МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный университет телекоммуникаций им. проф. М. А. Бонч-Бруевича»
(СПбГУТ)

Факультет Информационных систем и технологий
Кафедра Безопасности информационных систем

Отчет

По предмету «Алгоритмы и структуры данных»
о практической работе №3:
«Сортировка числовых массивов. Некоторые методы сортировки.»

Выполнил студент гр. ИСТ-022:

Волков Кирилл

//_____//
оценка

Проверил: Бородянский Ю.М.

//_____//
Подпись

г. Санкт-Петербург
2021

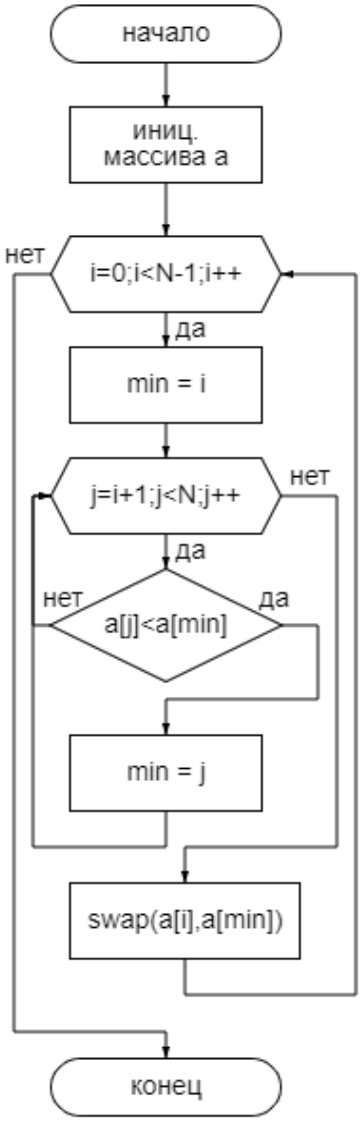
Задача работы

Применить на практике три метода сортировки массива. Проанализировать результаты работы программ по затраченного времени. Сделать выводы об эффективности.

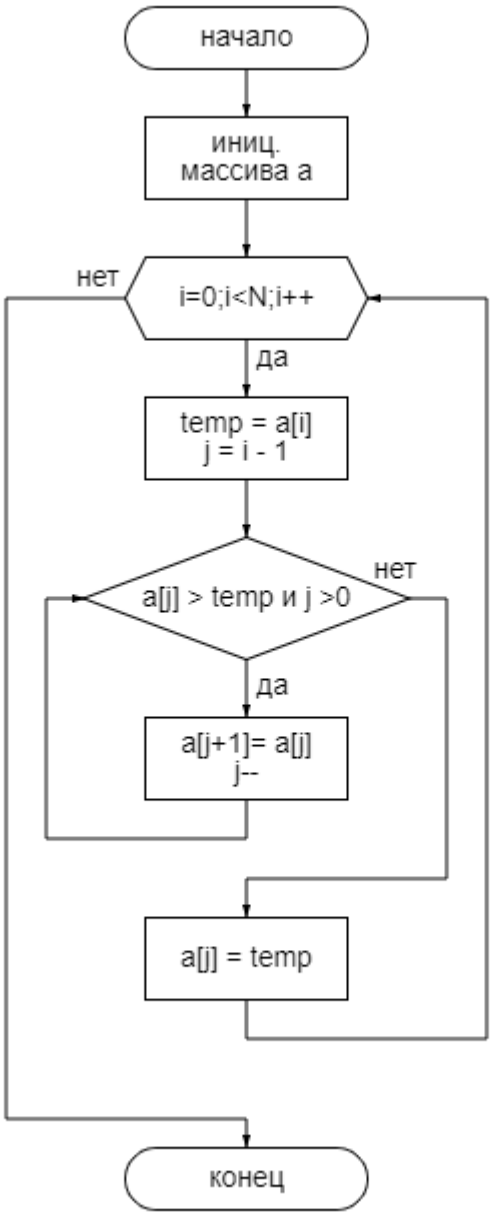
Ход работы

Разработаем программу, которая создаёт файл со случайным массивом. Затем сортирует его по возрастанию и тоже записывает файл, аналогично с массивом по убыванию. Итогом является три файла. Затем массив из каждого файла сортируется каждым методом, засекается время. Результатом является вывод в консоль времени работы методов с разными массивами.

