

Алгорит

Алгоритм – последовательность **действий**, которую **ИСПОЛНИТЕЛЬ** должен выполнить, чтобы решить задачу

Свойства алгоритма

1. Дискретность
2. Понятность
3. Детерминированность
4. Результативность
5. Массовость

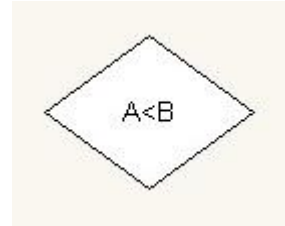
Алгоритмы бывают: линейные, с ветвлениями, циклические.

Способы записи алгоритмов

1. Словесная запись последовательности действий.
2. С помощью математических формул.
3. С помощью схем.
4. С помощью псевдокода.
5. В виде программы.



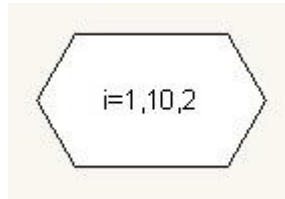
Начало



Ветвления



Ввод/вывод



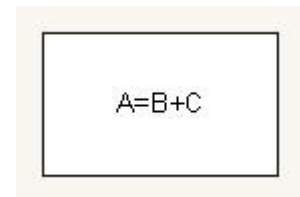
Циклы



Вывод



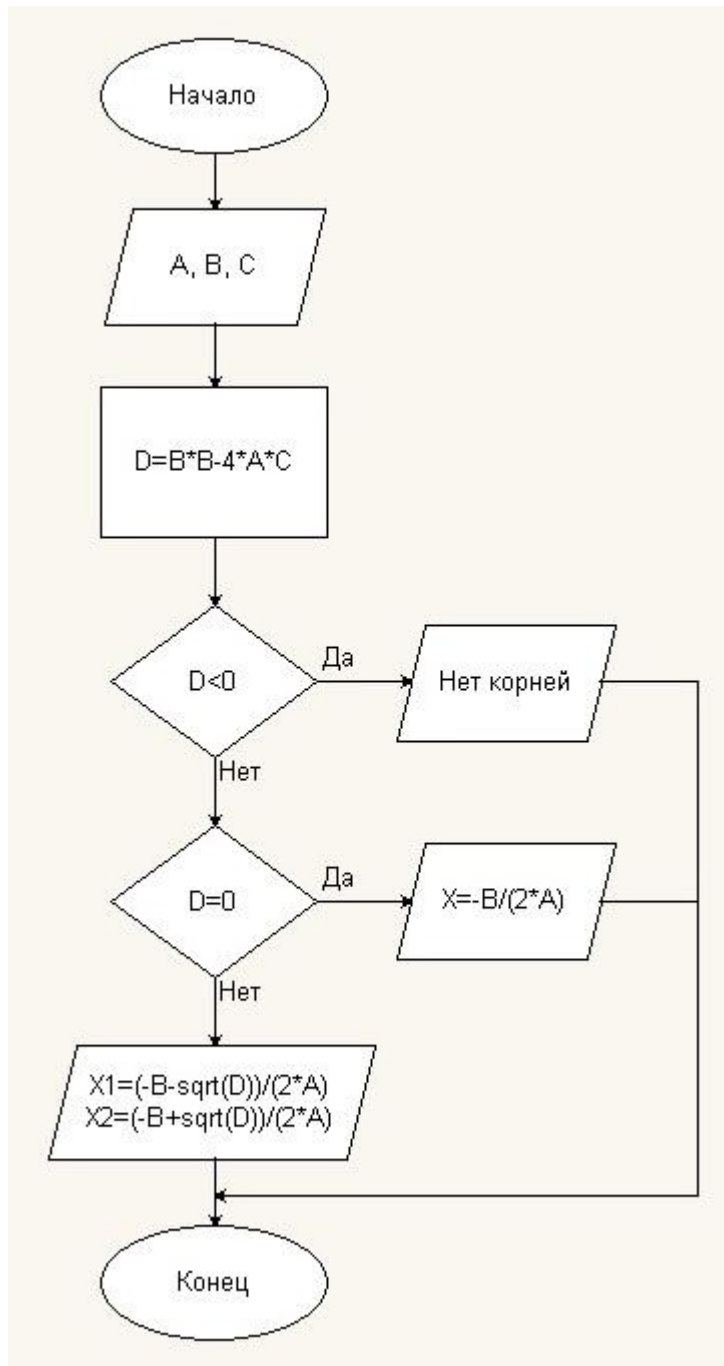
Вызовы подпрограмм



Вычисления

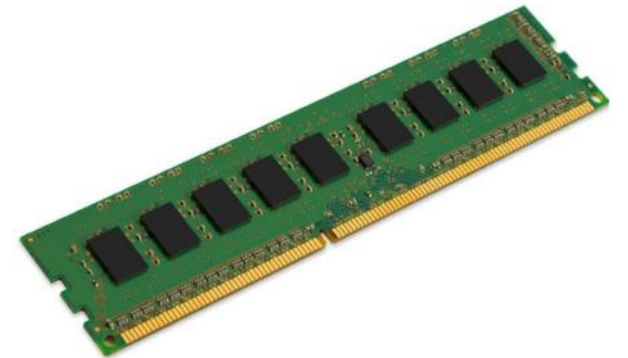
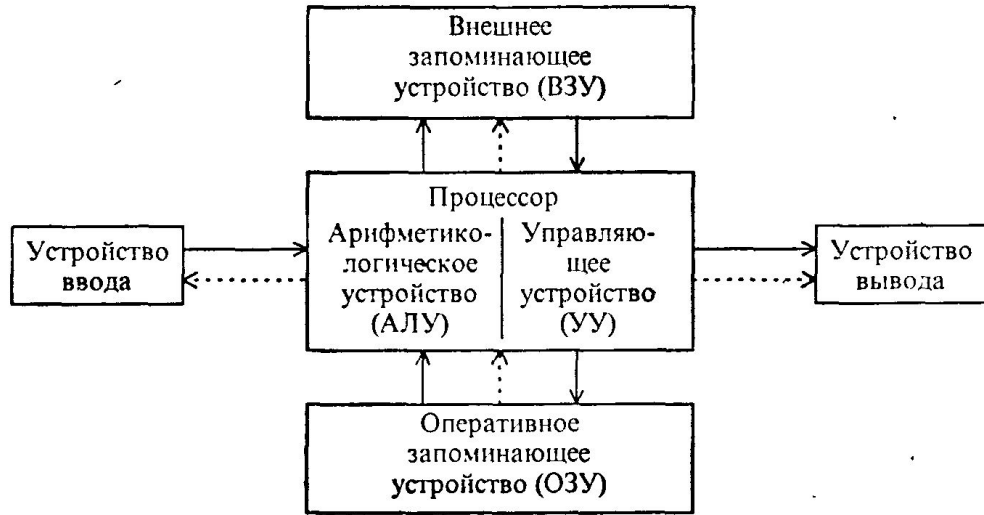


