Министерство образования Российской Федерации  
НИЖНЕВАРТОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И МАТЕМАТИКИ

**Контрольная работа по предмету**

**«Языки и методы программирования»**

**Вариант 19**

Выполнил:

студент гр. 3312 ИВТ

*Лысенко Олеся Евгеньевна*

Нижневартовск – 2024

# Исходный код программы (Visual Studio C++)

ConsoleApplication.cpp (основная программа):

#include "All\_Ex.h"

#include <iostream>

#include <fstream>

#include <ctime>

#include <vector>

#include <algorithm>

#include <conio.h>

int main() {

setlocale(LC\_ALL, "Rus");

using namespace std;

char key;

do {

int choice;

cout << "Select a function to perform:" << endl;

cout << "1. The first function" << endl;

cout << "2. The second function" << endl;

cout << "3. The third function" << endl;

cout << "4. Exit the program" << endl;

cout << "Select the function number (or 4 to exit): ";

cin >> choice;

switch (choice) {

case 1:

runFirstCode();

break;

case 2:

runSecondCode();

break;

case 3:

runThirdCode();

break;

case 4:

cout << "Exit the program." << endl;

return 0;

default:

cout << "Incorrect choice." << endl;

break;

}

cout << "Press any key to continue or ESC to exit...";

key = \_getch();

cout << endl;

} while (key != 27);

return 0;

}

All\_Ex.h (объявление функций/заданий):

#ifndef HEADER3\_H

#define HEADER3\_H

#include <string>

#include <vector>

#include <iostream>

#include "All\_Ex.h"

void runThirdCode();

void runFirstCode();

void runSecondCode();

#endif

All\_Ex2.cpp (задания):

#include <iostream>

#include <fstream>

#include <algorithm>

#include <random>

#include <string>

#include <sstream>

#include <vector>

#include <ctime>

#include <conio.h>

#include "All\_Ex.h"

//1.Дан массив целых чисел из n элементов, заполненный случайным образом числами из заданного пользователем промежутка.

// Найти максимальный из четных элементов и возвести его в квадрат. Отсортировать получившийся массив и вывести его на экран и в файл.

void runFirstCode() {

using namespace std;

srand(time(nullptr));

int n;

cout << "Enter the size of the array: ";

cin >> n;

int lower\_bound, upper\_bound;

cout << "Enter the lower and upper boundaries of the gap separated by a space: ";

cin >> lower\_bound >> upper\_bound;

int\* arr = new int[n];

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

arr[i] = rand() % (upper\_bound + 1 - lower\_bound) + lower\_bound;

cout << arr[i] << " ";

}

int max\_even = 0;

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

if (arr[i] % 2 == 0 && arr[i] > max\_even) {

max\_even = arr[i];

}

}

if (max\_even == 0) {

cout << "There are no even numbers in the array." << endl;

delete[] arr;

}

else {

cout << "\nThe maximum even element: " << max\_even << endl;

max\_even \*= max\_even;

cout << "The square of the maximum even element: " << max\_even << endl;

}

for (int i = 0; i < n - 1; ++i) {

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

if (arr[j] > arr[j + 1]) {

swap(arr[j], arr[j + 1]);

}

}

}

cout << "Sorted array:" << endl;

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

cout << arr[i] << " ";

}

cout << endl;

ofstream outfile("sorted\_array.txt");

if (outfile.is\_open()) {

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

outfile << arr[i] << " ";

}

outfile.close();

cout << "The sorted array is written to a file sorted\_array.txt." << endl;

}

else {

cerr << "The file could not be opened for writing." << endl;

}

delete[] arr;

cout << "Press any key to continue...";

\_getch();

system("cls");

}

// 2.Дан двумерный массив размером n\*m, заполненный случайными числами.

// Найти строку с максимальным произведением элементов и в ней первый четный элемент и вывести их в файл.

void runSecondCode() {

using namespace std;

srand(time(nullptr));

int n, m;

cout << "Enter the number of rows and columns (n and m): ";

cin >> n >> m;

int\*\* matrixn = new int\* [n];

int\* matrixm = new int[m];

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

matrixn[i] = new int[m];

for (int j = 0; j < m; ++j) {

matrixn[i][j] = rand() % 100 + 1;

cout << matrixn[i][j] << " ";

}

cout << endl;

}

int max\_product = 0;

int max\_row\_index = -1;

int max\_even\_element = 0;

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

int product = 1;

for (int j = 0; j < m; ++j) {

product \*= matrixn[i][j];

}

if (product > max\_product) {

max\_product = product;

max\_row\_index = i;

}

}

if (max\_row\_index != -1) {

bool found\_even = false;

max\_product = 1;

for (int j = 0; j < m; ++j) {

max\_product \*= matrixn[max\_row\_index][j];

if (matrixn[max\_row\_index][j] % 2 == 0 && !found\_even) {

max\_even\_element = matrixn[max\_row\_index][j];

found\_even = true;

}

}

}

ofstream outfile("result.txt");

if (outfile.is\_open()) {

if (max\_row\_index != -1) {

outfile << "A string with the maximum product of elements:" << endl;

for (int j = 0; j < m; ++j) {

outfile << matrixn[max\_row\_index][j] << " ";

}

outfile << endl;

outfile << "The first even element in this row: " << max\_even\_element << endl;

outfile << "The product of the elements: " << max\_product << endl;

}

else {

outfile << "The array is empty." << endl;

}

outfile.close();

cout << "The result is written to a file result.txt." << endl;

}

else {

cerr << "The file could not be opened for writing." << endl;

}

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

delete[] matrixn[i];

}

delete[] matrixn;

delete[] matrixm;

cout << "Press any key to continue...";

\_getch();

system("cls");

}

//3 .Создать массив, элементами которого являются структуры – список команд «Что? Где? Когда?».

