**Московский государственный технический университет им. Н.Э. Баумана**

Факультет «Информатика и управление»

Кафедра «ИУ5»

Курс «Основы информатики»

# Отчет по лабораторной работе №5
# «Сортировка одномерного числового массива»

Выполнил:
студент группы ИУ5-13Б
Слоква А.
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Папшев И. С.
Подпись и дата:

Москва, 2020 г.

## Оглавление

## Постановка задачи

### *Само задание*

ЛР состоит из двух частей.

В первой части выполняется разработка и тестирование функций, реализующих алгоритмы сортировки массивов методом выбора максимального (минимального) элемента и методом пузырькового всплытия.

Во второй части создаются шаблоны разработанных в п.1 функций сортировки и с использованием шаблонов выполняется сортировка массива структур и сравнение быстродействия алгоритмов в зависимости от размера и упорядоченности элементов массива.

В Visual Studio решение, реализующее ЛР, должно состоять из двух проектов:

первый проект реализует часть1 задания, а второй – часть 2.

Часть 1.

1.1 Разработать функции для сортировки целочисленного числового массива методом выбора максимального (минимального) элемента и методом пузырькового всплытия. В методе пузырькового всплытия цикл сравнений начинать с конца сортируемого массива (см. п.5 раздела «Указания по выполнению задания»).Для сравнения быстродействия алгоритмов дополнительно включить в функции операторы для подсчета количества выполненных при сортировке сравнений и перестановок элементов массива.

1.2 Выполнить сортировку тестового массива по возрастанию и по убыванию значений элементов массива каждой из двух функций и распечатать отсортированный массив и количество сделанных при сортировке сравнений и перестановок элементов массива. *Распечатку результатов сортировки выполнять в функции main(), а данные для печати должны возвращаться из функций, выполняющих сортировку* (функция сортировки не должна печатать данные).

Часть 2.

2.1 Создать шаблоны для функций сортировки, разработанных в части 1 ЛР:

- один шаблон для сортировки методом пузырькового всплытия, начиная с конца массива;
- один шаблон для сортировки методом выбора максимального (минимального) элемента.

Шаблоны поместить в заголовочный файл и подключить его к проекту.

2.2 Создать структуры Date и Student и массив group из 10 элементов типа Student.

```
struct Date{
        int day;
        int month;
        int year;
};
struct Student{
   string name;
   Date birthDay;
   char id[6];//номер зачётной книжки
};
```

2.3 Используя шаблоны, выполнить сортировку массива group по трем признакам:

- по возрастанию значения поля name,
- по убыванию значения поля birthDay,
- по возрастанию номера зачетной книжки.

2.4 Сравнить быстродействия алгоритмов сортировки в зависимости от размера и упорядоченности элементов массива. Для сравнения быстродействия алгоритмов используйте целочисленные массивы.

Результаты должны содержать числа сравнений и перестановок, выполненных каждой из функций в процессе сортировки массивов размером 100 и10000 чисел для различных состояний упорядоченности элементов массивов (см. п.7 раздела «Указания по выполнению задания»). Возможность изменения длины массива реализуйте с помощью динамического массива, а для его инициализации используйте датчик случайных чисел (см. п.3.3 раздела “Примеры работы с массивами”). Образец таблицы результатов сравнения алгоритмов сортировки приведен в п.8 раздела «Указания по выполнению задания». Элементы массивов не распечатывать.

Объясните результаты сравнения.

### *Указания по выполнению задания*

1. Разработку программы начинайте с создания функции *int Menu().* Использование меню является удобным способом разделения программы на части. Это ускорит разработку и отладку программы и проверку преподавателем правильности выполнения пунктов задания ЛР.
2. Для отладки программы ***обязательно*** разработайте контрольный пример - ***статический*** массив из 10 элементов (см. п. 7.3.1 раздела “Примеры работы с массивами”). Значения элементов тестового массива и их количество должны быть минимально-достаточными, чтобы обеспечивать удобство контроля и полноту тестирования. Нужно предусмотреть возможность простого изменения размера и значений тестового массива в программе.
3. В процессе отладки используйте пошаговый режим выполнения программы и пошаговое сравнение значений в окне отладчика с результатами контрольного примера. Для ускорения отладки используйте «останов с условием».
4. В методе пузырькового всплытия цикл сравнений (внутренний цикл) можно начинать как с конца, так и с начала массива. Если массив не отсортирован, то эти варианты реализации метода равноценны. При работе с данными обычно приходится добавлять или удалять данные в уже отсортированный массив. Удаление данных выполняется путем сдвига «хвоста» массива влево, а новые элементы добавляются в ***конец*** массива и затем выполняется его сортировка. В этом случае первый вариант оказывается более эффективным.
5. Для того, чтобы один и тот же шаблон можно было бы использовать для сортировки массивов из различных элементов по различным условиям, для сравнения элементов сортируемых массивов в шаблоне нужно разработать разные функции сравнения. Эти функции должны иметь одинаковую сигнатуру, чтобы в шаблоне их можно было бы вызывать через указатель на эти функции
   bool (*cmp)(T obj1,T obj2), где
   T – тип элементов в сортируемом массиве, который является параметром шаблона.
   Этот указатель нужно добавить в список параметров в шаблон функции сортировки.

