**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра АПУ**

Курсовая РАБОТА

**по дисциплине «Программирование»**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 4391 |  | Зайцева О.Ю. |
| Преподаватель |  | Писарев А.С. |

Санкт-Петербург

2024

**ЗАДАНИЕ**

**НА КУРСОВУЮ РАБОТУ**

|  |  |  |
| --- | --- | --- |
| Студентка Зайцева О.Ю. | | |
| Группа 4391 | | |
|  | | |
| Исходные данные:  Программа должна оперировать тремя квадратными матрицами (А, В и С), размерность которых задается (или определяется) на фазе работы программы.  Необходимо:  1. Реализовать динамические структуры данных и алгоритмы их обработки, позволяющие поддерживать выполнение следующих функций:  1) консольный ввод/вывод данных о матрицах А, В и С;  2) файловый ввод/вывод данных о матрицах А, В и С;  3) интерактивное редактирование элементов матриц;  2. Заменить столбцы матрицы С аналогичными столбцами либо из А, либо из В в зависимости от того, в какой матрице сумма элементов соответствующего столбца больше  Программа должна поддерживать систему меню и подменю, пункты которых соответствуют выполнению функций, предусмотренных заданием.  Содержание пояснительной записки:  «Содержание», «Введение», «Цель и задачи», «Описание кода»,  «Тестирование», «Заключение», «Список использованных  источников», «Приложение А. Полный текст программы» | | |
| Предполагаемый объем пояснительной записки:  Не менее 00 страниц. | | |
| Дата выдачи задания: 7 ноября 2024 | | |
| Дата сдачи реферата: 13 декабря 2024 | | |
| Дата защиты реферата: 14 декабря 2024 | | |
| Студентка |  | Зайцева О.Ю. |
| Преподаватель |  | Писарев А.С. |

**Аннотация**

Курсовая работа включает разработку программы на языке C++, которая демонстрирует работу с тремя квадратными матрицами (A, B и C) с динамически определяемой размерностью. В проекте реализованы функции для консольного и файлового ввода/вывода данных о матрицах, а также интерактивного редактирования их элементов. Основное внимание уделяется алгоритму, который заменяет столбцы матрицы C аналогичными столбцами из матриц A или B в зависимости от суммы элементов соответствующих столбцов. Если сумма элементов столбца в матрице A больше, чем в матрице B, соответствующий столбец из A заменяет столбец в C; в противном случае происходит замена на столбец из B. Результаты работы программы подтверждают успешное выполнение всех операций с матрицами и корректное отображение изменённой матрицы C. Программа имеет удобный пользовательский интерфейс, обеспечивающий простое взаимодействие через консоль.

**содержание**

[ВВЕДЕНИЕ 6](#_Toc185034461)

[1. ЦЕЛЬ И ЗАДАЧИ 7](#_Toc185034462)

[1.1. Цель 7](#_Toc185034463)

[1.2. Задачи 7](#_Toc185034464)

[2. ОПИСАНИЕ КОДА 7](#_Toc185034465)

[2.1. Подключение библиотек 7](#_Toc185034466)

[2.2. Использование пространства имен 8](#_Toc185034467)

[2.3. Функция для ввода элементов матрицы с клавиатуры 8](#_Toc185034468)

[2.4. Функция для вывода матрицы на экран 9](#_Toc185034469)

[2.5. Функция для редактирования элементов матрицы 9](#_Toc185034470)

[2.6. Функция для сохранения матрицы в файл 11](#_Toc185034471)

[2.7. Функция для загрузки матрицы из файла 12](#_Toc185034472)

[2.8. Функция для вычисления суммы элементов указанного столбца матрицы 13](#_Toc185034473)

[2.9. Функция для замены в матрице С на основе сумм столбцов матриц А и В 14](#_Toc185034474)

[2.10. Функция для освобождения памяти матрицы 15](#_Toc185034475)

[2.11. Основная функция 16](#_Toc185034476)

[2.12. Основное меню программы 17](#_Toc185034477)

[2.13. Освобождение выделенной памяти с помощью функции deleteMatrix 18](#_Toc185034478)

[3. ТЕСТИРОВАНИЕ 18](#_Toc185034479)

[3.1. Ввод данных и успешное тестирование 18](#_Toc185034480)

[3.2. Отображение сохранённых матриц в файле 20](#_Toc185034481)

[ЗАКЛЮЧЕНИЕ 21](#_Toc185034482)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 22](#_Toc185034483)

[ПРИЛОЖЕНИЕ А. ПОЛНЫЙ ТЕКСТ ПРОГРАММЫ 23](#_Toc185034484)

# **ВВЕДЕНИЕ**

В условиях современного мира, где обработка данных становится все более актуальной, матричные операции играют ключевую роль в различных областях науки и техники. Данная курсовая работа посвящена разработке программы на языке C++, которая осуществляет работу с тремя квадратными матрицами, обозначенными как A, B и C. Размерность этих матриц определяется на этапе выполнения программы, что обеспечивает гибкость и адаптивность в работе с различными наборами данных.

Целью проекта является создание функционального программного обеспечения, которое включает в себя консольный и файловый ввод/вывод данных о матрицах, а также интерактивное редактирование их элементов. Одной из ключевых задач является реализация алгоритма, который позволяет заменять столбцы матрицы C аналогичными столбцами из матриц A или B в зависимости от суммы элементов соответствующих столбцов. Это требует применения динамических структур данных и эффективных алгоритмов для оптимизации производительности и управления памятью.

Программа также включает систему меню и подменю, что позволяет пользователям легко выбирать необходимые операции и управлять процессом работы с матрицами. Таким образом, проект направлен на создание комплексного решения для обработки данных в виде матриц, что может быть полезно в различных вычислительных задачах и исследованиях. Код, представленный в работе, не только решает конкретную задачу, но и углубляет понимание принципов работы с динамическими структурами данных, а также теоретических и практических аспектов реализации алгоритмов на языке C++.

## **1. ЦЕЛЬ И ЗАДАЧИ**

### 1.1. Цель

Разработка программы для работы с тремя квадратными матрицами (A, B и C), которая включает в себя создание, редактирование, сохранение и загрузку данных, а также выполнение анализа на основе суммы элементов столбцов.

### 1.2. Задачи

Создать функцию для динамического выделения памяти под квадратную матрицу.

Реализовать функцию для освобождения памяти, занятой матрицей.

Разработать функции для консольного ввода и вывода матриц.

Создать функции для загрузки и сохранения матриц в файл.

Реализовать функцию для изменения значений элементов матриц пользователем.

Разработать функцию, которая заменяет столбцы матрицы C аналогичными столбцами из матриц A или B в зависимости от суммы элементов соответствующих столбцов.

Создать меню и подменю для управления программой и выполнения всех предусмотренных функций.

## **2. ОПИСАНИЕ КОДА**

### 2.1. Подключение библиотек

#include <iostream>: Подключает библиотеку для работы с вводом и выводом в стандартный поток (консоль).

#include <fstream>: Подключает библиотеку для работы с файловым вводом и выводом.

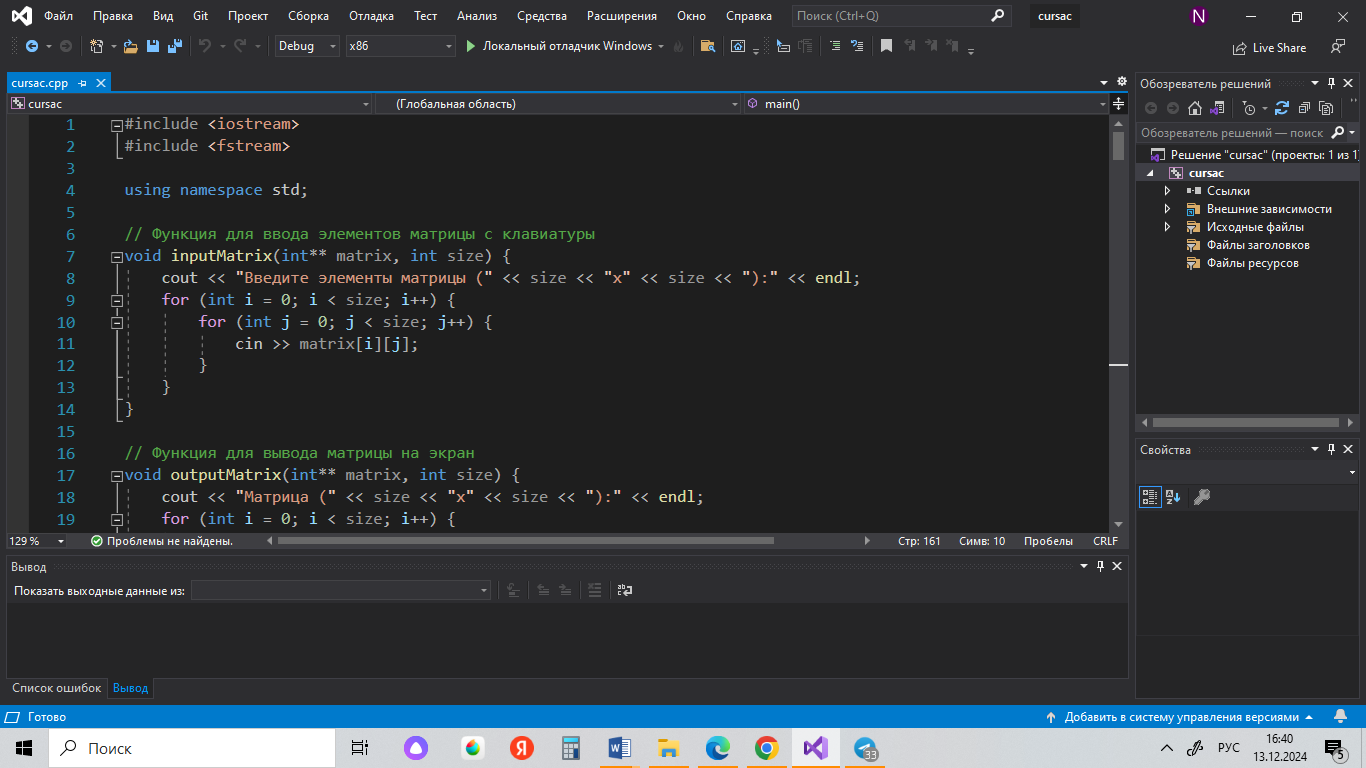


Рисунок 1 – Подключенная библиотека

### 2.2. Использование пространства имен

Инструкция using namespace std представляет собой команду для использования стандартного пространства имён std.

