

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**"МИРЭА - Российский технологический университет"**

**РТУ МИРЭА**

Институт радиоэлектроники и автоматики

Кафедра геоинформационных систем

**ОТЧЕТ**

**ПОПРАКТИЧЕСКОЙРАБОТЕ№ 12**

*Элементы алгоритмизации и процедурного программирования*

**по дисциплине**

**«**ИНФОРМАТИКА**»**

|  |  |
| --- | --- |
| Выполнил студент группы *ИКБО-10-23* | *Враженко Д.О.* |

|  |  |
| --- | --- |
| Принял  *доцент кафедры ГИС, к.т.н.* | *Воронов Г.Б.* |

|  |  |  |
| --- | --- | --- |
| Практическая  работа выполнена | «\_\_»\_\_\_\_\_\_\_\_\_\_\_2023 г. | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* |

|  |  |  |
| --- | --- | --- |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_\_\_\_\_2023 г. | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* |

Москва 2023

СОДЕРЖАНИЕ

[1 ПОСТАНОВКА ЗАДАЧИ 3](#__RefHeading___Toc4052_795040542)

[2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ 4](#__RefHeading___Toc219_4179944220)

[2.1 Функция main() 4](#__RefHeading___Toc221_4179944220)

[2.2 Функция length\_of\_matrix() 4](#__RefHeading___Toc2038_2144581395)

[2.3 Функция answer\_to\_question() 5](#__RefHeading___Toc2040_2144581395)

[2.4 Функция matrix\_from\_user(int\* matrix, int M) 6](#__RefHeading___Toc2042_2144581395)

[2.5 Функция matrix\_from\_random(int\* matrix, int M) 7](#__RefHeading___Toc2044_2144581395)

[2.6 Функция cout\_matrix(int\* matrix, int M) 7](#__RefHeading___Toc2046_2144581395)

[2.7 Функция matrix\_sort(int\* matrix, int M) 8](#__RefHeading___Toc2048_2144581395)

[2.8 Структурированный код программы с комментариями 9](#__RefHeading___Toc1232_847701130)

[2.9 Примеры тестирования 18](#__RefHeading___Toc1188_1364753697)

[3 ВЫВОДЫ 21](#__RefHeading___Toc4056_795040542)

[4 ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ 22](#__RefHeading___Toc4058_795040542)

# ПОСТАНОВКА ЗАДАЧИ

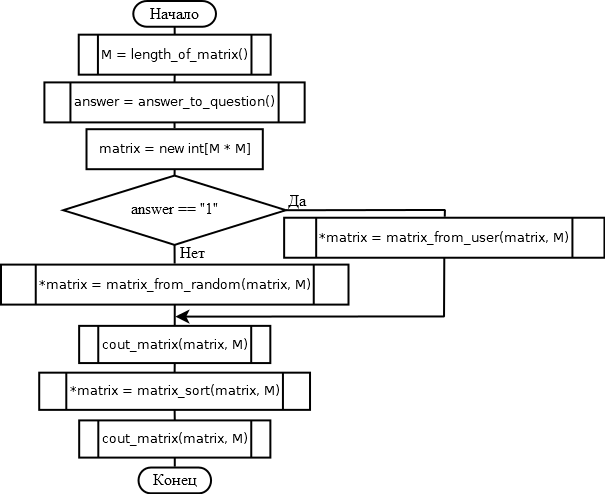
Требуется разработать блок-схему алгоритма и написать программу обра­ботки данных в соответствии с выбранным и согласованным с преподавателем вариантом. При этом требуется контролировать типы и диапазоны вводимых данных, а также предусмотреть обработку других исключительных ситуаций (если они есть), например, ситуацию деления на ноль. Блок-схема должна быть полной, т.е. должна описывать и процесс диалога с пользователем, и контроль вводимых данных, и подпрограммы вычислений с обработкой возможных ис­ключительных операций. Блок-схема должна изображаться по ГОСТу. При об­наружении ошибки ввода или ошибки вычислений программа должна информа­тивно уведомлять пользователя о причине ошибки. Если ошибка произошла на этапе ввода данных, то программа должна просить пользователя повторить ввод.

