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

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Основы программирования»

Отчет по лабораторной работе №5 «Сортировка массивов»

Выполнил:
студент группы ИУ5-15Б
Трифонов Дмитрий
Подпись и дата:

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

Москва, 2021 г.

Постановка задачи

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[8];//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.

Разработка алгоритма

Описание алгоритмов методов сортировки:
Каждый метод сортировки реализован в виде отдельной функции, принимающей на вход:

1. Массив элементов std::vector <T>
2. Функцию для сравнения элементов (int (*cmp) (T, T))

То есть функции могут сортировать векторы любых типов, в том числе могут сортировать массивы структур.
Метод cmp возвращает:

1, если первый аргумент **больше** второго
0, если оба аргумента **равны**
-1, если первый аргумент **меньше** второго

Описание алгоритмов сортировки:

- Метод пузырька
  Каждый элемент, начинай с конца, начинаем проверять на то, больше ли он предыдущего элемента, или меньше. Если меньше, то будем перемещать начальный элемент в начало массива. Таким образом все элементы переместим на своё место в отсортированном массиве
- Метод выбора минимального элемента
  Сначала рассматриваем весь массив и ищем его минимальный элемент. Ставим его на первое место. После будем рассматривать все элементы массива, кроме тех, которые мы определили на свои места и среди них будем находить минимальный элемент и ставить на своё место.

Описание входных, выходных и вспомогательных данных:

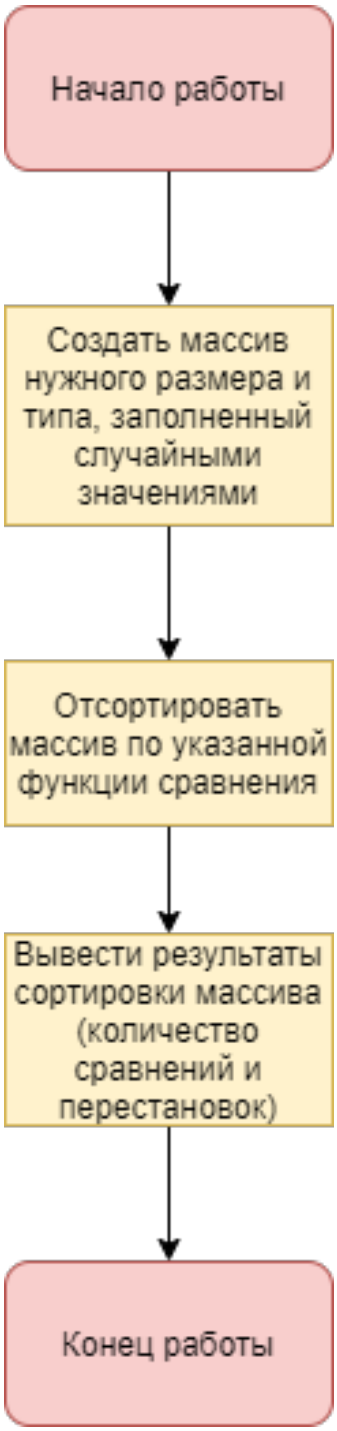
- Входные данные:
  - int arraySize – количество элементов в массиве
  - int arrayType – тип массива(отсортированный/произвольный/отсортированный в обратном порядке)

  Для второй задачи добавляются:
  - int (*cmpType) (Student, Student) – указатель на функцию сравнения
- Промежуточные данные:
  Только для первой задачи:
  - Вектор для сортировки генирируется с помощью функции std::vector <int> getRandomVector(); В нём создаётся вектор vector <int> v, который с помощью стандартной библиотеки random заполняется случайными числами, и, если требуется, сортируется по возрастанию/по убыванию
  - Функция compare для сравнения двух чисел int compare(int a, int b) – описание используемых методов сравнения см. выше
    Только для второй задачи:
  - string names[] - массив имён для создания структур Student с ними
  - Функция int (*cmpType) (Student, Student). Эта функция используется для того, чтобы отсортировывать массив структур в обратном порядке. То есть, изначально инициализируется функция cmpType. А потом, если требуется сортировать вектор по возрастанию, compare делается равным cmpType, а если требуется сортировать по убыванию, то compare = reverseCompare, (int reverseCompare(Student a, Student b) – возвращает -(*cmpType)(a, b)).