Конец

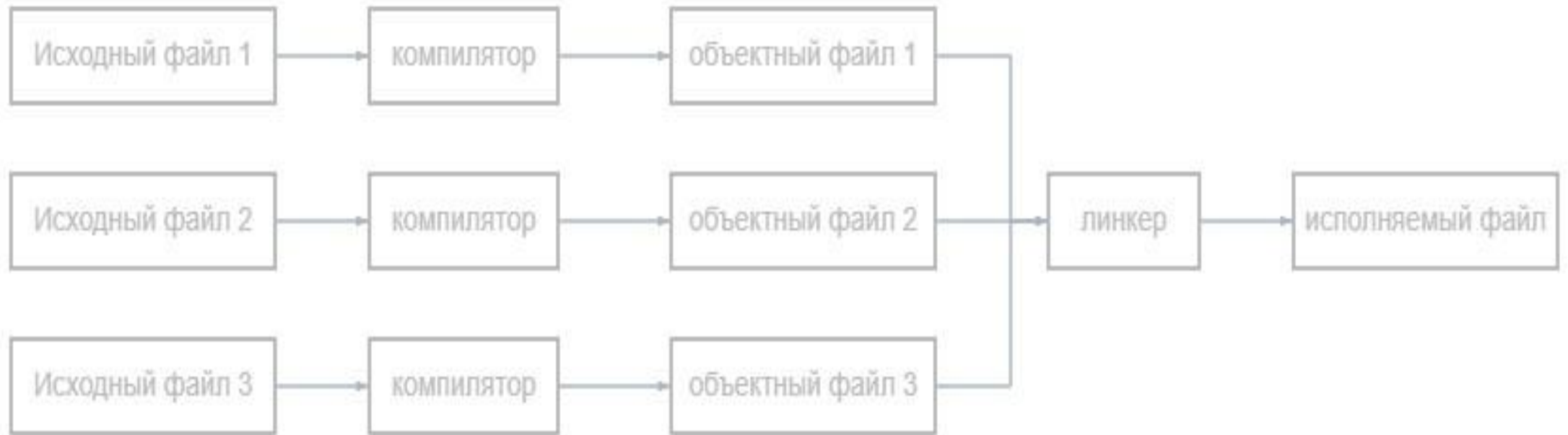


Пример блок-схемы алгоритма решения квадратного уравнения вида $ax^2+bx+c=0$

Устройство компьютера



Получение исполняемого файла



Переменные

e

void – пустой тип

char - символы

int – целые числа

long – целые числа

float – вещественные числа

double – вещественные числа (двойная точность)

bool – логические значения (true и false)

signed или unsigned - только для целочисленных
ТИПОВ

const – для описания констант

Имена могут состоять из букв, цифр, символа подчеркивания

Имена не могут совпадать с ключевыми словами

В именах не могут встречаться пробелы

Имена могут начинаться только с буквы или символа
подчеркивания

Глобальные переменные

инициализируются 0

остальные не инициализируются!

```
#include <iostream>
using namespace std;
int main(int argc, char *argv[]){

    //тут должен быть текст
    программы
    /*
     * это комментарий
     */

    return 0;
}
```

Операци и

+ сложение
- вычитание
* умножение
/ деление
% остаток от деления
++ инкремент
-- декремент
+= сложить и присвоить
-= вычесть и присвоить
*= умножить и присвоить
/= разделить и присвоить
%= взять остаток и
присвоить

= присваивание

sizeof(объект) – вычисление размера объекта в
байтах

делить на 0
нельзя!

< меньше
> больше
<= меньше или
равно
>= больше или
равно
!= не равно
== равно

&& логическое И
|| логическое
ИЛИ
! логическое НЕ

ПОТОКОВЫЙ ВВОД/ВЫВОД

cin – ПОТОК ВВОДА
cout – ПОТОК ВЫВОДА

```
cout << x << "ТЕКСТ" << 'a' << endl;
```

```
cin >> x;
```

\n – новая строка
\r – возврат каретки
\t – горизонтальная табуляция

hex – вывод в шестнадцатеричной системе счисления
dec – вывод в десятичной системе счисления (по умолчанию)

scientific – вывод чисел в экспоненциальной форме
precision(3) – задает значащих цифр при выводе
setw(4) – задает ширину поля вывода

```
#include <iostream>
```

```
#include <iomanip>
```

```
using namespace std;
```

```
int main(){
```

```
    double x=276.23579654219e+16;
```

```
    float y=3.5;
```

```
    cout.precision(8);
```

```
    cout.setf(ios::scientific);
```

```
    cout<<x<<endl;
```

```
    cout<<y<<endl;
```

```
    return 0;
```

```
}
```


Форматирование ввода/вывода

Манипулятор	Описание
endl	Помещение в выходной поток символа конца строки '\n'
dec	Установка основания 10-ой системы счисления
oct	Установка основания 8-ой системы счисления
hex	Установка основания 16-ой системы счисления
setbase	Вывод базовой системы счисления
width(ширина)	Устанавливает ширину поля вывода
fill('символ')	Заполняет пустые знакоместа значением символа
precision(точность)	Устанавливает количество значащих цифр в числе (или после запятой) в зависимости от использования fixed
fixed	Показывает, что установленная точность относится к количеству знаков после запятой
showpos	Показывает знак + для положительных чисел
scientific	Выводит число в экспоненциальной форме
get()	Ожидает ввода символа
getline(указатель, количество)	Ожидает ввода строки символов. Максимальное количество символов ограничено полем количество

```
#include <iostream>
using namespace std;
int main()
{
    double a = -112.234;
    double b = 4.3981;
    int c = 18;
    cout << endl << "double number:" << endl;
    cout << "width(10)" << endl;
    cout.width(10);
    cout << a << endl << b << endl;
    cout << "fill('0')" << endl;
    cout.fill('0');
    cout.width(10);
    cout << a << endl << b << endl;
    cout.precision(5);
    cout << "precision(5)" << endl << a << endl << b << endl;
    cout << "fixed" << endl << fixed << a << endl << b << endl;
    cout << "showpos" << endl << showpos << a << endl << b << endl;
    cout << "scientific" << endl << scientific << a << endl << b << endl;
    cout << endl << "int number:" << endl;
    cout << showbase << hex << c << " " << showbase << oct << c << " ";
    cout << showbase << dec << c << endl;
    cin.get();
    return 0;
}
```