// В записи должны содержаться название, имя капитана, фамилия капитана, город, статус (школьная, студенческая, взрослая, корпоративная), рейтинг.

// Отсортировать записи по рейтингу.

// Запрограммировать возможность вывода всех команд из города, введенного пользователем, запись данных в файл и чтение из файла уже записанных данных.

using namespace std;

struct Command {

string name;

string captain\_first\_name;

string captain\_last\_name;

string city;

string status;

int rating;

};

vector<string> readDataFromFile(const string& filename) {

vector<string> data;

ifstream file(filename);

if (file.is\_open()) {

string value;

while (getline(file, value)) {

data.push\_back(value);

}

file.close();

}

else {

cerr << "Unable to open file" << filename << endl;

}

return data;

}

string getRandomValue(const vector<string>& values) {

if (!values.empty()) {

random\_device rd;

mt19937 gen(rd());

uniform\_int\_distribution<> dis(0, values.size() - 1);

int index = dis(gen);

return values[index];

}

else {

cerr << "No values available." << endl;

return "";

}

}

void printCommands(const Command commands[], int size) {

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

cout << "Name: " << commands[i].name << ", Captain: " << commands[i].captain\_first\_name << " "

<< commands[i].captain\_last\_name << ", City: " << commands[i].city << ", Status: " << commands[i].status

<< ", Rating: " << commands[i].rating << endl;

}

}

bool sortByRating(const Command& cmd1, const Command& cmd2) {

return cmd1.rating < cmd2.rating;

}

void runThirdCode() {

setlocale(LC\_ALL, "Ru");

srand(time(nullptr));

int arraySize;

cout << "Enter the number of commands for the random: ";

cin >> arraySize;

Command\* commands = new Command[arraySize];

vector<string> names = readDataFromFile("names.txt");

vector<string> cities = readDataFromFile("cities.txt");

vector<string> statuses = readDataFromFile("statuses.txt");

vector<string> captain\_first\_names = readDataFromFile("first\_names.txt");

vector<string> captain\_last\_names = readDataFromFile("last\_names.txt");

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

commands[i].name = getRandomValue(names);

commands[i].captain\_first\_name = getRandomValue(captain\_first\_names);

commands[i].captain\_last\_name = getRandomValue(captain\_last\_names);

commands[i].city = getRandomValue(cities);

commands[i].status = getRandomValue(statuses);

commands[i].rating = rand() % 10 + 1;

}

cout << "Commands before sorted:" << endl;

printCommands(commands, arraySize);

sort(commands, commands + arraySize, [](const Command& cmd1, const Command& cmd2) { return sortByRating(cmd1, cmd2); });

cout << "After sorted Commands:" << endl;

printCommands(commands, arraySize);

string cityInput;

cout << "\nEnter a city to display commands from: ";

cin >> cityInput;

cout << "Commands from " << cityInput << ":" << endl;

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

if (commands[i].city == cityInput) {

cout << "Name: " << commands[i].name << ", Captain: " << commands[i].captain\_first\_name << " "

<< commands[i].captain\_last\_name << ", City: " << commands[i].city << ", Status: " << commands[i].status

<< ", Rating: " << commands[i].rating << endl;

}

}

ofstream outFile("commands.txt");

if (outFile.is\_open()) {

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

outFile << "[ " << i + 1 << " ]" << " commands\_name: " << commands[i].name << " | captain\_first\_name: " << commands[i].captain\_first\_name << " | captain\_last\_name: " << commands[i].captain\_last\_name << " | city: "

<< commands[i].city << " | status: " << commands[i].status << " | rating: " << commands[i].rating << endl;

}

outFile.close();

cout << "\nData written to file 'commands.txt'." << endl;

}

else {

cerr << "Unable to open file for writing." << endl;

}

delete[] commands;

