



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
ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 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
2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ.....	4
2.1 Функция main().....	4
2.2 Функция length_of_matrix().....	4
2.3 Функция answer_to_question().....	5
2.4 Функция matrix_from_user(int* matrix, int M).....	6
2.5 Функция matrix_from_random(int* matrix, int M).....	7
2.6 Функция cout_matrix(int* matrix, int M).....	7
2.7 Функция matrix_sort(int* matrix, int M).....	8
2.8 Структурированный код программы с комментариями.....	9
2.9 Примеры тестирования.....	18
3 ВЫВОДЫ.....	21
4 ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ.....	22

1 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 $M \times M$, 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.

2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ

2.1 Функция `main()`

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

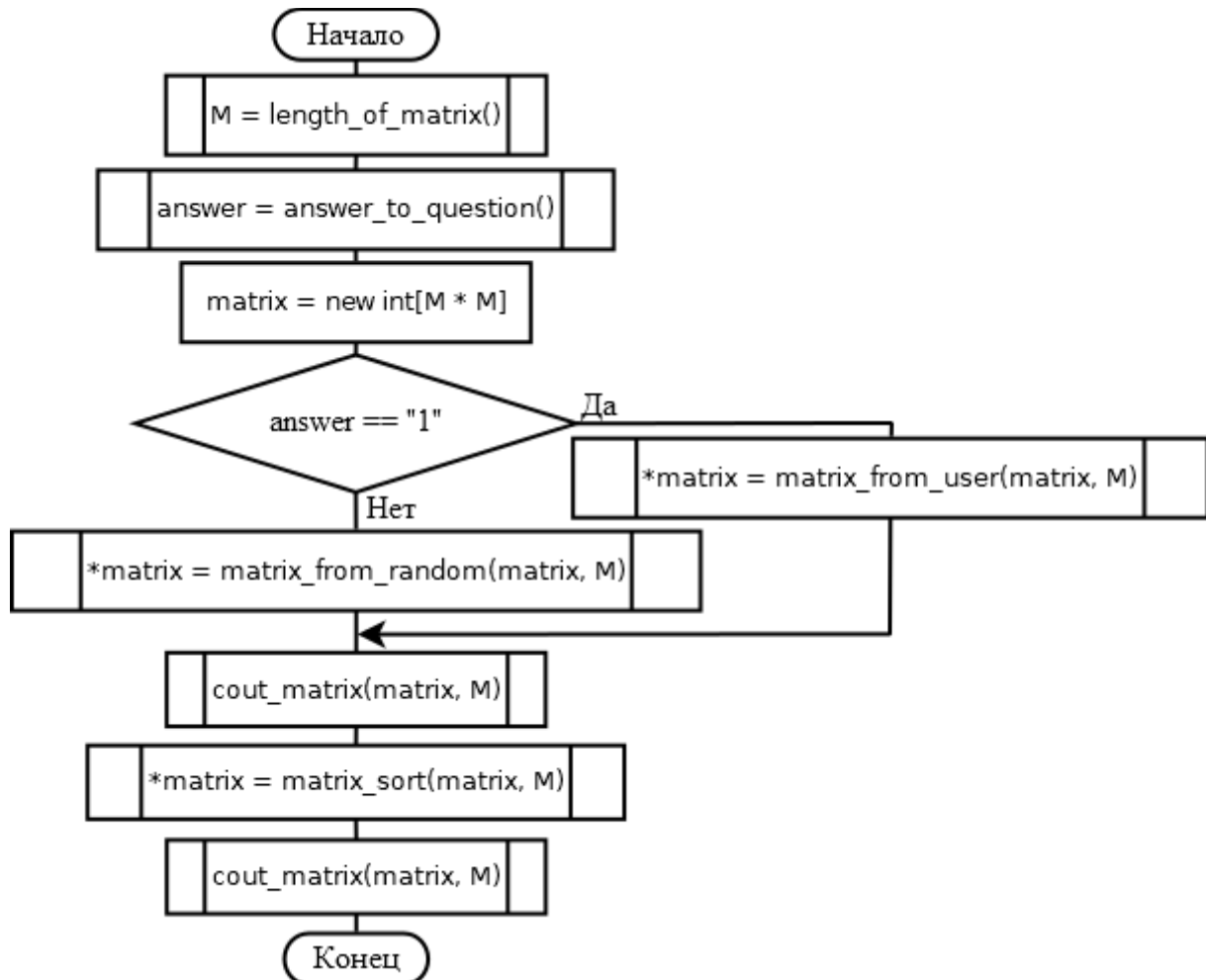


Рисунок 1 – Блок-схема функции `main()`

2.2 Функция `length_of_matrix()`

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

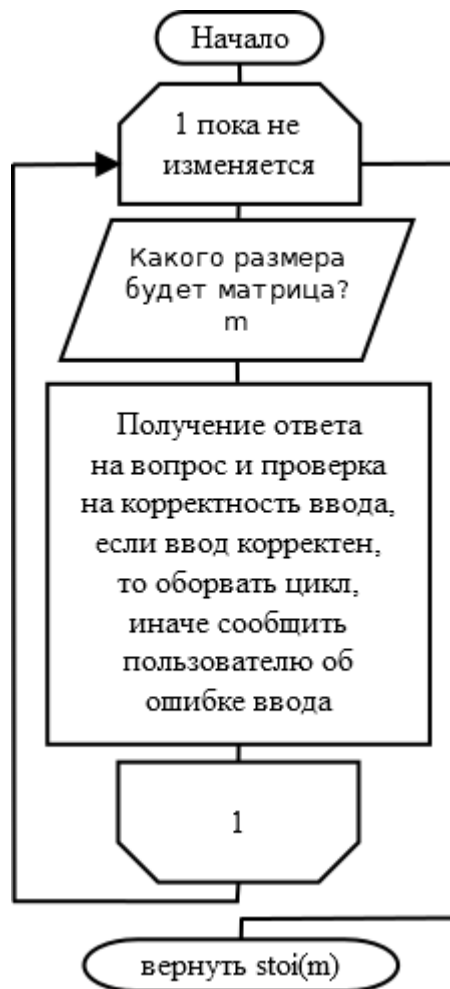


Рисунок 2 – Блок-схема функции `length_of_matrix()`

2.3 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.

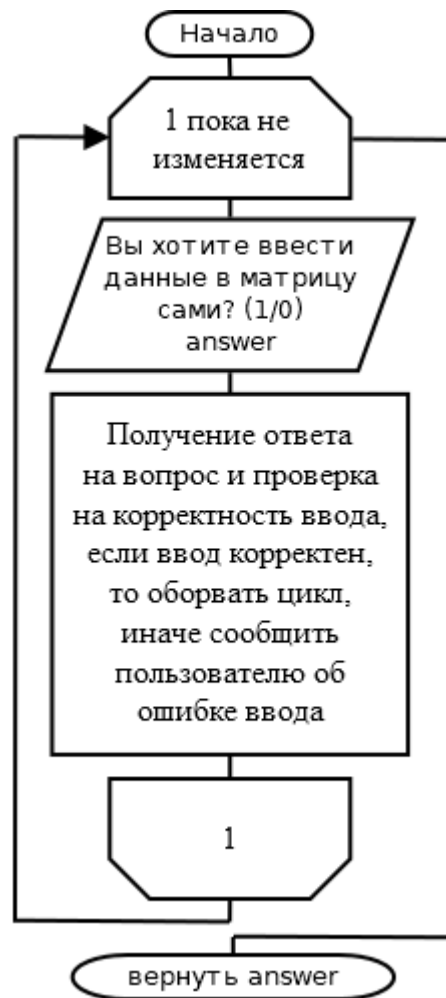


Рисунок 3 – Блок-схема функции `answer_to_question()`

2.4 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.