Личный вариант: 2.7. Создать квадратную матрицу размера M×M, где M является целым числом из диапазона [2, 5]. Конкретный размер матрицы задает­ся пользователем. Матрица содержит только целые числа из диапазона [1, 100], которые могут быть как случайными, так и вводиться пользователем. Отсорти­ровать по убыванию элементы, принадлежащие или лежащие ниже главной диа­гонали матрицы, остальные элементы умножить на минус один. Результаты об­работки матрицы вывести на экран.

# ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ

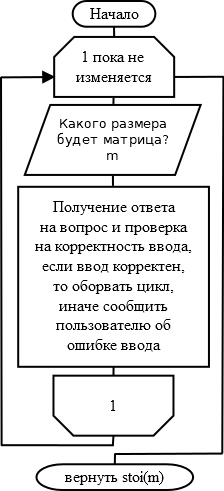
## Функция main()

Данная функция обязательна в программе. На рис. 1 представлена её блок-схема.

Рисунок 1 – Блок-схема функции main()

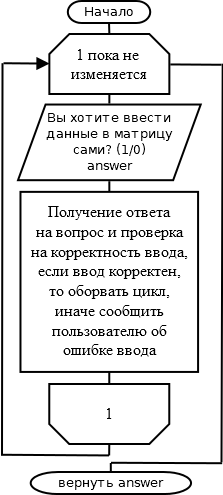
## Функция length\_of\_matrix()

Данная функция узнаёт от пользователя размер матрицы и проверяет ввод на корректность. На рис. 2 представлена блок-схема этой функции.

Рисунок 2 – Блок-схема функции length\_of\_matrix()

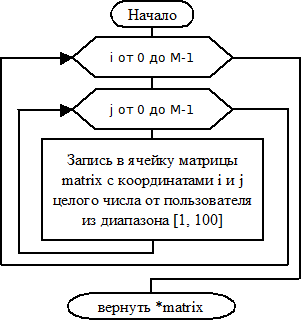
## Функция answer\_to\_question()

Данная функция узнаёт от пользователя, хочет он вводить данные в мат­рицу сам или нет, и проверяет ввод на корректность. На рис. 3 представлена блок-схема этой функции.

Рисунок 3 – Блок-схема функции answer\_to\_question()

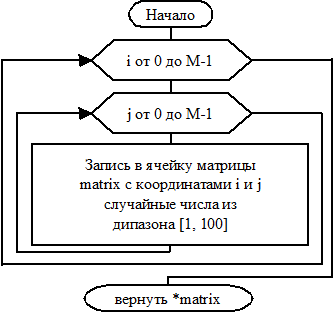
## Функция matrix\_from\_user(int\* matrix, int M)

Данная функция позволяет пользователю заполнять матрицу вручную, а также проверяет ввод на корректность. На рис. 4 представлена блок-схема этой функции.

Рисунок 4 – Блок-схема функции matrix\_from\_user(int\* matrix, int M)

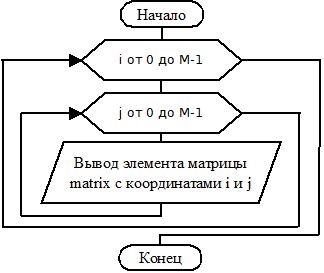
## Функция matrix\_from\_random(int\* matrix, int M)

Данная функция заполняет матрицу случайными числами из диапазона [1, 100]. На рис. 5 представлена блок-схема этой функции.

