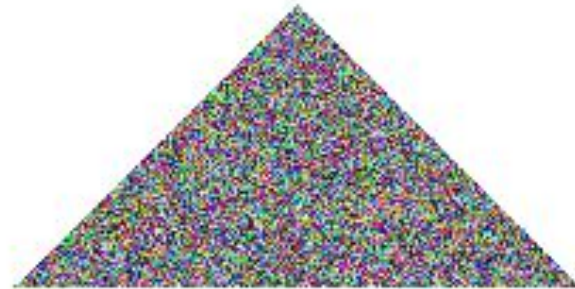
# Задания

---

«5»: Заполнить область точками случайного цвета:



или



# Конец фильма

---

**ПОЛЯКОВ Константин Юрьевич**  
**д.т.н., учитель информатики высшей**  
**категории,**  
**ГОУ СОШ № 163, г. Санкт-Петербург**  
**[kpolyakov@mail.ru](mailto:kpolyakov@mail.ru)**