Операторы

- инструкции препроцессора
; - пустой оператор
{операторы} – составной оператор
if – условный оператор
switch – оператор множественного выбора
for – оператор пошагового цикла
while – оператор цикла с предусловием
do – оператор цикла с постусловием
continue – оператор продолжения
break – оператор разрыва
goto – оператор безусловного перехода
return – оператор возврата

константы

132 – десятичная константа
0132 – восьмеричная константа
0x132 – шестнадцатеричная константа
276.23e+16 – константа с плавающей точкой
'a' – символ
"abcd" – строка (оканчивается нулем)

Условный оператор if

```
if(выражение)
{
    операторы
}
```

```
if(выражение)
{
    операторы
    1
}
else{
    операторы
    2
}
```

```
if(выражение)
{
    операторы
    1
}
else if{
    операторы
    2
}
else{
    операторы
    3
}
```

```
switch(выражение){
case КОНСТАНТНОЕ выражение
1:
    операторы 1;
break;
case КОНСТАНТНОЕ выражение
2:
    операторы 2;
break;
case КОНСТАНТНОЕ выражение
3:
    операторы 3;
break;
default:
    операторы 4;
break;
}
```

условный оператор может быть вложенным
нулевое значение выражения интерпретируется как
false
ненулевой значение интерпретируется как true

Циклы

```
for(начальное выражение; условное выражение; выражение  
приращения){  
    операторы;  
}
```

```
while(выражение)  
{  
    операторы;  
}
```

```
do{  
    операторы;  
}while(выражение)  
;
```

циклы могут быть вложенными

break – досрочное завершение цикла

continue – досрочное завершение текущей
итерации

Математическая библиотека
<cmath>

round(a) – округление до целого по правилам арифметики

floor(a) – округление вниз

ceil(a) – округление вверх

trunc(a) – округление в сторону нуля

fabs(a) – модуль

sqrt(a) – квадратный корень

pow(a,b) – возведение в степень

exp(a) – экспонента

log(a) – натуральный логарифм

log10(a) – десятичный логарифм

sin(a) – синус угла (угол в радианах)

cos(a) – косинус угла (угол в радианах)

tan(a) – тангенс угла (угол в радианах)

atan(a) – арктангенс (возвращает угол в радианах)

acos(a) – арккосинус (возвращает угол в радианах)

asin(a) – арксинус (возвращает угол в радианах)

Указател и

Указатель – это
адрес

тип *имя;

int *ptr; //указатель на int

*ptr – разименовывание
указателя

& - операция взятия адреса

```
#include <iostream>
using namespace std;
int main(){
    int a;
    int *c;
    c = &a;
    a=4;
    cout<<a<<endl;
    *c=9;
    cout<<a<<endl;
    return 0;
}
```

Массивы

Массив — это набор однотипных данных, расположенных в памяти в непрерывном отрезке. Массивы бывают одномерные и многомерные, статические и динамические.

Индекс	0	1	2	3	4	6
Массив	45	51	8	3	56	21

В

Динамический массив — это массив, размер которого определяется во время выполнения программы.

Размер массива — это общее количество ячеек, входящих в массив.

Размерность массива — это количество измерений массива (один для одномерных массивов, два для двумерных и т. д.)

Массивы используют для удобства организации доступа и обработки большого количества однотипных данных. Организация данных в виде массивов позволяет эффективно использовать циклы для их обработки.

Каждая ячейка массива имеет свой номер (индекс) или несколько номеров (индексов) для многомерных массивов.

Имя массива — это указатель на его элемент с индексом 0

Одномерные массивы

тип имя[размер];
наименьший индекс 0
наибольший индекс
(размер-1)

статический массив
int mas[10];

динамический массив
тип *имя = new тип[размер];
int *mas = new int[10];
delete []mas;

Массивы автоматически не
инициализируются!

```
#include <iostream>
using namespace std;
int main(){
    int mas[10];
    for(int i=0; i<10; i++){
        mas[i]=rand();
    }
    for(int i=0; i<10; i++){
        cout<<mas[i]<<endl;
    }
    return 0;
}
```


Многомерные массивы

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(){
    int mas[5][4];
    for(int i=0; i<5; i++){
        for(int j=0; j<4; j++){
            mas[i][j]=rand();
        }
    }
    for(int i=0; i<5; i++){
        for(int j=0; j<4; j++){
            cout<<setw(10)<<mas[i][j];
        }
        cout<<endl;
    }
    return 0;
}
```

```
int main(){
    int **mas = new int*[5];
    for(int i=0; i<5; i++) mas[i]=new int[4];
    for(int i=0; i<5; i++){
        for(int j=0; j<4; j++){
            mas[i][j]=rand();
        }
    }
    for(int i=0; i<5; i++){
        for(int j=0; j<4; j++){
            cout<<setw(10)<<mas[i][j];
        }
        cout<<endl;
    }
    for(int i=0; i<5; i++) delete [] mas[i];
    delete [] mas;
    return 0;
}
```

Расположение в памяти многомерных массивов

Двухмерный

1,1	1,2	1,3
2,1	2,2	2,3
3,1	3,2	3,3

Расположение в
памяти

1,1	1,2	1,3	2,1	2,2	2,3	3,1	3,2	3,3
-----	-----	-----	-----	-----	-----	-----	-----	-----

Символьные строки

Строка – это массив символов, который заканчивается нулевым

СИМВОЛОМ

```
const char str[]="это строка"; //константная
```

строка

```
char str[50]; //массив длиной 50 символов
```

```
#include <iostream>
using namespace std;
int main(){
    const char str[]="this is string";
    cout<<str<<endl;
    return 0;
}
```

```
#include <iostream>
using namespace std;
int main(){
    char str[50];
    cin.getline(str, 50);
    cout<<str<<endl;
    return 0;
}
```

Тип
std::string

```
#include <iostream>
#include <string>

using namespace std;

int main(){
    string str;
    str="This is string";
    cout<<str<<endl;
    return 0;
}
```

```
#include <iostream>
#include <string>

using namespace std;

int main(){
    string name;
    string age;
    cout<<"Enter your full name: ";
    cin>>name;
    cout<<"Enter your age: ";
    cin>>age;
    cout <<"Your name is "<<name<<" and your age is "<<age;
    return 0;
}
```

Оператор извлечения >> возвращает символы из входного потока данных только до первого пробела.

Все остальные символы остаются в cin, ожидая следующего извлечения.

Использование std::getline

```
#include <iostream>
#include <string>

using namespace std;

int main(){
    string name;
    string age;
    cout<<"Enter your full name: ";
    getline(cin,name);
    cout<<"Enter your age: ";
    getline(cin,age);
    cout <<"Your name is "<<name<<" and your age is "<<age;
    return 0;
}
```

Использование std::getline совместно с std::cin

```
#include <iostream>
#include <string>

using namespace std;

int main(){
    string name;
    int age;
    cout<<"Enter your age: ";
    cin>>age;
    cout<<"Enter your full name: ";
    getline(cin,name);
    cout <<"Your name is "<<name<<" and your age is "<<age;
    return 0;
}
```

Поток cin захватывает число вместе с символом новой строки. Число извлекается в переменную, а символ новой строки остается в потоке ввода. Поэтому getline сразу заканчивает ввод.