Рисунок 5 – Блок-схема функции matrix\_from\_random(int\* matrix, int M)

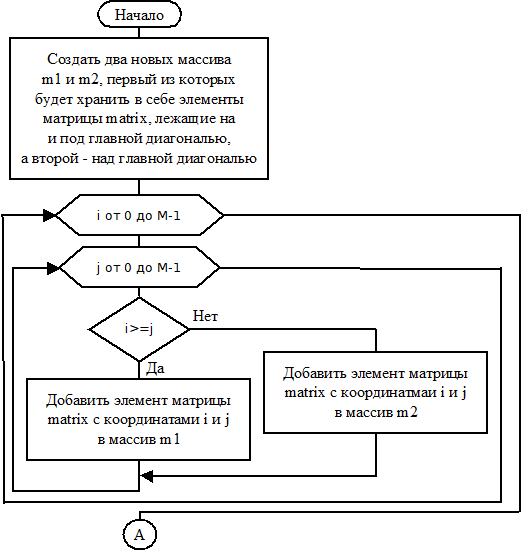
## Функция cout\_matrix(int\* matrix, int M)

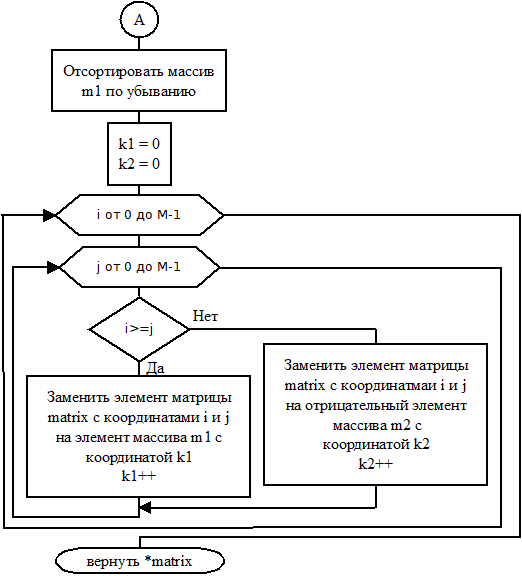
Данная функция выводит матрицу на экран поэлементно. На рис. 6 представлена блок-схема этой функции.

Рисунок 6 – Блок-схема функции cout\_matrix(int\* matrix, int M)

## Функция matrix\_sort(int\* matrix, int M)

Данная функция сортирует матрицу, как написано в условии. На рис. 7 и рис. 8 представлена блок-схема этой функции.

Рисунок 7 – Первая часть блок-схемы функции matrix\_sort(int\* matrix, int M)

Рисунок 8 – Вторая часть блок-схемы функции matrix\_sort(int\* matrix, int M)

## Структурированный код программы с комментариями

#include <iostream> // Подключение библиотеки iostream

#include <string> // Подключение библиотеки string

#include <windows.h> // Подключение библиотеки windows.h

#include <cstdlib> // Подключение библиотеки cstdlib

using namespace std; // Объявление пространства имём std

int length\_of\_matrix() { // Объявление функции length\_of\_matrix

string m; // Создание переменной m

while (true) { // Запуск цикла, который будет работать до тех пор, пока пользователь не введет целое число из диапазона [2, 5]

cout << "Введите целое значение M (количество строк и столбцов матрицы) в диапазоне [2, 5]\n"; // Вывод предложения в кавычках на экран

cout.width(8); // Размер следующего выводимого сообщения (8 символов)

cout << "Ввод: "; // Вывод слова в кавычках на экран, засчёт предыдущей команды перед словом будет отступ в 2 символа

cin >> m; // Получение переменной m от пользователя

if (m == "2" || m == "3" || m == "4" || m == "5") // Если переменная m равна одному из значений (2, 3, 4 или 5), то программа идёт по этому пути

break; // Завершение работы цикла

else { // Иначе

cout << "Допущена ошибка! Вы должны вводить только целые числа из диапазона [2, 5]! Повторите попытку ввода!\n"; // Вывод предложения в кавычках на экран

m.clear(); // Очищение значения переменной m

}

}

return stoi(m); // Возвращение целочисленного значения длины стороны квадратной матрицы

