МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

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

**Дисциплина “Программирование”**

Отчёт по лабораторной работе №3

Методы сортировки

Преподаватели: Корниенко Иван Григорьевич.

Федин Алексей Константинович

Выполнила студентка 405 группы: Сорокина Ульяна Сергеевна

Санкт-Петербург

2021

### 1 Постановка задачи

Упорядочить каждый столбец матрицы по возрастанию.

### 2 Исходные данные

Исходными данными являются вещественные числа.

### 3 Особые ситуации

Необходимо рассмотреть следующую особую ситуацию:

количество столбцов и строк должно быть больше нуля.

### 4 Математические методы и алгоритмы решения задач

1. Сортировка пузырьком: будем идти по массиву слева направо. Если текущий элемент больше следующего, меняем их местами. Делаем так, пока массив не будет отсортирован. Заметим, что после первой итерации самый большой элемент будет находиться в конце массива, на правильном месте. После двух итераций на правильном месте будут стоять два наибольших элемента, и так далее. Очевидно, не более чем после n итераций массив будет отсортирован. Таким образом, асимптотика в худшем и среднем случае – O(n2), в лучшем случае – O(n).  
2. Сортировка отбором: на очередной итерации будем находить минимум в массиве после текущего элемента и менять его с ним, если надо. Таким образом, после i-ой итерации первые i элементов будут стоять на своих местах. Асимптотика: O(n2) в лучшем, среднем и худшем случае. Нужно отметить, что эту сортировку можно реализовать двумя способами – сохраняя минимум и его индекс или просто переставляя текущий элемент с рассматриваемым, если они стоят в неправильном порядке. Первый способ оказался немного быстрее, поэтому он и реализован.

3. Сортировка вставками: создадим массив, в котором после завершения алгоритма будет лежать ответ. Будем поочередно вставлять элементы из исходного массива так, чтобы элементы в массиве-ответе всегда были отсортированы. Асимптотика в среднем и худшем случае – O(n2), в лучшем – O(n). Реализовывать алгоритм удобнее по-другому (создавать новый массив и реально что-то вставлять в него относительно сложно): просто сделаем так, чтобы отсортирован был некоторый префикс исходного массива, вместо вставки будем менять текущий элемент с предыдущим, пока они стоят в неправильном порядке.

4. Сортировка Шелла: используем ту же идею, что и сортировка с расческой, и применим к сортировке вставками. Зафиксируем некоторое расстояние. Тогда элементы массива разобьются на классы – в один класс попадают элементы, расстояние между которыми кратно зафиксированному расстоянию. Отсортируем сортировкой вставками каждый класс. В отличие от сортировки расческой, неизвестен оптимальный набор расстояний. Существует довольно много последовательностей с разными оценками. Последовательность Шелла – первый элемент равен длине массива, каждый следующий вдвое меньше предыдущего. Асимптотика в худшем случае – O(n2). Последовательность Хиббарда – 2n — 1, асимптотика в худшем случае – O(n1,5), последовательность Седжвика (формула нетривиальна, можете ее посмотреть по ссылке ниже) — O(n4/3), Пратта (все произведения степеней двойки и тройки) — O(nlog2n). Отмечу, что все эти последовательности нужно рассчитать только до размера массива и запускать от большего от меньшему (иначе получится просто сортировка вставками). Также я провел дополнительное исследование и протестировал разные последовательности вида si = a \* si — 1 + k \* si — 1 (отчасти это было навеяно эмпирической последовательностью Циура – одной из лучших последовательностей расстояний для небольшого количества элементов). Наилучшими оказались последовательности с коэффициентами a = 3, k = 1/3; a = 4, k = 1/4 и a = 4, k = -1/5.