Примеры функций сравнения:

bool cmp1(int a, int b){ return a > b;} //для сортировки целочисленного массива по возрастанию;

bool cmp2(int a, int b){ return a < b;} ;} //для сортировки целочисленного массива по убыванию;

bool cmp3(char* a, char* b){ return strcmp(a , b)>0;} //для сортировки массива из символьных строк по возрастанию;

6. Состояния массива:

I – исходный со случайно расположенными числами,
II – исходный массив, предварительно отсортированный по возрастанию,
III - исходный массив, предварительно отсортированный по убыванию.

Предварительно отсортированный массив представляет собой отсортированный случайный массив, в котором последний элемент заменен вручную на элемент, меньший его нулевого элемента (для массива, отсортированного по возрастанию) или на элемент, больший его нулевого элемента (для массива, отсортированного по убыванию). Таким образом имитируется добавление новых элементов в конец существующего отсортированного массива.

Образец таблицы сравнения результатов быстродействия алгоритмов сортировки.

| Алгоритм | Размер массива | | | | | |
|---|---|---|---|---|---|---|
| | 100 | | | | | |
| | Состояние массива | | | | | |
| | I | | II | | III | |
| | compare | swap | compare | swap | compare | swap |
| bubbleEnd | | | | | | |
| minMax | | | | | | |

## Разработка алгоритма

### *Краткое описание алгоритма*

*1 часть*

Сначала происходит ввод пользователем массива для сортировки или выбор стандартного массива для отладки. Далее у пользователя запрашивается выбор метода сортировки, после чего на соответствующую методу функцию подаётся сам массив, его длина, функция, отвечающая за направление сортировки, и ссылки на переменные в которые функция метода будет записывать количества сравнений и перестановок. После работы функции программа выводит пользователю отсортированный массив, а также данные о количестве сравнений и перестановок. Далее пользователь может повторить алгоритм снова или выйти из программы.

*2 часть*

В этой программе пользователь выбирает между выполнением пункта заданий с 2.1 — 2.3 и выполнением задания 2.4. Если пользователь выбрал первое, программа генерирует массив студентов, выводит его и просит пользователя выбрать метод сортировки и параметр, по которому будет вестись сортировка. После выбора пользователя на соответствующую методу функцию подаётся массив со студентами, их количество (10), функция, которая сравнивает выбранный пользователем  параметр, ссылки на переменные в которые она будет записывать количество сравнений и перестановок. Далее пользователю выводится отсортированный массив, а также количество сравнений и перестановок. В случае выбора выполнения задания 2.4, у пользователя запрашивается длина массива, который будет использоваться для сравнения, после чего программа псевдослучайно генерирует 2 одинаковых экземпляра массива нужной длины. Далее, для каждого метода этот массив будет приводиться поочередно во все 3 состоянии. Результаты сортировки из этих состояний будут выведены пользователю (количество сравнений и перестановок).

*Метод выбора максимального (минимального)*

Этот метод поочередно проходит в массив в поисках максимального (минимального) элемента. После его нахождения, он его меняет местами с начальным элементом массива и делает тоже самое уже со следующего элемента. После того, как он дошел таким образом до конца, он возвращает отсортированный массив.

*Метод пузырька*

Этот метод поочередно проходит по массиву и сравнивает 2 соседних значения. В случае, если они расположены в неправильном порядке, он меняет их местами. После того, как он дошел до конца массива, он начинает проверять снова до (в случае если метод работает с начала массива, то «от») места, где последний раз был обмен значений.

***Описание переменных***

*1 часть*

*Входные данные:*

1. *int* source[] - исходный массив, который будет сортироваться. Вводится либо пользователем, либо в него копируется массив для отладки.
2. *int* len — длина массива *source*. Указывается пользователем, либо берется длина массива для отладки.
3. *int* method — пользовательский выбор метода сортировки.

