**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 массива»

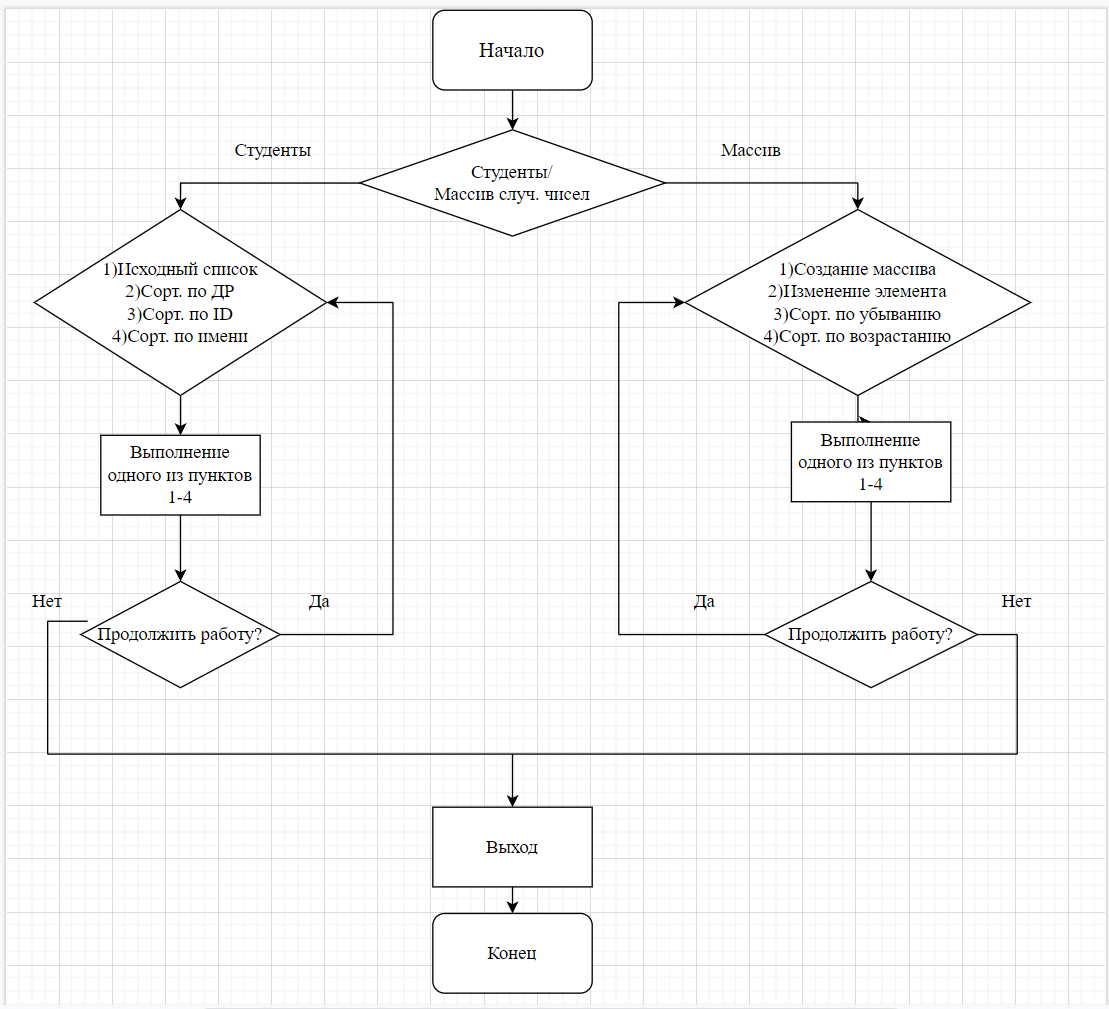
|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-15Б |  | преподаватель каф. ИУ5 |
| Коновалов Илья |  | Папшев И.С. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2021 г.

**Задача:**

Разработать функции, реализующие алгоритмы сортировки массивов методом выбора максимального (минимального) элемента и методом пузырького всплытия и создать шаблоны этих функций. Сравнить быстродействие алгоритмов сортировки в зависимости от размера и упорядоченности элементов массива.