5. Быстрая сортировка: выберем некоторый опорный элемент. После этого перекинем все элементы, меньшие его, налево, а большие – направо. Рекурсивно вызовемся от каждой из частей. В итоге получим отсортированный массив, так как каждый элемент меньше опорного стоял раньше каждого большего опорного. Асимптотика: O(nlogn) в среднем и лучшем случае, O(n2). Наихудшая оценка достигается при неудачном выборе опорного элемента. Моя реализация этого алгоритма совершенно стандартна, идем одновременно слева и справа, находим пару элементов, таких, что левый элемент больше опорного, а правый меньше, и меняем их местами. Помимо чистой быстрой сортировки, участвовала в сравнении и сортировка, переходящая при малом количестве элементов на сортировку вставками. Константа подобрана тестированием, а сортировка вставками — наилучшая сортировка, подходящая для этой задачи.

### 5 Форматы представления данных

Таблица 1 – Классы, используемые в программе

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя класса** | **Имя переменной** | **Тип** | **Описание** |
| Array | - | - | Взаимодействие с пользователем |
| columns | int | Количество столбцов |
| rows | int | Количество строчек |
| arr | double\*\* | Массив элементов |
|  | **Метод** | **Тип** | **Описание** |
|  | operator [] | double\* | Доступа к массиву класса |
|  | operator = | Array & | Присваивание значений элементов класса другому классу |
|  | getColumns | int | Доступа к элементу column s |
|  | getRows | int | Доступа к элементу rows |
|  | operator == | bool | Сравнивает массивы |
|  | sort | void | Сортирует массив(используется для тестов) |
|  | getMaxLengthValue | int | Возвращает самое длинное число |
|  | print | void | Вывод массива |
| Input | - | - | Абстрактный класс |
| read | virtual Array | Считывания данных в массив |
|  | isFileInput | virtual bool | Возвращает true если тип ввода из файла |
| KeyboardInput | read | Array | Считывание данных из консоли |
|  | isFileInput | bool | Возвращает true если тип ввода из файла |

Продолжение таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя класса** | **Имя переменной** | **Тип** | **Описание** |
| FileInput | read | Array | Считывание данных из файла |
|  | isFileInput | bool | Возвращает true если тип ввода из файла |
| RandomInput | read | Array | Рандомное заполнение массива |
|  | isFileInput | bool | Возвращает true если тип ввода из файла |
| ISort | - | - | Абстрактный класс |
|  | comparison | int | Подсчет количества сравнений |
|  | premutation | int | Подсчет количества перестановок |
|  | **Метод** | **Тип** | **Описание** |
|  | sort | virtual Array | Функция сортировки массива |
|  | getComparison | int | Возвращает количество сравнений |
|  | getPremutation | int | Возвращает количество перестановок |
| Bubble\_Sort | - | - | Класс сортировки пузырьком |
|  | sort | Array | сортировка пузырьком |
| Selection\_Sort | - | - | Класс сортировки выбором |
|  | sort | Array | сортировка выбором |
| Insertion\_Sort | - | - | Класс сортировки вставками |
|  | sort | Array | сортировка вставками |
| Shell\_Sort | - | - | Класс сортировки шелла |
|  | sort | Array | сортировка шелла |
| Quick\_Sort | - | - | Класс быстрой сортировки |
|  | sort | Array | быстрая сортировки |

Таблица 2 – константы, используемые в программе

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя переменной** | **Тип** | **Значение** | **Описание** |
| ALL\_TEST\_PASSED | const int | 5 | Проверка: прошли ли все тесты проверку |

Таблица 3 – перечисляемые типы, используемые в программе

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя типа** | **Имя переменной** | **Значение** | **Описание** |
| ChoiceTypeInput | - | - | Используется для проверки выбора пользователя |
| CHOICE\_KEYBOARD | 1 | Используется для выбора способа ввода через клавиатуру |
| CHOICE\_FILE | 2 | Используется для выбора способа ввода через клавиатуру |
| CHOICE\_RANDOM | 3 | Используется для выбора заполнения массива рандомными числами |

### 6 Структура программы

Таблица 4 – Функции, составляющие модуль SaveData.h

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| saveInputArray | Сохранение введённых данных |
| saveChangeArray | Сохранение преобразованных данных |

Таблица 5 – Функции, составляющие модуль FormattingInput.cpp

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| getInt | Проверяет является ли введённое число цифрой и возвращает его целым числом |
| getDouble | Проверяет является ли введённое число цифрой и возвращает его числом типа double |

