Лабораторная работа по теме «Тестирование интеграции»

Цель работы. Получить практические навыки отладки программ с помощью отладчикасреды программирования.

Отчёт по алгоритму решения задачи:

- **Ввод данных:**
  - Ввести размеры первого массива: количество строк N1 и столбцов M1.
  - Ввести размеры второго массива: количество строк N2 и столбцов M2.
  - Ввести элементы первого массива.
  - Ввести элементы второго массива.
- **Обработка массива 1:**
  - Для каждой строки массива 1:
    1) Посчитать сумму элементов на нечетных позициях (1, 3, 5 и т.д.)
    2) Если сумма положительна, увеличить счетчик подходящих строк т.е. count1.
- **Обработка массива 2:**
  - Для каждой строки массива 2:
    1) Посчитать сумму элементов на нечетных позициях.
    2) Если сумма положительна, увеличить счетчик подходящих строк count2.
- **Вывод результата:**
  - Если count1 и count2 равны нулю, вывести сообщение, что подходящих строк нет.
  - Иначе — вывести количество подходящих строк для каждого массива.

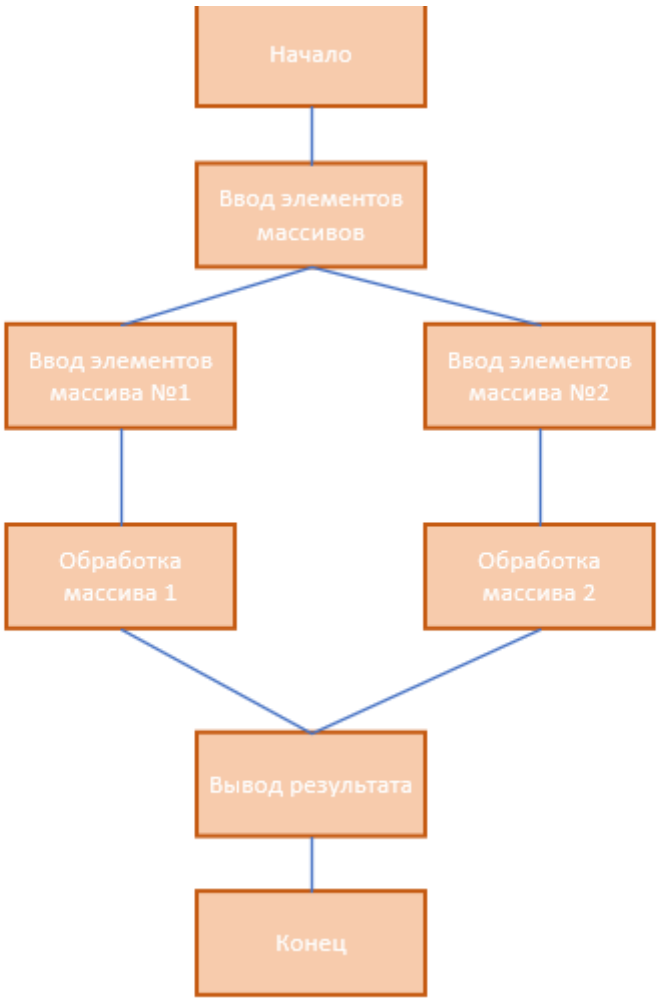

Рисунок 1 – Блок схема

Текст программы:

```
#include <iostream>
#include <vector>
using namespace std;
bool checkRow(const vector<int>& row) {
    int sum = 0;
    for (size_t i = 0; i < row.size(); i += 2) {
        sum += row[i];
    }
    return sum > 0;
}
```

```
int main() {
    setlocale(LC_ALL, "Russian");
    int N1, M1, N2, M2;
    cout << "Введите количество строк первого массива (не более 10): ";
    cin >> N1;
    cout << "Введите количество столбцов первого массива (не более 10): ";
    cin >> M1;
    cout << "Введите количество строк второго массива (не более 10): ";
    cin >> N2;
    cout << "Введите количество столбцов второго массива (не более 10): ";
    cin >> M2;
    if (N1 > 10 || M1 > 10 || N2 > 10 || M2 > 10) {
        cout << "Размер массива превышает 10. Программа завершена." << endl;
        return 1;
    }
    vector< vector<int> > array1(N1, vector<int>(M1));
    vector< vector<int> > array2(N2, vector<int>(M2));
    // Ввод массива 1
    cout << "Введите элементы первого массива:" << endl;
    for (int i = 0; i < N1; ++i) {
        cout << "Строка " << i + 1 << ": ";
        for (int j = 0; j < M1; ++j) {
            cin >> array1[i][j];
        }
    }
    // Ввод массива 2
    cout << "Введите элементы второго массива:" << endl;
```

```
    for (int i = 0; i < N2; ++i) {
        cout << "Строка " << i + 1 << ": ";
        for (int j = 0; j < M2; ++j) {
            cin >> array2[i][j];
        }
    }
    int count1 = 0, count2 = 0;
    // Подсчет подходящих строк
    for (const auto& row : array1) {
        if (checkRow(row))
            ++count1;
    }
    // Подсчет подходящих строк
    for (const auto& row : array2) {
        if (checkRow(row))
            ++count2;
    }
    if (count1 == 0 && count2 == 0) {
        cout << "Нет ни для одного массива строк, удовлетворяющих условию."
<< endl;
    }
    else {
        if (count1 > 0)
            cout << "Количество строк первого массива, где сумма элементов на
нечетных позициях положительна: " << count1 << endl;
        if (count2 > 0)
            cout << "Количество строк второго массива, где сумма элементов на
нечетных позициях положительна: " << count2 << endl;
```

```
    }
    return 0;
}
```

Контрольные вопросы

1. Тестирование программы — процесс проверки работоспособности и корректности программы для выявления ошибок.

2. Отладка — процесс поиска и исправления ошибок, обнаруженных в ходе тестирования.

3. Стадии тестирования: планирование, разработка тестов, выполнение тестов, анализ результатов, исправление ошибок.

4. Подходы к формированию тестовых наборов: случайный, структурированный, основанный на покрытиях, аналитический.

5. Покрытие операторов — тесты, охватывающие выполнение каждого оператора хотя бы один раз.

6. Покрытие решений — тесты, охватывающие все возможные ветви решений в коде.

7. Покрытие условий — тесты, проверяющие все логические условия и их комбинации.

8. Комбинаторное покрытие условий — создание тестов, охватывающих все возможные комбинации условий.

9. Метод эквивалентных разбиений — разделение входных данных на классы эквивалентности, для тестирования по представителям.

10. Метод анализа граничных значений — тестирование на границах диапазонов входных данных.

11. Метод анализа причинно-следственных связей — тестирование на основе анализа причин и следствий в программе.