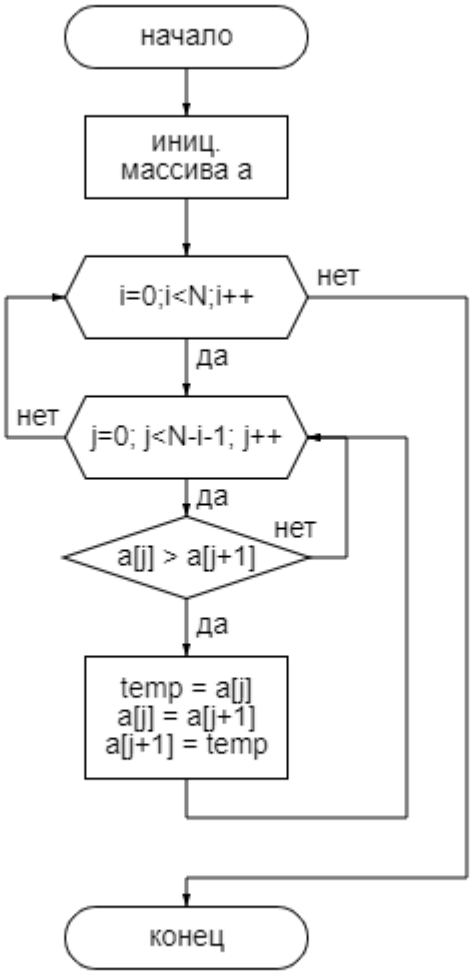
Блок-схема алгоритма выбора



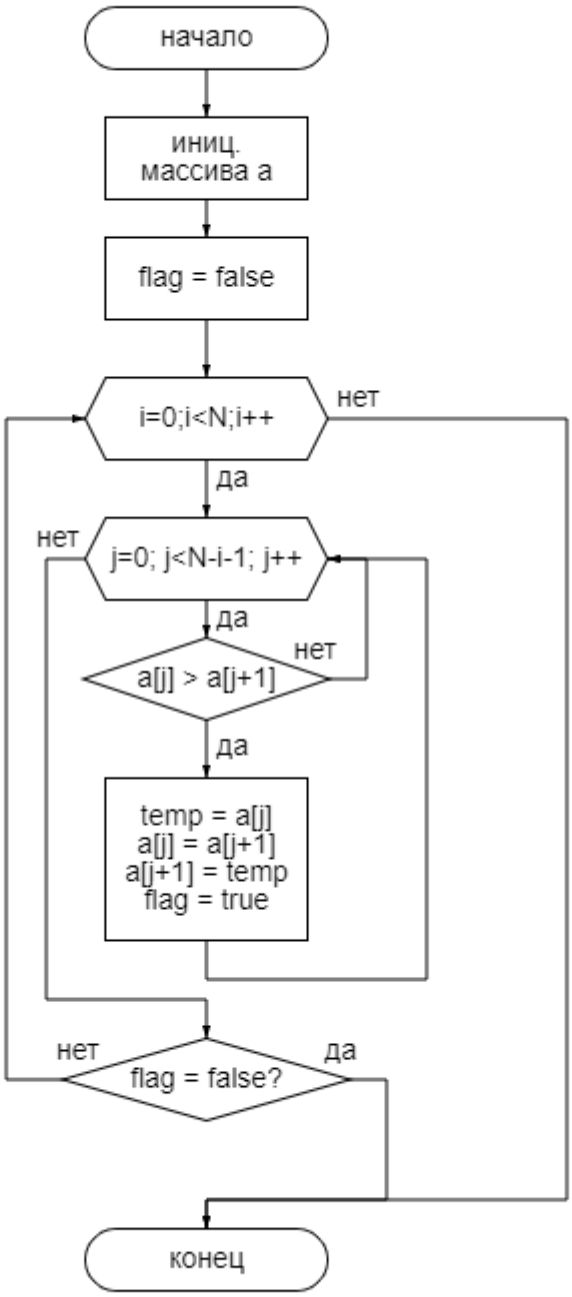
Блок-схема алгоритма вставки



Блок-схема алгоритма пузырька



Блок-схема метода пузырька с оптимизацией



Сложность Big-O

Метод выбора: O(n^2).

