МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»
(ГУАП)

Кафедра 44 Вычислительных систем и сетей
(наименование)

ОТЧЁТ ПО ПРАКТИКЕ
ЗАЩИЩЁН С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

доцент, канд. техн. наук, доцент		Л. Н. Бариков
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЁТ ПО ПРАКТИКЕ

вид практики учебная

тип практики вычислительная

на тему индивидуального задания изучение методов сортировки структур данных;

совершенствование навыков процедурного программирования на языке *C*/*C*++

при решении задач обработки статических и динамических массивов.

выполнен Ивановым Савелием Михайловичем
фамилия, имя, отчество обучающегося в творительном падеже

по направлению подготовки 09.03.01 Информатика и вычислительная техника
код наименование направления

наименование направления

направленности 02 Компьютерные технологии, системы и
код наименование направленности

сети
наименование направленности

Обучающийся группы № 4342		С. М. Иванов
номер	подпись, дата	инициалы, фамилия

Санкт–Петербург 2024

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

на прохождение учебной практики обучающегося направления подготовки/специальности 09.03.01

1. Фамилия, имя, отчество обучающегося: Иванов Савелий Михайлович

2. Группа: 4342

3. Тема индивидуального задания: изучение методов сортировки структур данных; совершенствование навыков процедурного программирования на языке *C*/*C*++ при решении задач обработки статических и динамических массивов; разработка пользовательского интерфейса.

Исходные данные:

- разработать математическую модель описания поставленной задачи и структурировать её с целью достижения требуемого решения;
- используя технологию процедурного программирования реализовать заданный метод сортировки и применить его для указанных фрагментов числовой матрицы в соответствии с индивидуальным заданием;
- разработать пользовательский интерфейс, позволяющий одновременно наблюдать на экране исходную и результирующую матрицу.

4. Содержание отчетной документации:

4.1. индивидуальное задание;

4.2. отчёт, включающий в себя:

- титульный лист (отчет по практике);
- материалы о выполнении индивидуального задания;
- выводы по результатам практики;
- список использованных источников.

4.3. отзыв руководителя от профильной организации (при прохождении практики в профильной организации).

5. Срок представления отчета на кафедру: «_01_»_____06_____2024 г.

Руководитель практики

доцент, канд. техн. наук, доцент		Л.Н. Бариков
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

Задание принял к исполнению:

Обучающийся

_____	_____	С. М. Иванов
дата	подпись	инициалы, фамилия

Содержание

Введение

Цель работы:

1. изучение методов сортировки структур данных;
2. совершенствование навыков процедурного программирования на языке C/C++ при решении задач обработки статических и динамических массивов;
3. разработка пользовательского интерфейса.

Вариант работы — 17.

Метод сортировки — сортировка по убыванию методом обмена с флагом перестановки.

Глава 1. Общие сведения

На 1.1 изображена область сортировки элементов матриц.

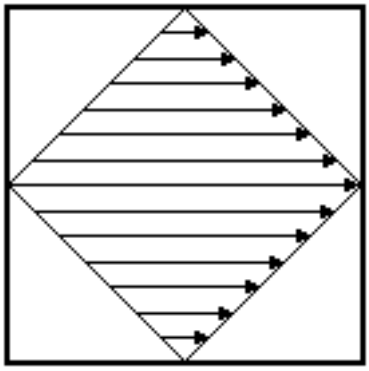

Рисунок 1.1 – Область сортировки элементов матриц

Глава 2. Аналитический раздел

2.1. Формулировка задачи

Используя технологию процедурного программирования реализовать заданный метод сортировки и применить его для указанных фрагментов числовой матрицы в соответствии с индивидуальным заданием. Разработать пользовательский интерфейс, позволяющий:

1. осуществлять ввод исходных данных;
2. наблюдать на экране монитора одновременно исходную и отсортированную матрицу.

2.2. Математическая модель

При решении задачи, а значит в тексте программы, кроме функции `main()` необходимо реализовать следующие функции:

- функцию ввода значений элементов матрицы;
- функцию сортировки элементов матрицы по убыванию методом обмена с флагом перестановки;
- функцию вывода значений элементов матрицы.

Поскольку в задании конкретно не сказано, какой тип массива использовать, выбираем динамический двумерный массив, в котором произвольное количество строк n и столбцов m. Выделение памяти под такой массив происходит на этапе исполнения программы в соответствии со значениями переменных n и m.