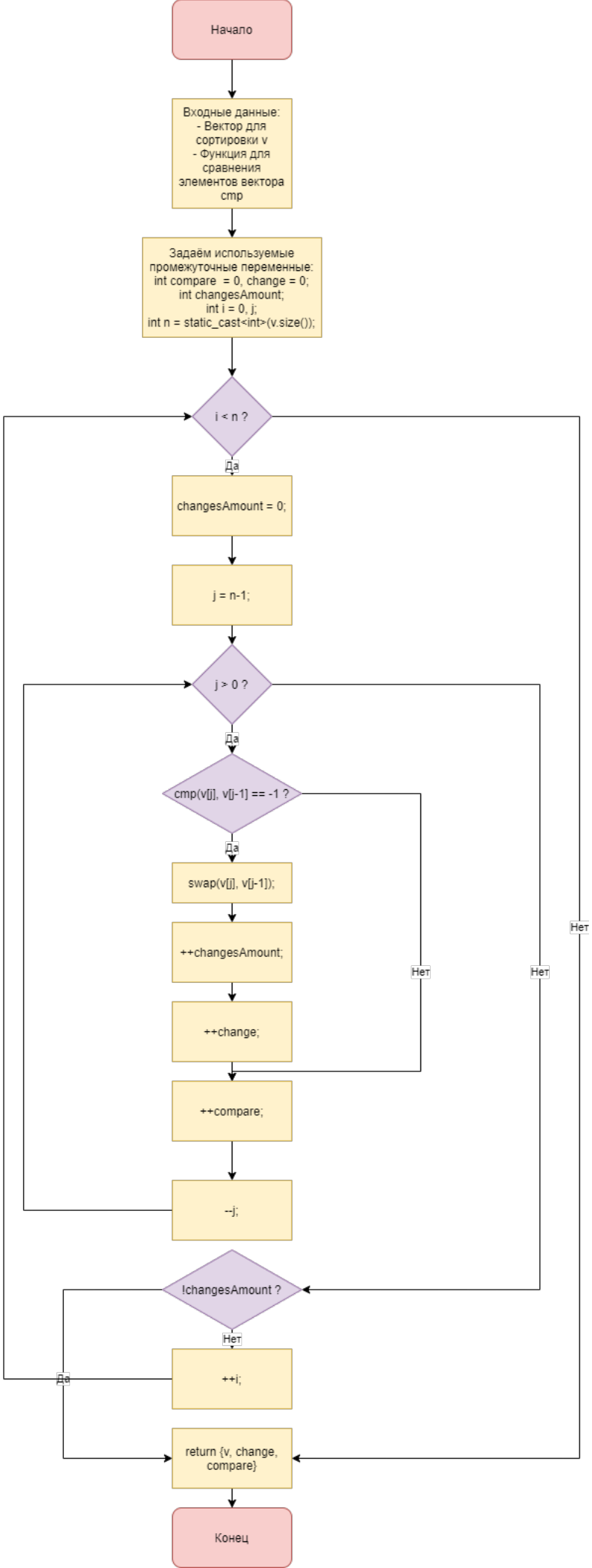
- Случайная структура типа Student генерируется функцией Student getRandomStudent(). Используя те же средства библиотеки random создается и заполняется Student s, который потом и возвращается.
- Массив для сортировки генерируется функцией vector <Student> getRandomVector(). Сначала создается массив vector <Student> v и заполняется случайными значениями. После вектор сортируется по возрастанию/убыванию, если этого требует пользователь.
- Так же, для сравнения структур Student написаны несколько тривиальных функций. Для сравнения по датам перегружены операторы >, <, ==.
  Общие:
- Структура sortResult - Состоит из отсортированного массива vector <T> vec, количества сравнений и количества перестановок (int compare, int swap соответственно).
- Сортирует массивы методом минимального элемента функция template <typename T> sortResult <T> doMinElSort(vector <T> v, int (*cmp) (T, T)). Задаются int compare, int swap для подсчёта количества сравнений и перестановок соответсвенно.
  Для итерации по массиву задаются int l, j. Так же есть переменная int index (исходит из алгоритма сортировки методом минимального элемента).
  Для хранения длины вектора создаётся int n.
- Для сортировки методом пузырька используется аналогичная предыдущей функция `template <typename T> sortResult<T> doBubbleSort(std::vector <T> v, int (*cmp)(T, T))`. В плане промежуточных данных отличие от предыдущей функции лишь в том, что не используется int index, но используется int comparesAmount (проверка на то, является ли массив отсортированным – см. алгоритм сортировки пузырьком).