```
#include <iostream>
#include <string>

using namespace std;

int main(){
    string name;
    int age;
    cout<<"Enter your age: ";
    cin>>age;
    cin.ignore(100,'\n');
    cout<<"Enter your full name: ";
    getline(cin,name);
    cout <<"Your name is "<<name<<" and your age is "<<age;
    return 0;
}
```


Операции с std::string

- = присваивание значения
- + конкатенация двух строк (строки и символа)
- += добавление в конец одной строки другой строки (символа)
- == посимвольное сравнение на равенство
- != посимвольное сравнение на неравенство
- <, >, <=, >= лексикографическое сравнение

Методы

std::string

- size – возвращает длину строки (s.size())
- resize – изменяет длину строки (s.resize(n) или s.resize(n,c))
- clear – очищает строку (s.clear())
- empty – возвращает true, если строка пуста (s.empty())
- push_back – добавляет в конец строки символ (s.push_back(c))
- append – добавляет в конец строки символ, другую строку или фрагмент другой строки
- erase – удаляет символы из строки
- insert – вставляет в середину строки символы, другую строку или фрагмент другой строки
- substr – возвращает подстроку
- replace – заменяет фрагмент строки
- find – ищет в данной строке первое вхождение другой строки
- rfind – ищет в данной строке последнее вхождение другой строки
- c_str – возвращает указатель на область памяти, в которой хранятся символы

<https://server.179.ru/tasks/cpp/total/161.html>

```
#include <iostream>
#include <string>

using namespace std;

int main(){
    string str;
    cout<<"На каком языке написана эта
программа?"<<endl;
    cin>>str;
    if(str=="C++")cout<<"Верно!"<<endl;
    else cout<<"Вы ошиблись!"<<endl;
    return 0;
}
```

```
#include <iostream>
#include <cstring>

using namespace std;

int main(){
    char str[100];
    cout<<"На каком языке написана эта
программа?"<<endl;
    cin>>str;
    if(strcmp(str,"C++")==0)cout<<"Верно!"<<endl;
    else cout<<"Вы ошиблись!"<<endl;
    return 0;
}
```

Подпрограммы

Подпрограмма — это часть программы, оформленная как самостоятельная единица, которая может вызываться из других частей программы.

```
тип_функции имя_функции  
(аргументы){  
    тело  
}
```

```
void func(); //прототип функции  
void func(){  
    cout<<"Мы внутри func"<<endl;  
    return;  
}
```

```
#include <iostream>  
  
using namespace std;  
  
void func(){  
    cout<<"Мы внутри func"<<endl;  
    return;  
}  
  
int main(){  
    cout<<"Мы внутри main"<<endl;  
    func();  
    cout<<"Мы внутри main"<<endl;  
    return 0;  
}
```

Передача аргументов через глобальные переменные

```
#include <iostream>

using namespace std;

int x, y, s;

void sum(){
    s=x+y;
    return;
}

int main(){
    cin>>x>>y;
    sum();
    cout<<"Сумма="<<s<<endl;
    return 0;
}
```

Передача аргументов по значению

```
#include <iostream>

using namespace std;

int sum(int a, int b){
    return a+b;
}

int main(){
    int x, y, s;
    cin>>x>>y;
    s=sum(x, y);
    cout<<"Сумма="<<s<<endl;
    return 0;
}
```

```
#include <iostream>

using namespace std;

void func(int arg){
    arg=arg+5;
    return;
}

int main(){
    int a=3;
    cout<<a<<endl;
    func(a);
    cout<<a<<endl;
    return 0;
}
```

Передача аргументов по ссылке

```
#include <iostream>

using namespace std;

void func(int &arg){
    arg=arg+5;
    return;
}

int main(){
    int a=3;
    cout<<a<<endl;
    func(a);
    cout<<a<<endl;
    return 0;
}
```

Передача аргументов через указатель

```
#include <iostream>

using namespace std;

void func(int* arg){
    *arg=*arg+5;
    return;
}

int main(){
    int a=3;
    cout<<a<<endl;
    func(&a);
    cout<<a<<endl;
    return 0;
}
```

1. **Указатель** может быть переназначен любое количество раз, в то время как **ссылка** после привязки не может быть перемещена на другую ячейку памяти.
2. **Указатели** могут указывать "в никуда" (быть равными NULL), в то время как **ссылка** всегда указывает на определенный объект.

Передача массива в функцию

```
#include <iostream>
using namespace std;
int mas[10];
int sum(){
    int s=0;
    for(int i=0; i<10; i++)s+=mas[i];
    return s;
}
int main(){
    for(int i=0; i<10; i++){
        mas[i]=rand();
    }
    for(int i=0; i<10; i++){
        cout<<mas[i]<<endl;
    }
    cout<<"Сумма="<<sum()<<endl;
    return 0;
}
```

```
#include <iostream>
using namespace std;
int sum(int mas[]){ //размер указывать не обязательно
    int s=0;
    for(int i=0; i<10; i++)s+=mas[i];
    return s;
}
int main(){
    int mas[10];
    for(int i=0; i<10; i++){
        mas[i]=rand();
    }
    for(int i=0; i<10; i++){
        cout<<mas[i]<<endl;
    }
    cout<<"Сумма="<<sum(mas)<<endl;
    return 0;
}
```

```
#include <iostream>
using namespace std;
int sum(int mas[], int size){
    int s=0;
    for(int i=0; i<size; i++)s+=mas[i];
    return s;
}
int main(){
    int mas[10];
    for(int i=0; i<10; i++){
        mas[i]=rand();
    }
    for(int i=0; i<10; i++){
        cout<<mas[i]<<endl;
    }
    cout<<"Сумма="<<sum(mas, sizeof(mas)/sizeof(int))<<endl;
    return 0;
}
```