*Выходные данные:*

1. *int* array[] - отсортированный массив
2. *int* compares — количество сравнений
3. *int* swap — количество перестановок

*Константы:*

1. *const int* forDebug[] - массив для отладки
2. *const int* lenForDebug — длина массива *forDebug*
3. *const string* menu...[] - массивы с элементами меню

*Функция minmax:*

1. *int* value — наибольшее (наименьшее) значение в массиве
2. *int* index — индекс *value*
3. *int* i — счетчик пройденных элементов массива
4. *int n* — счетчик, используется для нахождении максимального (минимального) элемента в оставшейся части массива

*Функция bubble:*

1. *int* oldlastSwap — индекс последнего обмена в предыдущей итерации
2. *int* lastSwap — индекс последнего обмена в текущей итерации. В дальнейшем станет *oldlastSwap*
3. *int* i — счетчик для прохода по массиву.

*Остальные:*

1. *int* answer — ответ пользователя, используется для проверка его корректности
2. *int* a, b — переменный для сравнения

3. *int* first, second — индексы в функции *swap* элементов, которые нужно поменять местами
4. *int, int** ret — значение для возврата из функции.

*Часть 2*

Элементы *ret, len, first, second, array, compares, swaps, value, index, oldlastSwap, lastSwap, a, b, menu…* - имеют то же значение, что и в части 1. Повторяющиеся элементы в дальнейшем я указывать не буду.

*Входные данные:*

1. *int* choose — выбор пользователя в главном меню.

*Константы:*

1. *const string* name[] - массив с именами студентов.