Таблица 6 – Функции, составляющие модуль OpenFile.h

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| openFileInput | Реализует открытие файлов для считывания |
| openFileOutput | Реализует открытие файлов для записи |

Таблица 7 – Функции, составляющие модуль MainMenu.h

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| endProgramMenu | Выводит меню завершения программы |
| mainMenu | Меню выбора работы с массивом или тестирования программы |
| greeting | Приветствие |

Таблица 8 – Функции, составляющие модуль ArrayWork.cpp

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| arrayWork | Реализует работу с массивом |

Таблица 9 – Функции, составляющие модуль Test.cpp

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| test | Тестирует программу |

Таблица 10 – Функции, составляющие модуль Input.cpp

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| choiceInputType | Обрабатывает выбор типа ввода |

### 7 Блок-схем алгоритмов программы



Рисунок 1 – Блок-схема алгоритма сортировки пузырьком



Рисунок 2 – Блок-схема алгоритма сортировки выбором



Рисунок 3 – Блок-схема алгоритма сортировки вставками



Рисунок 4 – Блок-схема алгоритма сортировки шелла



Рисунок 5 – Блок-схема алгоритма быстрой сортировки

  

Рисунок 6 – Продолжение блок-схемы алгоритма быстрой сортировки

### 8 Описание хода выполнения лабораторной работы

1. В ходе лабораторной работы было создано решение (Solution) в интегрированной среде разработки Microsoft Visual Studio C++ 2019. В нем был создан проект.
2. В ходе выполнения программы была выявлена проблема: при вводе данных из файла, пользователь должен был вводить пути к 2 файлам для ввода размера и массива. Для решения данной проблемы было принято решение объединить ввод размера и массива из одного файла.
3. В ходе выполнения программы была выявлена проблема: после отработки функций, куда передавался класс Array, массив удалялся. Для решения этой проблемы в класс был добавлен конструктор копирования.
4. В ходе выполнения программы была выявлена проблема: у пользователя была возможность ввести некорректный размер массива. Для решения этой проблемы была добавлена проверка вводимого числа.

### 9 Результаты работы программы

В результате сортировки программа выводит двумерный массив с отсортированными по возрастанию столбцами.

### 10 Исходный текст программы

[Начало Main.cpp ---]

#include <iostream>

#include <string>

#include "MainMenu.h"

using namespace std;

enum ChoiceTypeInput {

CHOICE\_ARRAY = 1,

CHOICE\_TESTING

};

int getInt();

void arrayWork();

void test();

int main() {

greeting();

int choice = 0;

string isChoiceRepeat = "";

do {

do {

mainMenu();

choice = getInt();

switch (choice) {

case CHOICE\_ARRAY:

arrayWork();

break;

case CHOICE\_TESTING:

test();

break;

default:

cout << "There is no such action. Try again" << endl;

}

} while ( (choice != CHOICE\_ARRAY) && (choice != CHOICE\_TESTING) );

do {

endProgramMenu();

cin >> isChoiceRepeat;

if (isChoiceRepeat.compare("n") && isChoiceRepeat.compare("y")) {

cout << "There is no such action." << endl;

}

} while (isChoiceRepeat.compare("n") && isChoiceRepeat.compare("y"));

} while (isChoiceRepeat.compare("n"));

return 0;

}

[--- Конец Main.cpp]

[Начало ArrayWork.cpp ---]

#include <iostream>

#include <iomanip>

#include "Input.h"

#include "SaveData.h"

#include "Sort.h"

using namespace std;

unique\_ptr<Input> choiceInputType();

void arrayWork(void) {

unique\_ptr<Input> inp = choiceInputType();

Array arr = inp->Read();

cout << "Input array:" << endl;

arr.print();

BubbleSort bubbleSort;

Array arrayBubbleSort = bubbleSort.sort(arr);

cout << "BubbleSort array:" << endl;

arrayBubbleSort.print();

SelectionSort selectionSort;

Array arrSelectionSort = selectionSort.sort(arr);

cout << "SelectionSort array:" << endl;

arrSelectionSort.print();

InsertionSort insertionSort;

