

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:**

Зобнин Илья Михайлович

**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 (один раз) и упорядоченные (для каждого из методов) массивы данных. Составить сравнительную таблицу эффективности методов, в которой необходимо указать число сравнений и перестановок переменных в каждом методе сортировки. Упорядочить диагональные элементы матрицы по возрастанию.

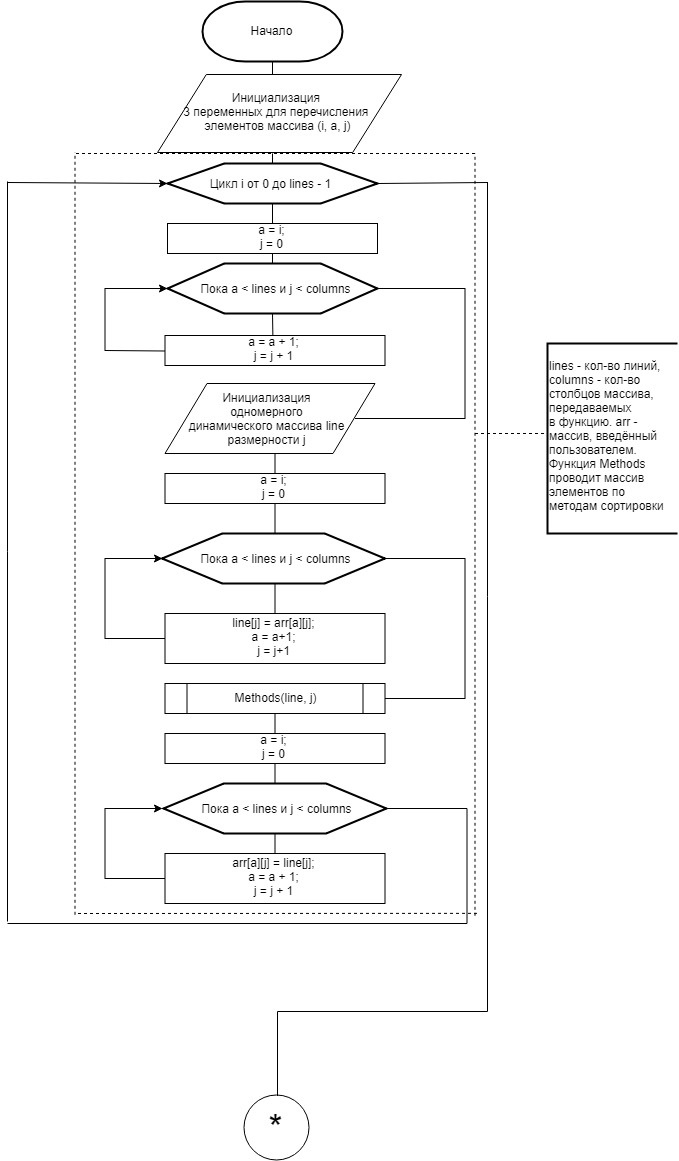
# Исходные данные

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 2-х, 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



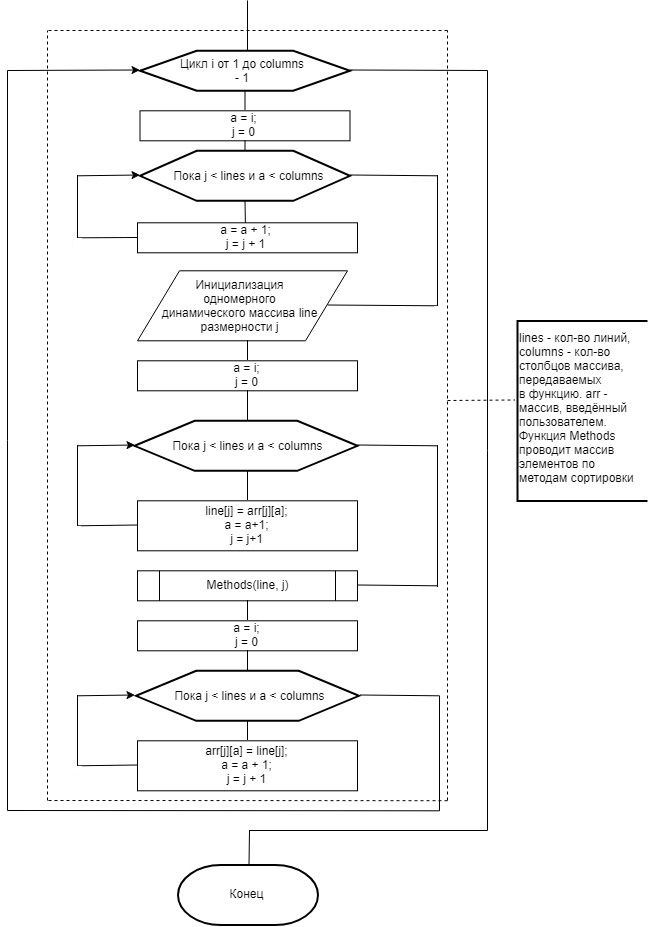


Рисунок 1 - блок-схема перевода диагонали в строку для более удобного выполнения задания



Рисунок 2 – блок-схема функции Methods, проводящей строку по всем методам сортировки