***Схема алгоритма***

*Часть 1*

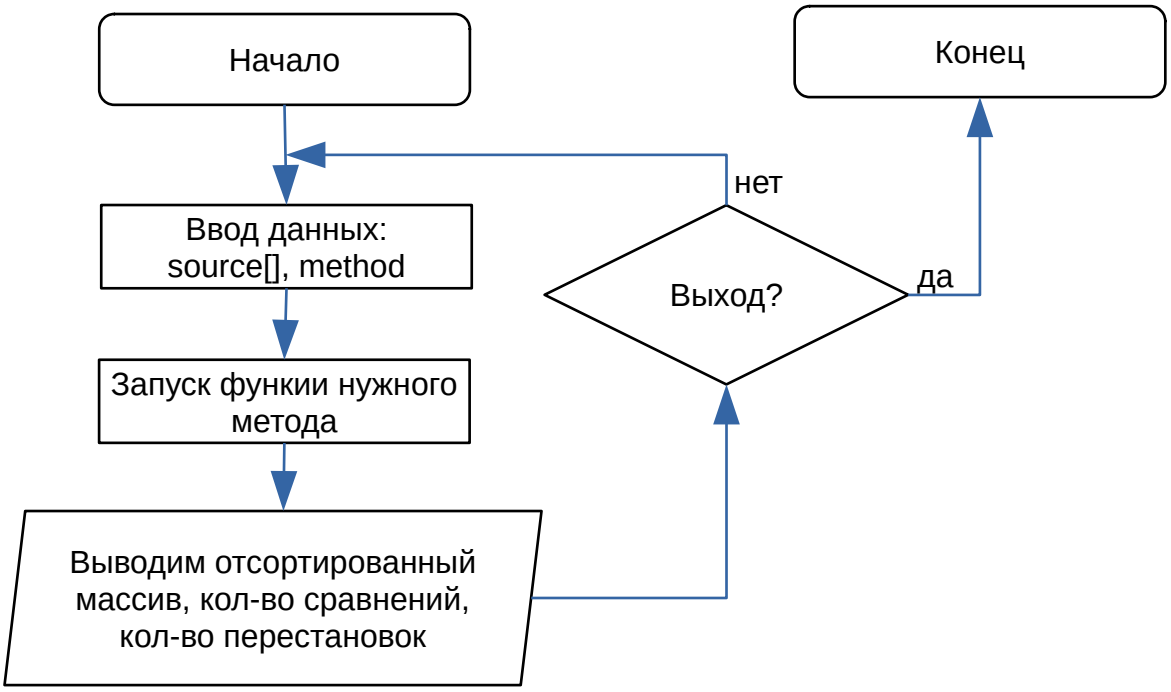
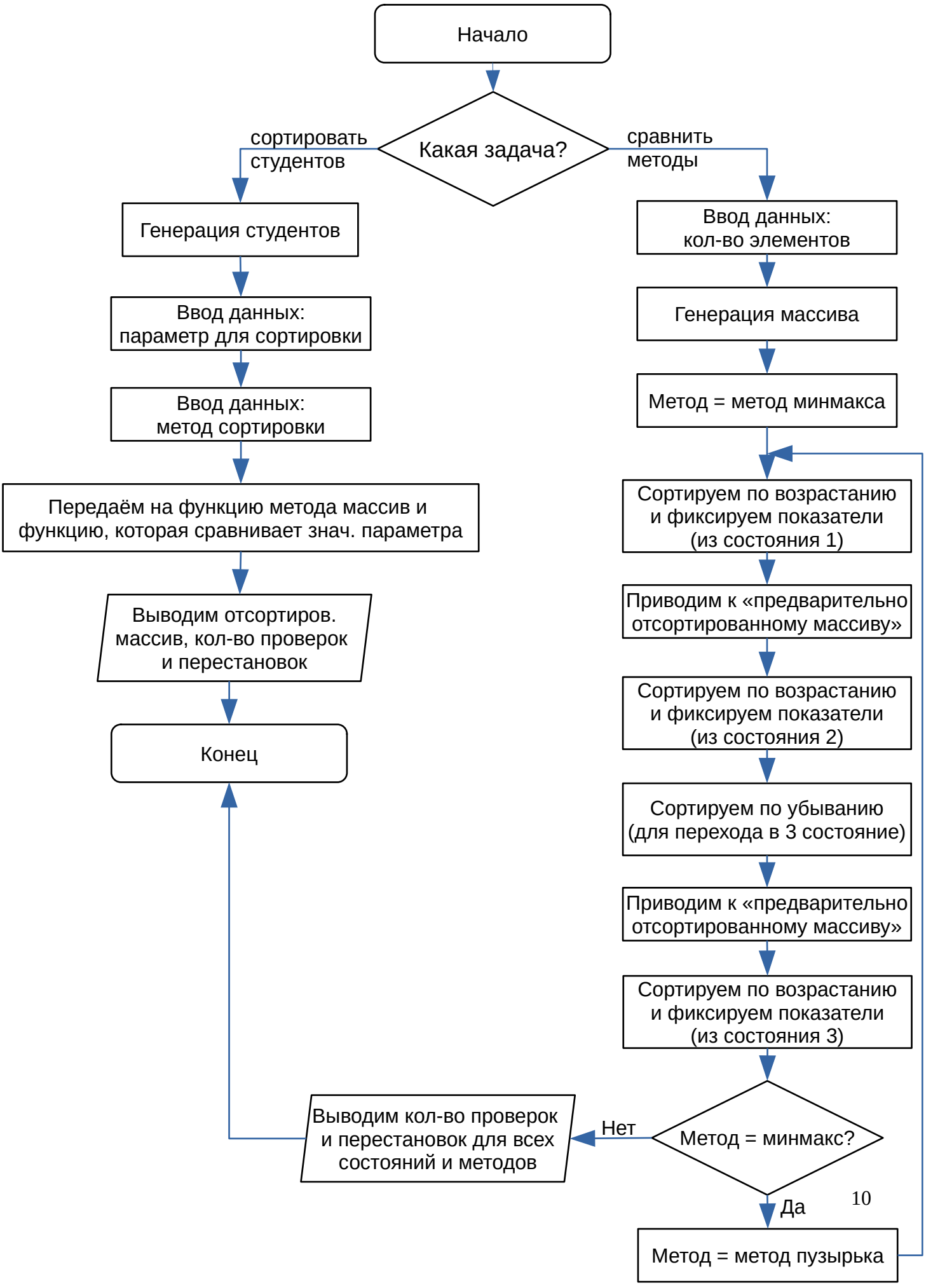
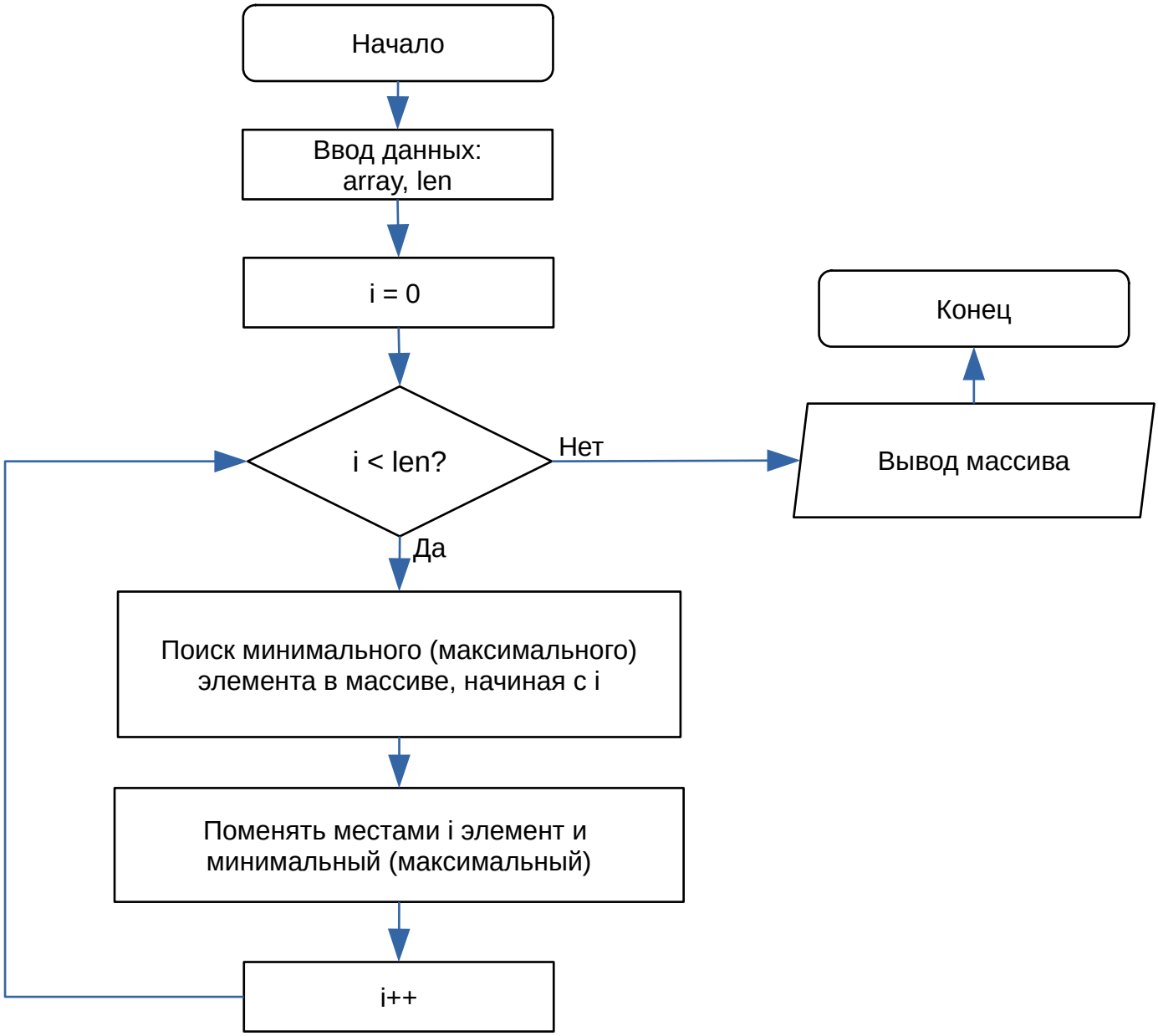



