Федеральное агентство по образованию Российской Федерации

Государственное образовательное учреждение

высшего профессионального образования

Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий математики и механики

Направление подготовки: «Фундаментальная информатика и информационные технологии»

Отчёт по лабораторной работе

**Разработка структуры хранения класса верхнетреугольных матриц.**

Выполнила:

студентка ИИТММ гр. 381906-2

Гордей М.В.

Проверила:

программист лаб. суперкомпьютерных технологий

и высокопроизводительных

вычислений каф. математического обеспечения

и суперкомпьютерных технологий

Усова М. А.

Нижний Новгород

2020 г.

Содержание

I. [Введение 3](#_Toc24457830)

II. [Постановка задачи](#_Toc24457831) 4

III. [Руководство пользователя](#_Toc24457832) 5

IV. [Руководство программиста](#_Toc24457833) 6

1. [Описание структуры программы](#_Toc24457834) 6

2. [Описание структур данных](#_Toc24457835) 7

3. [Описание алгоритмов](#_Toc24457836) 7

V. [Эксперименты](#_Toc24457837) 9

VI. [Заключение](#_Toc24457838) 10

VII. [Литература](#_Toc24457839) 11

VIII. [Приложение 1](#_Toc24457840)2

# Введение

Матрицы и векторы – важнейшие понятия в курсе линейной алгебры. Матрица в математическом понимании – объект, записываемый в виде прямоугольной таблицы, которая представляет собой совокупность строк и столбцов, на пересечении которых находятся её элементы. Векторами называются матрицы, состоящие из одной строки или одного столбца.

Чаще всего в математике работают с матрицами общего вида, но также в математических приложениях используются матрицы специального вида – треугольные, диагональные.

В этой работе созданы классы для работы с матрицами и векторами. Верхнетреугольные матрицы - квадратные матрицы, элементы которой ниже главной диагонали равны нулю.

# 2. Постановка задачи

Написать классы для работы с векторами и матрицами, использовать шаблоны.

Продемонстрировать корректную работу на примере (проверить правильность выполнения операций, проверка свойств матрицы)

Создание программных средств для эффективного хранения матриц верхнетреугольного вида, выполнение основных операций как сложение/вычитание умножение, копирование, сравнение. В процессе выполнения лабораторной работы требуется использовать систему контроля версий Git и фрэймворк для разработки автоматических тестов Google Test. Сделать выводы о проделанной работе.

# 3. Руководство пользователя

Запускаем программу:

1. Создать математические объекты типа Matrix или Vector, в зависимости от необходимости.
2. Расширить объекты при помощи доступа к защищенным полям(гетеры) или при помощи конструкторов
3. Выполнить необходимые математические операции, производимые с векторами и матрицами, применимые в алгебре (умножение, сложение, вычитание матриц и векторов при подходящем размере, умножение вектора на матрицу и наоборот при помощи матричного умножения), проверка работы перегруженных операций, конструкторов.

# 4. Руководство программиста

## 4.1. Описание структуры программы

Программа состоит из нескольких блоков:

ALL\_BUILD, gtest, ZERO\_CHECK - вспомогательные записи, ZERO\_CHECK перезапустит cmake. ALL\_BUILD - это цель, которая строит весь проект в активное решение

В блоке vectorlib содержится шаблонный класс Vector, объявлены все методы и их определения класса векторов, и его наследник My Matrix, в котором объявлены все методы класса матриц и их определения, объявлены и реализованы функции перегрузки операторов матрично-векторного умножения

В блоке app содержится один файл main с примером работы с классами класс Vector, Matrix

Также есть блок test, в которых содержится набор тестов, с помощью которых мы проверяем корректную работу программы

***4.2 Описание структур данных***

В программе определены два следующих шаблонных класса:

Class Vector,

Class Matrix

Внутри класса Vector определены следующие protected поля:

* T \*x – шаблонный указатель.
* Int length длина вектора.
* Int StartIndex– стартовая ячейка

Внутри класса Matrix определен следующий набор public-методов

(плюс конструкторы и деструктор):