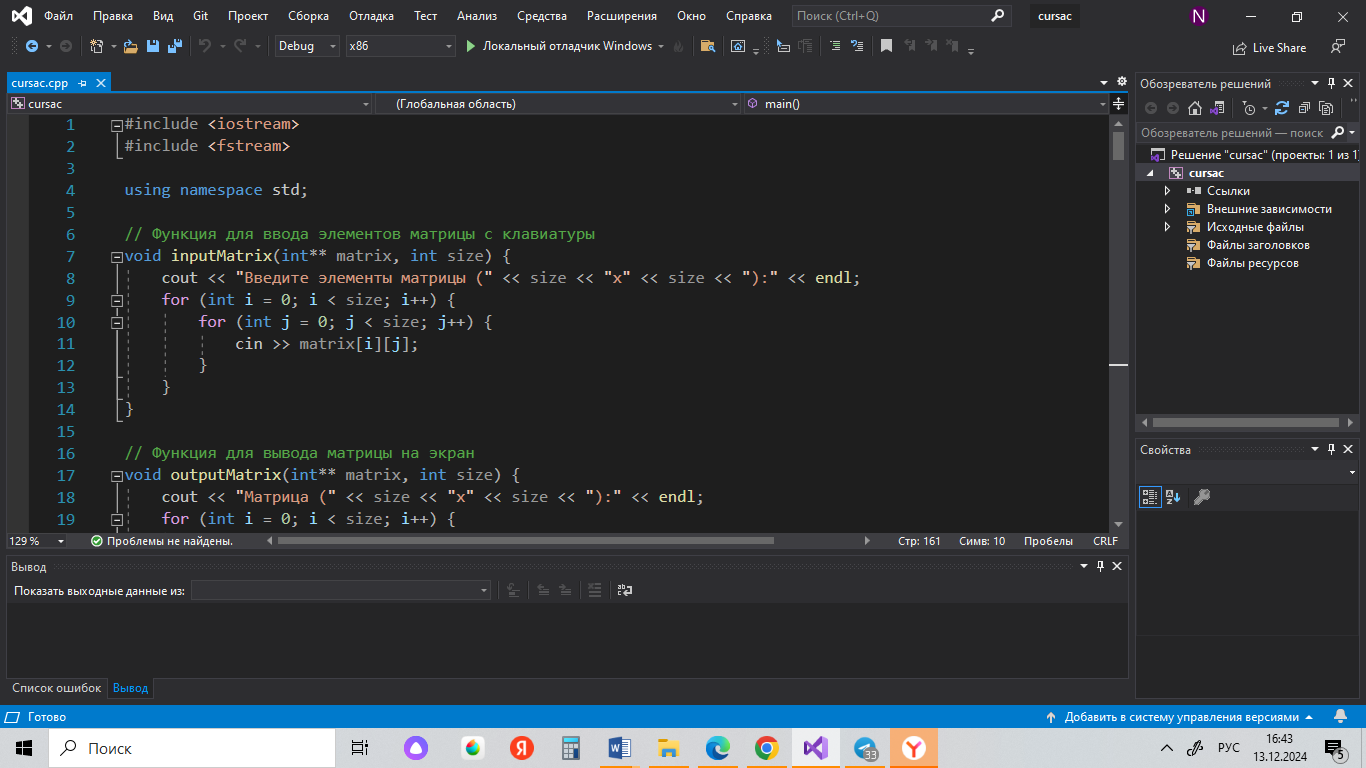


Рисунок 2 – Пространство имён

### 2.3. Функция для ввода элементов матрицы с клавиатуры

Была написана функция inputMatrix, которая предназначена для ввода элементов квадратной матрицы с клавиатуры.

Указатель на указатель (int\*\* matrix), представляет собой двумерный массив (в данном случае — матрицу). Этот параметр позволяет функции работать с динамически выделенной памятью для матрицы. Целочисленный параметр (int size), который указывает размерность матрицы (так как матрица квадратная, это число будет равно количеству строк и столбцов). Написана строка, которая выводит сообщение на экран, информируя пользователя о том, что он должен ввести элементы матрицы размером size x size. Далее был написан двойной цикл для ввода элементов. Внешний цикл (for (int i = 0; i < size; i++)) проходит по всем строкам матрицы. Переменная i представляет индекс текущей строки. Внутренний цикл (for (int j = 0; j < size; j++)) проходит по всем столбцам текущей строки. Переменная j представляет индекс текущего столбца. Далее было написано считывание значения, введенного пользователем, и сохранение его в соответствующую ячейку матрицы (в строке i и столбце j).

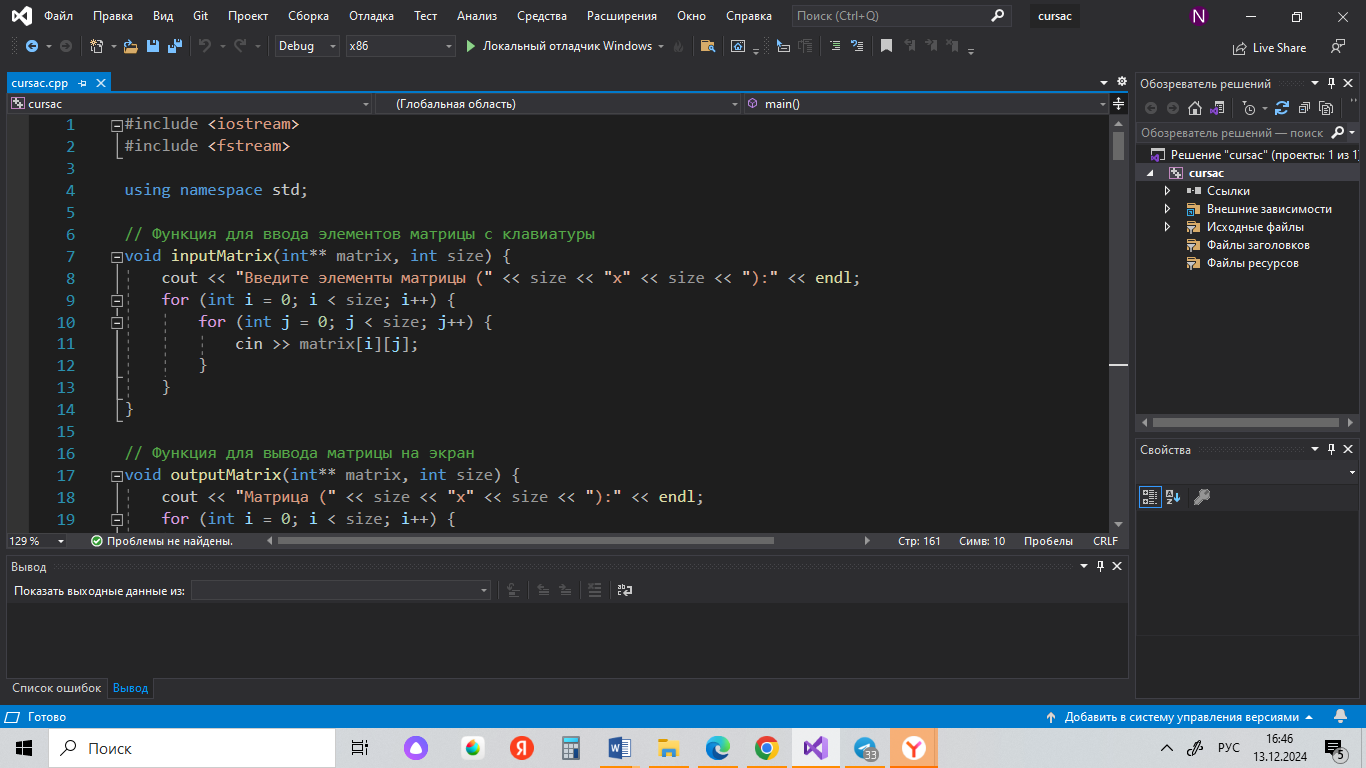


Рисунок 3 – Функция для ввода элементов матрицы с клавиатуры

### 2.4. Функция для вывода матрицы на экран

Была написана функция outputMatrix, которая предназначена для вывода элементов квадратной матрицы на экран. Написана строка, которая выводит на экран сообщение, информирующее пользователя о том, что будет показана матрица размером size x size. Используется cout для вывода текста в стандартный поток (консоль), а endl для перехода на новую строку после вывода. Создан двойной цикл для вывода элементов матрицы. Внешний цикл (for (int i = 0; i < size; i++)) проходит по всем строкам матрицы. Переменная i представляет индекс текущей строки. Внутренний цикл (for (int j = 0; j < size; j++)) проходит по всем столбцам текущей строки. Переменная j представляет индекс текущего столбца. cout << matrix[i][j] << " " считывает значение из ячейки матрицы, находящейся в строке i и столбце j, и выводит его на экран. После каждого значения добавляется пробел для удобства восприятия. cout << endl;. После завершения внутреннего цикла (т.е. после вывода всех элементов текущей строки) происходит переход на новую строку. Это позволяет выводить элементы матрицы построчно, что делает вывод более читаемым.