Метод вставки: O(n^2).

Метод пузырька: O(n^2).

Метод пузырька с оптимизацией: в лучше случае O(n) и O(n^2) в худшем.

Результат работы программы

```
Количетсво элементов = 10000

Метод выбора
Неотсортированным массив.              Время работы:   175 млс
Отсортированный по возрастанию массив.  Время работы:   174 млс
Отсортированный по убыванию массив.     Время работы:   179 млс

Метод вставки
Неотсортированный массив.              Время работы:   143 млс
Отсортированный по возрастанию массив.  Время работы:   0 млс
Отсортированный по убыванию массив.     Время работы:   210 млс

Метод пузырька
Неотсортированный массив.              Время работы:   389 млс
Отсортированный по возрастанию массив.  Время работы:   198 млс
Отсортированный по убыванию массив.     Время работы:   343 млс

Метод пузырька(Оптимизированный)
Неотсортированный массив.              Время работы:   22 млс
Отсортированный по возрастанию массив.  Время работы:   0 млс
Отсортированный по убыванию массив.     Время работы:   325 млс
Process returned 0 (0x0)   execution time : 3.001 s
Press any key to continue.
```

Вывод

В ходе практической работы были применены знания о одномерных массивах, на практике проверены алгоритмы сортировки. Найдены их сложности в нотации Big-O. Анализ времени работы позволяет сделать следующие выводы: самым эффективным является оптимизированный метод пузырька, самым худшим же обычный метод пузырька. Средним по эффективности оказались метод вставки и выбора, причём алгоритм вставки немного выигрывает в эффективности.

Код программы смотрите в приложении 1.

