

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт (технический университет)»

Дисциплина: «Программирование»

Отчёт по лабораторной работе № 2

**Лабораторная работа №2. Методы сортировки**

**Выполнил студент группы №485:**

**Проверили:**

Иван Григорьевич Корниенко

Алексей Константинович Федин

Санкт-Петербург

2019

# Постановка задачи

Необходимо составить программу для сортировки массива данных методами: пузырька, отбора, вставки, Шелла и быстрой сортировки. Вывести на экран неупорядоченный и упорядоченные массивы данных. Составить сравнительную таблицу эффективности методов, в которой необходимо указать число сравнений и перестановок переменных в каждом методе сортировки. Упорядочить каждую нечетную строку матрицы по возрастанию суммы значений цифр элементов матрицы.

# Исходные данные

В качестве исходных данных программа использует вводимое пользователем количество строк и столбцов. В случае выбора пользователем заполнения массива из файла, программа запросит ввести путь к этому файлу, где первыми двумя значениями должны быть количества строк и столбцов, остальные же числа – элементы массива.

# Особые ситуации

* Если значения строк или столбцов меньше 1, то программа попросит ввести эти значения заново.
* Если пользователь при указании пути к файлу будет использовать запрещённые имена, например: con, aux и т.д., то программа попросит ввести путь к файлу заново.
* Если в файле, из которого должен быть заполнен массив, вместо числа будет найден символ, то программа попросит пользователя исправить файл и ввести путь заново.