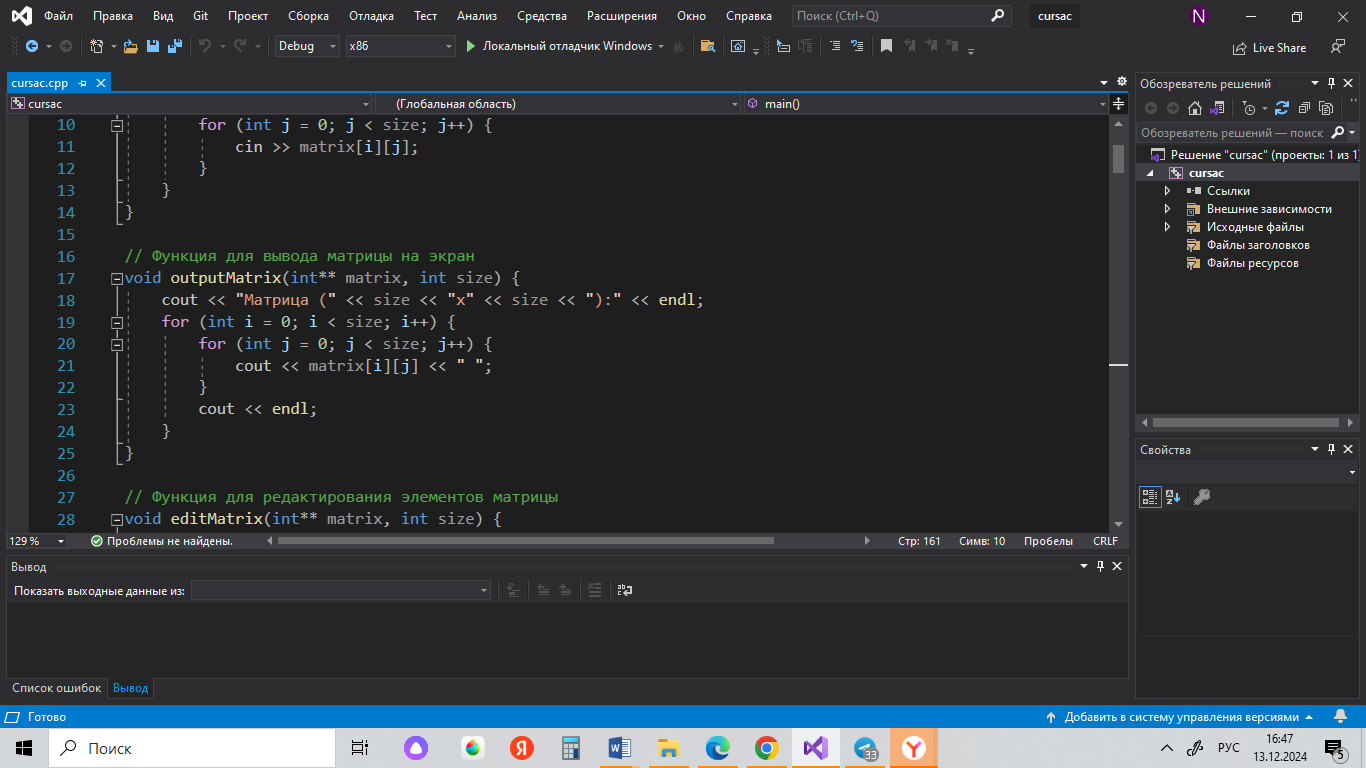


Рисунок 4 – Функция для вывода матрицы на экран

### 2.5. Функция для редактирования элементов матрицы

Написана функция editMatrix, которая предназначена для редактирования элементов квадратной матрицы. Была объявлена переменная choice, которая будет использоваться для хранения выбора пользователя (хочет ли он продолжить редактирование или выйти). Написан цикл редактирования. Начинается цикл с do, который будет выполняться как минимум один раз. Внутри него объявляются три переменные: row – для хранения индекса строки, которую пользователь хочет редактировать; col – для хранения индекса столбца, который пользователь хочет редактировать; value – для хранения нового значения, которое пользователь хочет установить в выбранной ячейке матрицы. Далее пользователю предлагается ввести индексы строки и столбца, которые он хочет редактировать. Сообщение информирует о допустимых значениях индексов (от 0 до size - 1). Дальше следует проверка, которая смотрит находятся ли введенные индексы в допустимых пределах. Если оба индекса корректны, выполняется следующий блок кода. Ввод нового значения и обновление матрицы. Пользователю предлагается ввести новое значение для указанной ячейки матрицы. После этого значение обновляется в матрице по указанным индексам. Если индексы некорректны, выводится сообщение об ошибке: "Неверный индекс!". После завершения редактирования текущего элемента матрицы программа спрашивает пользователя, хочет ли он продолжить редактирование. Вводится символ choice, который будет использоваться для определения, продолжать ли цикл. Цикл будет продолжаться, пока пользователь вводит 1 (да). Если пользователь введет любой другой символ (например, 2 для "нет"), цикл завершится, и функция editMatrix завершится.

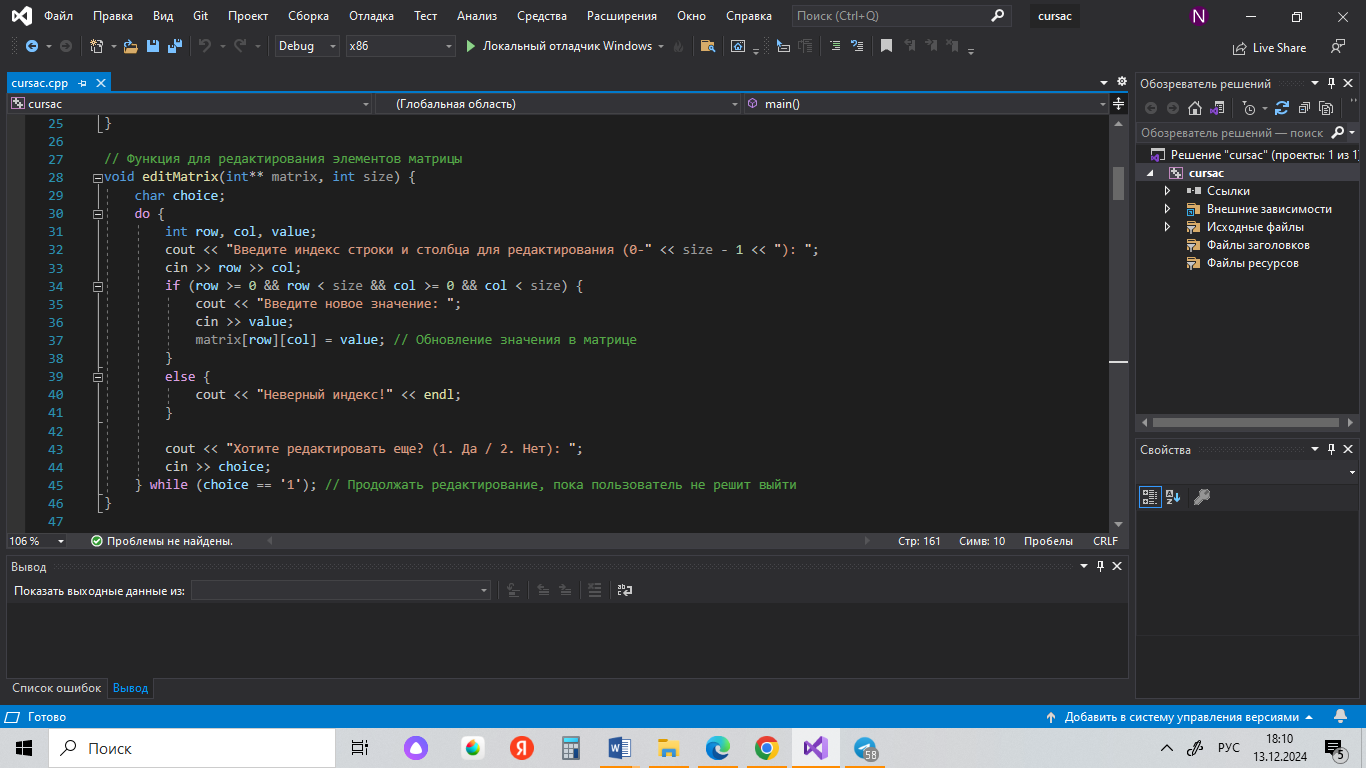


Рисунок 5 – Функция для редактирования элементов матрицы

### 2.6. Функция для сохранения матрицы в файл

Была написана функция saveMatrixToFile, которая предназначена для сохранения квадратной матрицы в файл. const string& filename – константная ссылка на строку, представляющую имя файла. Использование const предотвращает изменение строки, а & позволяет избежать копирования строки, что экономит память. Был создан объект file типа ofstream (output file stream) для записи в файл с именем, переданным в параметре filename. Это инициирует процесс открытия файла для записи. Далее проверяется, успешно ли открыт файл. Метод is\_open() возвращает true, если файл успешно открыт для записи. Если файл не удалось открыть (например, из-за отсутствия прав на запись или неправильного имени файла), программа выполнит блок else. Если файл открыт успешно, происходит запись матрицы в файл. Внешний цикл проходит по строкам (от 0 до size - 1). Внутренний цикл проходит по элементам каждой строки (от 0 до size - 1). Каждый элемент матрицы записывается в файл, за которым следует пробел (" "), чтобы отделить элементы в строке. После записи всех элементов строки добавляется перевод строки (endl), чтобы следующая строка матрицы начиналась с новой строки в файле. После завершения записи данных файл закрывается с помощью метода close(). Это освобождает ресурсы и гарантирует, что все данные были корректно записаны в файл. После успешного сохранения матрицы в файл выводится сообщение на экран, информирующее пользователя о том, что матрица была успешно сохранена в указанный файл. Если файл не был открыт (например, из-за недостатка прав доступа или других проблем), выводится сообщение об ошибке: "Не удалось открыть файл для записи." Это позволяет пользователю понять, что операция не удалась.

