

# Программирование на языке Си

- § 54. Алгоритм и его свойства
- § 55. Простейшие программы
- § 56. Вычисления
- § 57. Ветвления
- § 58. Циклические алгоритмы
- § 59. Процедуры
- § 60. Функции
- § 61. Рекурсия

# Программирование на языке Си

## § 54. Алгоритм и его свойства

# Что такое алгоритм?

**Алгоритм** — это точное описание порядка действий, которые должен выполнить исполнитель для решения задачи за конечное время.

**Исполнитель** – это устройство или одушевленное существо (человек), способное понять и выполнить команды, составляющие алгоритм.

**Формальные исполнители:** не понимают (и не могут понять) смысл команд.



Мухаммед ал-Хорезми  
(ок. 783—ок. 850 гг.)

# Свойства алгоритма

**Дискретность** — алгоритм состоит из отдельных команд, каждая из которых выполняется за конечное время.

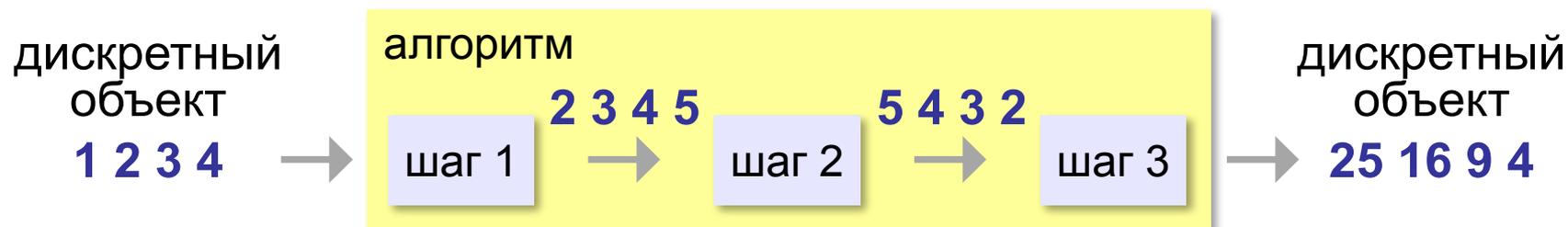
**Детерминированность** (определённость) — при каждом запуске алгоритма с одними и теми же исходными данными получается один и тот же результат.

**Понятность** — алгоритм содержит только команды, входящие в **систему команд исполнителя**.

**Конечность** (результативность) — для корректного набора данных алгоритм должен завершаться через конечное время.

**Корректность** — для допустимых исходных данных алгоритм должен приводить к правильному результату.

# Как работает алгоритм?



- получает на вход дискретный объект
- в результате строит другой дискретный объект (или выдаёт сообщение об ошибке)
- обрабатывает объект по шагам
- на каждом шаге получается новый дискретный объект

# Способы записи алгоритмов

---

- **естественный язык**

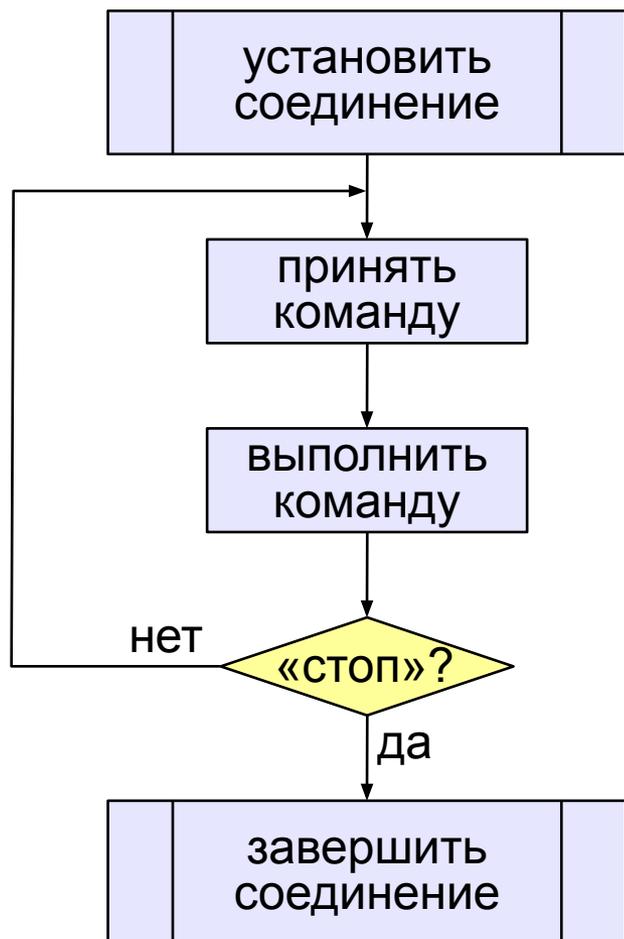
```
установить соединение  
пока не принята команда «стоп»  
    принять команду  
    выполнить команду  
завершить сеанс связи
```

- **псевдокод**

```
установить соединение  
начало цикла  
    принять команду  
    выполнить команду  
конец цикла при команда = 'stop'  
завершить сеанс связи
```

# Способы записи алгоритмов

- блок-схема



- программа

```
установитьСоединение
начало цикла
  cmd = получитьКоманду
  выполнитьКоманду (cmd)
конец при cmd = 'stop'
закретьСоединение
```

# Программирование на языке Си

## § 55. Простейшие программы

# Простейшая программа

это основная программа

```
main ()  
{  
    // это основная программа  
    /* здесь записывают  
       операторы */  
}
```

комментарии после //  
не обрабатываются

это тоже комментарий



Что делает эта программа?

# Вывод на экран

```
main ()
```

```
{
```

"\n" – новая строка

```
printf ("2+");
```

```
printf ("2=?\n");
```

```
printf ("Ответ: 4");
```

```
}
```

**Протокол:**

2+

Ответ: 4

# Подключение библиотечных функций

```
#include <stdio.h>
main ()
{
    printf ("2+");
    printf ("2=?\n");
    printf ("Ответ: 4");
    getchar ();
}
```

стандартные функции  
ввода и вывода

ждать нажатия любой  
клавиши

# Задания

---

«В»: Вывести на экран текст «лесенкой»

Вася

пошел

гулять

«С»: Вывести на экран рисунок из букв

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

# Сложение чисел

**Задача.** Ввести с клавиатуры два числа и найти их сумму.

**Протокол:**

Введите два целых числа

компьютер

25 30

пользователь

25+30=55

компьютер считает сам!

?

1. Как ввести числа в память?
2. Где хранить введенные числа?
3. Как вычислить?
4. Как вывести результат?

## Сумма: псевдокод

```
main ()  
{  
    // ввести два числа  
    // вычислить их сумму  
    // вывести сумму на экран  
}
```

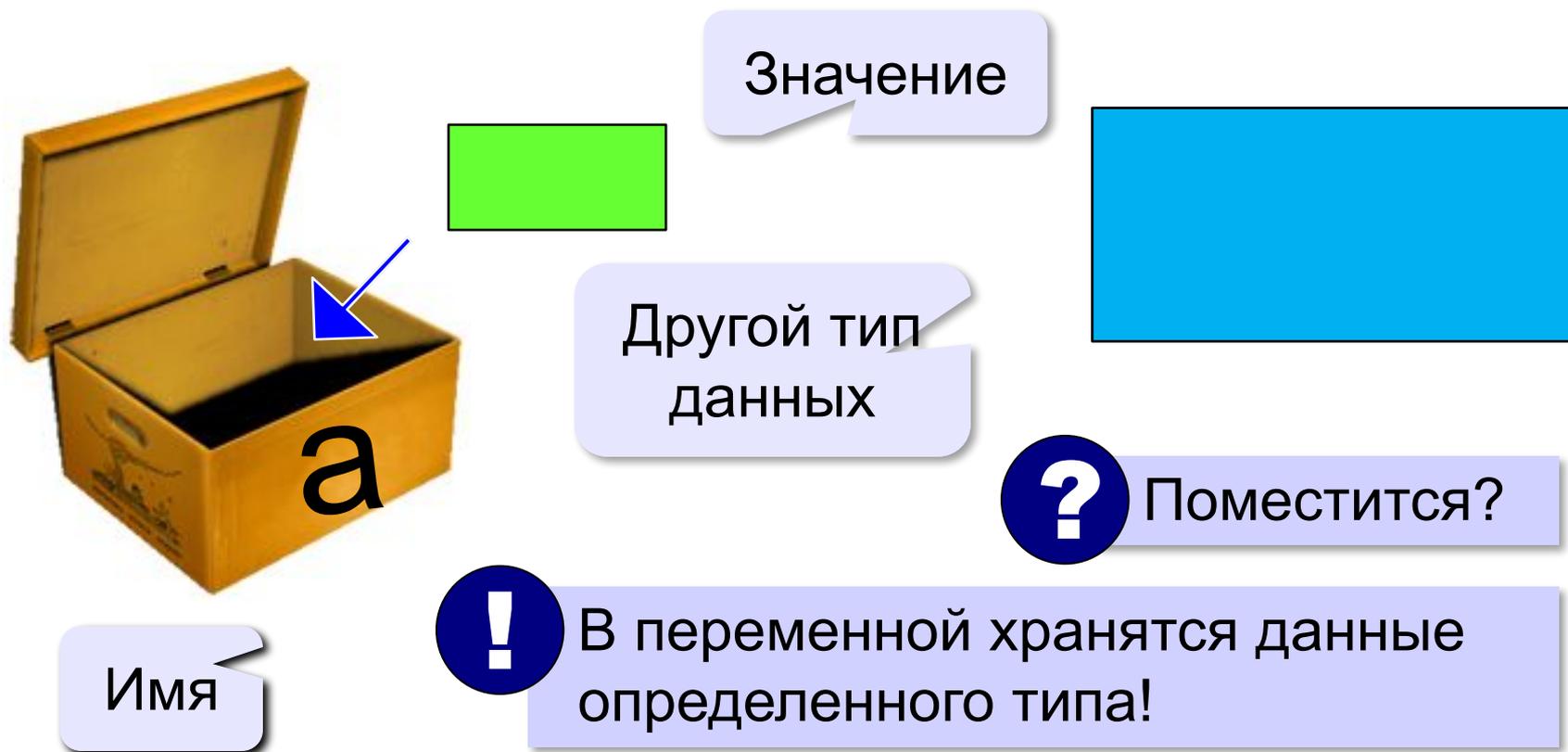
**Псевдокод** – алгоритм на русском языке с элементами языка программирования.