Array arrInsertionSort = insertionSort.sort(arr);

cout << "InsertionSort array:" << endl;

arrInsertionSort.print();

ShellSort shellSort;

Array arrShellSort = shellSort.sort(arr);

cout << "ShellSort array:" << endl;

arrShellSort.print();

QuickSort quickSort;

Array arrQuickSort = quickSort.sort(arr);

cout << "QuickSort array:" << endl;

arrQuickSort.print();

cout << endl << " Comparison Permutation" << endl;

cout << "Bubble Sort:" << setw(14) << bubbleSort.getComparison() << setw(14) << bubbleSort.getPermutation() << endl;

cout << "Selection Sort:" << setw(11) << selectionSort.getComparison() << setw(14) << selectionSort.getPermutation() << endl;

cout << "Insertion Sort:" << setw(11) << insertionSort.getComparison() << setw(14) << insertionSort.getPermutation() << endl;

cout << "Shell Sort:" << setw(15) << shellSort.getComparison() << setw(14) << shellSort.getPermutation() << endl;

cout << "Quick Sort:" << setw(15) << quickSort.getComparison() << setw(14) << quickSort.getPermutation() << endl;

if (inp->isFileInput() == false) {

saveInputArray(arr);

}

saveChangeArray(arrShellSort);

}

[--- Конец ArrayWork.cpp]

[Начало Sort.cpp ---]

#include <cmath>

#include <iostream>

#include "Sort.h"

#include "ArrayClass.h"

using namespace std;

int ISort::getComparison(void) {

return comparison;

}

int ISort::getPermutation(void) {

return permutation;

}

Array BubbleSort::sort(Array arr) {

for (int j = 0; j < arr.getColumns(); j++) {

for (int k = arr.getRows() - 1; k > 0; k--) {

int cyclePermutations = 0;

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

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

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

permutation++;

cyclePermutations++;

}

comparison++;

}

if (cyclePermutations == 0) {

break;

}

}

}

return arr;

}

Array SelectionSort::sort(Array arr) {

int minValueIndex = 0;

for (int j = 0; j < arr.getColumns(); j++) {

for (int k = 0; k < (arr.getRows() - 1); k++) {

minValueIndex = k;

for (int i = k + 1; i < arr.getRows(); i++) {

if (arr[i][j] < arr[minValueIndex][j]) {

minValueIndex = i;

}

comparison++;

}

if (minValueIndex != k) {

swap(arr[minValueIndex][j], arr[k][j]);

permutation++;

}

}

}

return arr;

}

Array InsertionSort::sort(Array arr) {

for (int j = 0; j < arr.getColumns(); j++) {

for (int i = 1; i < arr.getRows(); i++) {

int rowValue = i;

while ( ((rowValue - 1) >= 0) && (arr[rowValue][j] < arr[rowValue - 1][j]) ) {

swap(arr[rowValue][j], arr[rowValue - 1][j]);

permutation++;

comparison++;

rowValue--;

}

if ((rowValue - 1) != -1) {

comparison++;

}

}

}

return arr;

}

Array ShellSort::sort(Array arr) {

for (int j = 0; j < arr.getColumns(); j++) {

int d = arr.getRows() / 2;

while (d != 0) {

for (int k = d; k < arr.getRows(); k++) {

for (int i = k; (i - d) >= 0; i -= d) {

if (arr[i][j] < arr[i - d][j]) {

swap(arr[i][j], arr[i - d][j]);

permutation++;

} else {

comparison++;

break;

}

comparison++;

}

}

d /= 2;

}

}

return arr;

}

Array QuickSort::recursionSort(Array arr, int low, int high, int j) {

int center = (high + low) / 2;

double safeValue = 0;

for (int i = low; i < center; i++) {

if (arr[i][j] > arr[center][j]) {

safeValue = arr[i][j];

for (int k = i; k < center; k++) {

arr[k][j] = arr[k + 1][j];

}

arr[center][j] = safeValue;

i--;

center--;

permutation++;

}

comparison++;

