МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Национальный исследовательский**

**Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**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

Колесова Кристина Юрьевна

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

ассистент каф. МОСТ ИИТММ

Лебедев Илья Геннадьевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Подпись

Нижний Новгород

2018.

Оглавление

[1. Введение 3](#_Toc532739452)

[2. Постановка задачи 4](#_Toc532739453)

[3. Руководство пользователя 5](#_Toc532739454)

[4. Руководство программиста 6](#_Toc532739455)

[4.1. Описание структуры программы 6](#_Toc532739456)

[4.2. Описание структур данных 7](#_Toc532739457)

[4.3. Описание алгоритмов 9](#_Toc532739458)

[5. Заключение 12](#_Toc532739459)

[6. Список литературы 13](#_Toc532739460)

# 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.

Кроме 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, что сделает их более эффективными.

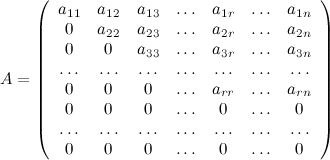
Верхнетреугольные матрицы, которые представляют собой квадратные матрицы, элементы которой ниже главной диагонали равны нулю (см. рис. 1).

Рисунок 1. Верхнетреугольная матрица

Цель данной лабораторной работы – разработка структуры хранения матриц специального вида.

# Постановка задачи

Выполнение работы предполагает решение следующих задач:

1. Реализация класса для работы с вектором TVector
2. Реализация класса матриц TMatrix
3. Создание класса MyException для обработки исключений, которые

могут возникнуть при выполнении различных операций

1. Реализация тестов на базе Google Test

# Руководство пользователя

Примеры использования классов TVector и TMatrix аналогичны, рассмотрим их работу на примере TVector. Пример реализован с помощью консольного приложения, которое демонстрирует функции.

При запуске программы на консоль будут выводиться две матрицы. Далее будут выводиться результаты операций с этими матрицами: сложение, вычитание, умножение.

После этого пользователю предлагается ввести матрицу с консоли. После ввода пользователем матрицы она будет выведена на экран.

На этом работа программы заканчивается.

# Руководство программиста

## Описание структуры программы

В программе содержатся следующие модули:

* Модули Vector и Matrix.   
  Содержат примеры реализаций программ с использованием классов вектор и матрица. 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*

Класс *TVector* является шаблонным. В классе 2 поля int Size – размер вектора и T\* Mas – указатель на область памяти для хранения вектора данных типа T. Поля класса объявлены со спецификатором private.

1. 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.

1. 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.

1. 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.

1. 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.

* Класс TMatrix

Класс TMatrix является шаблонным и является наследником TVector со спецификатором public.

1. 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.

1. 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) – оператор присваивания.

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

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

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

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

1. 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.

## Описание алгоритмов

* Перегрузка операторов сложения и вычитания

Для операций сложения и вычитания требуется сложить/вычесть соответствующие элементы каждой из матриц. Поскольку у нас матрица представлена как *TVector <TVector>*, то когда мы будем складывать элементы строки одной матрицы с элементами строки другой матрицы, мы будем выполнять сложение соответствующих векторов. И так далее для каждой строки. Следовательно, при выполнении аналогичных операторов для *TVector* *<TVector>* будет получен верный результат и для *TMatrix.*

* Перегрузка оператора умножения

Произведением матриц А и B является такая матрица C = AB, у которой элемент c\_ij, 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.

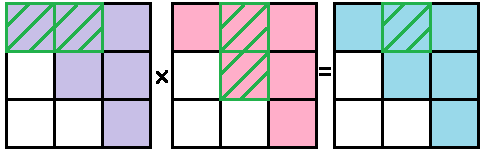
Реализация перегрузки оператора умножения представлена, с помощью трех вложенных циклов: по строкам первой матрицы, по столбцам второй матрицы и по элементам текущего столбца второй матрицы.

Рисунок 2. Умножение матриц

# Эксперименты

Эксперименты были проведены на ПК со следующими характеристиками:

1. Операционная система: Windows 10 Домашняя
2. Процессор AMD A8-7410 APU 2.20GHz
3. Версия Visual Studio: 2017

Результаты экспериментов занесены в таблицу 1.

|  |  |  |
| --- | --- | --- |
| Кол-во элементов | Время работы оператора сложения (в млс) | Время работы оператора умножения (в млс) |
| 1 000 | 38 | 9 936 |
| 5 000 | 851 | 1 264 723 |
| 10 000 | 3742 | 10 610 219 |

Таблица 1. Время выполнения сложения и умножения матриц

По данной таблице построены графики 1 и 2.

График 1. Зависимость времени работы алгоритма сложения двух матриц от числа элементов

График 2. Зависимость времени работы алгоритма умножения двух матриц от числа элементов

Проанализировав таблицу 1 и графики 1 и 2 можем сделать заключения о сложности двух рассмотренных алгоритмов:

1. Сложность алгоритма сложения составляет
2. Сложность алгоритма умножения составляет

# Заключение

В результате выполнения лабораторной работы была разработана библиотека, позволяющая работать с векторами, матрицами и матрицами специального – верхнетреугольного.

Были реализованы класс для работы с векторами (TVector) и класс для работы с матрицами (Tmatrix), в том числе и методы работы с этими классами.

Были реализованы тесты для проверки работоспособности вышеперечисленных классов на базе GoogleTest.

# Список литературы

1. Википедия. Статья «Матрица (математика)»: [https://ru.wikipedia.org/wiki/Битовое\_поле].
2. Васильев А.Н. Самоучитель С++ с примерами и задачами. -СПб.: Наука и Техника, 2016. -480с.
3. Гергель В.П. Методические материалы по курсу «Методы программирования 2», Нижний Новгород, 2015.