**Компьютер не может исполнить псевдокод!**

# Переменные

**Переменная** – это величина, имеющая имя, тип и значение. Значение переменной можно изменять во время работы программы.



# Имена переменных

**МОЖНО** использовать

- латинские буквы (A-Z, a-z)

заглавные и строчные буквы **различаются**

- цифры

имя не может начинаться с цифры

- знак подчеркивания \_

**НЕЛЬЗЯ** использовать

~~• русские буквы~~

~~• скобки~~

~~• знаки +, =, !, ? и др.~~

Какие имена правильные?

**AXby R&B 4Wheel Вася “PesBarbos”**

**TU154 [QuQu] \_ABBA A+B**

# Объявление переменных

---

## Типы переменных:

- `int` // целая
- `float` // вещественная
- и другие...

выделение  
места в памяти

## Объявление переменных:

тип – целые

список имен  
переменных

```
int a, b, c;
```

# Тип переменной

---

- область допустимых значений
- допустимые операции
- объём памяти
- формат хранения данных
- для предотвращения случайных ошибок

## Начальные значения:

```
int a, b = 1, c = 55;
```

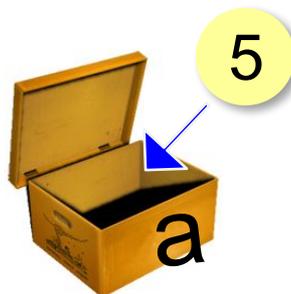


Что в переменной **a**?

# Как записать значение в переменную?

оператор  
присваивания

```
a = 5;
```

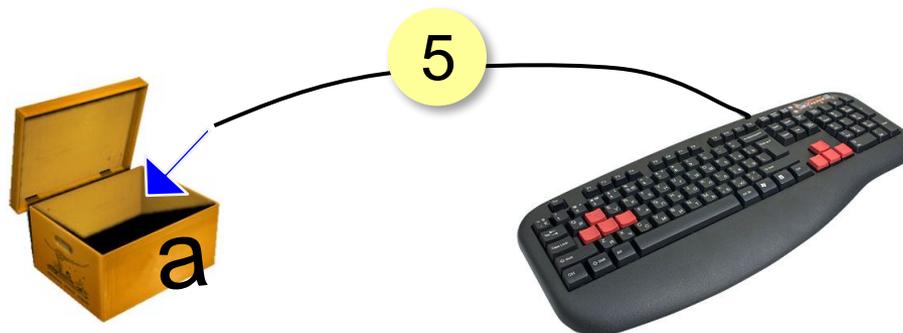


При записи нового значения старое стирается!

**Оператор** – это команда языка программирования (инструкция).

**Оператор присваивания** – это команда для записи нового значения в переменную.

# Ввод значения с клавиатуры



1. Программа ждет, пока пользователь введет значение и нажмет *Enter*.
2. Введенное значение записывается в переменную **a**.

# Ввод значения с клавиатуры

функция  
ввода

формат  
ввода

**%d** – целое

**%f** – вещественное

адрес переменной **a**

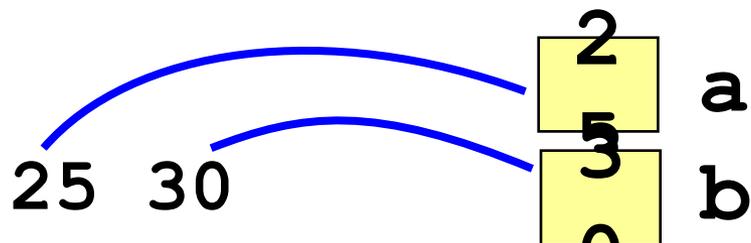
```
scanf ( "%d" , &a ) ;
```

ввести целое число и записать в память по адресу переменной **a**

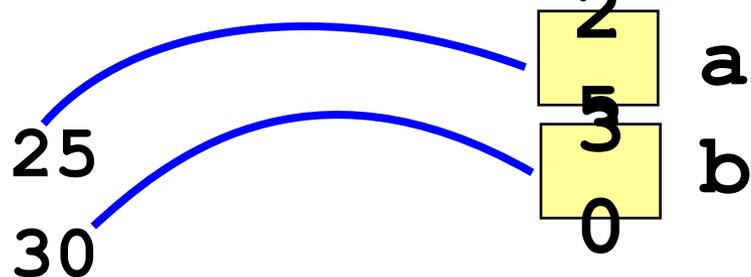
# Ввод значений двух переменных

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

через пробел:



через *Enter*:



# Изменение значений переменной

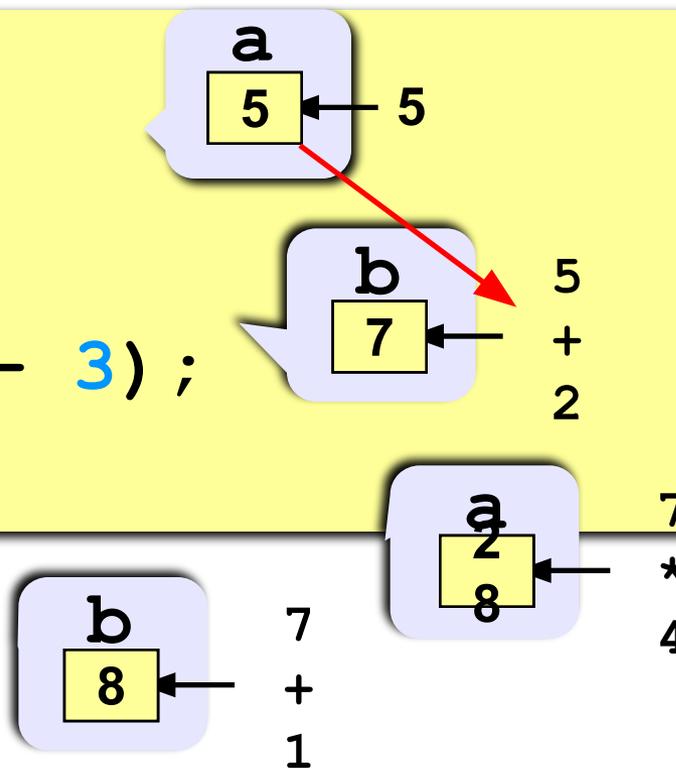
```
int a, b;
```

```
a = 5;
```

```
b = a + 2;
```

```
a = (a + 2) * (b - 3);
```

```
b = b + 1;
```



# Вывод данных

---

формат вывода

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

// вывод значения  
// переменной a

```
printf("%d\n", a);
```

// ... и переход  
// на новую строку

"\n" – новая строка

# Вывод данных

---

```
printf( "Привет!" ); //вывод текста
```

```
printf( "Ответ: %d", c );
```

//вывод текста и значения переменной c

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

# Сложение чисел: простое решение

```
#include <stdio.h>
main()
{
    int a, b, c;
    scanf( "%d%d", &a, &b );
    c = a + b;
    printf( "%d", c );
    getchar();
}
```



Что плохо?

# Сложение чисел: полное решение

```
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);  
}
```

подсказка

Протокол:

КОМПЬЮТЕР

Введите два целых числа

25 30

ПОЛЬЗОВАТЕЛЬ

25+30=55

# Снова про оператор вывода

## Вычисление выражений:

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

## Форматный вывод:

```
a = 123;
```

```
printf( "%5d" , a );
```

123

5 знаков

# Программирование на языке Си

## § 56. Вычисления

# Типы данных

---

- `int` // целое
- `long int` // длинное целое
- `float` // вещественное
- `double` // веществ. двойной точности
- `bool` // логические значения
- `char` // символ

# Арифметическое выражения

$$a = (c + b * 5 * 3 - 1) / 2 * d;$$

**Приоритет** (старшинство):

- 1) скобки
- 2) умножение и деление
- 3) сложение и вычитание

$$a = \frac{c + b \cdot 5 \cdot 3 - 1}{2} \cdot d$$

# Деление

Результат деления целого на целое – **целое** число  
(остаток отбрасывается):

```
int a = 3, b = 4;
```

```
float x;
```

```
x = 3 / 4;
```

```
x = 3. / 4;
```

```
x = 3 / 4.;
```

```
x = a / 4;
```

```
x = a / 4.;
```

```
x = a / b;
```

```
x = float(a) / 4;
```

```
x = a / float(b);
```



Что запишется в **x**?

# Остаток от деления

`%` – остаток от деления

```
int a, b, d;  
d = 85;  
b = d / 10;  
a = d % 10;  
d = a % b;  
d = b % a;
```

Для отрицательных чисел:

```
int a = -7;  
b = a / 2;  
d = a % 2;
```



В математике не так!

остаток  $\geq 0$

$$-7 = (-4) * 2 + 1$$

# Сокращенная запись операций

---

```
int a, b;
```

```
...
```

```
a ++; // a = a + 1;
```

```
a --; // a = a - 1;
```

```
a += b; // a = a + b;
```

```
a -= b; // a = a - b;
```

```
a *= b; // a = a * b;
```

```
a /= b; // a = a / b;
```

```
a %= b; // a = a % b;
```

# Вещественные числа



Целая и дробная части числа разделяются точкой!

## Форматы вывода:

```
float x = 123.456;
printf("%f\n", x);
printf("%10.2f\n", x);
```

6 цифр в дробной части

123.456001

\_\_\_\_\_123.46

всего знаков

в дробной части

6 значащих цифр

```
printf("%g\n", x);
printf("%10.2g\n", x);
```

123.456

\_\_\_1.2e+002

значащих цифр

$1,2 \cdot 10^2$

# Вещественные числа

## Экспоненциальный формат:

```
float x;
```

```
x = 1./30000;
```

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

```
x = 12345678.;
```

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

 $3,333333 \cdot 10^{-5}$ 

3.333333e-005

1.234568e+007

```
float x = 123.456;
```

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

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

 $1,234568 \cdot 10^7$ 

1.234560e+002

1.23e+002

всего знаков

в дробной части

# Стандартные функции

подключить  
математическую  
библиотеку

```
#include <math.h>
```

- `abs (x)` — модуль целого числа
- `fabs (x)` — модуль вещественного числа
- `sqrt (x)` — квадратный корень
- `sin (x)` — синус угла, заданного **в радианах**
- `cos (x)` — косинус угла, заданного **в радианах**
- `exp (x)` — экспонента  $e^x$
- `ln (x)` — натуральный логарифм
- `pow (x, y)` —  $x^y$ : возведение числа  $x$  в степень  $y$
- `floor (x)` — округление «вниз»
- `ceil (x)` — округление «вверх»

```
float x;  
x = floor(1.6); // 1  
x = ceil(1.6); // 2
```

```
x = floor(-1.6); // -2  
x = ceil(-1.6); // -1
```

# Случайные числа

## Случайно...

- встретить друга на улице
- разбить тарелку
- найти 10 рублей
- выиграть в лотерею

## Случайный выбор:

- жеребьевка на соревнованиях
- выигравшие номера в лотерее

## Как получить случайность?



# Случайные числа на компьютере

## Электронный генератор



- нужно специальное устройство
- нельзя воспроизвести результаты

**Псевдослучайные числа** – обладают свойствами случайных чисел, но каждое следующее число вычисляется по заданной формуле.