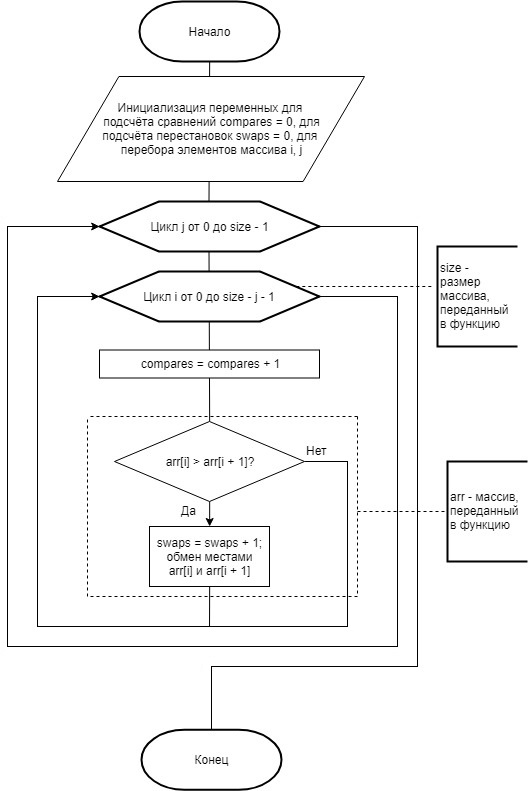
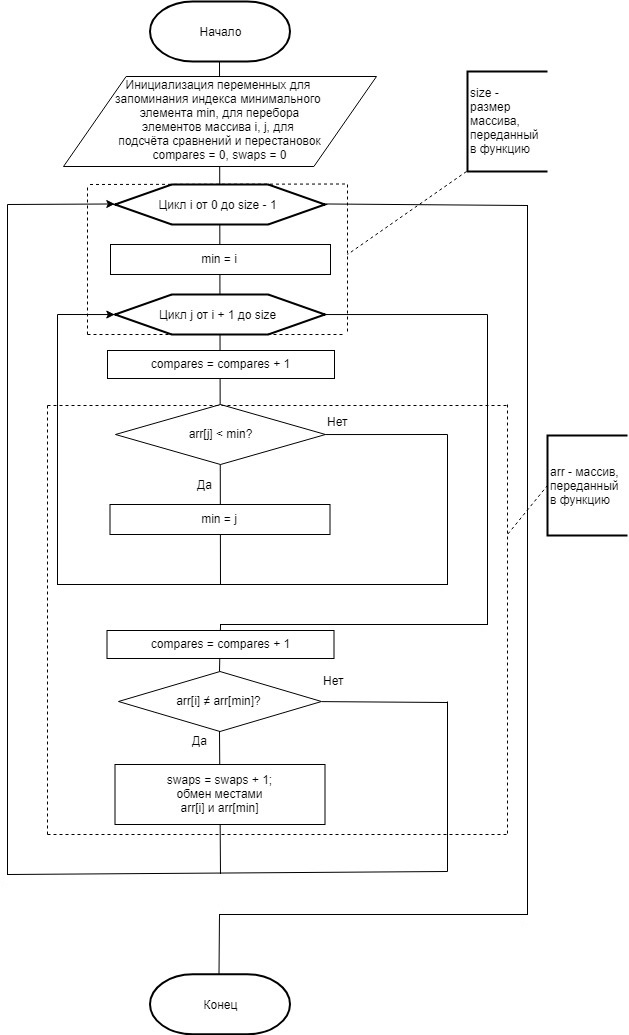


