МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Нижегородский государственный университет им. Н.И. Лобачевского»**

**Национальный исследовательский университет**

**Институт информационных технологий, математики и механики**

**Кафедра математического обеспечения и суперкомпьютерных технологий**

**Отчет по учебной практике**

**«Структура хранения матриц специального вида»**

**Выполнил:** студент группы 381706-1

Силенко Дмитрий Игоревич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

ассистент каф. МОСТ ИИТММ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лебедев И.Г

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc532981957)

[2. Постановка задачи 4](#_Toc532981958)

[3. Руководство пользователя 5](#_Toc532981959)

[4. Руководство программиста 7](#_Toc532981960)

[4.1. Описание структуры программы 7](#_Toc532981961)

[4.2. Описание структур данных 7](#_Toc532981962)

[4.3. Описание алгоритмов 9](#_Toc532981963)

[5. Эксперименты 11](#_Toc532981964)

[6. Заключение 12](#_Toc532981965)

# Введение

В рамках лабораторной работы ставится задача создания программных средств, поддерживающих эффективное хранение матриц специального вида (верхнетреугольных) и выполнение основных операций над ними:

* сложение/вычитание;
* копирование;
* сравнение.

Но для начала разберемся, а что же вообще такое матрица.

Матрица - математический объект, записываемый в виде прямоугольной таблицы элементов кольца или поля (например, целых, действительных или комплексных чисел), которая представляет собой совокупность строк и столбцов, на пересечении которых находятся её элементы. Количество строк и столбцов задает размер матрицы. Они применяются. К примеру, для компактной записи систем неравенств или уравнений и упрощения дальнейшей работы с ними.

Матрицы бывают разных видов. Чаще используются прямоугольные матрицы (представляющие из себя двумерный массив значений), но кроме них есть также и специальные виды матриц. К примеру, диагональные или треугольные. Они имеют более специфичную структуру и, соответственно, для них необходим специальный способ хранения, дабы избежать хранения в памяти не использующихся ячеек (к примеру, если диагональную матрицу хранить как прямоугольную). Здесь мы конкретнее остановимся на подвиде треугольных матриц, а именно, верхнетреугольных матрицах.

Верхнетреугольная матрица – квадратная матрица А, у которой все элементы ниже главной диагонали равны 0:

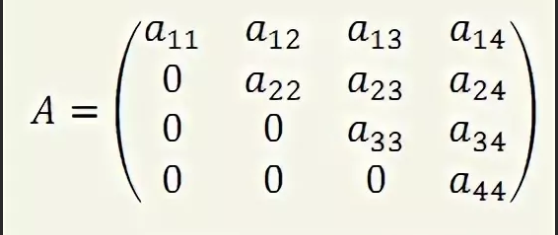


Рис.1 (Верхнетреугольная матрица)

# Постановка задачи

Выполнение работы предполагает решение следующих задач:

1. Разработка и реализация вспомогательного шаблонного класса для создания верхнетреугольных матриц TVector.
2. Разработка и реализация шаблонного класса матриц TMatrix
3. Пример программы, демонстрирующая работу классов TMatrix и TVector.
4. Написание набора автоматических тестов с использованием Google C++ Testing Framework и проверка работоспособности методов классов.

# Руководство пользователя

Поскольку работа подразделяется на написание вспомогательного, но все же самостоятельного, класса TVector и расширяющий его класс TMatrix, то примеры программ у них различны.

Вектор:

При запуске программы на экран выводятся два случайно генерируемых вектора. А так же результаты арифметических операций над ними. После этого, пользователю предлагается ввести собственный целочисленный вектор размерности 5.

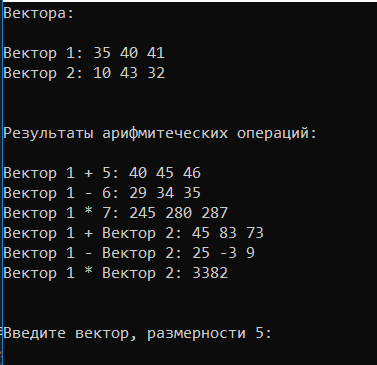


Рис.2 (Вывод случайно заданных векторов и результатов операций над ними)

Когда пользователь введет свой вектор. Программа запишет эти значения в новый вектор и выведет его на экран.