## Метод середины квадрата (Дж. фон Нейман)

зерно

564321

в квадрате

- малый период  
(последовательность повторяется через  $10^6$  чисел)

318458191041

209938992481

# Линейный конгруэнтный генератор

$$X = (a * X + b) \% c \quad | \quad \text{интервал от } 0 \text{ до } c-1$$

$$X = (X + 3) \% 10 \quad | \quad \text{интервал от } 0 \text{ до } 9$$

$$X = 0 \rightarrow 3 \rightarrow 6 \rightarrow 9 \rightarrow 2 \rightarrow 5 \rightarrow 8$$

$$8 \rightarrow 1 \rightarrow 4 \rightarrow 7 \rightarrow 0$$

зерно

защипливание



Важно правильный выбор параметров  $a$ ,  $b$  и  $c$ !

**Компилятор GCC:**

$$a = 1103515245$$

$$b = 12345$$

$$c = 2^{31}$$

# Генератор случайных чисел

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

Генератор на отрезке [0,RAND\_MAX]:

```
int X, Y;  
X = rand(); // псевдослучайное число  
Y = rand(); // это уже другое число!
```

англ. *random* – случайный

Целые числа на отрезке [a,b]:

```
int X, Y;  
X = a + rand() % (b - a + 1);  
Y = a + rand() % (b - a + 1);
```

?

Почему так?

[0, b-a]

# Задачи

---

«А»: Ввести с клавиатуры три целых числа, найти их сумму, произведение и среднее арифметическое.

**Пример:**

Введите три целых числа:

5 7 8

$$5+7+8=20$$

$$5*7*8=280$$

$$(5+7+8)/3=6.667$$

«В»: Ввести с клавиатуры координаты двух точек (А и В) на плоскости (вещественные числа). Вычислить длину отрезка АВ.

**Пример:**

Введите координаты точки А:

5.5 3.5

Введите координаты точки В:

1.5 2

$$\text{Длина отрезка АВ} = 4.272$$

# Задачи

---

**«С»:** Получить случайное трехзначное число и вывести через запятую его отдельные цифры.

**Пример:**

Получено число 123.

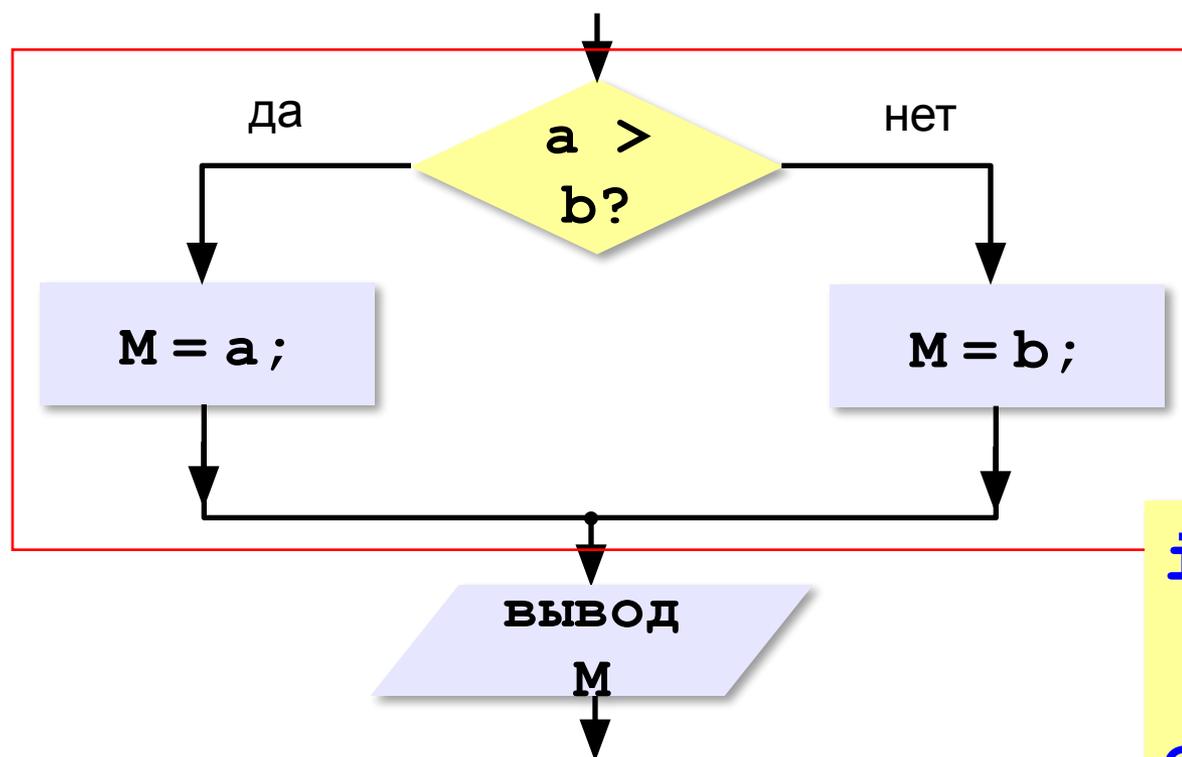
Его цифры 1, 2, 3.

# Программирование на языке Си

## **§ 57. Ветвления**

# Условный оператор

Задача: **изменить порядок действий** в зависимости от выполнения некоторого условия.



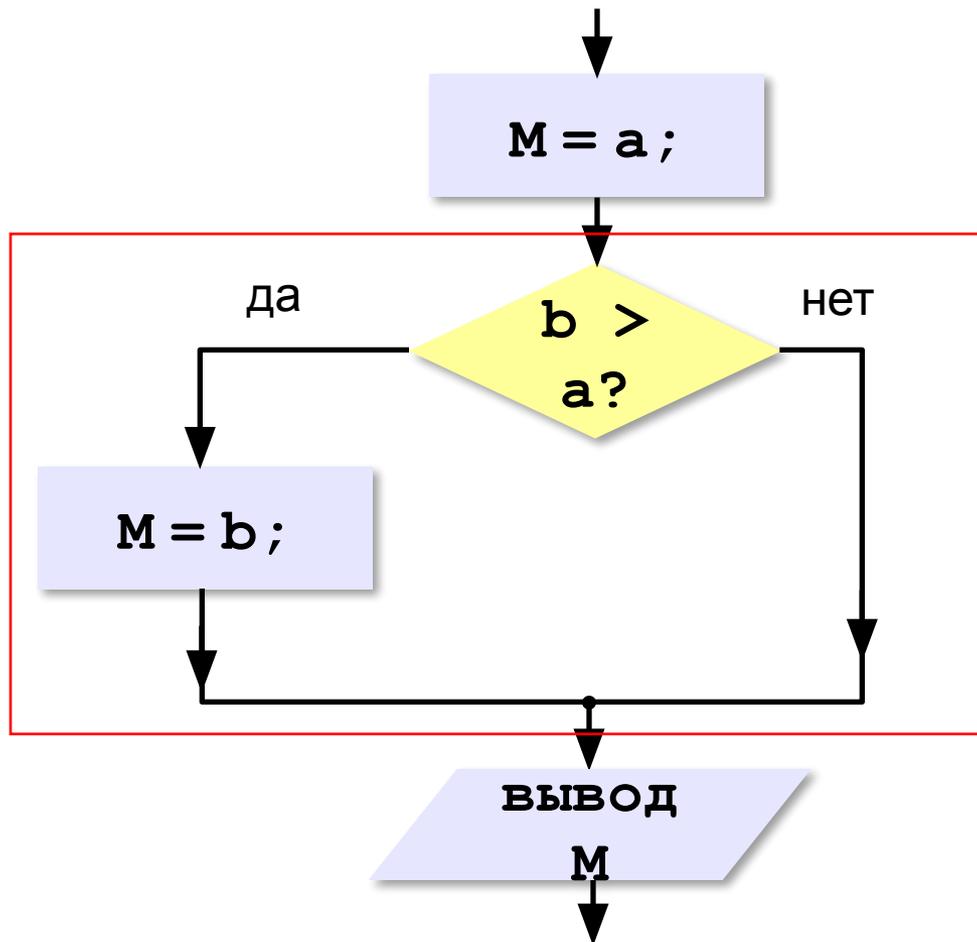
полная  
форма  
ветвления



Если  $a = b$ ?

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

# Условный оператор: неполная форма



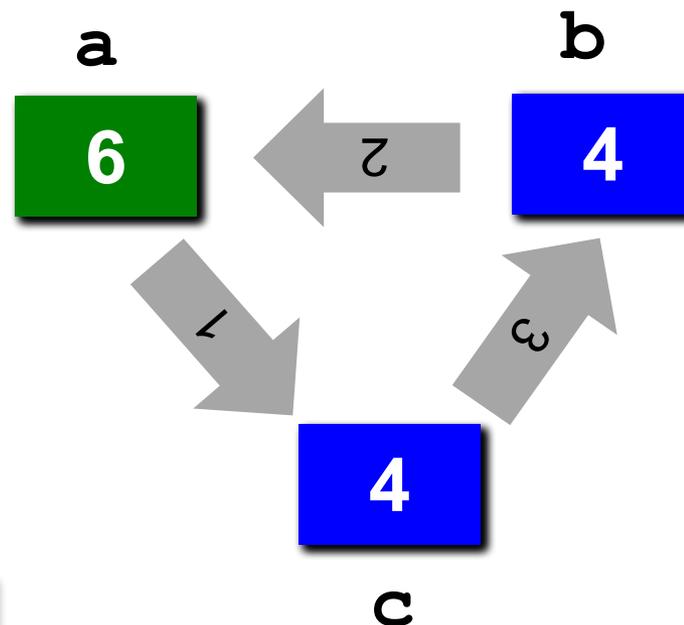
```
M = a;  
if ( b > a )  
    M = b;
```

неполная  
форма  
ветвления

# Условный оператор

```
if ( a < b )  
{  
  c = a;  
  a = b;  
  b = c;  
}
```

? Что делает?