}

for (int i = high; i > center; i--) {

if (arr[i][j] < arr[center][j]) {

safeValue = arr[i][j];

for (int k = i; k > center; k--) {

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

}

arr[center][j] = safeValue;

i++;

center++;

permutation++;

}

comparison++;

}

if ((center - low) > 1) {

arr = recursionSort(arr, low, center, j);

}

if ((high - center) > 1) {

arr = recursionSort(arr, center + 1, high, j);

}

return arr;

}

Array QuickSort::sort(Array arr) {

for (int j = 0; j < arr.getColumns(); j++) {

arr = recursionSort(arr, 0, arr.getRows() - 1, j);

}

return arr;

}

[--- Конец Sort.cpp]

[Начало SaveData.cpp ---]

#include <iostream>

#include <iomanip>

#include <fstream>

#include <string>

#include "OpenFile.h"

#include "ArrayClass.h"

using namespace std;

void saveInputArray(Array arr) {

cout << endl << "Do you want to save the initial data in a file?" << endl

<< "y - Yes" << endl

<< "n - No" << endl

<< "Choice: ";

string choiceRepeat = "";

do {

cin >> choiceRepeat;

if (choiceRepeat.compare("n") && choiceRepeat.compare("y")) {

cout << "There is no such action." << endl

<< "Choice: ";

}

} while (choiceRepeat.compare("n") && choiceRepeat.compare("y"));

if (choiceRepeat.compare("n")) {

ofstream file = openFileOutput();

file << arr.getRows() << endl << arr.getColumns() << endl;

int length = arr.getMaxLengthValue();

for (int i = 0; i < arr.getRows(); i++) {

for (int j = 0; j < arr.getColumns(); j++) {

file << setw(length) << arr[i][j] << " ";

}

file << endl;

}

file.close();

}

};

void saveChangeArray(Array arr) {

cout << endl << "Do you want to save the result of the program?" << endl

<< "y - Yes" << endl

<< "n - No" << endl

<< "Choice: ";

string choiceRepeat = "";

do {

cin >> choiceRepeat;

if (choiceRepeat.compare("n") && choiceRepeat.compare("y")) {

cout << "There is no such action." << endl

<< "Choice: ";

}

} while (choiceRepeat.compare("n") && choiceRepeat.compare("y"));

if (choiceRepeat.compare("n")) {

ofstream file;

file = openFileOutput();

for (int i = 0; i < arr.getRows(); i++) {

for (int j = 0; j < arr.getColumns(); j++) {

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

}

file << endl;

}

file.close();

}

}

[--- Конец SaveData.cpp]

[Начало OpenFile.cpp ---]

#include <iostream>

#include <fstream>

#include <string>

#include <filesystem>

using namespace std;

ofstream openFileOutput(void) {

ofstream file;

string path = "";

bool fileOpenComplit = false;

do {

cout << "Enter the path to the file" << endl;

cin >> path;

if (ifstream(path)) {

error\_code ec;

if (!filesystem::is\_regular\_file(path, ec)) {

cout << "The file could not be opened. Try again." << endl;

continue;

}

cout << "This file exists. Do you want overwrite a file?" << endl;

string choice;

do {

cout << "y - Yes" << endl

<< "n - No" << endl

<< "Choice: ";

cin >> choice;

if (choice.compare("n") && choice.compare("y")) {

cout << "There is no such action." << endl

<< "Do you want overwrite a file?" << endl;

}

} while (choice.compare("n") && choice.compare("y"));

if (!choice.compare("n")) {

continue;

}

}

file.open(path);

error\_code ec;

if (!filesystem::is\_regular\_file(path, ec)) {

cout << "The file could not be opened." << endl;

continue;

}

if (!file.is\_open()) {

cout << "The file could not be opened." << endl;

continue;

}

fileOpenComplit = true;

} while (!fileOpenComplit);

return file;

}

ifstream openFileInput(void) {

cout << "Enter the path to the file" << endl;

ifstream file;

string path;

bool isOpeningFile = false;

do {

error\_code ec;

do {

cin >> path;

if (!filesystem::is\_regular\_file(path, ec)) {

cout << "The file could not be opened." << endl

<< "Enter a different name or path to the file: ";

