**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Основы информатики»

Отчет по лабораторной работе №5

# «Сортировка одномерного числового массива»

Вариант 7

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-13 |  | преподаватель каф. ИУ5 |
| Коновалов Максим |  | Папшев И.С |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2020 г.

## Содержание отчета

Отчет по лабораторной работе состоит из 4-х разделов, отражающих основные этапы разработки программы:

- Постановка задачи;

- Разработка алгоритма;

- Кодирование (соответствующий раздел отчета называется «Текст программы»);

- Тестирование (соответствующий раздел отчета называется «Анализ результатов»).

В разделе «Постановка задачи» приведен текст задания.

Раздел «Разработка алгоритма» содержит следующую информацию:

* краткое описание алгоритма (для 2-х методов сортировки);
* описание входных, выходных и вспомогательных данных с указанием их идентификаторов и типов;
* схему алгоритмов, состоящую из двух частей: общей (укрупненной) схемы и уточненной схемы ***одного из блоков***:
* Метод пузырькового всплытия (блок 0);
* Метод минимакса (блок 1);

Номер блока равен остатку от деления порядкового номера студента в списке группы на 2.

В разделе «Текст программы» приведен листинг программы, включающий ***необходимые комментарии***.

В разделе «Анализ результатов» приведены распечатки экранов и таблица результатов для п.2 задания.   
Для экономии краски изображения имеют белый фон.

## 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[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.

## Разработка алгоритма

* краткое описание алгоритма (для 2-х методов сортировки);

**Метод Максимального-Минимального:**

Шаг 1. Среди всех N элементов находим максимум (так как сортируем по

убыванию) и ставим его на 1-ое место (1-ий и n-ый элемент меняются местами)

Шаг 2. Среди оставшихся N-1 элементов, то есть начиная со 2-го, находим

максимум и ставим его на 2-ое место (2-ий и n-ый элемент меняются местами)

Шаг 3. Среди оставшихся N-2 элементов, начиная с 3-го, находим максимум и

ставим его на 3-е место (3-ий и n-ый элемент меняются местами)

**Пузырьковая сортировка**

Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются N-1 раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован. При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент массива ставится на своё место в конце массива рядом с предыдущим «наибольшим элементом», а наименьший элемент перемещается на одну позицию к началу массива («всплывает» до нужной позиции как пузырёк в воде, отсюда и название алгоритма).

* описание входных, выходных и вспомогательных данных с указанием их идентификаторов и типов;

***Для первой задачи используются след. данные***

int N,ni,swapCounter; //N – длинна массива, ni число проверок в цикле, swapCounter – количество перестановок

***Для 2-ой задачи используются след. данные :***

Student group[10] = // Объявляем и инициализируем группу студентов из 10 человек

{

{"Vasya",1,5,2001,"yy43-n"},

{"Anton",5,7,2002,"1443-r"},

{"Kolya",3,8,2000,"u243-i"},

{"Leha",14,12,2005,"1253-u"},

{"Alex",27,7,2006,"1223-a"},

{"Senya",10,2,2003,"1143-f"},

{"Danil",19,12,2000,"1643-k"},

{"Vova",18,4,2001,"1288-h"},

{"Anatolii",8,11,2001,"1773-a"},

{"Peter",10,11,2001,"1343-n"}

};

Student Tmax = {" ",0,0,0," "}; //Предварительный "Максимальный студент"

int TmaxIndex; //Индекс максимального эллемента

struct Date

{

int day; //День

int month; //Месяц

int year; //Год

};

struct Student

{

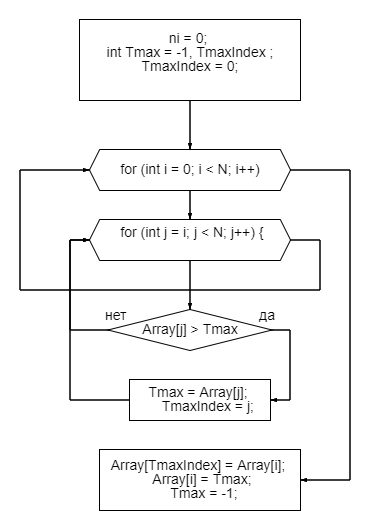
string name; //Имя сутдента

Date birthDay; //День рождения

char id[8]; //номер зачётной книжки

};

typedef bool (\*CMP)(Student& obj1, Student& obj2); //Создаем имя новому \*типу, который является указателем на функцию, принимающую 2 объекта

* схему алгоритмов, состоящую из двух частей: общей (укрупненной) схемы и уточненной схемы ***одного из блоков***:
* ~~Метод пузырькового всплытия (блок 0);~~
* Метод минимакса (блок 1);

Номер блока равен остатку от деления порядкового номера студента в списке группы на 2. (7%2 =1)