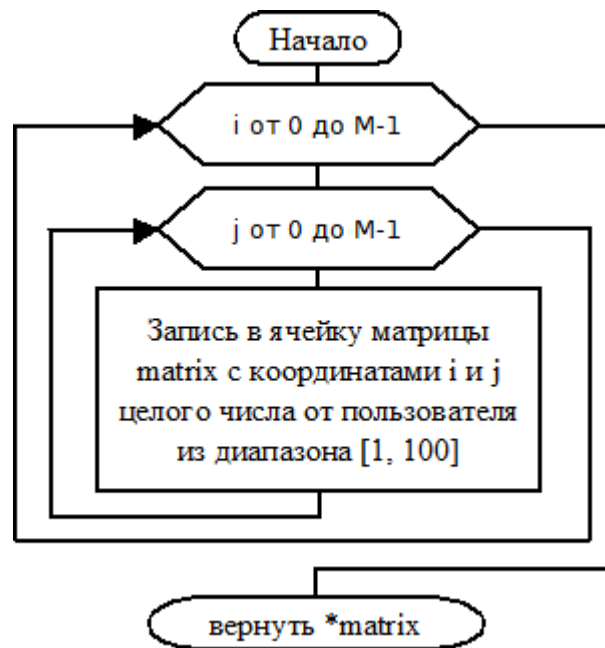


Рисунок 4 – Блок-схема функции `matrix_from_user(int* matrix, int M)`

2.5 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.

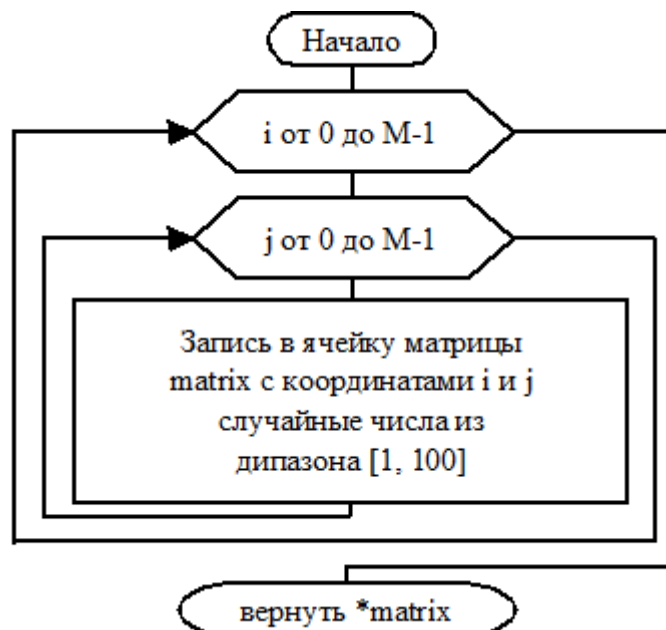


Рисунок 5 – Блок-схема функции `matrix_from_random(int* matrix, int M)`

2.6 Функция `cout_matrix(int* matrix, int M)`

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

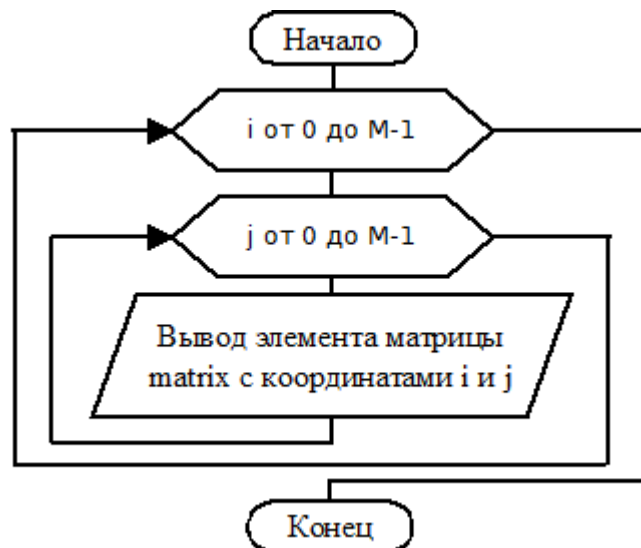


Рисунок 6 – Блок-схема функции `cout_matrix(int* matrix, int M)`

2.7 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.