}

} while (!filesystem::is\_regular\_file(path, ec));

file.open(path);

if (file.is\_open()) {

isOpeningFile = true;

} else {

cout << "The file could not be opened." << endl

<< "Enter a different name or path to the file: ";

isOpeningFile = false;

}

} while (isOpeningFile == false);

return file;

}

[--- Конец OpenFile.cpp]

[Начало Menu.cpp ---]

#include <iostream>

using namespace std;

void valueInputMenu(void) {

cout << endl << "What do you want?" << endl

<< "1)Enter data from the keyboard" << endl

<< "2)Enter data from file" << endl

<< "3)Enter data from random" << endl

<< "Choice: ";

}

void endProgramMenu(void) {

cout << endl << "Do you want to repeat the program?" << endl

<< "y - Yes" << endl

<< "n - No" << endl

<< "Choice: ";

}

void mainMenu(void) {

cout << endl << "What do you want?" << endl

<< "1)Working with an array" << endl

<< "2)Start test" << endl

<< "Choice: ";

}

[--- Конец Menu.cpp]

[Начало Input.cpp ---]

#include <iostream>

#include <string>

#include <ctime>

#include <fstream>

#include "OpenFile.h"

#include "ArrayClass.h"

#include "Input.h"

using namespace std;

enum ChoiceTypeInput {

CHOICE\_KEYBOARD = 1,

CHOICE\_FILE,

CHOICE\_RANDOM,

};

int getInt(void);

double getDouble(void);

void valueInputMenu(void);

Array KeyboardInput::Read() {

int numberRows = 0;

do {

cout << endl << "Enter the number of rows of the array: ";

numberRows = getInt();

if (numberRows <= 0) {

cout << endl << "rows must be > 0. Try again." << endl;

}

} while (numberRows <= 0);

int numberColumns = 0;

do {

cout << endl << "Enter the number of columns of the array: ";

numberColumns = getInt();

if (numberColumns <= 0) {

cout << endl << "columns must be > 0. Try again." << endl;

}

} while (numberColumns <= 0);

Array arr(numberRows, numberColumns);

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

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

cout << i + 1 << '.' << j + 1 << ")";

arr[i][j] = getDouble();

}

}

return arr;

}

bool KeyboardInput::isFileInput() {

return false;

}

Array FileInput::Read() {

int rows = 0;

int columns = 0;

while (true) {

cout << endl << "Input data" << endl;

ifstream file = openFileInput();

try {

string valueInFile;

getline(file, valueInFile);

rows = stoi(valueInFile);

if (rows <= 0) {

cout << endl << "rows in file must be > 0. Try again." << endl;

continue;

}

getline(file, valueInFile);

columns = stoi(valueInFile);

if (columns <= 0) {

cout << endl << "columns in file must be > 0. Try again." << endl;

continue;

}

Array arr(rows, columns);

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

string value;

string RowValues = "";

getline(file, RowValues);

int columnNumber = 0;

for (int j = 0; j < RowValues.length(); j++) {

if ((RowValues[j] == ' ') && (value != "")) {

arr[i][columnNumber] = stod(value);

columnNumber++;

value = "";

} else if (RowValues[j] != ' ') {

value += RowValues[j];

}

}

}

cout << "Input rows = " << rows << endl;

cout << "Input columns = " << columns << endl;

file.close();

return arr;

} catch (invalid\_argument) {

cout << endl << "The value in the file are not correct. Try again." << endl;

}

}

}

bool FileInput::isFileInput() {

return true;

}

Array RandomInput::Read() {

int rows = 0;

int columns = 0;

do {

cout << endl << "Enter the number of rows of the array: ";

rows = getInt();

if (rows <= 0) {

cout << endl << "rows must be > 0. Try again." << endl;

}

} while (rows <= 0);

do {

cout << endl << "Enter the number of columns of the array: ";

columns = getInt();

if (columns <= 0) {

cout << endl << "columns must be > 0. Try again." << endl;

}

} while (columns <= 0);