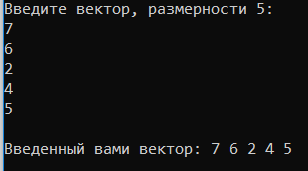


Рис.3 (Ввод вектора пользователя и вывод его же на экран)

Матрица:

При запуске. Программа выводит на экран две случайно генерируемых верхнетреугольных матрицы.

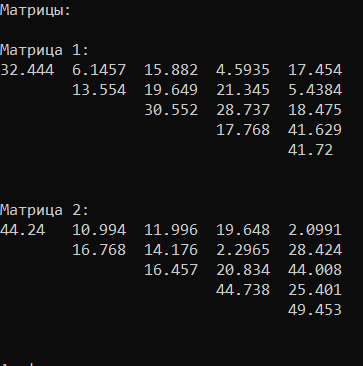


Рис.4 (Случайно заданные матрицы)

А так же результаты арифметических операций над ними

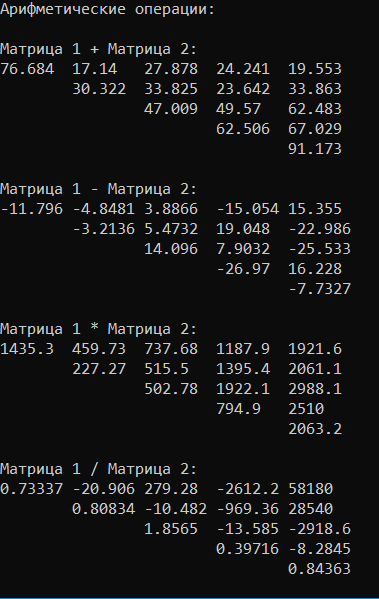


Рис.5 (Результаты арифметических операций)

После чего пользователю предлагается самому ввести верхнетреугольную матрицу размерностью 5, сразу после чего введенная им матрица будет выведена на экран

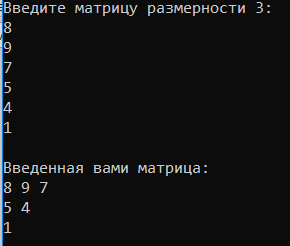


Рис.6 (Ввод матрицы пользователем и вывод ее же на экран)

# Руководство программиста

## Описание структуры программы

Программа состоит из следующих модулей:

* Модуль *VectorLib*. Статическая библиотека. Включает в себя заголовочный файл *Vector.h*, в котором описаны методы с реализацией шаблонного класса вектор *TVector*.
* Модуль *MatrixLib*. Статическая библиотека. Включает в себя заголовочный файл *Matrix.h*, в котором описаны методы с реализацией шаблонного класса матрица *TMatrix*.
* Модули *VectorTest* и *MatrixTest*. Набор тестов для обоих классов. Включает в себя файлы *VectorTest.cpp* и *MatrixTest.cpp.* Реализованы они с помощью использования фреймворка Google Test.
* Модули *Vector* и *Matrix*. Пример использования классов. Включают в себя файлы с реализацией *Main.cpp* и *Matrix­\_ main.cpp.*

## Описание структур данных

*Класс TVector:*

Поля:

\*vector – массив элементов вектора

length – длина вектора

Конструкторы и деструктор:

TVector(int len = 0);

TVector(const TVector<T> &V);

~TVector();

Методы:

int GetLength() const; - размер вектора

virtual T& operator[](int pos); - доступ по индексу

Перегрузки:

TVector<T> operator +(const TVector<T> &vec); - сложение векторов

TVector<T> operator -(const TVector<T> &vec); - вычитание векторов

T operator \*(TVector<T> &vec); - умножение векторов

TVector<T> operator+(const T &vol); - сложение с числом

TVector<T> operator-(const T &vol); - вычитание числа

TVector<T> operator\*(const T &vol); - умножение на число

bool operator==(const TVector &vec) const; - сравнение на равенство

bool operator!=(const TVector &vec) const; - сравнение на неравенство

virtual TVector& operator=(const TVector &vec); - присваивание

friend istream& operator >> (istream &in, TVector<Type1> &vec); - ввод вектора с клавиатуры

friend ostream& operator << (ostream &out, const TVector<Type1> &vec); - вывод вектора на экран