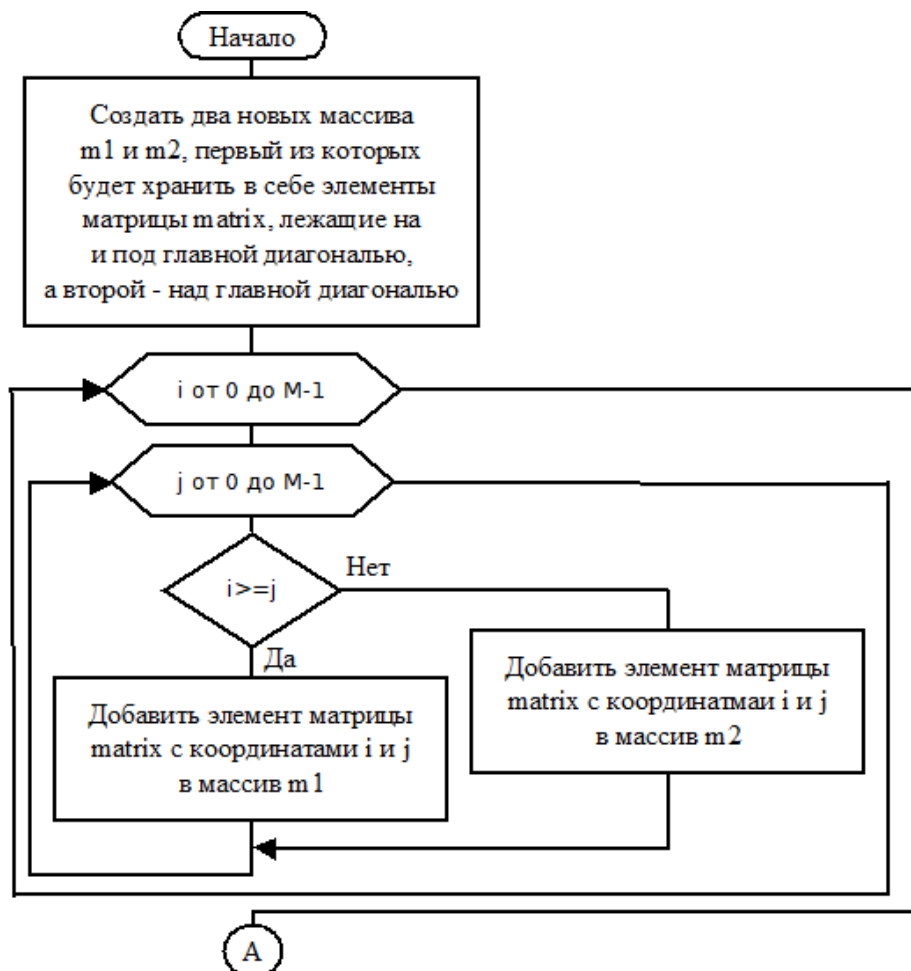


Рисунок 7 – Первая часть блок-схемы функции `matrix_sort(int* matrix, int M)`

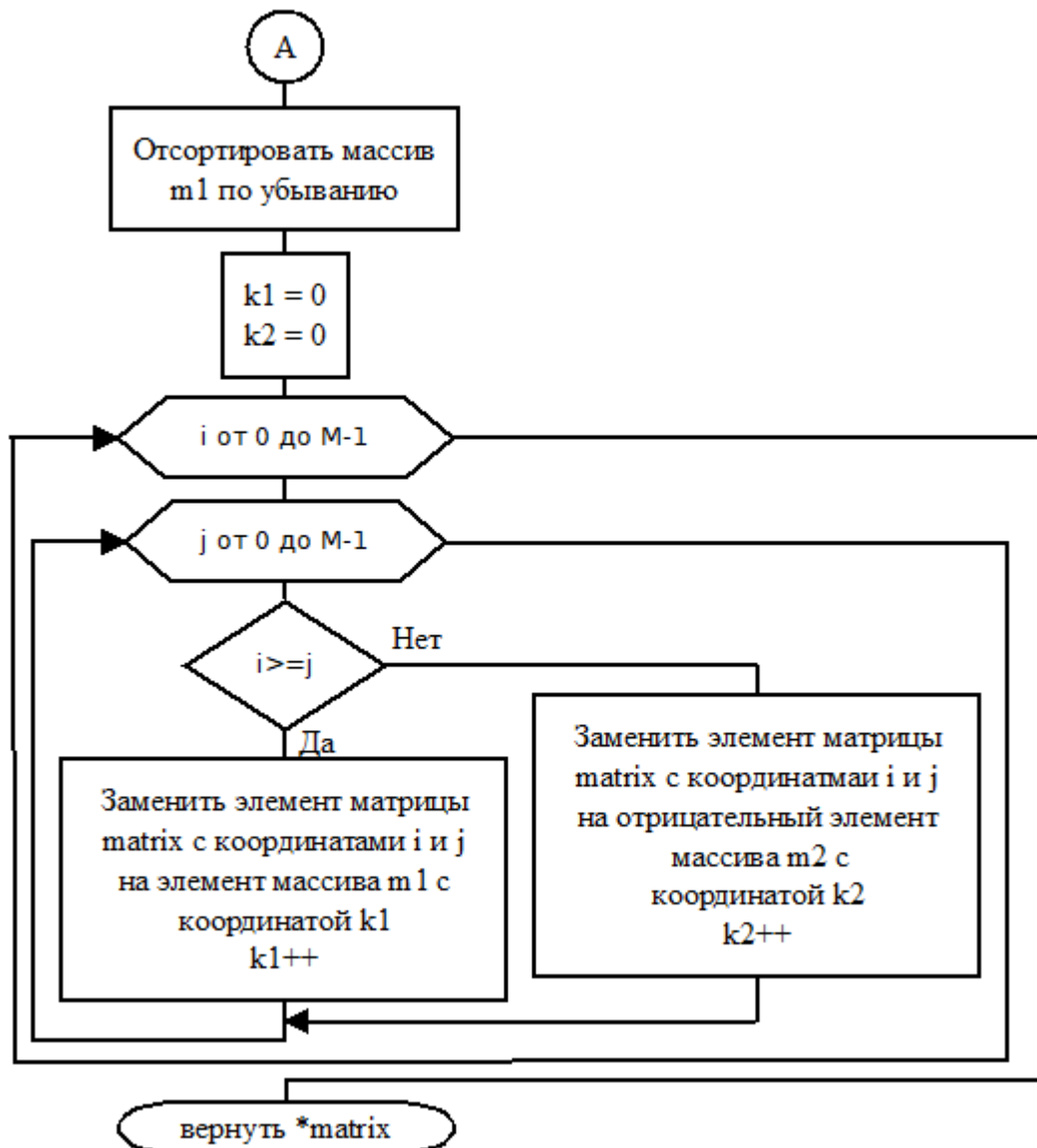


Рисунок 8 – Вторая часть блок-схемы функции matrix_sort(int* matrix, int M)

2.8 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 имён 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 М (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, за счёт 