}

string answer\_to\_question() { // Объявление функции answer\_of\_question

string answer; // Создание переменной answer

while (true) { // Запуск цикла, который будет работать до тех пор, пока пользователь не ответит на вопрос корректно

cout << "Вы хотите ввести данные в матрицу самостоятельно? (1/0)\n"; // Вывод предложения в кавычках на экран

cout.width(8); // Размер следующего выводимого сообщения (8 символов)

cout << "Ввод: "; // Вывод слова в кавычках на экран, засчёт предыдущей команды перед словом будет отступ в 2 символа

cin >> answer; // Получение переменной answer от пользователя

if (answer == "1" || answer == "0") // Если переменная answer равна одному из значений (0 или 1), то программа идёт по этому пути

break; // Завершение работы цикла

else { // Иначе

cout << "Допущена ошибка! Вы должны ответить на вопрос только одним символом (1/0)! Повторите попытку ввода!\n"; // Вывод предложения в кавычках на экран

answer.clear(); // Очищение значения переменной answer

}

}

return answer; // Возвращение ответа на вопрос

}

int matrix\_from\_user(int \*matrix, int M) { // Объявление функции matrix\_from\_user

cout << "Далее Вам требуется вводить значения элементов в диапазоне [1, 100] (к примеру: A[1][0] = 5)\n"; // Вывод предложения в кавычках на экран

for (int i = 0; i < M; i++) { // Цикл for для строк матрицы с шагом 1

for (int j = 0; j < M; j++) { // Цикл for для столбцов матрицы с шагом 1

string X; // Создание переменной X, в которую программа будет записывать полученный результат от пользователя

bool f = false; // Создание переменной f, которая будет отвечать за работу цикла, и присваивание ей значения false

while (!f) { // Пока значение f равно false, будет работать цикл

cout.width(5); // Размер следующего выводимого сообщения (5 символов)

cout << "A[" << i << "][" << j << "] = "; // Вывод символов в кавычках на экран и индексов элемента матрицы, засчёт предыдущей команды перед буквой A будет отступ в 3 символа

cin >> X; // Получение переменной X от пользователя

int l = X.length(); // Создание переменной l и присваивание ей длины строки X

bool p = true; // Создание переменной p, которая будет отвечать за тип вводимого значения (целое число или нет), и присваивание ей значения true (подразумевается, что пользователь ввёл целое число)

for (int k = 0; k < l; k++) { // Цикл for для всех сиволов строки X с шагом 1

if (X[0] == '0') {

cout << "Допущена ошибка! Вы должны вводить только целые числа из диапазона [1, 100]! Повторите попытку ввода!\n"; // Вывод предложения в кавычках на экран

X.clear(); // Очищение значения переменной X

p = false; // Присваивание переменной p значения false

break; // Завершение работы цикла

}

if (!isdigit(X[k])) { // Проверка на то, является ли элемент строки не цифрой

cout << "Допущена ошибка! Вы должны вводить только целые числа из диапазона [1, 100]! Повторите попытку ввода!\n"; // Вывод предложения в кавычках на экран

X.clear(); // Очищение значения переменной X

p = false; // Присваивание переменной p значения false

break; // Завершение работы цикла

}

}

if (p) { // Если переменная p не изменилась, то будет работать код ниже

int x = stoi(X); // Создание переменной x и присваивание ей целочисленного значения X с помощью функции перевода строки в челое число

if (x >= 1 && x <= 100) { // Если число находится в диапазоне [1, 100], то программа пойдет по этому пути

\*(matrix + i \* M + j) = x; // Присваивание элементу матрицы значения x

f = true; // Присваивание переменной f значения true

}

else { // Иначе

cout << "Допущена ошибка! Вы должны вводить только целые числа из диапазона [1, 100]! Повторите попытку ввода!\n"; // Вывод предложения в кавычках на экран

X.clear(); // Очищение значения переменной X

}

}

}

}

}

return \*matrix; // Возвращение matrix

}

int matrix\_from\_random(int\* matrix, int M) { // Объявление функции matrix\_from\_random

srand(time(0)); // Команда, отвечающая за случайную генерацию чисел при каждом запуске программы

cout << "Массив будет заполнен случайными целыми числами в диапазоне [1, 100].\n"; // Вывод предложения в кавычках на экран

for (int i = 0; i < M; i++) { // Цикл for для строк матрицы с шагом 1

for (int j = 0; j < M; j++) // Цикл for для столюцов матрицы с шагом 1

\*(matrix + i \* M + j) = rand() % 100 + 1; // Генерация случайных чисел для элементов матрицы из диапазона [1, 100]