Рисунок 3 – блок-схема сортировки пузырьками

 – Рисунок 4 - блок-схема сортировки выборкой

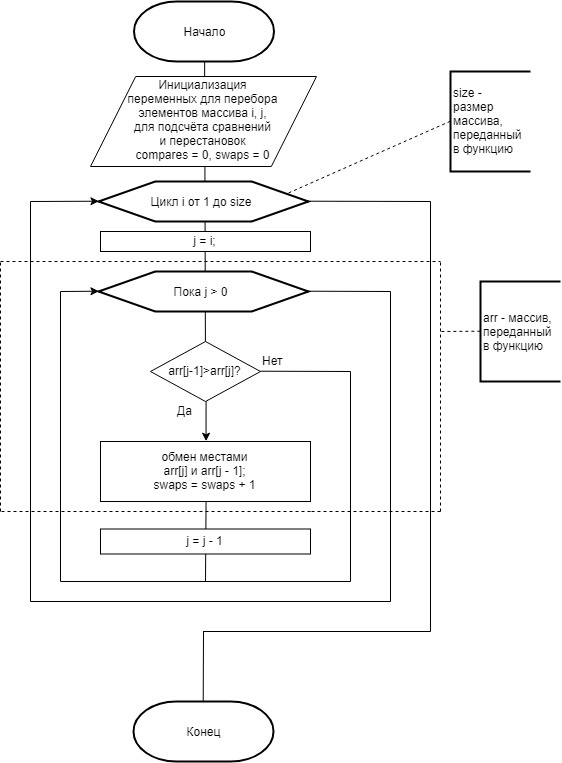


Рисунок 5 – блок-схема сортировки вставками

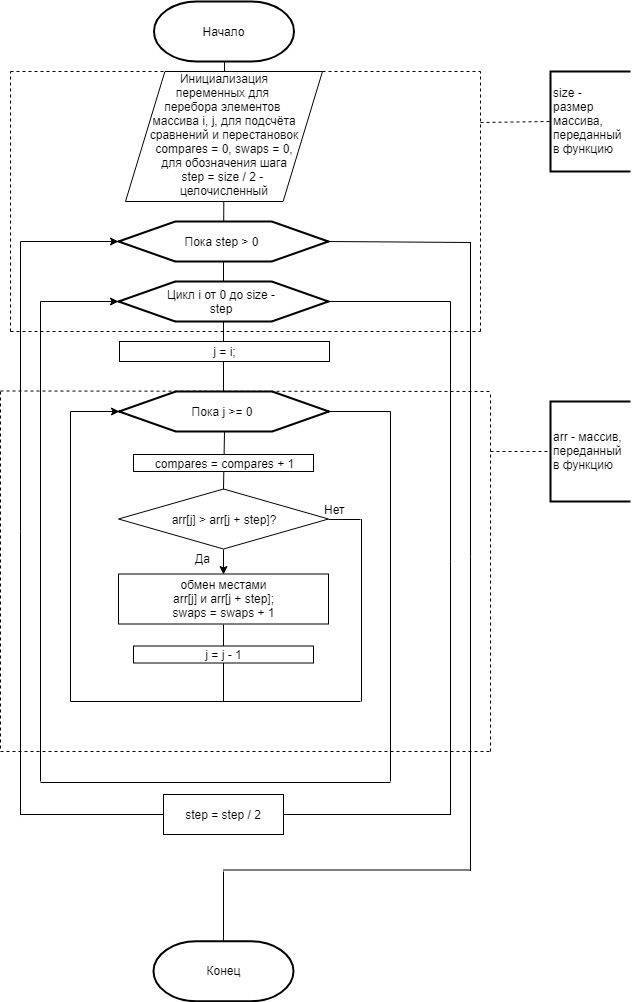
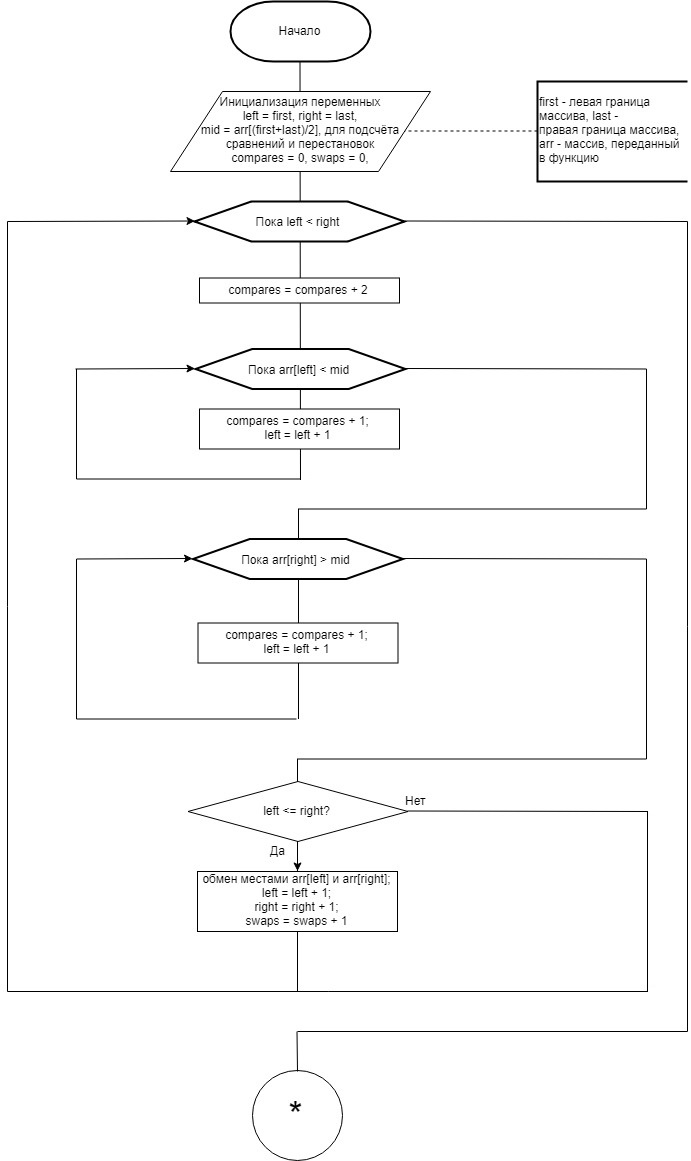


Рисунок 6 – блок-схема сортировки Шелла



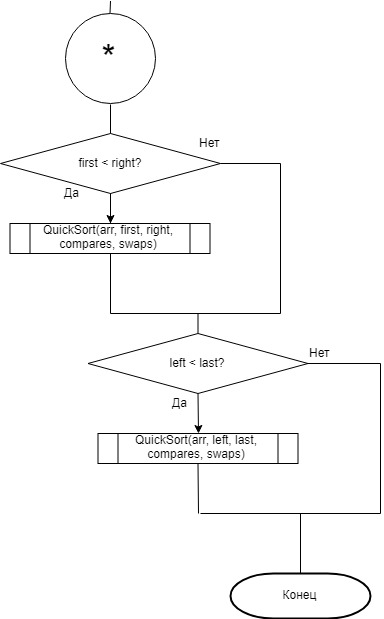


Рисунок 7 – блок-схема быстрой сортировки

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** |
| lines | unsigned int | Количество строк массива |
| columns | unsigned int | Количество столбцов массива |
| running | bool | Определяет, запущена ли программа |
| menu | unsigned int | Для ввода пункта меню |
| i | int | Для цикла чередования строк |
| j | int | Для цикла чередования столбцов |
| filePath | string | Для ввода пути к файлу |
| isDataLoaded | bool | Для проверки, загрузились ли данные из файла |
| myFile | ifstream | Для считывания данных из файла |
| arr | int\*\* | Массив данных, вводимый пользователем |
| choice | bool | Выбор пользователя сохранить данные в файл или нет |
| compares | int | Количество сравнений |
| swaps | int | Количество перестановок |
| line | int\* | Линия из диагонали |
| 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 |
| arrCopy | int\* | Для копирования диагонали |
| a | int | Для чередования эл-ов диагонали |
| right | int | Правая граница в быстрой сортировке |
| left | int | Левая граница в быстрой сортировке |
| pivot | int | Разрешающий элемент |

|  |  |  |
| --- | --- | --- |
| Продолжение Таблицы 1 | | |
| **Имя** | **Тип** | **Описание** |
| toBreakAndContinue | bool | Для выхода из цикла в случае неверно прочитанных данных из файла |
| leftHold | int | Для удержания индекса левого элемента |
| rightHold | int | Для удержания индекса правого элемента |
| first | int | Для обозначения изначальной левой границы |
| last | int | Для обозначения изначальной правой границы |

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\_MIN | const int | -2147483647 | Минимальное целое число |
| INT\_MAX | const int | 2147483647 | Максимальное целое число |
| maxMenuNumber | const int | 3 | Максимальный номер пункта меню |
| minSize | const int | 2 | Минимальное значение строки/столбца |
| name | const string | 10 | Массив строк из названий элементов для составления таблицы |

# Структура программы

В силу большого количества функций программа разделена на семь исполняемых модулей, из которых один является основным и отвечает за запуск программы, пять оставшихся содержат в себе функции, необходимые для работы программы. Последний файл отвечает за предкомпиляцию и сокращает время сборки программы с 10,3 до 1,3-х секунд.