? Можно ли обойтись без переменной c?

# Знаки отношений

---

**>** **<** больше, меньше

**>=** больше или равно

**<=** меньше или равно

**==** равно

**!=** не равно

# Вложенные условные операторы

Задача: в переменных **a** и **b** записаны возрасты Андрея и Бориса. Кто из них старше?

?

Сколько вариантов?

```
if ( a == b )  
    printf("Одного возраста");  
else  
    if ( a > b )  
        printf("Андрей старше");  
    else  
        printf("Борис старше");
```

?

Зачем нужен?

вложенный  
условный оператор

# Задачи

---

**«А»:** Ввести три целых числа, найти максимальное из них.

**Пример:**

Введите три целых числа :

1 5 4

Максимальное число 5

**«В»:** Ввести пять целых чисел, найти максимальное из них.

**Пример:**

Введите пять целых чисел :

1 5 4 3 2

Максимальное число 5

# Задачи

---

**«С»:** Ввести последовательно возраст Антона, Бориса и Виктора. Определить, кто из них старше.

**Пример:**

Возраст Антона: 15

Возраст Бориса: 17

Возраст Виктора: 16

Ответ: Борис старше всех.

**Пример:**

Возраст Антона: 17

Возраст Бориса: 17

Возраст Виктора: 16

Ответ: Антон и Борис старше Виктора.

# Сложные условия

Задача: набор сотрудников в возрасте **25-40 лет**  
(включительно).

сложное условие

```
if ( v >= 25 && v <= 40 )  
    printf ("подходит");  
else  
    printf ("не подходит");
```

**&&** «И»

**||** «ИЛИ»

**!** «НЕ»

## Приоритет :

- 1) отношения (<, >, <=, >=, ==, !=)
- 2) ! («НЕ»)
- 3) && («И»)
- 4) || («ИЛИ»)

# Задачи

---

**«А»:** Напишите программу, которая получает три числа и выводит количество одинаковых чисел в этой цепочке.

**Пример:**

Введите три числа:

5 5 5

Все числа одинаковые.

**Пример:**

Введите три числа:

5 7 5

Два числа одинаковые.

**Пример:**

Введите три числа:

5 7 8

Нет одинаковых чисел.

# Задачи

---

**«В»:** Напишите программу, которая получает номер месяца и выводит соответствующее ему время года или сообщение об ошибке.

**Пример:**

**Введите номер месяца :**

**5**

**Весна .**

**Пример:**

**Введите номер месяца :**

**15**

**Неверный номер месяца .**

# Задачи

---

**«С»:** Напишите программу, которая получает возраст человека (целое число, не превышающее 120) и выводит этот возраст со словом «год», «года» или «лет». Например, «21 год», «22 года», «25 лет».

**Пример:**

Введите возраст: **18**

Вам 18 лет.

**Пример:**

Введите возраст: **21**

Вам 21 год.

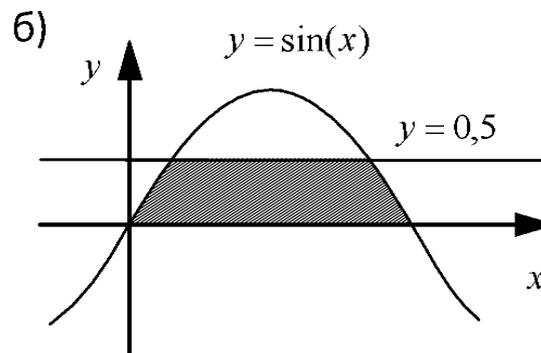
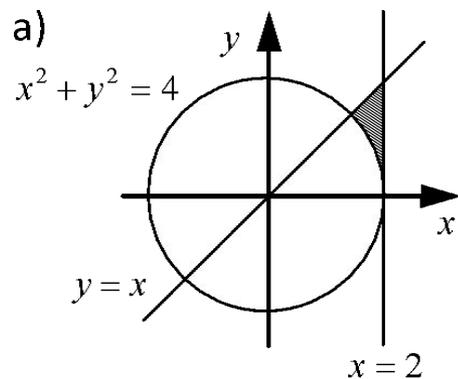
**Пример:**

Введите возраст: **22**

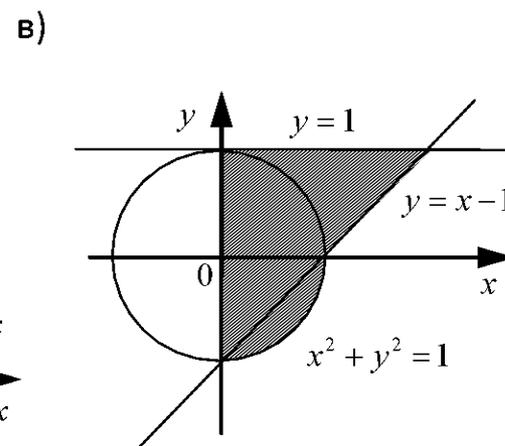
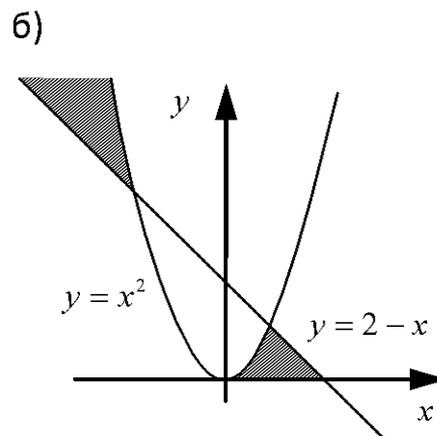
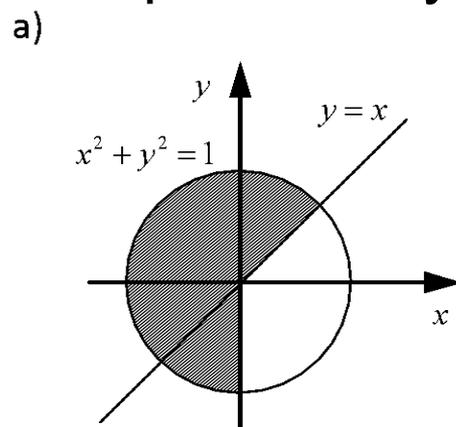
Вам 22 года.

# Задачи

«А»: Напишите условие, которое определяет заштрихованную область.

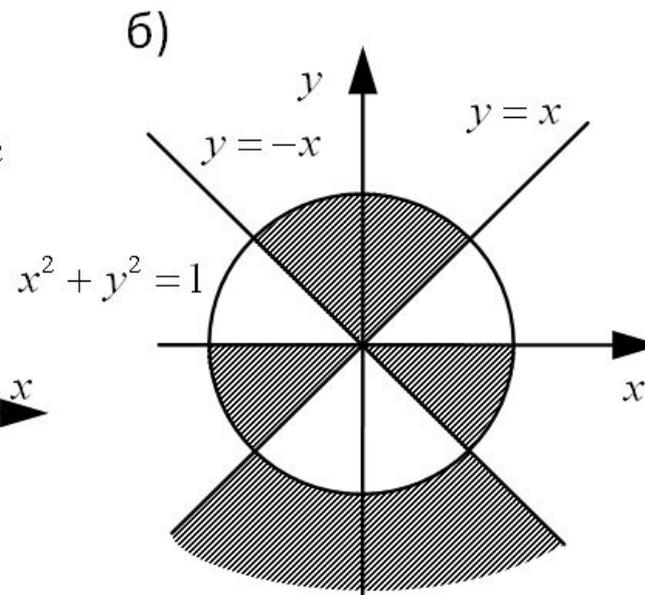
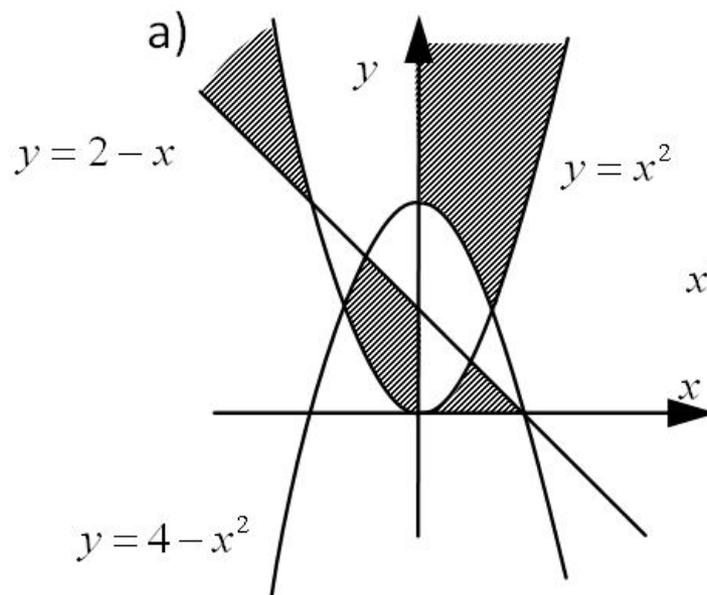


«В»: Напишите условие, которое определяет заштрихованную область.



# Задачи

«С»: Напишите условие, которое определяет заштрихованную область.