}

return \*matrix; // Возвращение matrix

}

void cout\_matrix(int\* matrix, int M) { // Объявление функции cout\_matrix

for (int i = 0; i < M; i++) { // Цикл for для строк матрицы с шагом 1

for (int j = 0; j < M; j++) { // Цикл for для столбцов матрицы с шагом 1

cout.width(5); // Размер следующего выводимого сообщения (5 символов)

cout << \*(matrix + i \* M + j); // Вывод на экран элемента матрицы

}

cout << endl; // Переход на новую строку

}

}

int matrix\_sort(int\* matrix, int M) { // Объявление функции matrix\_sort

int k1 = ((M \* M) - M) / 2 + M; // Создание переменной k1, которая является количеством элементов, лежащих на главной диагонали и под ней

int\* m1 = new int[k1]; // Создание одномерно массива m1 и выделение места для него

int k2 = (M \* M) - k1; // Создание переменной k2, которая является количеством элементов, лежащих над главной диагональю

int\* m2 = new int[k2]; // Создание одномерно массива m2 и выделение места для него

k1 = 0; // Обнуление k1

k2 = 0; // Обнуление k2

for (int i = 0; i < M; i++) { // Цикл for для строк матрицы с шагом 1

for (int j = 0; j < M; j++) { // Цикл for для столбцов матрицы с шагом 1

if (i >= j) { // Если элемент лежит на главной диагонали или под ней, то программа пойдет по этому пути

m1[k1] = \*(matrix + i \* M + j); // Присваивание элементу одномерного массива m1 значения элемента матрицы matrix

k1++; // Увеличение k1 на 1

}

else { // Иначе

m2[k2] = \*(matrix + i \* M + j); // Присваивание элементу одномерного массива m2 значения элемента матрицы matrix

k2++; // Увеличение k2 на 1

}

}

}

int swap; // Создание переменной swap, которая будет отвечать за промежуточное значение для обмена значениями при сортировке

for (int i = 0; i < k1; i++) { // Цикл for для первого элемента сравнения

for (int j = i + 1; j < k1 + 1; j++) { // Цикл for для второго элемента сравнения

if (m1[i] < m1[j]) { // Если первый элемент меньше второго, то они меняются местами

swap = m1[j]; // Присваивание переменной swap значения переменной m1[j]

m1[j] = m1[i]; // Присваивание переменной m1[j] значения переменной m1[i]

m1[i] = swap; // Присваивание переменной m1[i] значения переменной swap

}

}

}

k1 = 1; // Присваивание k1 значения 1

k2 = 0; // Присваивание k2 значения 0

for (int i = 0; i < M; i++) { // Цикл for для строк матрицы с шагом 1

for (int j = 0; j < M; j++) { // Цикл for для столбцов матрицы с шагом 1

if (i >= j) { // Если элемент лежит на главной диагонали или под ней, то программа пойдет по этому пути

\*(matrix + i \* M + j) = m1[k1]; // Присваивание элементу матрицы matrix значения элемента отсортированного одномерного массива m1

k1++; // Увеличение k1 на 1

}

else { // Иначе

\*(matrix + i \* M + j) = -m2[k2]; // Присваивание элементу матрицы matrix значения элемента одномерного массива m2, умноженного на -1

k2++; // Увеличение k2 на 1

}

}

}

delete[] m1; // Удаление одномерного массива m1

delete[] m2; // Удаление одномерного массива m2

return \*matrix; // Возвращение matrix

}

int main() // Объявление функции main

{

SetConsoleCP(CP\_UTF8); // Разрешение на использование русских символов в консоли

SetConsoleOutputCP(CP\_UTF8); // Разрешение на использование русских символов в консоли

int M = length\_of\_matrix(); // Создание переменной M, в которой будет храниться размер матрицы

string answer = answer\_to\_question(); // Создание переменной answer, в которую программа будет записывать полученный результат от пользователя

int\* matrix = new int[M \* M]; // Создание переменной matrix, которая является целочисленной квадратной матрицы, и выделение места для неё

if (answer == "1") // Если переменная answer равна 1, то программа пойдёт по этому пути

