



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий
Кафедра вычислительной техники

КУРСОВАЯ РАБОТА

по дисциплине «Алгоритмические основы обработки данных»

Тема курсовой работы: «Верхняя треугольная матрица чисел»

Студент группы ИВБО-01-20, Манохин Дмитрий Александрович

(учебная группа, фамилия, имя, отчество студента)

(подпись студента)

Руководитель курсовой работы зав.каф. ВТ, доцент, к.т.н. Платонова О.В

(должность, звание, учёная степень)

(подпись

руководителя)

Консультант старший преподаватель Асадова Ю.С.

(должность, звание, учёная степень)

(подпись

консультанта)

Работа представлена к защите « ____ » _____ 2021г.

Допущен к защите « ____ » _____ 2021г.

Москва 2021



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий

Кафедра вычислительной техники

Утверждаю

Заведующий кафедрой ВТ

О.В. Платонова

Подпись

ФИО

«02» сентября 2021 г.

ЗАДАНИЕ

на выполнение курсовой работы по дисциплине
«Алгоритмические основы обработки данных»

Студент Манохин Дмитрий Александрович Группа ИВБО-01-20

Тема «Верхняя треугольная матрица чисел»

Исходные данные: Язык программирования C++, QT Framework, техническое задание

Перечень вопросов, подлежащих разработке, и обязательного графического материала:

- создание верхней треугольной матрицы заданного размера с нулевыми значениями всех элементов;
- установка значения элемента матрицы с индексами i, j ;
- получение значения элемента матрицы с индексами i, j ;
- построчный ввод матрицы;
- построчный вывод на экран матрицы;
- разность двух треугольных матриц;
- проверка равенства двух матриц; запись матрицы в файл (первое число файла – размер матрицы);
- чтение матрицы из текстового файла (первое число файла – размер матрицы);
- присваивание одной матрицы другой;

Срок представления к защите курсовой работы:

до «30» декабря 2021 г.

Задание на выполнение курсовой работы выдал

Подпись руководителя

(О.В. Платонова)

Ф.И.О. руководителя

«02» сентября 2021 г.

Задание на курсовую работу получил

исполнителя

Подпись обучающегося

(Д.А. Манохин)

Ф.И.О.

«02» сентября 2021 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	5
1 ТЕХНИЧЕСКОЕ ЗАДАНИЕ	6
1.1 Введение	6
1.1.1 Наименование программы	6
1.1.2 Краткая характеристика области применения программы	6
1.2 Основания для разработки	6
1.3 Назначение разработки	7
1.4 Требования, предъявляемые к программе	7
1.4.1 Требования к функциональным характеристикам программы.....	7
1.4.2 Требования к техническим средствам, используемым при работе программы	7
1.4.3 Требования к языкам и среде разработки программы.....	8
1.4.4 Требования к информационным структурам на входе и на выходе программы	8
1.5 Требования к программной документации	8
1.6 Этапы разработки	8
2 ОБЗОР СПОСОБОВ ОРГАНИЗАЦИИ ДАННЫХ И ОБОСНОВАНИЕ ВЫБОРА СТРУКТУРЫ ДАННЫХ.....	9
2.1 Массив.....	9
2.2 Стек	9
2.3 Очередь	10
2.4 Связный список	10
2.5 Вектор	10
2.6 Множества	11
2.7 Выбор структуры данных	11
3 ОПИСАНИЕ ПРОГРАММЫ	12
3.1 Общие сведения.....	12
3.1.1 Наименование программы	12

3.1.2 Программное обеспечение, необходимое для функционирования программы	12
3.2 Функциональное назначение программы (классы решаемых задач и функциональные ограничения на применения)	12
3.3 Описание логической структуры программы.....	13
3.3.1 Алгоритмы, используемые в программе	13
3.3.2 Структура программы с описанием функций составных частей и связей между ними.....	35
3.4 Технические средства, которые используются при работе программы	37
3.5 Вызов программы.....	37
3.6 Входные данные	46
3.7 Выходные данные	46
ЗАКЛЮЧЕНИЕ	47
СПИСОК ЛИТЕРАТУРЫ	48
ПРИЛОЖЕНИЕ А	49

ВВЕДЕНИЕ

Студенты, начинающие изучать математический анализ, часто сталкиваются с проблемой по работе с треугольными матрицами. Эта проблема затрагивает достаточно большое количество учеников.

Специально для этого было разработано данное приложение, которое позволяет работать с верхними треугольными матрицами. Это позволит ученикам старших классов и студентам познакомиться и освоить навыки взаимодействия с треугольными матрицами.

Целью данной курсовой работы является получение практических навыков по предмету «Алгоритмические основы обработки данных».

1 ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1.1 Введение

Составленное техническое задание по дисциплине «Алгоритмические основы обработки данных» является документом к курсовой работе, который отражает все этапы разработки программного продукта, а также процесс проектирования и выявление требований, предъявляемых конечному продукту.

1.1.1 Наименование программы

Название данного приложения «Верхняя треугольная матрица» будет напрямую связываться с темой курсовой работы: «Верхняя треугольная матрица чисел». Данное название полностью отражает целевое назначение программного продукта. Английский вариант названия: «Upper Triangular Matrix». Краткое наименование «UTM».

1.1.2 Краткая характеристика области применения программы

Программа предназначена для удобной работы с верхними треугольными матрицами чисел в образовании, а также в повседневной жизни некоторых людей. Данное приложение будет полезным для учеников школ и колледжей, преподавателей, студентов и людей, которые работают в предприятиях.

1.2 Основания для разработки

Основанием для разработки является курсовая работа по дисциплине «Алгоритмические основы обработки данных», предусмотренная учебным планом направления подготовки 09.03.01 «Информатика и вычислительная техника» профиля «Вычислительные машины, комплексы, системы и сети», а также интерес учащегося в разборе выбранной темы.

1.3 Назначение разработки

Созданное приложение предназначено для понимания свойств и работы верхних треугольных матриц.

1.4 Требования, предъявляемые к программе

1.4.1 Требования к функциональным характеристикам программы

В приложении должны быть реализованы следующие операции:

- создание верхней треугольной матрицы заданного размера с нулевыми значениями всех элементов;
- установка значения элемента матрицы с индексами i, j ;
- получение значения элемента матрицы с индексами i, j ;
- построчный ввод матрицы;
- построчный вывод на экран матрицы;
- разность двух треугольных матриц;
- проверка равенства двух матриц;
- запись матрицы в файл;
- чтение матрицы из текстового файла;
- присваивание одной матрицы другой.

1.4.2 Требования к техническим средствам, используемым при работе программы

Персональный компьютер пользователя должен быть оснащен графическим адаптером, также должна быть установлена ОС Windows (не ниже Windows 7).

1.4.3 Требования к языкам и среде разработки программы

Для разработки используется язык программирования C++, в качестве среды разработки выступает Visual Studio. Для разработки графического интерфейса пользователя задействован механизм Visual Studio.

1.4.4 Требования к информационным структурам на входе и на выходе программы

В качестве входных данных программа принимает целые числа, являющие элементами матрицы, которые вводятся с клавиатуры. Выходные данные представляют собой матрицу целых чисел.

1.5 Требования к программной документации

1. Пояснительная записка оформляется в соответствии с ЛНА РТУ МИРЭА.

2. Проектная документация, составленная в соответствии с ГОСТ.

В процессе создания приложения вся проделанная работа документируется, должны быть сохранены все детали разработки, а также трудности, с которыми пришлось столкнуться. Всё выше перечисленное должно быть отражено в пояснительной записке, которая прилагается к работе.

1.6 Этапы разработки

1. Обзор способов организации данных и обоснование выбора структуры данных для выполнения операций: 02.09.2021 – 22.09.2021.

2. Разработка программы: 23.09.2021 – 30.11.2021.

3. Разработка программной документации: 01.12.2021 – 10.12.2021.

4. Оформление пояснительной записки: 11.12.2021 – 16.12.2021.

5. Защита курсовой работы: 23.12.2021 – 30.12.2021.

2 ОБЗОР СПОСОБОВ ОРГАНИЗАЦИИ ДАННЫХ И ОБОСНОВАНИЕ ВЫБОРА СТРУКТУРЫ ДАННЫХ

Структура данных — программная единица, позволяющая хранить и обрабатывать множество однотипных и/или логически связанных данных в вычислительной технике. Для добавления, поиска, изменения и удаления данных структура данных предоставляет некоторый набор функций, составляющих её интерфейс. Существует несколько основных типов структур данных, некоторые из них мы рассмотрим для выбора самой оптимальной в использовании.

2.1 Массив

Массив — структурированный тип данных. Величина, описанная как массив состоит из конечного числа других величин. Тип элементов массива называют базовым типом. Элементы массива упорядочены по индексам, определяющим положение элемента в массиве. Массивы могут быть как одномерными, так и двумерными, то есть иметь в качестве элементов массивы. Также массив может быть динамическим, его длина может изменяться во время работы программы.

2.2 Стек

Стек — линейная структура данных, функционирующая по принципу LIFO (Last in first out), для данной структуры доступно три базовые операции: добавление в начало, просмотр, удаление элемента. Последний добавленный элемент выходит первым. Добавление и удаление элементов происходит путем использования операций push и pop. Русский вариант названий: «толкать» и «выскакивать». Самым легким примером для понимания работы стека является оружейный магазин или стопка тарелок. Чтобы добраться до самого нижнего элемента необходимо избавиться от всех верхних.

2.3 Очередь

Очередь – структура данных, использующая принцип FIFO (First In First Out). Первый добавленный элемент покинет структуру первым. Добавление и удаление элементов происходит путем использования операций push и pop. Русский вариант названий: «толкать» и «выскакивать». Элемент, находящийся в начале очереди называется головой, а находящийся в конце называется хвостом.

2.4 Связный список

Связный список – это одна из базовых структур данных. Его часто сравнивают с массивом, так большинство структур реализуются либо с помощью массива, либо с помощью связного списка. Связный список состоит из группы узлов, которые вместе образуют последовательность. Каждый узел содержит фактические данные, которые в нем хранятся, и указатель на следующий узел в последовательности. Также существуют двусвязные списки, в которых у каждого узла есть указатель как на следующий, так и на предыдущий элемент в списке. Основные операции в связном списке включают добавление, удаление и поиск элемента в списке.

2.5 Вектор

Вектор – это динамический массив, который может самостоятельно управлять выделенной себе памятью. Использование векторов упрощает работу над кодом, так как исключает возможность утечек памяти. Доступ к элементам вектора осуществляется подобно массиву. Для добавления элементов в вектор используется функция `push_back()`, в которой передается добавляемый элемент. Так же для вектора существует операция `insert`, используя которую можно вставлять новые элементы между старыми элементами. Данная функция самостоятельно сдвигает элементы внутри массива.

2.6 Множества

Множества хранят данные без определённого порядка и без повторяющихся элементов. В множествах помимо добавления и удаления элементов, есть функции которые работают для двух множеств: объединение множеств, возврат пересечения множеств, возврат разницы множеств, и проверка содержит ли одно множество все элементы другого множества.

2.7 Выбор структуры данных

В ходе работы программы будут создаваться объекты, принадлежащие одному классу, нам необходима структура данных, которая будет способна хранить в себе указатели на эти объекты. Стек и очередь не подходят для выполнения необходимой нам задачи, так как их функции не будут использоваться в данной программе. Также, в программе необходимо будет работать с коэффициентами, что невозможно при использовании множеств. Кроме того, размер массива объектов не будет определен заранее. Поэтому целесообразно будет использовать динамический массив. Это единственная структура данных способная решить поставленную задачу, помимо этого задание требует использование динамического массива.

3 ОПИСАНИЕ ПРОГРАММЫ

3.1 Общие сведения

В процессе выполнения курсовой работы и исполнения назначенных условий было создано программное решение для операционной системы Windows для проведения различных операций с верхними треугольными матрицами. Программа предоставляет инструменты для проведения операций, обозначенных в техническом задании.

3.1.1 Наименование программы

Название программы формировалось исходя из ее предназначения и основного функционала. «Верхняя треугольная матрица чисел». Английский вариант названия: «Upper Triangular Matrix». Краткое наименование «UTM».

3.1.2 Программное обеспечение, необходимое для функционирования программы

Для корректного функционирования данного программного продукта необходимо, чтобы на персональном компьютере или ноутбуке была установлена ОС от компании Microsoft, а именно Windows (Windows 10). Также требуется наличие графического адаптера, чтобы устройство могло справляться с обработкой отображения консоли приложения. Другие требования к устройству не предусмотрены.

3.2 Функциональное назначение программы (классы решаемых задач и функциональные ограничения на применения)

Поскольку программа является консольной, то необходимо соблюсти корректный ввод, иначе программа выдаст ошибку, предусмотренную в программной коде.

3.3 Описание логической структуры программы

Для реализации данной задачи необходимо реализовать один класс, описывающий сущность Matrix(матрица), и описать его взаимодействие, используя принципы объектно-ориентированного программирования.

3.3.1 Алгоритмы, используемые в программе

Для написания программы необходимо использовать функции библиотек «`iostream`» и «`conio.h`», стандартные потоки ввода/вывода `cin/cout`, условные операторы и циклы; для работы с файлами необходимо подключить библиотеку «`fstream`»; библиотеку «`vector`» необходимо подключить для использования структуры `vector`; работа со строками осуществляется средствами библиотеки «`string`»; для вызова консоли необходимо подключить библиотеку «`iostream`».

Реализация класса «Matrix» представлена на рисунках 3.1 – 3.15 в виде блок-схем с описанием действий.

На рисунке 3.1 изображен конструктор класса, который создает динамический массив, заполненный нулями, для работы с верхней матрицей чисел.

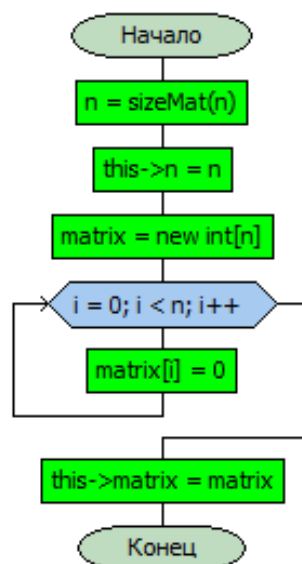


Рисунок 3.1 – Конструктор класса

На рисунке 3.2 изображен метод, целью которого является возвращение изменение значения элемента матрицы.

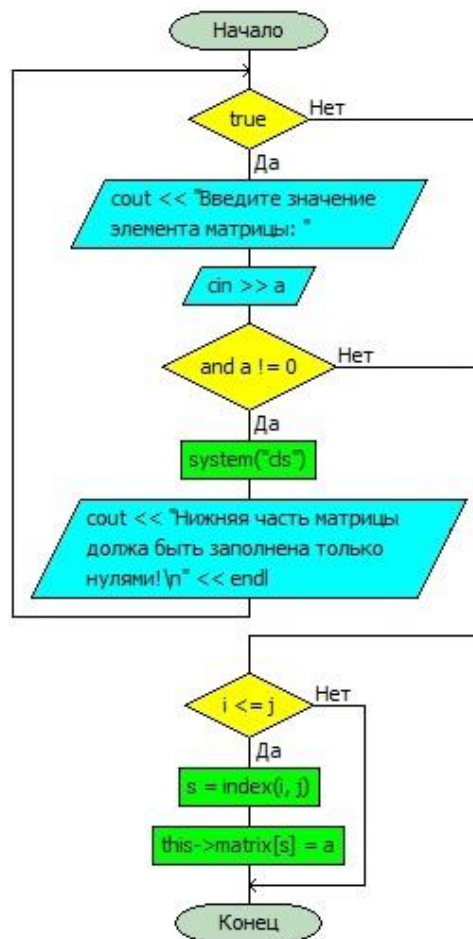


Рисунок 3.2 – Метод setEl

На рисунке 3.3 изображен метод, целью которого является возвращение значения элемента матрицы.

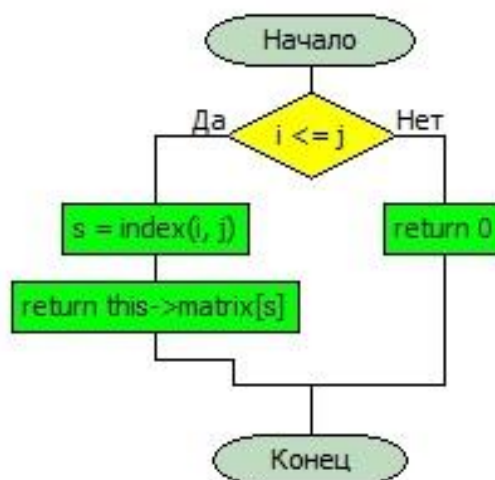


Рисунок 3.3 – Метод getEl

На рисунке 3.4 изображен метод, целью которого является вывести значения динамического массива, в виде верхней треугольной матрицы чисел.