*Класс TMatrix:*

Конструкторы и деструктор:

TMatrix(int len = 10);

TMatrix(const TMatrix &matr);

TMatrix(const TVector<TVector<T> > &matr);

virtual ~TMatrix();

Перегрузки:

TMatrix operator+(const TMatrix &matr); - сложение

TMatrix operator-(const TMatrix &matr); - вычитание

TMatrix<T> operator\*(const TMatrix<T> &Matr); - умножение

TMatrix<T> operator/(const TMatrix<T> &MT); - деление

bool operator==(const TMatrix &matr) const; - сравнение на равенство

bool operator!=(const TMatrix &matr) const; - сравнение на неравенство

TMatrix& operator=(const TMatrix &matr); - присваивание

friend istream& operator>>(istream &in, TMatrix<Type2> &matr); - ввод матрицы с клавиатуры

friend ostream & operator<<(ostream &out, const TMatrix<Type2> &matr); - вывод матрицы на экран

## Описание алгоритмов

Умножение верхнетреугольных матриц:

Произведением матриц А и B является такая матрица C = AB, у которой элемент , стоящий в *i*-ой строке и *j*-ом столбце, равен сумме произведений элементов *i*-ой строки матрицы А на соответствующие элементы *j*-го столбца матрицы B.

Реализация перегрузки оператора умножения представлена, с помощью трех вложенных циклов: по строкам первой матрицы, по столбцам второй матрицы и по элементам текущего столбца второй матрицы.