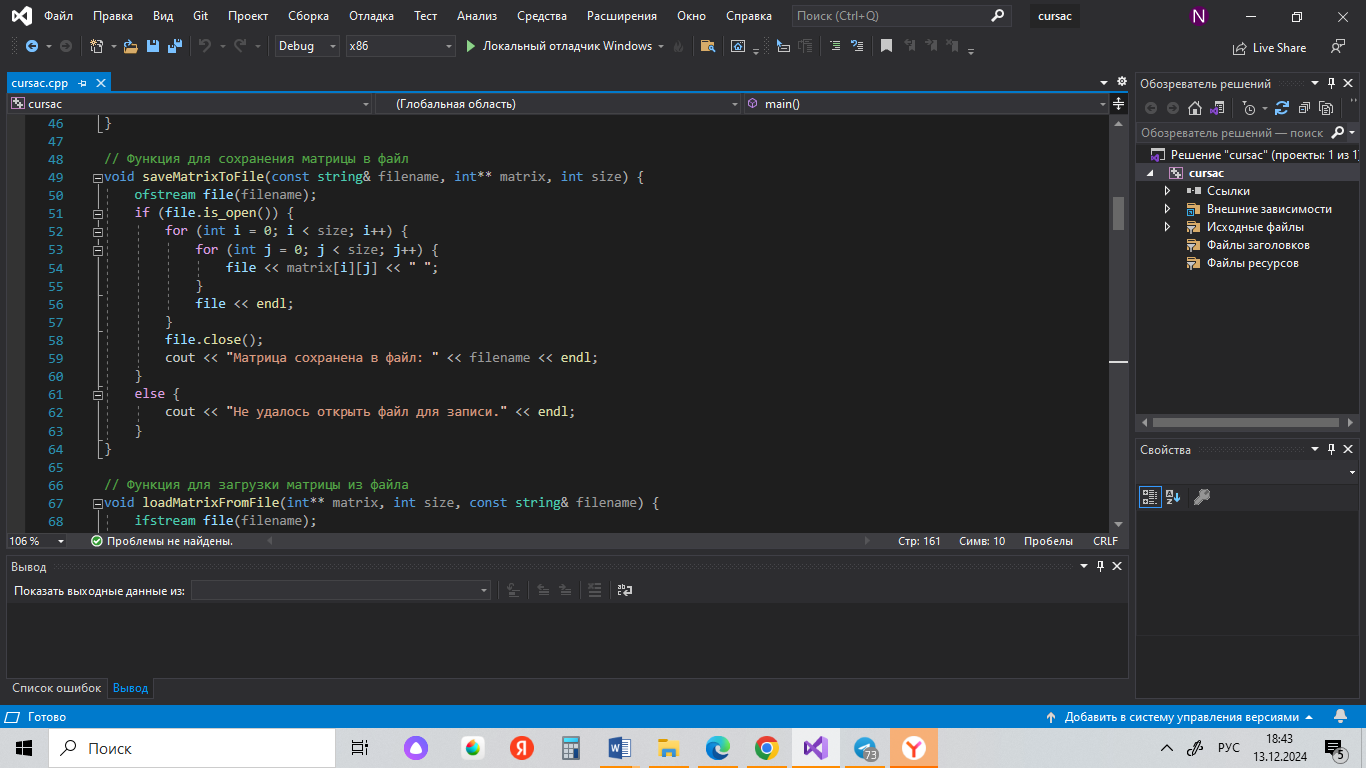


Рисунок 6 – Функция для сохранения матрицы в файл

### 2.7. Функция для загрузки матрицы из файла

Была написана функция loadMatrixFromFile, которая предназначена для загрузки квадратной матрицы из файла. Был создан объект file типа ifstream для чтения из файла с именем, переданным в параметре filename. Если файл с таким именем не существует, объект file не сможет его открыть. Далее проверяется, успешно ли открыт файл. Метод is\_open() возвращает true, если файл открыт успешно, и false, если произошла ошибка (например, если файл не существует или недоступен для чтения). Если файл открыт успешно, происходит чтение матрицы из файла. Внешний цикл проходит по строкам (от 0 до size - 1). Внутренний цикл проходит по элементам каждой строки (от 0 до size - 1). Каждый элемент матрицы считывается из файла с помощью оператора `>>` и сохраняется в соответствующей ячейке матрицы `matrix[i][j]`. Это позволяет загружать значения, которые были ранее сохранены в файл. Далее идёт закрытие файла, освобождая ресурсы. Это важно для предотвращения утечек памяти и обеспечения того, что все операции с файлом завершены. Далее выводится сообщение об успешной загрузке матрицы из файла, указывая имя файла, из которого была загружена матрица. Если файл не был открыт (например, из-за отсутствия файла или недостатка прав доступа), выводится сообщение об ошибке: "Не удалось открыть файл для чтения." Это позволяет пользователю понять, что операция не удалась.

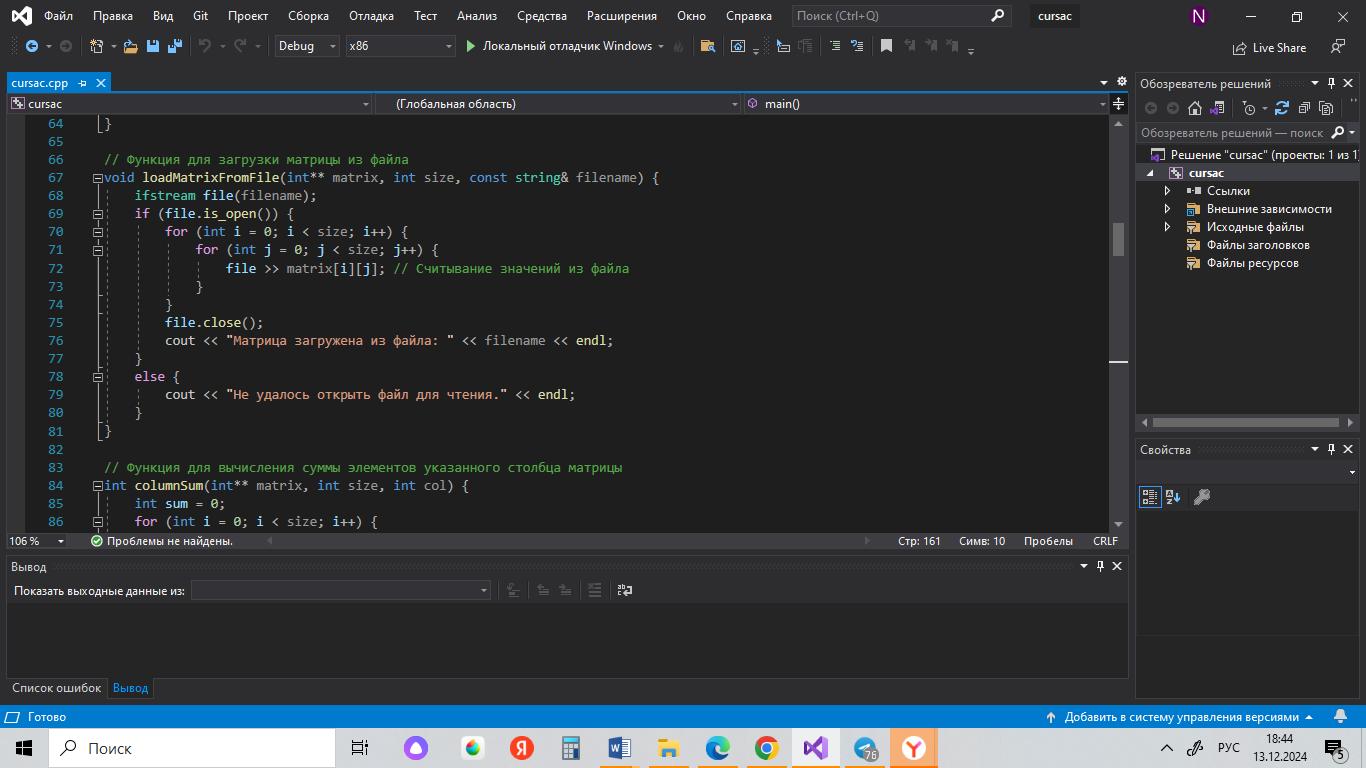


Рисунок 7 – Функция для загрузки матрицы из файла

### 2.8. Функция для вычисления суммы элементов указанного столбца матрицы

Была написана функция columnSum, которая предназначена для вычисления суммы элементов указанного столбца матрицы. Была создана переменная sum, инициализированная нулем. Эта переменная будет использоваться для накопления суммы элементов указанного столбца. Создан цикл для суммирования элементов столбца. Он проходит по всем строкам матрицы. for (int i = 0; i < size; i++) этот цикл проходит от 0 до size - 1, позволяя обработать все строки матрицы. sum += matrix[i][col];.на каждой итерации добавляется элемент matrix[i][col] к переменной sum. Это означает, что функция берет значение элемента в строке i и столбце col и добавляет его к текущему значению sum. Таким образом, за один проход по всем строкам суммируются все значения в указанном столбце. Функция (return sum) возвращает значение переменной sum, которая теперь содержит сумму всех элементов в указанном столбце. Это значение может быть использовано в других частях программы для анализа или вывода.

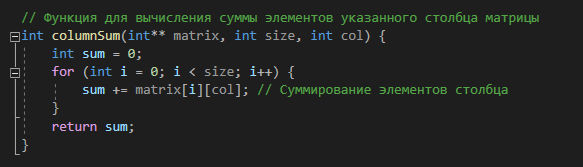


Рисунок 8 – Функция для вычисления суммы элементов указанного столбца матрицы

### 2.9. Функция для замены в матрице С на основе сумм столбцов матриц А и В

Была создана функция replaceColumns, которая предназначена для замены столбцов в матрице `C` на основе сумм соответствующих столбцов матриц A и В. int\*\* C – указатель на указатель, представляющий матрицу C, в которую будут записаны результаты. int\*\* A – указатель на указатель, представляющий матрицу A, из которой могут быть взяты значения. int\*\* B – указатель на указатель, представляющий матрицу B, из которой также могут быть взяты значения. Цикл проходит по всем столбцам матриц, где j — индекс текущего столбца. Этот цикл будет выполняться size раз, в результате чего будет обработан каждый столбец от 0 до size - 1. Далее вычисляются суммы элементов столбца j для матриц A и B с помощью функции columnSum: sumA – сумма элементов столбца j матрицы A. sumB – сумма элементов столбца j матрицы B. Далее было проведено сравнение сумм, если сумма столбца j в матрице A больше, чем в матрице B, то программа переходит к внутреннему циклу и там проходит по всем строкам i (от 0 до size - 1) и заменяет элементы столбца C[j] на соответствующие элементы столбца A[j]. Если сумма столбца j в матрице B больше, чем в матрице A, то выполняется аналогичный блок кода. Внутренний цикл заменяет элементы столбца C[j] на соответствующие элементы столбца B[j].