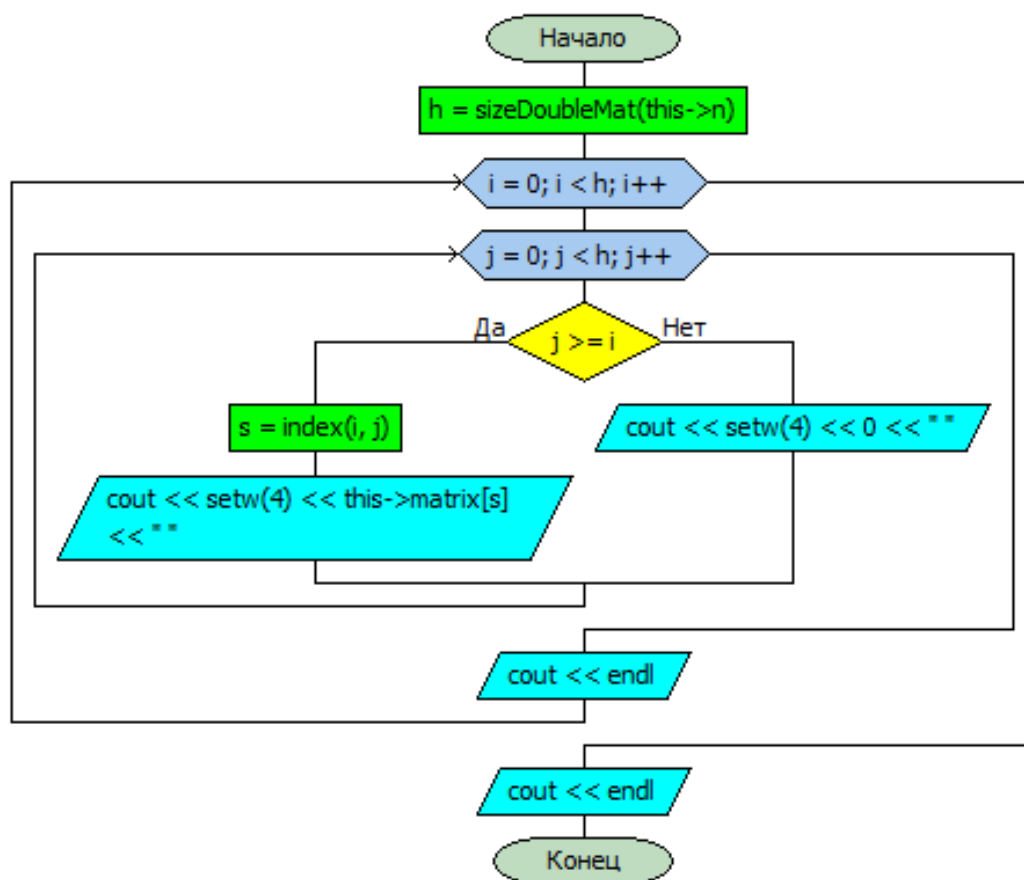


Рисунок 3.4 – Метод outPut

На рисунке 3.5 изображен метод, целью которого является возвращение размера динамического одномерного массива текущего объекта, который используется для хранения верхних значений треугольной матрицы чисел.

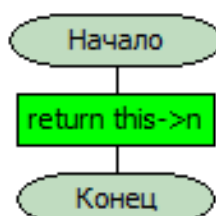


Рисунок 3.5 – Метод getSize

На рисунке 3.6 изображен метод, который позволяет пользователю ввести значения верхней треугольной матрицы с клавиатуры в консоли, также идет проверка на правильность ввода значений.

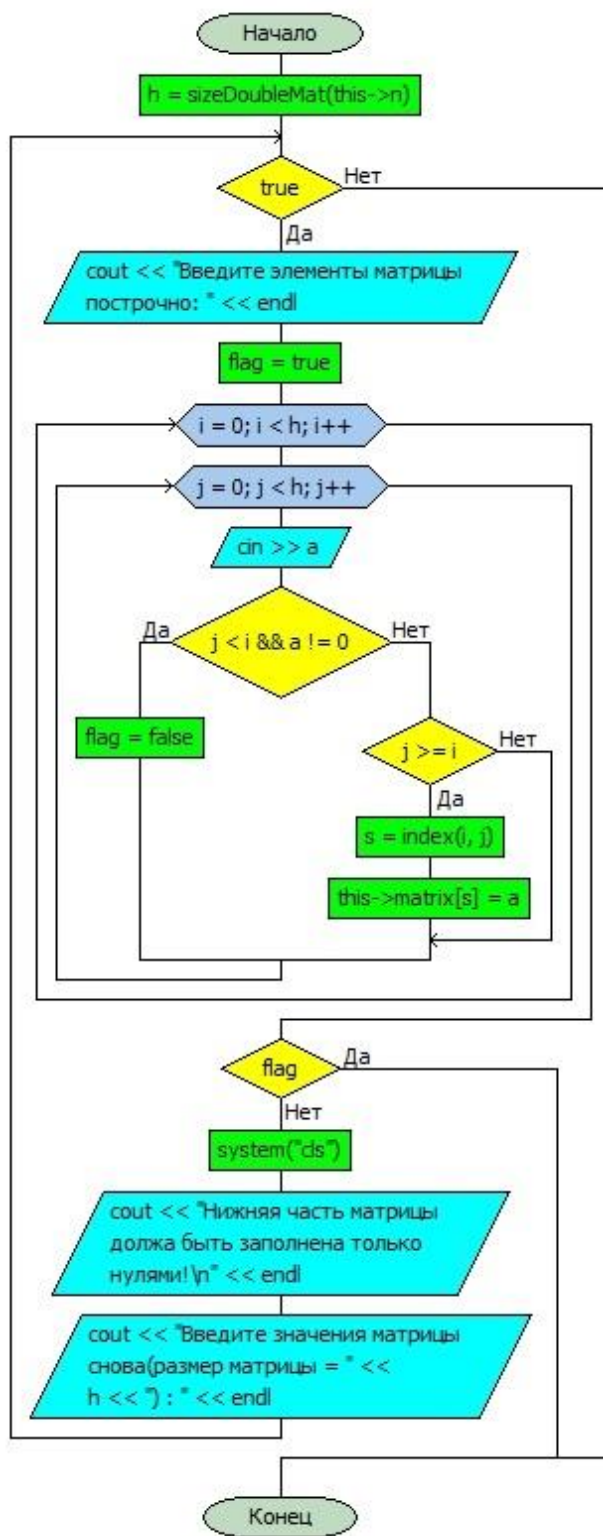


Рисунок 3.6 – Метод input

На рисунке 3.7 изображен метод, целью которого является вычислить, если это возможно, разность двух треугольных матриц.

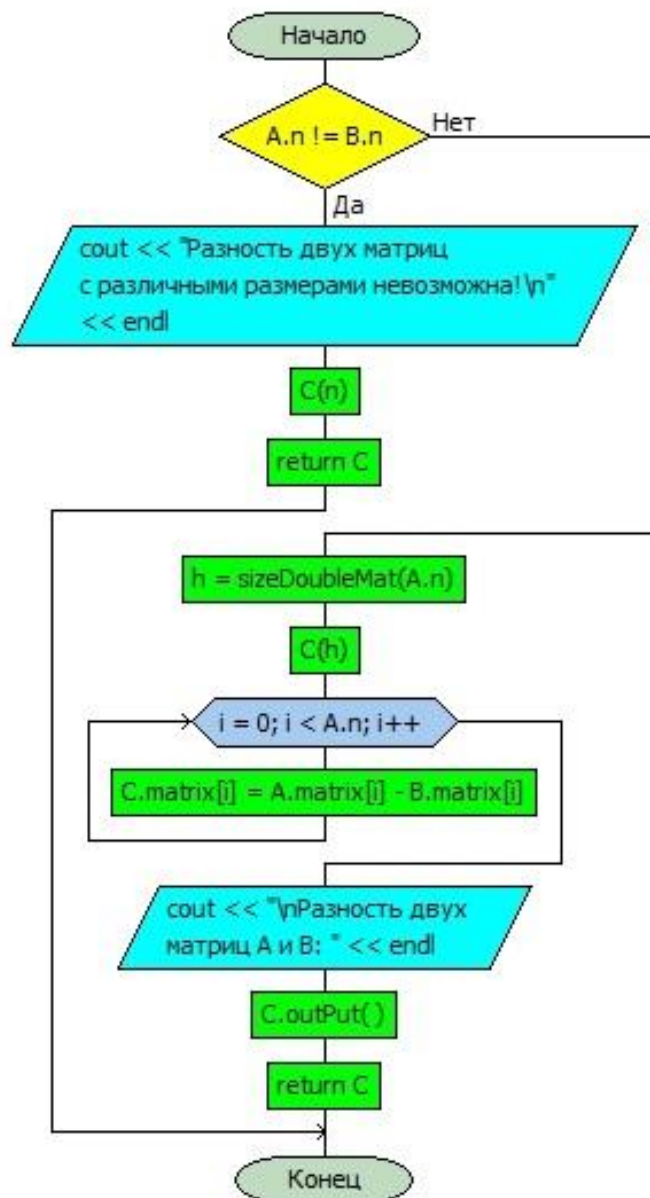


Рисунок 3.7 – Метод `diff`

На рисунке 3.8 изображен метод, целью которого является возвращение проверки матриц на симметричность, если они одинаковые по размерам.

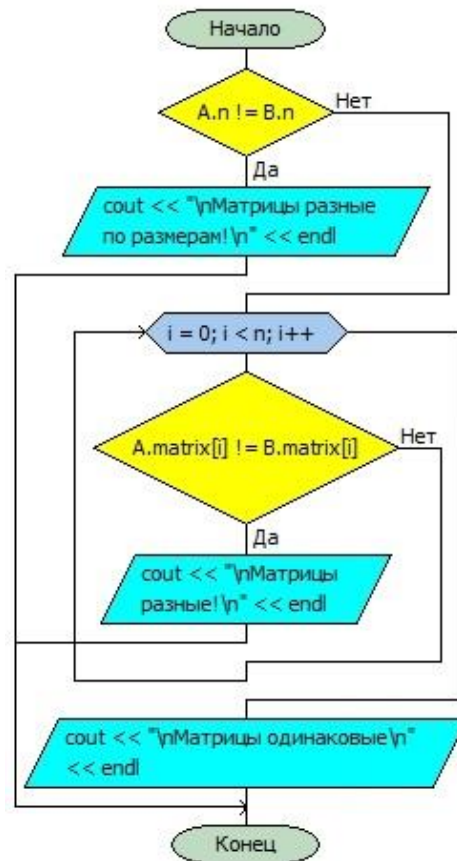


Рисунок 3.8 – Метод simI

На рисунке 3.9 изображен метод, целью которого является возвращение значения размера матрицы по размеру одномерного динамического массива.

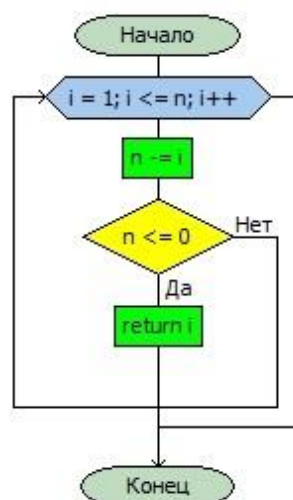


Рисунок 3.9 – Метод sizeDoubleMat

На рисунке 3.10 изображен метод, который позволяет пользователю вывести матрицу в созданный файл на рабочем столе. Также, метод проверяет название файла, вводимое пользователем, на наличие в конце типа файла, если оно присутствует.

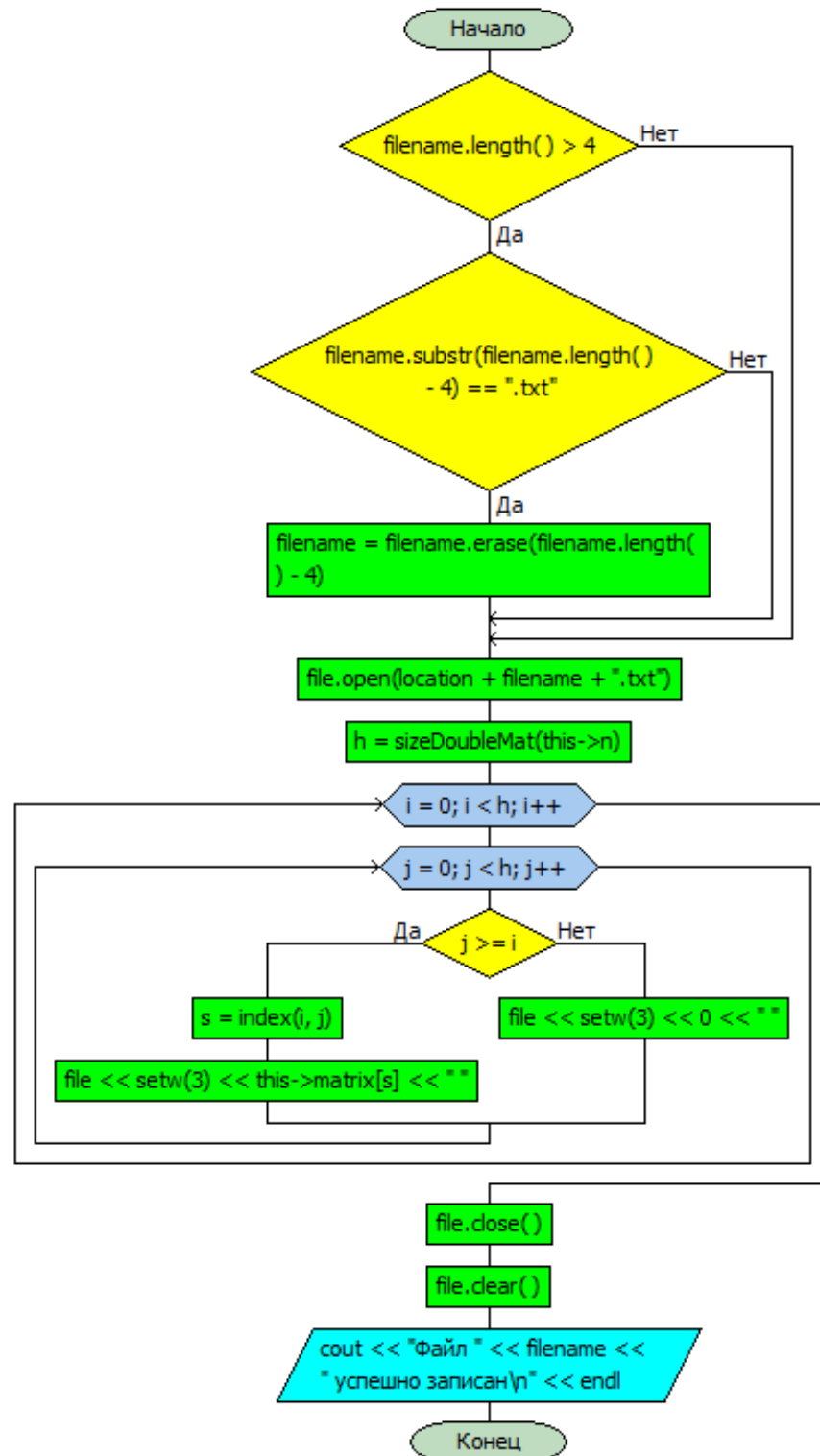


Рисунок 3.10 – Метод writeFile

На рисунке 3.11 изображен метод, который позволяет считывать значения для новой матрицы с текстового файла, созданного на рабочем столе пользователя.

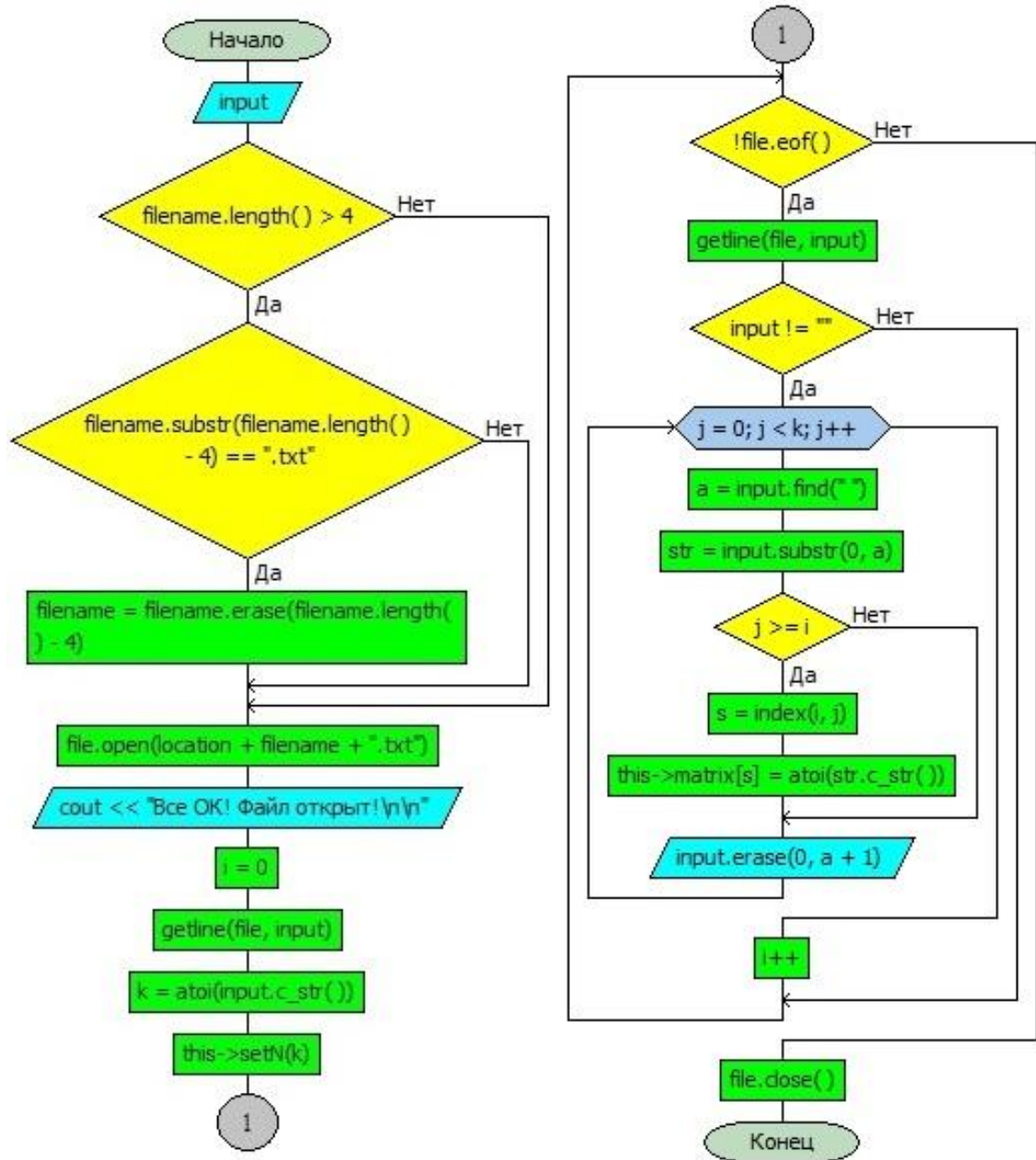


Рисунок 3.11 – Метод readFile

На рисунке 3.12 изображен метод, целью которого является присваивание значений одной матрицы к другой.