# Множественный выбор

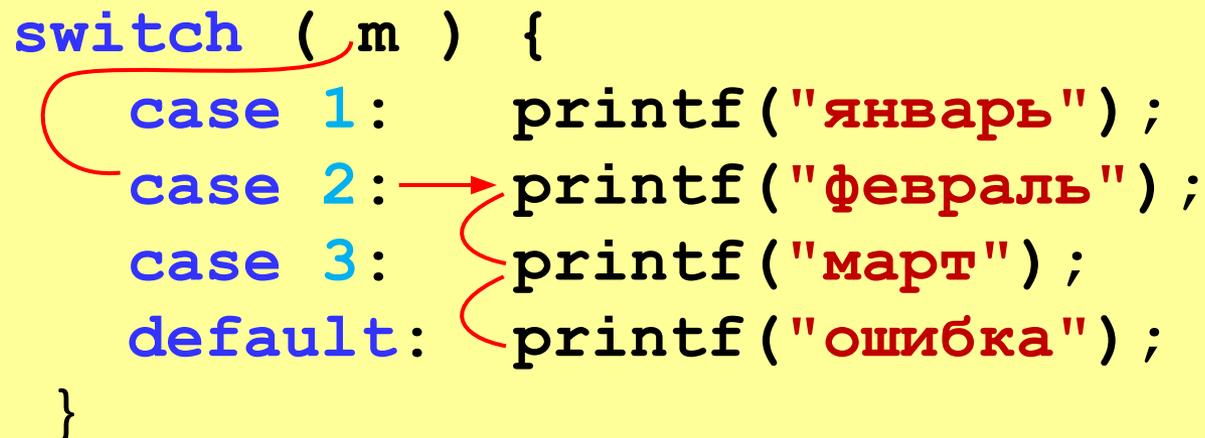
```
if (m == 1) printf("январь");  
if (m == 2) printf("февраль");  
...  
if (m == 12) printf("декабрь");
```

```
switch ( m ) {  
    case 1: printf("январь");  
            break;  
    case 2: printf("февраль");  
            break;  
    ...  
    case 12: printf("декабрь");  
             break;  
    default: printf("ошибка");  
}
```

# Множественный выбор

Если не ставить **break**:

```
switch ( m ) {  
    case 1:    printf ( "январь" ) ;  
    case 2:    printf ( "февраль" ) ;  
    case 3:    printf ( "март" ) ;  
    default:   printf ( "ошибка" ) ;  
}
```



При  $m = 2$ : февральмартошибка

# Множественный выбор

```
char c;  
c = getchar();  
switch (c)  
{  
  case 'а':  
    printf("антилопа\n");  
    printf("Анапа\n");  
    break;  
  ...  
  case 'я':  
    printf("ягуар\n");  
    printf("Якутск\n");  
    break;  
  default: printf("Ошибка!");  
}
```

ждать нажатия клавиши,  
получить её код

несколько  
операторов в  
блоке

# Программирование на языке Си

## **§ 58. Циклические алгоритмы**

# Что такое цикл?

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

## Два вида циклов:

- цикл с **известным** числом шагов (сделать 10 раз)
- цикл с **неизвестным** числом шагов (делать, пока не надоест)

*Задача.* Вывести на экран 10 раз слово «Привет».



Можно ли решить известными методами?

# Повторения в программе

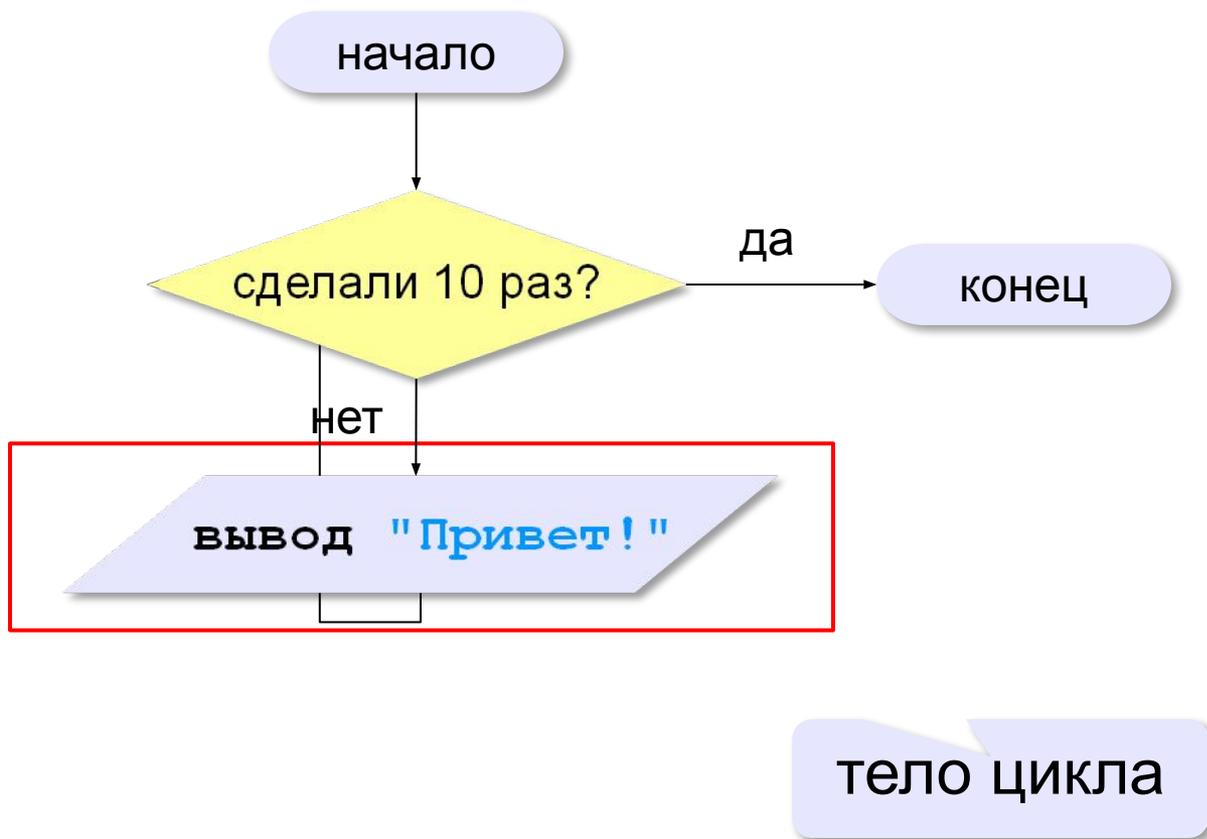
---

```
printf ("Привет\n" );  
printf ("Привет\n" );  
...  
printf ("Привет\n" );
```



Что плохо?

# Блок-схема цикла



# Как организовать цикл?

```
счётчик = 0
пока счётчик < 10
    printf("Привет\n");
    увеличить счётчик на 1
```

результат операции  
автоматически  
сравнивается с нулём!

```
счётчик = 10
пока счётчик > 0
    printf("Привет\n");
    уменьшить счётчик на 1
```



Какой способ удобнее для процессора?

# Цикл с условием

**Задача.** Определить **количество цифр** в десятичной записи целого положительного числа, записанного в переменную  $n$ .

```
счётчик = 0
```

```
пока  $n > 0$ 
```

```
    отсечь последнюю цифру  $n$   
    увеличить счётчик на 1
```

$n$	счётчик
-----	---------

1234	0
------	---

Как отсечь последнюю цифру?

```
 $n = n / 10;$ 
```

Как увеличить счётчик на 1?

```
счётчик = счётчик + 1;
```

```
счётчик ++;
```

# Цикл с условием

начальное значение  
счётчика

условие  
продолжения

заголовок  
цикла

```
count = 0;  
while ( n > 0 )  
{  
    n = n / 10;  
    count ++;  
}
```

конец  
цикла

тело цикла



Цикл с предусловием – проверка на входе в цикл!

# Цикл с условием

---

При известном количестве шагов:

```
k = 0;
while ( k < 10 )
{
    printf ( "привет\n" );
    k ++;
}
```

Защелкивание:

```
k = 0;
while ( k < 10 )
{
    printf ( "привет\n" );
}
```

# Сколько раз выполняется цикл?

```
a = 4; b = 6;  
while ( a < b ) a = a + 1;
```

2 раза

a = 6

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

# Цикл с постусловием

заголовок  
цикла

```
do
```

```
{
```

```
printf("Введите n > 0: ");
```

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

```
}
```

```
while ( n <= 0 );
```

тело цикла

условие  
продолжения

- при входе в цикл условие **не проверяется**
- цикл всегда выполняется **хотя бы один раз**

# Задачи

---

**«А»:** Напишите программу, которая получает два целых числа  $A$  и  $B$  ( $0 < A < B$ ) и выводит квадраты всех натуральных чисел в интервале от  $A$  до  $B$ .

**Пример:**

**Введите два целых числа :**

**10 12**

$10 * 10 = 100$

$11 * 11 = 121$

$12 * 12 = 144$

**«В»:** Напишите программу, которая получает два целых числа и находит их произведение, не используя операцию умножения. Учтите, что числа могут быть отрицательными.

**Пример:**

**Введите два числа :**

**10 -15**

$10 * (-15) = -150$

# Задачи

---

**«С»:** Ввести натуральное число  $N$  и вычислить сумму всех чисел Фибоначчи, меньших  $N$ . Предусмотрите защиту от ввода отрицательного числа  $N$ .

**Пример:**

Введите число  $N$ :

**10000**

Сумма 17710

## Задачи-2

---

**«А»:** Ввести натуральное число и найти сумму его цифр.

**Пример:**

Введите натуральное число:

**12345**

Сумма цифр 15.

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

**Пример:**

Введите натуральное число:

**12342**

Нет.

**Пример:**

Введите натуральное число:

**12245**

Да.

## Задачи-2

---

**«С»:** Ввести натуральное число и определить, верно ли, что в его записи есть две одинаковые цифры (не обязательно стоящие рядом).

**Пример:**

**Введите натуральное число:**

**12342**

**Да .**

**Пример:**

**Введите натуральное число:**

**12345**

**Нет .**

# Цикл с переменной

Задача. Вывести все степени двойки от  $2^1$  до  $2^{10}$ .



Можно ли сделать с циклом «пока»?

```
k = 1;
n = 2;
while ( k <= 10 )
{
    printf ("%d\n", n);
    n *= 2;
    k ++;
}
```

```
n = 2;
for ( k=1; k<=10; k++ )
{
    printf ("%d\n", n);
    n *= 2;
}
```

ЦИКЛ С  
переменной

## Цикл с переменной: другой шаг

```
for ( k = 10; k >= 1; k-- )  
    printf( "%d\n", k*k );
```



Что получится?

```
for ( k = 1; k <= 10; k += 2 )  
    printf( "%d\n", k*k );
```

100  
81  
64  
49  
36  
25  
16  
9  
4  
1

1  
9  
25  
49  
81

## Сколько раз выполняется цикл?

```
a = 1;  
for ( i = 1; i <= 3; i++ ) a = a + 1;
```

a = 4

```
a = 1;  
for ( i = 3; i <= 1; i++ ) a = a + 1;
```

a = 1

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

**зацикливание**

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

a = 4

# Задачи

---

«А»: Найдите все пятизначные числа, которые при делении на 133 дают в остатке 125, а при делении на 134 дают в остатке 111.