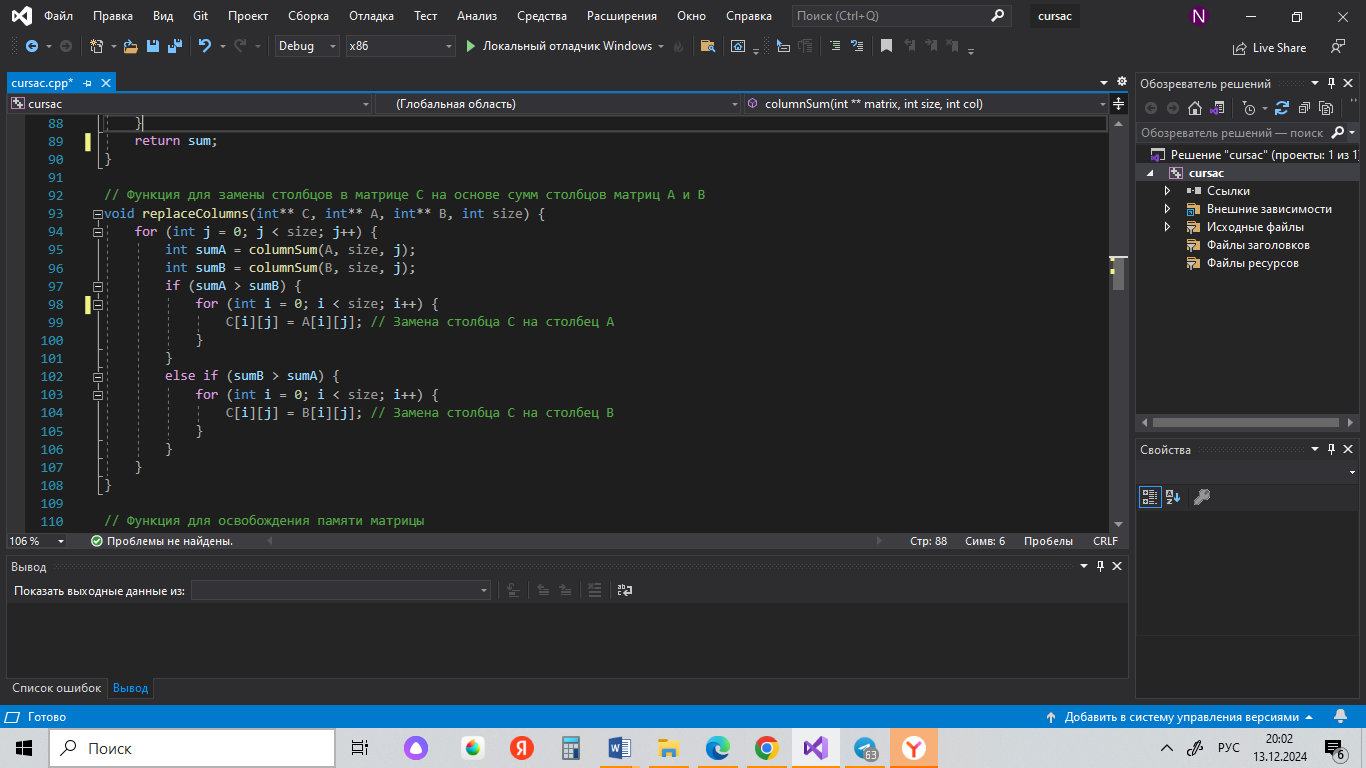


Рисунок 9 – Функция для замены в матрице С на основе сумм столбцов матриц А и В

### 2.10. Функция для освобождения памяти матрицы

Была написана функция deleteMatrix, которая предназначена для освобождения памяти, выделенной под двумерный массив (матрицу) в C++.

Данный цикл проходит по всем строкам матрицы и выполняется от 0 до size - 1, позволяя обработать все строки матрицы. На каждой итерации освобождается память, выделенная для i-й строки матрицы. Здесь используется оператор delete[], который предназначен для освобождения памяти, выделенной для массивов. Это важно делать, чтобы избежать утечек памяти. После освобождения памяти для всех строк матрицы, необходимо освободить память, выделенную для самого массива указателей, который хранит адреса этих строк. Этот оператор освобождает память, выделенную для массива указателей, который указывает на строки матрицы. Это также делается с помощью delete[], так как массив указателей также был выделен динамически.

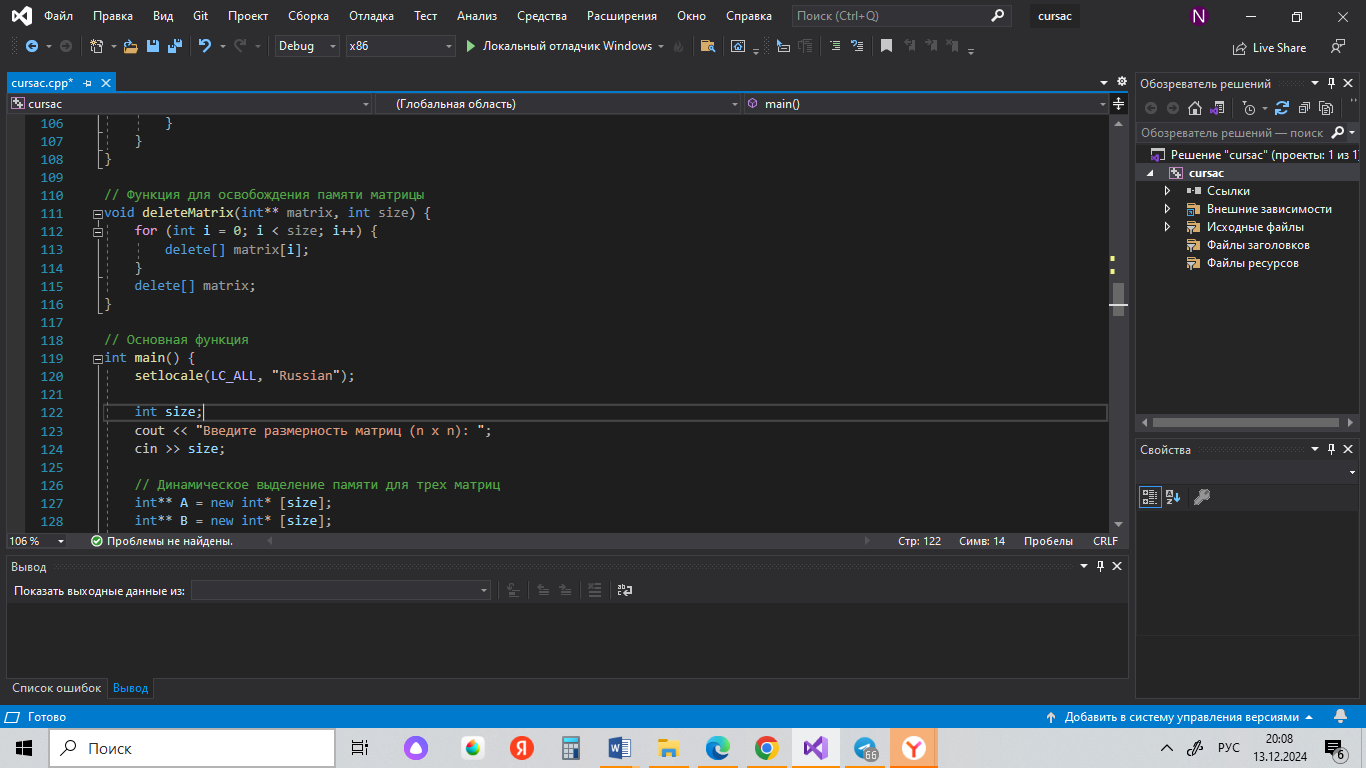


Рисунок 10 – Функция для освобождения памяти матрицы

### 2.11. Основная функция

Создано тело функции. Добавлена функция setlocale. Она позволяет устанавливать различные параметры в зависимости от географического положения пользователя, например, позволяет пользоваться тем алфавитом и временем, которые есть в этой стране. Для моей программы было указано “RU”. Было выполнено объявление переменной size, которая будет использоваться для хранения размерности квадратных матриц (количество строк и столбцов). Далее был написан вывод сообщения на экран, предлагая пользователю ввести размерность матриц. Далее идёт считывание введенного пользователем значения и сохранение его в переменной size. Было написано динамическое выделение памяти для трех матриц: A, B и C. После этого идёт выделение памяти для строк матриц.