Рисунок 3.12 – Метод assignment

На рисунке 3.13 изображен метод, целью которого является возвращение размера одномерного массива для хранения значений по размеру матрицы.

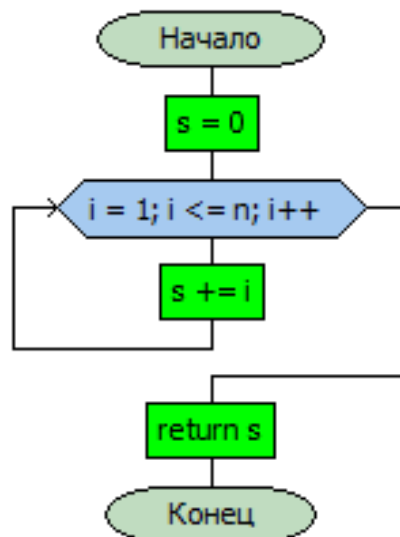


Рисунок 3.13 – Метод sizeMat

На рисунке 3.14 изображен метод, целью которого является изменение параметра объекта, отвечающего за размер одномерного массива.

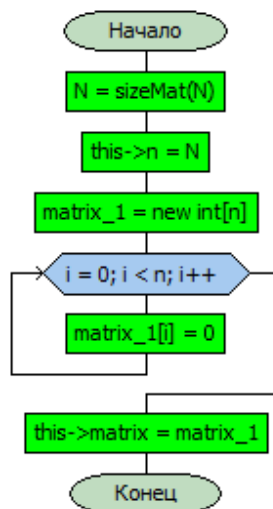


Рисунок 3.14 – Метод setN

На рисунке 3.15 изображен метод, целью которого является возвращение значения индекса одномерного массива по двум индексам матрицы.

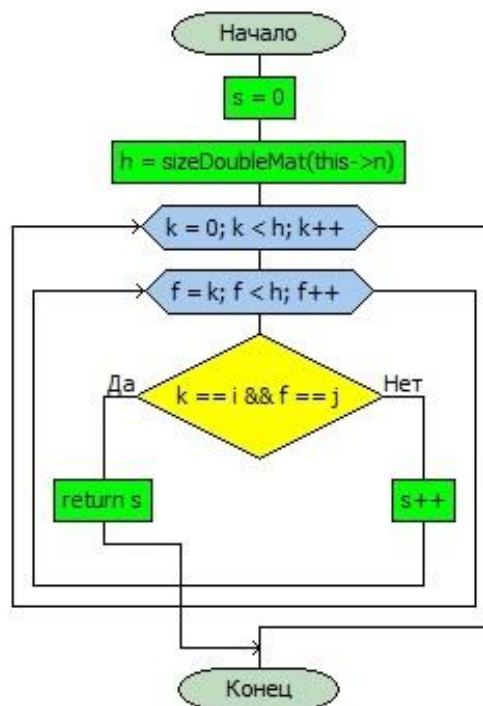


Рисунок 3.15 – Метод index

Реализация главной и других функций показана на рисунках 3.16 – 3.31 в виде блок-схем с описанием действий.

На рисунке 3.16 изображена функция, которая проверяет правильность ввода размера матрицы (только положительные, целые числа), если пользователь ввел правильное значение, то функция возвращает это число.

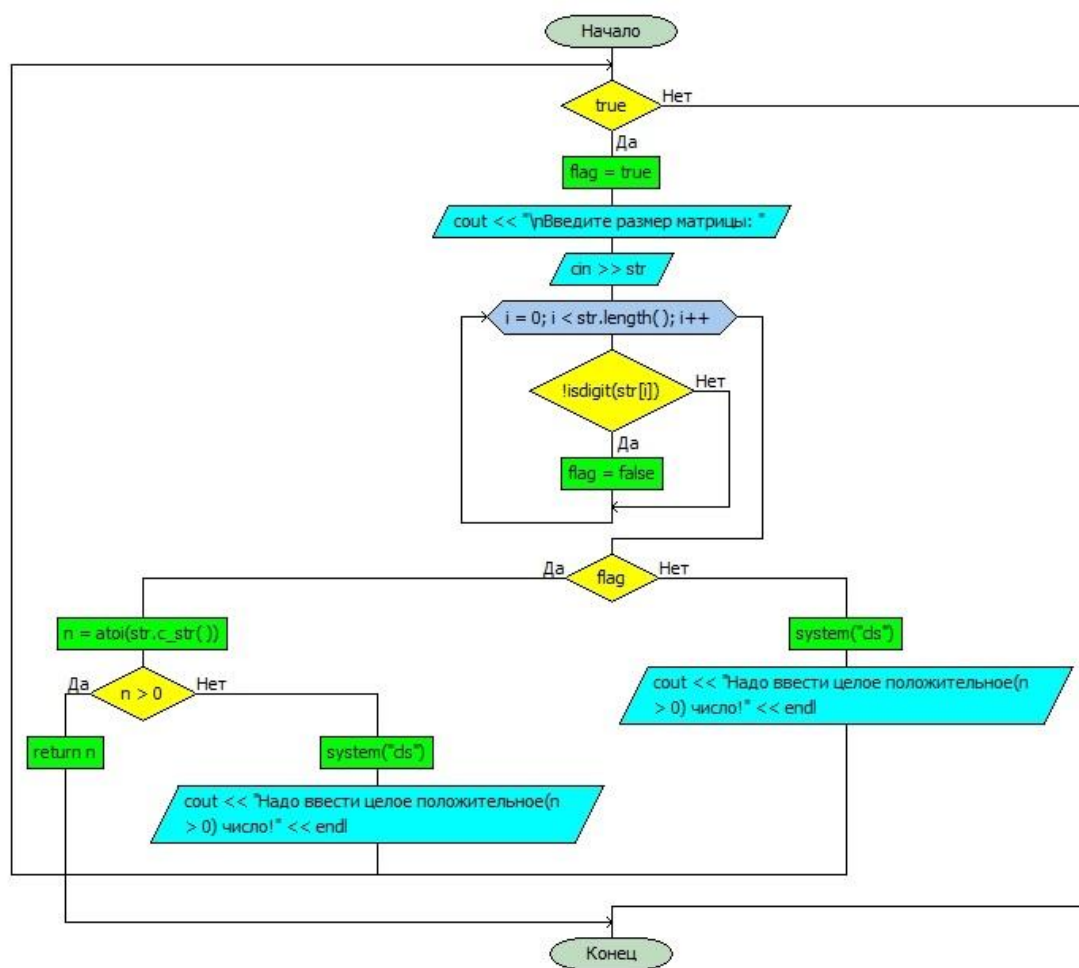


Рисунок 3.16 – Функция vvodSizeMat

На рисунке 3.17 изображена функция, целью которой является задача цвета текста выводимого на консоль.

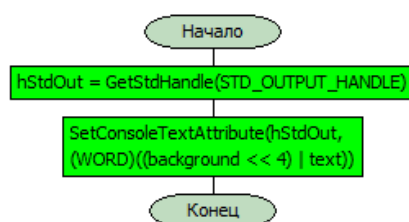


Рисунок 3.17 – Функция SetColor

На рисунке 3.18 изображена функция, которая проверяет правильность ввода выбора пункта (только те, что присутствуют на экране). В случае ошибки будет выведено соответствующее предупреждение и просьбы ввести еще раз. В случае правильного ввода, функция возвращает число, которое ввел пользователь.

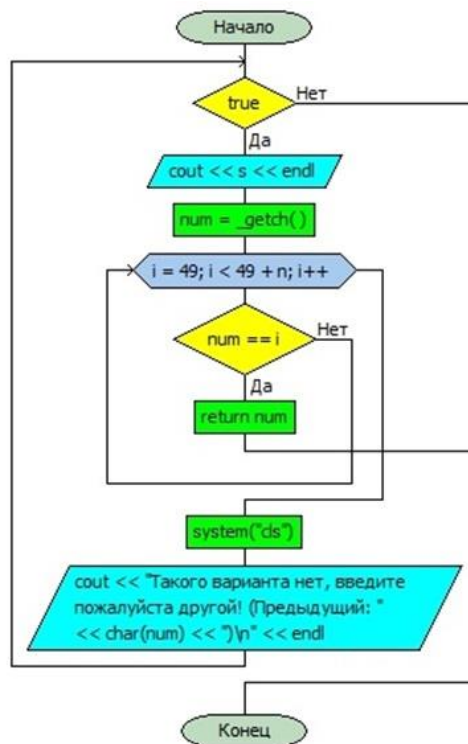


Рисунок 3.18 – Функция getchOption

На рисунке 3.19 изображена функция, целью которой является досрочный выход из программы с выводом сообщения об этом.

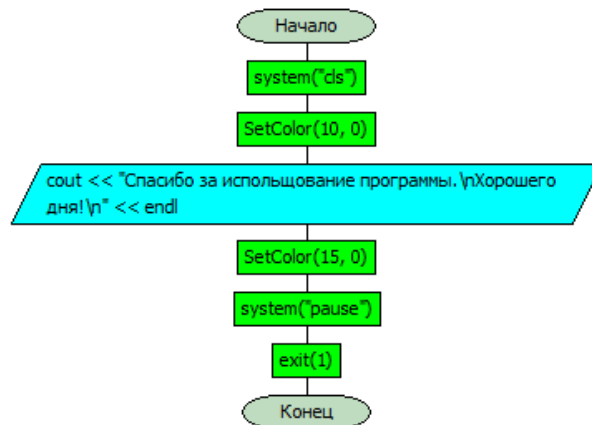


Рисунок 3.19 – Функция exit

На рисунке 3.20 изображена функция, которая проверяет наличие созданного файла, на рабочем столе, с названием, которое вводит пользователь. В случае отсутствия файла, будет выведено соответствующее сообщение.

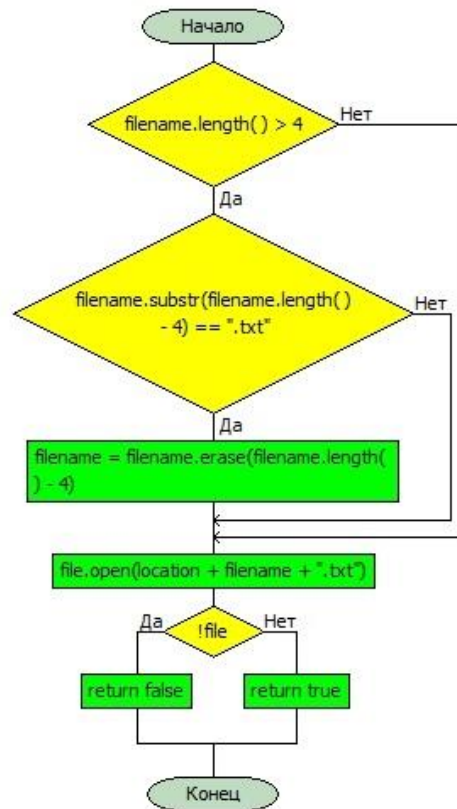


Рисунок 3.20 – Функция openFile

На рисунке 3.21 изображена функция, целью которой является вывод сообщения о неправильном вводе пункта в главном меню пользователем.

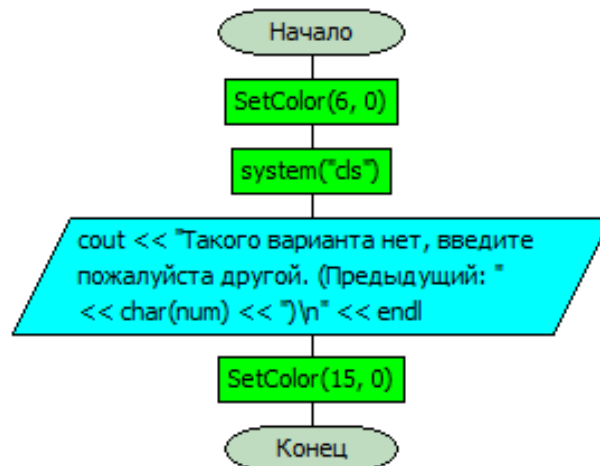


Рисунок 3.21 – Функция nothing

На рисунке 3.22 изображена функция, которая выводит на консоль меню ввода матриц и предлагает пользователю ввод значений с клавиатуры или из файла.

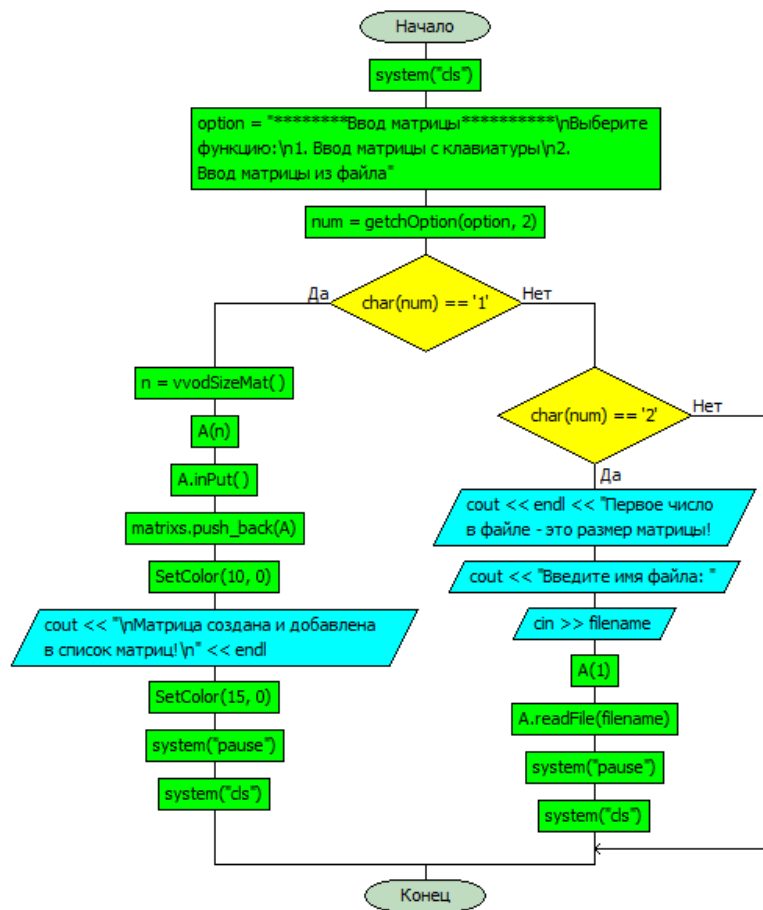


Рисунок 3.22 – Функция z1

На рисунке 3.23 изображена функция, которая выводит на консоль меню вывода матриц и предлагает пользователю вывод значений на консоль или в файл.

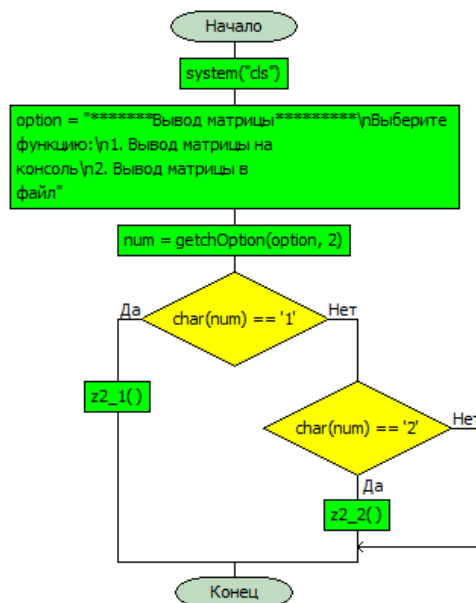


Рисунок 3.23 – Функция z2

На рисунке 3.24 изображена функция, которая предлагает пользователю выбор матрицы для вывода, в случае если их больше одной, и вызывает метод вывода матриц на консоль.