```
#include <iostream>
using namespace std;
int sum(int mas[], int size){
    int s=0;
    for(int i=0; i<size; i++)s+=mas[i];
    return s;
}
int main(){
    int mas[10];
    for(int i=0; i<10; i++){
        mas[i]=rand();
    }
    for(int i=0; i<10; i++){
        cout<<mas[i]<<endl;
    }
    cout<<"Сумма="<<sum(mas, 5)<<endl;
    return 0;
}
```

```

#include <iostream>
using namespace std;
int sum(int mas[], int size){
    int s=0;
    for(int i=0; i<10; i++)mas[i]=rand();
    for(int i=0; i<size; i++)s+=mas[i];
    return s;
}
int main(){
    int mas[10];
    int s;
    for(int i=0; i<10; i++)mas[i]=0;
    for(int i=0; i<10; i++){
        cout<<mas[i]<<endl;
    }
    cout<<endl<<endl;
    s=sum(mas, sizeof(mas)/sizeof(int));
    for(int i=0; i<10; i++){
        cout<<mas[i]<<endl;
    }
    cout<<"Сумма="<<s<<endl;
    return 0;
}

```

Массивы всегда передаются по ссылке!

```

#include <iostream>
using namespace std;
void print_mas(int matr[][4]);
void init_mas(int matr[][4]);
int main(){
    int matr[5][4];
    for(int i=0; i<5; i++){
        for(int j=0; j<4; j++)matr[i][j]=0;
    }
    print_mas(matr);
    cout<<endl;
    init_mas(matr);
    print_mas(matr);
    return 0;
}
void init_mas(int matr[][4]){
    for(int i=0; i<5; i++){
        for(int j=0; j<4; j++)matr[i][j]=i*j;
    }
    return;
}
void print_mas(int matr[][4]){
    for(int i=0; i<5; i++){
        for(int j=0; j<4; j++)cout<<matr[i][j]<<" ";
        cout<<endl;
    }
    return;
}
}

```

Передача матрицы в функцию

```
#include <iostream>
#include <iomanip>
using namespace std;
int mas[5][4];
int sum(int row, int col){
    int s=0;
    for(int i=0; i<row; i++){
        for(int j=0; j<col; j++)s+=mas[i][j];
    }
    return s;
}
int main(){
    for(int i=0; i<5; i++){
        for(int j=0; j<4; j++)mas[i][j]=rand();
    }
    for(int i=0; i<5; i++){
        for(int j=0; j<4; j++)cout<<setw(10)<<mas[i][j];
        cout<<endl;
    }
    cout<<"Сумма="<<sum(5, 4)<<endl;
    return 0;
}
```

```
#include <iostream>
#include <iomanip>
using namespace std;
int sum(int** mas, int row, int col){
    int s=0;
    int* array=(int*)mas;
    for(int i=0; i<row; i++){
        for(int j=0; j<col; j++)s+=array[i*row+j];
    }
    return s;
}
int main(){
    int mas[5][4];
    for(int i=0; i<5; i++){
        for(int j=0; j<4; j++)mas[i][j]=rand();
    }
    for(int i=0; i<5; i++){
        for(int j=0; j<4; j++)cout<<setw(10)<<mas[i][j];
        cout<<endl;
    }
    cout<<"Сумма="<<sum((int**)mas, 5, 4)<<endl;
    return 0;
}
```

Рекурсивные функции

```
#include <iostream>
#include <iomanip>
using namespace std;
int fact(int n){
    if(n>2)return fact(n-1)*n;
    else return n;
}
int main(){
    int n;
    cin>>n;
    cout<<n<<"!="<<fact(n)<<endl;
    return 0;
}
```

```
#include <iostream>
#include <iomanip>
using namespace std;
int fact(int n){
    int f;
    cout<<"Функция вызвана с аргументом "<<n<<endl;
    if(n>2)f=fact(n-1)*n;
    else f=n;
    cout<<"результат "<<f<<endl;
    return f;
}
int main(){
    int n, f;
    cin>>n;
    f=fact(n);
    cout<<n<<"!="<<f<<endl;
    return 0;
}
```

Перегрузка функций

1. По возвращаемому значению
2. По количеству аргументов
3. По типам аргументов

```
#include <iostream>
#include <iomanip>
using namespace std;
int sum(int a, int b){
    return a+b;
}
int sum(int a, int b, int c){
    return a+b+c;
}
int main(){
    int a=7, b=5, c=4;
    cout<<a<<'+ '<<b<< '='<<sum(a, b)<<endl;
    cout<<a<<'+ '<<b<<'+ '<<c<< '='<<sum(a, b, c)<<endl;
    return 0;
}
```

Отбрасывание
const.

```
#include <iostream>
```

```
using namespace std;
```

```
int main() {
```

```
    const volatile int w = 10;
```

```
    int *wr = const_cast <int *> (&w);
```

```
    *wr = 20;
```

```
    std::cout << w << std::endl;
```

```
    return 0;
```

```
}
```

**Так делать НЕ
НАДО!**

Файловый ввод/вывод

```
#include <fstream>
using namespace std;
int main(){
    ofstream file;      //создаем объект типа ofstream
    file.open("file.txt"); //открываем файл
    file<<"Hello world!"; //вывод в файл
    file.close();      //закрываем файл
    return 0;
}
```

Флаги открытия файлов:

ios::in – открыть файл для чтения

ios::out – открыть файл для записи

ios::ate – при открытии переместить указатель в конец файла

ios::app – открыть файл для записи в конец файла

ios::trunc – удалить содержимое файла, если он существует

ios::binary – открытие файла в двоичном режиме


```
#include <fstream>
using namespace std;
int main(){
    ofstream file;
    file.open("file.txt");
    file<<"Hello world!";
    file.close();
    return 0;
}
```

```
#include <fstream>
using namespace std;
int main(){
    fstream file;
    file.open("file.txt",ios::out);
    file<<"Hello world!";
    file.close();
    return 0;
}
```

Проверка существования
файла

```
#include <fstream>
#include <iostream>
using namespace std;
int main(){
    fstream file;
    file.open("file.txt",ios::in);
    if(file){
        cout<<"Файл уже существует!"<<endl;
    }
    else{
        file.close();
        file.open("file.txt",ios::out);
        file<<"Hello world!";
    }
    file.close();
    return 0;
}
```

Запись в конец существующего
файла

```
#include <fstream>  
#include <iostream>
```

```
using namespace std;
```

```
int main(){  
    fstream file;  
    file.open("file.txt",ios::out|ios::app);  
    file<<"Hello world!";  
    file.close();  
    return 0;  
}
```

Перезапись
файла