«В»: Натуральное число называется **числом Армстронга**, если сумма цифр числа, возведенных в N-ную степень (где N – количество цифр в числе) равна самому числу. Например,  $153 = 1^3 + 5^3 + 3^3$ .  
Найдите все трёхзначные Армстронга.

# Задачи

---

«С»: Натуральное число называется автоморфным, если оно равно последним цифрам своего квадрата.  
Например,  $25^2 = 625$ . Напишите программу, которая получает натуральное число N и выводит на экран все автоморфные числа, не превосходящие N.

**Пример:**

Введите N:

1000

$$1 * 1 = 1$$

$$5 * 5 = 25$$

$$6 * 6 = 36$$

$$25 * 25 = 625$$

$$76 * 76 = 5776$$

# Вложенные циклы

---

**Задача.** Вывести все простые числа в диапазоне от 2 до 1000.

```
сделать для n от 2 до 1000
    если число n простое то
        вывод n
```

нет делителей [2.. n-1]:  
проверка в цикле!



Что значит «простое число»?

# Вложенные циклы

```
for ( n = 2; n <= 1000; n++ )  
{  
    count = 0;  
    for ( k = 2; k < n; k++ )  
        if ( n % k == 0 )  
            count++;  
    if ( count == 0 )  
        printf ( "%d\n", n );  
}
```

ВЛОЖЕННЫЙ ЦИКЛ

# Вложенные циклы

```
for ( i = 1; i <= 4; i++ )  
  {  
    for ( k = 1; k <= i; k++ )  
      {  
        ...  
      }  
  }
```

1	1
2	1
2	2
3	1
3	2
3	3
4	1
4	2
4	3
4	4



Как меняются переменные?



Переменная внутреннего цикла изменяется быстрее!

# Поиск простых чисел – как улучшить?

$$n = k \cdot m, \quad k \leq m \Rightarrow k^2 \leq n \Rightarrow k \leq \sqrt{n}$$

```
while ( k <= sqrt(n) )
{
  ...
}
```



Что плохо?

```
count = 0;
k = 2;
while ( k*k <= n )
{
  if ( n % k == 0 ) count++;
  k++;
}
```



Как ещё улучшить?

```
while ( k*k <= n && (count == 0) ) {
  ...
}
```

# Задачи

---

**«А»:** Напишите программу, которая получает натуральные числа  $A$  и  $B$  ( $A < B$ ) и выводит все простые числа в интервале от  $A$  до  $B$ .

**Пример:**

**Введите границы диапазона:**

10 20

11 13 17 19

**«В»:** В магазине продается мастика в ящиках по 15 кг, 17 кг, 21 кг. Как купить ровно 185 кг мастики, не вскрывая ящики? Сколькими способами можно это сделать?

# Задачи

---

**«С»:** Ввести натуральное число  $N$  и вывести все натуральные числа, не превосходящие  $N$  и делящиеся на каждую из своих цифр.

**Пример:**

Введите  $N$ :

15

1 2 3 4 5 6 7 8 9 11 12 15

# Программирование на языке Си

## § 59. Процедуры

# Зачем нужны процедуры?

```
printf ( "Ошибка программы" );
```

много раз!

```
void Error()  
{  
    printf("Ошибка программы");  
}
```

```
main()  
{  
    int n;  
    scanf ( "%d", &n );  
    if ( n < 0 ) Error();  
    ...  
}
```

ВЫЗОВ  
процедуры

# Что такое процедура?

**Процедура** – вспомогательный алгоритм, который выполняет некоторые действия.

- в момент вызова процедура должна уже быть известна
- в программе может быть **много процедур**
- чтобы процедура заработала, нужно **вызвать** её по имени из основной программы или из другой процедуры

# Процедура с параметрами

Задача. Вывести на экран запись целого числа (0..255) в 8-битном двоичном коде.

много раз!

Алгоритм:

$$178 \Rightarrow 10110010_2$$

?

Как вывести первую цифру?

n =  $\overset{7}{1} \overset{6}{0} \overset{5}{1} \overset{4}{1} \overset{3}{0} \overset{2}{0} \overset{1}{1} \overset{0}{0}_2$  разряды

n / 128

n % 128

?

Как вывести вторую цифру?

n1 / 64

# Процедура с параметрами

**Задача.** Вывести на экран запись целого числа (0..255) в 8-битном двоичном коде.

**Решение:**

```
k = 128;  
while ( k > 0 )  
{  
    printf ( "%d", n / k );  
    n = n % k;  
    k = k / 2;  
}
```

178  $\Rightarrow$  10110010

n	k	ВЫВОД
178	128	1



Результат зависит от n!

# Процедура с параметрами

```
void printBin ( int n )  
{  
    int k;  
    k = 128;  
    while ( k > 0 )  
    {  
        printf ( "%d", n / k );  
        n = n % k;  
        k = k / 2;  
    }  
}
```

локальные  
переменные

**Параметры** – данные,  
изменяющие работу  
процедуры.

```
main ()  
{  
    printBin ( 99 );  
}
```

значение параметра  
(**аргумент**)

# Несколько параметров

---

```
void printSred ( int a, int b )  
{  
    printf ( "%f", (a+b)/2. );  
}
```

# Задачи

---

«А»: Напишите процедуру, которая принимает параметр – натуральное число  $N$  – и выводит на экран линию из  $N$  символов '–'.

**Пример:**

Введите  $N$ :

10

-----

«В»: Напишите процедуру, которая выводит на экран в столбик все цифры переданного ей числа, начиная с первой.

**Пример:**

Введите натуральное число:

1234

1

2

3

4

# Задачи

---

**«С»:** Напишите процедуру, которая выводит на экран запись переданного ей числа в римской системе счисления.

**Пример:**

**Введите натуральное число:**

**2013**

**ММХІІІ**

# Изменяемые параметры

**Задача.** Написать процедуру, которая меняет местами значения двух переменных.

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

передача по  
значению



Процедура работает с копиями переданных значений параметров!

```
main ()  
{  
    int x = 2, y = 3;  
    Swap ( x, y );  
    printf ( "%d %d", x, y );  
}
```



Почему не работает?

2 3

# Изменяемые параметры (Си)

передаются адреса  
переменных

передача по  
адресу

```
void Swap ( int * adrA, int * adrB )  
{  
    int c;  
    c = *adrA; *adrA = *adrB; *adrB = c;  
}
```

значение переменной  
по адресу

**Вызов:**

```
int a, b;  
Swap ( &a, &b );    // правильно  
Swap ( 2, 3 );     // неправильно  
Swap ( &a, b+3 );  // неправильно
```

# Изменяемые параметры (C++)

переменные могут изменяться

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

передача по  
ССЫЛКЕ

## ВЫЗОВ:

```
int a, b;  
Swap ( a, b );    // правильно  
Swap ( 2, 3 );   // неправильно  
Swap ( a, b+3 ); // неправильно
```

# Задачи

---

**«А»:** Напишите процедуру, которая переставляет три переданные ей числа в порядке возрастания.

**Пример:**

**Введите три натуральных числа:**

**10 15 5**

**5 10 15**

**«В»:** Напишите процедуру, которая сокращает дробь вида  $M/N$ . Числитель и знаменатель дроби передаются как изменяемые параметры.

**Пример:**

**Введите числитель и знаменатель дроби:**

**25 15**

**После сокращения: 5/3**

# Задачи

---

**«С»:** Напишите процедуру, которая вычисляет наибольший общий делитель и наименьшее общее кратное двух натуральных чисел и возвращает их через изменяемые параметры.

**Пример:**

Введите два натуральных числа :

**10 15**

**НОД (10 , 15) =5**

**НОК (10 , 15) =30**

# Программирование на языке Си

## § 60. Функции

# Что такое функция?

**Функция** – это вспомогательный алгоритм, который возвращает *значение-результат* (число, символ или объект другого типа).

**Задача.** Написать функцию, которая вычисляет сумму цифр числа.

**Алгоритм:**

```
сумма = 0
пока n != 0
    сумма = сумма + n % 10
    n = n / 10
```

# Сумма цифр числа

```
int sumDigits ( int n )  
{  
    тип результата  
    int sum = 0;  
    while ( n != 0 )  
    {  
        sum += n % 10;  
        n /= 10;  
    }  
    return sum;  
}
```

передача  
результата

```
main()  
{  
    printf ( "%d", sumDigits(12345) );  
}
```

# Использование функций

```
x = 2*sumDigits (n+5) ;  
z = sumDigits (k) + sumDigits (m) ;  
if ( sumDigits (n) % 2 == 0 )  
{  
    printf ( "Сумма цифр чётная\n" ) ;  
    printf ( "Она равна %d", sumDigits (n) ) ;  
}
```



Функция, возвращающая целое число, может использоваться везде, где и целая величина!

# Задачи

---

**«А»:** Напишите функцию, которая находит наибольший общий делитель двух натуральных чисел.

**Пример:**

Введите два натуральных числа:

**7006652 112307574**

$\text{НОД}(7006652, 112307574) = 1234.$

**«В»:** Напишите функцию, которая определяет сумму цифр переданного ей числа.

**Пример:**

Введите натуральное число:

**123**

Сумма цифр числа 123 равна 6.

# Задачи

---

**«С»:** Напишите функцию, которая «переворачивает» число, то есть возвращает число, в котором цифры стоят в обратном порядке.

**Пример:**

**Введите натуральное число:**

**1234**

**После переворота: 4321.**

# Логические функции

---

Задача. Найти все простые числа в диапазоне от 2 до 100.

```
main ()
{
    int i;
    for ( i = 2; i <= 100; i++)
        if ( isPrime(i) )
            printf ( "%d\n", i );
}
```

функция,  
возвращающая  
логическое  
значение (да/нет)

# Функция: простое число или нет?

**?** Какой алгоритм?

```
bool isPrime ( int n )  
{  
    int count = 0, k = 2;  
    while ( k*k <= n && count == 0 )  
    {  
        if ( n % k == 0 )  
            count ++;  
        k ++;  
    }  
    return (count == 0);  
}
```

```
if ( count == 0 )  
    return true;  
else return false;
```

# Логические функции: использование



Функция, возвращающая логическое значение, может использоваться везде, где и логическая величина!

```
scanf ( "%d" , &n ) ;  
while ( isPrime(n) )  
{  
    printf ("простое число\n") ;  
    scanf ( "%d" , &n ) ;  
}
```

# Задачи

---

**«А»:** Напишите логическую функцию, которая определяет, является ли переданное ей число совершенным, то есть, равно ли оно сумме своих делителей, меньших его самого.

