ВВЕДЕНИЕ

Цель – составить программу, которая сравнивает соответствующие элементы двух введённых массивов. При этом программа должна учитывать, что массивы могут быть разного размера.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- Составить IDEFO - диаграмму лабораторной работы;
- Составить схемы необходимых алгоритмов;
- Реализовать алгоритмы на языке Python 3;
- Протестировать реализацию алгоритмов.

ОСНОВНАЯ ЧАСТЬ

1. Аналитическая часть

1.1 Функциональное моделирование

Для определения функций разрабатываемой системы и границ лабораторной работы на основе выданного задания была сформирована диаграмма в IDEFO - нотации, представленной на рисунке 1.

Лабораторная работа описывается одним общим блоком A0.

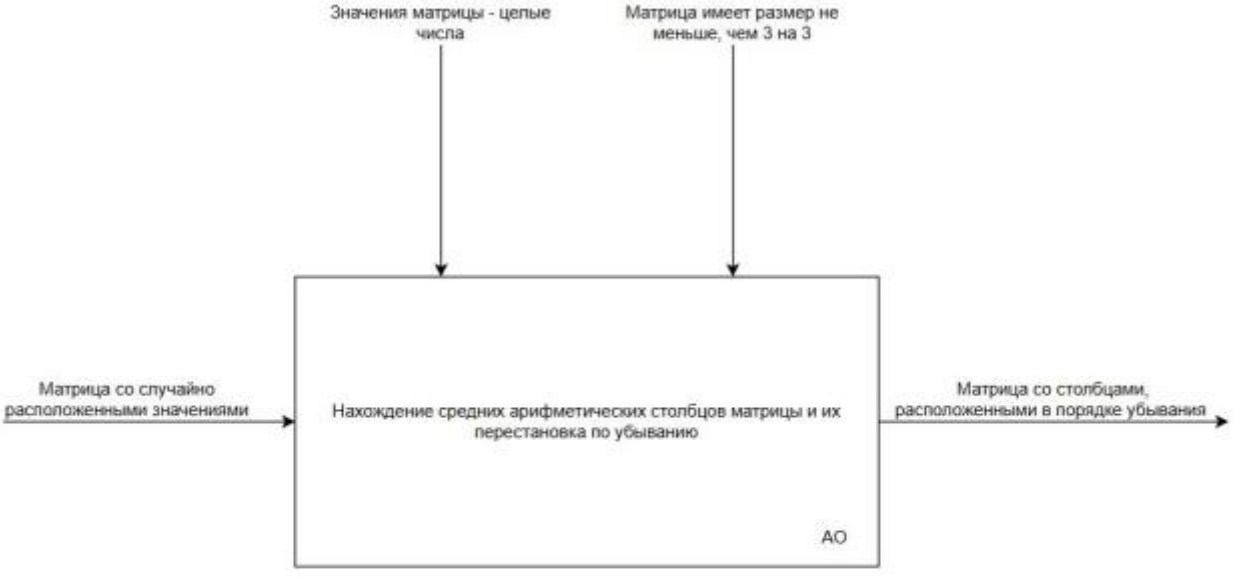

Рисунок 1 – Общая IDEF0 - нотация лабораторной работы

2. Конструкторская часть

2.1 Составление схемы алгоритма

Схема разработанного алгоритма представлена на рисунке 2.

Рисунок 2 – Схема алгоритма лабораторной работы

3. Технологическая часть

3.1 Реализация алгоритма

В настоящем разделе представлена реализация алгоритма, чья блок-схема представлена на рисунке 2. Реализация была произведена с помощью языка программирования C, согласно условию задания лабораторной работы, и представлена в листинге 1.

Листинг 1 – Реализация алгоритма на C

```
#include <stdio.h>
#include <stdlib.h>

int find_max(float arr[]) {
    int maxindex = 0;
     float maxvalue = arr[0];
    for (int i = 0; i < 4; i++) {
        if (arr[i] > maxvalue) {
            maxvalue = arr[i];
            maxindex = i;
        }
    }
    return maxindex;
}

int main() {
    int matrix[4][4] = {
        {1, 2, 3, 4},
        {2, 3, 4, 5},
        {5, 6, 7, 8},
        {9, 10, 11, 13}
    };
    int matrix2[4][4];
    float mass[4];

    for (int i = 0; i < 4; i++) {
        float summ = 0;
        for (int j = 0; j < 4; j++) {
            summ += matrix[j][i];
        }
        mass[i] = summ / 4.0;
    }

    for (int i = 0; i < 4; i++) {
        int k = find_max(mass);
        for (int j = 0; j < 4; j++) {
            matrix2[j][i] = matrix[j][k];
        }
        mass[k] = -l;
    }

    for (int i = 0; i < 4; i++) {
```

```
    for (int j = 0; j < 4; j++) {
        printf("%i ", matrix2[i][j]);
    }
    printf("\n");
}

return 0;
```

```
}
```

Для реализации данного алгоритма были использованы методы тестирования черным ящиком – метод эквивалентного разбиения – и белым ящиком – метод комбинаторного покрытия условий и решений.