* Vector() – конструктор по умолчанию, не принимает никаких параметров, инициализирует все поля 0
* Vector(Vector<T> & \_v) – конструктор копирования, принимает на вход объект типа TVector, создает объект с теми же характеристиками что и переданный, и массивом того же содержимого.
* Vector() – деструктор, очищает память;
* Int Lenght () – метод возвращающий длину вектора.
* void SetStartIndex () – метод, устанавливающий стартовый индекс
* T operator [](const int index) – перегрузка оператора индексации, позволяет возвращать элемент вектора по индексу;
* Vector<T> operator+(const Vector<T>& \_v) – перегрузка оператора суммы, позволяет возвращать вектор, являющийся суммой двух векторов;
* Vector<T> operator-(const Vector<T>& \_v) – перегрузка оператора разности, позволяет возвращать вектор, являющийся разностью двух векторов;
* Vector<T> operator = (const Vector<T>& \_v) – перегрузка оператора присвоить, позволяет присваивать один объект типа Vector другому;
* bool operator ==(const Vector<T>& \_v) – перегрузка оператора сравнения, позволяет сравнивать объекты типа Vector;
* bool operator !=(const Vector<T>& \_v) – перегрузка оператора не равно, true Если операнды имеют одинаковое значение; в противном случае возвращается false
* T operator\*(const Vector<T>& \_v) – перегрузка оператора умножить, позволяет перемножать объекты типа Vector;
* friend ostream& operator<<(std::ostream& stream, const Vector<T1>& A)– перегрузка оператора вывода, позволяет выводить вектора на экран;
* friend istream& operator>>(std::istream& stream, Vector<T1>& A) – перегрузка оператора ввода, позволяет вводить вектора

Шаблонный класс Matrix – публичный наследник класса Vector, полей в нем не определено.

* Matrix(const TMatrix&m) – конструктор копирования.
* Matrix(Vector<Vector<T> >&m) конструктор с параметром вектор
* Bool operator ==(const Matrix<T>&m) – перегрузка оператора равно, позволяет сравнивать объекты типа Matrix;
* Bool operator =(const Matrix&)const – перегрузка оператора присваивание
* Matrix operator +(const Matrix&m) – перегрузка оператора суммы
* Matrix operator -(const Matrix&m) - перегрузка оператора разности
* Matrix operator \*(const Matrix&m) - перегрузка оператора умножить
* friend ostream& operator <<(ostream&, Matrix<T1>&A) – перегрузка оператора вывода
* friend istream& operator >>(istream&, Matrix<T1>&A) - перегрузка оператора ввода

## 4.3 Описание алгоритмов

*Алгоритм суммирования или вычитания векторов*

Сначала выполняется проверка на совпадение размерности. Для сложения векторов необходимо сложить соответствующие координаты этих векторов. Создается вектор для записи результата. По циклу соответствующие координаты складываются или вычитаются, в результате работы возвращается результирующий вектор.

*Алгоритм умножения матриц:*

Если чисто столбцов в первой матрице совпадает с числом строк во второй, то эти матрицы можно перемножить. На входе выполняется эта проверка. Создается матрица для записи результата. В цикле каждая строка первой матрицы почленно умножается с каждым столбцом второй матрицы. Возвращается результирующая матрица.

Алгоритмы нахождения произведения вектора на матрицу и матрицы на вектор аналогичны алгоритму умножения матриц.

## 

# 5.Эксперименты

Проверим корректность выполнения нашей программы. Проверим свойства верхнетреугольных матриц.

Найдем определитель матрицы.Определитель верхнетреугольной матрицы равен произведению чисел, стоящих на главной диагонали. Проверим, корректны ли вычисления определителя в работе нашей программы.

