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**

**«Верхнетреугольная матрица»**

**Выполнил:** студент группы 381706-1

Суслов Егор Игоревич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

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

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc533083472)

[2. Постановка задачи 4](#_Toc533083473)

[3. Руководство пользователя 5](#_Toc533083474)

[4. Руководство программиста 6](#_Toc533083475)

[4.1. Описание структуры программы 6](#_Toc533083476)

[4.2. Описание структур данных 6](#_Toc533083477)

[4.3. Описание алгоритмов 7](#_Toc533083478)

[5. Заключение 8](#_Toc533083479)

[6. Литература 9](#_Toc533083480)

# Введение

Понятие Матрица в европейской науке было введено в работах У. Гамильтона и А. Кэли в середине XIX века.

Матричные обозначения широко распространены в современной математике и её приложениях. Матрица – полезный аппарат для исследования многих задач теоретической и прикладной математики. Так, одной из важнейших является задача нахождения решения систем линейных алгебраических уравнений.

Следствием разнообразия областей применения матричного аппарата в современной науке является наличие в любом из больших математических программных комплексов (Mathcad, Mathematica, Derive, Mapple) подсистем, выполняющих операции над матрицами, а также существование специальных программных библиотек (ScalaPack, PlaPack), рассчитанных на обработку огромных (десятки и сотни тысяч строк) матриц, в том числе с использованием распределенных (параллельных) вычислений.

Помимо матриц общего вида, для которых наиболее естественной и наиболее часто используемой представляется программная реализация в виде двумерного массива, 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 потому более эффективных.

В данной работе рассматривается один из подходов к созданию класса оперирующего с верхнетреуголными матрицами.

Верхнетреугольной называется матрица, в которой все элементы под главной диагональю равны нулю:

Определение матрицы возможно также через понятие Вектор. Вектор – в математике – набор a[i] (чисел, математических выражений), состоящий из n элементов.

Тогда Матрица из m строк и n столбцов может быть определена как Вектор из n элементов, где каждый элемент, в свою очередь, является вектором из m элементов.

# Постановка задачи

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

• сложение/вычитание;

• умножение;

• копирование;

• сравнение.

Программные средства должны содержать:

• класс Вектор (на шаблонах);

• класс Матрица (на шаблонах);

• тестовое приложение, позволяющее задавать матрицы и осуществлять основные операции над ними.

# Руководство пользователя

При запуске консольного приложения «Мatrix\_main.exe» пользователю выводятся на консоль сначала две верхнетреугольные матрицы без инициализации (все значения выше главной диагонали заполнены нулями , а ниже главной диагонали пусто), затем выводятся две аналогичные, но уже инициализированные числами (наперед заданными в «Matrix\_main.cpp») матрицы и производятся действия (сложение, вычитание, умножение, деление) с выводом результата и описанием каждого действия на консоль.

# Руководство программиста

## Описание структуры программы

Программа состоит из следующих модулей:

* VectorLib - статическая библиотека содержащая заголовочный файл «Vector.h» с шаблонным классом TVector и файл кода «Vector.cpp».
* MatrixLib - статическая библиотека содержащая заголовочный файл «Matrix.h» с шаблонным классом TMatrix и файл кода «Matrix.cpp».
* MatrixTest и VectorTest содержат 21 и 23 теста, описанные в файлах «MatrixTest.cpp» и «VectorTest.cpp» соответственно, разработанных с помощью использования Google C++ Testing Framework.
* Vector и Matrix в которых содержаться файлы кода «Vector\_main.cpp» и «Matrix\_main.cpp» которые наглядно демонстрируют использование вышеописанных библиотек.
* Модуль ExceptionLib – библиотека, содержащая класс исключений.

## Описание структур данных

#### 4.2.1 Класс TVector

Класс TVector является шаблонным классом. В нем определены поля со спецификатором доступа protected:

* T \*pvector; - указатель на вектор
* int size; - размер вектора

Далее методы со спецификатором доступа public:

* TVector(int s = 0); - конструктор по умолчанию.
* TVector(const TVector &v); - конструктор копирования.
* virtual ~TVector(); - деструктор.
* int GetSize() const; - метод возвращающий размер вектора.
* virtual T& operator[](int pos); - оператор индексации.
* bool operator==(const TVector &v) const; - сравнение.
* bool operator!=(const TVector &v) const; - сравнение.
* virtual TVector& operator=(const TVector &v); - присваивание.
* TVector operator+(const T &val); - прибавить число.
* TVector operator-(const T &val); - вычесть число.
* TVector operator\*(const T &val); - умножить на число.
* TVector operator+(const TVector &v); - сложение векторов.
* TVector operator-(const TVector &v); - вычитание векторов.
* T operator\*(const TVector &v); - скалярное произведение векторов.
* template <class Type>

friend istream& operator>>(istream &in, TVector<Type>& v); - перегрузка ввода векторов.

* template <class Type>

friend ostream& operator<<(ostream &out, const TVector<Type>& v); - перегрузка вывода для векторов

#### 4.2.1 Класс TMatrix

#### Наследуется от «вектора векторов» (class TMatrix : public TVector<TVector<T> >)

Содержит public методы:

* TMatrix(int s = 5); - конструктор по умолчанию.
* TMatrix(const TMatrix &mt); - конструктор копирования принимающий матрицу.
* TMatrix(const TVector<TVector<T> > &mt); ); - конструктор копирования принимающий «вектор векторов».
* virtual ~TMatrix(); - деструктор.
* bool operator==(const TMatrix &mt) const; - сравнение матриц.
* bool operator!=(const TMatrix &mt) const; - сравнение матриц.
* TMatrix& operator=(const TMatrix &mt); - присваивание матрицы
* TMatrix operator+(const TMatrix &mt); - сложение матриц.
* TMatrix operator-(const TMatrix &mt) – вычитание матриц.
* TMatrix<T> operator\*(const TMatrix<T> &mt); - умножение матриц.
* TMatrix<T> operator/(const TMatrix<T> &mt); - деление матриц.
* template <class Type>

friend istream& operator>>(istream &in, TMatrix<Type> &mt); - перегрузка ввода матриц.

* template <class Type>

friend ostream& operator<<(ostream &out, const TMatrix<Type> &mt); -перегрузка вывода для матриц

## Описание алгоритмов

#### Так как класс TMatrix yнаследуется от «вектора векторов» (class TMatrix : public TVector<TVector<T> >), то все операции с матрицами выполняются над векторами.

# Заключение

Все поставленные задачи были выполнены это подтверждается Google C++ Testing Framework (все 44 теста на обе статические библиотеки выполняются успешно) и успешным выполнение файлов кода «Vector\_main.cpp» и «Matrix\_main.cpp» (успешно создаются консольные приложения работающие с векторами и матрицами – «векторами векторов» соответственно.

# Литература

* Книги:

1. A.O. Грудзинский. Методы программирования, Издательство Нижегородского госуниверситета, 2006.
2. Топп У., Форд У. Структуры данных в С++. - М. Бином, 1999.
3. Мейн М., Савитч У. Структуры данных и другие объекты в С++. - М.: Издательский дом “Вильямс”, 2003.
4. Гергель В.П. Методические материалы по курсу «Методы программирования 2», Нижний Новгород, 2015.

* Ссылки в Internet:

1. Кодировка [https://vscode.ru/articles/encoding-rus-files.html].
2. Учебно-методическое пособие из электронной библиотеки ННГУ: «ЛАБОРАТОРНЫЙ ПРАКТИКУМ» : [http://www.unn.ru/books/met\_files/Pract\_ADS.pdf].