*Иллюстрация 1: Часть 1*

*Часть 2*


Начало
Какая задача?
сортировать студентов
сравнить методы
Генерация студентов
Ввод данных: параметр для сортировки
Ввод данных: метод сортировки
Передаём на функцию метода массив и функцию, которая сравнивает знач. параметра
Выводим отсортиров. массив, кол-во проверок и перестановок
Конец
Ввод данных: кол-во элементов
Генерация массива
Метод = метод минмакса
Сортируем по возрастанию и фиксируем показатели (из состояния 1)
Приводим к «предварительно отсортированному массиву»
Сортируем по возрастанию и фиксируем показатели (из состояния 2)
Сортируем по убыванию (для перехода в 3 состояние)
Приводим к «предварительно отсортированному массиву»
Сортируем по возрастанию и фиксируем показатели (из состояния 3)
Метод = минмакс?
Нет
Да
Выводим кол-во проверок и перестановок для всех состояний и методов
Метод = метод пузырька

*Метод выбора минимального (максимального)*




**Текст программы**

***Часть 1***

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

using namespace std;

//Создание массива для отладки
const int forDebug[] = {37, 20, 58, 49, 41, 51, 99, 89, 22, 36};
const int lenForDebug = sizeof(forDebug) / sizeof(forDebug[0]);

//Функция для ввода и возможной проверки данных
int input(string massage, bool (*check)(int, int) = [](int b, int a) {return
true;}, int a = -1) {
    int ret;

    cout << massage << ": ";
```

```
    if (!(cin >> ret) || !check(ret, a)) {
        cout << "Введены некорректные данные" << endl;
        if (cin.fail()) {
            cin.clear();
            while (cin.get() != '\n');
        }
        return(input(massage, check, a));
    }
    return ret;
}

//Функция для проверки данных при работе menu с input
bool checkLen(int answer, int len) { return(answer <= len && answer > 0); }
//Функция для создания меню и получение корректного ответа
int menu(const string options[], int len, string header, string invite, int
start = 0) {
    cout << string(40, '=') << endl;
    cout << header << endl;

    for (int i = start; i < len; i++) {
        cout << i + 1 - start << ") " << options[i] << endl;
    }
    return input(invite, checkLen, len) + start;
}

//Проверка данных при вводе кол-во элементов
bool checkOneMore(int answer, int len) { return(answer > 1); }
//Функция для ввода массива
int* inputArr(int &n) {
    n = input("Введите кол-во элементов в массиве", checkOneMore);
    int *ret = new int[n];
    for (int i = 0; i < n; i++) {
        ret[i] = input(to_string(i + 1) + " элемент");
    }
    return(ret);
}