Приложение 1

```
#include <iostream>
#include <time.h>
#include <Windows.h>
#include <fstream>

using namespace std;

void creat_mass (){
    ofstream mass_rand_f("mass_rand.txt");
    ofstream mass_sort_f1("mass_sort_asc.txt");
    ofstream mass_sort_f2("mass_sort_desc.txt");
    int length = 10000;
    int mass_rand[length], temp, id;
    for (int i = 0; i < length; i++)
    {
        mass_rand[i] = 1 + rand() % 100;
        mass_rand_f<<mass_rand[i]<<endl;
    }

    for (int i = 1; i < length; i++)
    {
        temp = mass_rand[i]; // текущее значение элемента массива
        id = i - 1; //  индекс предыдущего элемента массива
        while (id >= 0 && mass_rand[id] > temp)
```

```
        {
            mass_rand[id + 1] = mass_rand[id]; // перестановка элементов массива
            mass_rand[id] = temp;
            id--;
        }
    }
    for (int i = 0; i < length; i++)
    mass_sort_f1<<mass_rand[i]<<endl;
    for (int i = 1; i < length; i++)
    {
        temp = mass_rand[i]; // текущее значение элемента массива
        id = i - 1; //  индекс предыдущего элемента массива
        while (id >= 0 && mass_rand[id] < temp)
        {
            mass_rand[id + 1] = mass_rand[id]; // перестановка элементов массива
            mass_rand[id] = temp;
            id--;
        }
    }
    for (int i = 0; i < length; i++)
    mass_sort_f2<<mass_rand[i]<<endl;
}

void selection_sort (){
    ifstream mass_rand_f("mass_rand.txt");
```

```
ifstream mass_sort_f_asc("mass_sort_asc.txt");
ifstream mass_sort_f_desc("mass_sort_desc.txt");


int length = 10000;
int mass[length];
for (int i = 0; i < length; i++)
    mass_rand_f>>mass[i];



int minim, cnt=0;
unsigned int  start_time =  clock();
for (int i=0; i<length-1; i++){
    minim = i;
    for (int j=i+1;j<length;j++){
        if (mass[j]<mass[minim])
            minim = j;
    }
    swap(mass[i],mass[minim]);
    cnt++;
}
unsigned int end_time  = clock();
cout<<"\nКоличетсво элементов = 10000";
cout<<"\n\nМетод выбора\nНеотсортированным массив.\t\tВремя работы:\
t"<<end_time - start_time<<" млс";
```

```
    for (int i = 0; i < length; i++)
        mass_sort_f_asc>>mass[i];

    start_time =  clock();
    for (int i=0; i<length-1; i++){
        minim = i;
        for (int j=i+1;j<length;j++){
            if (mass[j]<mass[minim])
                minim = j;
        }
        swap(mass[i],mass[minim]);
    }
    end_time = clock();
    cout<<"\nОтсортированный по возрастанию массив.\tВремя работы:\
t"<<(end_time-start_time)<<" млс";

    for (int i = 0; i < length; i++)
        mass_sort_f_desc>>mass[i];

    start_time =  clock();
    for (int i=0; i<length-1; i++){
        minim = i;
        for (int j=i+1;j<length;j++){
            if (mass[j]<mass[minim])
                minim = j;
```

```
        }
        swap(mass[i],mass[minim]);
    }
    end_time = clock();
    cout<<"\nОтсортированный по убыванию массив.\tВремя работы:\
t"<<(end_time-start_time)<<" млс";



}

void vstavka_sort (){
    ifstream mass_rand_f("mass_rand.txt");
    ifstream mass_sort_f_asc("mass_sort_asc.txt");
    ifstream mass_sort_f_desc("mass_sort_desc.txt");

    int length = 10000;

    int mass[length], temp, j;

    for (int i = 0; i < length; i ++)
        mass_rand_f>>mass[i];

    unsigned int start_time = clock();

    for (int i = 1; i < length; i++)
```

```
    {
        temp = mass[i]; // текущее значение элемента массива
        j = i - 1; //  индекс предыдущего элемента массива
        while (j >= 0 && mass[j] > temp)
        {
            mass[j+ 1] = mass[j]; // перестановка элементов массива
            j--;
        }
        mass[j] = temp;
    }

    unsigned int end_time = clock();

    cout<<"\n\nМетод вставки\nНеотсортированный массив.\t\tВремя работы:\
t"<<end_time-start_time<<" млс";

    for (int i = 0; i < length; i ++)
    mass_sort_f_asc>>mass[i];

    start_time = clock();

    for (int i = 1; i < length; i++)
    {
        temp = mass[i]; // текущее значение элемента массива
        j = i - 1; //  индекс предыдущего элемента массива
```

```
    while (j >= 0 && mass[j] > temp)
    {
        mass[j + 1] = mass[j]; // перестановка элементов массива
        mass[j] = temp;
        j--;
    }
}

end_time = clock();

cout<<"\nОтсортированный по возрастанию массив.\tВремя работы:\
t"<<end_time-start_time<<" млс";

for (int i = 0; i < length; i ++)
mass_sort_f_desc>>mass[i];

start_time = clock();

for (int i = 1; i < length; i++)
{
    temp = mass[i]; // текущее значение элемента массива
