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**

**ЛАБОРАТОРНАЯ РАБОТА №2**

**«Структура хранения для матриц специального вида»**

**Выполнил:** студент группы 381706-2

Макарихин Семён Андреевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Руководитель:**

Ассистент кафедры МОСТ

Лебедев Илья Геннадьевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

Нижний Новгород

2019

Содержание

[1.Введение 3](#_Toc532897542)

[2. Цели и задачи 4](#_Toc532897543)

[3. Руководство пользователя 5](#_Toc532897545)

[4. Руководство программиста 7](#_Toc532897546)

[4.1. Описание структуры программы 7](#_Toc532897547)

[4.2. Описание функций и процедур, их алгоритмов 7](#_Toc532897548)

[5. Эксперименты 1](#_Toc532897549)2

[6. Заключение 1](#_Toc532897550)3

[7. Литература 1](#_Toc532897551)4

[8. Приложение 1](#_Toc532897552)5

# Введение

**Ма́трица** — [математический объект](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82), записываемый в виде прямоугольной таблицы элементов [кольца](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D1%8C%D1%86%D0%BE_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) или [поля](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B5_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) (например, [целых](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BB%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE), [действительных](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%B9%D1%81%D1%82%D0%B2%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE) или [комплексных](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%BB%D0%B5%D0%BA%D1%81%D0%BD%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE) чисел), которая представляет собой совокупность [строк](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D1%82%D1%80%D0%BE%D0%BA%D0%B0_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86%D1%8B&action=edit&redlink=1) и [столбцов](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D1%82%D0%BE%D0%BB%D0%B1%D0%B5%D1%86_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86%D1%8B&action=edit&redlink=1), на пересечении которых находятся её элементы. Количество строк и столбцов задает размер матрицы. Хотя исторически рассматривались, например, треугольные матрицы[[1]](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)#cite_note-1), в настоящее время говорят исключительно о матрицах прямоугольной формы, так как они являются наиболее удобными и общими.

Матрицы широко применяются в математике для компактной записи систем линейных [алгебраических](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D1%83%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5) или [дифференциальных](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%84%D1%84%D0%B5%D1%80%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%83%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F) уравнений. В этом случае количество строк матрицы соответствует числу уравнений, а количество столбцов — количеству неизвестных. В результате решение [систем линейных уравнений](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BB%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D1%8B%D1%85_%D0%B0%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D1%85_%D1%83%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B9) сводится к операциям над матрицами.

В математике рассматривается множество различных типов и видов **матриц**. Таковы, например, [единичная](https://ru.wikipedia.org/wiki/%D0%95%D0%B4%D0%B8%D0%BD%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86%D0%B0), [симметричная](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86%D0%B0), [кососимметричная](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%81%D0%BE%D1%81%D0%B8%D0%BC%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86%D0%B0), [верхнетреугольная](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D1%80%D1%85%D0%BD%D0%B5%D1%82%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) (нижнетреугольная) и т. п. матрицы.

Особое значение в теории матриц занимают всевозможные [нормальные формы](https://ru.wikipedia.org/w/index.php?title=%D0%9D%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D0%B0_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86%D1%8B&action=edit&redlink=1), то есть канонический вид, к которому можно привести матрицу заменой координат. Наиболее важной (в теоретическом значении) и проработанной является теория [жордановых нормальных форм](https://ru.wikipedia.org/wiki/%D0%96%D0%BE%D1%80%D0%B4%D0%B0%D0%BD%D0%BE%D0%B2%D0%B0_%D0%BD%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D0%B0). На практике, однако, используются такие нормальные формы, которые обладают дополнительными свойствами, например, устойчивостью.

# 2. Цели и 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;
* копирование;
* сравнение.

В процессе выполнения лабораторной работы требуется использовать систему контроля версий [Git](https://git-scm.com/book/ru/v2) и фрэймворк для разработки автоматических тестов [Google Test](https://github.com/google/googletest).

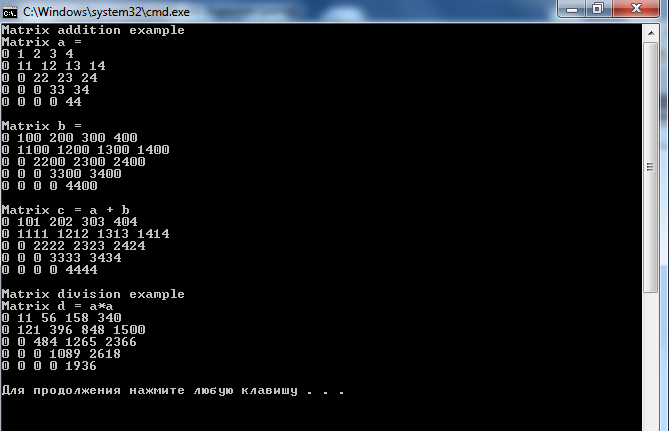
Выполнение работы предполагает решение следующих задач:

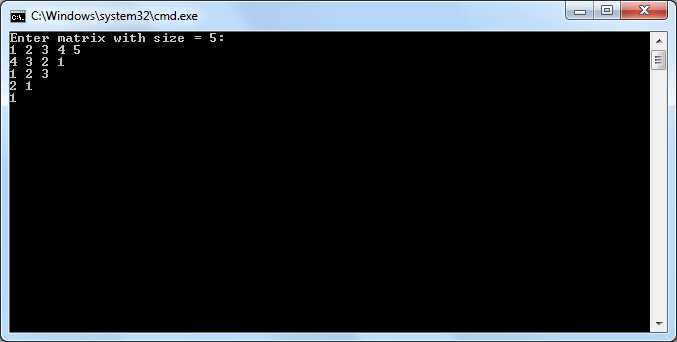
1. Реализация методов шаблонного класса TVector согласно заданному интерфейсу.
2. Реализация методов шаблонного класса TMatrix согласно заданному интерфейсу.
3. Обеспечение работоспособности тестов и примера использования.
4. Реализация заготовок тестов, покрывающих все методы классов TVector и TMatrix.
5. Модификация примера использования в тестовое приложение, позволяющее задавать матрицы и осуществлять основные операции над ними.

# 3. Руководство пользователя

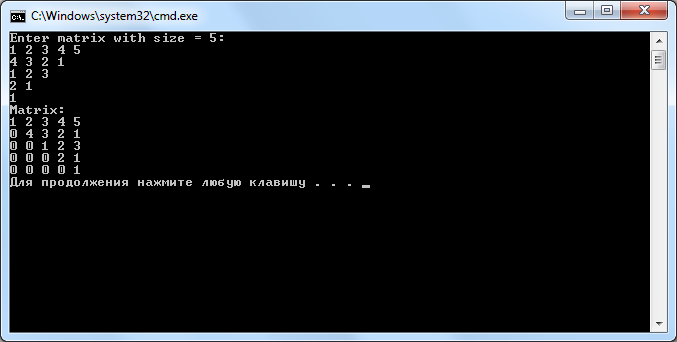
-Запускаем программу из файла matrix.cpp.

-Программа выведет пример двух верхнетреугольных матриц, результат их сложения, а потом пример перемножения двух матриц.



В появившемся окне консоли пользователю необходимо ввести матрицу.

Программа выведет заданную матрицу в следующем виде:



# 4. Руководство программиста

Разработка системы вычисления проводились в среде “Microsoft Visual Studio 2017”.

Для реализации алгоритмов будет использовано 2 шаблонных класса:

* Класс «Вектор» (TVector), реализованный с использованием массива
* Класс «Матрица» (TMatrix), реализованный с использованием класса вектора.

## 4.1. Описание структуры программы

Модульная структура программы:

1. utvector.h, utvector.cpp – модуль с классом TVector, в котором определен интерфейс шаблонного класса и реализация его методов.
2. utmatrix.h, utmatrix.cpp – модуль с классом TMatrix, использующим класс TVector, в котором определен интерфейс шаблонного класса и реализация его методов.

* sample\_matrix.cpp , sample\_input\_output.cpp, sample\_performance\_check.cpp– модуль программы тестирования, с которым работает пользователь, в котором проводятся эксперименты.

1. test\_main.cpp, test\_tvector.cpp, test\_tmatrix.cpp – модуль с функциями тестирования для созданных классов. Содержат 28 тестов для класса TVector и 25 теста для класса TMatrix.

## 4.2. Описание функций и процедур, их алгоритмов

Рассмотрим реализацию методов шаблонного класса TVector:

template <class VectorType>

class TVector

1. TVector(int s = 1, int si = 0) - конструктор класса, принимающий длину вектора и стартовый индекс элемента. По умолчанию создается вектор длины 1 со стартовым индексом элемента 0.

2. TVector(const TVector &v) - конструктор копирования.

Принимает ссылку на объект класса TVector.

3. ~TVector() - деструктор. Освобождает выделенную под вектор память.

4. int GetSize() - возвращает длину вектора.

5. int GetStartIndex() - возвращает стартовый индекс элемента.

6. VectorType& operator[](int pos) – перегрузка оператора индексации.

Возвращает элемент, стоящий на заданной позиции.

7. bool operator == (const TVector &v) const – перегрузка оператора сравнения.

Выполняется проверка векторов на равенство. Возвращает true, если равенство выполняется, false в противном случае.

8. bool operator!=(const TVector &v) const - перегрузка оператора сравнения.

Выполняется проверка векторов на неравенство. Возвращает true, если неравенство выполняется, false в противном случае.

9. TVector& operator=(const TVector &v) – перегрузка оператора присваивания одного вектора другому.

Присваивает полям первого объекта класса поля второго объекта класса.

10. TVector operator+(const VectorType &val) – перегрузка оператора сложения элемента вектора с числом.

Возвращает вектор, равный поэлементного сложения исходного вектора с числом.

11. TVector operator-(const VectorType &val) - перегрузка оператора вычитания числа из элемента вектора.

Возвращает вектор, равный поэлементной разности исходного вектора с числом.

12. TVector operator\*(const VectorType &val) – умножение вектора на число.

Возвращает вектор, равный поэлементному произведению исходного вектора на число.

13. TVector operator+(const TVector &v) – перегрузка оператора сложения векторов.

Если длины векторов и стартовые индексы элементов совпадают, то к первому вектору поэлементного прибавляется второй.

14. TVector operator-(const TVector &v) - перегрузка оператора вычитания векторов.

Если длины векторов и стартовые индексы элементов совпадают, то из одного вектора поэлементного вычитается другой.

15. VectorType operator\*(const TVector &v) – перегрузка оператора скалярного произведения.

Если длины векторов и стартовые индексы элементов совпадают, то создается временная переменная, в которую записывается сумма произведений элементов соответствующих векторов.

Дружественные функции класса TVector:

friend istream& operator>>(istream &in, TVector &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 &out, const TVector &v) - вывод вектора на консоль.

Принимает ссылку на стандартный поток вывода и ссылку на объект класса TVector, возвращает ссылку на стандартный поток вывода.

Рассмотрим реализацию методов шаблонного класса TMatrix:

template <typename MatrixType>

class TMatrix : public TVector<TVector<MatrixType>>

1. TMatrix(int s = 10) - конструктор класса, принимающий размер матрицы (количество векторов).

По умолчанию создается матрица 10х10.

2. TMatrix(const TMatrix &mt) - конструктор копирования.

Принимает ссылку на объект класса TMatrix.

3. TMatrix(const TVector<TVector<MatrixType> > &mt) - конструктор преобразования типа.

4. bool operator==(const TMatrix &mt) const– перегрузка оператора сравнения.

Вызывается аналогичный оператор сравнения для TVector и возвращается результат его работы.

5. bool operator!=(const TMatrix &mt) const – перегрузка оператора сравнения.

Вызывается аналогичный оператор сравнения для TVector и возвращается результат его работы.

6. TMatrix& operator= (const TMatrix &mt) – перегрузка оператора присваивания.

Вызывается оператор присваивания для TVector. Возвращается ссылка на объект класса TMatrix.

7. TMatrix operator+ (const TMatrix &mt) – перегрузка оператора сложения двух матриц.

Вызывается оператор сложения для TVector и возвращается результат его работы.

8. TMatrix operator- (const TMatrix &mt) - перегрузка оператора вычитания одной матрицы из другой.

Вызывается оператор разности для TVector и возвращается результат его работы.

9. TMatrix inverse\_matrix() const – метод класса, который возвращает обратную матрицу.

10. TMatrix operator\* (const TMatrix &mt) – перегрузка оператора умножения для двух матриц.

Реализация перегрузки оператора умножения представлена, с помощью трех вложенных циклов:

11. TMatrix operator/ (const TMatrix &mt) – перегрузка оператора деления для двух матриц.

Использует метод inverse\_matrix() для приходящей матрицы, получая обратную ей. Перемножает две матрицы.

Дружественные функции класса TMatrix:

friend istream& operator>>(istream &in, TMatrix &mt) - ввод матрицы через консоль. Принимает ссылку на стандартный поток ввода и ссылку на объект класса TMatrix, возвращает ссылку на стандартный поток ввода.

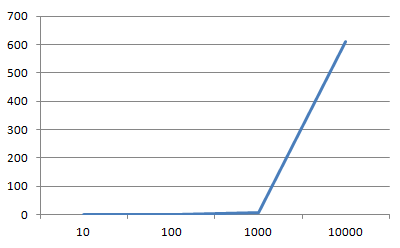
friend ostream & operator<<( ostream &out, const TMatrix &mt) - вывод матрицы на консоль. Принимает ссылку на стандартный поток вывода и ссылку на объект класса TMatrix, возвращает ссылку на стандартный поток вывода.

# 5. Эксперименты

В качестве примера рассмотрим перегрузку оператора сложения для класса матрицы (TMatrix).

Теоретическая сложность выполнения алгоритма O(n2).

Мы провели измерение присваивая размер 10, 100, …, 1000000 матрице. Ниже вы можете увидеть график зависимости времени выполнения операции сложения от размера матрицы. По приведенным данным можно сделать вывод, что практическая сложность выполнения алгоритма равна теоретической.



По горизонтали – размер матрицы.

По вертикали - время выполнения программы.

# 6. Заключение

В результате лабораторной работы была разработана структура данных для хранения вектора и верхнетреугольной матрицы, а также освоены такие инструменты разработки программного обеспечения, как система контроля версий [Git](https://git-scm.com/book/ru/v2).

# 7. Литература

1. Лабораторный практикум: Учебно-методическое пособие / Мееров И.Б. [и др.] – Нижний Новгород: Нижегородский госуниверситет , 2017. – 105с.
2. <https://ru.wikipedia.org/wiki/Треугольная_матрица>
3. <http://www.simumath.net/library/book.html?code=Matr_Alg_triangular_matrices>
4. <http://portal.tpu.ru/SHARED/k/KONVAL/Sites/Russian_sites/1/10.htm>

# 8. Приложение