```
#include <fstream>  
#include <iostream>
```

```
using namespace std;
```

```
int main(){  
    fstream file;  
    file.open("file.txt",ios::out|ios::trunc);  
    file<<"Hello world!";  
    file.close();  
    return 0;  
}
```

Чтение массива из файла

```
#include <fstream>
#include <iostream>

using namespace std;

int main(){
    fstream file;
    int mas[10];
    file.open("file.txt",ios::in);
    for(int i=0; i<10; i++)file>>mas[i];
    file.close();
    for(int i=0; i<10; i++){
        cout<<mas[i]<<endl;
    }
    return 0;
}
```

```
#include <fstream>
#include <iostream>

using namespace std;

int main(){
    fstream file;
    int mas[5][4];
    file.open("file.txt",ios::in);
    for(int i=0;i<5;i++){
        for(int j=0;j<4;j++)file>>mas[i][j];
    }
    file.close();
    for(int i=0;i<5;i++){
        for(int j=0;j<4;j++)cout<<mas[i][j]<<" ";
        cout<<endl;
    }
    return 0;
}
```

Чтение строк из файла

```
#include <fstream>
#include <iostream>
```

```
using namespace std;
```

```
int main(){
    fstream file;
    string str;
    file.open("file.txt",ios::in);
    file>>str;
    file.close();
    cout<<str;
    return 0;
}
```

```
#include <fstream>
#include <iostream>
```

```
using namespace std;
```

```
int main(){
    fstream file;
    string str;
    file.open("file.txt",ios::in);
    getline(file,str);
    file.close();
    cout<<str;
    return 0;
}
```

Произвольный доступ к файлу

```
#include <fstream>
#include <iostream>
using namespace std;
int main(){
    fstream file;
    string str;
    file.open("file.txt",ios::in|ios::out);
    file.seekg(6, ios::beg); //курсор чтения в позицию 6 от начала файла
    cout<<"Позиция для чтения "<<file.tellg()<<endl;
    file>>str;
    cout<<str<<endl;
    file.clear(); //так как был достигнут конец файла при чтении
    file.seekp(6, ios::beg);
    cout<<"Позиция для записи "<<file.tellp()<<endl;
    file<<"peoples!";
    file.close();
    return 0;
}
```

ios::beg – начало файла
ios::cur – текущее
положение
ios::end – конец файла

Определение достижения конца файла

```
#include <fstream>
#include <iostream>
using namespace std;
int main(){
    fstream file;
    int mas[20];
    int n=0;
    file.open("file.txt",ios::in);
    for(int i=0; i<20; i++){
        if(!file.eof()){
            file>>mas[i];
            n++;
        }
        else break;
    }
    file.close();
    for(int i=0; i<n; i++)cout<<mas[i]<<endl;
    return 0;
}
```

```
#include <fstream>
#include <iostream>
using namespace std;
int main(){
    fstream file;
    int mas[10];
    file.open("file.txt",ios::in);
    for(int i=0; i<10; i++){
        file>>mas[i];
        if(file.eof() && i<9){
            cout<<"В файле недостаточно
данных!"<<endl;
            return -1;
        }
    }
    file.close();
    for(int i=0; i<10; i++)cout<<mas[i]<<endl;
    return 0;
}
```

Двоичные файлы

```
#include <fstream>
#include <iostream>

using namespace std;

int main(){
    fstream file;
    int mas[10];
    file.open("file.txt",ios::in|ios::binary);
    for(int i=0; i<10; i++){
        mas[i]=file.get(); //читает байт из файла
    }
    file.close();
    for(int i=0; i<10; i++)cout<<mas[i]<<endl;
    return 0;
}
```

```
#include <fstream>
#include <iostream>

using namespace std;

int main(){
    fstream file;
    int mas[10];
    file.open("file.txt",ios::in|ios::binary);
    for(int i=0; i<10; i++){
        file.read((char*)&mas[i],sizeof(mas[i]));
    }
    file.close();
    for(int i=0; i<10; i++)cout<<hex<<mas[i]<<endl;
    return 0;
}
```

```
#include <fstream>
#include <iostream>

using namespace std;

int main(){
    fstream file;
    file.open("file.txt",ios::out|ios::binary);
    for(int i=1; i<255; i++){
        file.put(i);
    }
    file.close();
    return 0;
}
```


Битовые операции

& битовое И
| битовое ИЛИ
^ битовое исключающее ИЛИ
~ битовое НЕ
<< битовый сдвиг влево
>> битовый сдвиг вправо
&= битовое И и присвоить
|= битовое ИЛИ и присвоить
^= битовое исключающее ИЛИ и
присвоить

```
#include <iostream>
using namespace std;
void PrintBin(unsigned int num);
void PrintBin(unsigned char num);
int main(){
    unsigned int x;
    x=0xFFAA5500;
    PrintBin(x);
    return 0;
}
void PrintBin(unsigned int num){
    cout<<"0b";
    for(unsigned int i=0; i<sizeof(num)*8; i++){
        if(num & 0x80000000)cout<<'1';
        else cout<<'0';
        num<<=1;
    }
    cout<<endl;
    return;
}
void PrintBin(unsigned char num){
    cout<<"0b";
    for(unsigned int i=0; i<8; i++){
        if(num & 0x80)cout<<'1';
        else cout<<'0';
        num<<=1;
    }
    cout<<endl;
    return;
}
```

Битовое И

Применяется обычно для сброса в ноль битов в переменной

```
int main(){
    unsigned char x,y,z;
    x=0xAA; //переменная
    y=0x70; //маска
    PrintBin(x);
    cout<<"AND"<<endl;
    PrintBin(y);
    cout<<endl;
    z=x&y;
    PrintBin(z);
    cout<<endl;
    cout<<(unsigned int)x<<endl<<"and"<<endl<<(unsigned int)y<<endl<<(unsigned int)z<<endl;
    return 0;
}
```

Битовое ИЛИ

Применяется обычно для установки в единицу битов в переменной

```
int main(){
    unsigned char x,y,z;
    x=0xAA; //переменная
    y=0x70; //маска
    PrintBin(x);
    cout<<"OR"<<endl;
    PrintBin(y);
    cout<<endl;
    z=x|y;
    PrintBin(z);
    cout<<endl;
    cout<<(unsigned int)x<<endl<<"or"<<endl<<(unsigned int)y<<endl<<(unsigned int)z<<endl;
    return 0;
}
```

Битовое исключяющее ИЛИ

Применяется обычно для инверсии битов в переменной