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 символов 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 целое 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 столбцов 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 swar; // Создание переменной swar, 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
```

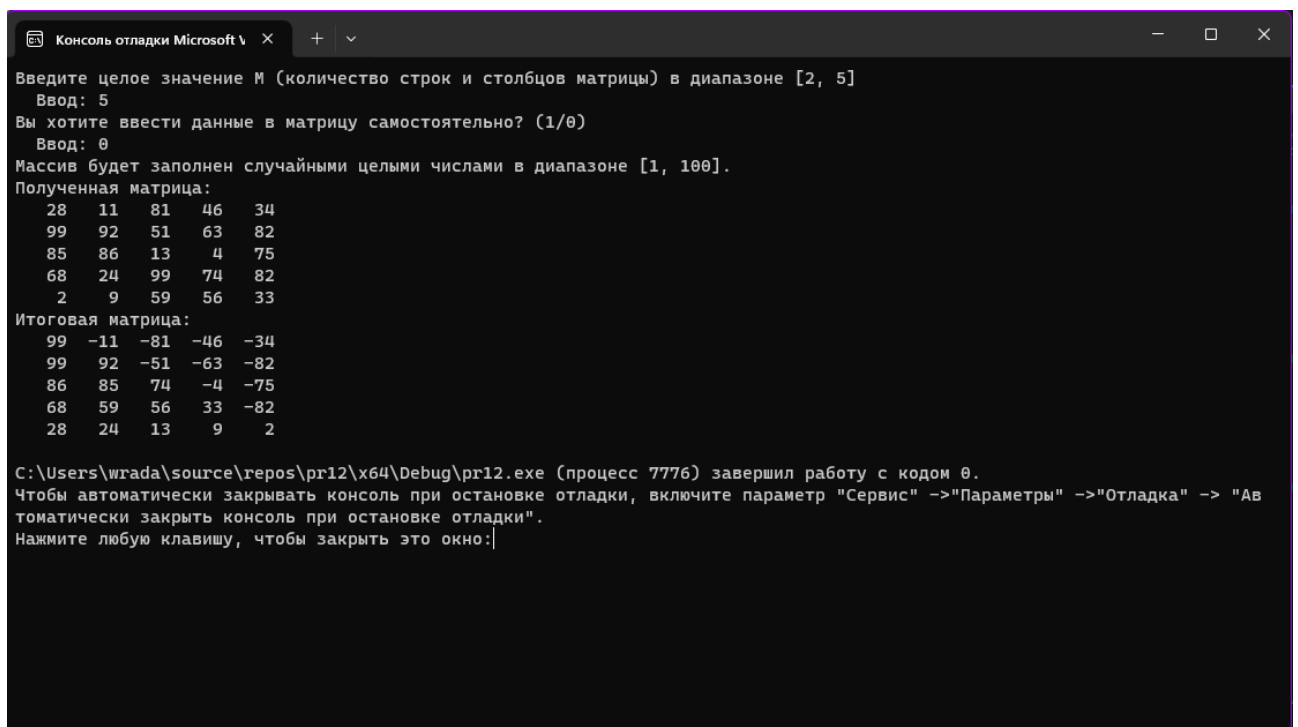
```
delete[] matrix; // Удаление матрицы matrix
```