# Математические методы и алгоритмы решения задач

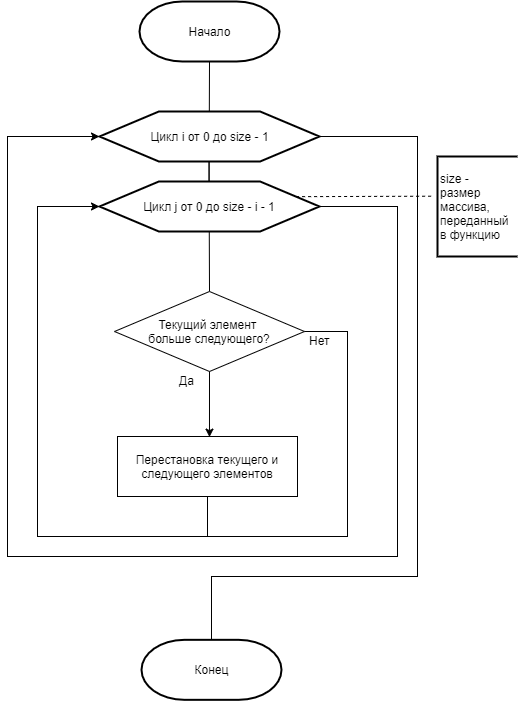


Рисунок 1 – блок-схема сортировки пузырьком

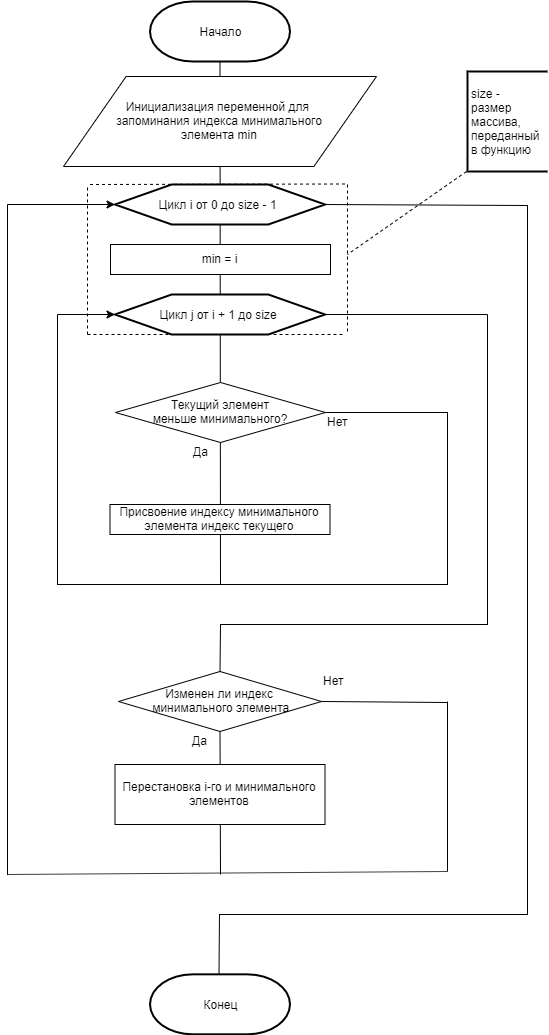


Рисунок 2 – блок-схема сортировки отбором

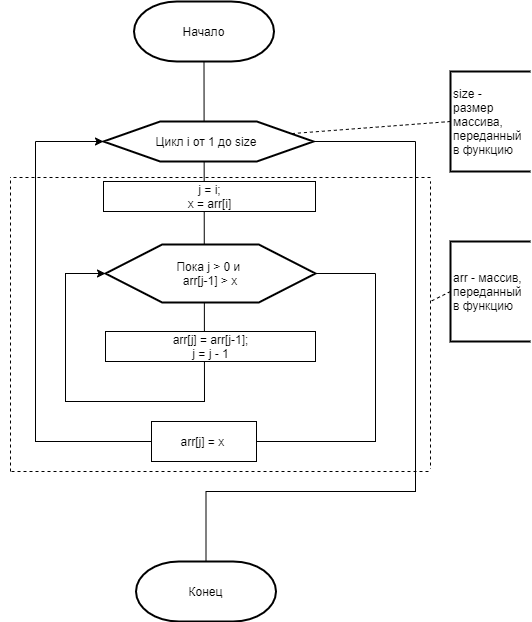


Рисунок 3 – блок-схема сортировки вставками

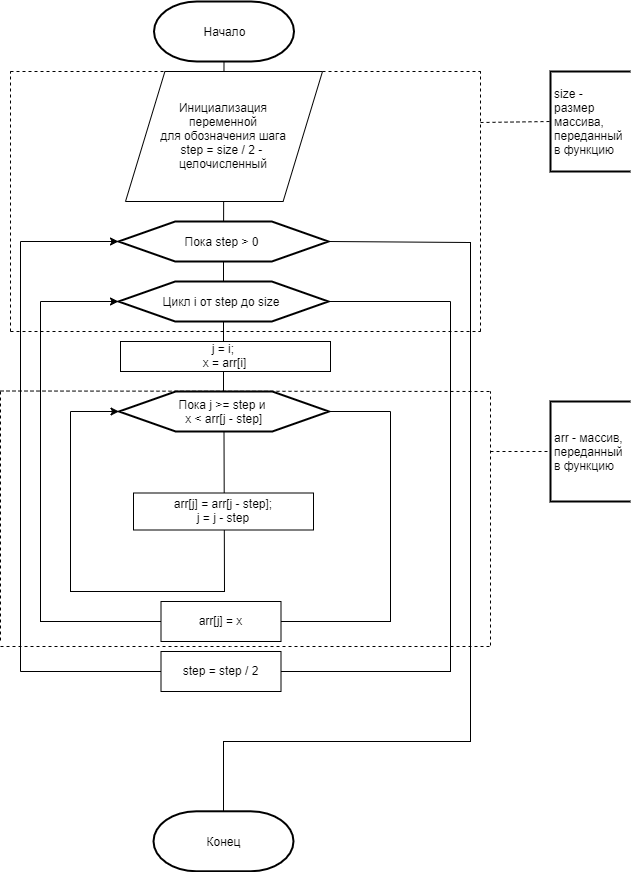


Рисунок 4 – блок-схема сортировки Шелла

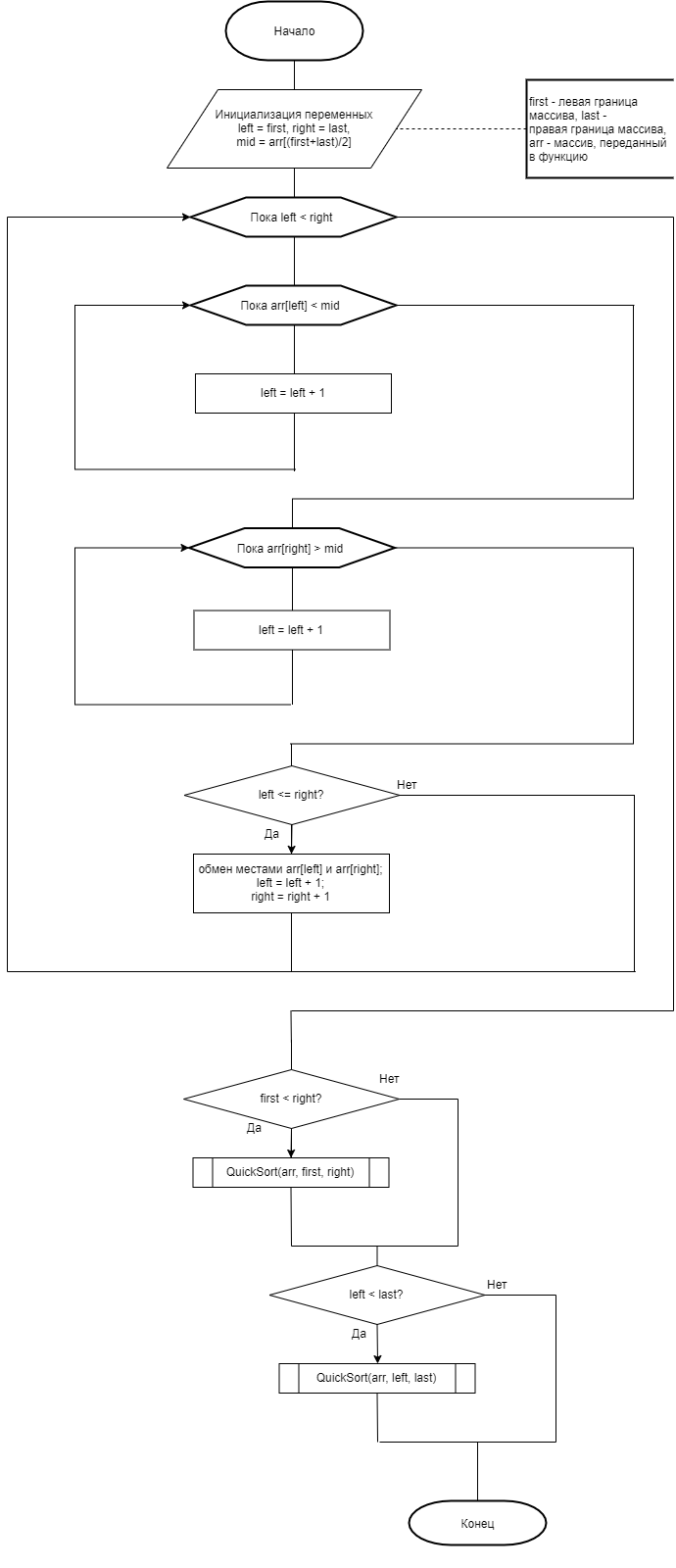


Рисунок 5 – блок-схема быстрой сортировки

Согласно постановке задачи, для составления программы будут использоваться алгоритмы, блок-схемы которых представлены выше.

# Форматы представления данных

Программа использует следующие переменные:

Таблица 1 – Переменные, используемы в программе

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Описание** |
| n | int | Количество строк массива |
| m | int | Количество столбцов массива |
| running | bool | Определяет, запущена ли программа |
| menu | int | Для ввода пункта меню |
| i | int | Для цикла чередования строк |
| j | int | Для цикла чередования столбцов |
| filePath | string | Для ввода пути к файлу |
| file | ifstream | Для считывания данных из файла |
| origArr | int\*\* | Массив данных, вводимый пользователем |
| resultArr | int\*\* | Отсортированный массив |
| choice | bool | Выбор пользователя сохранить данные в файл или нет |
| compares | int | Количество сравнений |
| swaps | int | Количество перестановок |
| line | int\* | Нечетная строка из массива |
| min | int | Для задания индекса минимального эл-та в сортировке отбором |
| step | int | Для обозначения шага в сортировке Шелла |
| comparesAndSwaps | int\*\* | Для счёта сравнений и перестановок |
| right | int | Правая граница в быстрой сортировке |
| left | int | Левая граница в быстрой сортировке |
| pivot | int | Разрешающий элемент |

Для задания максимального и минимального размера массива, а также обозначения максимального пункта меню используются следующие константы:

Таблица 2 – Константы, используемы в программе

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Тип** | **Значение** | **Описание** |
| INT\_MAX | const int | 2147483647 | Максимальное целое число |
| maxMenuNumber | const int | 5 | Максимальный номер пункта меню |
| minSize | const int | 1 | Минимальное значение строки/столбца |
| maxRandValue | const int | 100 | Максимальный случайный элемент |

# Структура программы

В силу большого количества функций программа разделена на пять исполняемых модулей, из которых один является основным и отвечает за запуск программы, четыре оставшихся содержат в себе функции, необходимые для работы программы.