- Выходные данные:
  - Каждая функция сортировки возвращает структуру sortResult, описанную выше. Выводятся эти данные с помощью функции template <typename T> void printSortResults(sortResult<T> res). Векторы же можно вывести функциями template <typename T> void printArray(vector <T> v) или void printStudents(vector <Students> v).

Общая схема алгортима:



Уточненная схема сортировки методом пузырька



Текст программы
task1.cpp


```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <numeric>
#include <random>
#include <conio.h>
#include "../../lib/labFunc.h"
int arraySize = 10;
short arrayType = 0;
using namespace std;

int compare(int a, int b) {
    if (a > b)
        return 1;
    else if (a == b)
        return 0;
    else
        return -1;
}

vector <int> getRandomVector() {
    std::random_device rd;
    std::mt19937 gen(rd());
    std::uniform_int_distribution<> distrib(-10, 10);
    vector <int> v;
    for (int i = 0; i < arraySize; ++i) {
        v.push_back(distrib(gen));
    }

    switch (arrayType) {
    case 1:
        sort(v.begin(), v.end());
        v.push_back(v[0]);
        v.erase(v.begin());
        break;
    case 2:
        sort(v.rbegin(), v.rend());
        v.push_back(v[0]);
        v.erase(v.begin());
        break;
    }
    return v;
}

void inputArraySize() {
    cout << "Введите размер массива." << endl;
    cin >> arraySize;
    if (arraySize <= 0)
        arraySize = 10;
```

```
50  }
51
52  void inputArrayType() {
53      cout << "Введите тип массива:" << endl;
54      cout << "0 - случайное состояние" << endl;
55      cout << "1 - отсортированный" << endl;
56      cout << "2 - сортировка в обратном порядке" << endl;
57      cin >> arrayType;
58
59      if (!((arrayType >= 0) && (arrayType < 3)))
60          arrayType = 0;
61
62  }
63  int menu() {
64      cout << "========================Menu=========================" << endl;
65      cout << "| 1 - Сортировка методом минимального элемента       |" << endl;
66      cout << "| 2 - Сортировка пузырьковым методом                 |" << endl;
67      cout << "| 3 - Задать количество элементов массива            |" << endl;
68      cout << "| 4 - Задать начальное состояние массива             |" << endl;
69      cout << "| 5 - Выход                                          |" << endl;
70      cout << "=====================================================" << endl;
71      cout << "Выберите номер:" << endl;
72      int choice;
73      cin >> choice;
74      while (cin.fail()) {
75          cout << "Ошибка ввода. Повторите ввод\n";
76          cin.clear();
77          cin.ignore(10, '\n');
78          cin >> choice;
79      }
80      return choice;
81  }
82  int main() {
83      system("chcp 1251 > nul");
84      int (*cmp) (int, int) = compare;
85
86      while (true) {
87          switch (menu()) {
88          case 1:
89              system("cls");
90              cout << "Сортировка массива минимального элемента" << endl;
91              cout << "Размер массива: " << arraySize << endl;
92              printSortResult(doMinElSort(getRandomVector(), cmp));
93              system("cls");
94              break;
95          case 2:
96              system("cls");
97              cout << "Сортировка массива методом пузырька" << endl;
98              cout << "Размер массива: " << arraySize << endl;
```

```
 99              printSortResult(doBubbleSort(getRandomVector(), cmp));
100              system("cls");
101              break;
102          case 3:
103              system("cls");
104              inputArraySize();
105              system("cls");
106              break;
107          case 4:
108              system("cls");
109              inputArrayType();
110              system("cls");
111              break;
112          case 5:
113              return 0;
114          default:
115              cout << "Некорректная операция." << endl;
116              cout << "Для выхода нажмите 5." << endl;
117              cout << "Для продолжения нажмите любую клавишу" << endl;
118              while (!_kbhit());
119              system("cls");
120          }
121      }
122      return 0;
123  }
```