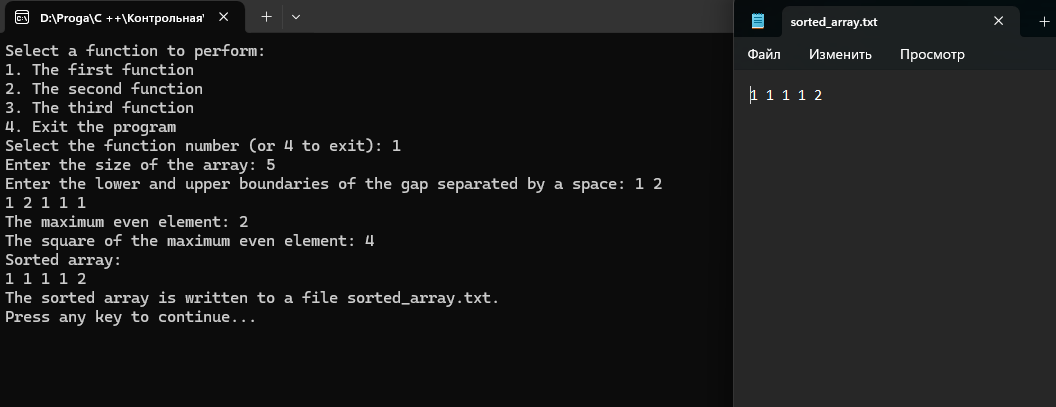
cout << "Press any key to continue...";

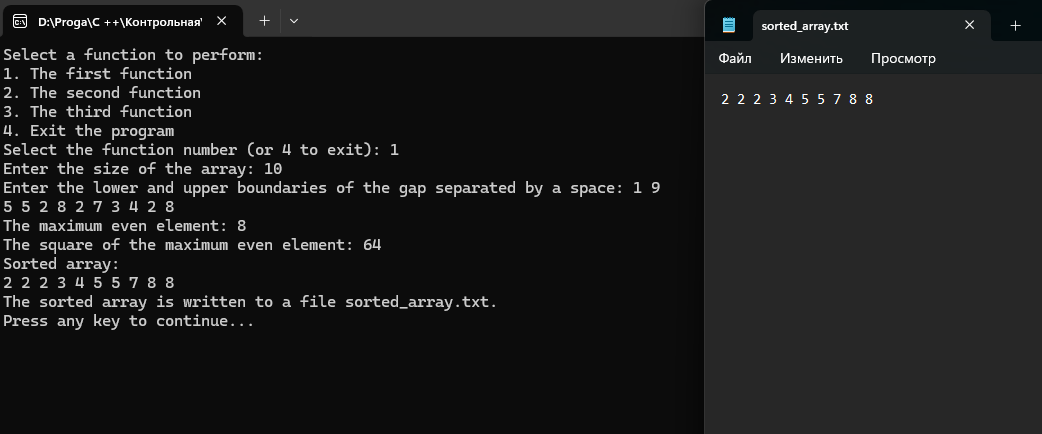
\_getch();

system("cls");

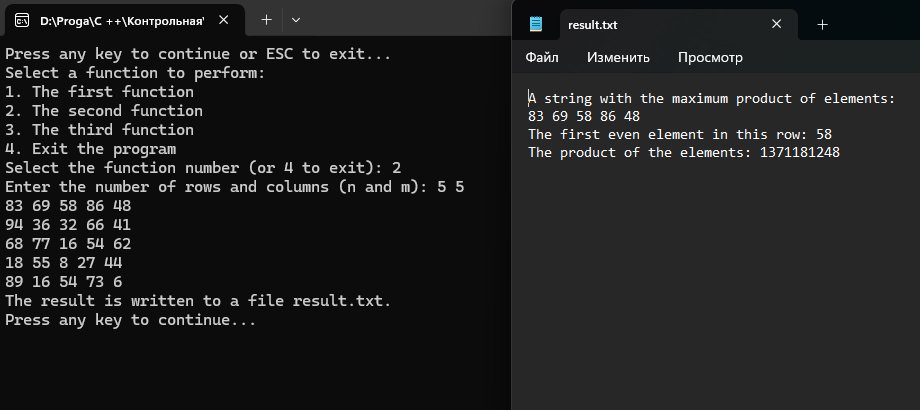
# **Запуск программы**

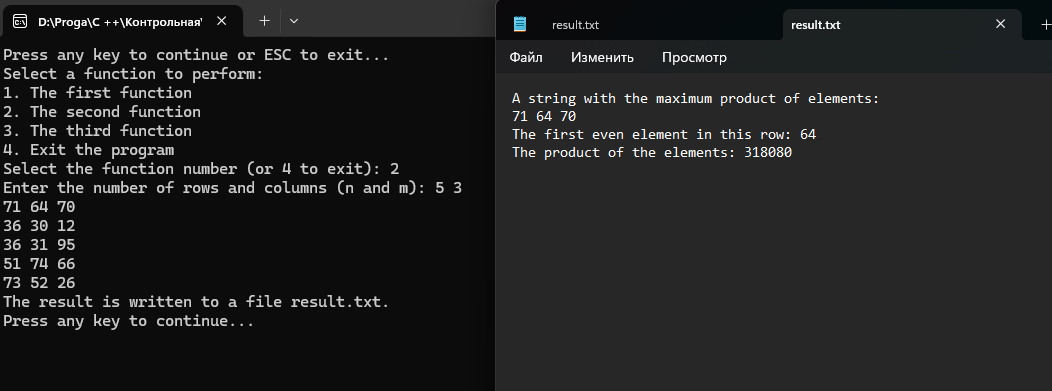
Задание 1:





Задание 2:





Задание 3:

