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

Полетуева Анастасия Николаевна

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

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

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc533083407)

[2. Постановка задачи 4](#_Toc533083408)

[3. Руководство пользователя 5](#_Toc533083409)

[4. Руководство программиста 6](#_Toc533083410)

[4.1. Описание структуры программы 6](#_Toc533083411)

[4.2. Описание структур данных 6](#_Toc533083412)

[4.3. Описание алгоритмов 6](#_Toc533083413)

[5. Заключение 8](#_Toc533083414)

[6. Литература 9](#_Toc533083415)

# Введение

**Ма́трица** — 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.

Матрицы являются одним из самых удобных способов представления информации. Они применяются в математике для компактной записи систем линейных алгебраических или дифференциальных уравнений. Также они нашли широкое применение в программировании, так как матричный способ представления графической информации является наиболее простым и удобным, если не единственным наиболее правильным.

Помимо прямоугольных матриц, для которых 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 с учетом этого они могут оказать более эффективными.

В данной работе будут рассмотрены верхнетреугольные матрицы, которые представляют собой квадратные матрицы, элементы которой ниже главной диагонали равны нулю.

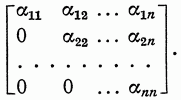


Рисунок 1. Верхнетреугольная матрица

# Постановка задачи

В рамках лабораторной работы ставится задача разработки верхнеугольной матрицы (класс TMatrix). Построение данной структуры данных рационально производить на основе вектора (TVector). Для работы с этой структурой данных будут реализованы операции:

* сложение и вычитание двух матриц
* умножение двух матриц
* умножение матрицы на скаляр

Выполнение работы предполагает решение следующих задач:

1. Разработка и реализация классов – TVector, TMatrix.
2. Разработка программы, использующую класс TMatrix.
3. Создание класса Expectation для обработки исключений, которые могут возникнуть при выполнении различных операций.
4. Реализация набора тестов, написанных с использованием GoogleTesting Framework.

# Руководство пользователя

При запуске программы на консоль будут выведены 2 автоматически созданные матрицы: a и b, а также матрица c, являющаяся результатом сложения этих двух матриц.

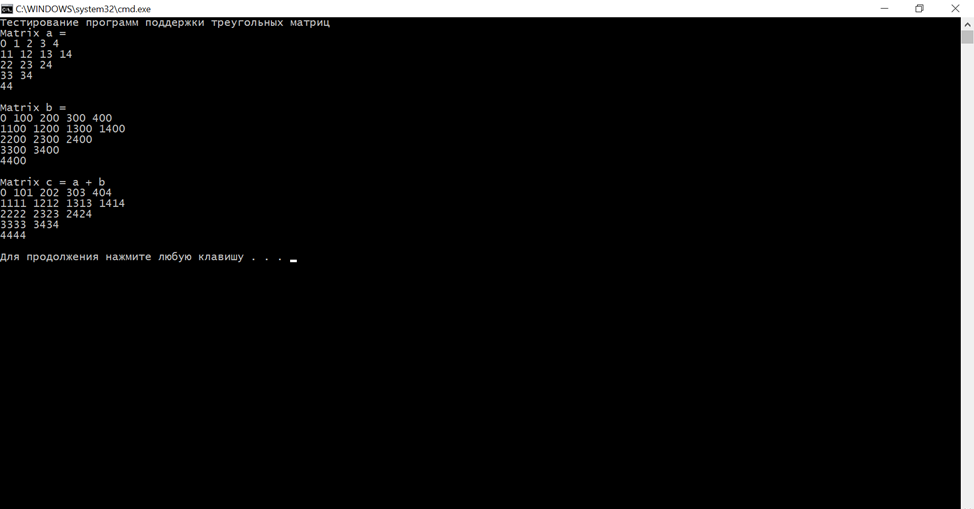


Рисунок 2. Вывод на консоль

# Руководство программиста

## Описание структуры программы

Программа состоит из следующих модулей:

* Модуль Vector. Содержит пример использования матрицы. Реализация в файле main.cpp.
* Модуль VectorLib – статическая библиотека. Содержит файл Vector.h, в котором определен интерфейс и реализация шаблонного класса вектор TVector, и файл Matrix.h, в котором определен интерфейс и реализация шаблонного класса матрица TMatrix.
* Модуль VectorTest. Содержат для каждого из классов (*TVector* и *TMatrix*) наборы тестов, реализованные в файлах *vector\_test.cpp* и *matrix\_test.cpp* с помощью использования Google C++ Testing Framework.

## Описание структур данных

#### 4.2.1 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 \*pVector* – указатель на область памяти для хранения вектора.

*Int StartIndex* – индекс первого элемента вектора.

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

*TVector(int s, int si = 0)* – конструктор по умолчанию.

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

*~TVector()* – деструктор.

**Методы для работы с классом TVector:**

*int GetSize() c* – 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.

**4.2.2 Класс TMatrix – верхнетреугольная матрица.**

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

**Перегрузка оператора умножения.**

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.

# Заключение

В ходе выполнения лабораторной была разработана программа, реализующая структуру хранения данных – верхнеугольная матрица. Реализованы методы, описанные в «Описание структур данных». Программная реализация была продемонстрирована на примере. Разработаны тесты на основе Google C++ Testing Framework.

# Литература

1. Гергель В.П. Методические материалы по курсу «Методы программирования» (часть 1), 2015, -96с.
2. Википедия: свободная электронная энциклопедия: на русском языке [https://ru.wikipedia.org/wiki/Матрица\_(математика)]