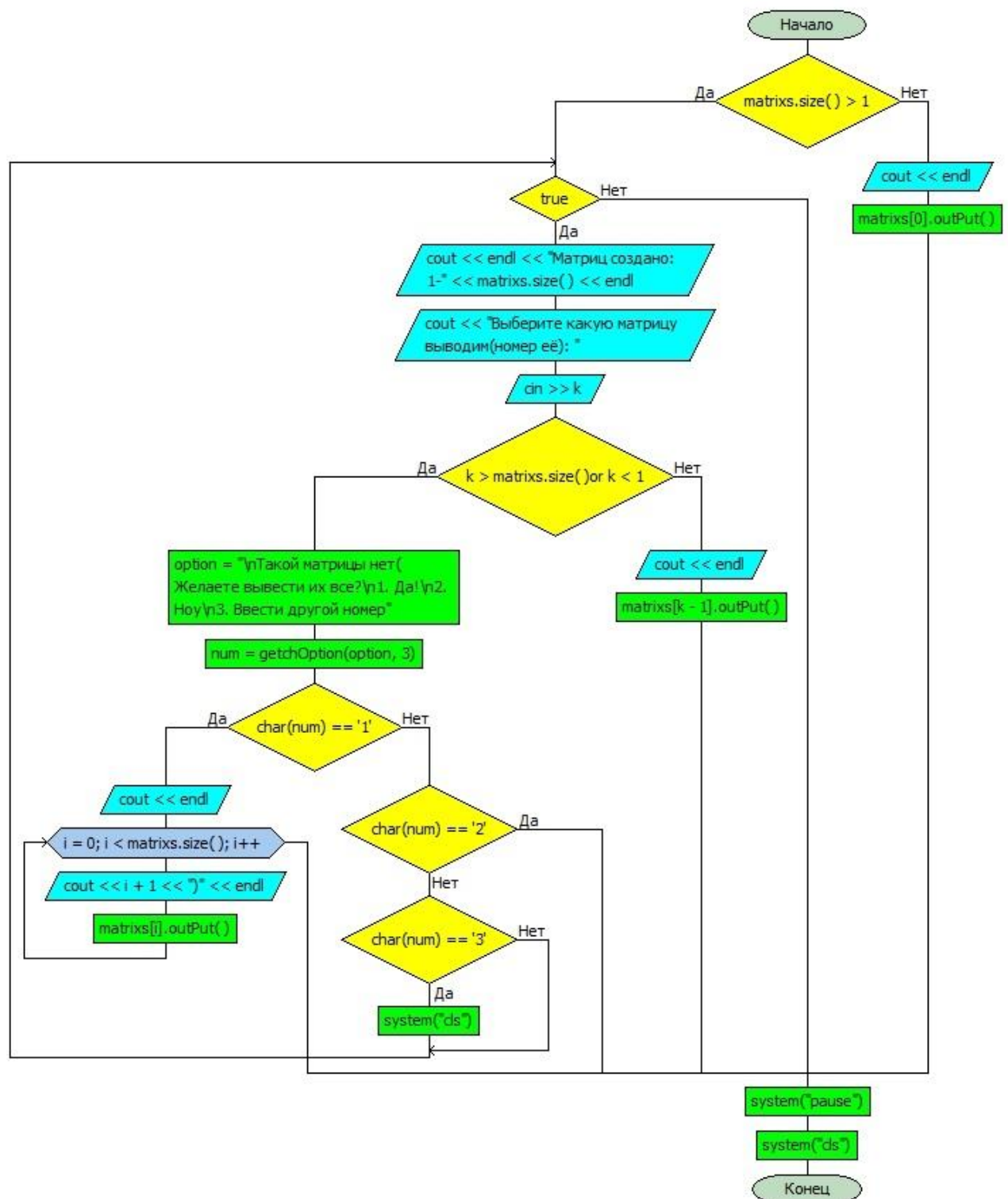


Рисунок 3.24 – Функция `z2_1`

На рисунке 3.25 изображена функция, которая предлагает пользователю выбор матрицы для вывода, в случае если их больше одной, и вызывает метод вывода матриц в файл.

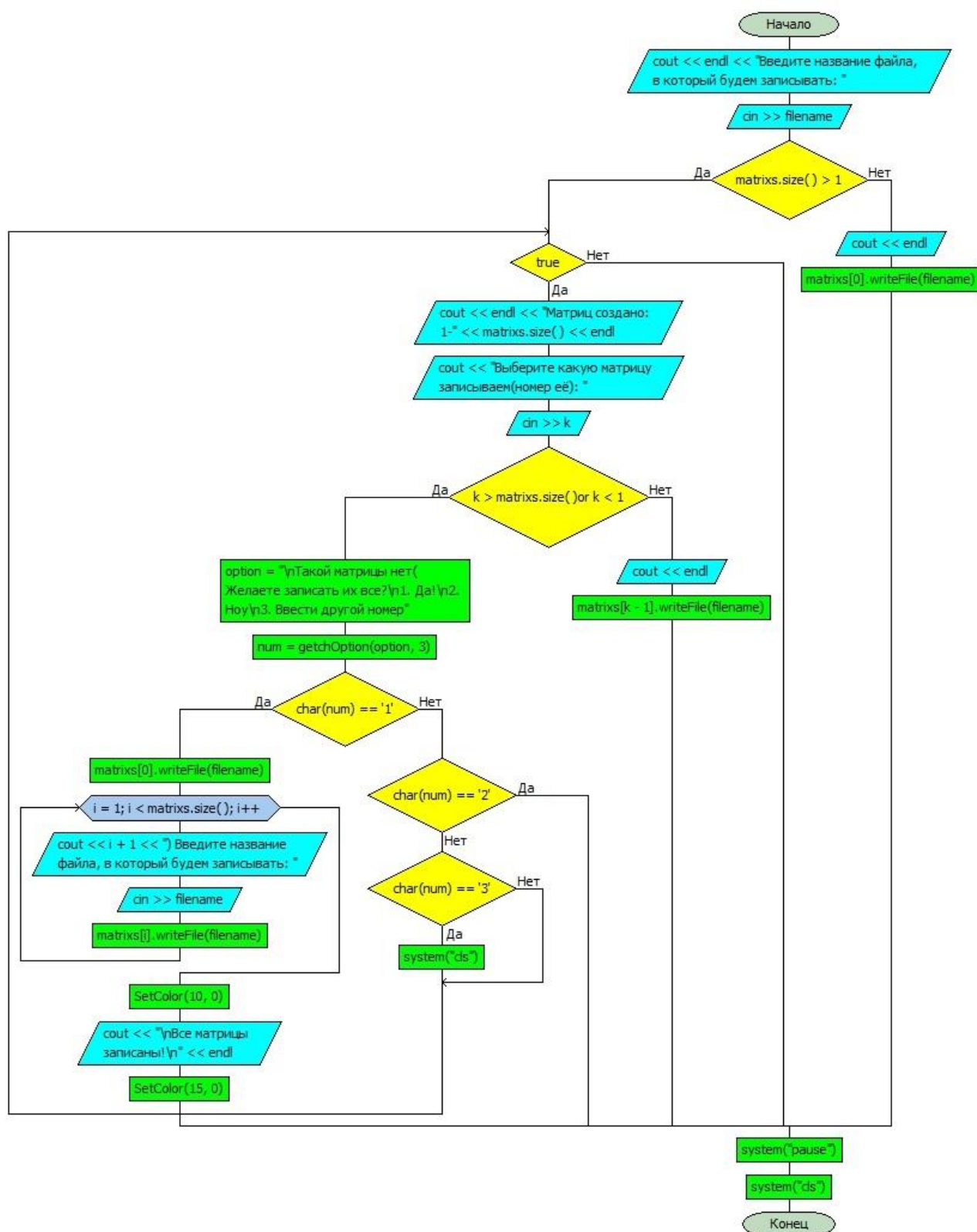


Рисунок 3.25 – Функция `z2_2`

На рисунке 3.26 изображена функция, которая предлагает пользователю выбор матрицы для работы, после чего просит ввести число и индексы матрицы, в которой будет изменяться значение на новое.

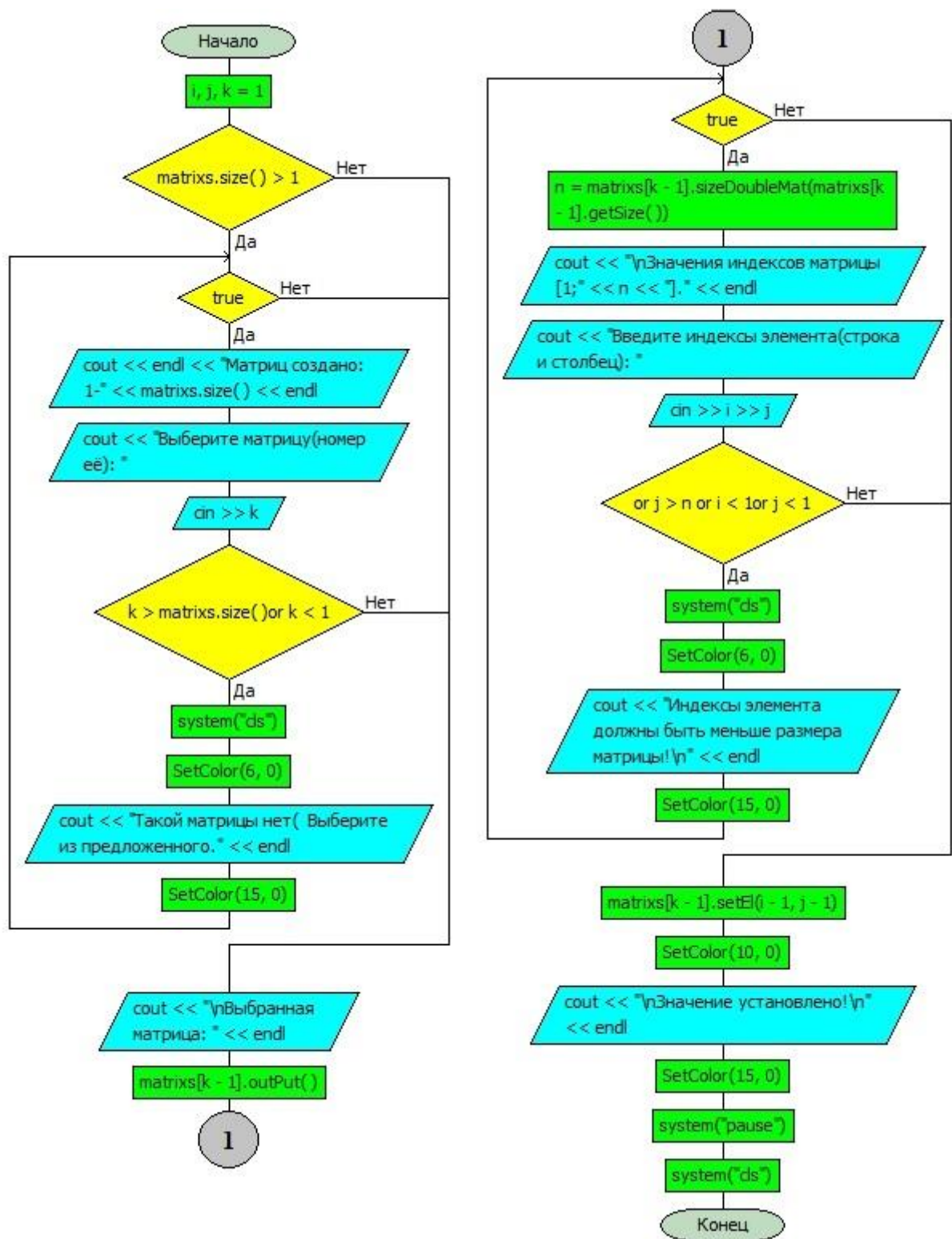


Рисунок 3.26 – Функция z3

На рисунке 3.27 изображена функция, которая предлагает пользователю выбор матрицы для работы, после чего просит ввести индексы матрицы для вывода значения матрицы, расположенного по введенным индексам, на консоль.

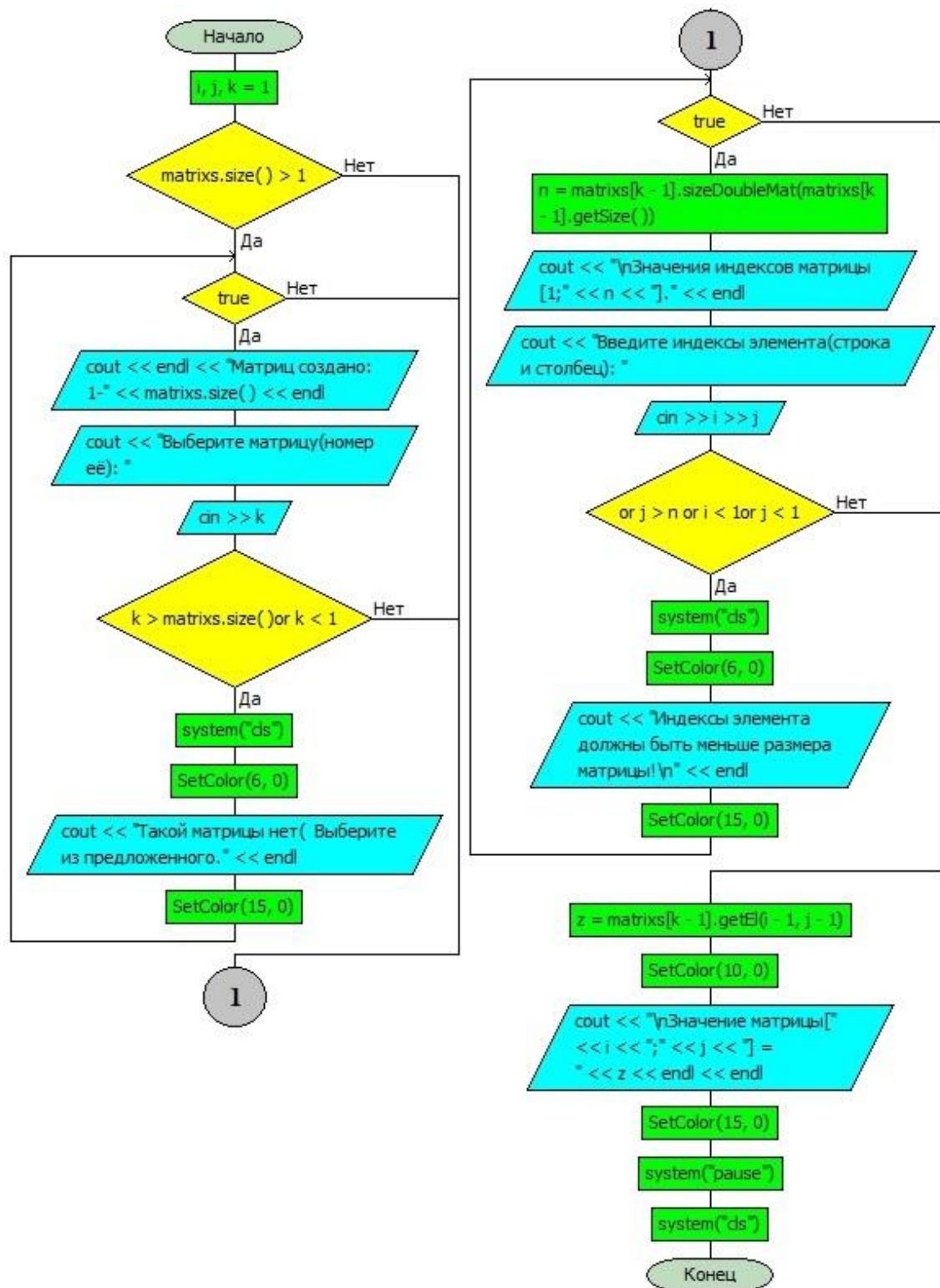


Рисунок 3.27 – Функция z4

На рисунке 3.28 изображена функция, которая предлагает пользователю выбор двух матриц для работы, после чего вызывает метод для расчета разности двух выбранных матриц.