Array arr(rows, columns);

srand((unsigned int)time(NULL));

int randomLimits = 0;

do {

cout << endl << "Random numbers in a range (-a, a)" << endl

<< "Enter a: ";

randomLimits = getInt();

if (randomLimits <= 0) {

cout << endl << "a must be > 0. Try again." << endl;

}

} while (randomLimits <= 0);

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

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

if (rand() % 2) {

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

} else {

arr[i][j] = -1 \* (rand() % randomLimits);

}

}

}

return arr;

}

bool RandomInput::isFileInput() {

return false;

}

unique\_ptr<Input> choiceInputType() {

int choice = 0;

do {

valueInputMenu();

choice = getInt();

if (choice == CHOICE\_KEYBOARD) {

return unique\_ptr<KeyboardInput>(new KeyboardInput);

} else if (choice == CHOICE\_FILE) {

return unique\_ptr<FileInput>(new FileInput);

} else if (choice == CHOICE\_RANDOM) {

return unique\_ptr<RandomInput>(new RandomInput);

} else {

cout << "There is no such action" << endl;

}

} while ((choice != CHOICE\_KEYBOARD) && (choice != CHOICE\_FILE)

&& (choice != CHOICE\_RANDOM));

return 0;

}

[--- Конец Input.cpp]

[Начало Greeting.cpp ---]

#include <iostream>

using namespace std;

void greeting(void){

cout << "Laboratory work 3" << endl

<< "Using the C ++ language for mathematical calculations" << endl

<< "Order each column of a matrix in ascending order." << endl

<< "Student of group 405, Ulyana Sorokina. 2021 year" << endl;

}

[--- Конец Greeting.cpp]

[Начало FormattingInput.cpp ---]

#include <iostream>

#include <limits>

using namespace std;

int getInt(void) {

int input = 0;

cin >> input;

while (cin.fail()) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Input Error. Enter the number." << endl;

cin >> input;

}

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

return input;

}

double getDouble(void) {

double input = 0;

cin >> input;

while (cin.fail()) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Input Error. Enter the number." << endl;

cin >> input;

}

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

return input;

}

[--- Конец FormattingInput.cpp]

[Начало ArrayClass.cpp ---]

#include <iomanip>

#include <iostream>

#include <string>

#include "ArrayClass.h"

using namespace std;

Array::Array(int arrayRows, int ArrayColumns) : rows(arrayRows), columns(ArrayColumns) {

arr = new double\*[rows];

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

arr[i] = new double[columns];

}

}

Array::Array(const Array& oldArray) : columns(oldArray.columns), rows(oldArray.rows) {

arr = new double\*[rows];

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

arr[i] = new double[columns];

}

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

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

arr[i][j] = oldArray.arr[i][j];

}

}

}

Array::~Array() {

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

delete[](arr[i]);

}

delete[] arr;

}

int Array::getColumns(void) {

return columns;

}

int Array::getRows(void) {

return rows;

}

int Array::getMaxLengthValue(void) {

double maxValue = 0;

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

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

if (abs(arr[i][j]) > abs(maxValue)) {

maxValue = arr[i][j];

}

}

}

string stringMaxValue = to\_string(maxValue);

return stringMaxValue.length();

}

void Array::print(void) {

int length = getMaxLengthValue();

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

cout << "|";

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

cout << setw(length) << arr[i][j] << "|";

}

cout << endl;

}

}

void Array::sort(void) {

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

for (int k = rows - 1; k > 0; k--) {

int cyclePermutations = 0;

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

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

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

cyclePermutations++;

}

}

if (cyclePermutations == 0) {

break;

}

}

}

}

Array & Array::operator = (Array&& oldArray) {

if (this != &oldArray) {

if (arr != nullptr) {

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

delete[](arr[i]);

}

delete arr;

}

rows = oldArray.rows;

columns = oldArray.columns;

arr = oldArray.arr;

oldArray.arr = nullptr;

oldArray.rows = 0;

oldArray.columns = 0;

}

return \*this;

}

double\* Array::operator [] (int index) {

return arr[index];

}

bool Array::operator == (Array& compareArr) {

if (compareArr.rows == rows && compareArr.columns == columns) {

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

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

if (arr[i][j] != compareArr[i][j]) {

return false;