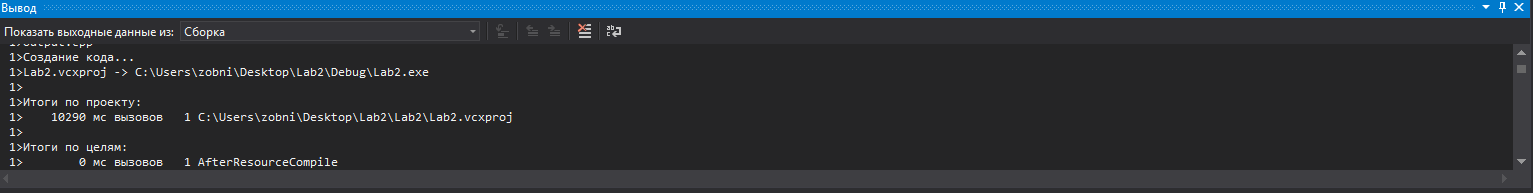


Рисунок 8 – время сборки программы с выключенным предкомпилированным заголовком

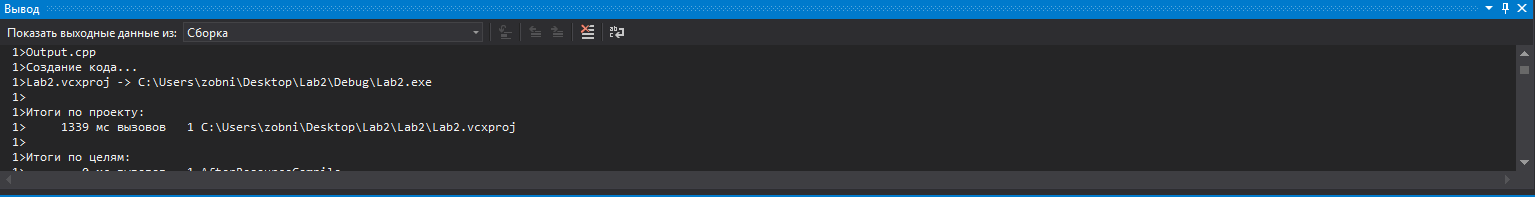


Рисунок 9 – время сборки программы с включённым предкомпилированным заголовком

Модуль Lab2:

Таблица 3 – Функции, составляющие модуль Lab2

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| main | Начало программы |

Модуль Menu:

Таблица 4 – Функции, составляющие модуль Menu

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| MainMenu | Вывод главного меню, выбор пункта меню |

Модуль ArrFilling:

Таблица 5 – Функции, составляющие модуль ArrFilling

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| FileInput | Ввод элементов из файла в массив |
| ManualInput | Ввод элементов массива из консоли вручную |
| RandomFilling | Заполнение массива случайными элементами |

Модуль ArrChange:

Таблица 6 – Функции, составляющие модуль ArrChange

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| ArrCopy | 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 |
| Methods | Функция, отвечающая за проход диагональю через все методы сортировки |
| ArrChange | Функция, отвечающая за преобразование диагонали в строку и прохождение через методы сортировки |

Модуль Output:

Таблица 7 – Функции, составляющие модуль Output

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| SetColor | Изменение цвета текста, выводимого в консоль |
| OutputArrInFile | Вывод исходного массива в файл |
| OutputResultInFile | Вывод результата в файл |
| OutputDataInFile | Проверка на корректность введенного пути файла |
| OutputInConsole | Вывод массива в консоль |

Модуль InputAndCheck:

Таблица 8 – Функции, составляющие модуль InputAndCheck

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

# Описание хода выполнения лабораторной работы

* В ходе лабораторной работы было создано решение (Lab2) в интегрированной среде разработки Microsoft Visual Studio C++ 2017. В нём был создан проект.
* При выполнении самого задания было принято решение напрямую работать с диагоналями массива. Однако выяснилось, что это значительно труднее, чем работа с обычным одномерным массивом данных. Поэтому было решено преобразовывать каждую диагональ в одномерный массив для её последующей более простой обработки во всех методах.
* В созданном проекте нужно было включить все библиотеки, использованные в программе, в предкомпилированный заголовок pch.h для её более быстрой сборки.
* 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

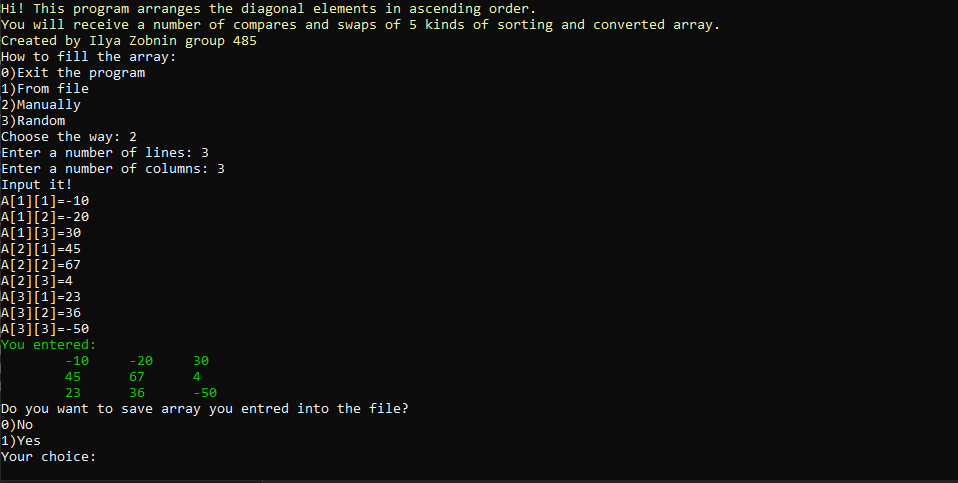
В результате работы программа выводит два массива различных цветов, первый массив является исходным, а второй — результатом работы программы, а также таблица с методами сортировки и их значениями сравнений и перестановок