\*matrix = matrix\_from\_user(matrix, M); // Матрица заполняется пользователем

else // Иначе

\*matrix = matrix\_from\_random(matrix, M); // Матрица заполняется случайными числами

cout << "Полученная матрица:\n"; // Вывод предложения в кавычках на экран

cout\_matrix(matrix, M); // Вывод матрицы на экран

\*matrix = matrix\_sort(matrix, M); // Сортировка матрицы по условию

cout << "Итоговая матрица:\n"; // Вывод предложения в кавычках на экран

cout\_matrix(matrix, M); // Вывод матрицы на экран

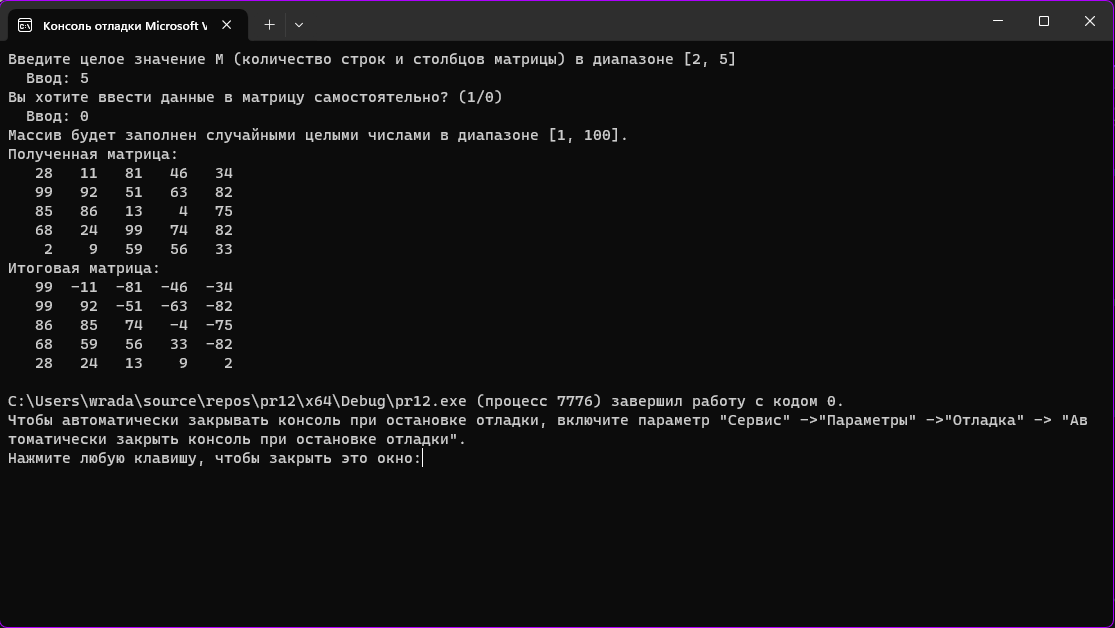
delete[] matrix; // Удаление матрицы matrix