task2.cpp


```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <numeric>
#include <random>
#include <conio.h>
#include "../../lib/labFunc.h"

using namespace std;

int arraySize = 10;
short arrayType = 0;
string names[3] = {"Name1", "Name2", "Name3"};
int (*cmpType) (Student, Student);
int (*compare) (Student, Student) = compareNames;
int reverseCompare(Student a, Student b) {
    return -((*cmpType)(a, b));
}

Student getRandomStudent() {
    std::random_device rd;
    std::mt19937 gen(rd());
    std::uniform_int_distribution<> dName(0, 2);
    std::uniform_int_distribution<> dID(0, 9);
    std::uniform_int_distribution<> dDay(1, 31);
    std::uniform_int_distribution<> dMonth(1, 12);
    std::uniform_int_distribution<> dYear(1900, 2021);
    Student s =  { names[dName(gen)], {dDay(gen), dMonth(gen), dYear(gen)},
      ""};
    for (int i = 0; i < 8; ++i)
        s.id[i] = dID(gen) + '0';
    return s;
}

vector <Student> getRandomVector() {
    vector <Student> v;
    for (int i = 0; i < arraySize; ++i) {
        v.push_back(getRandomStudent());
    }

    switch (arrayType) {
    case 1:
        v = doMinElSort(v, compare).vec;
        v.push_back(v[0]);
        v.erase(v.begin());
        break;
    case 2:
        if (compare == cmpType)
            v = doMinElSort(v, reverseCompare).vec;
```

```
49          else
50              v = doMinElSort(v, cmpType).vec;
51          v.push_back(v[0]);
52          v.erase(v.begin());
53          break;
54      }
55      return v;
56  }
57
58  void inputArraySize() {
59      cout << "Введите размер массива." << endl;
60      cin >> arraySize;
61      if (arraySize <= 0)
62          arraySize = 10;
63  }
64
65  void inputArrayType() {
66      cout << "Введите тип массива:" << endl;
67      cout << "0 - случайное состояние" << endl;
68      cout << "1 - отсортированный" << endl;
69      cout << "2 - сортировка в обратном порядке" << endl;
70      cin >> arrayType;
71
72      if (!((arrayType >= 0) && (arrayType < 3)))
73          arrayType = 0;
74
75  }
76
77  void inputCompareType() {
78      cout << "Введите параметр для сравнения" << endl;
79      cout << "1 - name по возрастанию" << endl;
80      cout << "2 - birthDay по убыванию" << endl;
81      cout << "3 - id по возрастанию" << endl;
82      short choice; cin >> choice;
83      if (choice <= 0 || choice > 3)
84          choice = 1;
85      switch (choice) {
86      case 1:
87          cmpType = compareNames;
88          compare = cmpType;
89          break;
90      case 2:
91          cmpType = compareBirthDays;
92          compare = reverseCompare;
93          break;
94      case 3:
95          cmpType = compareIds;
96          compare = cmpType;
97          break;
```

```
 98      }
 99  }
100
101  short menu() {
102      cout << "========================Menu=========================" << endl;
103      cout << "| 1 - Сортировка методом минимального элемента       |" << endl;
104      cout << "| 2 - Сортировка пузырьковым методом                 |" << endl;
105      cout << "| 3 - Задать количество элементов массива            |" << endl;
106      cout << "| 4 - Задать начальное состояние массива             |" << endl;
107      cout << "| 5 - Задать параметр для сравнения                  |" << endl;
108      cout << "| 6 - Выход                                          |" << endl;
109      cout << "=====================================================" << endl;
110      cout << "Выберите номер:" << endl;
111      int choice;
112      cin >> choice;
113      while (cin.fail()) {
114          cout << "Ошибка ввода. Повторите ввод\n";
115          cin.clear();
116          cin.ignore(10, '\n');
117          cin >> choice;
118      }
119      return choice;
120  }
121  int main() {
122      system("chcp 1251 > nul");
123
124      while (true) {
125          short choice = menu();
126          system("cls");
127          switch (choice) {
128          case 1:
129              printSortResult(doMinElSort(getRandomVector(), compare));
130              break;
131          case 2:
132              printSortResult(doBubbleSort(getRandomVector(), compare));
133              break;
134          case 3:
135              inputArraySize();
136              break;
137          case 4:
138              inputArrayType();
139              break;
140          case 5:
141              inputCompareType();
142              break;
143          case 6:
144              return 0;
145          default:
146              cout << "Некорректная операция." << endl;
```

```
147             cout << "Для продолжения нажмите любую клавишу" << endl;
148             while (!_kbhit());
149         }
150         system("cls");
151     }
152     return 0;
153 }
```