Рисунок 10 – Заполнение массива с клавиатуры

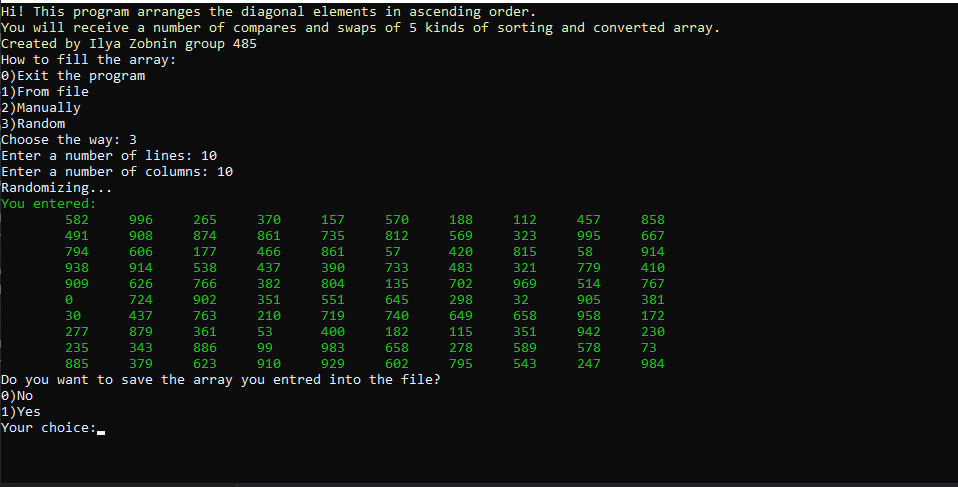


Рисунок 11 – Заполнение массива случайными числами

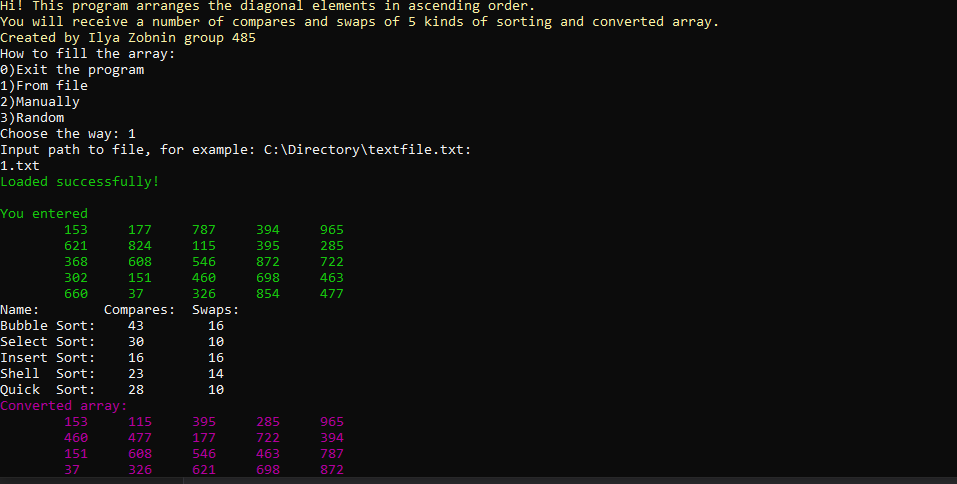


Рисунок 12 – Заполнение массива из файла

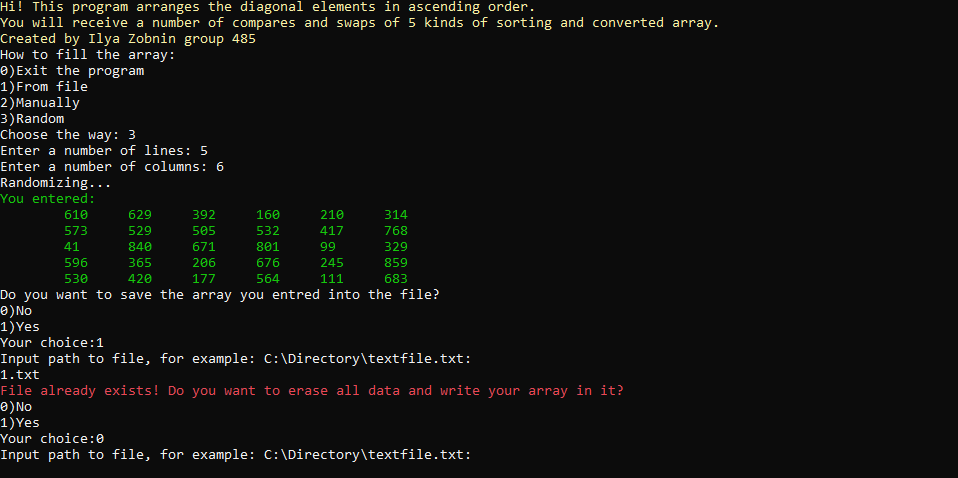


Рисунок 13 – Попытка сохранить в уже созданный файл

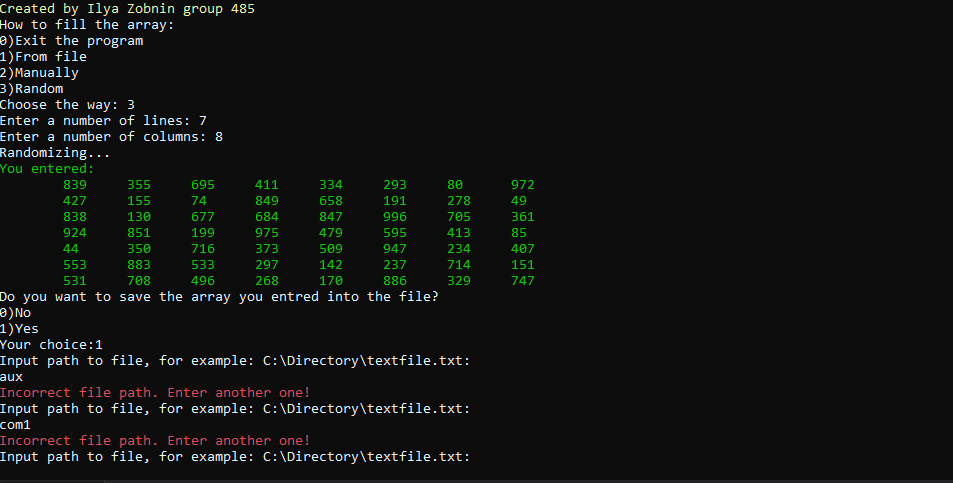


Рисунок 14 – Попытка сохранить в файл с запрещенным именем

# Текст программы

# [--- Начало программы ---]

**// pch.cpp**

**// Лабораторная работа №2.**

**// Студент группы 485, Зобнин Илья Михайлович. 2019 год**

#include "pch.h"

**// pch.h**

**// Лабораторная работа №2.**

**// Студент группы 485, Зобнин Илья Михайлович. 2019 год**

#pragma once

#include <iostream>

#include <Windows.h>

#include <filesystem>

#include <fstream>

#include <string>

#include <iomanip>

**// ArrFilling.h**

**// Лабораторная работа №2.**

**// Студент группы 485, Зобнин Илья Михайлович. 2019 год**

#pragma once

void FileInput();

void RandomFilling();

void ManualInput();

**// Menu.h**

**// Лабораторная работа №2.**

**// Студент группы 485, Зобнин Илья Михайлович. 2019 год**

#pragma once

bool MainMenu();