    j = i - 1; //  индекс предыдущего элемента массива
    while (j >= 0 && mass[j] > temp)
    {
```

```
            mass[j + 1] = mass[j]; // перестановка элементов массива
            mass[j] = temp;
            j--;
        }
    }

    end_time = clock();

    cout<<"\nОтсортированный по убыванию массив.\tВремя работы:\
t"<<end_time-start_time<<" млс";
}

void bubble_sort (){
    ifstream mass_rand_f("mass_rand.txt");
    ifstream mass_sort_f_asc("mass_sort_asc.txt");
    ifstream mass_sort_f_desc("mass_sort_desc.txt");

    int length = 10000;

    int mass[length], temp;

    for (int i = 0; i < length; i ++)
        mass_rand_f>>mass[i];

    unsigned int start_time = clock();
```

```
for (int i = 0; i < length - 1; i++) {
    for (int j = 0; j < length - i - 1; j++) {
        if (mass[j] > mass[j + 1]) {
            // меняем элементы местами
            temp = mass[j];
            mass[j] = mass[j + 1];
            mass[j + 1] = temp;
        }
    }
}

unsigned int end_time = clock();

cout<<"\n\nМетод пузырька\nНеотсортированный массив.\t\tВремя работы:\t"<<end_time-start_time<<" млс";

for (int i = 0; i < length; i ++)
    mass_sort_f_asc>>mass[i];

start_time = clock();

for (int i = 0; i < length - 1; i++) {
    for (int j = 0; j < length - i - 1; j++) {
```

```
            if (mass[j] > mass[j + 1]) {
                // меняем элементы местами
                temp = mass[j];
                mass[j] = mass[j + 1];
                mass[j + 1] = temp;
            }
        }
    }

    end_time = clock();

    cout<<"\nОтсортированный по возрастанию массив.\tВремя работы:\
t"<<end_time-start_time<<" млс";

    for (int i = 0; i < length; i ++)
        mass_sort_f_desc>>mass[i];

    start_time = clock();

    for (int i = 0; i < length - 1; i++) {
        for (int j = 0; j < length - i - 1; j++) {
            if (mass[j] > mass[j + 1]) {
                // меняем элементы местами
                temp = mass[j];
                mass[j] = mass[j + 1];
```

```
            mass[j + 1] = temp;
        }
    }
}

end_time = clock();

cout<<"\nОтсортированный по убыванию массив.\tВремя работы:\
t"<<end_time-start_time<<" млс";

}

void bubble_sort_opt (){
    ifstream mass_rand_f("mass_rand.txt");
    ifstream mass_sort_f_asc("mass_sort_asc.txt");
    ifstream mass_sort_f_desc("mass_sort_desc.txt");

    int length = 10000;

    int mass[length], temp;

    bool flag;

    for (int i = 0; i < length; i ++)
        mass_rand_f>>mass[i];
```

```
    unsigned int start_time = clock();

    for (int i = 0; i < length - 1; i++) {
        for (int j = 0; j < length - i - 1; j++) {
            flag = false;
            if (mass[j] > mass[j + 1]) {
                // меняем элементы местами
                temp = mass[j];
                mass[j] = mass[j + 1];
                mass[j + 1] = temp;
                flag = true;
            }
        }
        if (!flag)
            break;
    }

    unsigned int end_time = clock();

    cout<<"\n\nМетод пузырька(Оптимизированный)\nНеотсортированный массив.\t\tВремя работы:\t"<<end_time-start_time<<" млс";

    for (int i = 0; i < length; i ++)
```

```
        mass_sort_f_asc>>mass[i];


    start_time = clock();


    for (int i = 0; i < length - 1; i++) {
        flag = false;
        for (int j = 0; j < length - i - 1; j++) {
            if (mass[j] > mass[j + 1]) {
                // меняем элементы местами
                temp = mass[j];
                mass[j] = mass[j + 1];
                mass[j + 1] = temp;
                flag = true;
            }
        }
        if (!flag)
        break;
    }


    end_time = clock();


    cout<<"\nОтсортированный по возрастанию массив.\tВремя работы:\
t"<<end_time-start_time<<" млс";


    for (int i = 0; i < length; i ++)
```

```
        mass_sort_f_desc>>mass[i];


    start_time = clock();


    for (int i = 0; i < length - 1; i++) {
        flag = false;
        for (int j = 0; j < length - i - 1; j++) {
            if (mass[j] > mass[j + 1]) {
                // меняем элементы местами
                temp = mass[j];
                mass[j] = mass[j + 1];
                mass[j + 1] = temp;
                flag = true;
            }
        }
        if (!flag)
        break;
    }


    end_time = clock();


    cout<<"\nОтсортированный по убыванию массив.\tВремя работы:\
t"<<end_time-start_time<<" млс";


}
```

```
int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    creat_mass();

    selection_sort();

    vstavka_sort ();

    bubble_sort();

    bubble_sort_opt ();

}
```