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 для матриц специального вида»**

**Выполнил:** студент группы 381706-1

Власов Андрей Сергеевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

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

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc532232359)

[2. Постановка задачи 4](#_Toc532232360)

[3. Руководство пользователя 5](#_Toc532232361)

[4. Руководство программиста](#_Toc532232362) 6

[4.1 Описание структуры программы](#_Toc532232363) 6

[4.2 Описание структур данных](#_Toc532232364) 6

[4.3 Описание алгоритмов](#_Toc532232365) 8

[5. Заключение 1](#_Toc532232366)0

[6. Литература 1](#_Toc532232367)1

# 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.

В данной лабораторной работе будет рассмотрена матрица специального вида – верхнетреугольная.

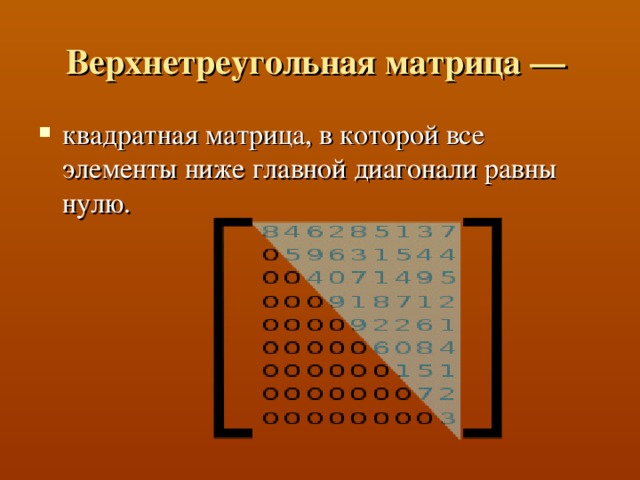


Рисунок . Верхнетреугольная матрица

Представлять такие матрицы в виде двумерного массива нерационально, поэтому для экономии памяти будут хранится только элементы главной диагонали и выше. Для этого будут использоваться вектора.

Цель данной лабораторной работы: реализация структуры данных для верхнетреугольных матриц.

# Постановка задачи

Выполнение работы предполагает решение следующих задач:

1. Разработка и реализация вспомогательного класса для создания верхнетреугольных матриц TVector.
2. Разработка и реализация класса матриц TMatrix
3. Пример использования классов TMatrix и TVector.
4. Набор автоматических тестов с использованием Google C++ Testing Framework.

# Руководство пользователя

Рассмотрим пример использования классов TMatrix и TVector. Подробно рассмотрим только класс TMatrix, т.к. TVector проще и во многом очень похож на TMatrix.

При запуске программы на консоль будут выведены 2 автоматически созданные матрицы. Ниже будут расположены результаты выполнения операций сложения, вычитания и умножения между ними.

Затем пользователю будет предложено ввести матрицу с консоли. Сразу после ввода она будет выведена на экран. На этом работа программы прекращается.

# Руководство программиста

## 4.1 Описание структуры программы

Программа состоит из следующих модулей:

* Модули *Vector* и *Matrix*.

Содержат примеры реализаций программ с использованием классов вектор и матрица соответственно. Файлы с реализацией *Vector\_ main.cpp* и *Matrix­\_ main.cpp.*

* Модуль *VectorLib*.

Содержит файл *Vector.h*, в котором определен интерфейс и реализация шаблонного класса вектор *TVector*.

* Модуль *MatrixLib*.

Содержит файл *Matrix.h*, в котором определен интерфейс и реализация шаблонного класса матрица *TMatrix.*

* Модули *VectorTest* и *MatrixTest*.

Содержат наборы тестов для классов *TVector* и *TMatrix*, реализованные в файлах *VectorTest.cpp* и *MatrixTest.cpp* с помощью использования фреймворка Google Test.

## Описание структур данных

#### Класс TVector

Рассмотрим класс *TVector* подробно. Класс является шаблонным.

**Дружественные функции:**

*friend istream& operator>>(istream &is, TVector<FriendT> &V)* – ввод вектора через консоль. Принимает ссылку на стандартный поток ввода и ссылку на объект класса *TVector*, возвращает ссылку на стандартный поток ввода.

friend ostream& operator<<(ostream &os, const TVector<FriendT> &V) – вывод вектора на консоль. Принимает ссылку на стандартный поток вывода и ссылку на объект класса *TVector*, возвращает ссылку на стандартный поток вывода.

**Поля класса , объявленные со спецификатором private**:

*int size* – размерность вектора.

*T \*vec* – указатель на область памяти для хранения вектора.