Конкретный тип значений элементов матрицы в задании не указан, поэтому для получения легко модернизируемой программы с целью изменения типа обрабатываемых данных определяем новые типы с использованием `typedef`:

- тип значений элементов матрицы `(element)`;
- тип «указатель на `element`» `(*row)`;
- тип «указатель на указатель на `element`» `(*matrix)`.

Исполнение программы (функция `main()`) начинается с объявления переменной `int  width`, задающей размер матрицы. Вводим исходные данные (функция `read_width()`).

Далее с использованием операции `new` производим выделение области динамической памяти под массив указателей на строки матрицы и помещаем адрес этой области в переменную m. Затем организуем цикл выделения памяти под каждую строку матрицы с номерами от 0 до width $-$ 1.

Теперь необходимо задать значения всех элементов матрицы. Для этого в функцию `main()` помещаем вызов функции `read_matrix()`, параметрами которой являются значение указателя m и значения числа строк и столбцов матрицы. Числа строк и столбцов матрицы передаются по значению переменной `width`.

Назначением функции `read_matrix()` является ввод значений элементов матрицы. Доступ к элементам двумерного массива (матрицы) осуществляется по двум индексам: номеру строки и номеру столбца, на пересечении которых находится данный элемент массива. Поэтому перебираем все строки (от 0 до width $-$ 1). Внутри каждой строки перебираем все столбцы (от 0 до width $-$ 1) и задаём значения элементов, лежащих на пересечении строки и столбца с текущими номерами.

Для этапа тестирования с целью сокращения времени прохождения теста осуществляем автоматическое заполнение значений элементов массива с использованием формулы:

$$m[i][j] = i * \text{width} + j; \tag{2.1}$$

После этого функция `read_matrix()` завершает свою работу и передает управление в функцию `main()`.

Выводим на экран исходную матрицу с автоматически заданными значениями. Для этого в функцию `main()` помещаем вызов функции `print_matrix()`, параметрами которой являются значение указателя m и значения числа строк и столбцов матрицы. Числа строк и столбцов матрицы передаются по значению переменной `width`.

При исполнении этой функции организуем цикл перебора всех строк матрицы (от 0 до width $-$ 1). Внутри каждой строки перебираем все столбцы (от 0 до width $-$ 1). Для того чтобы вывести матрицу максимально красиво, задаём ширину поля для выводимого параметра (`cout.width(4)`).

После этого функция `print_matrix()` завершает свою работу и передает управление в функцию `main()`.

Теперь приступаем к решению основной задачи – к сортировке значений строк матрицы с использованием заданного метода сортировки, вызывая функцию `sort_matrix()`, параметрами которой являются значение указателя m и значения числа строк и столбцов матрицы. Числа строк и столбцов матрицы передаются по значению переменной `width`.

При исполнении этой функции перебираем все строки (от 0 до width $-$ 1). Для каждой строки применяем заданный алгоритм сортировки значений её

элементов. После этого функция `sort_matrix()` завершает свою работу и передает управление в функцию `main()`.

Выводим отсортированную матрицу на экран, используя вызов функции `print_matrix()`.

Выполнение программы завершается. Задача решена.