**// ArrChange.h**

**// Лабораторная работа №2.**

**// Студент группы 485, Зобнин Илья Михайлович. 2019 год**

#pragma once

void ArrChange(int \*\*arr, int lines, int columns);

**// Output.h**

**// Лабораторная работа №2.**

**// Студент группы 485, Зобнин Илья Михайлович. 2019 год**

#pragma once

void OutputInConsole(int \*\*arr, int lines, int columns);

void SetColor(int color);

void OutputOrigArr(int \*\*arr, int lines, int columns);

void OutputResult(int \*\*arr, int lines, int columns, int sourceArrSum, int changedArrSum);

**// InputAndCheck.h**

**// Лабораторная работа №2.**

**// Студент группы 485, Зобнин Илья Михайлович. 2019 год**

#pragma once

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

int GetInt();

int GetUnsignedInt();

bool GetBool();

**// Lab2.cpp**

**// Лабораторная работа №2.**

**// Студент группы 485, Зобнин Илья Михайлович. 2019 год**

#include "pch.h"

#include "Output.h"

#include "InputAndCheck.h"

#include "Menu.h"

using namespace std;

int main() {

bool running = 1;

SetColor(yellow);

cout << "Hi! This program arranges the diagonal elements in ascending order." << endl <<

"You will receive a number of compares and swaps of 5 kinds of sorting and converted array." << endl << "Created by Ilya Zobnin group 485" << endl;

SetColor(white);

while (running)

running = MainMenu();//переход к функции, демонстрирующей меню

return 0;

}

**// Menu.cpp**

**// Лабораторная работа №2.**

**// Студент группы 485, Зобнин Илья Михайлович. 2019 год**

#include "pch.h"

#include "InputAndCheck.h"

#include "ArrFilling.h"

using namespace std;

enum { close, fromFile, manually, random };

bool MainMenu() {//функция, требующая от пользователя выбрать метод заполнения массива

bool running = true;

cout << "How to fill the array: " << endl << "0)Exit the program" << endl << "1)From file" << endl << "2)Manually" << endl << "3)Random" << endl;

cout << "Choose the way: ";

int menu;

const int maxMenuNumber = 3;

menu = GetUnsignedInt();

if (menu > maxMenuNumber) {//ограничение по максимальному элементу меню

cout << "Try again: " << endl;

MainMenu();

}

else

switch (menu) {//переход к соответствующей функции

case fromFile: {

FileInput();

running = true;

break;

}

case manually: {

ManualInput();

running = true;

break;

}

case random: {

RandomFilling();

running = true;

break;

}

case close: {

running = close;

break;

}

}

return running;

}

**// ArrFilling.cpp**

**// Лабораторная работа №2.**

**// Студент группы 485, Зобнин Илья Михайлович. 2019 год**

#include "pch.h"

#include "InputAndCheck.h"

#include "Output.h"

#include "ArrChange.h"

using namespace std;

using namespace experimental::filesystem;

const int minSize = 2;

void FileInput() {//заполнение массива из файла

string filePath;

bool isDataLoaded = false;

int lines, columns;

do {

cout << "Input path to file, for example: C:\\Directory\\textfile.txt: " << endl;

cin >> filePath;

if (!ifstream(filePath)) {//если файла не существует

SetColor(red);

cout << "File does not exist! Input another path!" << endl;

SetColor(white);

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

continue;

}

if (!is\_regular\_file(filePath)) {//проверка на валидность имени (защита от aux, com и т.д.)

SetColor(red);

cout << "Incorrect file path. Enter another one!" << endl;

SetColor(white);

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

continue;

}

ifstream myFile(filePath);

if (!myFile) {//если нет доступа к файлу

SetColor(red);

cout << "Access denied! Enter another path!" << endl;

SetColor(white);

myFile.close();

continue;

}

if (!(myFile >> lines) || lines < minSize || !(myFile >> columns) || columns < minSize) {//считывание кол-ва строк и столбцов и проверка

SetColor(red);//на валидность введённых данных

cout << "Incorrect array size!" << endl;

SetColor(white);

myFile.close();

continue;

}

int \*\*arr = new int\*[lines];//если все проверки пройдены успешно, создаётся двумерный массив

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

arr[i] = new int[columns];

int count = 0, i, j;

bool toBreakAndContinue = false;//для выхода из двух циклов в случае ввода невалидного значения и

//перехода в начало программы

for (i = 0; i < lines; i++)//считывание элементов массива и проверка их на валидность

if (!toBreakAndContinue)

for (j = 0; j < columns; j++)

if (!toBreakAndContinue) {

if (!(myFile >> arr[i][j])) {//если введено не int значение, выход из циклов и переход в начало программы

SetColor(red);

cout << "Invalid value type or not enough elements in file. Edit the file and try again! " << endl;

SetColor(white);

myFile.close();

toBreakAndContinue = true;

for (i = 0; i < lines; i++)//удаление созданного массива

delete[] arr[i];

delete[] arr;

}

count++;

}

else

break;

else

break;

if (toBreakAndContinue)

continue;

SetColor(green);

cout << "Loaded successfully!" << endl << endl;

cout << "You entered";

OutputInConsole(arr, lines, columns);

SetColor(white);

isDataLoaded = true;

myFile.close();

ArrChange(arr, lines, columns);//выполнение задания по изменению массива

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

delete[] arr[i];

delete[] arr;

} while (!isDataLoaded);

}

void RandomFilling() {//случайное заполнение

srand(static\_cast <unsigned int> (time(nullptr)));//для генерации случайных чисел при каждом запуске

int lines, columns;

bool choice;

cout << "Enter a number of lines: ";

lines = GetUnsignedInt();

cout << "Enter a number of columns: ";

columns = GetUnsignedInt();//запрос кол-ва строк и стобцов от пользователя

if (lines >= minSize && columns >= minSize) {

int \*\*arr = new int\*[lines];//создаётся двумерный массив

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

arr[i] = new int[columns];

cout << "Randomizing..." << endl;

for (int i = 0; i < lines; i++)//заполнение случайными значениями

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

arr[i][j] = rand() % 1001;

SetColor(green);

cout << "You entered: ";

OutputInConsole(arr, lines, columns);//вывод созданного массива в консоль

SetColor(white);

cout << endl << "Do you want to save the array you entred into the file?" << endl

