Министерство науки и высшего образования Российской Федерации

Федерального государственного бюджетного образовательного учреждения высшего образования

Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)

Московский техникум космического приборостроения

**ЛАБОРАТОРНАЯ РАБОТА №24**

**ТЕСТИРОВАНИЕ ИНТЕГРАЦИИ**

Специальность: 09.02.07 Информационные системы и программирование — квалификация «Программист»

Группа ТИП-53

|  |  |
| --- | --- |
| Проверил  Разработал | И.К. Широкова  В.О.Кузьминых |
|  |  |

Москва 2023

СОДЕРЖАНИЕ

1. Постановка задачи 3
2. Блок-схема алгоритма решения задачи 4
3. Код программы на С++ 5
4. Тесты для отладки 6
5. Контрольные вопросы 8
6. Вывод 9

1 Постановка задачи

**Цель занятия**: получить практические навыки отладки программ с помощью отладчика среды программирования.

**Оборудование, технические и программные средства:** персональный компьютер, среда программирования Visual Studio 2022, среда MS Visio. (или аналог)

**Задание:**

1. Составить в виде блок-схемы алгоритм решения задачи.
2. Создать программу решения задачи на любом алгоритмическом языке программирования.
3. Отладить программу с использованием инструментальных средств.
4. Составить отчет по лабораторной работе.

**Задача**: Дана вещественная матрица размерности n \* m. Вывести номера строк, содержащих больше положительных элементов, чем отрицательных.

2 Блок-схема алгоритма решения задачи

На рисунке 2.1 изображена блок-схема алгоритма решения задачи

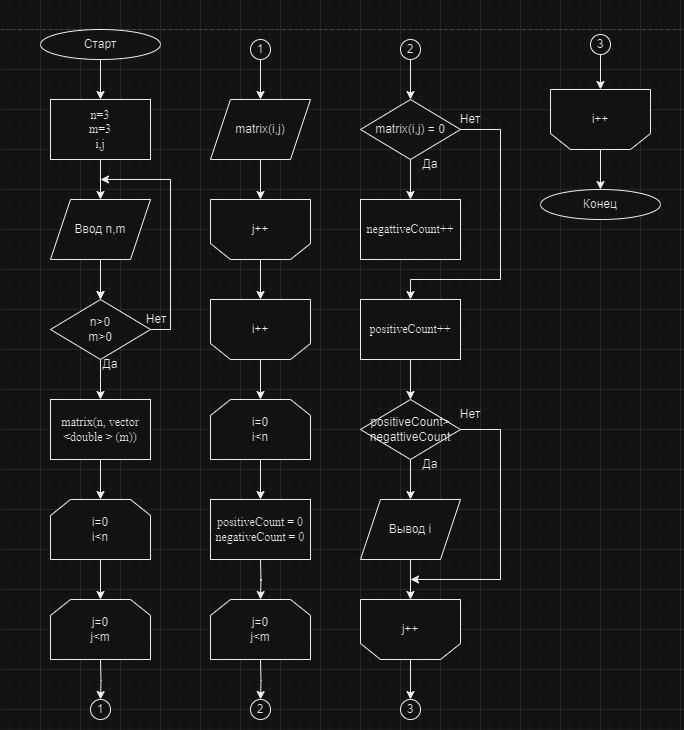


Рисунок 2.1 – Блок-схема решения задачи

3 Код программы на С++

Ниже приведен код программы для решения задачи на языке программирования Java и результат его выполнения (Рисунок 3.1)

#include <iostream>

#include <vector>

#include <iostream>

#include <vector>

using namespace std;

int main() {

setlocale(LC\_ALL, "Rus");

// Ввод размерностей матрицы

int n, m;

cout << "Введите количество строк: ";

cin >> n;

if (n <= 0) {

cout << "Количество строк не может быть 0!" << endl;

}

cout << "Введите количество столбцов: ";

cin >> m;

if (m <= 0) {

cout << "Количество столбцов не может быть 0!" << endl;

}

else {

// Ввод матрицы

vector<vector<double>> matrix(n, vector<double>(m));

cout << "Введите элементы матрицы:" << std::endl;

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

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

cout << "Элемент [" << i + 1 << "][" << j + 1 << "]: ";

cin >> matrix[i][j];

}

}

// Поиск строк с большим числом положительных элементов

cout << "Номера строк с большим числом положительных элементов:" << std::endl;

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

int positiveCount = 0;

int negativeCount = 0;

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

if (matrix[i][j] > 0) {

positiveCount++;

}

else if (matrix[i][j] < 0) {

negativeCount++;

}

}

if (positiveCount > negativeCount) {

cout << i + 1 << " ";

}

}

}

return 0;