Зададим две матрицы, посчитаем их определитель

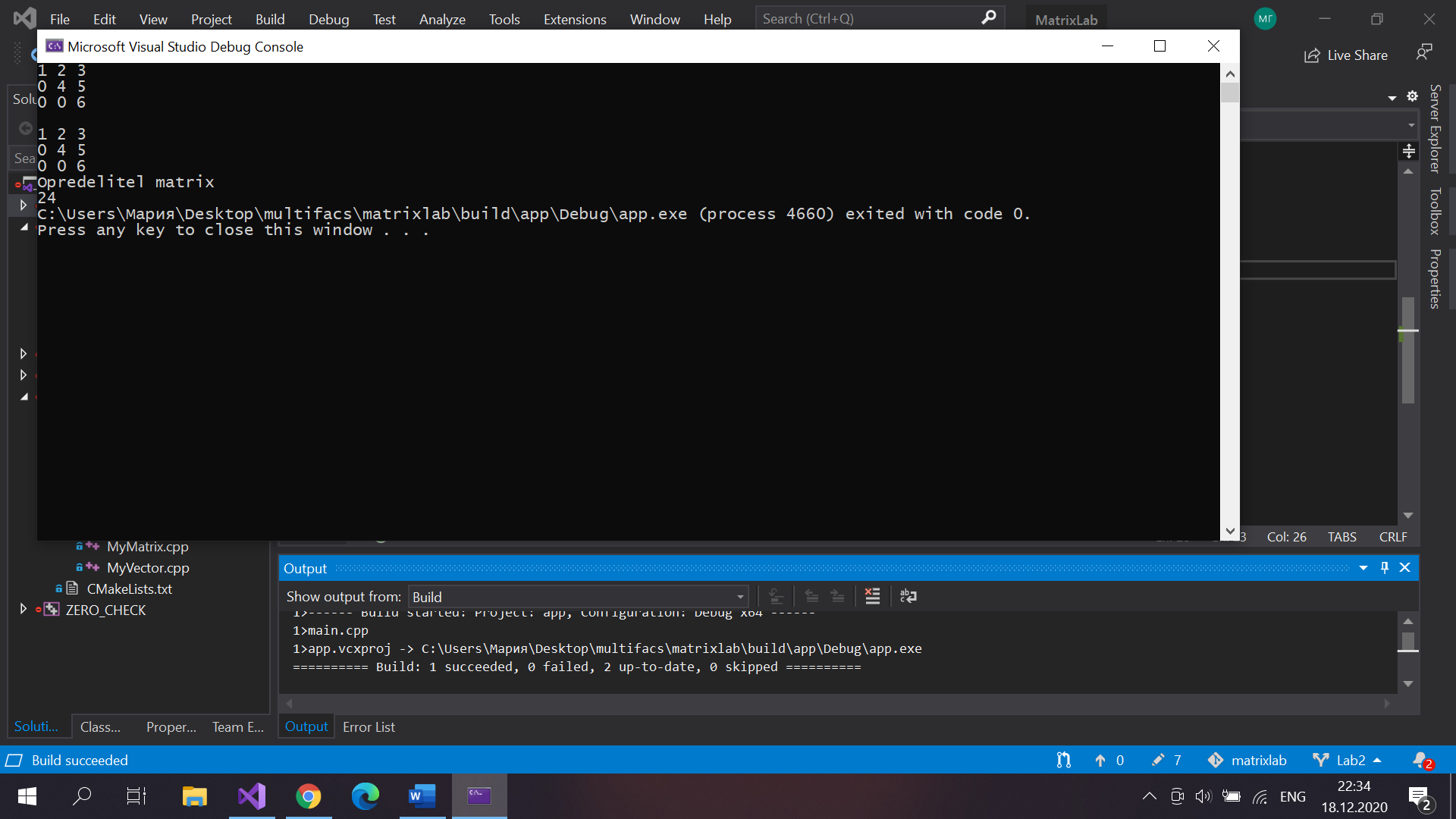


Рисунок 1: интерфейс работы программы(данные, необходимые для экспериментов в работе, вычисление определителя)

Перемножая элементы на главной диагонали видим, что программа работает корректно.

Проверим результат работы умножения матриц.