<< "0)No" << endl

<< "1)Yes" << endl

<< "Your choice:";

choice = GetBool();//запрос от пользователя, хочет ли он сохранить полученный массив в файл

if (choice) {

bool origOrResult = true;

OutputDataInFile(arr, lines, columns, origOrResult);

}

ArrChange(arr, lines, columns);//выполнение задания по изменению массива

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

delete[] arr[i];

delete[] arr;

}

else {

SetColor(red);

cout << "Invalid lines and columns values, try again!" << endl;

SetColor(white);

RandomFilling();

}

}

void ManualInput() {//ввод всех данных вручную

int lines, columns;

bool choice;

cout << "Enter a number of lines: ";

lines = GetUnsignedInt();

cout << "Enter a number of columns: ";

columns = GetUnsignedInt();

if (lines >= minSize && columns >= minSize) {

int \*\*arr = new int\*[lines];//создание двумерного массива

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

arr[i] = new int[columns];

cout << "Input it!" << endl;

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

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

cout << "A[" << i + 1 << "]"

<< "[" << j + 1 << "]=";

arr[i][j] = GetInt();//ввод значений вручную

}

SetColor(green);

cout << "You entered: ";

OutputInConsole(arr, lines, columns);//вывод изначального массива в консоль

SetColor(white);

cout << endl << "Do you want to save array you entred into the file?" << endl

<< "0)No" << endl

<< "1)Yes" << endl

<< "Your choice:";

choice = GetBool();//запрос от пользователя, хочет ли он сохранить полученный массив в файл

if (choice) {

bool origOrResult = true;

OutputDataInFile(arr, lines, columns, origOrResult);

}

ArrChange(arr, lines, columns);//выполнение задания по изменению массива и вывода сумм

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

delete[] arr[i];

delete[] arr;

}

else {

SetColor(red);

cout << "Invalid lines and columns values, try again!" << endl;

SetColor(white);

ManualInput();

}

}

**// ArrChange.cpp**

**// Лабораторная работа №2.**

**// Студент группы 485, Зобнин Илья Михайлович. 2019 год**

#include "pch.h"

#include "InputAndCheck.h"

#include "Output.h"

using namespace std;

void ArrCopy(int \*arr, int \*arrCopy, int columns) {//копирование одного массива в другой

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

arrCopy[i] = arr[i];

}

void BubbleSort(int \*arr, int size, int &compares, int &swaps) {

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

for (int i = 0; i < size - j - 1; i++) {//перебор массива

compares++;

if (arr[i] > arr[i + 1]) {//если i-ый элемент больше следующего

swaps++;

int tmp = arr[i];

arr[i] = arr[i + 1];//обмен местами

arr[i + 1] = tmp;

}

}

}

}

void SelectingSort(int \*arr, int size, int &compares, int &swaps) {

int i, j, min, tmp;

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

min = i;//минимальный элемент по умолчанию i-ый

for (j = i + 1; j < size; j++) {//нахождение индекса минимального элемента

compares++;

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

min = j;

}

compares++;

if (arr[i] != arr[min]) {//если найден другой минимальный элемент

swaps++;

tmp = arr[i];

arr[i] = arr[min];//обмен местами

arr[min] = tmp;

}

}

}

void InsertSort(int \*arr, int size, int &compares, int &swaps){

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

int j = i;

while (j) {//пока находится элемент больше до i-го

compares++;

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

swaps++;

int tmp = arr[j - 1];

arr[j - 1] = arr[j];

arr[j] = tmp;

}

else

break;

j--;

}

}

}

void ShellSort(int \*arr, int size, int &compares, int &swaps) {

int step = size / 2;

while (step) {

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

int j = i;

while (j >= 0) {//сравнение j - го и j + step эл-та

compares++;

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

swaps++;

int tmp = arr[j];

arr[j] = arr[j + step];

arr[j + step] = tmp;

}

else

break;

j--;

}

}

step /= 2; //деление шага на два

}

}

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

{

int mid;

int left = first, right = last;

mid = arr[(left + right) / 2]; //вычисление опорного элемента

do

{

compares += 2;

while (arr[left] < mid) {

compares++;

left++;

}

while (arr[right] > mid) {

compares++;

right--;

}

if (left <= right) //перестановка элементов

{

swaps++;

int tmp = arr[left];

arr[left] = arr[right];

arr[right] = tmp;

left++;

right--;

}

} while (left < right);

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

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

}

void Methods(int \*arr, int size, int \*comparesAndSwaps) {//одномерный массив проходит по всем методам сортировки

int \*arrCopy = new int[size];//создание нового массива

ArrCopy(arr, arrCopy, size);//копирование переданного в функцию массива в только что созданный

BubbleSort(arrCopy, size, comparesAndSwaps[0], comparesAndSwaps[1]);//сам метод, после снова копирование массива

ArrCopy(arr, arrCopy, size);

SelectingSort(arrCopy, size, comparesAndSwaps[2], comparesAndSwaps[3]);

ArrCopy(arr, arrCopy, size);

InsertSort(arrCopy, size, comparesAndSwaps[4], comparesAndSwaps[5]);

ArrCopy(arr, arrCopy, size);

ShellSort(arrCopy, size, comparesAndSwaps[6], comparesAndSwaps[7]);

int first = 0, last = size - 1;

QuickSort(arr, first, last, comparesAndSwaps[8], comparesAndSwaps[9]);//передаётся не копия массива, а сам массив

delete[] arrCopy; //для его последующего внесения в диагональ

}

void ArrChange(int \*\*arr, int lines, int columns) {//функция для перевода диагонали в одномерный массив

string name[10] = { "Bubble Sort:", "", "Select Sort:", "", "Insert Sort:", "", "Shell Sort:", "", "Quick Sort:" };

int i, j, a, \*comparesAndSwaps = new int[10];//массив для подсчёта сравнений и перестановок

for (i = 0; i < 10; i++)

comparesAndSwaps[i] = 0;

for (i = 0; i < lines - 1; i++) {//проход по всем строкам кроме последней

a = i;

j = 0;

while (a < lines && j < columns) {//подсчёт элементов в диагонали

a++;

j++;

}

int \*line = new int[j];//создание нового массива размерности кол-ва элементов в диагонали

a = i;

j = 0;

while (a < lines && j < columns) {//копирование эл-ов из диагонали в новый массив

line[j] = arr[a][j];

a++;

j++;

