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

Шашкин Евгений Вадимович

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

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

Нижний Новгород

2018.

**Оглавление**

[1. Введение. 2](#_Toc533105642)

[2. Постановка задачи. 3](#_Toc533105643)

[3. Руководство пользователя. 4](#_Toc533105644)

[4. Руководство программиста. 5](#_Toc533105645)

[4.1. Описание структуры программы. 5](#_Toc533105646)

[4.2. Описание структур данных. 5](#_Toc533105647)

[4.3. Описание алгоритмов. 6](#_Toc533105648)

[5. Эксперименты. 7](#_Toc533105649)

[6. Заключение. 8](#_Toc533105650)

[7. Литература. 9](#_Toc533105651)

# Введение.

**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. Хотя исторически рассматривались, например, треугольные матрицы, в настоящее время говорят исключительно о матрицах прямоугольной формы, так как они являются наиболее удобными и общими.

Существуют треугольные матрицы, которые в свою очередь делятся на верхнюю треугольную матрицу и нижнюю треугольную матрицу. Верхней треугольной матрицей называется квадратная матрица, у которой все элементы ниже главной диагонали равны нулю. А нижней треугольной матрицей называется матрица, у которой все элементы выше главной диагонали равны нулю.

В программировании матрицей называется двумерный массив. Для представления вышеописанных частных случаев квадратных матриц (верхние треугольные, нижние треугольные и т.д.) удобнее разработать отдельную структуру хранения и обработки. Так, например, для хранения верхних треугольных матриц необязательно хранить нули. Достаточно хранить элементы выше главной диагонали и саму главную диагональ.

# Постановка задачи.

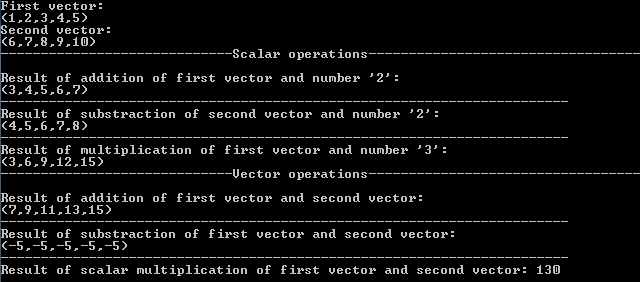
Нужно разработать структуру данных, которая хранит в памяти верхнюю треугольную матрицу, как массив векторов. Для этого нужно:

* Реализовать класс векторов **TVector**.
* Протестировать класс **TVector**.
* Реализовать класс матриц **TMatrix**, как наследника от класса **TVector**.
* Протестировать класс **TMatrix**.
* Продемонстрировать работу классов **TVector** и **TMatrix**.
* Реализовать класс исключений **TMyException** для обработки исключений, которые могут возникнуть в процессе работы программы.

# Руководство пользователя.

Рассмотрим пример использования класс **TVector**:

При запуске программы на экран выводятся два вектора и результаты операций, проведенных над ними.



*Рис.1 Демонстрация класса TVector.*

Теперь рассмотрим пример использования класса **TMatrix**:

В начале работы программы на экран выводятся две верхних треугольных матрицы.

Затем выводятся результаты операций над матрицами, так же, как и в примере использования класса **TVector**.

# Руководство программиста.

# Описание структуры программы.

Программа состоит из модулей:

* **Vector** – содержит в себе файл **vector\_main.cpp** с реализацией примера использование класса **TVector**.
* **VectorLib –** содержит в себе файл **Vector.h**, в котором описан и реализован класс **TVector**.
* **VectorTest** – содержит в себе файл **test\_vector.cpp**, в котором находится набор тестов, для проверки работоспособности класса **TVector**.
* **Matrix** – содержит в себе файл **matrix\_main.cpp** с реализацией примера использование класса **TMatrix**.
* **MatrixLib –** содержит в себе файл **Matrix.h**, в котором описан и реализован класс **TMatrix**.
* **MatrixTest** – содержит в себе файл **test\_matrix.cpp**, в котором находится набор тестов, для проверки работоспособности класса **TMatrix**.

# Описание структур данных.

**Класс TVector.**

Класс **TVector** является шаблонным и содержит два поля со спецификатором **protected**:

* **T\* vector** – указатель на область памяти для хранения вектора.
* **int size** – размер вектора.