```
return 0; // Завершение работы программы
```

```
}
```

2.9 Примеры тестирования

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



```
Консоль отладки Microsoft V
Введите целое значение M (количество строк и столбцов матрицы) в диапазоне [2, 5]
Ввод: 5
Вы хотите ввести данные в матрицу самостоятельно? (1/0)
Ввод: 0
Массив будет заполнен случайными целыми числами в диапазоне [1, 100].
Полученная матрица:
28 11 81 46 34
99 92 51 63 82
85 86 13 4 75
68 24 99 74 82
2 9 59 56 33
Итоговая матрица:
99 -11 -81 -46 -34
99 92 -51 -63 -82
86 85 74 -4 -75
68 59 56 33 -82
28 24 13 9 2
C:\Users\wrada\source\repos\pr12\x64\Debug\pr12.exe (процесс 7776) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно:
```

Рисунок 8 – Ввод корректных данных №1

```
Консоль отладки Microsoft Visual Studio
Введите целое значение М (количество строк и столбцов матрицы) в диапазоне [2, 5]
Ввод: 3
Вы хотите ввести данные в матрицу самостоятельно? (1/0)
Ввод: 1
Далее Вам требуется вводить значения элементов, представленных в виде A[i][j] = X (к примеру: A[1][0] = 5)
A[0][0] = 78
A[0][1] = 23
A[0][2] = 45
A[1][0] = 1
A[1][1] = 65
A[1][2] = 15
A[2][0] = 76
A[2][1] = 92
A[2][2] = 42
Полученная матрица:
78 23 45
1 65 15
76 92 42
Итоговая матрица:
92 -23 -45
78 76 -15
65 42 1
C:\Users\wrad\source\repos\pr12\x64\Debug\pr12.exe (процесс 16284) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" -> "Параметры" -> "Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно:
```

Рисунок 9 – Ввод корректных данных №2