**Пример:**

Введите натуральное число:

**28**

Число 28 совершенное.

**Пример:**

Введите натуральное число:

**29**

Число 29 не совершенное.

# Задачи

---

**«В»:** Напишите логическую функцию, которая определяет, являются ли два переданные ей числа взаимно простыми, то есть, не имеющими общих делителей, кроме 1.

**Пример:**

Введите два натуральных числа:

**28 15**

Числа 28 и 15 взаимно простые.

**Пример:**

Введите два натуральных числа:

**28 16**

Числа 28 и 16 не взаимно простые.

# Задачи

---

**«С»:** Простое число называется гиперпростым, если любое число, получающееся из него откидыванием нескольких цифр, тоже является простым. Например, число 733 – гиперпростое, так как и оно само, и числа 73 и 7 – простые. Напишите логическую функцию, которая определяет, верно ли, что переданное ей число – гиперпростое. Используйте уже готовую функцию `isPrime`, которая приведена в учебнике.

## Пример:

Введите натуральное число:

**733**

Число 733 гиперпростое.

## Пример:

Введите натуральное число:

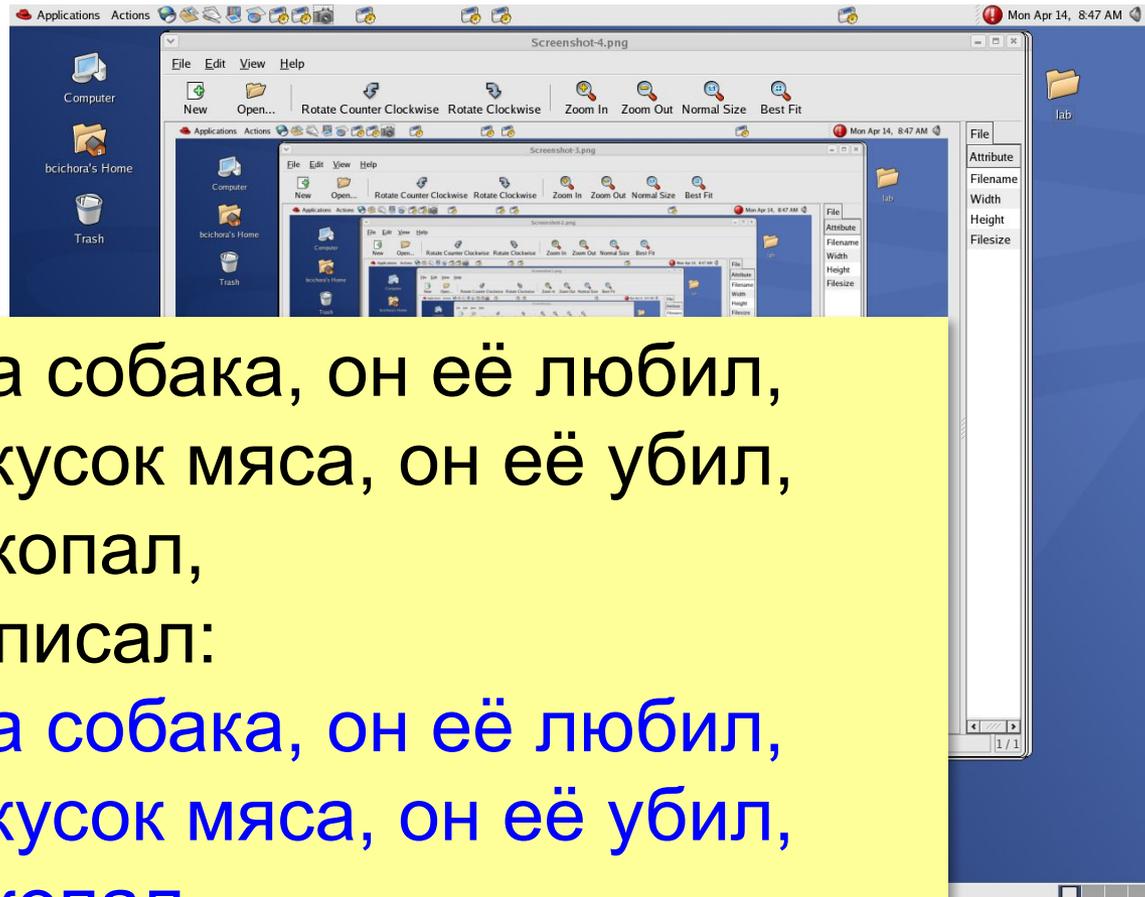
**19**

Число 19 не гиперпростое.

# Программирование на языке Си

## § 61. Рекурсия

# Что такое рекурсия?



У попа была собака, он её любил,  
Она съела кусок мяса, он её убил,  
В землю закопал,  
Надпись написал:  
У попа была собака, он её любил,  
Она съела кусок мяса, он её убил,  
В землю закопал,  
Надпись написал:

...

# Что такое рекурсия?

## Натуральные числа:

- 1 – натуральное число
- если  $n$  – натуральное число, то  $n + 1$  – натуральное число

индуктивное  
определение

**Рекурсия** — это способ определения множества объектов через само это множество на основе заданных простых базовых случаев.

## Числа Фибоначчи:

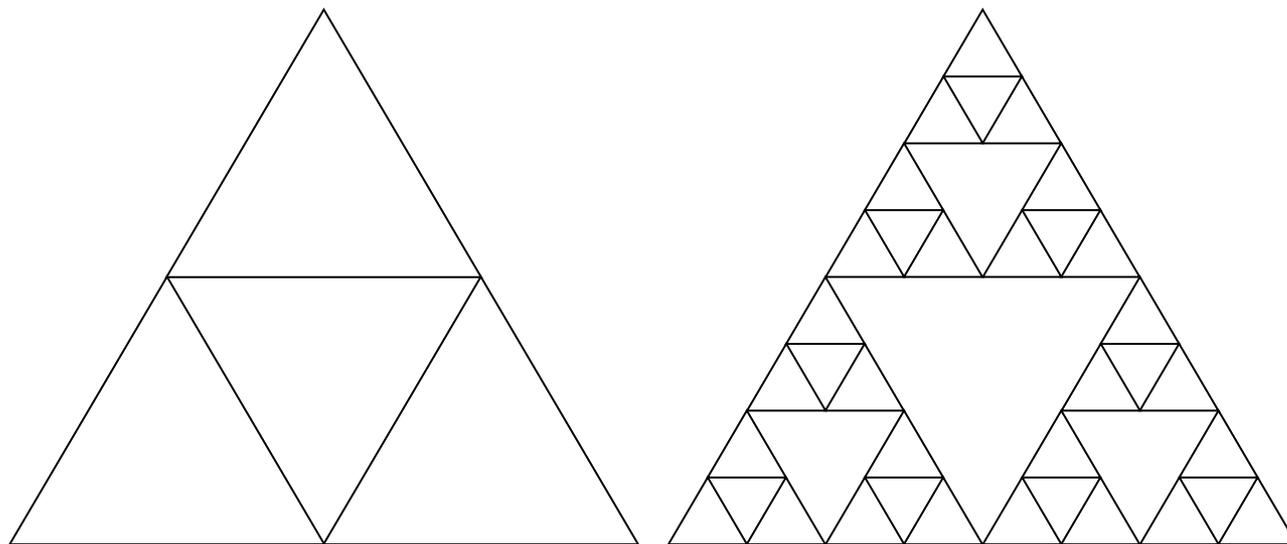
- $F_1 = F_2 = 1$
- $F_n = F_{n-1} + F_{n-2}$  при  $n > 2$

**1, 1, 2, 3, 5, 8, 13, 21, 34, ...**

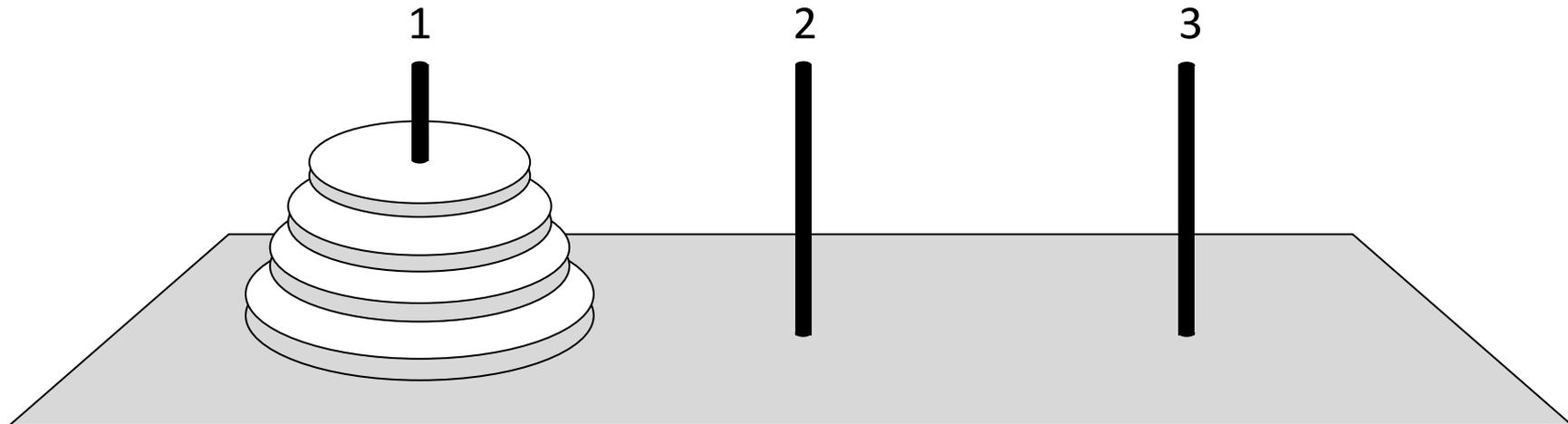
# Фракталы

**Фракталы** – геометрические фигуры, обладающие самоподобием.

**Треугольник Серпинского:**



# Ханойские башни



- за один раз переносится один диск
- класть только меньший диск на больший
- третий стержень вспомогательный

перенести ( $n$ , 1, 3)

перенести ( $n-1$ , 1, 2)

1  $\rightarrow$  3

перенести ( $n-1$ , 2, 3)

# Ханойские башни – процедура

СКОЛЬКО

откуда

куда

```
void Hanoi ( int n, int k, int m )
{
    int p;
    p = 6 - k - m;
    Hanoi ( n-1, k, p );
    printf ( "%d -> %d\n", k, m );
    Hanoi ( n-1, p, m );
}
```

рекурсия

рекурсия

номер вспомогательного  
стержня (1+2+3=6!)