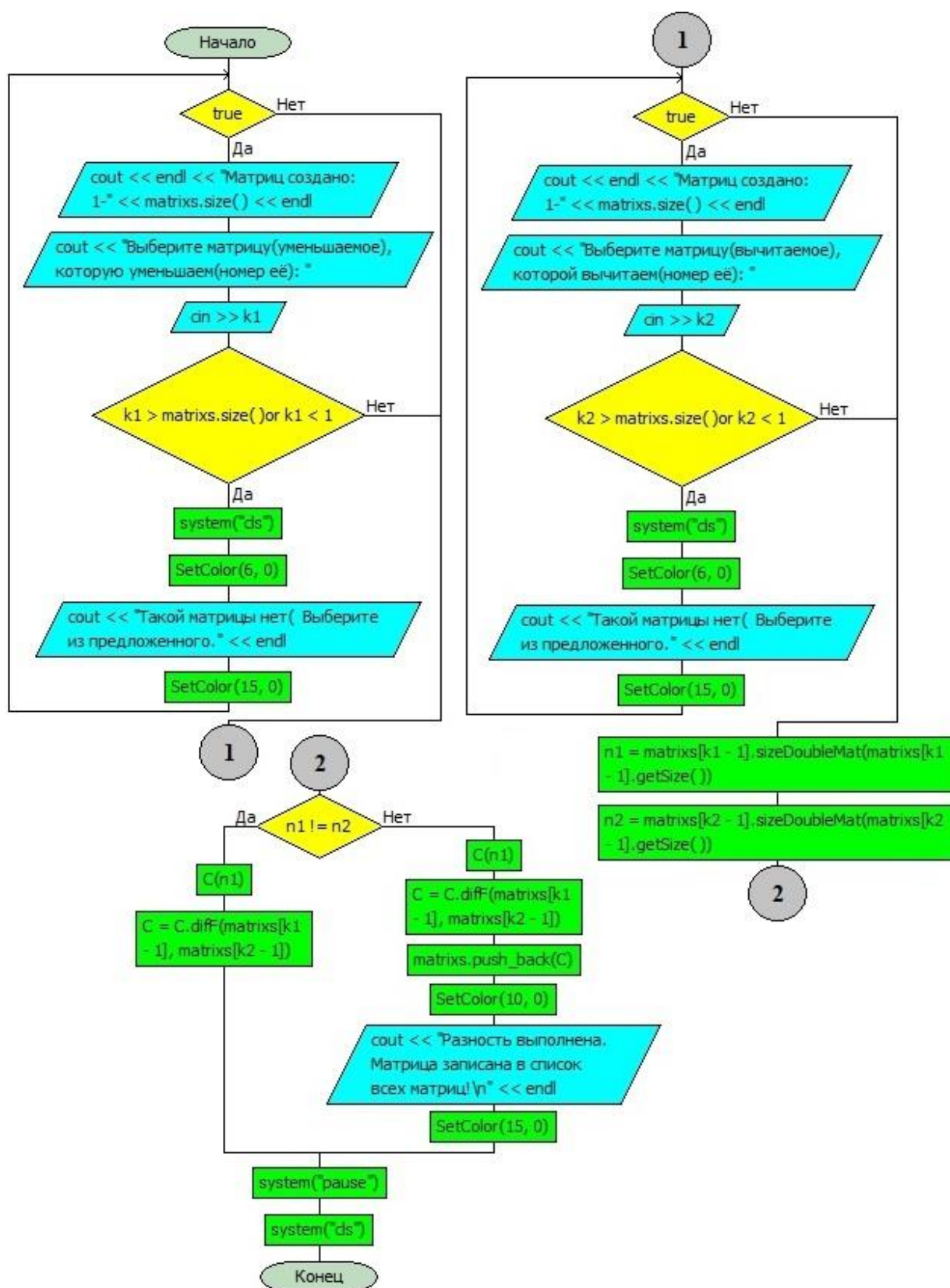


Рисунок 3.28 – Функция z5

На рисунке 3.29 изображена функция, которая предлагает пользователю выбор двух матриц для работы, после чего вызывает метод для проверки симметричности двух выбранных матриц.

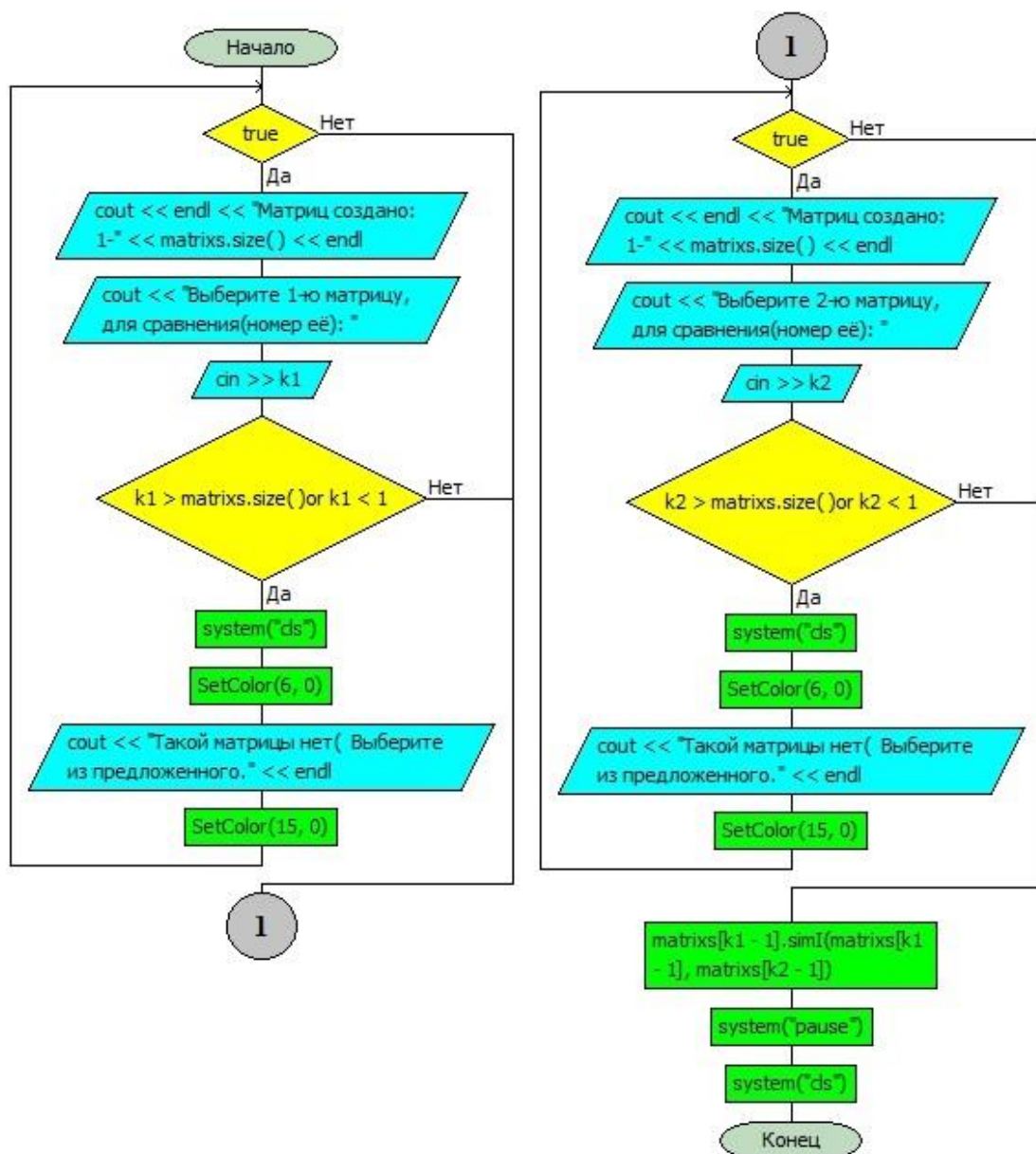


Рисунок 3.29 – Функция z6

На рисунке 3.30 изображена функция, которая предлагает пользователю выбор двух матриц для работы, после чего вызывает метод для присвоения значений второй матрицы в первую, которые выбрал пользователь.

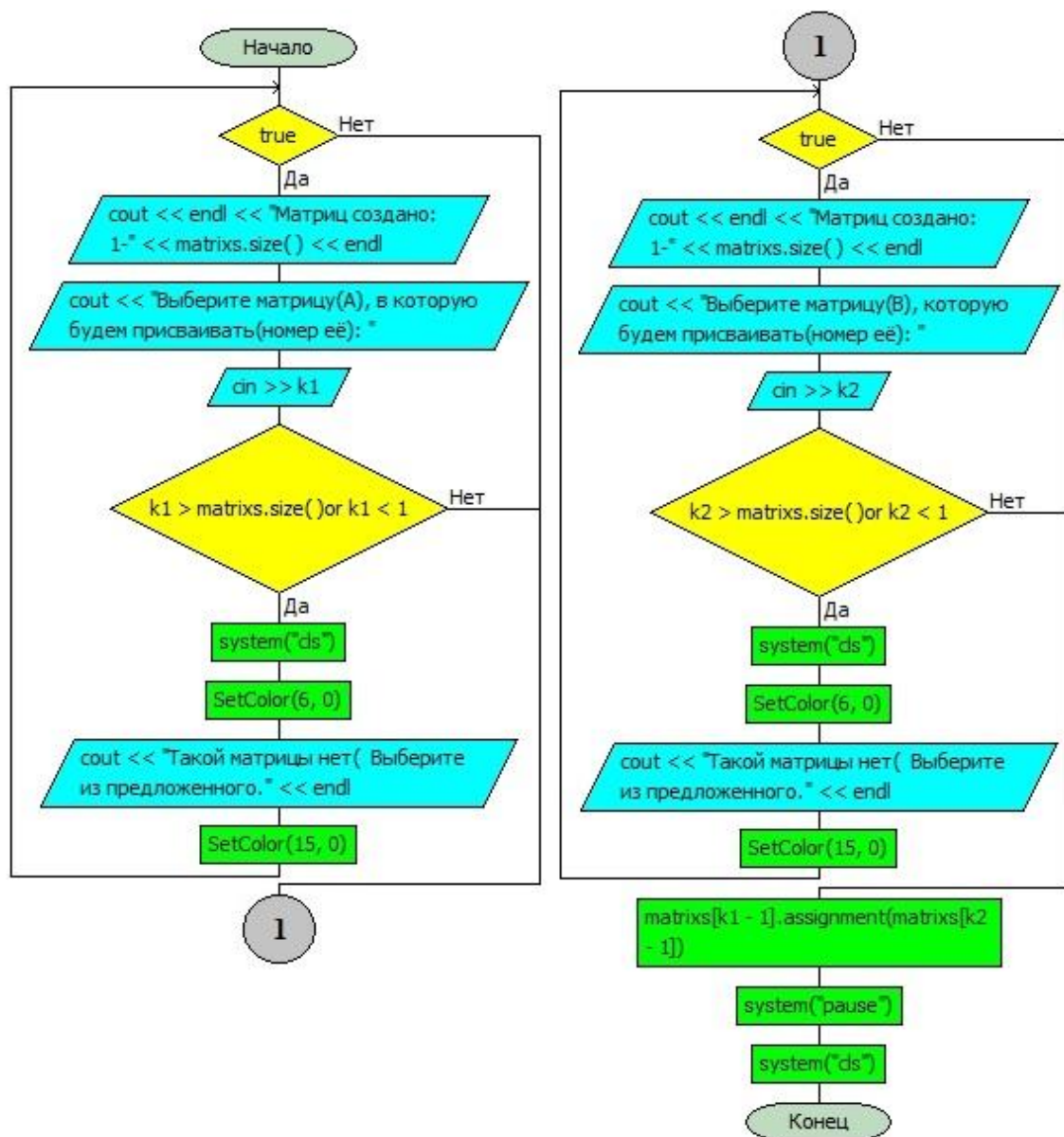


Рисунок 3.30 – Функция z7

На рисунке 3.31 изображена функция, которая отображает главное меню на консоли с количеством и названием пунктов, в зависимости от количества матриц. При выборе одного из пунктов пользователем будет вызвана соответствующая функция, которая имеет свой оригинальный алгоритм работы.

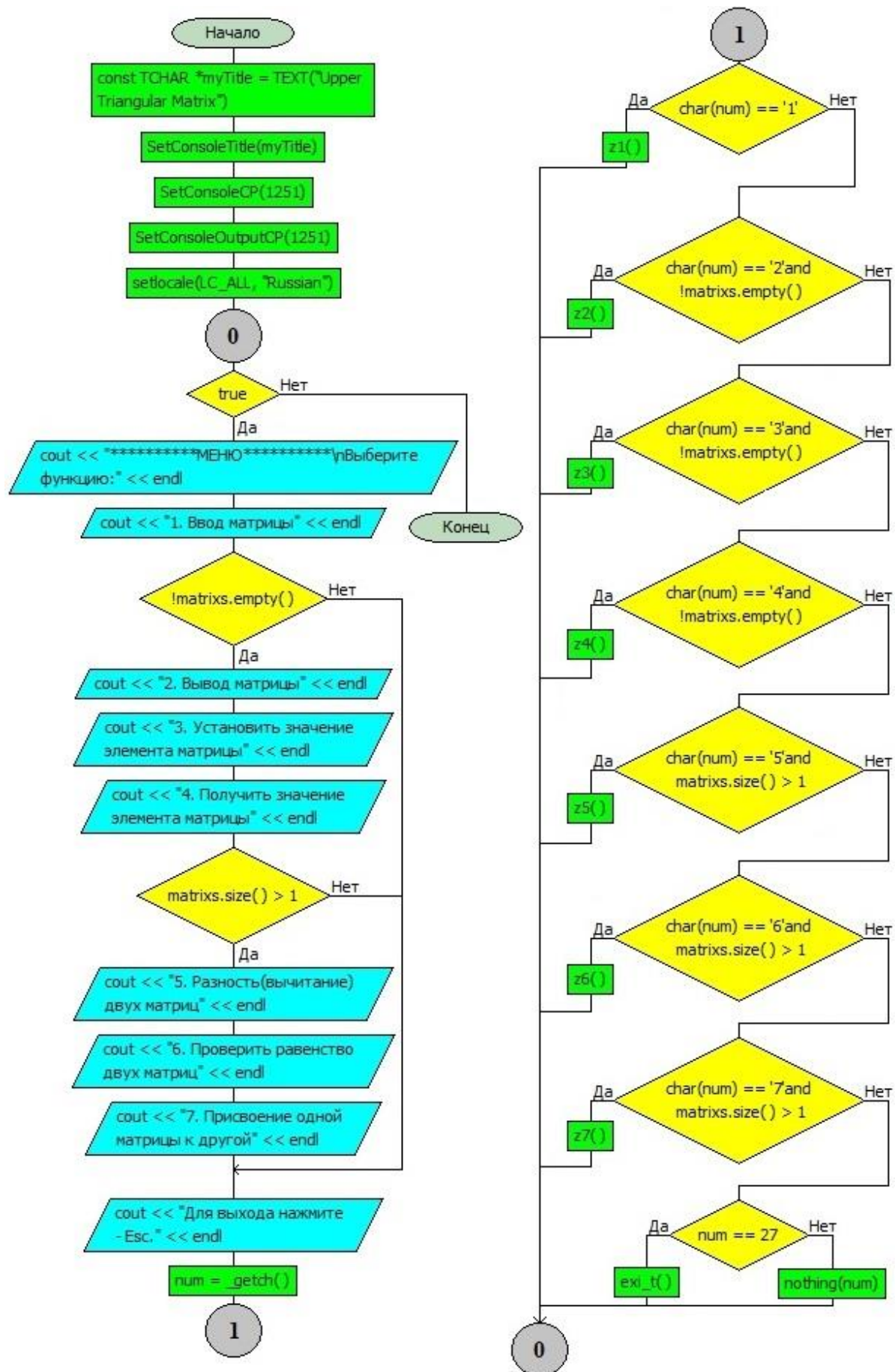


Рисунок 3.31 – Основная функция main

3.3.2 Структура программы с описанием функций составных частей исвязей между ними

Для описания структуры программы необходимо описать функционал написанной программы с использованием UML-диаграмм.

3.3.2.1 Класс матриц

Класс Matrix, реализующий сущность матриц, UML-диаграмма класса представлена на рисунке 3.32.

- поля:
 - `int n` – размер одномерного массива создаваемого для объекта;
 - `int* matrix` – ссылка на одномерный массив.
- методы:
 - `Matrix()` – конструктор класса;
 - `void setEl()` – метод, реализующий алгоритм изменения значения элемента матрицы;
 - `void getEl()` – метод, реализующий алгоритм вывода значения элемента матрицы;
 - `void intPut()` – метод, реализующий алгоритм ввода элементов матрицы с клавиатуры;
 - `void outPut()` – метод, реализующий алгоритм вывода матрицы на консоль;
 - `Matrix difF()` – метод, реализующий алгоритм вычисления разницы двух матриц;
 - `void simI()` – метод, реализующий алгоритм проверки матриц на равенство;
 - `void writeFile()` – метод, реализующий алгоритм записи матриц в текстовый файл;
 - `void readFile()` – метод, реализующий алгоритм чтения матриц из текстовых файлов;

- `void assignment()` – метод, реализующий алгоритм присвоения значений от одной матрице к другой;
- `void setN()` – метод, реализующий алгоритм изменения размера матрицы;
- `void sizeMat()` – метод, реализующий алгоритм возврата размера одномерного массива по размеру матрицы;
- `void index()` – метод, реализующий алгоритм получения индекса одномерного массива по индексам матрицы;
- `void sizeDoubleMat()` – метод, реализующий алгоритм возврата значения размера матрицы по размеру одномерного массива;
- `void getSize()` – метод, реализующий алгоритм возврата параметра матрицы.

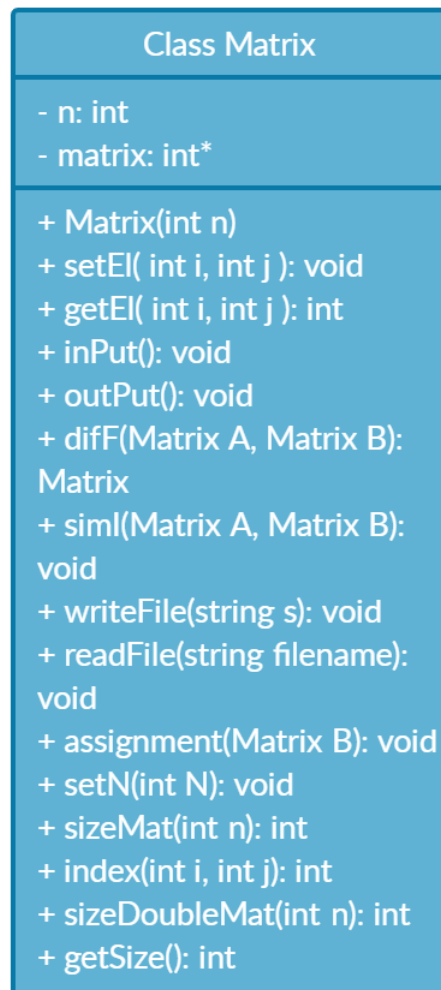


Рисунок 3.32 – UML-диаграмма класса Matrix

3.4 Технические средства, которые используются при работе программы

Для запуска данного программного продукта необходимо наличие графического адаптера. Чем больше видеопамати и оперативной памяти будет выделено для работы приложения, тем более быстро и корректно оно будет работать.