Модуль Source:

Таблица 3 – Функции, составляющие модуль Source

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| main | Запуск программы |
| Menu | Вывод главного меню, выбор пункта меню |

Модуль ArrInput:

Таблица 4 – Функции, составляющие модуль ArrInput

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| EndingOutput | Вывод треугольной и исходной матрицы в консоль и в файл |
| InputN | Ввод количества строк |
| InputM | Ввод количества столбцов |
| ManualInput | Заполнение массива вручную |
| RandomInput | Заполнение массива случайными элементами |
| FilePathCheckReturnForInput | Проверка и возврат пути к файлу |
| FileInput | Проверка и заполнение массива из файла |

Модуль ArrOutput:

Таблица 5 – Функции, составляющие модуль ArrOutput

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| SetColor | Изменение цвета текста, выводимого в консоль |
| CoutWithColor | Вывод в консоль текста с заданным цветом |
| NumberLenght | Подсчет длины числа |
| OutputInConsoleTriangleArr | Вывод в консоль треугольной матрицы |
| OutputInConsoleOrigArr | Вывод в консоль исходной матрицы |
| FilePathCheckReturnForOutput | Проверка и возврат пути файла |
| OutputInFile | Вывод обеих матриц в файл |

Модуль VarCheck:

Таблица 6 – Функции, составляющие модуль VarCheck

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| InputCheck | Проверка на правильность введённых данных задаваемого типа |
| GetInt | Проверка на правильность введённых данных типа int |
| GetBool | Проверка на правильность введённых данных типа bool |

Модуль ArrChange:

Таблица 7 – Функции, составляющие модуль ArrChange

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| LineCopy | Копирование строки, для последующего её изменения в методах сортировки |
| BubbleSort | Пузырьковый метод сортировки |
| SelectingSort | Метод отбора |
| InsertSort | Метод сортировки вставками |
| ShellSort | Метод сортировки Шелла |
| QuickSort | Быстрый метод сортировки |
| SumCalcAndSort | Подсчет сумм цифр элементов строки, составление массива из них, сортировка, построение отсортированной строки |
| ArrChange | Вычленение нечетных строк, создание отсортированного массива |

# Описание хода выполнения лабораторной работы

* В ходе лабораторной работы было создано решение (Lab2) в интегрированной среде разработки Microsoft Visual Studio C++ 2019. В нём был создан проект.
* При работе программы с файлами нужно было добавить проверки на валидность имени файла, а также на то, создан ли файл или нет при сохранении.
* При получении пользовательского ввода необходимо было добавить проверку на его соответствие предполагаемому типу данных и условиям выбора.
* Перед повторением программы необходимо очищать память для того, чтобы не возникало непредвиденных ошибок, связанных с заполнением областей памяти старыми числами.
* Программа после запуска выдавала одни и те же результаты, хотя в коде использовался вызов функции rand, возвращающей случайное число. После изучения справочной системы выяснилось, что необходимо использовать функцию srand для начальной инициализации генератора случайных чисел. После этого программа стала работать правильно.

# Результат работы программы

В результате работы программа выводит два массива различных цветов, первый массив является исходным, а второй — результатом работы программы, а также таблица с методами сортировки и их значениями сравнений и перестановок.