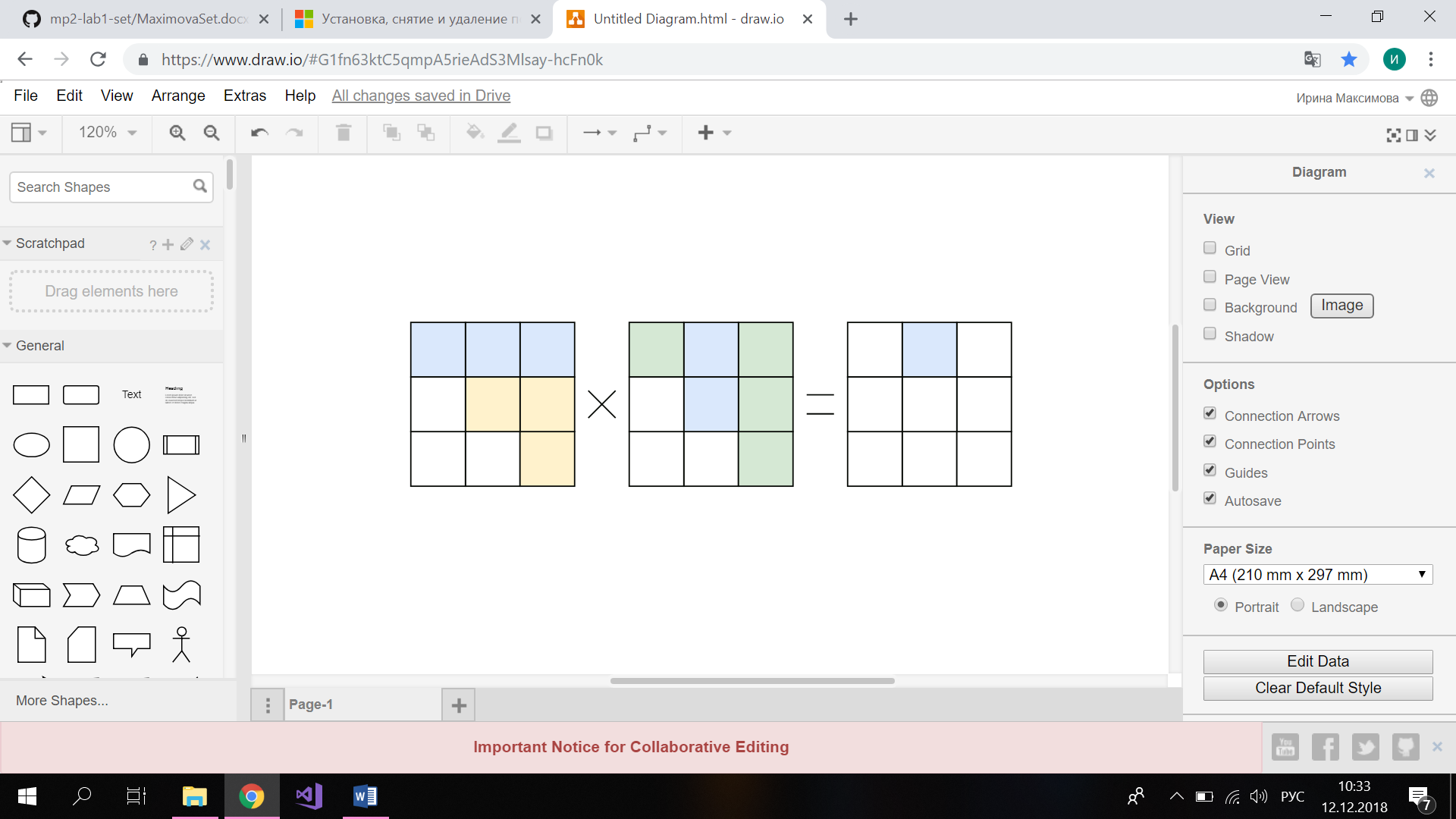


Рис.7 (Умножение верхнетреугольных матриц)

Деление верхнетреугольных матриц:

Делением матриц D и A является такая матрица C = D\*, где - обратная матрица к матрице A. Обратная матрица может быть найдена методом Гаусса: Записываем расширенную матрицу :