3.5 Вызов программы

При первом вызове программы пользователя приветствует окно с заголовком «Upper Triangular Matrix», и просит выбрать либо ввод матрицы, либо выход из программы, рисунок 3.33. Далее пользователю предлагается ряд действий, которые открываются в зависимости от количества матриц.

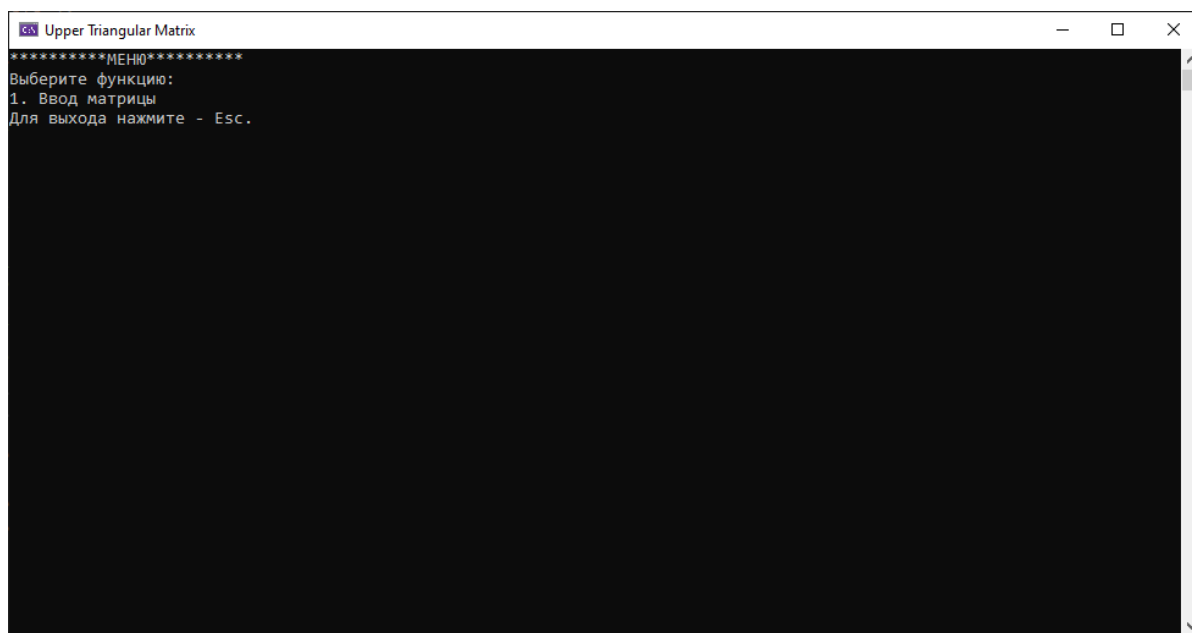


Рисунок 3.33 – Первый запуск программы

При выборе ввода матриц, будет предложен выбор ввода вручную с консоли или же из файла. Далее происходит ввод матриц и добавление их в вектор всех матриц. Файл, из которого считывается информация, должен находиться на рабочем столе компьютера. Возможные ошибки при вводе значений матрицы и названия файла также были предусмотрены. Все действия показаны на рисунках 3.34 – 3.36.

Upper Triangular Matrix

```
*****Ввод матрицы*****  
Выберите функцию:  
1. Ввод матрицы с клавиатуры  
2. Ввод матрицы из файла
```

Рисунок 3.34 – Меню ввода матриц

```
Upper Triangular Matrix  
*****Ввод матрицы*****  
Выберите функцию:  
1. Ввод матрицы с клавиатуры  
2. Ввод матрицы из файла  
  
Введите размер матрицы: 3  
Введите элементы матрицы построчно:  
1 2 3  
0 5 6  
0 0 9  
  
Матрица создана и добавлена в список матриц!  
  
Для продолжения нажмите любую клавишу . . .
```

Рисунок 3.35 – Ввод матрицы с клавиатуры

```
Upper Triangular Matrix  
*****Ввод матрицы*****  
Выберите функцию:  
1. Ввод матрицы с клавиатуры  
2. Ввод матрицы из файла  
  
Первое число в файле - это размер матрицы! Введите имя файла: matrixi  
Все ОК! Файл открыт!  
  
Матрица создана и добавлена в список матриц!  
  
Для продолжения нажмите любую клавишу . . .
```

Рисунок 3.36 – Ввод матрицы из файла

При добавлении одной матрицы станут доступны еще 2 пункта работы с матрицами, установка и получение значения элемента. При вводе двух матриц будут доступны все 7 пунктов работы программы, рисунок 3.37.

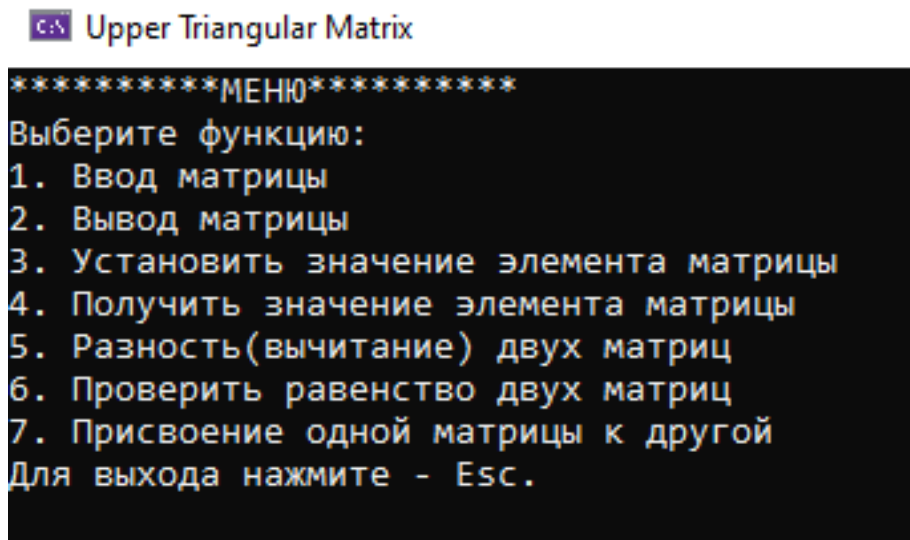


Рисунок 3.37 – Вид меню при двух и более матриц

При выборе установки значения элемента матрицы, пользователю будет предложено изменить значение одной из введенных им матриц, рисунок 3.38.

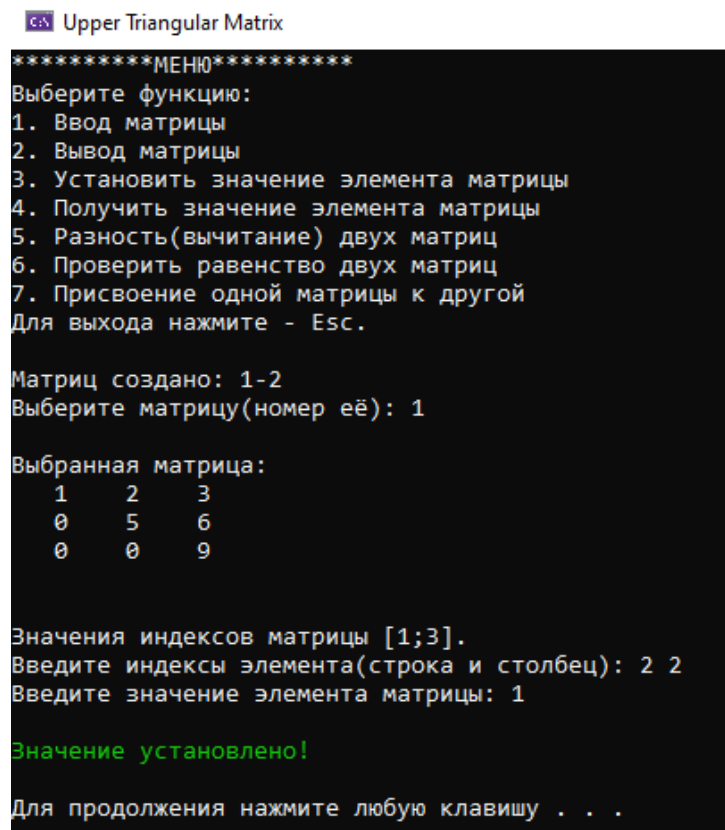


Рисунок 3.38 – Установка значения элемента матрицы

При выборе получения значения элемента матрицы, пользователю будет выведено значение выбранной им матрицы по двум индексам, рисунок 3.39.

```
Upper Triangular Matrix
*****МЕНЮ*****
Выберите функцию:
1. Ввод матрицы
2. Вывод матрицы
3. Установить значение элемента матрицы
4. Получить значение элемента матрицы
5. Разность(вычитание) двух матриц
6. Проверить равенство двух матриц
7. Присвоение одной матрицы к другой
Для выхода нажмите - Esc.

Матриц создано: 1-2
Выберите матрицу(номер её): 1

Значения индексов матрицы [1;3].
Введите индексы элемента(строка и столбец): 2 2

Значение матрицы[2;2] = 1

Для продолжения нажмите любую клавишу . . .
```

Рисунок 3.39 – Получение значения элемента матрицы

Если пользователь захочет узнать разность двух матриц, то ему достаточно выбрать пятый пункт. В данном случае, ему нужно будет выбрать над какими двумя матрицами будет выполняться операция вычитания, рисунок 3.40.

```
Upper Triangular Matrix
*****МЕНЮ*****
Выберите функцию:
1. Ввод матрицы
2. Вывод матрицы
3. Установить значение элемента матрицы
4. Получить значение элемента матрицы
5. Разность(вычитание) двух матриц
6. Проверить равенство двух матриц
7. Присвоение одной матрицы к другой
Для выхода нажмите - Esc.

Матриц создано: 1-3
Выберите матрицу(уменьшаемое), которую уменьшаем(номер её): 1

Матриц создано: 1-3
Выберите матрицу(вычитаемое), которой вычитаем(номер её): 3

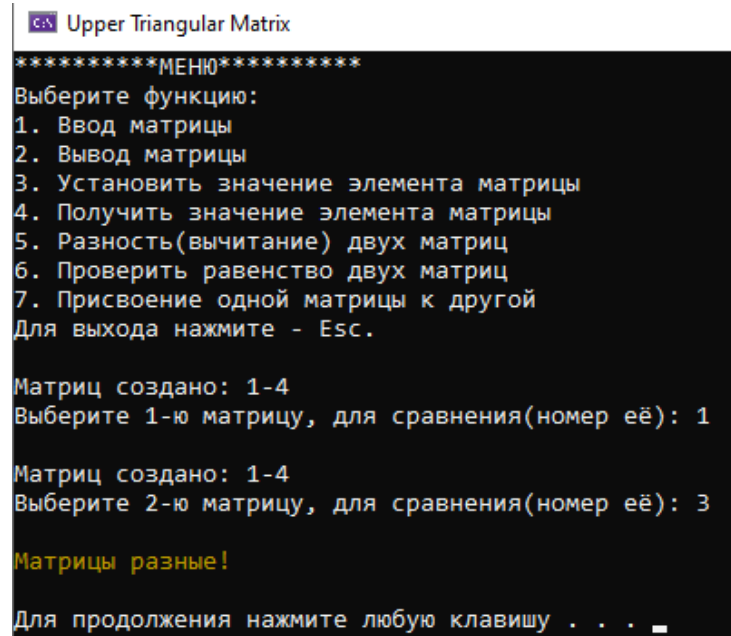
Разность двух матриц A и B:
  0  -1  1
  0  -6  -2
  0   0   8

Разность выполнена. Матрица записана в список всех матриц!

Для продолжения нажмите любую клавишу . . .
```

Рисунок 3.40 – Разность двух матриц одинакового размера

Если надо узнать одинаковые ли матрицы, то тут поможет шестой пункт. Нужно будет выбрать две матрицы, которые необходимо сравнить и будет высвечено сообщение, которое показывает соответственно разные они или одинаковые, рисунок 3.41.



```
Upper Triangular Matrix
*****МЕНЮ*****
Выберите функцию:
1. Ввод матрицы
2. Вывод матрицы
3. Установить значение элемента матрицы
4. Получить значение элемента матрицы
5. Разность(вычитание) двух матриц
6. Проверить равенство двух матриц
7. Присвоение одной матрицы к другой
Для выхода нажмите - Esc.

Матриц создано: 1-4
Выберите 1-ю матрицу, для сравнения(номер её): 1

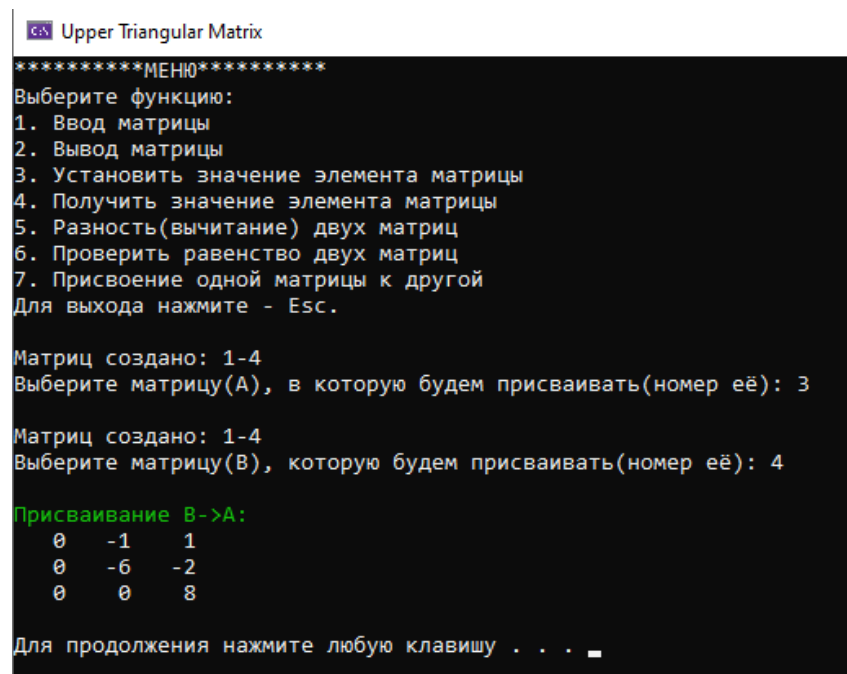
Матриц создано: 1-4
Выберите 2-ю матрицу, для сравнения(номер её): 3

Матрицы разные!

Для продолжения нажмите любую клавишу . . .
```

Рисунок 3.41 – Проверка равенства двух матриц

При выборе пункта присвоение матрицы, пользователю будет нужно выбрать две матрицы, с которыми будет выполняться операция присваивания, рисунок 3.42.



```
Upper Triangular Matrix
*****МЕНЮ*****
Выберите функцию:
1. Ввод матрицы
2. Вывод матрицы
3. Установить значение элемента матрицы
4. Получить значение элемента матрицы
5. Разность(вычитание) двух матриц
6. Проверить равенство двух матриц
7. Присвоение одной матрицы к другой
Для выхода нажмите - Esc.

Матриц создано: 1-4
Выберите матрицу(A), в которую будем присваивать(номер её): 3

Матриц создано: 1-4
Выберите матрицу(B), которую будем присваивать(номер её): 4

Присваивание B->A:
0  -1  1
0  -6  -2
0   0   8

Для продолжения нажмите любую клавишу . . .
```

Рисунок 3.42 – Присвоение одной матрицы к другой

При выборе вывода матриц, будет предложен выбор вывода на консоль или же в файл, который будет создан на рабочем столе компьютера. Далее происходит выбор матрицы, которую необходимо вывести. Данные операции изображены на рисунках 3.43 – 3.45.