```
int main(){
    unsigned char x,y,z;
    x=0xAA; //переменная
    y=0x70; //маска
    PrintBin(x);
    cout<<"XOR"<<endl;
    PrintBin(y);
    cout<<endl;
    z=x^y;
    PrintBin(z);
    cout<<endl;
    cout<<(unsigned int)x<<endl<<"xor"<<endl<<(unsigned int)y<<endl<<(unsigned int)z<<endl;
    return 0;
}
```

Битовое НЕ

Применяется для инверсии всех битов в переменной

```
int main(){
    unsigned char x,z;
    x=0xAA;
    PrintBin(x);
    z=~x;
    cout<<"NOT"<<endl;
    PrintBin(z);
    cout<<endl;
    cout<<(unsigned int)x<<endl<<"not"<<endl<<(unsigned int)z<<endl;
    return 0;
}
```

Установка
бита

```
int main(){
    unsigned char x;
    x=0x00;
    PrintBin(x);
    x|=(1<<3);
    PrintBin(x);
    return 0;
}
```

Сброс
бита

```
int main(){
    unsigned char x;
    x=0xFF;
    PrintBin(x);
    x&=~(1<<3);
    PrintBin(x);
    return 0;
}
```

Инверсия
бита

```
int main(){
    unsigned char x;
    x=0x55;
    PrintBin(x);
    x^=(1<<3);
    PrintBin(x);
    return 0;
}
```

БИТОВЫЕ СДВИГИ

Сдвиг влево эквивалентен умножению на степень

```
двойки() {  
    unsigned char x, y;  
    x=25;  
    PrintBin(x);  
    y=x<<2;  
    PrintBin(y);  
    cout<<(unsigned int)x<<endl<<(unsigned int)y<<endl;  
    return 0;  
}
```

Сдвиг вправо эквивалентен делению на степень

```
двойки() {  
    unsigned char x, y;  
    x=100;  
    PrintBin(x);  
    y=x>>2;  
    PrintBin(y);  
    cout<<(unsigned int)x<<endl<<(unsigned int)y<<endl;  
    return 0;  
}
```


Объединени я

Позволяют интерпретировать содержимое памяти по-разному в различных местах программы

```
union integer{
    unsigned int num;
    unsigned char mas[4];
};

int main(){
    integer number;
    number.num=0xFFAA0055;
    PrintBin(number.num);
    PrintBin(number.mas[0]);
    PrintBin(number.mas[3]);
    number.mas[3]=0;
    PrintBin(number.num);
    return 0;
}
```

Структур ы

```
#include <iostream>
using namespace std;
```

```
struct time{
    int hours;
    int min;
    int sec;
};
```

```
void TimePrint(time t);
time TimeSet(int num);
```

```
int main(){
    time t;
    int x;
    cin>>x;
    t=TimeSet(x);
    TimePrint(t);
    return 0;
}
```

```
void TimePrint(time t){
    cout<<t.hours<<':'<<t.min<<':'<<t.sec<<endl;
    return;
}
```

```
time TimeSet(int num){
    time t;
    t.sec=num%60;
    num/=60;
    t.min=num%60;
    num/=60;
    t.hours=num;
    return t;
}
```

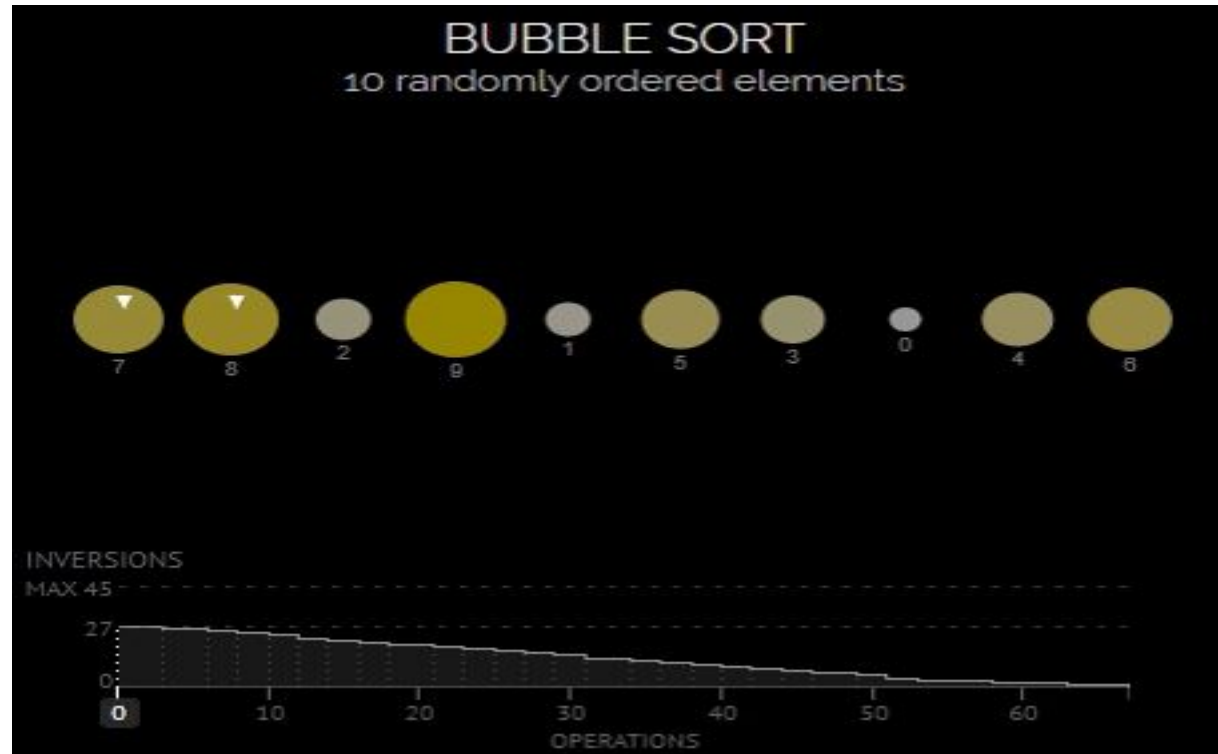
Классы

```
class Time{
    public:
    int hour;
    int min;
    int sec;
    void print();
    Time();
    Time(int num);
    Time operator+(int num);
};
Time::Time(){
    hour=0;
    min=0;
    sec=0;
    return;
}
Time::Time(int num){
    sec=num%60;
    num/=60;
    min=num%60;
    num/=60;
    hour=num;
}
Time Time::operator+(int num){
    return Time(sec+min*60+hour*3600+num);
}
void Time::print(){
    cout<<hour<<':'<<min<<':'<<sec<<endl;
    return;
}
```

