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**

**«Разработка структуры хранения матриц специального вида»**

**Выполнил:** студент группы 381706-1

Нечаева Екатерина Владимировна

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

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

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc560926)

[2. Постановка задачи 4](#_Toc560927)

[3. Руководство пользователя 5](#_Toc560928)

[4. Руководство программиста 6](#_Toc560929)

[4.1 Описание структуры программы 6](#_Toc560930)

[4.2 Описание структур данных 6](#_Toc560931)

[4.3 Описание алгоритмов 7](#_Toc560932)

[5. Заключение 10](#_Toc560933)

[6. Литература 11](#_Toc560934)

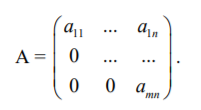
# Введение

Матрица – это таблица из m строк и n столбцов. Элементами матрицы могут быть объекты совершенно разнообразной природы: числа, переменные или, к примеру, иные матрицы.

Матрицы широко применяются в математике и физике для компактной записи и решения систем линейных алгебраических уравнений и систем дифференциальных уравнений. При этом количество строк матрицы соответствует количеству уравнений системы, а количество столбцов — количеству неизвестных величин. Матричный аппарат позволяет существенно упростить решение СЛАУ сведя его к операциям над матрицами.

Помимо матриц общего вида (прямоугольных) в математических приложениях выделяются различные матрицы специальных видов (треугольные, диагональные и т.д.). Для таких матриц предпочтительно создание собственных способов хранения и обработки, учитывающих специфику их структуры. И с учетом этого они могут оказать более эффективными.

В данной лабораторной работе будут рассмотрены верхнетреугольные матрицы. Верхнетреугольной называется матрица, в которой все элементы под главной диагональю равны нулю:



# Постановка задачи

В рамках лабораторной работы ставится задача реализации структуры данных, представляющей собой верхнетреугольную матрицу.

Для векторов необходимо реализовать:

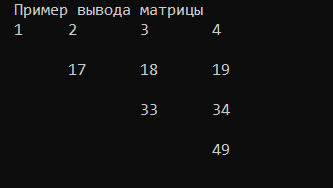
* индексация,
* присваивание одного вектора к другому,
* сложение и вычитание векторов,
* скалярное произведение векторов,
* умножение, сложение и вычитание вектора и числа,
* сравнение векторов,
* перегрузка ввода/вывода.

В свою очередь, для матриц будет представлены:

* присваивание одной матрицы другой,
* сложение и вычитание двух матриц,
* умножение двух матриц,
* деление двух матриц,
* сравнение матриц,
* перегрузка ввода/вывода.

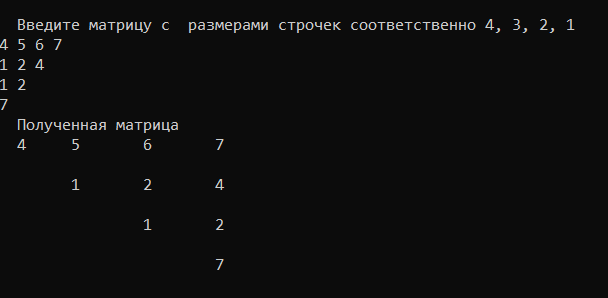
# Руководство пользователя

Программа начинает свою работу с вывода на консоль автоматически созданную матрицу.



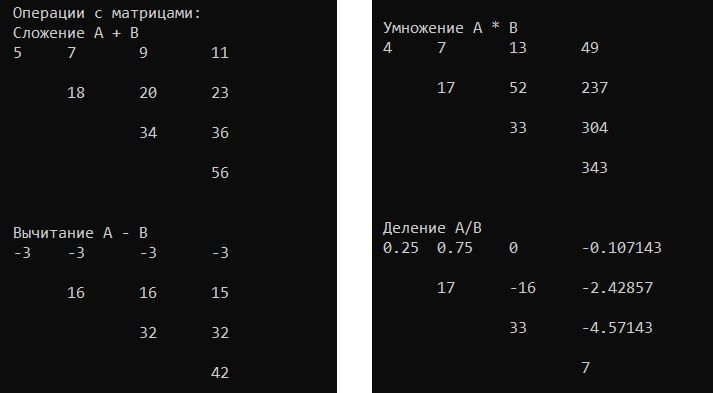
*Рисунок 1. Вывод на консоль матрицы.*

Следующим шагом программа запрашивает у пользователя ввести свою матрицу заданного размерами векторов. И выводит ее на консоль.