**Алгоритм:**

Блок-схема:

Программа:

Основной файл

Переменные:

int \*arr - указатель на int, динамический массив

int arrsize - размер массива arr

int indx - индекс изменяемого элемента

int nCmp, nSwp - количество сравнений и перестановок соответсвенно

Student group - массив структур типа Student

int cnt - размер массива group

Student \*orig - динамический массив типа Student

**Код программы:**

#include **<iostream>**#include **"Sort.h"**#include **<string>**#include **<random>**#include **<iomanip>  
  
using namespace** std;  
  
**struct** Date {  
 **int** day;  
 **int** month;  
 **int** year;  
};  
**struct** Student {  
 string name;  
 Date birthDay;  
 **char** id[8];  
};  
  
**int** Menu() {  
 cout << **"\n================СОРТИРОВКА МАССИВОВ===================\n"**;  
 cout << **"\t1 - Список группы студентов\n"**;  
 cout << **"\t2 - miniMax: Сортировка по убыванию даты рождения\n"**;  
 cout << **"\t3 - bubbleEnd: Сортировка по убыванию даты рождения\n"**;  
 cout << **"\t4 - miniMax: Сортировка по возрастанию id студента\n"**;  
 cout << **"\t5 - bubbleEnd: Сортировка по возрастанию id студента\n"**;  
 cout << **"\t6 - miniMax: Сортировка по возрастанию имени студента\n"**;  
 cout << **"\t7 - bubbleEnd: Сортировка по возрастанию id студента\n"**;  
 cout << **"\t8 - Создание и инициализация динамического массива из n целых чисел\n"**;  
 cout << **"\t9 - Изменение значения элемента динамического массива\n"**;  
 cout << **"\t10 - miniMax: Сортировка целочисленного динамического массива по убыванию\n"**;  
 cout << **"\t11 - bubbleEnd: Сортировка целочисленного динамического массива по возрастанию\n"**;  
 cout << **"\t0 - Выход\n"**;  
 **int** choice;  
 cout << **"Выберите действие: "**;  
 cin >> choice;  
 **while** (cin.fail()) {  
 cout << **"Ошибка ввода. Повторите ввод\n"**;  
 cin.clear();  
 cin.ignore(10, **'\n'**);  
 cin >> choice;  
 }  
 **return** choice;  
}  
  
**bool** cmpDate(Student obj1, Student obj2) {  
 **if** (obj1.birthDay.year != obj2.birthDay.year) {  
 **return** obj1.birthDay.year > obj2.birthDay.year;  
 }  
 **else** {  
 **if** (obj1.birthDay.month != obj2.birthDay.month)  
 **return** obj1.birthDay.month > obj2.birthDay.month;  
 **else** {  
 **if** (obj1.birthDay.day != obj2.birthDay.day)  
 **return** obj1.birthDay.day > obj2.birthDay.day;  
 }  
 }  
}  
  
**bool** cmpName(Student obj1, Student obj2) {  
 **return** obj1.name < obj2.name;  
}  
  
**bool** cmpID(Student obj1, Student obj2) {  
 **return** strcmp(obj1.id, obj2.id) < 0;  
}  
  
**bool** cmpMore(**int** n1, **int** n2) {  
 **return** n1 > n2;  
}  
  
**bool** cmpLess(**int** n1, **int** n2) {  
 **return** n1 < n2;  
}  
  
**int** printDate(**int**& a, **int**& b, **int**& c, string& d, **char** e[]) {  
 cout << **"\n"** << d << **"\t"**;  
 **if** (a < 10 && b < 10) {  
 cout << **"0"** << a << **".0"** << b << **"."** << c << **"\t"**;  
 }  
 **else** {  
 **if** (a < 10 && b >= 10) {  
 cout << **"0"** << a << **"."** << b << **"."** << c << **"\t"**;  
 }  
 **else** {  
 **if** (b < 10 && a >= 10) {  
 cout << a << **".0"** << b << **"."** << c << **"\t"**;  
 }  
 **else** {  
 **if** (b >= 10 && a >= 10) {  
 cout << a << **"."** << b << **"."** << c << **"\t"**;  
 }  
 }  
 }  
 }  
 cout << e << **"\n"**;  
 **return** 0;  
}  
  