//Функция для копирования массивов
template <typename T>
int* copyArr(T source[], int len) {
    int *ret = new int[len];
    for (int i = 0; i < len; i++) {
        ret[i] = source[i];
    }
    return ret;
}

//Функция для вывода в консоль массива в JS-подобном виде
void outArr(int arr[], int len) {
    cout << "[" << arr[0];
    for (int i = 1; i < len; i++) {
        cout << ", " << arr[i];
    }
    cout << "]" << endl;
}

//Функции для сравнения чисел
//Будут использоваться как аргемент для функ. сорт.
bool min(int a, int b) { return a < b; }
bool max(int a, int b) { return a > b; }
```

```
//Функция для обмена местами элементов в массиве
void swap(int array[], int first, int second) {
    int a = array[second];
    array[second] = array[first];
    array[first] = a;
}

//Функция сортировки выбором минимального (максимального)
void minmax(int array[], int len, bool (*fMinMax)(int, int), int &compares, int
&swaps) {
    int value; //Значение минимального (максимального)
    int index; //Его индекс
    for (int i = 0; i < len; i++) { //Идем по массиву
        value = array[i]; //Устанавливаем начальные значения
        index = i;
        for (int n = i; n < len; n++) { //Ищем минимальное (максимальное) с i-
ого элем.
            compares++;
            if (fMinMax(array[n], value)) {
               value = array[n];
               index = n;
            }
        }
        if (index != i) { //Когда находим - меняем местами
            swaps++;
            swap(array, index, i);
        }
    }
}

//Функция сортировки пузырьком
void bubble(int array[], int len, bool (*fMinMax)(int, int), int &compares, int
&swaps) {
    int oldlastSwap = 0, lastSwap; //Индекс последнего обмена
    while (oldlastSwap != -1) { //Ищем, пока происходят обмены
        lastSwap = -1;
        for (int i = len - 1; i > oldlastSwap; i--) {
            //Идем с начала в конец до послед. обмена
            compares++;
            if (fMinMax(array[i], array[i-1])) { //Ищем неправильный порядок
               //Когда нашли - меняем местами
               swap(array, i, i-1);
               swaps++;
               //И запоминаем где был последний обмен
               lastSwap = i;
            }
        }
        oldlastSwap = lastSwap;
    }
}

//Массивы с элеменатами меню
const string menuMethod[] = {"Метод минимума", "Метод максимума", "Метод
пузырька (по возростанию)", "Метод пузырька (по убыванию)"};
const string menuArray[] = {"Использовать старый массив", "Ввести новый массив",
"Использовать массив для отладки", "Выход"};
int main() {
    int *array, *source; //Указатели на массив
    int len; //Его длина

    int method; //Метод сортировки
```

```
    int compares, swaps; //Кол-во сравнений и перестановок

    bool unknowArr = true; //Если есть в source массив пользователя - False
    while (true) {
        switch (menu(menuArray, 4, "Массивы", "Выберите действие", unknowArr)) {
            case 4: //Выход
               return(0);
            case 2: //Ввод нового массива
               source = inputArr(len);
               unknowArr = false;
               break;
            case 3: //Использование массива для дебага
               delete[] source; //Избегаем утечки памяти
               //Так как forDebug - const, его нужно сначала скопировать
               source = copyArr(forDebug, lenForDebug);
               len = lenForDebug;
               cout << "Массив: ";
               outArr(source, len); //Выводим его
               unknowArr = true;
               break;
        }

        array = copyArr(source, len); //Копируем source, чтобы не потерять
исходные данные
        //Запрос метода сортировки
        method = menu(menuMethod, 4, "Метод сортировки", "Выберите метод
сортировки");

        compares = 0;
        swaps = 0;
        switch (method) {
            case 1: //Метод минимума
               minmax(array, len, min, compares, swaps);
```

```
}

//Имена студентов
const string names[] = {"Вася Пупкин", "Иван Иванов", "Марьиванна
Марьиванновна", "Петр Сидоров", "Сидр Петров", "Че Гевара", "Дунька
Раздолбаева", "Штирлиц", "Владимир Ульянов", "[засекреченно]"};

//Генерация массива студентов
void genRandomStudentsArray() {
    for (int i = 0; i < 10; i++) {
        students[i].birthDay = genRandomDate();
        students[i].name = names[i];
        genRandomCharArray(students[i].id);
    }
}

//Вывод информации о студенте в JS-подобном виде
void printStudent(Student stud) {
    cout << "{";
    cout << "name: " << stud.name << ",";
    cout << " birthDay: " << stud.birthDay.day << "."
    << stud.birthDay.month << "." << stud.birthDay.year << ",";
    cout << " id: " << stud.id;
    cout << "}" <<endl;
}