## 3- Текст программы

3.1

#include <iostream>

#include <iomanip>

using namespace std;

int N,ni,swapCounter;

void ShowCounter() {

cout << "Количество swap >> " << swapCounter<<endl;

cout << "Количество проверок >> " << ni << endl;

}

int Rand(int a, int b) {

b++;

return (a + rand() %(b-a) );

} //Функция рандома на отрезке [a,b]

void Randomize(int\* Array) {

for (int i = 0; i <= N; i++) {

Array[i] = Rand(1, 10);

}

}

void BubbleSort(int\* Array) {

ni = 0;

swapCounter = 0;

for (int i = N - 1; i >= 1; i--)

for (int j = 0; j < i; j++)

{

if (Array[j] > Array[j + 1])

{

int temp = Array[j];

Array[j] = Array[j + 1];

Array[j + 1] = temp;

swapCounter++;

}

ni++;

}

}

void MaxMinSort(int\* Array) {

ni = 0;

swapCounter = 0;

int Tmax = -1, TmaxIndex ;

TmaxIndex = 0;

for (int i = 0; i < N; i++) {

for (int j = i; j < N; j++) {

if (Array[j] > Tmax) {

Tmax = Array[j];

TmaxIndex = j;

swapCounter++;

}

ni++;

}

Array[TmaxIndex] = Array[i];

Array[i] = Tmax;

Tmax = -1;

}

}

void Reverse(int\* Array) {

for (int i = 0; i < N/2; i++) {

int temp = Array[i];

Array[i] = Array[N - i-1];

Array[N - i-1] = temp;

}

}

void Show(int\* Array) {

for (int i = 0; i < N; i++) {

cout << Array[i] << " ";

}

cout << " " << endl;

}

int main()

{

system("chcp 1251 > nul");

char on = 'y';

do {

N = 1;

cout << "Выберите размер массива>> ";

cin >> N;

int\* array1;

array1 = new int[N];

cout << "Как вы хотите заполнить массив?(1 - вручную, ост. цел значения - автоматически) >> ";

int type;

cin >> type;

if (type == 1) {

for (int i = 0; i < N; i++) {

cin >> array1[i];

}

}

else

{

Randomize(array1);

}

cout << "\n\n====================Пузырьковая сортировка====================" << endl;

cout << "Перед выполнением сортировки массив был изменен след. образом... \n";

Show(array1);

cout << "Сортировка не упорядоченного массива\n";

BubbleSort(array1);

Show(array1);

ShowCounter();

cout << "Сортировка упорядоченного по возрастанию массива\n";

cout << "array1[0]: " << array1[0];

cout << "; array1[9999]: " << array1[N-1] << endl;

array1[N-1] = array1[0] + 10;

cout << "array1[9999] = array1[0] + 10 = " << array1[N-1] << endl;

BubbleSort(array1);

Show(array1);

ShowCounter();

cout << "Сортировка упорядоченного по убыванию массива\n";

Reverse(array1);

BubbleSort(array1);

Show(array1);

ShowCounter();

cout << "\n\n====================Метод минмального, максимального====================" << endl;

cout << "Как вы хотите заполнить массив?(1 - вручную, ост. цел значения - автоматически) >> ";

cin >> type;

if (type == 1) {

for (int i = 0; i < N; i++) {

cin >> array1[i];

}

}

else

{

Randomize(array1);

}

Show(array1);

cout << "Сортировка не упорядоченного массива\n";

MaxMinSort(array1);

Show(array1);

ShowCounter();

cout << "Сортировка упорядоченного по возрастанию массива\n";

array1[N-1] = array1[0] + 10;

MaxMinSort(array1);

Show(array1);

ShowCounter();

cout << "Сортировка упорядоченного по убыванию массива\n";

Reverse(array1);

MaxMinSort(array1);

Show(array1);

ShowCounter();

// END OF PROGRAM BLOCK

cout << "\n\nПродолжить работу программы?(y/n) >> ";

cin >> on;

while (cin.fail()) {

cin.clear();

cout << "Введены некорректные данные, попробуйте снова >> ";

cin.ignore(10, '\n');

cin >> on;

}

} while (on != 'n');

return 0;

}

3.2

#include <iostream>

#include <string>

#include <iomanip>

#include "Header.h"

using namespace std;

int Menu() {

cout << "================СОРТИРОВКА МАССИВОВ===================\n";

cout << "\t1 - Исходный список группы студентов\n";

cout << "\t2 – miniMax: Сортировка по возрастанию значения поля name\n";

cout << "\t3 – miniMax: Сортировка по убыванию значения поля birthDay\n";

cout << "\t4 – miniMax: Сортировка по по возрастанию номера зачетной книжки\n";

cout << "\t5 – bubbleSort: Сортировка по возрастанию значения поля name\n";

cout << "\t6 – bubbleSort: Сортировка по убыванию значения поля birthDay\n";

cout << "\t7 – bubbleSort: Сортировка по по возрастанию номера зачетной книжки\n";

cout << "\t8 - Выход\n";

int choice;

cout << "Выберите действие\n";

cin >> choice;

while (cin.fail()) {//исключение зацикливания в случае ввода недопустимых символов

cout << "Ошибка ввода. Повторите ввод\n";

cin.clear();//сброс в потоке флага ошибки

cin.ignore(10, '\n');//извлечение из буфера потока введенных символов

cin >> choice;//повторный ввод

}

return choice;