```
int main(){
    int num;
    cin>>num;
    Time time(num);
    time.print();
    cin>>num;
    time=time+num;
    time.print();
    return 0;
}
```

Пузырьковая сортировка

Обходим массив от начала до конца, попутно меняя местами неотсортированные соседние элементы

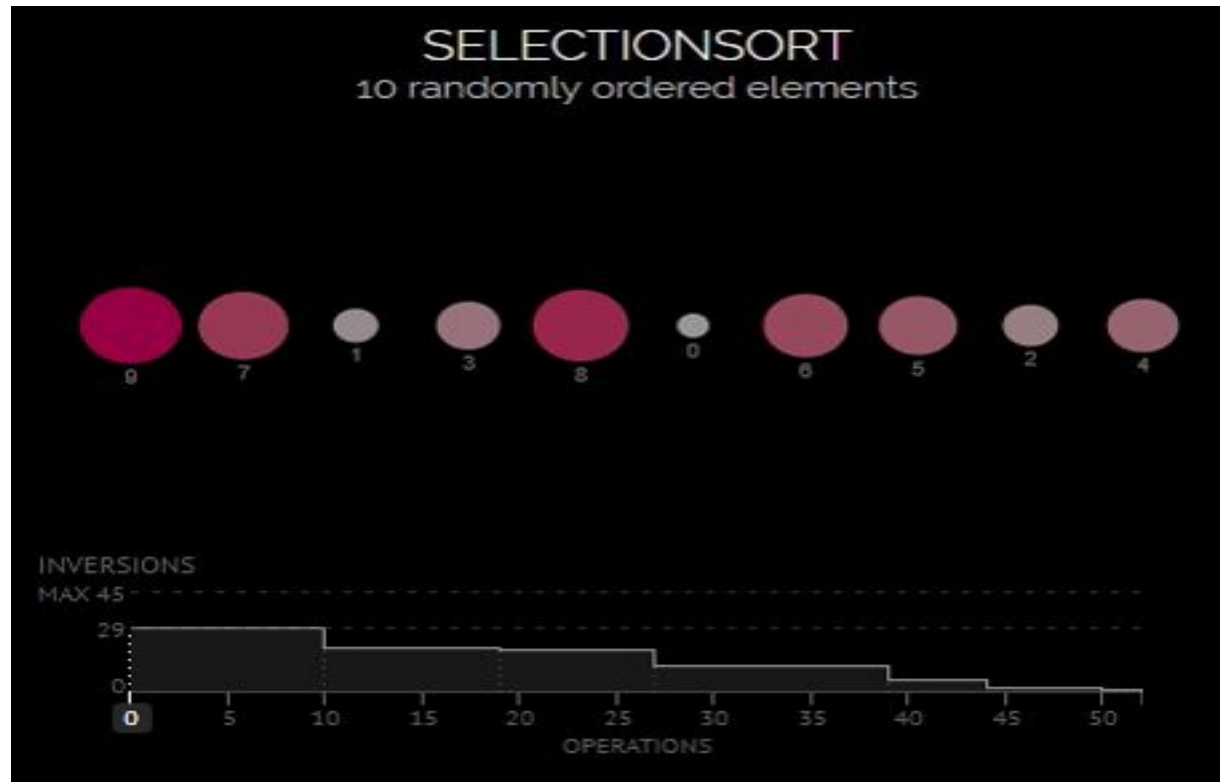


5	2	1	3	9	0	4	6	8	7
---	---	---	---	---	---	---	---	---	---

Сортировка

выбором

1. В неотсортированном подмассиве ищется локальный максимум (минимум).
2. Найденный максимум (минимум) меняется местами с последним (первым) элементом в подмассиве.
3. Если в массиве остались неотсортированные подмассивы — смотри пункт 1.



12	6	4	7	9	15	14	8	11	1	10	2	13	5	3
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

01:35

Сортировка вставками

На каждом шаге алгоритма мы берем один из элементов массива, находим позицию для вставки и вставляем

Пример — игра в карточное «Дурака». Когда мы тянем карту из колоды, смотрим на наши разложенные по возрастанию карты и в зависимости от достоинства вытянутой карты помещаем карту в соответствующее место.

6 5 3 1 8 7 2 4

Сортировка подсчетом

Считаем, сколько раз встречается то или иное число в массиве. Зная эти количества, быстро формируем уже упорядоченный массив.

5	6	1	8	8	7	1	0	4	8	3	2	1	3	4	6	1	1
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

Тестовая программа

```
#include <iostream>
#include <iomanip>
#include <fstream>
using namespace std;
int main() {
    fstream file;
    int x[100];
    int count;
    file.open("rand_100.txt",ios::in);
    if(!file){
        cout<<"Файл не открыт"<<endl;
        return -1;
    }
    for(int i=0; i<100; i++)file>>x[i];
    for(int i=0; i<100; i++){
        if(i%20==0 && i>0)cout<<endl;
        cout<<setw(3)<<x[i];
    }
    cout<<endl<<endl;
    count=sort(x, 100);
    for(int i=0; i<100; i++){
        if(i%20==0 && i>0)cout<<endl;
        cout<<setw(3)<<x[i];
    }
    cout<<endl<<endl<<"Количество итераций " <<count<<endl;
    return 0;
}
```


Пузырьковая сортировка

```
int sort(int* x, int size){
    int count=0;
    int temp;
    for(int i=0; i<size; i++){
        for(int j=0; j<size-1; j++){
            if(x[j]>x[j+1]){
                temp=x[j];
                x[j]=x[j+1];
                x[j+1]=temp;
            }
            count++;
        }
    }
    return count;
}
```

Сортировка выбором

```
int sort(int* x, int size){
    int count=0;
    int temp;
    int k;
    for(int i=0;i<size;i++){
        k=i;
        temp=x[i];
        for(int j=i+1;j<size;j++){
            if(x[j]<temp){
                k=j;
                temp=x[j];
            }
            count++;
        }
        x[k]=x[i];
        x[i]=temp;
    }
    return count;
}
```

Сортировка вставками

```
int sort(int* x, int size){
    int count=0;
    int temp;
    int j;
    for(int i=0;i<size;i++){
        temp=x[i];
        j=i-1;
        while(j>=0 && x[j]>temp){
            x[j+1]=x[j];
            j--;
            count++;
        }
        x[j+1]=temp;
    }
    return count;
}
```

```
int sort(int* x, int size){
    int count=0;
    char c[100];
    int k;
    for(int i=0;i<100;i++){
        c[i]=0;
        count++;
    }
    for(int i=0;i<100;i++){
        c[x[i]]++;
        count++;
    }
    k=0;
    for(int i=0;i<100;i++){
        while(c[i]>0){
            x[k]=i;
            c[i]--;
            k++;
            count++;
        }
    }
    return count;
}
```

Сортировка
подсчетом