2.3. Блок-схема алгоритма функции `sort_matrix(int, matrix)`

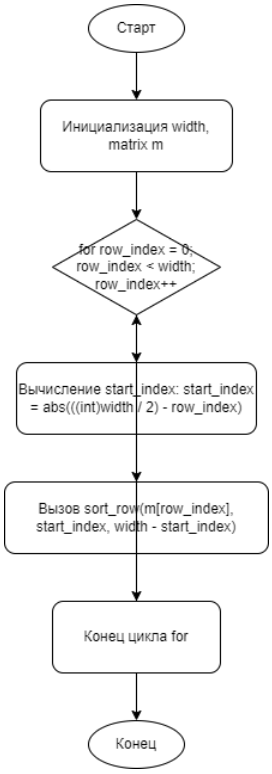

Рисунок 2.2 – Блок-схема алгоритма функции

2.4. Блок-схема алгоритма функции `sort_row(row, int, int)`

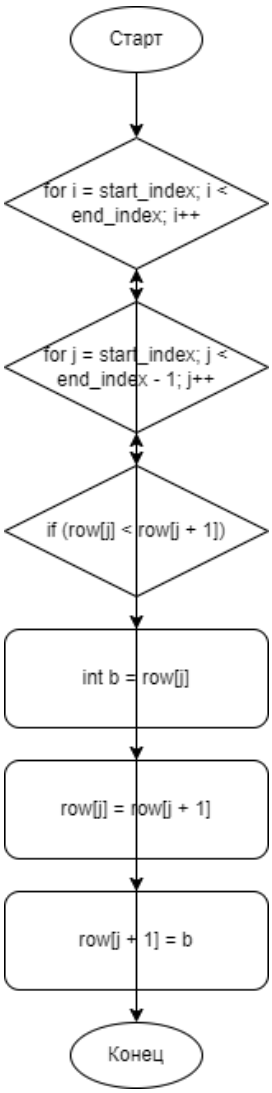

Рисунок 2.3 – Блок-схема алгоритма функции

Глава 3. Технологический раздел

3.1. Задача на программирование

Используя технологию процедурного программирования реализовать заданный метод сортировки и применить его для указанных фрагментов числовой матрицы в соответствии с индивидуальным заданием. Разработать пользовательский интерфейс, позволяющий:

1. осуществлять ввод исходных данных;
2. наблюдать на экране монитора одновременно исходную и отсортированную матрицу.

3.2. Исходный код

```
#ifndef S2_CS_LW10_MAIN_H
#define S2_CS_LW10_MAIN_H

#include <iostream>
#include <cstdlib>
#include <cmath>

using namespace std;

typedef int element;
typedef element *row;
typedef row *matrix;

int read_width();
```

```
void read_matrix(int, matrix);
void sort_row(row, int, int);
void clear_row(row, int);
void sort_matrix(int, matrix);
void print_matrix(int, matrix);

#endif //S2_CS_LW10_MAIN_H
```

Рисунок 3.1 – Листинг заголовочного файла на языке программирования C++

```
#include "main.hpp"

int read_width() {
    cout << "Enter half of matrix width: ";
    int n;
    cin >> n;
    return 2 * n;
}

void read_matrix(int width, matrix m) {
    char auto_or_manual;
    while (true) {
        cout << R"(Enter "A" to fill automaticly matrix, or "M" to fill
manually: )";
        cin >> auto_or_manual;
        if (auto_or_manual == 'a' || auto_or_manual == 'A') {
            for (int i = 0; i < width; i++) {
                for (int j = 0; j < width; j++) {
                    m[i][j] = i * width + j;
                }
            }
```

```
            return;
        } else if (auto_or_manual == 'm' || auto_or_manual == 'M') {
            cout << "Enter matrix " << width << 'x' << width << "\n";
            for (int y = 0; y < width; y++) {
                for (int x = 0; x < width; x++)
                    cin >> m[y][x];
            }
            return;
        }
    }
}

void sort_row(row row, int start_index, int end_index) {
    for (int i = start_index; i < end_index; i++) {
        for (int j = start_index; j < end_index - 1; j++) {
            if (row[j] < row[j + 1]) {
                int b = row[j];
                row[j] = row[j + 1];
                row[j + 1] = b;
            }
        }
    }
}

void clear_row(row row, int start_index, int end_index)
{
    for (int i = start_index; i < end_index; i++)
    {
        row[i] = DEFAULT_ELEMENT;
```

```
    }
}


void sort_matrix(int width, matrix m)
{
    for (int row_index = 0; row_index < width; row_index++)
    {
        int start_index = ((width / 2) - row_index - 1);
        if (start_index < 0) start_index = -1 * start_index - 1;
        sort_row(m[row_index], start_index, width - start_index);
        clear_row(m[row_index], 0, start_index);
        clear_row(m[row_index], width - start_index, width);
    }
}

void print_matrix(int width, matrix m) {
    for (int i = 0; i < width; i++) {
        cout << endl;
        for (int j = 0; j < width; j++) {
            cout.width(4);
            cout << m[i][j];
        }
    }
}

int main() {
    char quitOrRestart = 'r';
```

```
    do {
        quitOrRestart = 'r';
        int width = read_width();
        if (width <= 0) {
            cout << "Matrix width mast be positive";
            continue;
        }

        matrix m = new row[width];
        for (int i = 0; i < width; i++) {
            m[i] = new element[width];
        }

        read_matrix(width, m);

        cout << "\n\nOriginal matrix:";
        print_matrix(width, m);

        sort_matrix(width, m);

        cout << "\n\nSwaped matrix:";
        print_matrix(width, m);

        cout << "\n\nEnter \"R\" to run again, or enter any other letter
to quit programm: ";
        cin >> quitOrRestart;
    } while (quitOrRestart == 'r' || quitOrRestart == 'R');
```

```
    return 0;
}
```