}

struct Date

{

int day;

int month;

int year;

};

struct Student

{

string name;

Date birthDay;

char id[8];//номер зачётной книжки

friend ostream& operator<< (std::ostream& out, const Student& person);

};

std::ostream& operator<< (std::ostream& out, const Student& person)

{

out << person.name << " " << setw(7) << person.birthDay.day << "." << person.birthDay.month << "." << person.birthDay.year << setw(5) << " " << person.id;

return out;

}

template <typename T>

bool cmp1(T a, T b) { return a.name > b.name; } // Сортировка по имени

template <typename T>

bool cmp3(T a, T b) { return strcmp(a.id, b.id) > 0;} //Сортировка по зачетной книжке

template <typename T> //Сортировка по дате родждения

bool cmp2(T a, T b)

{

if (a.birthDay.year == b.birthDay.year) {

if (a.birthDay.month == b.birthDay.month)

{

return (a.birthDay.day > b.birthDay.day);

}

else return (a.birthDay.month > b.birthDay.month);

}

else

{

return(a.birthDay.year > b.birthDay.year);

}

} //Сортировка по дате

typedef bool (\*CMP)(Student& obj1, Student& obj2);

CMP cmp[3] = { cmp1,cmp2,cmp3 };

template <typename T>

void BubbleSort1(CMP c, T\* Array, unsigned short int lenght = 1) {

for (int i = lenght - 1; i >= 1; i--)

{

for (int j = 0; j < i; j++)

{

if (c(Array[j], Array[j + 1]))

{

auto temp = Array[j];

Array[j] = Array[j + 1];

Array[j + 1] = temp;

}

}

}

}

template <typename T>

void MaxMinSort(CMP c,T\* Array,unsigned short int lenght) {

Student Tmax = {" ",0,0,0," "};

int TmaxIndex;

TmaxIndex = 0;

for (int i = 0; i < lenght; i++) {

for (int j = i; j < lenght; j++) {

if (c(Array[j],Tmax)) {

Tmax = Array[j];

TmaxIndex = j;

}

}

Array[TmaxIndex] = Array[i];

Array[i] = Tmax;

Tmax = {" ",0,0,0," "};

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

Student group[10] =

{

{"Vasya",1,5,2001,"yy43-n"},

{"Anton",5,7,2002,"1443-r"},

{"Kolya",3,8,2000,"u243-i"},

{"Leha",14,12,2005,"1253-u"},

{"Alex",27,7,2006,"1223-a"},

{"Senya",10,2,2003,"1143-f"},

{"Danil",19,12,2000,"1643-k"},

{"Vova",18,4,2001,"1288-h"},

{"Anatolii",8,11,2001,"1773-a"},

{"Peter",10,11,2001,"1343-n"}

};

while (true) {

switch (Menu()) {

case 1:

Show(group,10);

break;

case 2:

MaxMinSort(cmp1, group, 10);

Reverse(group, 10);

Show(group, 10);

break;

case 3:

MaxMinSort(cmp2, group, 10);

Reverse(group, 10);

Show(group, 10);

break;

case 4:

MaxMinSort(cmp3, group, 10);

Reverse(group, 10);

Show(group, 10);

break;

case 5:

BubbleSort1(cmp1, group, 10);

Show(group, 10);

break;

case 6:

BubbleSort1(cmp2, group, 10);

Show(group, 10);

break;

case 7:

BubbleSort1(cmp3, group, 10);

Show(group, 10);

break;

case 8: return 0;//завершение работы

default:

cout << "Недопустимое действие.Повторите выбор.\n";

break;

}

}

}

## Анализ результатов

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Алгоритм | Размер массива | | | | | |
| 100 | | | | | |
| Состояние массива | | | | | |
| I | | II | | III | |
| compare | swap | compare | Swap | compare | swap |
| bubbleSort | 4881 | 2396 | 197 | 99 | 4940 | 4484 |
| minMax | 5050 | 296 | 5050 | 397 | 5050 | 100 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Алгоритм | Размер массива | | | | | |
| 10000 | | | | | |
| Состояние массива | | | | | |
| I | | II | | III | |
| compare | swap | compare | Swap | compare | swap |
| bubbleSort | 49483465 | 22646632 | 19997 | 9999 | 49492497 | 44994787 |
| minMax | 50005000 | 29198 | 50005000 | 35165 | 50005000 | 10000 |