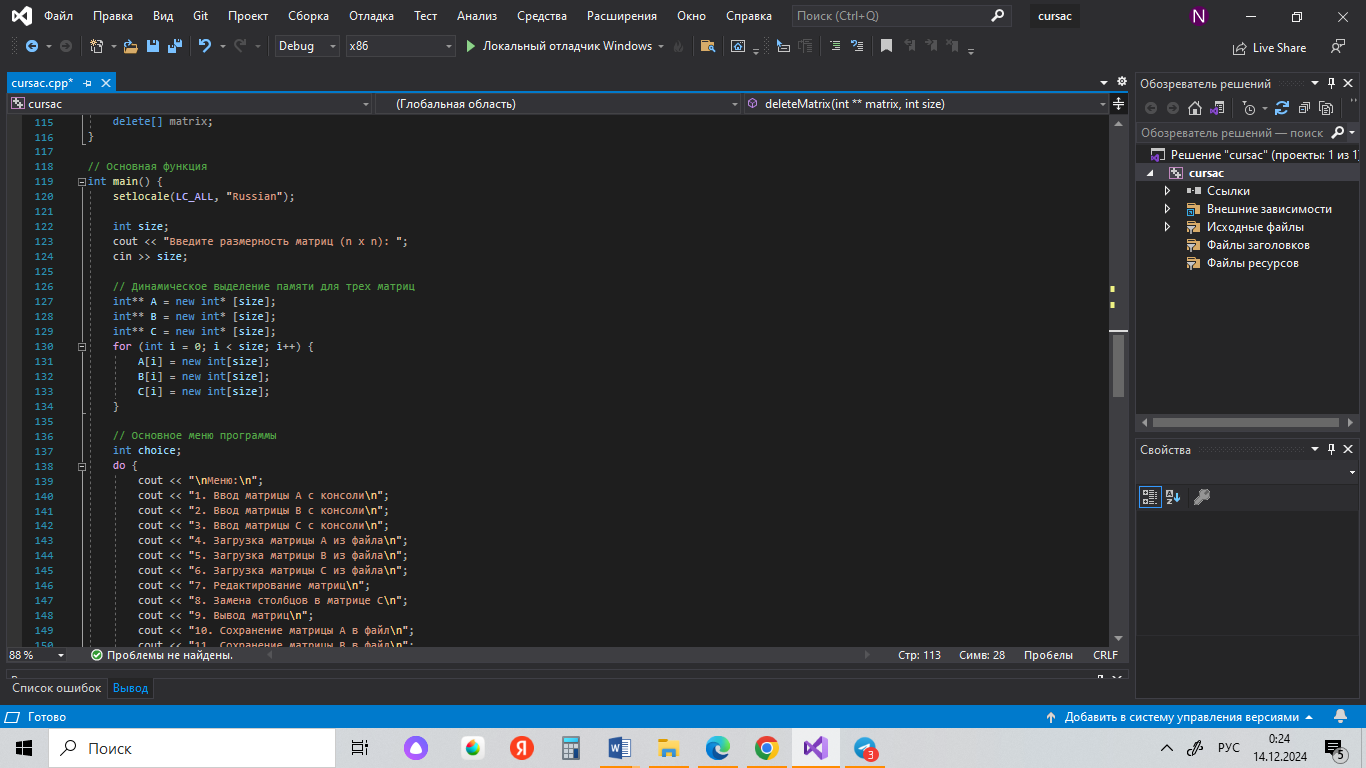


Рисунок 11 – Основная функция

### 2.12. Основное меню программы

Было написано меню программы, позволяющего пользователю выбирать различные действия, связанные с матрицами. Была объявлена переменная choice, которая будет использоваться для хранения выбора пользователя из меню. Цикл do while гарантирует, что содержимое блока будет выполнено хотя бы один раз. Цикл будет продолжаться до тех пор, пока значение choice не станет равным 13, что соответствует пункту "Выход". Это позволяет пользователю многократно выбирать действия из меню, пока он не решит завершить программу.

Далее был написан вывод меню. Каждая строка представляет собой отдельный пункт меню, который описывает доступные действия, которые пользователь может выбрать. Далее выводится сообщение, предлагающее пользователю ввести свой выбор. Затем идёт считывание введенного пользователем значения и сохранение его в переменной choice. Это значение будет использоваться для определения того, какое действие выполнить в соответствии с пунктом меню.

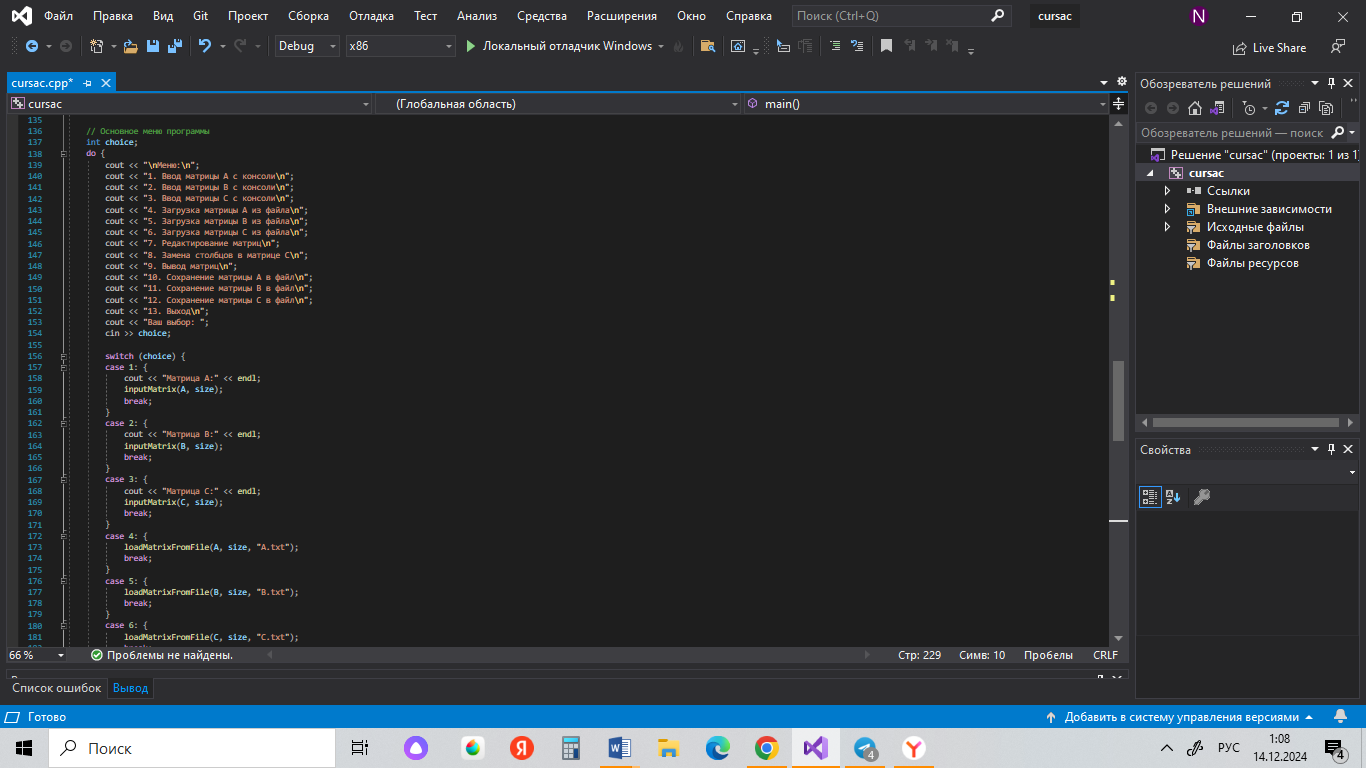


Рисунок 12 – Основное меню программы

### 2.13. Освобождение выделенной памяти с помощью функции deleteMatrix

Было написано освобождение выделенной памяти перед завершением программы, вызывая функцию deleteMatrix для каждой матрицы.

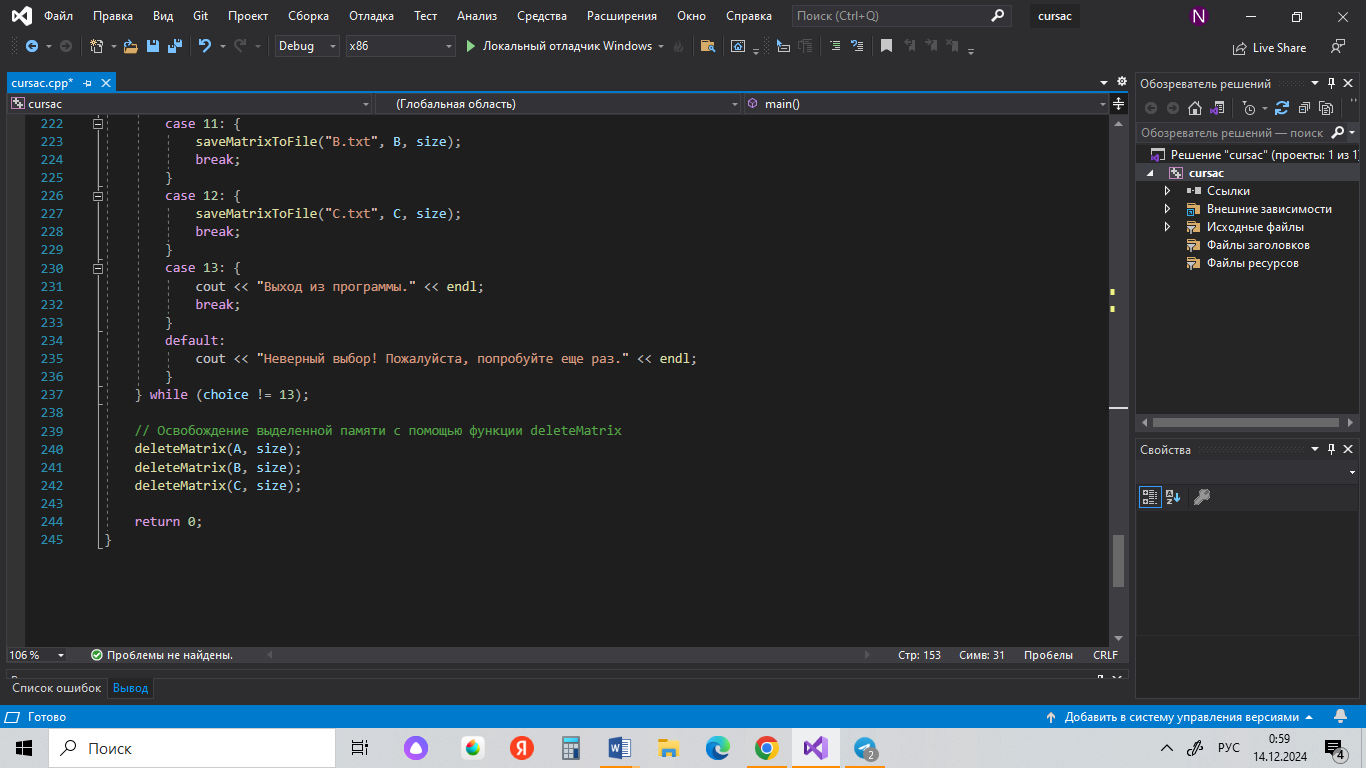


Рисунок 13 – Освобождение выделенной памяти с помощью функции deleteMatrix

## **3. ТЕСТИРОВАНИЕ**

### 3.1. Ввод данных и успешное тестирование

Была введена размерность квадратной матрицы, программа вывела меню и после было произведено заполнение числами матрицы А.