Далее идут методы класса со спецификатором доступа **public**:

* **TVector(int \_size=10)** – конструктор инициализатор.
* **TVector(TVector<T> &obj)** – конструктор копирования.
* **~TVector()** – деструктор.
* **int GetSize()** – возвращает размер вектора.
* **T& operator[] (int index)** – оператор доступа по индексу.
* **bool operator==(const TVector<T> &obj)** – оператор сравнения.
* **TVector<T>& operator=(const TVector<T> &obj)** – оператор присваивания.
* **TVector<T> operator+(const T &Scalar)** – оператор прибавления к вектору скаляра.
* **TVector<T> operator-(const T &Scalar)** – оператор вычитания из вектора скаляра.
* **TVector<T> operator\*(const T &Scalar)** – оператор умножения вектора на скаляр.
* **TVector<T> operator+(const TVector<T> &obj)** – оператор сложения двух векторов.
* **TVector<T> operator-(const TVector<T> &obj)** – оператор разности двух векторов.
* **T operator\*(const TVector<T> &obj)** – оператор скалярного умножения двух векторов.
* **friend std::istream& operator >> (std::istream &istr, TVector<T1> &vec)** – дружественная функция ввода вектора в поток.
* **friend std::ostream& operator << (std::ostream &ostr, TVector<T1> &vec)** – дружественная функция вывода вектора на экран.

**Класс TMatrix.**

Класс **TMatrix** унаследован от класса **TVector** и не имеет собственных полей со спецификатором доступа **protected**.

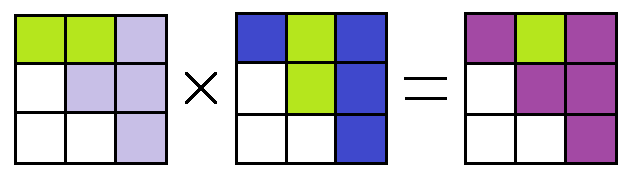
Методы со спецификатором доступа **public**:

* **TMatrix(int \_size = 10)** – конструктор инициализатор.
* **TMatrix(TMatrix<T> &obj)** – конструктор копирования.
* **TMatrix(TVector<TVector<T> > &obj)** – конструктор преобразования типа.
* **bool operator==(const TMatrix<T> &obj)** – оператор сравнения.
* **TMatrix<T>& operator=(const TMatrix<T> &obj)** – оператор присваивания.
* **TMatrix<T> operator+(const TMatrix<T> &obj)** – оператор сложения матриц.
* **TMatrix<T> operator-(const TMatrix<T> &obj)** – оператор разницы матриц.
* **TMatrix<T> operator\*(const TMatrix<T> &obj)** – оператор умножения матриц.
* **TMatrix<T> operator/(TMatrix<T> &obj)** – оператор деления матриц.
* **friend std::istream& operator >> (std::istream &istr, TMatrix<T1> &matrix)** – дружественная функция ввода матрицы в поток.
* **friend std::ostream& operator << (std::ostream &ostr, TMatrix<T1> &matrix)** – дружественная функция вывода матрицы на экран.

# Описание алгоритмов.

**Умножение матриц.**

Элементами матрицы C=A\*B являются равен сумме произведений элементов *i*-ой строки матрицы А на соответствующие элементы *j*-го столбца матрицы B.



*Рис. 2 Умножение верхних треугольных матриц.*

**Деление матриц.**

Результатом деления матрицы A на матрицу B является матрица где – матрица, обратная к матрице B*.* Обратная матрица может быть найдена следующим образом:

* Записываем матрицу
* С помощью элементарных преобразований строк нужно получить единичную матрицу справа .

Тогда матрица - обратная матрица к матрице B (.

# Эксперименты.

Параметры ПК:

* Операционная система: Windows 7 Ultimate.
* Процессор: AMD Athlon™ II X4 635 Processor 2.90 GHz.
* Версия Visual Studio: 2017.

|  |  |  |
| --- | --- | --- |
| Размер матрицы | Время работы оператора сложения (в секундах) | Время работы оператора умножения (в секундах) |
| 1000 | 0.01 | 1.868 |
| 5000 | 0.192 | 418.94 |
| 10000 | 1.305 | 6489.02 |

Таким образом, мы можем видеть, что операция сложения матриц имеет сложность O(), а операция умножения матриц имеет сложность O().

# Заключение.

В данной лабораторной работе мне удалось реализовать библиотеку для хранения и работы с верхними треугольными матрицами, а именно:

* Удалось реализовать класс **TVector**.
* Удалось протестировать класс **TVector** и обеспечить его работоспособность.
* Удалось реализовать класс **TMatrix**.
* Удалось протестировать класс **TMatrix** и обеспечить его работоспособность.
* Провести оценку производительности сложения и умножения матриц.

Таким образом, я смог реализовать структуру данных для хранения верхних треугольных матриц в памяти, как массива векторов.

# Литература.

1. Ссылка из Википедии про матрицы:

<https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D1%80%D0%B8%D1%86%D0%B0_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)>

1. Ссылка из Википедии про треугольные матрицы:

<https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B5%D1%83%D0%B3%D0%BE%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86%D0%B0>

1. Гергель В.П. Методические материалы по курсу «Методы программирования 2», 2015.