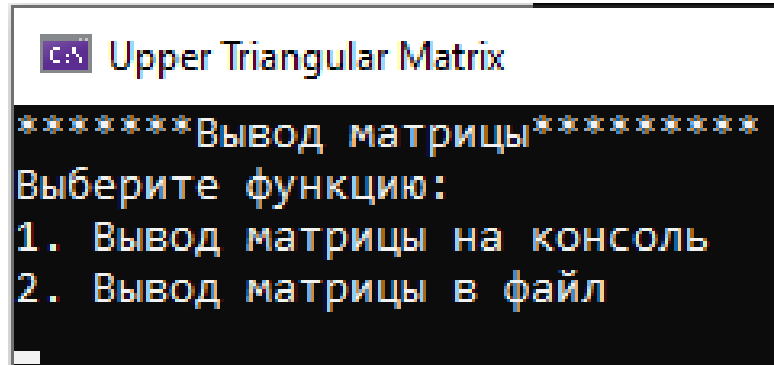


Рисунок 3.43 – Меню вывода матриц

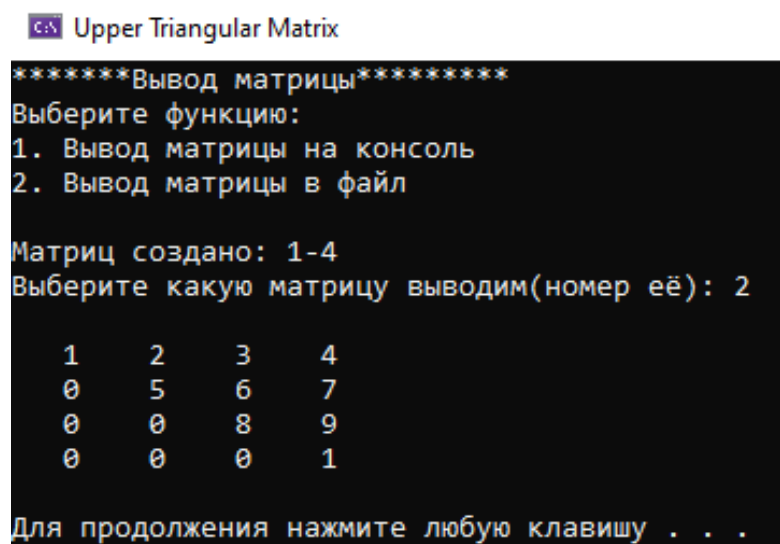


Рисунок 3.44 – Вывод матрицы на консоль

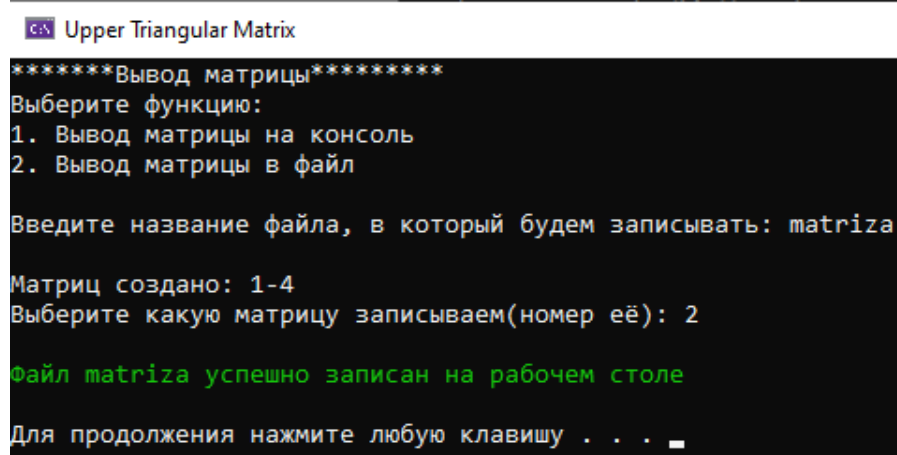


Рисунок 3.45 – Вывод матрицы в файл

Если пользователь захочет выйти из программы, ему достаточно нажать клавишу “Esc”, расположенную в верхнем левом углу клавиатуры. После нажатия данной клавиши, будет выведено сообщение о прекращении работы программы, рисунок 3.46.

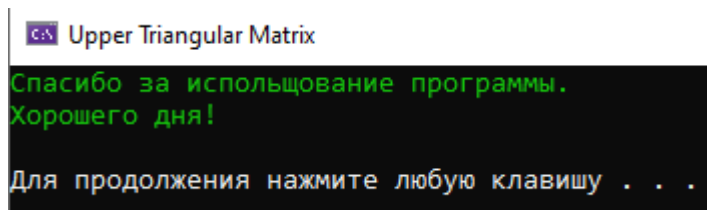


Рисунок 3.46 – Выход из программы

Все файлы, которые были до работы программы и созданные в процессе работы программы сохранены на рабочем столе и содержат информацию о вводимых матрицах и тех матрицах, которые пользователь захотел вывести. Файлы изображены на рисунках 3.47 – 3.48.

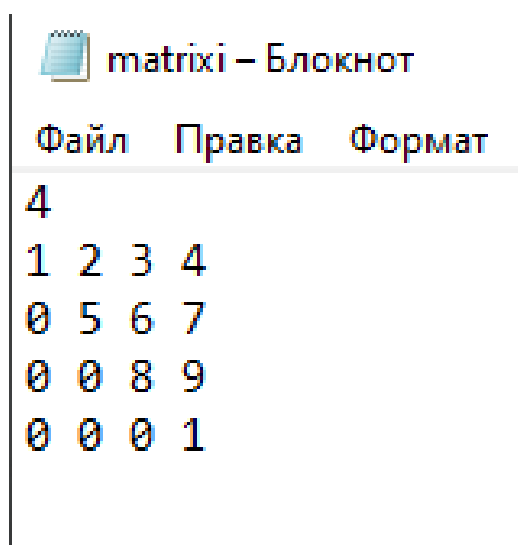


Рисунок 3.47 – Изначальный файл matrixi

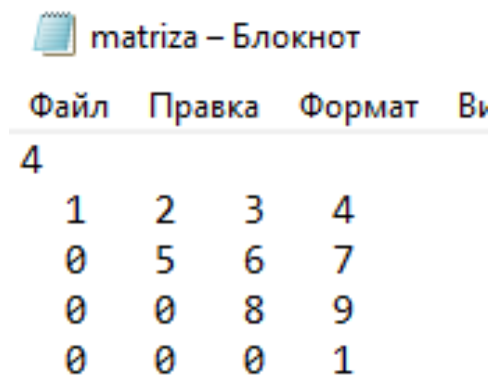


Рисунок 3.48 – Файл matriza, созданный в процессе работы программы

Возможные ошибки при работе с пользователем также были учтены и показаны на рисунках 3.49 – 3.53.

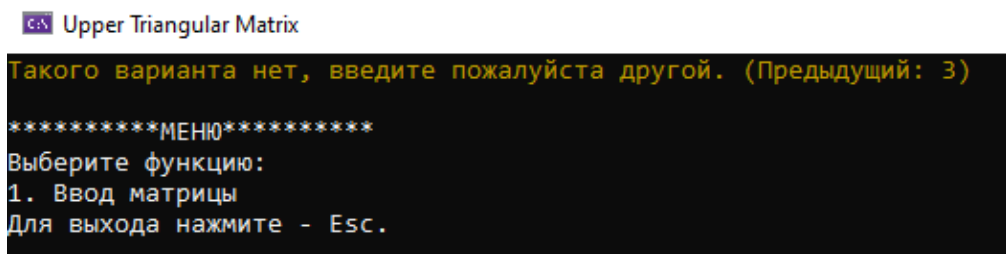


Рисунок 3.49 – Ошибка при вводе пункта, которого нет

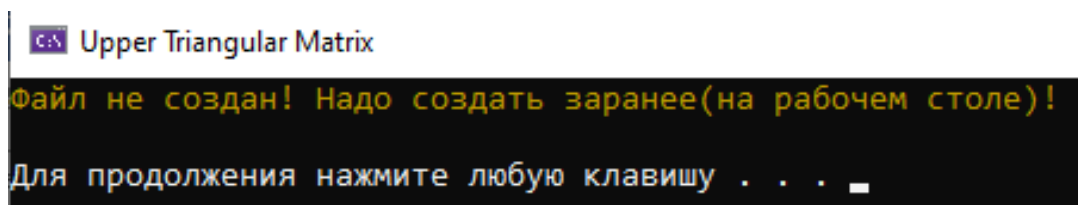


Рисунок 3.50 – Ошибка при попытке ввода из несуществующего файла

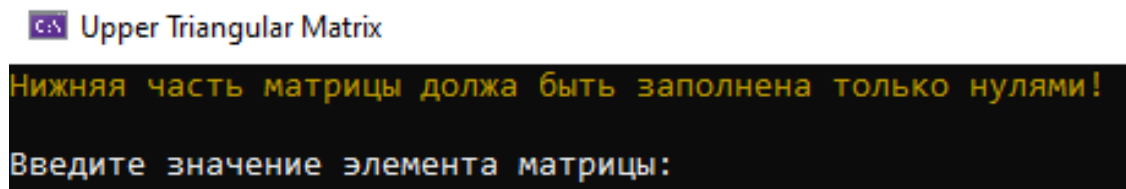


Рисунок 3.51 – Ошибка при установке значения нижней части не равного 0

```
Upper Triangular Matrix
*****Вывод матрицы*****
Выберите функцию:
1. Вывод матрицы на консоль
2. Вывод матрицы в файл

Матриц создано: 1-3
Выберите какую матрицу выводим(номер её): 4

Такой матрицы нет( Желаете вывести их все?
1. Да!
2. Ноу
3. Ввести другой номер

1)
  4   6   5   2
  0   5   3   2
  0   0   1   1
  0   0   0   9

2)
  1   2   3   4
  0   5   6   7
  0   0   8   9
  0   0   0   1

3)
  1   2   3
  0   4   5
  0   0   6

Для продолжения нажмите любую клавишу . . .
```

Рисунок 3.52 – Ошибка при выборе матрицы, которая не создана

```
Upper Triangular Matrix
*****МЕНЮ*****
Выберите функцию:
1. Ввод матрицы
2. Вывод матрицы
3. Установить значение элемента матрицы
4. Получить значение элемента матрицы
5. Разность(вычитание) двух матриц
6. Проверить равенство двух матриц
7. Присвоение одной матрицы к другой
Для выхода нажмите - Esc.

Матриц создано: 1-3
Выберите матрицу(A), в которую будем присваивать(номер её): 1

Матриц создано: 1-3
Выберите матрицу(B), которую будем присваивать(номер её): 3

Матрицы разные по размерам! Присваивание невозможно!

Для продолжения нажмите любую клавишу . . .
```

Рисунок 3.53 – Ошибка при работе с матрицами разных размеров

3.6 Входные данные

В качестве входных данных, в зависимости от действия, программа просит ввести либо целочисленный тип данных матрицы (размер и значения), либо название файла – строчный тип данных. Также для выбора действия в меню тоже необходимо вводить целочисленные данные.

3.7 Выходные данные

В ходе выполнения сеанса пользователем, программа отображает на экране данные верхних треугольных матриц, действия пользователя, а также возможности, предоставляемые пользователю. Результаты работы программы предоставляется записывать в текстовый файл с расширением «.txt», который находится на рабочем столе.

ЗАКЛЮЧЕНИЕ

В процессе решения данной задачи были получены навыки в разработке классов описывающих сущностей, разработки общего функционала в рамках объектов приложения. Был освоен алгоритм обработки структур данных в виде линейной структуры вектор. Также были использованы основные алгоритмы для обработки данных, например, чтение из файла или запись в него. Освоены элементы логического анализа данных и реализация алгоритма решения задачи для поставленной задачи. Освоены основные понятия ООП и способы написания программы- приложения на языке C++.

Была разработана программа, с операциями, представленными в приложении А.

СПИСОК ЛИТЕРАТУРЫ

1. Кубенский А.А. Структуры и алгоритмы обработки данных: объектноориентированный подход и реализация на С++ / А.А. Кубенский.— М.: БХВ-Петербург, 2017.— 300 с.
2. Стивен Прата. Язык программирования С++ (С++11). Лекции и упражнения, 6-е издание — М.: Вильямс, 2012. — 1248 с.
3. Седжвик, Р. Алгоритмы на С++ / Р. Седжвик.— М.:Вильямс, 2017.— 1056 с.
4. Шилдт Г. С++: базовый курс. 3-е изд. Пер. с англ. — М.: Вильямс, 2017
5. <https://habr.com/ru/post/337594/> — электронный ресурс, дата обращения 20.10.2021
6. Антик М.И. Дискретная математика [Электронный ресурс]: Учебное пособие /Антик М.И., Казанцева Л.В. — М.: МИРЭА — Российский технологический университет, 2018 — 1 электрон. опт. диск (CD-ROM).
7. Видео лекции по курсу «Объектно-ориентированное программирование» [Электронный ресурс]. АСО «Аврора».
8. Васильев А.Н. Объектно-ориентированное программирование на С++. Издательство: Наука и Техника. Санкт-Петербург, 2016г. 543 стр.

ПРИЛОЖЕНИЕ А

Исходный код программы представлен в Листингах А.1, А.2.

Листинг А.1 – CourseWork.h

```
#ifndef COURSEWORK_H
#define COURSEWORK_H
#define _CRT_SECURE_NO_WARNINGS //for function getenv

#include <iostream>
#include <string>
#include <fstream>
#include <Windows.h>
#include <iomanip>
#include <vector>
#include <conio.h>

using namespace std;

class Matrix {
    int n;
    int* matrix;

public:
    Matrix(int n);
    void setEl(int i, int j);
    int getEl(int i, int j);
    void inPut();
    void outPut();
    Matrix difF(Matrix A, Matrix B);
    void simI(Matrix A, Matrix B);
    void writeFile(string s);
    void readFile(string filename);
    void assignment(Matrix B);

    void setN(int N);
    int sizeMat(int n);
    int index(int i, int j);
    int sizeDoubleMat(int n);
    int getSize();
};

#endif
```

Листинг А.2 – CourseWork.cpp

```
// CourseWork.cpp : Этот файл содержит функцию "main". Здесь начинается и
заканчивается выполнение программы.
// Вариант 14

#include "CourseWork.h"

using namespace std;

string location = string(getenv("USERPROFILE")) + "\\Desktop\\"; //Global
Путь!
//string location = "C:\\Users\\Dima\\Desktop\\"; //Global Путь!
vector < Matrix > matrixs; //Global переменная

void SetColor(int text, int background) //Установка цвета
```

```

{
    HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(hStdOut, (WORD)((background << 4) | text));
}

int Matrix::sizeMat(int n) { //Размер одномерного массива по размеру матрицы
    int s = 0;
    for (int i = 1; i <= n; i++) {
        s += i;
    }
    return s;
}

int Matrix::sizeDoubleMat(int n) { //Размер матрицы по размеру одномерного массива
    for (int i = 1; i <= n; i++) {
        n -= i;
        if (n <= 0)
            return i;
    }
}

int Matrix::index(int i, int j) { //Получение индекса одномерного массива по матрице
    int s = 0;
    int h = sizeDoubleMat(this->n);
    for (int k = 0; k < h; k++) {
        for (int f = k; f < h; f++) {
            if (k == i && f == j)
                return s;
            else
                s++;
        }
    }
}

int Matrix::getSize() { //Возврат значения размера массива
    return this->n;
}

Matrix::Matrix(int n) //Создание верхней треугольной матрицы заданного размера с нулевыми значениями всех элементов (Конструктор)
{
    n = sizeMat(n);
    this->n = n;
    int* matrix = new int[n];
    for (int i = 0; i < n; i++) {
        matrix[i] = 0;
    }
    this->matrix = matrix;
}

void Matrix::setEl(int i, int j) { //Установка значения элемента матрицы с индексами i, j;
    int a, s;
    while (true) {
        cout << "Введите значение элемента матрицы: ";
        cin >> a;
        if (j < i and a != 0) {
            system("cls");
            SetColor(6, 0);
            cout << "Нижняя часть матрицы должна быть заполнена только нулями!\n" << endl;
            SetColor(15, 0);
        }
    }
}

```

```

        }
        else
            break;
    }
    if (i <= j) {
        s = index(i, j);
        this->matrix[s] = a;
    }
}

int Matrix::getEl(int i, int j) { //Получение значения элемента матрицы с
индексами i, j;
    if (i <= j) {
        int s;
        s = index(i, j);
        return this->matrix[s];
    }
    else
        return 0;
}

void Matrix::inPut() //Построчный ввод матрицы;
{
    int h = sizeDoubleMat(this->n);
    while (true) {
        cout << "Введите элементы матрицы построчно: " << endl;
        int a;
        bool flag = true;
        for (int i = 0; i < h; i++) {
            for (int j = 0; j < h; j++) {
                cin >> a;
                if (j < i && a != 0)
                    flag = false;
                else if (j >= i) {
                    int s = index(i, j);
                    this->matrix[s] = a;
                }
            }
        }
        if (flag)
            break;
        else {
            system("cls");
            SetColor(6, 0);
            cout << "Нижняя часть матрицы должна быть заполнена только
нулями!\n" << endl;
            cout << "Введите значения матрицы снова (размер матрицы = " << h <<
"): \n " << endl;
            SetColor(15, 0);
        }
    }
}

void Matrix::outPut() //Построчный вывод на экран матрицы;
{
    int h = sizeDoubleMat(this->n);
    for (int i = 0; i < h; i++) {
        for (int j = 0; j < h; j++) {
            if (j >= i) {
                int s = index(i, j);
                cout << setw(4) << this->matrix[s] << " ";
            }
            else

```

```

        cout << setw(4) << 0 << " ";
    }
    cout << endl;
}
cout << endl;
}

Matrix Matrix::difF(Matrix A, Matrix B) { //Разность двух треугольных матриц;
    if (A.n != B.n) {
        SetColor(6, 0);
        cout << "Разность двух матриц с различными размерами невозможна!\n" <<
endl;
        SetColor(15, 0);
        Matrix C(n);
        return C;
    }
    int h = sizeDoubleMat(A.n);
    Matrix C(h);
    for (int i = 0; i < A.n; i++) {
        C.matrix[i] = A.matrix[i] - B.matrix[i];
    }
    SetColor(10, 0);
    cout << "\nРазность двух матриц A и B: " << endl;
    SetColor(15, 0);
    C.outPut();
    return C;
}

void Matrix::simI(Matrix A, Matrix B) { //Проверка равенства двух матриц;
    if (A.n != B.n) {
        SetColor(6, 0);
        cout << "\nМатрицы разные по размерам!\n" << endl;
        SetColor(15, 0);
        return;
    }
    for (int i = 0; i < n; i++) {
        if (A.matrix[i] != B.matrix[i]) {
            SetColor(6, 0);
            cout << "\nМатрицы разные!\n" << endl;
            SetColor(15, 0);
            return;
        }
    }
    SetColor(10, 0);
    cout << "\nМатрицы одинаковые\n" << endl;
    SetColor(15, 0);
    return;
}

void Matrix::writeFile(string filename) { //Запись матрицы в файл (первое
число файла - размер матрицы);
    ofstream file;
    if (filename.length() > 4)
    {
        if (filename.substr(filename.length() - 4) == ".txt")
            filename = filename.erase(filename.length() - 4);
    }
    file.open(location + filename + ".txt");
    int h = sizeDoubleMat(this->n);
    file << h << "\n";
    for (int i = 0; i < h; i++) {
        for (int j = 0; j < h; j++) {
            if (j >= i) {
                int s = index(i, j);