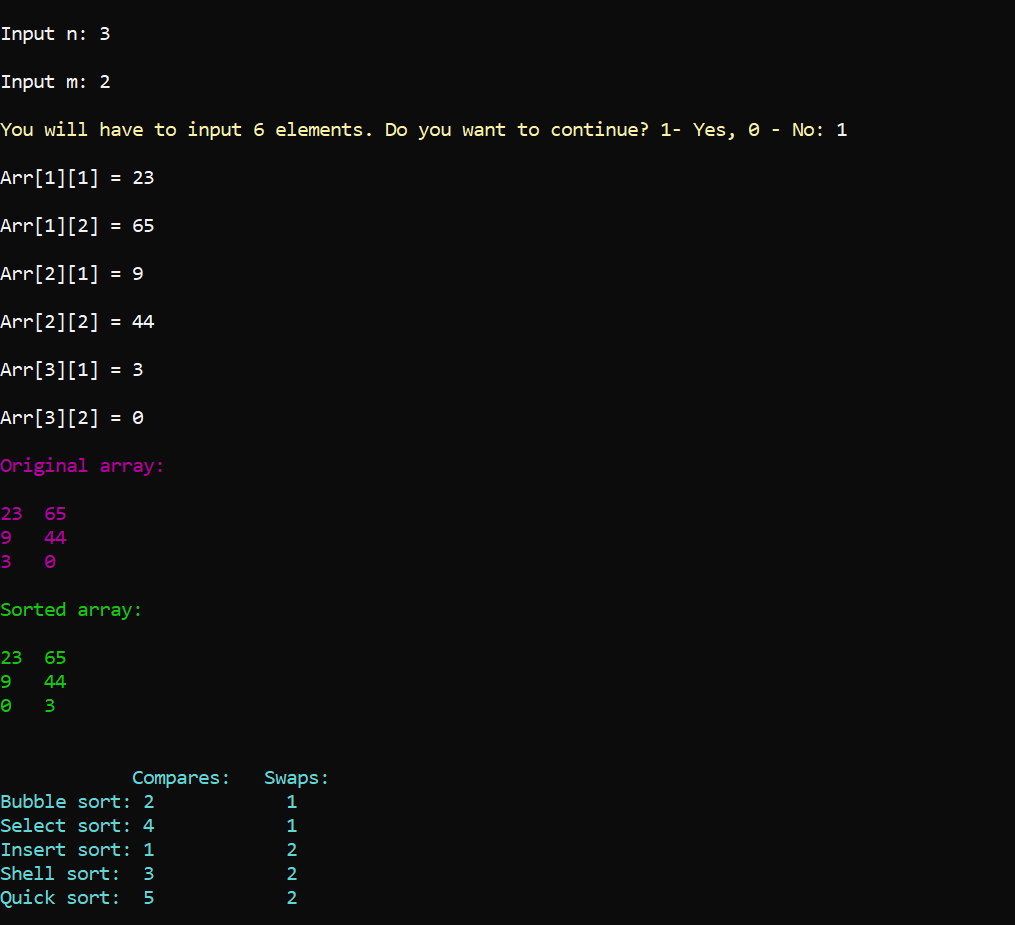


Рисунок 6 – Заполнение массива с клавиатуры



Рисунок 7 – Заполнение массива случайными числами

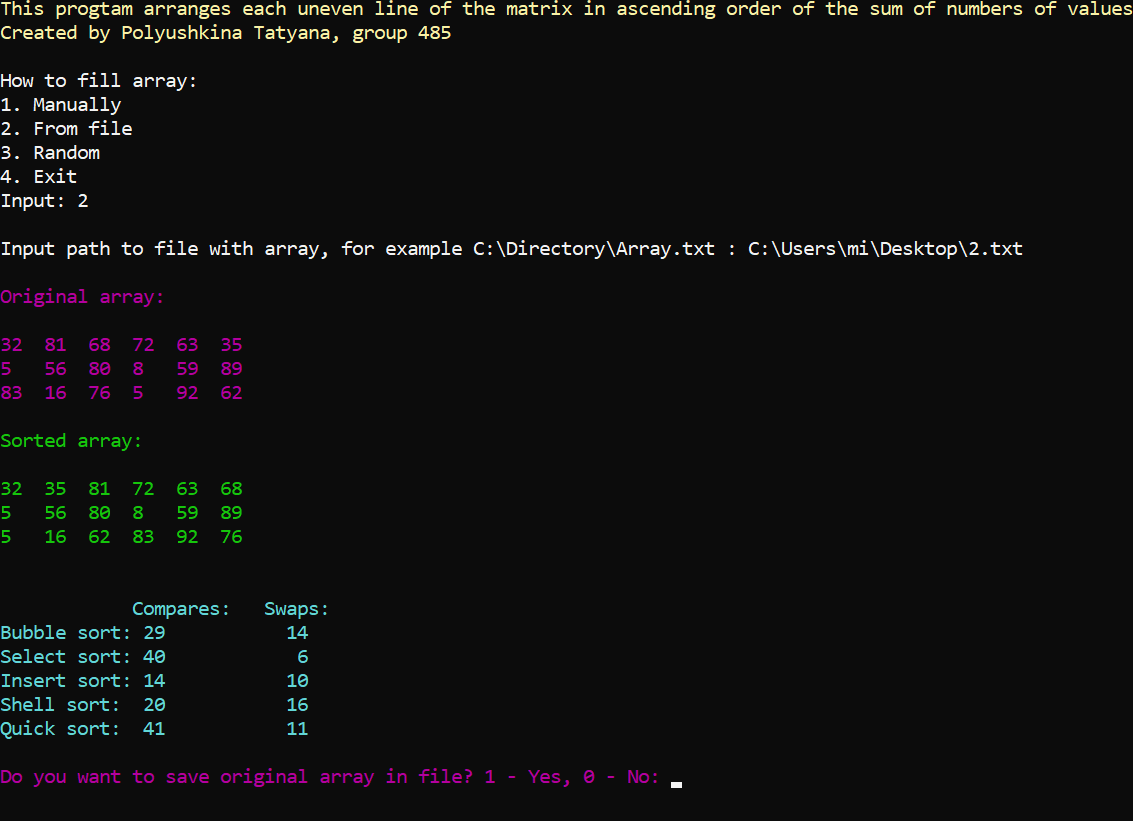


Рисунок 8 – Заполнение массива из файла

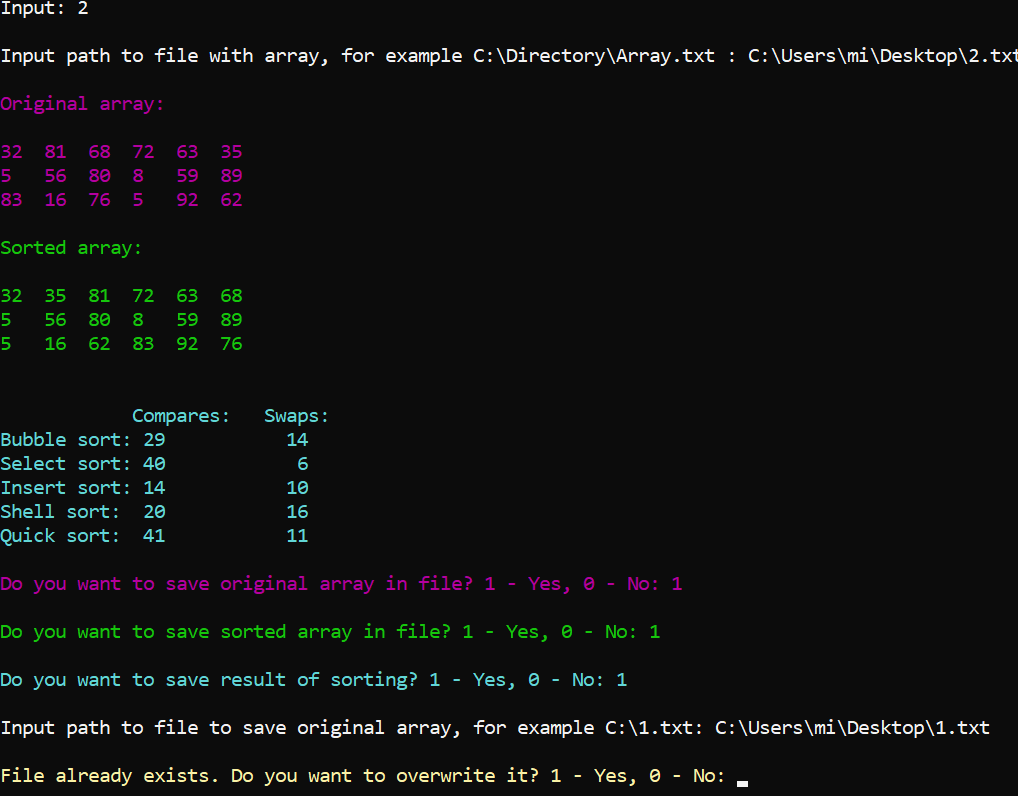


Рисунок 9 – Попытка сохранить в уже созданный файл

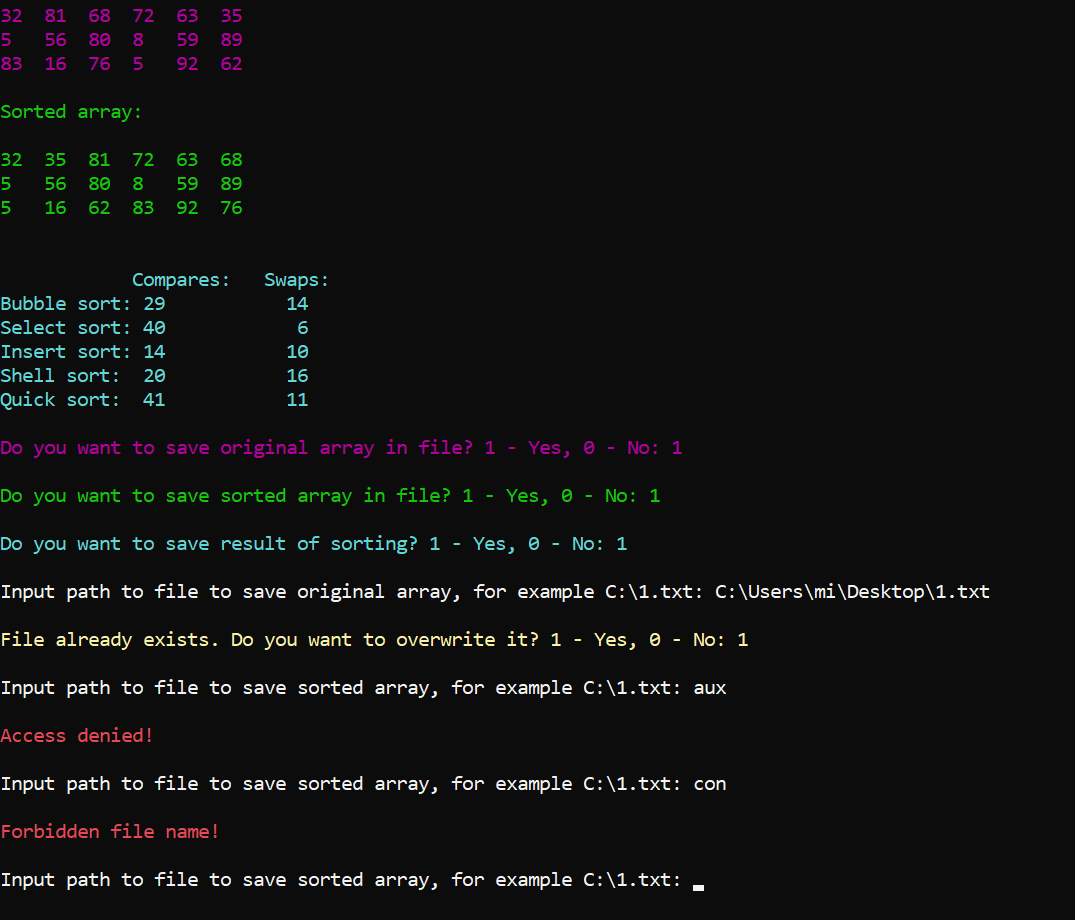


Рисунок 10 – Попытка сохранить в файл с запрещенным именем

# Текст программы

# [--- Начало программы ---]

**// ArrChange.h**

**// Лабораторная работа №2.**

**// Студент группы 485, Полюшкина Татьяна Юрьевна. 2019 год**

#pragma once

const int sortsNumber = 5;

const int parametersNumber = 2;

int\*\* ArrChange(int\*\* origArr, int n, int m, int comparesAndSwaps[sortsNumber][parametersNumber], int lenghtToSetw);

**// ArrInput.h**

**// Лабораторная работа №2.**

**// Студент группы 485, Полюшкина Татьяна Юрьевна. 2019 год**

#pragma once

void ManualInput();

void RandomInput();

void FileInput();

**// ArrOutput.h**

**// Лабораторная работа №2.**

**// Студент группы 485, Полюшкина Татьяна Юрьевна. 2019 год**

#pragma once

#include <string>

#include <Windows.h>

#include "ArrChange.h"

using namespace std;

enum Color { blue = 9, green, azure, red, purple, yellow, white };

void CoutWithColor(int color, string message);

void SetColor(int color);

void OutputInConsoleArr(int\*\* arr, int n, int m, int lenghtToSetw, string message, int color);