*Рисунок 2. Пример ввода матрицы пользователем.*

Последним шагом программа выполняет матричные операции с полученными данными, а именно: сложение, вычитание, умножение и деление матриц.

**

*Рисунок 3. Операции с матрицами.*

# Руководство программиста

## 4.1 Описание структуры программы

Программа состоит из следующих модулей:

* Модуль vector. Содержит пример использования программы с помощью классов TVector и TMatrix (mainvector.cpp)
* Модуль libvector – статическая библиотека. Содержит заголовочный файл vector.h, в котором реализован шаблонные классы TVector и TMatrix.
* Модуль vectorTect. Содержит тесты, проверяющие правильность работы методов(vectorTest.cpp и matrixTest.cpp)
* Модуль libthrows – библиотека исключений. Содержит заголовочный файл throws.h, в котором он реализован.

## Описание структур данных

#### Класс TException – класс исключений.

Содержит одно защищенное поле *string str,* переменная строкового типа, которая хранит сообщение об исключении.

Содержит публичные методы

* TException(std::string \_str); //конструктор-инициализатор
* void Print(); //вывод данного исключения на консоль

#### Класс TVector.

Защищенные поля:

* T\* vector;
* int size; //размер вектора
* int FirstInd; // стартовый индекс

Содержит публичные методы:

* TVector(const TVector<T> &A); //конструктор копирования
* TVector(int \_size = 5, int \_FirstInd = 0); //конструктор инициализации
* ~TVector();
* int GetSize(); // получить размер вектора
* int GetFirstInd(); // получить стартовый индекс
* TVector<T>& operator=(const TVector<T> &A); //присваивание
* T& operator[](int i); //индексация
* bool operator==(const TVector<T> &A); //сравнение
* bool operator!=(const TVector<T> &A); //сравнение
* TVector<T> operator\*(const T &a); // умножение на число
* TVector<T> operator+(const T &a); //прибавление числа
* TVector<T> operator-(const T &a); //вычитание числа
* TVector<T> operator+(const TVector<T> &A); //сложение
* TVector<T> operator-(const TVector<T> &A); //вычитание
* T operator\*(const TVector<T> &A); //умножение

**Класс TMatrix.**

Полей у данного класса нет, так как он наследуется от класса TVector.

Содержит публичные методы:

* TMatrix(int \_n = 3);
* TMatrix(const TMatrix<T> &A);
* TMatrix(const TVector<TVector<T> > &A); // конструктор преобразования типа
* bool operator==(const TMatrix<T> &A); //сравнение
* bool operator!=(const TMatrix<T> &A); //сравнение
* TMatrix& operator=(const TMatrix<T> &A); //присваивание
* TMatrix operator+(const TMatrix<T> &A); //сложение матриц
* TMatrix operator-(const TMatrix<T> &A); // вычитание
* TMatrix operator\*(const TMatrix<T> &A); // умножение матриц
* TMatrix operator/(const TMatrix<T> &A); // деление матриц

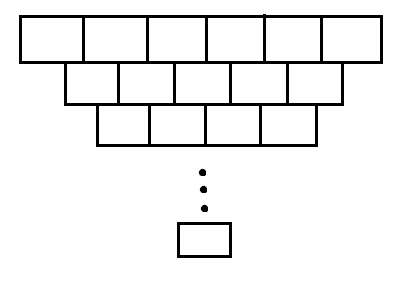
## Описание алгоритмов

Рассмотрим некоторые алгоритмы для класса TMatrix:

**Создание матрицы – конструктор инициализации, конструктор копирования и конструктор преобразования типа.**

* Конструктор инициализации.