}

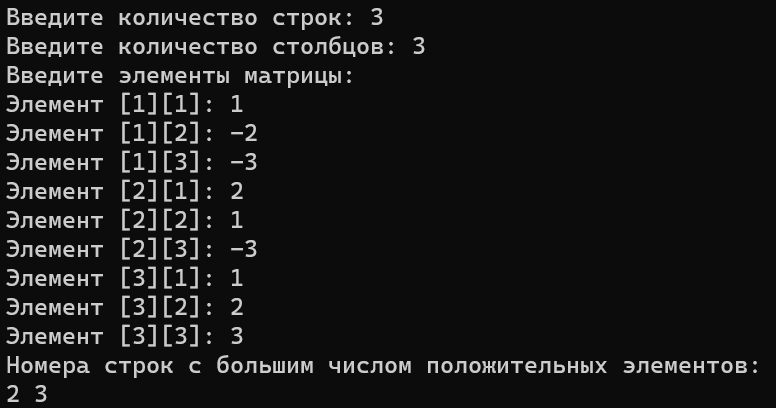


Рисунок 3.1 – Результат выполнения программы вывести номера строк, содержащих больше положительных элементов, чем отрицательных

4 Тесты для отладки

В программе присутствует проверки, которые не дают ввести отрицательное число при вводе кол-ва строк, и не дает ввести кол-во столбцов меньше 2

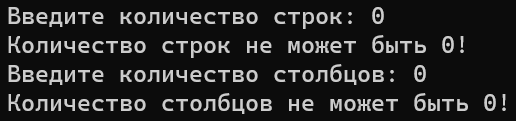


Рисунок 4.1 – Результат выполнения кода проверки ввода.

5 Контрольные вопросы

1. Что такое тестирование программы?

Тестирование программы — это процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом. Целью тестирования программного обеспечения является выявление дефектов и ошибок в программе, а также проверка соответствия программы требованиям. Тестирование программного обеспечения является неотъемлемой частью процесса разработки программного обеспечения.

1. Что такое отладка программы?

Отладка программы — это процесс обнаружения и устранения ошибок в

программе. Отладка программы является частью процесса тестирования

программного обеспечения.

1. Какие стадии тестирования выделяют при разработке программного обеспечения?

При разработке программного обеспечения выделяют следующие стадии

тестирования:

* Модельное тестирование - тестирование модели программы, созданной на ранних этапах разработки.
* Модульное тестирование - тестирование отдельных модулей программы.
* Интеграционное тестирование - тестирование взаимодействия модулей программы.
* Функциональное тестирование - тестирование функциональных возможностей программы.
* Нефункциональное тестирование - тестирование нефункциональных характеристик программы, таких как производительность, безопасность, масштабируемость, переносимость и т.д.
* Приемочное тестирование - тестирование программы заказчиком или конечным пользователем.

1. 4. Какие различают подходы в формировании тестовых наборов?

В формировании тестовых наборов различают следующие подходы:

Ручной подход - тестовые наборы формируются вручную, на основе опыта и знаний тестировщика.

Автоматический подход - тестовые наборы формируются автоматически, с помощью специальных инструментов.

1. В чем суть тестирования методом ―покрытия операторов?

Тестирование методом покрытия операторов заключается в том, что необходимо выполнить все операторы программы по крайней мере один раз. Это обеспечивает проверку того, что все операторы программы работают правильно.

1. В чем суть тестирования методом ―покрытия решений?

Тестирование методом покрытия решений заключается в том, что необходимо проверить все возможные решения всех ветвлений программы. Это обеспечивает проверку того, что все ветвления программы работают правильно.

1. В чем суть тестирования методом ―покрытия условий?

Тестирование методом покрытия условий заключается в том, что необходимо проверить все возможные значения всех условий программы. Это обеспечивает проверку того, что все условия программы работают правильно.

1. В чем суть тестирования методом ―комбинаторного покрытия условий?

Тестирование методом комбинаторного покрытия условий заключается в том, что необходимо проверить все возможные комбинации значений всех условий программы. Это обеспечивает проверку того, что все комбинации условий программы работают правильно.

1. В чем суть метода эквивалентных разбиений?

Метод эквивалентных разбиений заключается в том, что программа разделяется на эквивалентные множества, каждое из которых имеет одинаковое поведение. Для каждого эквивалентного множества формируется тестовый набор, который обеспечивает проверку работы этого множества.

1. В чем суть метода анализа граничных значений?

Метод анализа граничных значений заключается в том, что особое внимание уделяется значениям, находящимся на границах допустимых значений. Такие значения наиболее подвержены ошибкам.

1. В чем суть метода анализа причинно-следственных связей?

Метод анализа причинно-следственных связей заключается в том, что

выявляются причинно-следственные связи между различными частями

программы. Для каждой причинно-следственной связи формируется

тестовый набор, который обеспечивает проверку работы этой связи.

6 Вывод

Входе данной работы была написана программа, тесты для неё. Тесты проверяли различные случаи, включая пустые матрицы, матрицы с одной строкой, матрицы с множеством строк. Все тесты прошли успешно, что говорит о том, что программа работает корректно.