int NumberLenght(int number);

void OutputArrInFile(int\*\* arr, int n, int m);

void SaveResult(int comparesAndSwaps[sortsNumber][parametersNumber]);

**// VarCheck.h**

**// Лабораторная работа №2.**

**// Студент группы 485, Полюшкина Татьяна Юрьевна. 2019 год**

#pragma once

int GetInt();

bool GetBool();

**// Source.cpp**

**// Лабораторная работа №2.**

**// Студент группы 485, Полюшкина Татьяна Юрьевна. 2019 год**

#include <iostream>

#include <Windows.h>

#include "VarCheck.h"

#include "ArrOutput.h"

#include "ArrInput.h"

using namespace std;

enum MenuItems {

minMenuItem = 0, manual, file, random, close, maxMenuItem

};

bool Menu() { // Вывод меню и проверка введённых данных

cout << "How to fill array: " << endl << "1. Manually" << endl << "2. From file" << endl << "3. Random" << endl <<

"4. Exit" << endl << "Input: ";

int menuItem; // пункт меню для ввода пользователем

bool running = true; // Для возврата в main, проверка запущена ли программа

while (true) { // ввод пользователем элемента меню и его проверка

menuItem = GetInt(); // ввод элемента меню, проверка на int

if (menuItem > minMenuItem && menuItem < maxMenuItem) { // проверка на выход за пределы меню

break;

}

else CoutWithColor(red, "Try again: ");

}

switch (menuItem) // соотношение данных, если введеные пользователем данные совпадают с условием, то выполняются определенные действия.