Рисунок 3.2 – Листинг программного кода на языке программирования C++

3.3. Скриншот контрольного примера выполнения программы

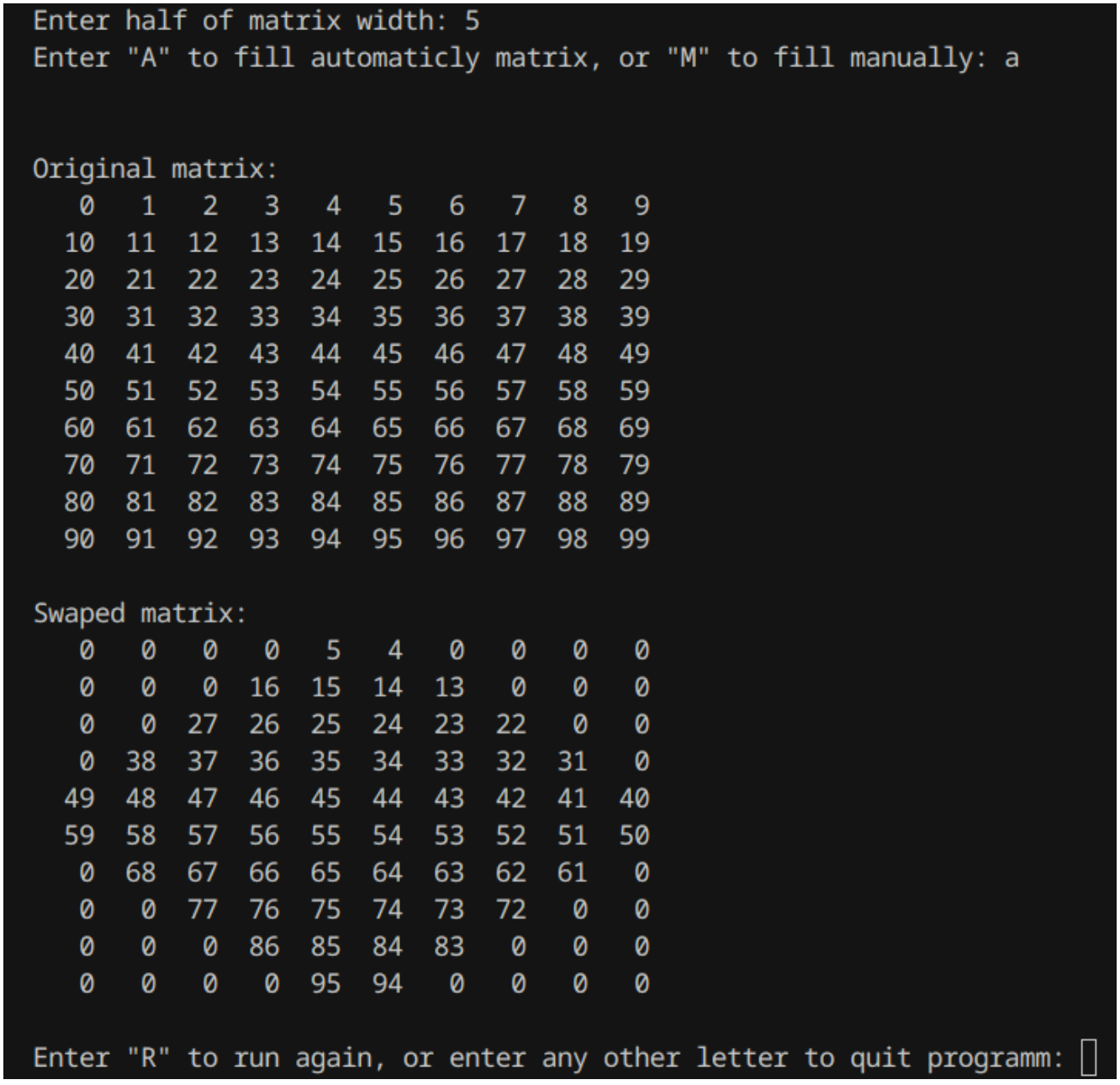

Рисунок 3.4 – Пример выполнения программы

Заключение

В процессе выполнения работы были изучены методы сортировки двумерных массивов, усовершенствованы навыки процедурного программирования и разработки программ с пользовательским интерфейсом на языке C/C++.

Была разработана математическая модель решения задачи, которая помогла структурировать задачу.

С использованием технологии процедурного программирования была реализована сортировка по убыванию методом обмена с флагом перестановки, который был применён для фрагментов матрицы, указанных на рисунке с заданием.