```

```

        file << setw(3) << this->matrix[s] << " ";
    }
    else
        file << setw(3) << 0 << " ";
    }
    file << "\n";
}
file.close(); //Закрытие файла
file.clear(); // Окончили запись файла
SetColor(10, 0);
cout << "Файл " << filename << " успешно записан на рабочем столе\n" <<
endl;
SetColor(15, 0);
}

bool openFile(string filename) { //Существование файла
    if (filename.length() > 4)
    {
        if (filename.substr(filename.length() - 4) == ".txt")
            filename = filename.erase(filename.length() - 4);
    }
    ifstream file;
    file.open(location + filename + ".txt");
    if (!file)
        return false;
    else
        return true;
}

void Matrix::readFile(string filename) { //Чтение матрицы из текстового файла
(первое число файла - размер матрицы);
    string input;
    if (filename.length() > 4)
    {
        if (filename.substr(filename.length() - 4) == ".txt")
            filename = filename.erase(filename.length() - 4);
    }
    ifstream file;
    file.open(location + filename + ".txt");

    if (!file) { //Файл не открыт
        system("cls");
        SetColor(6, 0);
        cout << "Файл не создан! Надо создать заранее(на рабочем столе)!\n" <<
endl;
        SetColor(15, 0);
        return;
    }
    else { //Файл открыт
        SetColor(10, 0);
        cout << "Все ОК! Файл открыт!\n\n";
        SetColor(15, 0);
    }

    int i = 0;
    getline(file, input);
    int k = atoi(input.c_str());
    this->setN(k);
    while (!file.eof())
    {
        getline(file, input);
        if (input != "") {
            for (int j = 0; j < k; j++) {

```

```

        int a = input.find(" ");
        string str = input.substr(0, a);
        if (j >= i) {
            int s = index(i, j);
            this -> matrix[s] = atoi(str.c_str());
        }
        input.erase(0, a + 1);
    }
    i++;
}

}
file.close(); //Закрытие файла
matrixs.push_back(*this);
SetColor(10, 0);
cout << "Матрица создана и добавлена в список матриц!\n" << endl;
SetColor(15, 0);
}

void Matrix::assignment(Matrix B) { //Присваивание одной матрицы другой.
    if (this->n != B.n) {
        SetColor(6, 0);
        cout << "\nМатрицы разные по размерам! Присваивание невозможно!\n" <<
endl;
        SetColor(15, 0);
        return;
    }
    for (int i = 0; i < n; i++) {
        this -> matrix[i] = B.matrix[i];
    }
    SetColor(10, 0);
    cout << "\nПрисваивание B->A: " << endl;
    SetColor(15, 0);
    this->outPut();
}

void Matrix::setN(int N) { //Изменение параметров(размера) матрицы
    N = sizeMat(N);
    this -> n = N;
    int* matrix_1 = new int[n];
    for (int i = 0; i < n; i++) {
        matrix_1[i] = 0;
    }
    this -> matrix = matrix_1;
}

int vvodSizeMat() { //Проверка вводимого размера массива
    string str;
    while (true) {
        bool flag = true;
        cout << "\nВведите размер матрицы: ";
        cin >> str;
        for (int i = 0; i < str.length(); i++) {
            if (!isdigit(str[i]))
                flag = false;
        }
        if (flag) {
            int n = atoi(str.c_str());
            if (n > 0)
                return n;
            else {
                system("cls");
                SetColor(6, 0);
                cout << "Надо ввести целое положительное(n > 0) число!" <<
endl;
            }
        }
    }
}

```

```

        SetColor(15, 0);
    }
}
else {
    system("cls");
    SetColor(6, 0);
    cout << "Надо ввести целое положительное(n > 0) число!" << endl;
    SetColor(15, 0);
}
}
}

int getchOption(string s, int n) { //Выбор развития программы
    int num;
    while (true) {
        cout << s << endl;
        num = _getch();
        for (int i = 49; i < 49 + n; i++) { //char(49) == int(1)
            if (num == i)
                return num;
        }
        system("cls");
        SetColor(6, 0);
        cout << "Такого варианта нет, введите пожалуйста другой! (Предыдущий: " << char(num) << ")\n" << endl;
        SetColor(15, 0);
    }
}

void z1() { //Ввод
    system("cls");
    string option = "*****Ввод матрицы*****\nВыберите функцию:\n1. Ввод матрицы с клавиатуры\n2. Ввод матрицы из файла";
    int num = getchOption(option, 2);
    if (char(num) == '1') { //Клавиатура
        int n = vvodSizeMat();
        Matrix A(n);
        A.inPut();
        matrixs.push_back(A);
        SetColor(10, 0);
        cout << "\nМатрица создана и добавлена в список матриц!\n" << endl;
        SetColor(15, 0);
        system("pause");
        system("cls");
    }
    else if (char(num) == '2') { //Файл
        string filename;
        cout << endl << "Первое число в файле - это размер матрицы! ";
        cout << "Введите имя файла: ";
        cin >> filename;
        Matrix A(1);
        A.readFile(filename);
        system("pause");
        system("cls");
    }
}

void z2_1() { //Вывод в консоль
    if (matrixs.size() > 1) {
        while (true) {
            int k;
            cout << endl << "Матриц создано: 1-" << matrixs.size() << endl;
            cout << "Выберите какую матрицу выводим(номер её): ";
            cin >> k;

```

```

        if (k > matrixs.size() or k < 1) {
            string option = "\nТакой матрицы нет( Желаете вывести их
все?\n1. Да!\n2. Ной\n3. Ввести другой номер";
            int num = getchOption(option, 3);
            if (char(num) == '1') {
                cout << endl;
                for (int i = 0; i < matrixs.size(); i++) {
                    cout << i + 1 << " " << endl;
                    matrixs[i].outPut();
                }
                break;
            }
            else if (char(num) == '2')
                break;
            else if (char(num) == '3')
                system("cls");
        }
        else {
            cout << endl;
            matrixs[k - 1].outPut();
            break;
        }
    }
}
else {
    cout << endl;
    matrixs[0].outPut();
}
system("pause");
system("cls");
}

void z2_2() { //Вывод в файл
    string filename;
    cout << endl << "Введите название файла, в который будем записывать: ";
    cin >> filename;
    if (matrixs.size() > 1) {
        while (true) {
            int k;
            cout << endl << "Матриц создано: 1-" << matrixs.size() << endl;
            cout << "Выберите какую матрицу записываем(номер её): ";
            cin >> k;
            if (k > matrixs.size() or k < 1) {
                string option = "\nТакой матрицы нет( Желаете записать их
все?\n1. Да!\n2. Ной\n3. Ввести другой номер";
                int num = getchOption(option, 3);
                if (char(num) == '1') {
                    matrixs[0].writeFile(filename);
                    for (int i = 1; i < matrixs.size(); i++) {
                        cout << i + 1 << " " << "Введите название файла, в который
будем записывать: ";
                        cin >> filename;
                        matrixs[i].writeFile(filename);
                    }
                    SetColor(10, 0);
                    cout << "\nВсе матрицы записаны!\n" << endl;
                    SetColor(15, 0);
                    break;
                }
                else if (char(num) == '2')
                    break;
                else if (char(num) == '3')
                    system("cls");
            }
        }
    }
}

```



```

        else {
            cout << endl;
            matrixs[k - 1].writeFile(filename);
            break;
        }
    }
}
else {
    cout << endl;
    matrixs[0].writeFile(filename);
}
system("pause");
system("cls");
}

void z2() { //Вывод
    system("cls");
    string option = "*****Вывод матрицы*****\nВыберите функцию:\n1.
Вывод матрицы на консоль\n2. Вывод матрицы в файл";
    int num = getchOption(option, 2);
    if (char(num) == '1') //Консоль
        z2_1();

    else if (char(num) == '2') //Файл
        z2_2();
}

void z3() { //Установка значения элемента
    int i, j, k = 1;
    if (matrixs.size() > 1) {
        while (true) {
            cout << endl << "Матриц создано: 1-" << matrixs.size() << endl;
            cout << "Выберите матрицу(номер её): ";
            cin >> k;
            if (k > matrixs.size() or k < 1) {
                system("cls");
                SetColor(6, 0);
                cout << "Такой матрицы нет(  Выберите из предложенного." <<
endl;
                SetColor(15, 0);
            }
            else
                break;
        }
    }
    cout << "\nВыбранная матрица: " << endl;
    matrixs[k - 1].outPut();
    while (true) {
        int n = matrixs[k - 1].sizeDoubleMat(matrixs[k - 1].getSize());
        cout << "\nЗначения индексов матрицы [1;" << n << "]." << endl;
        cout << "Введите индексы элемента(строка и столбец): ";
        cin >> i >> j;
        if (i > n or j > n or i < 1 or j < 1) {
            system("cls");
            SetColor(6, 0);
            cout << "Индексы элемента должны быть меньше размера матрицы!\n"
<< endl;
            SetColor(15, 0);
        }
        else
            break;
    }
    matrixs[k - 1].setEl(i - 1, j - 1);
    SetColor(10, 0);
}

```

```

        cout << "\nЗначение установлено!\n" << endl;
        SetColor(15, 0);
        system("pause");
        system("cls");
    }

void z4() { //Получение значения элемента
    int i, j, k = 1;
    if (matrixs.size() > 1) {
        while (true) {
            cout << endl << "Матриц создано: 1-" << matrixs.size() << endl;
            cout << "Выберите матрицу(номер её): ";
            cin >> k;
            if (k > matrixs.size() or k < 1) {
                system("cls");
                SetColor(6, 0);
                cout << "Такой матрицы нет( Выберите из предложенного." <<
endl;
                SetColor(15, 0);
            }
            else
                break;
        }
        while (true) {
            int n = matrixs[k - 1].sizeDoubleMat(matrixs[k - 1].getSize());
            cout << "\nЗначения индексов матрицы [1;" << n << "]." << endl;
            cout << "Введите индексы элемента(строка и столбец): ";
            cin >> i >> j;
            if (i > n or j > n or i < 1 or j < 1) {
                system("cls");
                SetColor(6, 0);
                cout << "Индексы элемента должны быть меньше размера матрицы!\n"
<< endl;
                SetColor(15, 0);
            }
            else
                break;
        }
        int z = matrixs[k - 1].getEl(i - 1, j - 1);
        SetColor(10, 0);
        cout << "\nЗначение матрицы[" << i << ";" << j << "] = " << z << endl <<
endl;
        SetColor(15, 0);
        system("pause");
        system("cls");
    }

void z5() { //Вычитание двух матриц(разность)
    int k1, k2;
    while (true) {
        cout << endl << "Матриц создано: 1-" << matrixs.size() << endl;
        cout << "Выберите матрицу(уменьшаемое), которую уменьшаем(номер её):
";
        cin >> k1;
        if (k1 > matrixs.size() or k1 < 1) {
            system("cls");
            SetColor(6, 0);
            cout << "Такой матрицы нет( Выберите из предложенного." << endl;
            SetColor(15, 0);
        }
        else
            break;
    }
}

```

```

while (true) {
    cout << endl << "Матриц создано: 1-" << matrixs.size() << endl;
    cout << "Выберите матрицу(вычитаемое), которой вычитаем(номер её): ";
    cin >> k2;
    if (k2 > matrixs.size() or k2 < 1) {
        system("cls");
        SetColor(6, 0);
        cout << "Такой матрицы нет( Выберите из предложенного." << endl;
        SetColor(15, 0);
    }
    else
        break;
}
int n1 = matrixs[k1 - 1].sizeDoubleMat(matrixs[k1 - 1].getSize());
int n2 = matrixs[k2 - 1].sizeDoubleMat(matrixs[k2 - 1].getSize());
if (n1 != n2) {
    Matrix C(n1);
    C = C.difF(matrixs[k1 - 1], matrixs[k2 - 1]);
}
else {
    Matrix C(n1);
    C = C.difF(matrixs[k1 - 1], matrixs[k2 - 1]);
    matrixs.push_back(C);
    SetColor(10, 0);
    cout << "Разность выполнена. Матрица записана в список всех матриц!\n"
<< endl;
    SetColor(15, 0);
}
system("pause");
system("cls");
}

void z6() { //Проверка равенства матриц
    int k1, k2;
    while (true) {
        cout << endl << "Матриц создано: 1-" << matrixs.size() << endl;
        cout << "Выберите 1-ю матрицу, для сравнения(номер её): ";
        cin >> k1;
        if (k1 > matrixs.size() or k1 < 1) {
            system("cls");
            SetColor(6, 0);
            cout << "Такой матрицы нет( Выберите из предложенного." << endl;
            SetColor(15, 0);
        }
        else
            break;
    }
    while (true) {
        cout << endl << "Матриц создано: 1-" << matrixs.size() << endl;
        cout << "Выберите 2-ю матрицу, для сравнения(номер её): ";
        cin >> k2;
        if (k2 > matrixs.size() or k2 < 1) {
            system("cls");
            SetColor(6, 0);
            cout << "Такой матрицы нет( Выберите из предложенного." << endl;
            SetColor(15, 0);
        }
        else
            break;
    }
    matrixs[k1 - 1].simI(matrixs[k1 - 1], matrixs[k2 - 1]);
    system("pause");
    system("cls");
}

```

```

void z7() { //Присвоение матриц
    int k1, k2;
    while (true) {
        cout << endl << "Матриц создано: 1-" << matrixs.size() << endl;
        cout << "Выберите матрицу(A), в которую будем присваивать(номер её): ";
";
        cin >> k1;
        if (k1 > matrixs.size() or k1 < 1) {
            system("cls");
            SetColor(6, 0);
            cout << "Такой матрицы нет( Выберите из предложенного." << endl;
            SetColor(15, 0);
        }
        else
            break;
    }
    while (true) {
        cout << endl << "Матриц создано: 1-" << matrixs.size() << endl;
        cout << "Выберите матрицу(B), которую будем присваивать(номер её): ";
        cin >> k2;
        if (k2 > matrixs.size() or k2 < 1) {
            system("cls");
            SetColor(6, 0);
            cout << "Такой матрицы нет( Выберите из предложенного." << endl;
            SetColor(15, 0);
        }
        else
            break;
    }
    matrixs[k1 - 1].assignment(matrixs[k2 - 1]);
    system("pause");
    system("cls");
}

void exi_t() { //Выход
    system("cls");
    SetColor(10, 0);
    cout << "Спасибо за испольтзование программы.\nХорошего дня!\n" << endl;
    SetColor(15, 0);
    system("pause");
    exit(1);
}

void nothing(int num) { //Ничего не выбрали из списка
    SetColor(6, 0);
    system("cls");
    cout << "Такого варианта нет, введите пожалуйста другой. (Предыдущий: " <<
char(num) << ")\n" << endl;
    SetColor(15, 0);
}

/*ГЛАВНАЯ ФУНКЦИЯ*/
int main()
{
    /* //Установка размера шрифта
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    CONSOLE_FONT_INFOEX fontInfo;
    fontInfo.cbSize = sizeof(fontInfo);
    GetCurrentConsoleFontEx(hConsole, TRUE, &fontInfo);
    wcsncpy(fontInfo.FaceName, L"Lucida Console");
    fontInfo.dwFontSize.Y = 15;
    SetCurrentConsoleFontEx(hConsole, TRUE, &fontInfo);
    */
}

```

```

static const TCHAR* myTitle = TEXT("Upper Triangular Matrix"); //Установка
заголовка консольного приложения
SetConsoleTitle(myTitle);

SetConsoleCP(1251);
SetConsoleOutputCP(1251);
setlocale(LC_ALL, "Russian");
int num;

while (true) { //Меню
    cout << "*****МЕНЮ*****\nВыберите функцию:" << endl;
    cout << "1. Ввод матрицы" << endl;
    if (!matrixs.empty()) {
        cout << "2. Вывод матрицы" << endl;
        cout << "3. Установить значение элемента матрицы" << endl;
        cout << "4. Получить значение элемента матрицы" << endl;
        if (matrixs.size() > 1) {
            cout << "5. Разность (вычитание) двух матриц" << endl;
            cout << "6. Проверить равенство двух матриц" << endl;
            cout << "7. Присвоение одной матрицы к другой" << endl;
        }
    }
    cout << "Для выхода нажмите - Esc." << endl;
    num = _getch();

    if (char(num) == '1') //Ввод
        z1();

    else if (char(num) == '2' and !matrixs.empty()) //Вывод
        z2();

    else if (char(num) == '3' and !matrixs.empty()) //Установка значения
элемента
        z3();

    else if (char(num) == '4' and !matrixs.empty()) //Получение значения
элемента
        z4();

    else if (char(num) == '5' and matrixs.size() > 1) //Вычитание двух
матриц (разность)
        z5();

    else if (char(num) == '6' and matrixs.size() > 1) //Проверка равенства
матриц
        z6();

    else if (char(num) == '7' and matrixs.size() > 1) //Присвоение матриц
        z7();

    else if (num == 27) //Если пользователь нажал - Esc (Выход)
        exit_t();

    else //Ничего не выбрали из списка
        nothing(num);
}
}

```