{

case manual:

ManualInput();

running = true;

break;

case file:

FileInput();

running = true;

break;

case random:

RandomInput();

running = true;

break;

case close:

running = false;

break;

}

return running; // если running = true - программа продолжается, если false, то программа завершается

}

int main() { // работа программы

CoutWithColor(yellow, "This progtam arranges each uneven line of the matrix in ascending order of the sum of numbers of values.\nCreated by Polyushkina Tatyana, group 485\n\n");

bool running = true; // Для определения запущенности программы

while (running)

{

running = Menu();

}

return 0;

}

**// ArrChange.cpp**

**// Лабораторная работа №2.**

**// Студент группы 485, Полюшкина Татьяна Юрьевна. 2019 год**

#include <iostream>

#include "ArrOutput.h"

using namespace std;

enum { comparesColumn, swapsColumn, bubbleLine = 0, selectLine, insertLine, shellLine, quickLine };

enum { bubbleSort, selectSort, insertSort, shellSort, quickSort };

struct LinkValuesAndSum {

int value;

int sum;

bool listed = false;

};

void BubbleSort(int\* line, int m, int& compares, int& swaps) {

bool lineChanged = true;

for (int i = 0; i < m && lineChanged; ++i) {

lineChanged = false;

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

++compares;

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

int temp = line[j];

line[j] = line[j + 1];

line[j + 1] = temp;

lineChanged = true;

++swaps;

}

}

}

}

void SelectSort(int\* line, int m, int& compares, int& swaps) {

int min;

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

min = i;

for (int j = i + 1; j < m; ++j) {

++compares;

if (line[j] < line[min])

min = j;

}

if (i != min) {

int temp = line[i];

line[i] = line[min];

line[min] = temp;

++swaps;

}

}

}

void InsertSort(int\* line, int m, int& compares, int& swaps) {

for (int i = 1; i < m; ++i) {

int x = line[i];

int j = i;

while (true) {

if (j > 0) {

++compares;

if (x < line[j - 1]) {

line[j] = line[j - 1];

--j;

}

else break;

}

else break;

}

++swaps;

line[j] = x;

}

}

void ShellSort(int\* line, int m, int& compares, int& swaps) {

for (int step = m / 2; step > 0; step /= 2) {

for (int i = step; i < m; ++i) {

int x = line[i];

int j = i;

while (true) {

if (j >= step) {

++compares;

if (x < line[j - step]) {

line[j] = line[j - step];

j -= step;

}

else break;

}

else break;

}

++swaps;

line[j] = x;

}

}

}

void QuickSort(int\* line, int first, int last, int& compares, int& swaps) {

int left = first;

int pivot = line[(first + last) / 2];

int right = last;

while (left < right) {

while (++compares, line[left] < pivot) {

++left;

}

while (++compares, line[right] > pivot) {

--right;

}

if (left <= right) {

int temp = line[left];

line[left] = line[right];

line[right] = temp;

++left;

--right;

++swaps;

}

}

if (first < right) QuickSort(line, first, right, compares,swaps);

if (left < last) QuickSort(line, left, last, compares, swaps);

}

void ArrCopy(int\*\* origArr, int\*\* copy, int n, int m) {

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

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

copy[i][j] = origArr[i][j];

}

void SumCalcAndSort(int\* origLine, int\* resultLine, int m, int comparesAndSwaps[sortsNumber][parametersNumber], int sortNum) {

int\* sumLine = new int[m];

LinkValuesAndSum\* arrOfLinks = new LinkValuesAndSum[m];

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

int value = abs(origLine[i]);

int sum = 0;

while (value) {

sum += value % 10;

value /= 10;

}

sumLine[i] = sum;

arrOfLinks[i].sum = sum;

arrOfLinks[i].value = origLine[i];

}

switch (sortNum)

{

case bubbleSort:

BubbleSort(sumLine, m, comparesAndSwaps[bubbleLine][comparesColumn], comparesAndSwaps[bubbleLine][swapsColumn]);

break;

case selectSort:

SelectSort(sumLine, m, comparesAndSwaps[selectLine][comparesColumn], comparesAndSwaps[selectLine][swapsColumn]);

break;

case insertSort:

InsertSort(sumLine, m, comparesAndSwaps[insertLine][comparesColumn], comparesAndSwaps[insertLine][swapsColumn]);

break;

case shellSort:

ShellSort(sumLine, m, comparesAndSwaps[shellLine][comparesColumn], comparesAndSwaps[shellLine][swapsColumn]);

break;

case quickSort:

int first = 0;

int last = m - 1;

QuickSort(sumLine, first, last, comparesAndSwaps[quickLine][comparesColumn], comparesAndSwaps[quickLine][swapsColumn]);

break;

}

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

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

if (sumLine[i] == arrOfLinks[j].sum && !arrOfLinks[j].listed) {

resultLine[i] = arrOfLinks[j].value;

arrOfLinks[j].listed = true;

break;

}

}

}

delete[] sumLine;

delete[] arrOfLinks;

}

int\*\* ArrChange(int\*\* origArr, int n, int m, int comparesAndSwaps[sortsNumber][parametersNumber], int lenghtToSetw) {

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

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

comparesAndSwaps[i][j] = 0;

}

string table[sortsNumber] = { "Bubble sort: ", "Select sort: ", "Insert sort: ", "Shell sort: ", "Quick sort: " };

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

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

resultArr[i] = new int[m];

}

for (int sortIterator = 0; sortIterator < sortsNumber; ++sortIterator) {

ArrCopy(origArr, resultArr, n, m);

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

if ((i + 1) % 2) {

SumCalcAndSort(origArr[i], resultArr[i], m, comparesAndSwaps, sortIterator);