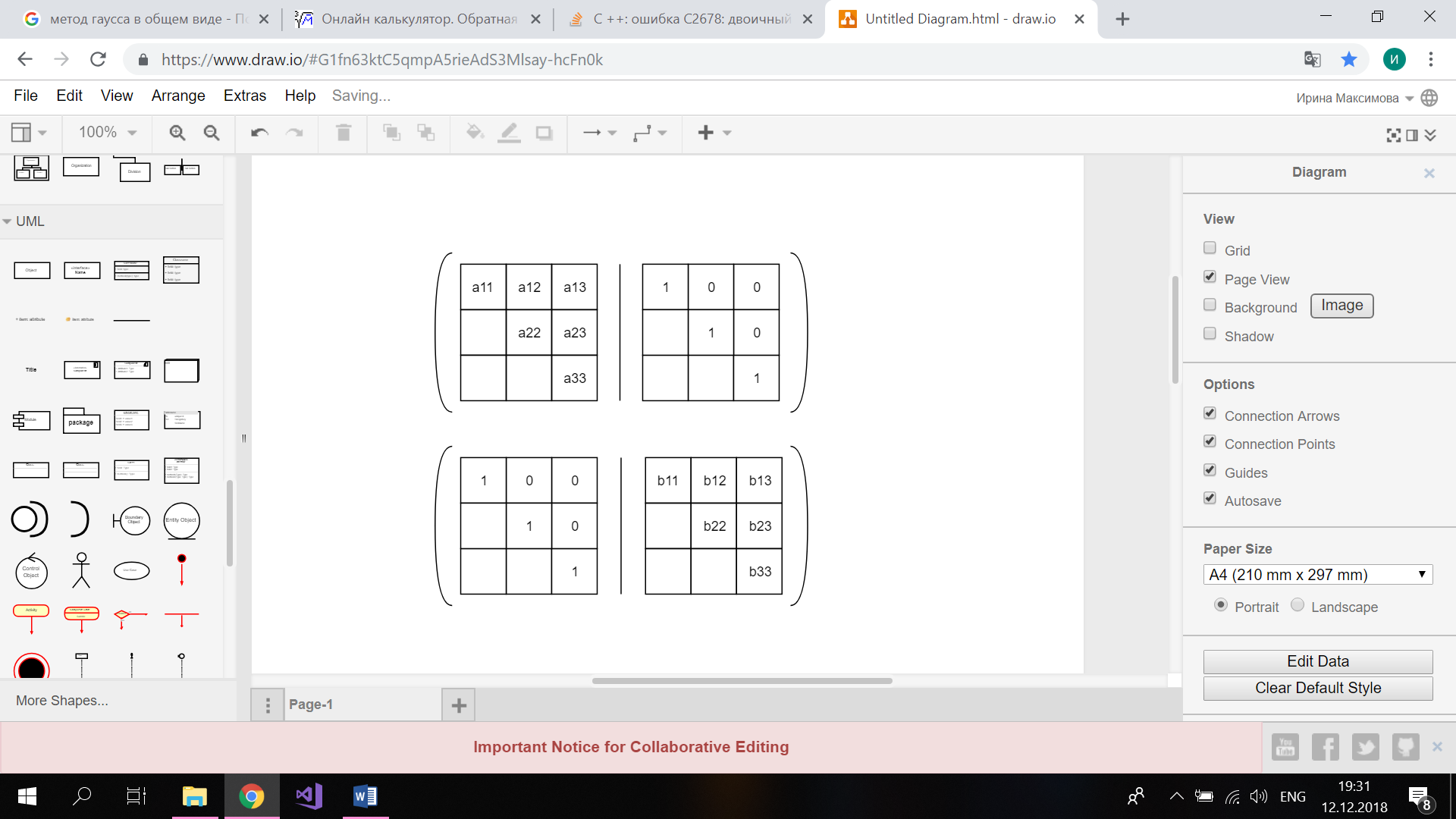


Рис. 8 (Расширенная матрица)

Далее с помощью элементарных преобразований строк матрицы справа от черты получаем единичную матрицу, а то что получается справа и есть искомая обратная матрица :

