«Отчет по ПЗ 1.2. Динамические массивы (матрицы)» Никита Фролов \*\*\*. Вариант № 22.

Задание: вывести номера столбцов, произведение элементов которых меньше единицы.

1. Постановка задачи:

Входные данные + ограничения:

А) Переменная n. Тип int. Размерность матрицы (количество строк).

Исключения: буква, буква с цифрой, строка, вещественное, меньше или равно 0, большие числа (зависит от наличия свободной оперативной памяти компьютера).

Б) Переменная m. Тип int. Размерность матрицы (количество столбцов).

Исключения: буква, буква с цифрой, строка, вещественное, меньше или равно 0, большие числа (зависит от наличия свободной оперативной памяти компьютера).

В) Переменная matrix[][]. Тип \*\*double. Элементы матрицы.

Исключения: буква, буква с цифрой, строка.

Выходные данные + ограничения:

А) Переменная matrix[][]. Тип \*\*double. Элементы матрицы.

Исключения: буква, буква с цифрой, строка.

Б) Переменная w. Тип int. Временная переменная для отображения столбца. Ограничения: буква, буква с цифрой, строка, вещественное, меньше или равно 0, большие числа (зависит от наличия свободной оперативной памяти компьютера).

В) Переменная ch. Тип char. Промежуточная переменная для проверки на отсутствие буквенных значений в размерах массивов

Исключения: нет.

Промежуточные данные + ограничения:

А) Переменная key. Тип bool. Флаг переменная для сигнализирования наличия ошибок.

Исключения: буква, буква с цифрой, строка, вещественное число, числа не равные false или true.

Б) Переменная tmp. Тип double. Промежуточная переменная для проверки на буквенное значение размера массива.

Исключения: буква, буква с цифрой, строка, значение меньше 1.

В) Переменная ch. Тип char. Промежуточная переменная для проверки на отсутствие буквенных значений в размерах массивов

Исключения: нет.

Г) Переменная mult. Тип double. Промежуточная переменная произведения элементов столбца.

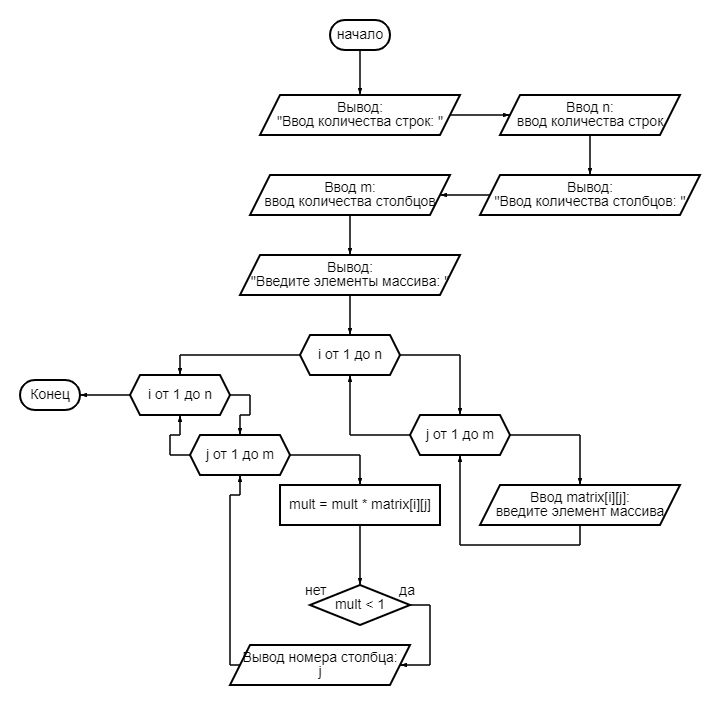
Исключения: буква, буква с цифрой, строка.

Среда разработки: Visual Studio 2022.

2. Разработка программы.

2.1 Структура программы:

2.2 Разработка схемы алгоритма:



* 1. Описание пользовательского интерфейса:

Взаимодействие пользователя с программой осуществляется с помощью диалога.

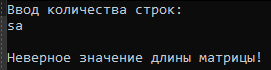
Вводимые данные диалога:

1. Ввод количества строк массива:

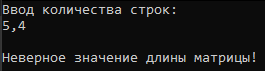
Вводимые значения – 5



Вводимые значения – “sa”



Вводимые значения – 5,4

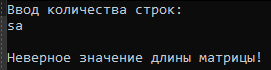


1. Ввод количества столбцов массива:

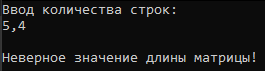
Вводимые значения – 5



Вводимые значения – “sa”



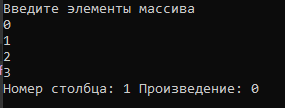
Вводимые значения – 5,4



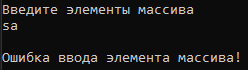
1. Ввод элементов массива:

Вводимые значения –

0 1 2 3



Вводимые значения – “sa”



1. Реализация и тестирование программ:
   1. Описание разработанной программы

Вся программа состоит из одной функции main. Дополнительных функций – нет. Для начала чрез проверки мы считываем размерность массива – количество строк. Потом также чрез проверки мы считываем вторую размерность массива – количество столбцов. Если введённые данные прошли все проверки, то мы переходим к вводу элементов массива. Каждый элемент также проверяется на корректность ввода. После чего идёт обработка массива и расчёт произведений элементов столбцов. В конце программы происходит вывод номеров столбцов, в которых произведение элементов меньше единицы. Каждая часть программы (кроме первой) выполняется при условии, что на предыдущих шагах не было ошибок.