?

Что плохо?

!

**Рекурсия никогда не остановится!**

# Ханойские башни – процедура

Рекурсивная процедура (функция) — это процедура (функция), которая вызывает сама себя напрямую или через другие процедуры и функции.

```
void Hanoi ( int n, int k, int m )
{
    int p;
    if ( n == 0 ) return;
    p = 6 - k - m;
    Hanoi ( n - 1, k, p );
    printf ( "%d -> %d\n", k, m );
    Hanoi ( n - 1, p, m );
}
```

условие выхода из рекурсии

```
main()
{
    Hanoi (4, 1, 3);
}
```

# Вывод двоичного кода числа

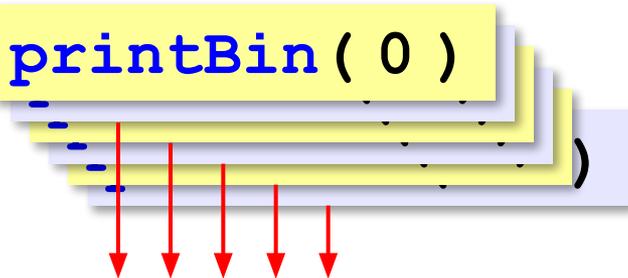
```
void printBin( int n )  
{  
    if ( n == 0 ) return;  
    printBin( n / 2 );  
    printf ( "%d", n % 2 );  
}
```

условие выхода из  
рекурсии

напечатать все  
цифры, кроме  
последней

ВЫВЕСТИ  
последнюю цифру

```
printBin( 0 )
```



**?** Как без рекурсии?

# Вычисление суммы цифр числа

```
int sumDig ( int n )  
{  
    int sum;  
    sum = n % 10;  
    if ( n >= 10 )  
        sum += sumDig ( n / 10 );  
    return sum;  
}
```

последняя цифра

рекурсивный вызов

?

Где условие окончания рекурсии?

sumDig ( 1234 )

4 + sumDig ( 123 )

4 + 3 + sumDig ( 12 )

4 + 3 + 2 + sumDig ( 1 )

4 + 3 + 2 + 1

# Алгоритм Евклида

**Алгоритм Евклида.** Чтобы найти НОД двух натуральных чисел, нужно вычитать из большего числа меньшее до тех пор, пока меньшее не станет равно нулю. Тогда второе число и есть НОД исходных чисел.

```
int NOD ( int a, int b )  
{  
    if ( a == 0 || b == 0 )  
        return a + b;  
    if ( a > b )  
        return NOD ( a - b, b );  
    else return NOD ( a, b - a );  
}
```

условие окончания  
рекурсии

рекурсивные вызовы

# Задачи

---

**«А»:** Напишите рекурсивную функцию, которая вычисляет НОД двух натуральных чисел, используя модифицированный алгоритм Евклида.

**Пример:**

Введите два натуральных числа :

**7006652 112307574**

**НОД (7006652 , 112307574) = 1234 .**

**«В»:** Напишите рекурсивную функцию, которая раскладывает число на простые сомножители.

**Пример:**

Введите натуральное число :

**378**

**378 = 2\*3\*3\*3\*7**

# Задачи

---

**«С»:** Дано натуральное число  $N$ . Требуется получить и вывести на экран количество всех возможных *различных* способов представления этого числа в виде суммы натуральных чисел (то есть,  $1 + 2$  и  $2 + 1$  – это один и тот же способ разложения числа 3). Решите задачу с помощью рекурсивной функции.

**Пример:**

Введите натуральное число:

**4**

Количество разложений: **4**.

# Как работает рекурсия?

**Факториал:**

$$N! = \begin{cases} 1, & N = 1 \\ N \cdot (N-1)!, & N > 1 \end{cases}$$

```
int Fact ( int N )
{
    int F;
    printf ( "-> N=%d\n" , N );
    if ( N <= 1 )
        F = 1;
    else F = N * Fact ( N - 1 );
    printf ( "<- N=%d\n" , N );
    return F;
}
```

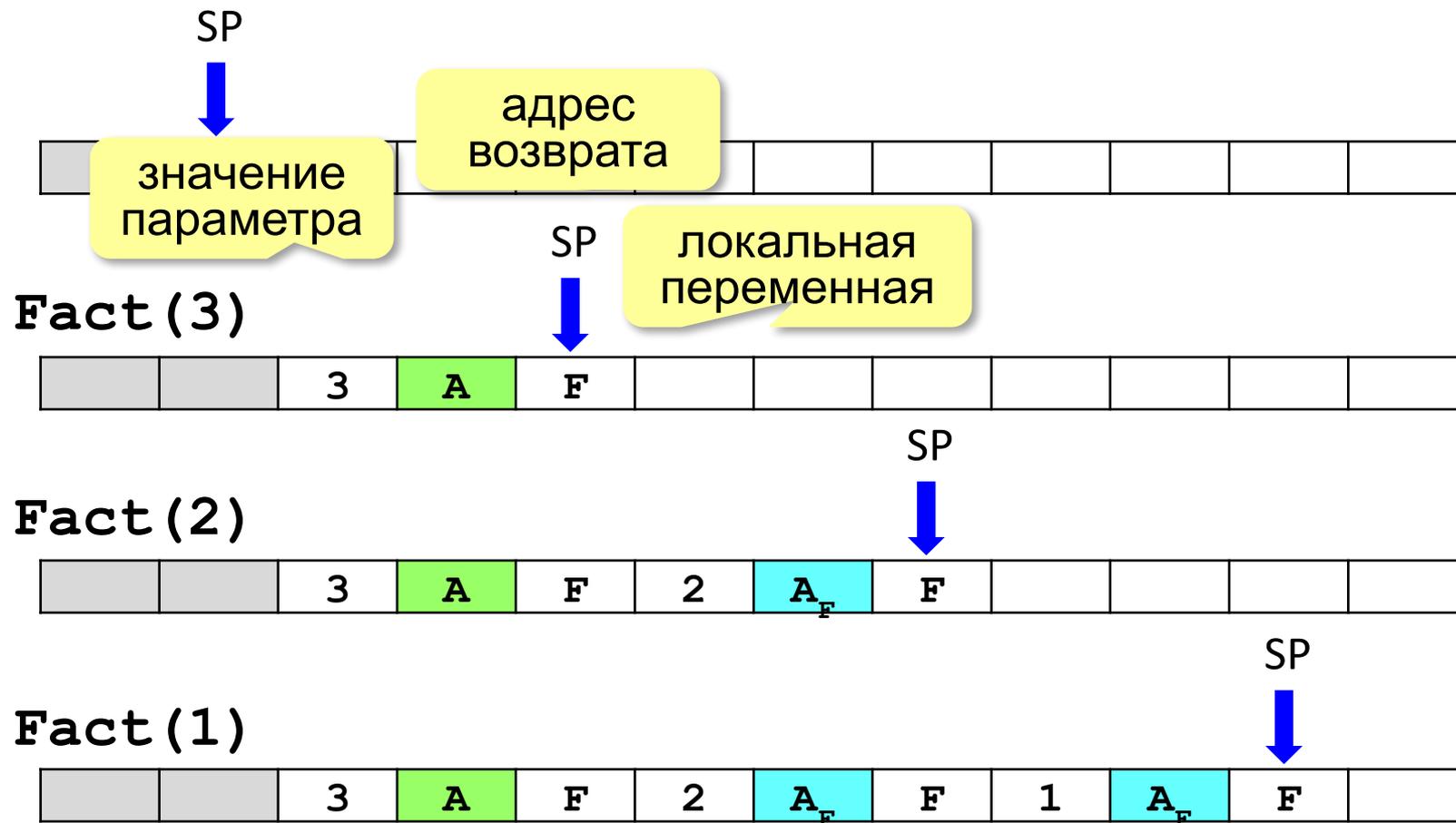
```
-> N = 3
    -> N = 2
        -> N = 1
            <- N = 1
        <- N = 2
    <- N = 3
```



Как сохранить состояние функции перед рекурсивным вызовом?

# Стек

**Стек** – область памяти, в которой хранятся локальные переменные и адреса возврата.



# Рекурсия – «за» и «против»

- с каждым новым вызовом расходуется память в стеке (возможно переполнение стека)
- затраты на выполнение служебных операций при рекурсивном вызове



▪ программа становится более короткой и понятной



▪ возможно переполнение стека

▪ замедление работы



Любой рекурсивный алгоритм можно заменить итерационным!

итерационный  
алгоритм

```
int Fact ( int N )
{
    int F;
    F = 1;
    for (i = 2; i <= N; i++)
        F = F * i;
    return F;
}
```

# Конец фильма

---

**ПОЛЯКОВ Константин Юрьевич**

д.т.н., учитель информатики

ГБОУ СОШ № 163, г. Санкт-Петербург

[kpolyakov@mail.ru](mailto:kpolyakov@mail.ru)

**ЕРЕМИН Евгений Александрович**

к.ф.-м.н., доцент кафедры мультимедийной

дидактики и ИТО ПГГПУ, г. Пермь

[eremin@pspu.ac.ru](mailto:eremin@pspu.ac.ru)

# Источники иллюстраций

---

1. [old-moneta.ru](http://old-moneta.ru)
2. [www.random.org](http://www.random.org)
3. [www.allruletka.ru](http://www.allruletka.ru)
4. [www.lotterypros.com](http://www.lotterypros.com)
5. [logos.cs.uic.edu](http://logos.cs.uic.edu)
6. [ru.wikipedia.org](http://ru.wikipedia.org)
7. иллюстрации художников издательства «Бином»
8. авторские материалы