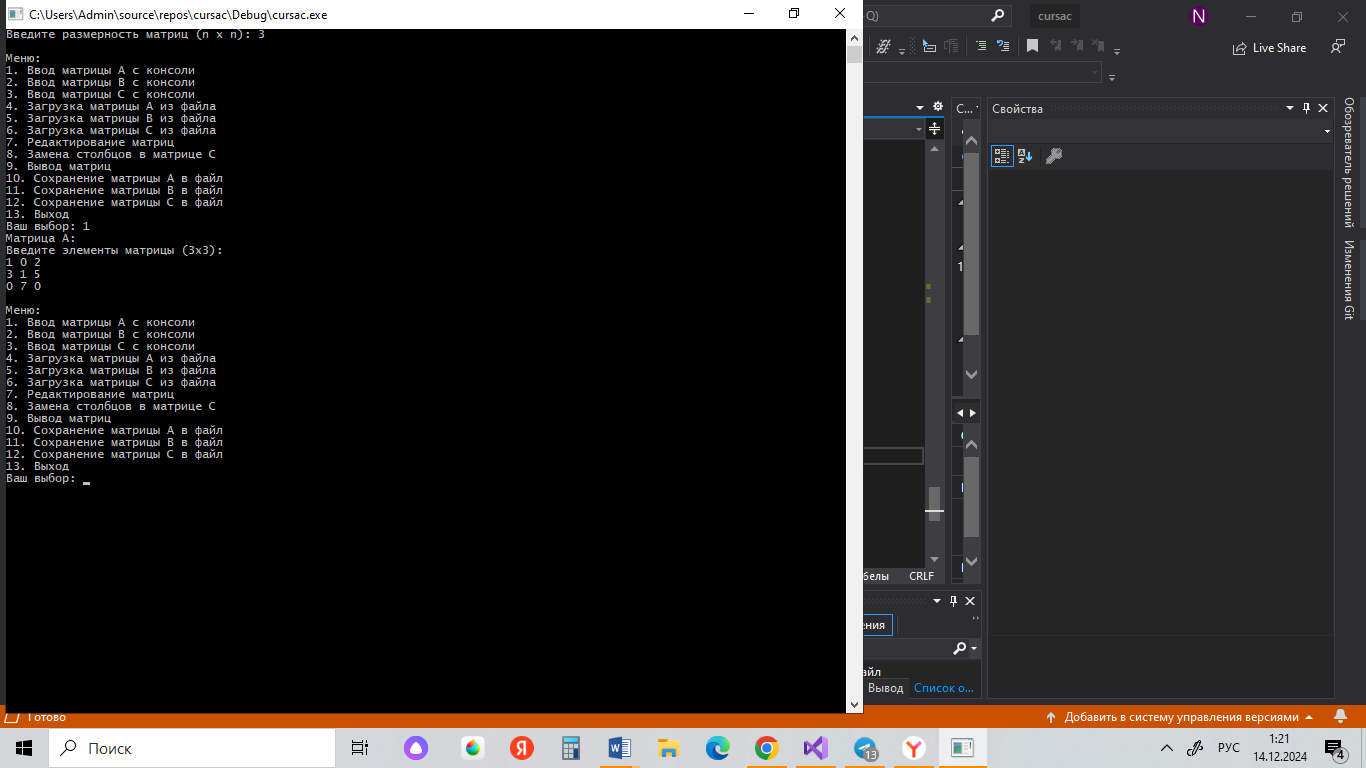


Рисунок 14 – Ввод данных

Было произведено редактирование элемента матрицы А.

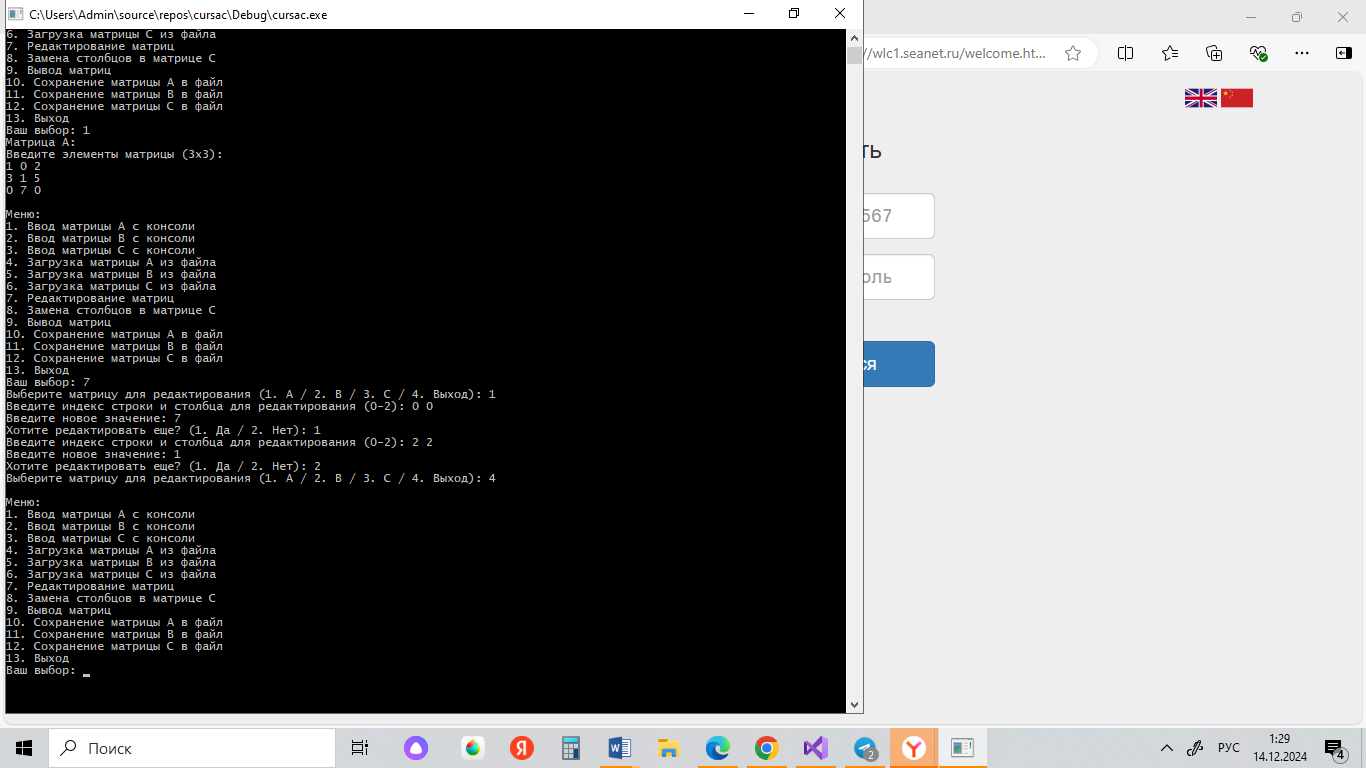


Рисунок 15 – Изменение элемента в матрице

Были заполнены матрица В и матрица С числами. После была выполнена замена столбцов матрицы C аналогичными столбцами из матриц A или B в зависимости от суммы элементов соответствующих столбцов. Программа выдала правильный результат.

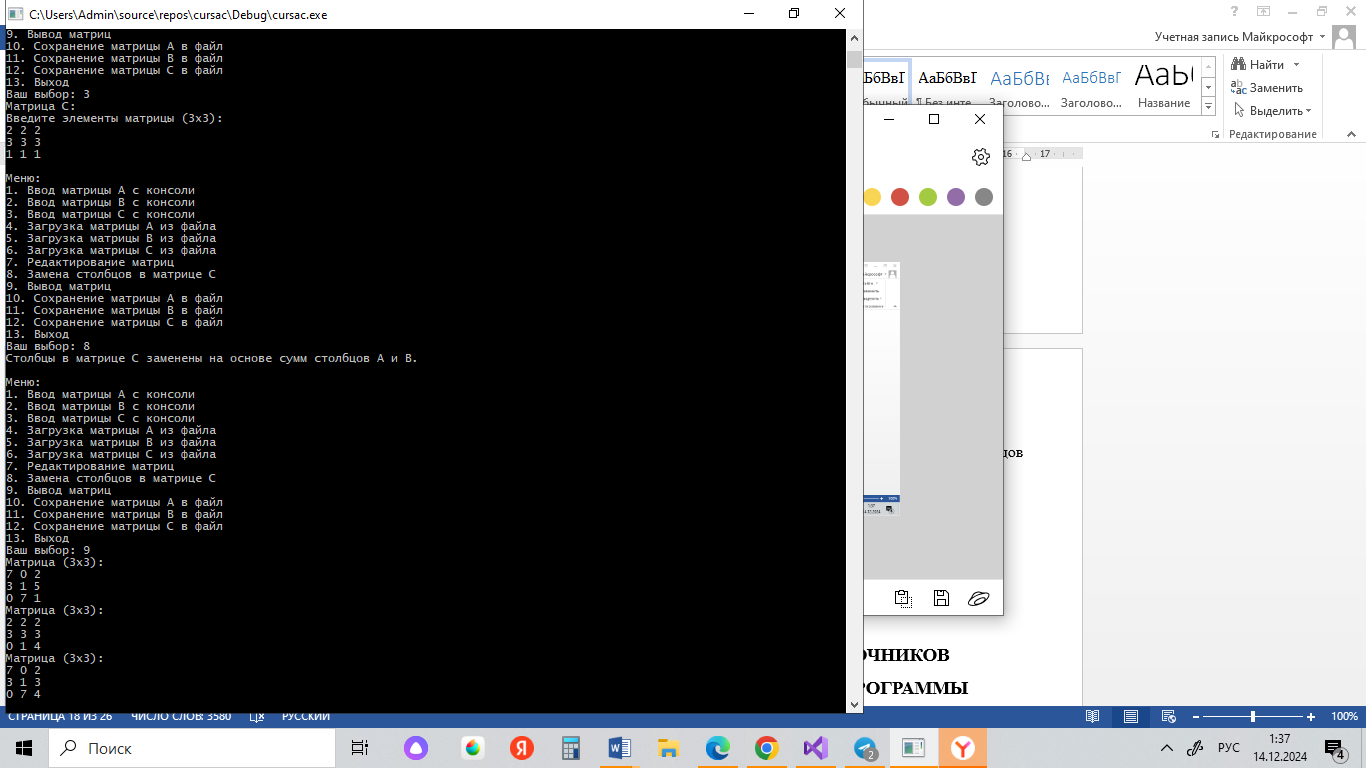


Рисунок 16 – Заполнение матриц и замена столбцов в матрице С

### 3.2. Отображение сохранённых матриц в файле

Матрица С была сохранена в файл. Программа успешно сохранила матрицу.