* 1. Тестирование программы

|  |  |  |
| --- | --- | --- |
| Входные данные | Результат программы | Смысл |
| n = a | Неверное значение длины матрицы! | Проверка на букву |
| n = 3a | Неверное значение длины матрицы! | Проверка на число c буквой |
| n = 3,2 | Неверное значение длины матрицы! | Проверка на целое число |
| n = 10000000000000000 | Неверное значение длины матрицы! | Выделение памяти массивам |
| n = 0 | Неверное значение длины матрицы! | Валидность размера массива |
| n = zxcgul | Неверное значение длины матрицы! | Проверка на строку |
| n =2 | 2 | Проверка корректности работы программы |
| m = b | Неверное значение длины матрицы! | Проверка на букву |
| m = 4a | Неверное значение длины матрицы! | Проверка на число c буквой |
| m = 3,2 | Неверное значение длины матрицы! | Проверка на целое число |
| m = 10000000000000000 | Неверное значение длины матрицы! | Выделение памяти массивам |
| m = 0 | Неверное значение длины матрицы! | Валидность размера массива |
| m = zxcgul | Неверное значение длины матрицы! | Проверка на строку |
| m = 5 | 5 | Проверка корректности работы программы |
| matrix[i][j] = a | Неверное значение элемента матрицы! | Проверка на букву |
| matrix[i][j] = 666rain | Неверное значение элемента матрицы! | Проверка на число c буквой |
| matrix[i][j] = asadsa | Неверное значение элемента матрицы! | Проверка на строку |
| matrix[i][j] = -2,6 | -2,6 | Проверка корректности работы программы |
| n = 2  m = 2  matrix[0][0] = 0  matrix[0][1] = 0  matrix[1][0] = 0  matrix[1][1] = 0 | Элементы вашей матрицы:  Элемент матрицы [0][0] = 0 Элемент матрицы [0][1] = 0  Элемент матрицы [1][0] = 0 Элемент матрицы [1][1] = 0  Количество подходящих столбцов: 2  Номер столбца: 1 Произведение: 0  Номер столбца: 2 Произведение: 0 | Проверка на корректность работы программы при отсутсвтии положительных и отрицательных чисел |
| n = 2  m = 2  matrix[0][0] = 1  matrix[0][1] = 1  matrix[1][0] = 1  matrix[1][1] = 1 | Элементы вашей матрицы:  Элемент матрицы [0][0] = 1 Элемент матрицы [0][1] = 1  Количество подходящих столбцов: 2 | Проверка на корректность работы программы cо всеми положительными столбцами |
| n = 2  m = 2  matrix[0][0] = 0  matrix[0][1] = 1  matrix[1][0] = 0  matrix[1][1] = 1 | Элементы вашей матрицы:  Элемент матрицы [0][0] = 0  Элемент матрицы [0][1] = 1  Элемент матрицы [1][0] = 0  Элемент матрицы [1][1] = 1  Количество подходящих столбцов: 1  Номер столбца: 1 Произведение: 0 | Проверка на корректность работы программы c положительными столбцами и с отрицательными |

Код:

#define \_CRT\_SECURE\_NO\_WARNINGS // Преопределение ошибок для scanf

#include <iostream> // Подключение стандартной библиотеки

using namespace std; // Использование стандартного пространства имён

int main(void) // Главная функция - main

{

setlocale(LC\_ALL, "Russian"); // Установка русского языка в консоль

bool key = true; // Переменная-переключатель во избежания ошибок

int m = 0, n = 0; // Объявление количества столбцов и строк матрицы

double tmp = 0; // Временная переменная для проверок

double mult = 1; // Теоретически подходящее произведение

char ch = 't'; // Временная перемення для проверок

double\*\* matrix; // Двумерный массив

int w = 0; // Переменная для вывода столбца

cout << "Ввод количества строк: " << endl;

if ((scanf("%lf%c", &tmp, &ch) == 2) && (ch == '\n') && (tmp == (int)tmp) && (tmp > 0)) // Проверка размерности матрицы

{

n = tmp;

}

else

{

cout << endl << "Неверное значение длины матрицы!" << endl;

key = false;

}

if (key == true)

{

cout << "Ввод колличества столбцов: " << endl;

if ((scanf("%lf%c", &tmp, &ch) == 2) && (ch == '\n') && (tmp == (int)tmp) && (tmp > 0)) // Проверка размерности матрицы

{

m = tmp;

}

else

{

cout << endl << "Неверное значение длины матрицы!" << endl;

key = false;

}

}

if (key == true) // Выделение памяти под массив

{

matrix = new double\* [n]; // Объявление матрицы (столбцов)

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

{

matrix[i] = new double[m]; // Объявление матрицы (строк)

}

cout << "Введите элементы массива" << endl;

for (int i = 0; i < n; i++) // Ввод элементов массива

{

if (key == false)

{

break;

}

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

{

if ((scanf("%lf%c", &tmp, &ch) == 2) && (ch == '\n')) // Проверка значения элемента матрицы на буквенное значение

{

matrix[i][j] = tmp;

}

else

{

cout << endl << "Ошибка ввода элемента массива!" << endl;

key = false;

break;

}

}

}

if (key == true) // Обработка массива

{

for (int i = 0; i < m; ++i, ++w)

{

for (int j = 0; j < n; ++j)

{

mult \*= matrix[j][i]; // Изменение значения произведения

}

if (mult < 1) // Проверяем на условие задачи

{

cout << "Номер столбца: " << w + 1 << " " << "Произведение: " << mult << endl;

}

mult = 1;

}

}

for (int i = 0; i < n; i++) // Очистка заненной памяти

{

delete[] matrix[i]; // Очистка памяти от строк матрицы

}

delete[] matrix; // Очистка остальной памяти

}

return 0;

}