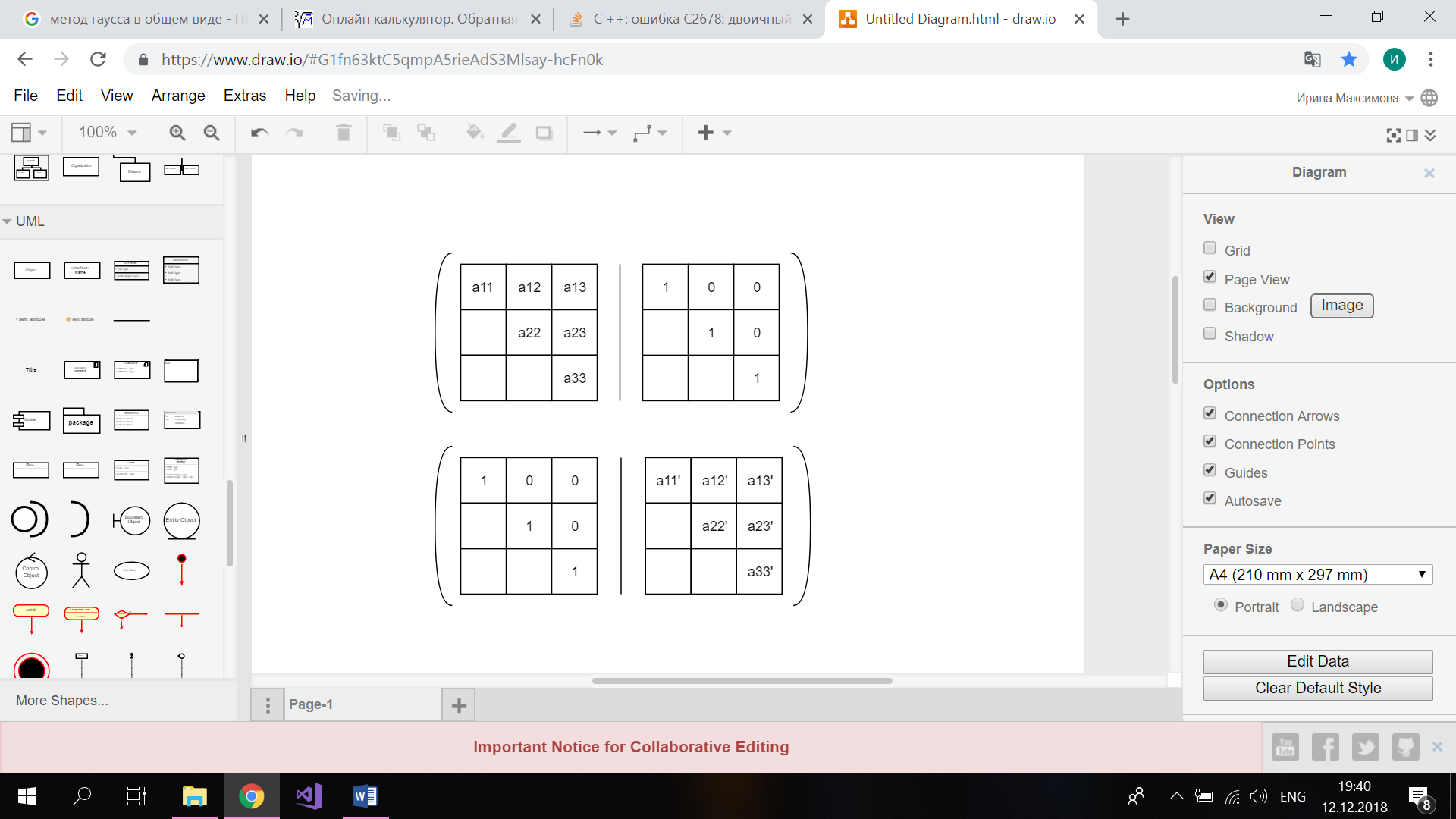


Рис. 9 (Конечная таблица метода Гаусса ))

# Эксперименты

Эксперименты проводились на ПК с следующими параметрами:

1. Операционная система: Windows 10 Домашняя
2. Процессор: Intel(R) Core™ i5-8250U CPU @ 1.60 GHz
3. Версия Visual Studio: 2017

|  |  |  |
| --- | --- | --- |
| Кол-во элементов | Время работы оператора сложения (в млс)  O(n2) | Время работы оператора умножения (в млс)  O(n3) |
| 1000 | 10 | 1775 |
| 3 000 | 80 | 65 548 |
| 6 000 | 324 | 665 548 |

В работе использовались классические методы сложения и умножения матриц для которых сложность составляет соответственно O(n2) и O(n3). Но из таблицы видно, что время ниже предполагаемого. Как же это объяснить?

Поскольку мы работаем только с верхнетреугольными матрицами и при этом используем метод храниния данных как вектор вектором, то по факту, мы храним только значения выше главной диагонали, никак не используя нижнюю половину. Тем самым отсекая практически половину значений, а значит экономя время примерно в 2 раза. Храня только заполненную часть матрицы, мы занимаем не памяти, а . За счет как раз этого и уменьшается время работы программы.

Вместо того чтобы хранить матрицу полностью используя память n2, мы храним только необходимую часть матрицы, занимая всего n\*(n+1)/2 памяти. За счет уменьшения объема используемой память и возрастет скорость работы программы, так как самой долгой операцией является операция доступа к данным.

# Заключение

Эта лабораторная работа позволила на практике познакомиться с методами представления верхнетреуголых матриц. Были реализованы как вспомогательный класс TVector, так и сам класс верхнетреугольных матриц TMatrix, в котором сами матрицы представлялись как вектор векторов. Тесты написаны по большей части вручную, что позволило лучше разобраться в принципе их работы и написания. Я смог добиться того, чтобы все тесты выполнялись, а пример использования работал без ошибок. Кроме этого, я провел оценку производительности такого метода хранения верхнетреугольных матриц.

1. **Литература**
2. Гергель В.П. Методические материалы по курсу «Методы программирования 2»: [<http://www.itmm.unn.ru/files/2018/10/Primer-1.2.-Struktury-hraneniya-matrits-spetsialnogo-vida.pdf>], 2015.
3. Википедия: свободная электронная энциклопедия: на русском языке [https://ru.wikipedia.org/wiki/Матрица\_(математика)]

[https://ru.wikipedia.org/wiki/Треугольная\_матрица]