```
C:\Users\wrad\source\repos\pr12\x64\Debug\pr12.exe
Введите целое значение М (количество строк и столбцов матрицы) в диапазоне [2, 5]
Ввод: 1
Допущена ошибка! Вы должны вводить только целые числа из диапазона [2, 5]! Повторите попытку ввода!
Введите целое значение М (количество строк и столбцов матрицы) в диапазоне [2, 5]
Ввод: 7
Допущена ошибка! Вы должны вводить только целые числа из диапазона [2, 5]! Повторите попытку ввода!
Введите целое значение М (количество строк и столбцов матрицы) в диапазоне [2, 5]
Ввод: 3.5
Допущена ошибка! Вы должны вводить только целые числа из диапазона [2, 5]! Повторите попытку ввода!
Введите целое значение М (количество строк и столбцов матрицы) в диапазоне [2, 5]
Ввод: -4
Допущена ошибка! Вы должны вводить только целые числа из диапазона [2, 5]! Повторите попытку ввода!
Введите целое значение М (количество строк и столбцов матрицы) в диапазоне [2, 5]
Ввод: dg
Допущена ошибка! Вы должны вводить только целые числа из диапазона [2, 5]! Повторите попытку ввода!
Введите целое значение М (количество строк и столбцов матрицы) в диапазоне [2, 5]
Ввод: 5b7
Допущена ошибка! Вы должны вводить только целые числа из диапазона [2, 5]! Повторите попытку ввода!
Введите целое значение М (количество строк и столбцов матрицы) в диапазоне [2, 5]
Ввод: 4
Вы хотите ввести данные в матрицу самостоятельно? (1/0)
Ввод: 6
Допущена ошибка! Вы должны ответить на вопрос только одним символом (1/0)! Повторите попытку ввода!
Вы хотите ввести данные в матрицу самостоятельно? (1/0)
Ввод: fut
Допущена ошибка! Вы должны ответить на вопрос только одним символом (1/0)! Повторите попытку ввода!
Вы хотите ввести данные в матрицу самостоятельно? (1/0)
Ввод: da
Допущена ошибка! Вы должны ответить на вопрос только одним символом (1/0)! Повторите попытку ввода!
Вы хотите ввести данные в матрицу самостоятельно? (1/0)
Ввод: да
Допущена ошибка! Вы должны ответить на вопрос только одним символом (1/0)! Повторите попытку ввода!
Вы хотите ввести данные в матрицу самостоятельно? (1/0)
Ввод: |
```

Рисунок 10 – Ввод некорректных данных №1

```
C:\Users\wrada\source\repos' X + v
Ввод: 5b7
Допущена ошибка! Вы должны вводить только целые числа из диапазона [2, 5]! Повторите попытку ввода!
Введите целое значение М (количество строк и столбцов матрицы) в диапазоне [2, 5]
Ввод: 4
Вы хотите ввести данные в матрицу самостоятельно? (1/0)
Ввод: 6
Допущена ошибка! Вы должны ответить на вопрос только одним символом (1/0)! Повторите попытку ввода!
Вы хотите ввести данные в матрицу самостоятельно? (1/0)
Ввод: fut
Допущена ошибка! Вы должны ответить на вопрос только одним символом (1/0)! Повторите попытку ввода!
Вы хотите ввести данные в матрицу самостоятельно? (1/0)
Ввод: da
Допущена ошибка! Вы должны ответить на вопрос только одним символом (1/0)! Повторите попытку ввода!
Вы хотите ввести данные в матрицу самостоятельно? (1/0)
Ввод: да
Допущена ошибка! Вы должны ответить на вопрос только одним символом (1/0)! Повторите попытку ввода!
Вы хотите ввести данные в матрицу самостоятельно? (1/0)
Ввод: 1
Далее Вам требуется вводить значения элементов, представленных в виде A[i][j] = X (к примеру: A[1][0] = 5)
A[0][0] = енг
Допущена ошибка! Вы должны вводить только целые числа из диапазона [1, 100]! Повторите попытку ввода!
A[0][0] = -2
Допущена ошибка! Вы должны вводить только целые числа из диапазона [1, 100]! Повторите попытку ввода!
A[0][0] = 110
Допущена ошибка! Вы должны вводить только целые числа из диапазона [1, 100]! Повторите попытку ввода!
A[0][0] = 567al354
Допущена ошибка! Вы должны вводить только целые числа из диапазона [1, 100]! Повторите попытку ввода!
A[0][0] = 2.9
Допущена ошибка! Вы должны вводить только целые числа из диапазона [1, 100]! Повторите попытку ввода!
A[0][0] = 67-3
Допущена ошибка! Вы должны вводить только целые числа из диапазона [1, 100]! Повторите попытку ввода!
A[0][0] = 56
A[0][1] = 23
A[0][2] = 9
```

Рисунок 11 – Ввод некорректных данных №2

3 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.

4 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.