}

}

}

return true;

}

return false;

}

[--- Конец ArrayClass.cpp]

[Начало Sort.h ---]

#pragma once

#include "ArrayClass.h"

class ISort {

protected:

int comparison = 0;

int permutation = 0;

public:

virtual Array sort(Array arr) = 0;

int getComparison(void);

int getPermutation(void);

virtual ~ISort() = default;

};

class BubbleSort final : public ISort {

public:

Array sort(Array arr);

};

class SelectionSort final : public ISort {

public:

Array sort(Array arr) ;

};

class InsertionSort final : public ISort {

public:

Array sort(Array arr);

};

class ShellSort final : public ISort {

public:

Array sort(Array arr);

};

class QuickSort final : public ISort {

Array recursionSort(Array arr, int low, int high, int j);

public:

Array sort(Array arr);

};

[--- Конец Sort.h]

[Начало SaveData.h ---]

#pragma once

#include "ArrayClass.h"

void saveInputArray(Array arr);

void saveChangeArray(Array arr);

[--- Конец SaveData.h]

[Начало MainMenu.h ---]

#pragma once

void endProgramMenu(void);

void mainMenu(void);

void greeting(void);

[--- Конец MainMenu.h]

[Начало Input.h ---]

#pragma once

#include "ArrayClass.h"

class Input {

public:

virtual Array Read() = 0;

virtual bool isFileInput() = 0;

virtual ~Input() = default;

};

class KeyboardInput final : public Input {

public:

Array Read();

bool isFileInput();

};

class FileInput final : public Input {

public:

Array Read();

bool isFileInput();

};

class RandomInput final : public Input {

public:

Array Read();

bool isFileInput();

};

[--- Конец Input.h]

[Начало ArrayClass.h ---]

#pragma once

class Array final {

double \*\*arr = nullptr;

int columns = 0;

int rows = 0;

public:

Array(int arrayRows, int arrayColumns);

Array(const Array& oldArr);

~Array();

int getColumns(void);

int getRows(void);

int getMaxLengthValue(void);

void print(void);

void sort(void);

double\* operator [] (int index);

bool operator == (Array& oldArr);

Array & operator = (Array&& oldArray);

Array & operator = (Array& oldArray) = delete;

};

[--- Конец ArrayClass.h]

[Начало Test.cpp ---]

#include <iostream>

#include "ArrayClass.h"

#include "Sort.h"

#include "Input.h"

using namespace std;

const int ALL\_TEST\_PASSED = 5;

void test(void) {

const int rows = 20;

const int columns = 20;

const int randomLimits = 100;

Array arr(rows, columns);

srand((unsigned int)time(NULL));

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

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

if (rand() % 2) {

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

}

else {

arr[i][j] = -1 \* (rand() % randomLimits);

}

}

}

int valueTestPassed = 0;

BubbleSort bubbleSort;

Array arrayBubbleSort = bubbleSort.sort(arr);

SelectionSort selectionSort;

Array arrSelectionSort = selectionSort.sort(arr);

InsertionSort insertionSort;

Array arrInsertionSort = insertionSort.sort(arr);

ShellSort shellSort;

Array arrShellSort = shellSort.sort(arr);

QuickSort quickSort;

Array arrQuickSort = quickSort.sort(arr);

arr.sort();

if (arr == arrayBubbleSort) {

valueTestPassed++;

} else {

cout << endl << "Test1 faild" << endl;

}

if (arr == arrSelectionSort) {

valueTestPassed++;

} else {

cout << endl << "Test2 faild" << endl;

}

if (arr == arrInsertionSort) {

valueTestPassed++;

} else {

cout << endl << "Test3 faild" << endl;

}

if (arr == arrShellSort) {

valueTestPassed++;

} else {

cout << endl << "Test4 faild" << endl;

}

if (arr == arrQuickSort) {

valueTestPassed++;

} else {

cout << endl << "Test5 faild" << endl;

}

if (valueTestPassed == ALL\_TEST\_PASSED) {

cout << endl << "All test passed" << endl;

}

}

[--- Конец Test.cpp]