Этот конструктор вызывает конструктор-иницализатор класса TVector с шаблоном <T> равным TVector<T>. Таким образом создается вектор, обозначим его за А, элементами которого также являются векторы, заданного размера, который принимает конструктор. После чего каждому элементу А, а именно векторам, присваивается размер, начиная от размера самой матрицы и уменьшая на единицу, и его стартовый индекс с помощью того же конструктора класса TVector, но с шаблоном <T>. В итоге получается треугольная матрица.

Первый вектор вектора А.

n-ый вектор вектора А.

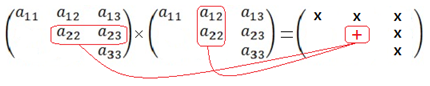
* Конструктор копирования.

Для того, чтобы реализовать этот конструктор, достаточно вызвать конструктор копирования класса TVector c шаблоном TVector<T>.

* Конструктор преобразования типа.

Работает аналогично, только этот конструктор принимает в качестве параметра не элемент класса TMatrix<T>, а элемент класса TVector<TVector<T>>.

**Умножение матриц.**

Произведением матриц A и B называется такая матрица С, элементы которой находятся по такому правилу: сij равен сумме произведений элементов i-той строки матрицы A на соответствующие элементы j-го столбца матрицы B . Таким образом, умножение осуществляется по правилу умножения строки на столбец.

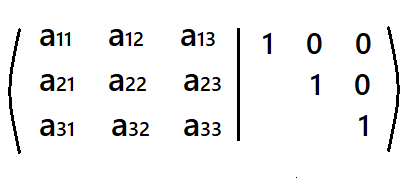
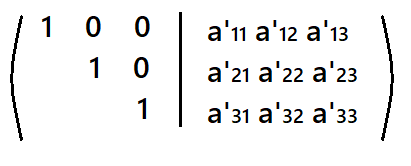
*Рисунок 4. Схематическое нахождение произведения матриц.*

**Деление матриц.**

Слово «деление» заключено в кавычки, потому что на самом деле матрицы делить нельзя. Операция деления заменяется операцией умножения одной матрица на матрицу, которая обратна второй матрице: C = D\*, где - обратная матрица к матрице A.

Чтобы найти обратную матрицу, воспользуемся методом Гаусса.

Сначала следует записать квадратную матрицу в тандеме с единичной матрицей: (A|E). Затем с помощью элементарных преобразований необходимо получить единичную матрицу слева, при этом справа нарисуется обратная матрица. Схематически решение выглядит следующим образом:



**🡪…🡪**

*Рисунок 5. Схематическое нахождение обратной матрицы методом Гаусса.*

# Заключение

В ходе выполнения данной лабораторной была реализована структура хранения матриц специального вида, а именно верхнетреугольной. Были реализованы методы, позволяющие выполнять операции с векторами и матрицами, а именно: сложение, вычитание, умножение векторов/матриц и деление матриц.

Таким образом, мы рассмотрели виды матриц, операции над ними и их применение в разных областях.

Были разработаны тесты, проверяющие корректную работу методов данных классов, с помощью Google C++ Testing Framework .

# Литература

1. Применение матрицы [Электронный ресурс]

<http://topuch.ru/interesnie-primeneniya-matrici-v-matematike-fizike-ekonomike-b/index.html>

1. Матрицы и их виды [Электронный ресурс]

<https://math1.ru/education/matrix/matrix.html>

1. Учебно-методическое пособие [Электронный ресурс]

<http://www.unn.ru/books/met_files/Pract_ADS.pdf>

1. Умножение матриц [Электронный ресурс]

<http://ru.solverbook.com/spravochnik/matricy/umnozhenie-matric/>