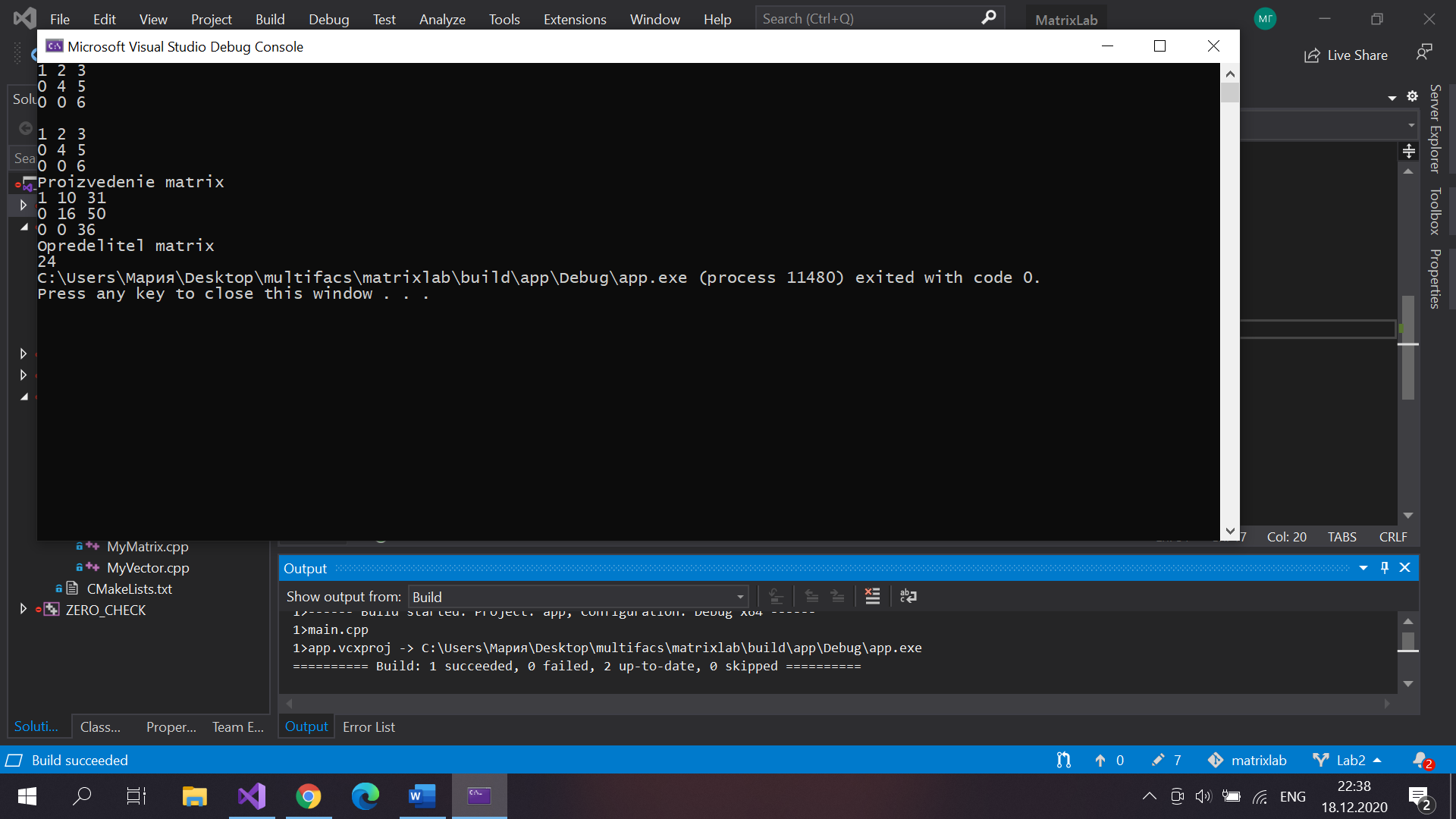


Рисунок 2: вычисление произведения двух матриц

Программа вычисляет произведение корректно.

# 6. Заключение

По итогу выполнения данной работы были написаны классы, позволяющие осуществлять удобную работу с матрицами и векторами. Были переопределены арифметические операции для работы с матрицами, реализованными в данной структуре, проведены эксперименты, подтверждающие корректную работу программы. Также работоспособность программы проверена на тестах.

# 7. Литература

1. <https://ru.wikipedia.org>
2. <https://habr.com>
3. Керниган, Б.У. Язык программирования С / Б.У. Керниган, Д.М. Ритчи

# 8. Приложение