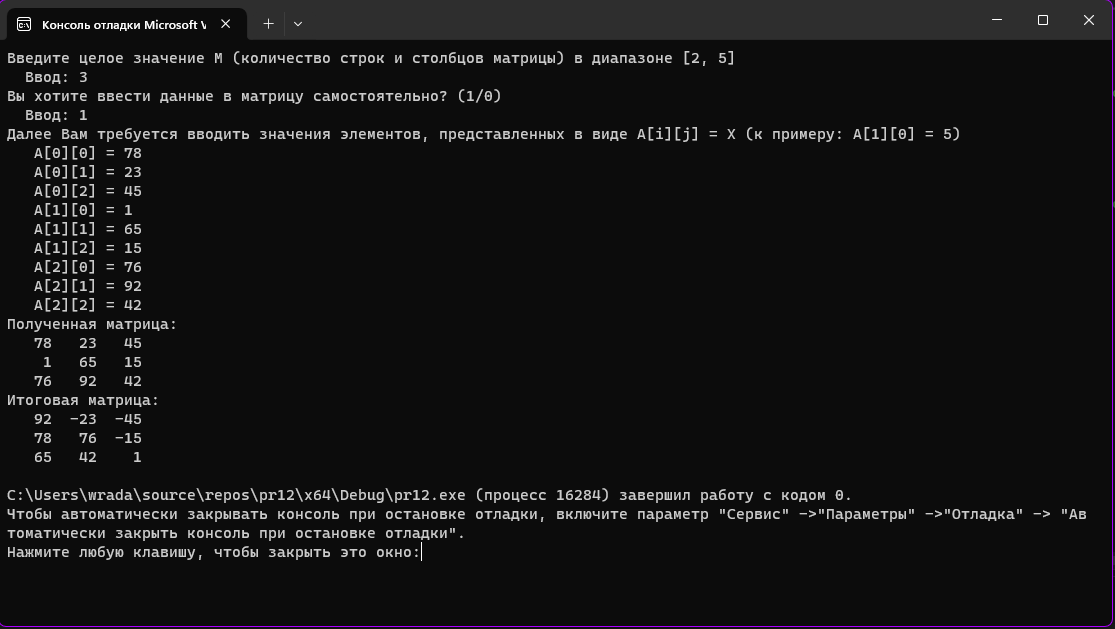
return 0; // Завершение работы программы