//Генерация случайного массива длины n из чисел
int* genRandomIntArray(int n) {
    int *ret = new int[n];
    for (int i = 0; i < n; i++) {
        ret[i] = rand();
    }
    return ret;
}

template <class T>
void swap(T array[], int first, int second) {
    T a = array[second];
    array[second] = array[first];
    array[first] = a;
}

template <class T>
void minmax(T array[], int len, bool (*fMinMax)(T, T), int &compares, int
&swaps) {
    T value;
    int index;
    for (int i = 0; i < len; i++) {
        value = array[i];
        index = i;
        for (int n = i; n < len; n++) {
            compares++;
            if (fMinMax(array[n], value)) {
               value = array[n];
               index = n;
            }
        }
        if (index != i) {
            swaps++;
            swap(array, index, i);
        }
```

```
    }
}

template <class T>
void bubble(T array[], int len, bool (*fMinMax)(T, T), int &compares, int
&swaps) {
    int oldlastSwap = 0, lastSwap;
    while (oldlastSwap != -1) {
        lastSwap = -1;
        for (int i = len - 1; i > oldlastSwap; i--) {
            compares++;
            if (fMinMax(array[i], array[i-1])) {
               swap(array, i, i-1);
               swaps++;
               lastSwap = i;
            }
        }
        oldlastSwap = lastSwap;
    }
}

bool min(int a, int b) { return a < b; }
bool max(int a, int b) { return a > b; }

template <typename T>
bool chars(T* a, T* b){ return strcmp(a, b) > 0; }

//Функции для сравнения параметров студентов
bool compareNames(Student a, Student b) { return chars(a.name.c_str(),
b.name.c_str()); }
bool compareIds(Student a, Student b) { return chars(a.id, b.id); }
bool compareDates(Student a, Student b) {
    Date aD = a.birthDay;
    Date bD = b.birthDay;
    if (aD.year == bD.year) {
        if (aD.month == bD.month) return(aD.day > bD.day);
        else return(aD.month > bD.month);
    } else return(aD.year > bD.year);
}

//Функция анализа метода сортировки
//На входе: функция сортировки, массив для сортировки, его длина
void analyzeMethod(void (*method)(int[], int, bool (int, int), int&, int&), int
ints[], int length) {
    //Массивы для результатов
    int compares[3] = {0, 0, 0};
    int swaps[3] = {0, 0, 0};
    int trash;

    method(ints, length, min, compares[0], swaps[0]); // Результаты состояния 1
    ints[length-1] = ints[0] - 10; //Приведение к "предварительно
отсортированному массиву"
    method(ints, length, min, compares[1], swaps[1]); // Результаты состояния 2
    method(ints, length, max, trash, trash); // Переход в сост. 3
    ints[length-1] = ints[0] + 10;
    method(ints, length, min, compares[2], swaps[2]); // Результаты состояния 3
    for (int i = 0; i < 3; i++) { //Вывод результатов
        cout << "\tСостояние " << i << ": Сравнений - " << compares[i] << "\
tПерестановок - " << swaps[i] << endl;
    }
}
```

```
int input(string massage, bool (*check)(int, int) = [](int b, int a) {return
true;}, int a = -1) {
    int ret;

    cout << massage << ": ";
    if (!(cin >> ret) || !check(ret, a)) {
        cout << "Введены некорректные данные" << endl;
        if (cin.fail()) {
            cin.clear();
            while (cin.get() != '\n');
        }
        return(input(massage, check, a));
    }
    return ret;
}

bool checkLen(int answer, int len) { return(answer <= len && answer > 0); }
int menu(const string options[], int len, string header, string invite, int
start = 0) {
    cout << string(40, '=') << endl;
    cout << header << endl;

    for (int i = start; i < len; i++) {
        cout << i + 1 - start << ") " << options[i] << endl;
    }
    return input(invite, checkLen, len) + start;
}

bool checkPositive(int answer, int len) { return answer > 0; }

//Массивы с элементами меню
const string mainMenu[] = {"Отсортировать студентов", "Сравнить алгоритмы"};
```

## Анализ результатов

***Скриншоты***

*Часть 1*

```
========================================
Массивы
1) Ввести новый массив
2) Использовать массив для отладки
3) Выход
Выберите действие: 2
Массив: [37, 20, 58, 49, 41, 51, 99, 89, 22, 36]
========================================
Метод сортировки
1) Метод минимума
2) Метод максимума
3) Метод пузырька (по возростанию)
4) Метод пузырька (по убыванию)
Выберите метод сортировки: 1
========================================
Результат: [20, 22, 36, 37, 41, 49, 51, 58, 89, 99]
Сравнений: 55
Перестановок: 8
========================================
Массивы
1) Ввести новый массив
2) Использовать массив для отладки
3) Выход
Выберите действие: 1
Введите кол-во элементов в массиве: 3
1 элемент: 3
2 элемент: 2
3 элемент: 1
========================================
Метод сортировки
1) Метод минимума
2) Метод максимума
3) Метод пузырька (по возростанию)
4) Метод пузырька (по убыванию)
Выберите метод сортировки: 3
========================================
Результат: [1, 2, 3]
Сравнений: 3
Перестановок: 3
========================================
Массивы
1) Использовать старый массив
2) Ввести новый массив
3) Использовать массив для отладки
4) Выход
Выберите действие:
```

*Часть 2*