**Конструкторы и методы класса объявлены со спецификатором public:**

*TVector<T>(int n = 0)* – конструктор по умолчанию.

*TVector<T>(const TVector<T> &V)* – конструктор копирования.

*virtual ~TVector<T>()* – деструктор.

**Методы для работы с классом TVector:**

*int GetSize() const* – возвращает размер вектора.

*T& operator[](int i)* – 0-based индексация.

*bool operator==(const TVector<T> &V) –* проверка векторов на равенство. Возвращает true, если равенство выполняется, false в противном случае.

*bool operator!=(const TVector<T> &V)* – проверка векторов на неравенство. Возвращает true, если неравенство выполняется, false в противном случае.

**Перегрузка операторов:**

*TVector operator-(const T &val) –* вычесть из вектора скаляр. Создается временный вектор. Каждая координата исходного вектора уменьшается на данное число.

*TVector operator+(const T &val) –* прибавить к вектору скаляр. Создается временный вектор. Каждая координата исходного вектора увеличивается на данное число.

*TVector operator\*(const T &val)* – умножение вектора на число. Создается временный вектор. Каждая координата исходного вектора умножается на данное число.

*TVector& operator=(const TVector<T> &v)* – оператор присваивания одного вектора другому.

*TVector operator+(const TVector<T> &v)* – сложение векторов. Если размерности векторов совпадают, то создается временный вектор, куда записывается результат поэлементного сложения соответствующих координат двух векторов.

*TVector operator-(const TVector<T> &v)* – вычитание векторов. Если размерности векторов совпадают, то создается временный вектор, куда записывается результат поэлементного вычитания соответствующих координат двух векторов.

*T operator\*(const TVector<T> &v) –* скалярное произведение. Если размерности векторов совпадают, то создается временная переменная, в которую записывается сумма произведений соответствующих координат.

#### Класс TMatrix – верхнетреугольная матрица.

template <class T> class TMatrix : public TVector<TVector<T>> – класс матриц является шаблонным и является наследником TVector как «вектор векторов».

**Дружественные функции:**

*friend istream& operator>>(istream &in, TMatrix<FriendT> &MT)* – ввод матрицы через консоль. Принимает ссылку на стандартный поток ввода и ссылку на объект класса *TMatrix*, возвращает ссылку на стандартный поток ввода.

friend ostream& operator<<(ostream &out, const TMatrix<FriendT> &MT) – вывод матрицы на консоль. Принимает ссылку на стандартный поток вывода и ссылку на объект класса *TMatrix*, возвращает ссылку на стандартный поток вывода.

**Элементы класса, объявленные со спецификатором public:**

*TMatrix(int s = 10)* – конструктор с параметром.

*TMatrix(const TMatrix &MT)* – конструктор копирования.

*TMatrix(const TVector<TVector<T> > &MT)* – конструктор преобразования типа.

*virtual ~TMatrix<T>()* – деструктор.

**Перегруженные операторы:**

*bool operator==(const TMatrix &MT) –* оператор проверки на равенство.

*bool operator!=(const TMatrix &MT)* – оператор проверки на неравенство.

*TMatrix& operator=(const TMatrix &MT) –* операторприсваивания.

*TMatrix operator+(const TMatrix &MT) –* оператор сложения.

*TMatrix operator-(const TMatrix &MT)* – оператор вычитания.

*TMatrix operator\*(const TMatrix &MT)* – оператор умножения.

## Описание алгоритмов

**Умножение 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.

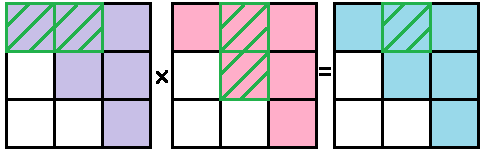


Рисунок 2 Умножение матриц.

**Умножение векторов.**

Проверяем размерность векторов. Если они не совпадают, выбрасываем исключение. Результатом умножения является сумма произведений элементов векторов с одинаковыми номерами.

# Заключение

В данной лабораторной работе была выполнена реализация структуры данных для верхнетреугольных матриц. А именно были разработаны библиотеки, реализующие классы вектора и матрицы, показан пример использования, а также написаны наборы тестов с использованием Google C++ Testing Framework.

# Литература

* Книги:

1. A.O. Грудзинский. Методы программирования, Издательство Нижегородского госуниверситета, 2006.
2. Васильев А.Н. Самоучитель С++ с примерами и задачами. -СПб.: Наука и Техника, 2016. -480с.

* Internet – ресурсы:

1. 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.
2. Википедия: свободная электронная энциклопедия: на русском языке [https://ru.wikipedia.org/wiki/Матрица\_(математика)]