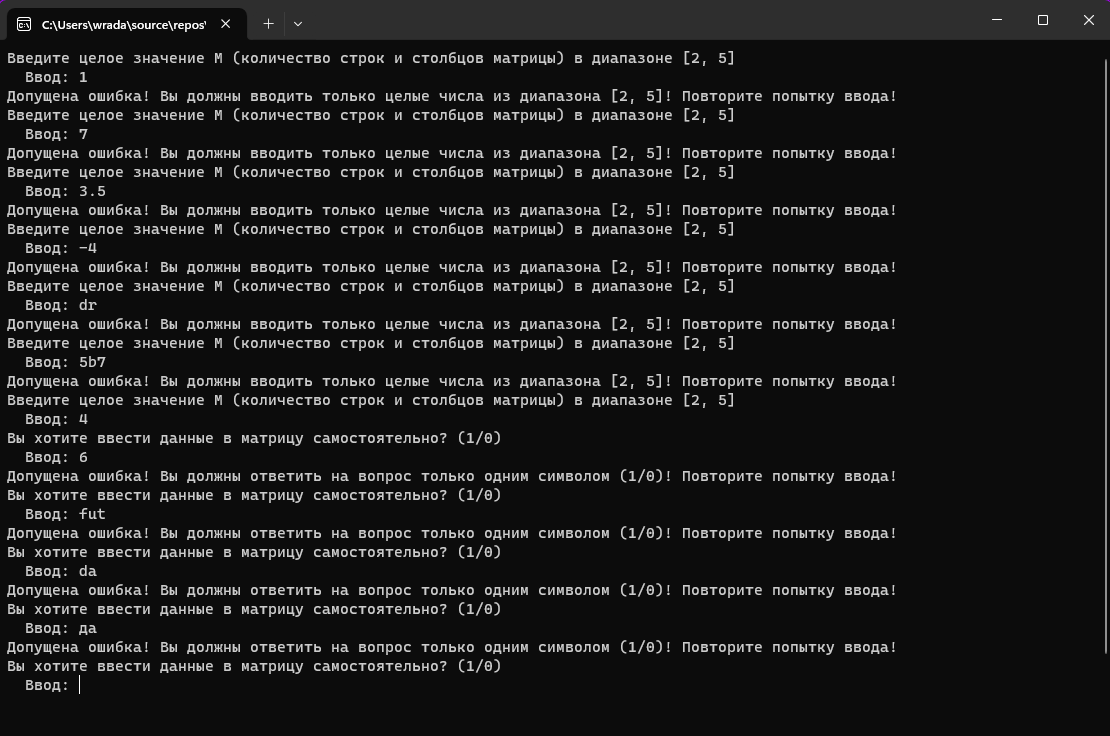
}

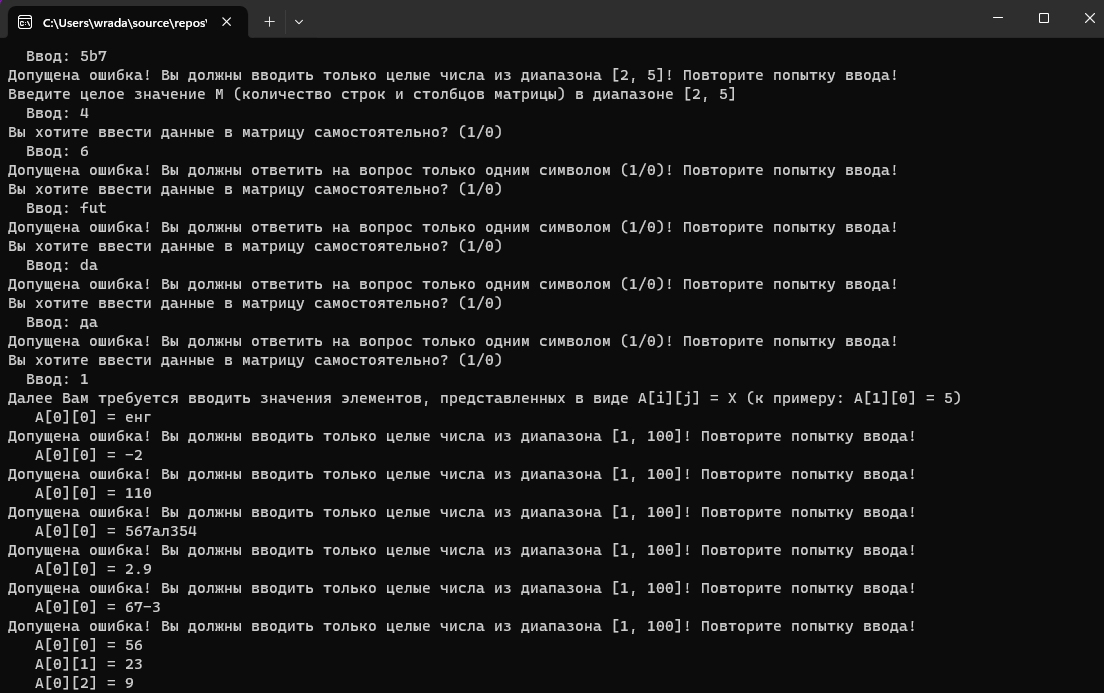
## Примеры тестирования

Примеры тестирования приведены на рис. 8, рис. 9, рис. 10 и рис. 11.

Рисунок 8 – Ввод корректных данных №1

Рисунок 9 – Ввод корректных данных №2

Рисунок 10 – Ввод некорректных данных №1

Рисунок 11 – Ввод некорректных данных №2

# ВЫВОДЫ

В ходе выполнения практической работы была разработана блок-схема ал­горитма и написана программа обработки данных в соответствии с выбранным и согласованным с преподавателем вариантом. При этом были проконтролиро­ваны типы и диапазоны вводимых данных, а также предусмотрена обработка других исключительных ситуаций (если они есть), например, ситуацию деления на ноль. Блок-схема является полной, т.е. описывает и процесс диалога с поль­зователем, и контроль вводимых данных, и подпрограммы вычислений с обра­боткой возможных исключительных операций. Блок-схема изображена по ГО­СТу. При обнаружении ошибки ввода или ошибки вычислений программа ин­формативно уведомляла пользователя о причине ошибки. Если ошибка проис­ходила на этапе ввода данных, то программа просила пользователя повторить ввод.

# ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

1. Информатика: Методические указания по выполнению практических работ / С.С. Смирнов, Д.А. Карпов — М., МИРЭА — Российский технологический университет, 2020. – 102 с. [83-90]
2. Воронов Г.Б. Информатика: Лекции по информатике / Г.Б. Воронов — М., МИРЭА — Российский технологический университет, 2023.
3. ГОСТ 19.701 – 90. Схемы алгоритмов, программ, данных и систем. Москва: Стандартинформ, 2010. – 24 с.