}

}

OutputInConsoleArr(resultArr, n, m, lenghtToSetw, table[sortIterator] + "\n\n", blue + sortIterator);

}

return resultArr;

}

**// ArrInput.cpp**

**// Лабораторная работа №2.**

**// Студент группы 485, Полюшкина Татьяна Юрьевна. 2019 год**

#include <iostream>

#include <ctime>

#include <string>

#include <fstream>

#include <filesystem>

#include "ArrOutput.h"

#include "VarCheck.h"

#include "ArrOutput.h"

#include "ArrChange.h"

using namespace std;

using namespace std::filesystem;

const int minArrSize = 1;

const int maxRandValue = 100;

const int indent = 2;

void EndingOutput(int\*\* origArr, int\*\* resultArr, int n, int m, int lenghtToSetw, int comparesAndSwaps[sortsNumber][parametersNumber]) {

string table[5] = { "Bubble sort: ","Select sort: ", "Insert sort: ", "Shell sort: ", "Quick sort: " };

SetColor(azure);

cout << "\n Compares: Swaps:\n";

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

cout << table[i];

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

cout << comparesAndSwaps[i][j] << setw(13);

}

cout << endl;

}

cout << endl;

SetColor(purple);

cout << "Do you want to save original array in file? 1 - Yes, 0 - No: ";

bool saveOrig = GetBool(); // запрос у пользователя, хочет ли он сохранить в файл исходный массив

SetColor(white);

if (saveOrig) {

OutputArrInFile(origArr, n, m);

CoutWithColor(green, "\nSaved successfully!\n\n");

}

else {

cout << endl;

}

SetColor(blue);

cout << "Do you want to save sorted array in file? 1 - Yes, 0 - No: ";

bool saveSorted = GetBool();

SetColor(white);

if (saveSorted) {

OutputArrInFile(resultArr, n, m);

CoutWithColor(green, "\nSaved successfully!\n\n");

}

else {

cout << endl;

}

SetColor(azure);

cout << "Do you want to save result of sorting? 1 - Yes, 0 - No: ";

bool saveResult = GetBool();

SetColor(white);

if (saveResult){

SaveResult(comparesAndSwaps);

CoutWithColor(green, "\nSaved successfully!\n\n");

}

else

cout << endl;

for (int i = 0; i < n; ++i) { // освобождение памяти, выделенной под столбцы двумерного массива

delete[] origArr[i];

delete[] resultArr[i];

}

delete[] origArr; // освобождение памяти, выделенной под строки двумерного массива

delete[] resultArr;

}

int InputN() { //ввод количества строк

cout << "\nInput number of lines: ";

int n;

while (true) { // ввод пользователем элемента n и его проверка

n = GetInt();

if (n > minArrSize) {

break;

}

else

CoutWithColor(red, "\nTry again: ");

}

return n;

}

int InputM() { //ввод количества строк

cout << "\nInput number of column: ";

int m;

while (true) { // ввод пользователем элемента n и его проверка

m = GetInt();

if (m > minArrSize) {

break;

}

else

CoutWithColor(red, "\nTry again: ");

}

return m;

}

void ManualInput() { // ввод вручную

int n; // количество элементов на ввод пользователю

int m;

int ArrSize;

int maxNumberForSetw = 0; // Максимальное число, для вычисления длины отступа

bool userAgreed = false; // Хочет ли пользователь ввести количество элементов

while (!userAgreed) { // пока пользователь не согласен ввести количество элементов для расчёта ArrSize

n = InputN(); // ввод n

m = InputM(); //ввод m

ArrSize = (n \* m); // вычисление ArrSize

string message = "\nYou will have to input ";

message += to\_string(ArrSize);

message += " elements. Do you want to continue? 1- Yes, 0 - No: ";

CoutWithColor(yellow, message);

userAgreed = GetBool();

}

int\*\* origArr = new int\* [n]; // создание двумерного массива размерностью n

for (int i = 0; i < n; ++i) { // выделение памяти под столбцы исходного массива

origArr[i] = new int[m];

}

for (int i = 0; i < n; ++i) { // присваевание элементам массива введенные пользователем значения

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

cout << endl << "Arr[" << i + 1 << "]["<< j + 1 << "] = ";

origArr[i][j] = GetInt();

if (abs(origArr[i][j]) > maxNumberForSetw) {

maxNumberForSetw = abs(origArr[i][j]);

}

}

}

int comparesAndSwaps[sortsNumber][parametersNumber];

int lenghtToSetw = NumberLenght(maxNumberForSetw) + indent;

OutputInConsoleArr(origArr, n, m, lenghtToSetw, "\nOriginal array:\n\n", purple);

int\*\* resultArr = ArrChange(origArr, n, m, comparesAndSwaps, lenghtToSetw);

EndingOutput(origArr,resultArr, n, m, lenghtToSetw, comparesAndSwaps);

}

void RandomInput() { // генерация случайных чисел для массива

srand(static\_cast<unsigned int>(time(nullptr))); // для установки начала последовательности генерируемой rand()

int n = InputN();// ввод n

int m = InputM(); // ввод m

int maxNumberForSetw = 0; // Максимальное число, для вычисления длины отступа

int\*\* origArr = new int\* [n]; // создание двумерного массива

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

origArr[i] = new int[m];

}

for (int i = 0; i < n; ++i) { // присваевание элементам массива сгенерированные значения

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

origArr[i][j] = rand() % maxRandValue;

if (abs(origArr[i][j]) > maxNumberForSetw) {

maxNumberForSetw = abs(origArr[i][j]);

}

}

}

int comparesAndSwaps[sortsNumber][parametersNumber];

int lenghtToSetw = NumberLenght(maxNumberForSetw) + indent;

OutputInConsoleArr(origArr, n, m, lenghtToSetw, "\nOriginal array:\n\n", purple);

int\*\* resultArr = ArrChange(origArr, n, m, comparesAndSwaps, lenghtToSetw);

EndingOutput(origArr, resultArr, n, m, lenghtToSetw, comparesAndSwaps);

}

string FilePathCheckReturnForInput() { // проверка пути для чтения из файла

string filePath;

bool isPathValid = false;

while (!isPathValid) { // while isPathValid == false, проверка валидности введенного пути

cout << "\nInput path to file with array, for example C:\\Directory\\Array.txt : ";

cin >> filePath;

cin.ignore(INT\_MAX, '\n');

if (!ifstream(filePath)) { // существует ли файл

CoutWithColor(red, "\nFile does not exist!\n");

continue;