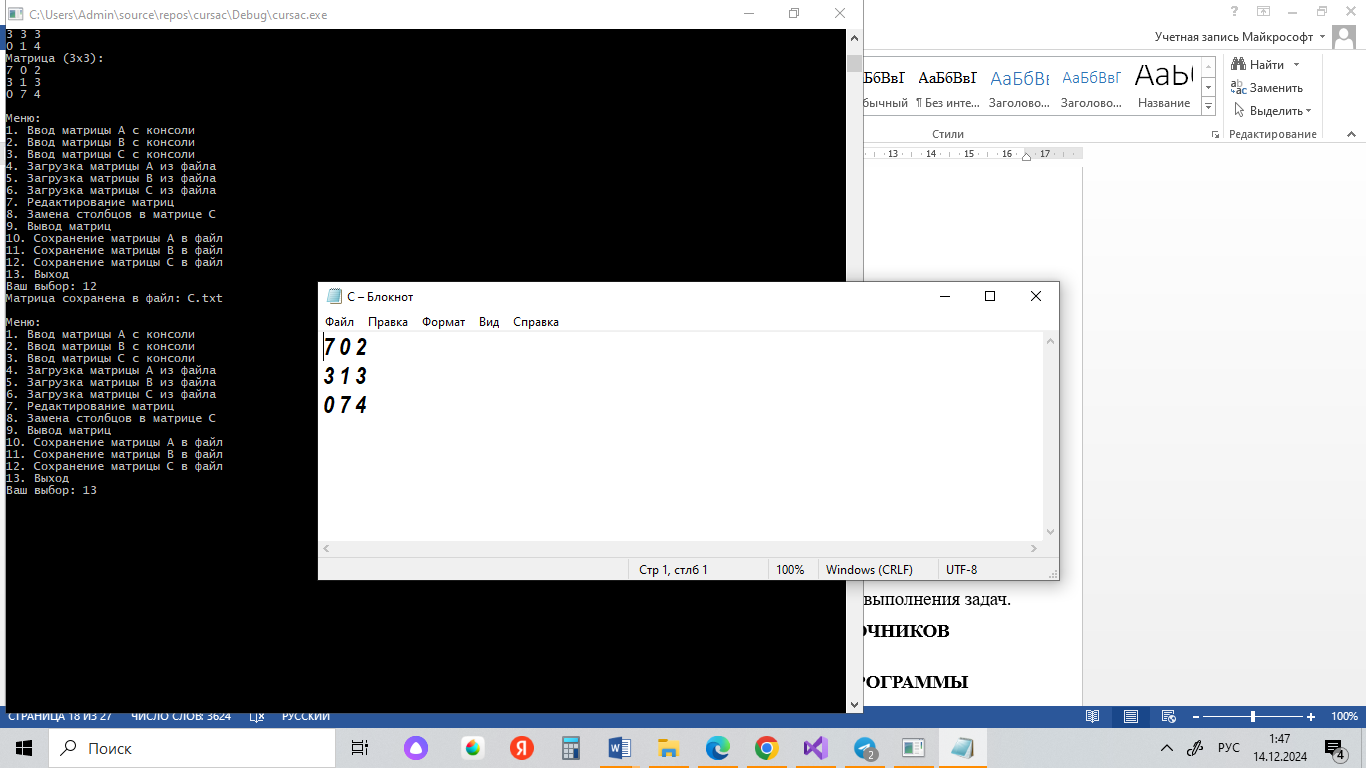


Рисунок 17 – Матрица С сохранена в файл

# **ЗАКЛЮЧЕНИЕ**

При выполнении курсовой работы была разработана программа для работы с квадратными матрицами, включающей создание, редактирование, сохранение и загрузку данных, а также анализ частоты элементов. Цель была достигнута, то есть разработана программа, посредством выполнения задач.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. https://metanit.com/cpp/tutorial/

2. http://cppstudio.com/

3. Бьярне Страуструп. Язык программирования С++. Краткий курс, второе издание.

4. Брайан Керниган. Деннис Ритчи. Язык программирования. Второе издание.

5.Васильев А.Н. Программирование на С++ в примерах и задачах.

# **ПРИЛОЖЕНИЕ А. ПОЛНЫЙ ТЕКСТ ПРОГРАММЫ**

#include <iostream>

#include <fstream>

using namespace std;

// Функция для ввода элементов матрицы с клавиатуры

void inputMatrix(int\*\* matrix, int size) {

cout << "Введите элементы матрицы (" << size << "x" << size << "):" << endl;

for (int i = 0; i < size; i++) {

for (int j = 0; j < size; j++) {

cin >> matrix[i][j];

}

}

}

// Функция для вывода матрицы на экран

void outputMatrix(int\*\* matrix, int size) {

cout << "Матрица (" << size << "x" << size << "):" << endl;

for (int i = 0; i < size; i++) {

for (int j = 0; j < size; j++) {

cout << matrix[i][j] << " ";

}

cout << endl;

}

}

// Функция для редактирования элементов матрицы

void editMatrix(int\*\* matrix, int size) {

char choice;

do {

int row, col, value;

cout << "Введите индекс строки и столбца для редактирования (0-" << size - 1 << "): ";

cin >> row >> col;

if (row >= 0 && row < size && col >= 0 && col < size) {

cout << "Введите новое значение: ";

cin >> value;

matrix[row][col] = value; // Обновление значения в матрице

}

else {

cout << "Неверный индекс!" << endl;

}

cout << "Хотите редактировать еще? (1. Да / 2. Нет): ";

cin >> choice;

} while (choice == '1'); // Продолжать редактирование, пока пользователь не решит выйти

}

// Функция для сохранения матрицы в файл

void saveMatrixToFile(const string& filename, int\*\* matrix, int size) {

ofstream file(filename);

if (file.is\_open()) {

for (int i = 0; i < size; i++) {

for (int j = 0; j < size; j++) {

file << matrix[i][j] << " ";

}

file << endl;

}

file.close();

cout << "Матрица сохранена в файл: " << filename << endl;

}

else {

cout << "Не удалось открыть файл для записи." << endl;

}

}

// Функция для загрузки матрицы из файла

void loadMatrixFromFile(int\*\* matrix, int size, const string& filename) {

ifstream file(filename);

if (file.is\_open()) {

for (int i = 0; i < size; i++) {

for (int j = 0; j < size; j++) {

file >> matrix[i][j]; // Считывание значений из файла

}

}

file.close();

cout << "Матрица загружена из файла: " << filename << endl;

}

else {

cout << "Не удалось открыть файл для чтения." << endl;

}

}

// Функция для вычисления суммы элементов указанного столбца матрицы

int columnSum(int\*\* matrix, int size, int col) {

int sum = 0;

for (int i = 0; i < size; i++) {

sum += matrix[i][col]; // Суммирование элементов столбца

}

return sum;

}

// Функция для замены столбцов в матрице C на основе сумм столбцов матриц A и B

void replaceColumns(int\*\* C, int\*\* A, int\*\* B, int size) {

for (int j = 0; j < size; j++) {

int sumA = columnSum(A, size, j);

int sumB = columnSum(B, size, j);

if (sumA > sumB) {

for (int i = 0; i < size; i++) {

C[i][j] = A[i][j]; // Замена столбца C на столбец A

}

}

else if (sumB > sumA) {

for (int i = 0; i < size; i++) {

C[i][j] = B[i][j]; // Замена столбца C на столбец B

}

}

}

}

// Функция для освобождения памяти матрицы

void deleteMatrix(int\*\* matrix, int size) {

for (int i = 0; i < size; i++) {

delete[] matrix[i];

}

delete[] matrix;

}

// Основная функция

int main() {

setlocale(LC\_ALL, "Russian");

int size;

cout << "Введите размерность матриц (n x n): ";

cin >> size;

// Динамическое выделение памяти для трех матриц

int\*\* A = new int\* [size];

int\*\* B = new int\* [size];

int\*\* C = new int\* [size];

for (int i = 0; i < size; i++) {

A[i] = new int[size];

B[i] = new int[size];

C[i] = new int[size];

}

// Основное меню программы

int choice;

do {

cout << "\nМеню:\n";

cout << "1. Ввод матрицы A с консоли\n";

cout << "2. Ввод матрицы B с консоли\n";

cout << "3. Ввод матрицы C с консоли\n";

cout << "4. Загрузка матрицы A из файла\n";

cout << "5. Загрузка матрицы B из файла\n";

cout << "6. Загрузка матрицы C из файла\n";

cout << "7. Редактирование матриц\n";

cout << "8. Замена столбцов в матрице C\n";

cout << "9. Вывод матриц\n";

cout << "10. Сохранение матрицы A в файл\n";

cout << "11. Сохранение матрицы B в файл\n";

cout << "12. Сохранение матрицы C в файл\n";

cout << "13. Выход\n";

cout << "Ваш выбор: ";

cin >> choice;

switch (choice) {

case 1: {

cout << "Матрица A:" << endl;

inputMatrix(A, size);

break;

}

case 2: {

cout << "Матрица B:" << endl;

inputMatrix(B, size);

break;

}

case 3: {

cout << "Матрица C:" << endl;

inputMatrix(C, size);

break;

}

case 4: {

loadMatrixFromFile(A, size, "A.txt");

break;

}

case 5: {

loadMatrixFromFile(B, size, "B.txt");

break;

}

case 6: {

loadMatrixFromFile(C, size, "C.txt");

break;

}

case 7: {

char matrixChoice;

do {

cout << "Выберите матрицу для редактирования (1. A / 2. B / 3. C / 4. Выход): ";

cin >> matrixChoice;

if (matrixChoice == '1') {

editMatrix(A, size);

}

else if (matrixChoice == '2') {

editMatrix(B, size);

}

else if (matrixChoice == '3') {

editMatrix(C, size);

}

else if (matrixChoice == '4') {

break;

}

else {

cout << "Неверный выбор матрицы!" << endl;

}

} while (true);

break;

}

case 8: {

replaceColumns(C, A, B, size);

cout << "Столбцы в матрице C заменены на основе сумм столбцов A и B." << endl;

break;

}

case 9: {

outputMatrix(A, size);

outputMatrix(B, size);

outputMatrix(C, size);

break;

}

case 10: {

saveMatrixToFile("A.txt", A, size);

break;

}

case 11: {

saveMatrixToFile("B.txt", B, size);

break;

}

case 12: {

saveMatrixToFile("C.txt", C, size);

break;

}

case 13: {

cout << "Выход из программы." << endl;

break;

}

default:

cout << "Неверный выбор! Пожалуйста, попробуйте еще раз." << endl;

}

} while (choice != 13);

// Освобождение выделенной памяти с помощью функции deleteMatrix

deleteMatrix(A, size);

deleteMatrix(B, size);

deleteMatrix(C, size);

return 0;

}