**int** iRandom(**int** a, **int** b) {  
 **return** a + ((b - a + 1) \* rand()) / RAND\_MAX;  
}  
  
**void** randomArr(**int**\*& arr, **int**& arsz) {  
 cout << **"Введите размер массива\n"**;  
 cin >> arsz;  
 arr = **new int**[arsz];  
 srand((**unsigned int**)time(NULL));  
 rand();  
 **for** (**int** i = 0; i < arsz; i++) {  
 arr[i] = iRandom(0, arsz);  
 }  
}  
  
**int** main()  
{  
  
 **int**\* arr = 0;  
 **int** arrsize = 0, indx = 0;  
  
 **int** nCmp = 0, nSwp = 0;  
 Student group[] = {  
 {**"Андрей"**, {9, 9, 2003}, **"1234569"**},  
 {**"Иван"**, {9, 10, 2003}, **"1234567"**},  
 {**"Ольга"**, {1, 1, 2004}, **"1234560"**},  
 {**"Игорь"**, {1, 10, 2003}, **"0123456"**},  
 {**"Мария"**, {30, 1, 2003}, **"0034567"**},  
 };  
 **int** n = **sizeof**(group) / **sizeof**(Student);  
 Student\* orig = **new** Student[n];  
 **for** (**int** i = 0; i < n; i++) {  
 orig[i] = group[i];  
 }  
  
 **while** (**true**) {  
 **switch** (Menu()) {  
 **case** 1:  
 system(**"cls"**);  
 cout << **"Исходный список группы студентов:\n"**;  
 **for** (**int** i = 0; i < n; i++) {  
 printDate(orig[i].birthDay.day, orig[i].birthDay.month, orig[i].birthDay.year,  
 orig[i].name, orig[i].id);  
 }  
 cout << **"\n\n"**;  
 **break**;  
  
 **case** 2:  
 system(**"cls"**);  
 cout << **"miniMax: Сортировка по убыванию даты рождения студента\n"**;  
 **for** (**int** i = 0; i < n; i++) {  
 group[i] = orig[i];  
 }  
 MinMax(group, n, cmpDate, nCmp, nSwp);  
 **for** (**int** i = 0; i < n; i++) {  
 printDate(group[i].birthDay.day, group[i].birthDay.month, group[i].birthDay.year,  
 group[i].name, group[i].id);  
 }  
 cout << **"\n\nSwaps = "** << nSwp << **"\nCompares = "** << nCmp;  
 cout << **"\n\n"**;  
 **break**;  
  
 **case** 3:  
 system(**"cls"**);  
 cout << **"bubbleEnd: Сортировка по убыванию даты рождения студента\n"**;  
 **for** (**int** i = 0; i < n; i++) {  
 group[i] = orig[i];  
 }  
 Bubble(group, n, cmpDate, nCmp, nSwp);  
 **for** (**int** i = 0; i < n; i++) {  
 printDate(group[i].birthDay.day, group[i].birthDay.month, group[i].birthDay.year,  
 group[i].name, group[i].id);  
 }  
 cout << **"\n\nSwaps = "** << nSwp << **"\nCompares = "** << nCmp;  
 cout << **"\n\n"**;  
 **break**;  
  
 **case** 4:  
 system(**"cls"**);  
 cout << **"miniMax: Сортировка по возрастанию id студента\n"**;  
 **for** (**int** i = 0; i < n; i++) {  
 group[i] = orig[i];  
 }  
 MinMax(group, n, cmpID, nCmp, nSwp);  
 **for** (**int** i = 0; i < n; i++) {  
 printDate(group[i].birthDay.day, group[i].birthDay.month, group[i].birthDay.year,  
 group[i].name, group[i].id);  
 }  
 cout << **"\n\nSwaps = "** << nSwp << **"\nCompares = "** << nCmp;  
 **break**;  
  
 **case** 5:  
 system(**"cls"**);  
 cout << **"bubbleEnd: Сортировка по возрастанию id студента\n"**;  
 **for** (**int** i = 0; i < n; i++) {  
 group[i] = orig[i];  
 }  
 Bubble(group, n, cmpID, nCmp, nSwp);  
 **for** (**int** i = 0; i < n; i++) {  
 printDate(group[i].birthDay.day, group[i].birthDay.month, group[i].birthDay.year,  
 group[i].name, group[i].id);  
 }  
 cout << **"\n\nSwaps = "** << nSwp << **"\nCompares = "** << nCmp;  
 cout << **"\n\n"**;  
 **break**;  
  