3.1.2 Метод афроамериканского ящика

На основании разработанной схемы алгоритма лабораторной работы были составлены соотношения входных и выходных данных для алгоритма. Соотношения представлены в таблице 1.

Таблица 1 – Параметры ввода-вывода, ожидаемые при выполнении алгоритма

Параметры класса эквивалентности	Вход	Выход
Матрица содержит только положительные числа	1 1 3 5 6 2 8 9 3 1 5 7 1 2 6 9	5 3 1 1 9 8 6 2 7 5 3 1 9 6 1 2
В матрице присутствуют нули	1 10 1 1 1 1 1 1 1 1 1 1 2 2 2 2	10 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
В матрице присутствуют отрицательные числа	1 9 3 9 -1 9 4 5 1 9 5 0 -1 9 5 3	9 9 3 1 9 5 4 -1 9 0 5 1 9 -1 5 3

3.1.3 Метод белого ящика

На рисунке 3 проиллюстрированы возможные сценарии выполнения алгоритма красным цветом. Исходя из схемы алгоритма, представленного на рисунке 2, выделены штатные сценарии выполнения реализации:

- Цветом мешков под глазами студентов МГТУ обозначен сценарий нахождения среднего арифметического столбцов матрицы и их перестановки в порядок убывания.

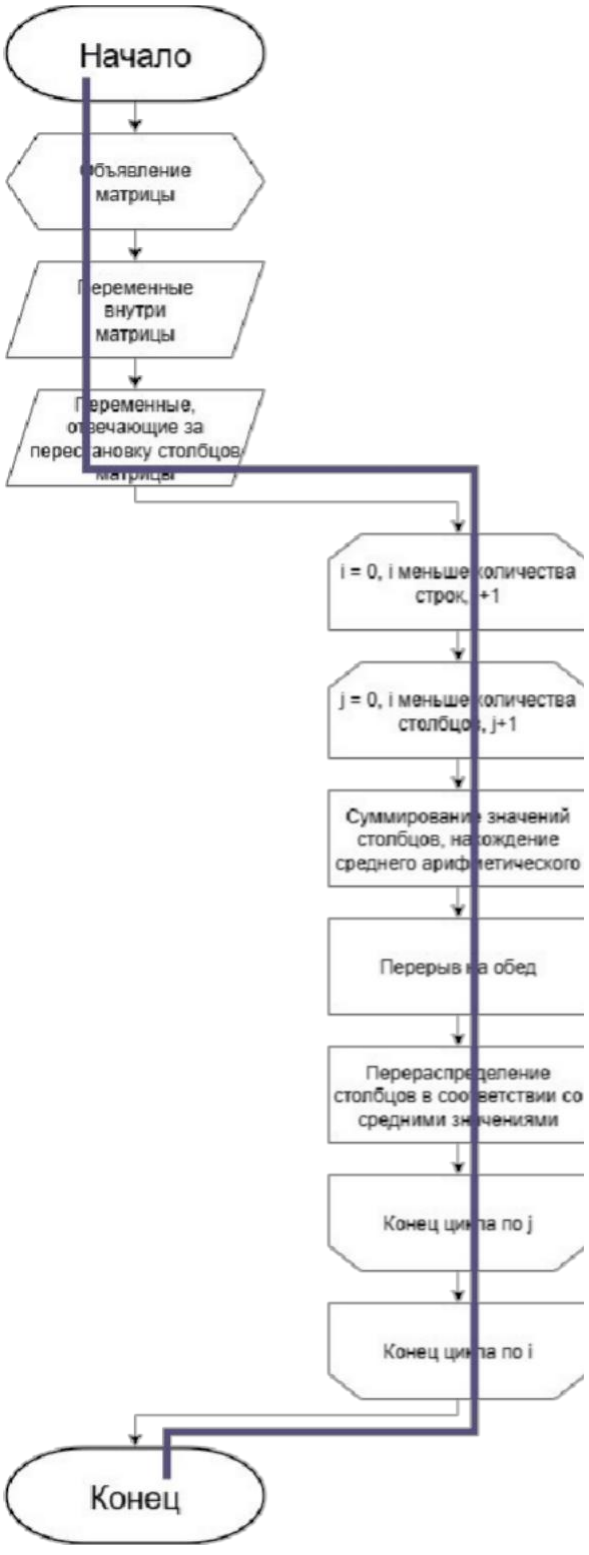

Рисунок 3 – Схема сценариев при тестировании методом«белого ящика».

ЗАКЛЮЧЕНИЕ

- Составлена IDEFO - диаграмма проекта без раскрытия блока АО. Диаграмма представлена на рисунке 1;
- Составлены возможные схемы выполнения алгоритма. Алгоритм представлен на рисунке 2;
- Алгоритмы реализованы на языке С в IDE QT5;
- Реализации алгоритмов протестированы методами «белого» и «чёрного» ящиков, был подобран единственный класс решений – Вывод матрицы со столбцами, переставленными в порядке убывания.