```
========================================
Главное меню
1) Отсортировать студентов
2) Сравнить алгоритмы
Выберите действие: 1
Массив студентов:
1: {name: Вася Пупкин, birthDay: 20.12.2002, id: 930DEWZ2LY}
2: {name: Иван Иванов, birthDay: 16.4.1997, id: 9B0U9Z5X45}
3: {name: Марьиванна Марьиванновна, birthDay: 17.12.1993, id: 0324N1VF
E3}
4: {name: Петр Сидоров, birthDay: 28.3.1999, id: 9VR78W6I0W}
5: {name: Сидр Петров, birthDay: 13.12.1991, id: 3J22NY075L}
6: {name: Че Гевара, birthDay: 12.9.2004, id: 48K3B48A5Y}
7: {name: Дунька Раздолбаева, birthDay: 16.5.1999, id: YJQJME7192}
8: {name: Штирлиц, birthDay: 20.8.1994, id: 604E4B0EPO}
9: {name: Владимир Ульянов, birthDay: 11.3.1996, id: 0HD55YVF75}
10: {name: [засекреченно], birthDay: 18.9.2000, id: 0G6BM46H84}
========================================
Сортировать по...
1) возрастанию значения поля name
2) убыванию значения поля birthDay
3) возрастанию номера зачетной книжки
Выберите как сортировать: 1
========================================
Метод сортировки
1) Метод выбора максимального (минимального)
2) Метод пузырька
Выберите метод сортировки: 1
Результат:
1: {name: Штирлиц, birthDay: 20.8.1994, id: 604E4B0EPO}
2: {name: Че Гевара, birthDay: 12.9.2004, id: 48K3B48A5Y}
3: {name: Сидр Петров, birthDay: 13.12.1991, id: 3J22NY075L}
4: {name: Петр Сидоров, birthDay: 28.3.1999, id: 9VR78W6I0W}
5: {name: Марьиванна Марьиванновна, birthDay: 17.12.1993, id: 0324N1VF
E3}
6: {name: Иван Иванов, birthDay: 16.4.1997, id: 9B0U9Z5X45}
7: {name: Дунька Раздолбаева, birthDay: 16.5.1999, id: YJQJME7192}
8: {name: Владимир Ульянов, birthDay: 11.3.1996, id: 0HD55YVF75}
9: {name: Вася Пупкин, birthDay: 20.12.2002, id: 930DEWZ2LY}
10: {name: [засекреченно], birthDay: 18.9.2000, id: 0G6BM46H84}
Сравнений: 55
Перестановок: 4
Press any key to continue.
```

```
========================================
Главное меню
1) Отсортировать студентов
2) Сравнить алгоритмы
Выберите действие: 2
========================================
Длина массива
1) 100
2) 10 000
3) Ввести свою
Выберите длину массива: 1
========================================
Метод выбора максимального (минимального):
        Состояние 0: Сравнений - 5050   Перестановок - 97
        Состояние 1: Сравнений - 5050   Перестановок - 99
        Состояние 2: Сравнений - 5050   Перестановок - 49
Метод пузырька:
        Состояние 0: Сравнений - 4895   Перестановок - 2629
        Состояние 1: Сравнений - 197    Перестановок - 99
        Состояние 2: Сравнений - 4950   Перестановок - 4851
Press any key to continue.
```

***Сравнение быстродействия алгоритмов***

| Алгоритм | Размер массива | | | | | |
|---|---|---|---|---|---|---|
| | 100 | | | | | |
| | Состояние массива | | | | | |
| | I | | II | | III | |
| | compare | swap | compare | swap | compare | swap |
| bubbleEnd | 4895 | 2629 | 197 | 99 | 450 | 4851 |
| minMax | 5050 | 97 | 5050 | 99 | 5050 | 49 |

| Алгоритм | Размер массива | | | | | |
|---|---|---|---|---|---|---|
| | 10 000 | | | | | |
| | Состояние массива | | | | | |
| | I | | II | | III | |
| | compare | swap | compare | swap | compare | swap |
| bubbleEnd | 49969106 | 25046659 | 19997 | 9999 | 49995000 | 49985001 |
| minMax | 50005000 | 9991 | 50005000 | 9999 | 50005000 | 4999 |