labFunc.h

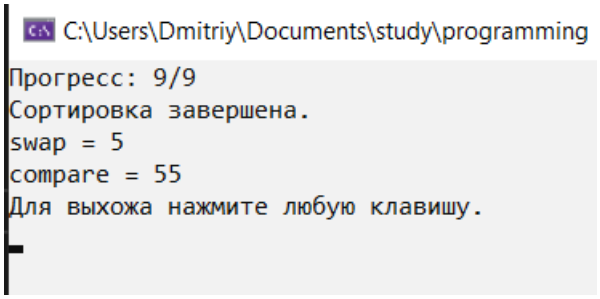

```
 1 #pragma once
 2 #include <conio.h>
 3 #include <string>
 4 #include <iostream>
 5 #include <conio.h>
 6 #include <vector>
 7 using namespace std;
 8
 9 struct Date {
10     int day;
11     int month;
12     int year;
13 };
14 struct Student {
15     std::string name;
16     Date birthDay;
17     char id[8];//номер зачётной книжки
18 };
19
20
21 template <typename T> struct sortResult {
22     vector <T> vec;
23     int change;
24     int compare;
25 };
26
27 bool operator > (Date a, Date b) {
28     return (a.year > b.year) || ((a.year == b.year) && (a.month > b.month)) || ↵
        \
29         ((a.year == b.year) && (a.month == b.month) && (a.day > b.day));
30 }
31
32 bool operator == (Date a, Date b) {
33     return ((a.day == b.day) && (b.month == a.month) && (b.year == a.year));
34 }
35
36 bool operator < (Date a, Date b) {
37     return (!(a == b)) && (!(a > b));
38 }
39
40 int compareNames(struct Student a, struct Student b) {
41     return (a.name == b.name ? 0 : (a.name < b.name ? -1 : 1));
42 }
43
44 int compareBirthDays(struct Student a, struct Student b) {
45     return (a.birthDay == b.birthDay ? 0 : (a.birthDay < b.birthDay ? -1 : 1));
46 }
47
48 int compareIds(struct Student a, struct Student b) {
```

```
49      for (int i = 0; i < 8; ++i) {
50          if (a.id[i] > b.id[i])
51              return 1;
52          else if (a.id[i] < b.id[i])
53              return 0;
54      }
55      return 0;
56  }
57
58  void printProgressBar(double start, double end) {
59      cout << '\r';
60      cout << "\x1b[0K";
61      cout << "Прогресс: " << start << '/' << end;
62  }
63
64
65
66
67  template <typename T> sortResult<T> doMinElSort(std::vector <T> v, \
68      int (*cmp) (T, T)) {
69      int change = 0, compare = 0;
70      int i, j, index;
71      int n = static_cast<int>(v.size());
72      for (i = 0; i < n; ++i) {
73          index = i;
74          printProgressBar(i, n - 1);
75          for (j = i + 1; j < n; ++j) {
76              //v[j] < v[index]
77              if (cmp(v[j], v[index]) == -1)
78                  index = j;
79              compare++;
80          }
81
82          if (index != i) {
83              swap(v[index], v[i]);
84              ++change;
85          }
86          ++compare;
87      }
88      return { v, change, compare };
89  }
90
91  template <typename T> sortResult<T> doBubbleSort(std::vector <T> v, \
92      int (*cmp)(T, T)) {
93      int change = 0, compare = 0, changesAmount;
94      int i, j;
95      int n = static_cast<int>(v.size());
96      for (i = 0; i < n; ++i) {
97          printProgressBar(i, n - 1);
```