 **case** 6:  
 system(**"cls"**);  
 MinMax(group, n, cmpName, nCmp, nSwp);  
 **for** (**int** i = 0; i < n; i++) {  
 group[i] = orig[i];  
 }  
 **for** (**int** i = 0; i < n; i++) {  
 printDate(group[i].birthDay.day, group[i].birthDay.month, group[i].birthDay.year,  
 group[i].name, group[i].id);  
 }  
 cout << **"\n\nSwaps = "** << nSwp << **"\nCompares = "** << nCmp;  
 cout << **"\n\n"**;  
 **break**;  
  
 **case** 7:  
 system(**"cls"**);  
 Bubble(group, n, cmpName, nCmp, nSwp);  
 **for** (**int** i = 0; i < n; i++) {  
 group[i] = orig[i];  
 }  
 **for** (**int** i = 0; i < n; i++) {  
 printDate(group[i].birthDay.day, group[i].birthDay.month, group[i].birthDay.year,  
 group[i].name, group[i].id);  
 }  
 cout << **"\n\nSwaps = "** << nSwp << **"\nCompares = "** << nCmp;  
 cout << **"\n\n"**;  
 **break**;  
  
 **case** 8:  
 system(**"cls"**);  
 **delete**[] arr;  
 randomArr(arr, arrsize);  
 cout << **"\n\n"**;  
 **break**;  
  
 **case** 9:  
 system(**"cls"**);  
 cout << **"Введите индекс изменяемого элемента массива\n"**;  
 cin >> indx;  
 cout << **"Введите новое значение элемента\n"**;  
 cin >> arr[indx - 1];  
 cout << **"\n\n"**;  
 **break**;  
  
 **case** 10:  
 system(**"cls"**);  
 MinMax(arr, arrsize, cmpMore, nCmp, nSwp);  
 cout << **"\n\nSwaps = "** << nSwp << **"\nCompares = "** << nCmp;  
 cout << **"\n\n"**;  
 **break**;  
  
 **case** 11:  
 system(**"cls"**);  
 Bubble(arr, arrsize, cmpLess, nCmp, nSwp);  
 cout << **"\n\nSwaps = "** << nSwp << **"\nCompares = "** << nCmp;  
 cout << **"\n\n"**;  
 **break**;  
  
 **case** 0:  
 **return** 0;  
 **break**;  
 }  
 }  
 system(**"pause"**);  
 **return** 0;  
}

Шаблон:

**template** <**typename** T> **void** Bubble(T arr[], **int** n, **bool**(\*cmp)(T obj1, T obj2), **int**& nCmp, **int**& nSwp)  
{  
 **int** i, j;  
 nCmp = 0;  
 nSwp = 0;  
 **int** flag = 1;  
 **for** (i = 0; (i < n-1) **and** (flag==1); i++) {  
 flag = 0;  
 **for** (j = n-1; j > i; j--) {  
 nCmp++;  
 **if** (cmp(arr[j], arr[j-1])) {  
 std::swap(arr[j], arr[j-1]);  
 flag = 1;  
 nSwp++;  
 }  
 }  
 }  
}

**Результаты:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Алгоритм | Размер массива | | | | | |
| 100 | | | | | |
| Состояние массива | | | | | |
| I | | II | | III | |
| compare | swap | compare | swap | compare | swap |
| bubbleEnd | 4845 | 2175 | 4950 | 59 | 4950 | 4908 |
| minMax | 4950 | 98 | 65 | 4950 | 4950 | 4988 |

I – исходный со случайно расположенными числами

II – исходный массив, предварительно отсортированный по возрастанию

III - исходный массив, предварительно отсортированный по убыванию

**Вывод:**

Научился разрабатывать функции, реализующие функции сортировки пузырьком и выбором минимального элемента, создавать шаблоны этих функций и использовать в качестве параметра указатель на функцию, чтобы сравнивать значения полей структуры.