}

try {

is\_regular\_file(filePath);

}

catch (...) {

CoutWithColor(red, "\nForbidden file name!\n");

continue;

}

ifstream file(filePath);

if (!file) { //проверка доступа к файлу

CoutWithColor(red, "\nAccess denied!\n");

continue;

}

isPathValid = true;

}

return filePath;

}

void FileInput() { //считывание данных из файла

bool valCorrect = false; // проверка на соответствие int и выхода из цикла while, если все значения корректы

int n;

int m;

int maxNumberForSetw = 0;

while (!valCorrect) {

string filePath = FilePathCheckReturnForInput();

ifstream file(filePath);

if (!(file >> n) || n <= minArrSize || !(file >> m)|| m <= minArrSize) { //проверка размера массива

CoutWithColor(red, "\nIncorrect array size.\n");

continue;

}

int\*\* origArr = new int\* [n]; // создание двумерного массива

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

origArr[i] = new int[m];

}

bool toContinue = false; // для перехода в начало цикла while, так как continue продолжит цикл for

bool toBreak = false;

for (int i = 0; i < n && !toBreak; i++) { // присваевание элементам массива сгенерированные значения

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

if (!(file >> origArr[i][j])) {

CoutWithColor(red, "\nIncorrect array value.\n");

toContinue = true;

toBreak = true;

break;

}

if (abs(origArr[i][j]) > maxNumberForSetw) {

maxNumberForSetw = abs(origArr[i][j]);

}

}

}

if (toContinue) {

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

delete[] origArr[i];

delete[] origArr;

continue;

}

valCorrect = true;

int comparesAndSwaps[sortsNumber][parametersNumber];

int lenghtToSetw = NumberLenght(maxNumberForSetw) + indent;

OutputInConsoleArr(origArr, n, m, lenghtToSetw, "\nOriginal array:\n\n", purple);

int\*\* resultArr = ArrChange(origArr, n, m, comparesAndSwaps, lenghtToSetw);

EndingOutput(origArr, resultArr, n, m, lenghtToSetw, comparesAndSwaps);

}

}

**// ArrOutput.cpp**

**// Лабораторная работа №2.**

**// Студент группы 485, Полюшкина Татьяна Юрьевна. 2019 год**

#include <iostream>

#include <fstream>

#include <Windows.h>

#include <string>

#include <filesystem>

#include "VarCheck.h"

#include "ArrOutput.h"

#include "ArrChange.h"

using namespace std;

using namespace std::filesystem;

void SetColor(int color) { // изменение цвета в консоли

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), color);

}

void CoutWithColor(int color, string message) { // вывод сообщения message с цветом color

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), color); // изменение цвета на color

cout << message; // вывод сообщения

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), white); // изменение цвета на белый

}

int NumberLenght(int number) { // вычисление длины числа

int lenght = 0;

while (number != 0) {

++lenght;

number /= 10;

}

return lenght;

}

void OutputInConsoleArr(int\*\* arr, int n, int m, int lenghtToSetw, string message, int color) {

SetColor(color);

cout << message;

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

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

cout << arr[i][j] << setw(lenghtToSetw);

}

cout << endl << left;

}

cout << endl << resetiosflags(ios::adjustfield);

SetColor(white);

}

string FilePathCheckReturnForOutput(string message) { // проверка пути для сохранения в файл

string filePath;

bool isPathValid = false;

while (!isPathValid) { // while isPathValid == false, проверка валидности введенного пути

cout << "\n" << message;

cin >> filePath;

cin.ignore(INT\_MAX, '\n');

if (ifstream(filePath)) { // существует ли файл

try {

if (is\_regular\_file(filePath)) { // проверка на запрещенные имена (aux, con..)

SetColor(yellow);

cout << "\nFile already exists. Do you want to overwrite it? 1 - Yes, 0 - No: ";

bool toOverwrite = GetBool();

SetColor(white);

if (!toOverwrite) {

continue;

}

}

}

catch (...) {

CoutWithColor(red, "\nForbidden file name!\n");

continue;

}

}

ofstream file(filePath); // создание файла по заданному пути

if (!file) { // проверка на доступ к созданию файла

CoutWithColor(red, "\nAccess denied!\n");

continue;

}

isPathValid = true;

}

return filePath; // возврат корректного пути

}

void OutputArrInFile(int\*\* arr, int n, int m) {

ofstream file(FilePathCheckReturnForOutput("Input path to file to save original array, for example C:\\1.txt: "));

file << n << endl << m << endl;

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

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

file << arr[i][j] << endl;

}

}

}

void SaveResult(int comparesAndSwaps[sortsNumber][parametersNumber]) {

ofstream file(FilePathCheckReturnForOutput("Input path to file to save the results of sorts working, for example C:\\1.txt: "));

string table[5] = { "Bubble sort: ","Select sort: ", "Insert sort: ", "Shell sort: ", "Quick sort: " };

file << " Compares: Swaps:\n";

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

file << table[i];

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

file << comparesAndSwaps[i][j] << setw(13);

}

file << endl;

}

file << endl;

}

**// VarCheck.cpp**

**// Лабораторная работа №2.**

**// Студент группы 485, Полюшкина Татьяна Юрьевна. 2019 год**

#include <iostream>

#include "ArrOutput.h"

using namespace std;

template <typename T> // создание шаблона T

T InputCheck() { // проверка ввода пользователя

T userInput; // создание переменной заданного типа для записи ввода пользователя

cin >> userInput;

while (cin.fail()) { // пока введенные данные не соответствуют типу

CoutWithColor(red, "\nTry again: ");

cin.clear(); // очищает cin.fail()

cin.ignore(INT\_MAX, '\n'); // очистка буфера

cin >> userInput;

}

cin.ignore(INT\_MAX, '\n');

return userInput;

}

int GetInt() { // ввод пользователем, проверка и возврат значения int

return InputCheck<int>();

}

bool GetBool() { // ввод пользователем, проверка и возврат значения bool

return InputCheck<bool>();

}

**[--- Конец программы ---]**