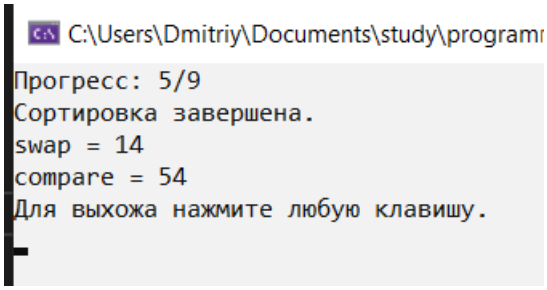
```
 98         changesAmount = 0;
 99         for (j = n - 1; j > 0; --j) {
100             //v[j] > v[j - 1]
101             if (cmp(v[j], v[j - 1]) == -1) {
102                 swap(v[j], v[j - 1]);
103                 ++changesAmount;
104                 ++change;
105             }
106             ++compare;
107         }
108         if (!changesAmount)
109             return { v, change, compare };
110     }
111     return { v, change, compare };
112 }
113
114 template <typename T> void printArray(vector <T> v) {
115     for (auto& i : v)
116         cout << i << endl;
117 };
118
119 void printStudents(vector <Student> v) {
120     for (auto& i : v) {
121         cout << i.name << " ";
122         auto b = i.birthDay;
123         cout << b.day << " " << b.month << " " << b.year << " ";
124         cout << i.id << endl << endl;
125     }
126 }
127
128 template <typename T> void printSortResult(sortResult<T> res) {
129
130     cout << endl << "Сортировка завершена." << endl;
131     cout << "swap = " << res.change << endl;
132     cout << "compare = " << res.compare << endl;
133     cout << "Для выхожа нажмите любую клавишу." << endl;
134     while (!_kbhit());
135 }
136
```

Анализ результатов
Задача 2

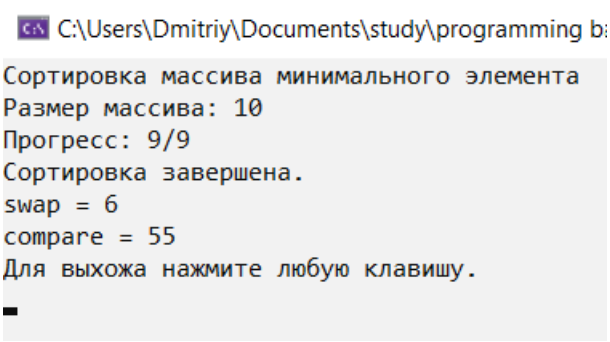
Метод минимального:

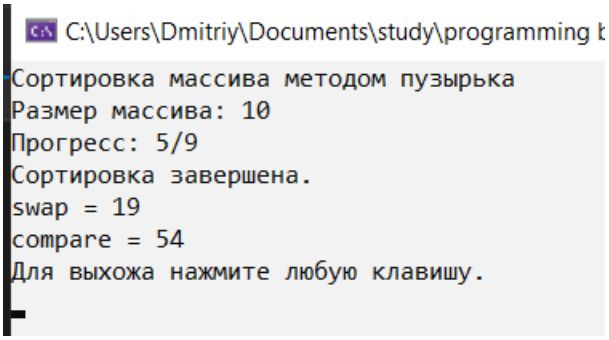


Метод пузырька:



Задача 1





Таблицы результатов:

Алгоритм	Размер массива					
	100					
	Состояние массива					
	I		II		III	
	compare	swap	compare	swap	compare	swap
bubbleEnd	8712	2344	198	87	9405	4631
minMax	5050	90	5050	19	5050	62

Алгоритм	Размер массива					
	10000					
	Состояние массива					
	I		II		III	
	compare	swap	compare	swap	compare	swap
bubbleEnd	23815435	95020497	19998	9520	94980501	47605331
minMax	50005000	9548	50005000	20	50005000	5781