}

Methods(line, j, comparesAndSwaps);//передача нового массива с эл-ми диагонали во все методы сортировки

a = i;

j = 0;

while (a < lines && j < columns) {//замещение исходного порядка эл-ов диагонали на полученный

arr[a][j] = line[j];

a++;

j++;

}

delete[] line;

}

for (i = 1; i < columns - 1; i++) {//проход по всем столбцам кроме 1-го последнего

a = i;

j = 0;

while (j < lines && a < columns) {//подсчёт элементов в диагонали

a++;

j++;

}

int \*line = new int[j];//создание нового массива размерности кол-ва элементов в диагонали

a = i;

j = 0;

while (j < lines && a < columns) {//копирование эл-ов из диагонали в новый массив

line[j] = arr[j][a];

a++;

j++;

}

Methods(line, j, comparesAndSwaps);//передача нового массива с эл-ми диагонали во все методы сортировки

a = i;

j = 0;

while (j < lines && a < columns) {//замещение исходного порядка эл-ов диагонали на полученный

arr[j][a] = line[j];

a++;

j++;

}

delete[] line;

}

cout << endl << "Name:\t " << "Compares: " << "Swaps:" << endl;

for (i = 0; i < 10; i += 2)//вывод таблицы с методами и их кол-вом сравнений и перестановок

cout << name[i] << setw(6) << comparesAndSwaps[i] << setw(10) << comparesAndSwaps[i + 1] << endl;

SetColor(purple);

cout << "Converted array:";

OutputInConsole(arr, lines, columns);//вывод полученного массива в консоль

SetColor(white);

cout << endl << "Do you want to save the result into the file?" << endl

<< "0)No" << endl

<< "1)Yes" << endl

<< "Your choice:";

bool choice = GetBool();//хочет ли пользователь сохранить полученный результат в файл

if (choice) {

bool origOrResult = false;

OutputDataInFile(arr, lines, columns, origOrResult, comparesAndSwaps);

}

}

**// InputAndCheck.cpp**

**// Лабораторная работа №2.**

**// Студент группы 485, Зобнин Илья Михайлович. 2019 год**

#include "pch.h"

using namespace std;

template <typename T>//использование шаблона для использования вариабельного типа данных

T GetInput() {

T userInput;

cin >> userInput;//ввод пользователем необходимых данных

while (cin.fail()) {//цикл пока ввод данных не соответствует заданному типу

cout << "Try again: " << endl;

cin.clear();//обнуление cin.fail

cin.ignore(INT\_MAX, '\n');//игнорирование введённых данных

cin >> userInput;//повторный ввод переменной

}

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

return userInput;

}

//все последующие функции используют предыдущую для ввода пользователем заданных типов данных

int GetInt() {

return GetInput<int>();

}

int GetUnsignedInt() {

int i = GetInput<int>();

if (i < 0) {

cout << "Try again: ";

return GetUnsignedInt();

}

else

return i;

}

bool GetBool(){

return GetInput<bool>();

}

**// Output.cpp**

**// Лабораторная работа №2.**

**// Студент группы 485, Зобнин Илья Михайлович. 2019 год**

#include "pch.h"

#include "InputAndCheck.h"

using namespace std;

using namespace experimental::filesystem;

void SetColor(int color) {//функция для изменения цвета текста в консоли

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

}

void OutputArrInFile(int \*\*arr, int lines, int columns, string path) {//функция для вывода кол-ва строк, столбцов и изначального массива

ofstream fileOutput;

fileOutput.open(path, ofstream::trunc);

fileOutput << lines << endl;

fileOutput << columns;

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

fileOutput << endl;

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

if (j == columns - 1)

fileOutput << arr[i][j];

else

fileOutput << arr[i][j] << " ";

}

}

fileOutput.close();

SetColor(green);

cout << "Saved successfully! " << endl;

SetColor(white);

}

void OutputResultInFile(int \*\*arr, int lines, int columns, string path, int \*comparesAndSwaps) {//функция для вывода изменённого массива, таблицы с методами сортировки

ofstream fileOutput;

string name[10] = { "Bubble Sort:", "", "Select Sort:", "", "Insert Sort:", "", "Shell Sort:", "", "Quick Sort:" };

fileOutput.open(path, ofstream::trunc);

fileOutput << "Name:\t " << "Compares: " << "Swaps:" << endl;

for (int i = 0; i < 10; i += 2)

fileOutput << name[i] << setw(6) << comparesAndSwaps[i] << setw(10) << comparesAndSwaps[i + 1] << endl;

fileOutput << "Converted Array: ";

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

fileOutput << endl;

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

fileOutput << arr[i][j] << " ";

}

fileOutput.close();

SetColor(purple);

cout << "Saved successfully! " << "\n\n";

SetColor(white);

}

void OutputInConsole(int \*\*arr, int lines, int columns) {//функция для вывода массива в консоль

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

cout << endl;

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

cout << "\t" << arr[i][j] << " ";

}

}

}

void OutputDataInFile(int \*\*arr, int lines, int columns, bool origOrResult, int \*comparesAndSwaps = 0) {

string filePath;

bool isDataSaved = false;

do {

cout << "Input path to file, for example: C:\\Directory\\textfile.txt: " << endl;

cin >> filePath;

if (ifstream(filePath))//проверка на существование файла

if (!is\_regular\_file(filePath)) {//проверка на запрещённые имена (aux, com и т.д.)

SetColor(red);

cout << "Incorrect file path. Enter another one!" << endl;

SetColor(white);

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

continue;

}

else{//хочет ли пользователь перезаписать содержимое

SetColor(red);

cout << "File already exists! Do you want to erase all data and write your array in it?" << endl;

SetColor(white);

cout << "0)No" << endl << "1)Yes" << endl << "Your choice:";

bool isAnother = GetBool();

if (!isAnother)

continue;

}

ofstream myFile(filePath, ofstream::app);

if (!myFile) {//проверка на доступ к уже существующему файлу

SetColor(red);

cout << "Access denied! Enter another path!" << endl;

SetColor(white);

myFile.close();

continue;

}

myFile.close();

if (origOrResult)//запись либо оригинального массива либо результата

OutputArrInFile(arr, lines, columns, filePath);

else

OutputResultInFile(arr, lines, columns, filePath, comparesAndSwaps);

isDataSaved = true;

} while (!isDataSaved);

} **[--- Конец программы ---]**