

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**  
**«Верхнетреугольные матрицы»**

**Выполнил:** студент группы 381706-1  
Митягина Дарья Сергеевна

\_\_\_\_\_ Подпись

**Научный руководитель:**

ассистент каф. МОСТ ИИТММ

\_\_\_\_\_ Лебедев И.Г

Нижний Новгород  
2018.

## Оглавление

1. Введение.....	2
2. Постановка задачи.....	3
3. Руководство пользователя.....	4
4. Руководство программиста .....	6
4.1. Описание структуры программы.....	6
4.2. Описание структур данных .....	7
4.3. Описание алгоритмов .....	10
4.4 Оценка сложности некоторых алгоритмов.....	13
5. Заключение .....	14
6. Литература .....	15

## 1. Введение

Матрица – в математике – прямоугольная таблица каких-либо элементов  $a(i,j)$ , состоящая из  $m$  строк и  $n$  столбцов. Набор элементов матрицы  $(a(1,1), a(2,2), \dots, a(n,n))$  называется главной диагональю.

Треугольной называется матрица, в которой все элементы под главной (или побочной) диагональю равны нулю.

Определение матрицы возможно также через понятие Вектор.

Вектор – в математике – набор  $a(i)$ , состоящий из  $n$  элементов.

Тогда Матрица из  $m$  строк и  $n$  столбцов может быть определена как вектор из  $n$  элементов, где каждый элемент, в свою очередь, является вектором из  $m$  элементов.

Цель работы: разработка структуры хранения матриц как набора векторов разной длины.

## 2. Постановка задачи

В рамках лабораторной работы ставится задача создания программных средств, поддерживающих эффективное хранение матриц специального вида (верхнетреугольных) и выполнение основных операций над ними:

1. сложение/вычитание/умножение;
2. копирование;
3. сравнение.

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:

Перед пользователем появляется окно:

```
<<< An example of using the Matrix will be implemented here >>>

|| Matr1 ||
1  2  3  4  5
2  4  6  8
3  6  9
4  8
5

|| Matr2 ||
300 400 500 600 700
1400 1500 1600 1700
2500 2600 2700
3600 3700
4700

|| Matr3 ||
1  2  3  4  5
2  3  4  5
3  4  5
4  5
5

|| Matr4 ||
0  2  4  6  8
2  4  6  8
4  6  8
6  8
8

|| Matr5 ||
0  0  0  0
0  0  0  0
0  0  0  0
0  0  0  0

|| Result1 ||
301 402 503 604 705
1402 1504 1606 1708
```

Рис. 1 – Пример использования

Сначала показывается *пример использования матриц*.

$\text{Result1} = \text{Matr1} + \text{Matr2};$

$\text{Result2} = \text{Matr1} - \text{Matr2};$

$\text{Result3} = \text{Matr3} * \text{Matr3};$

*Проверка операторов ввода-вывода:*

```
<<< Input matrix >>>
Enter matrix size
5
Enter matrix elements
Enter 5 coordinates:
1 2 3 4 5
Enter 4 coordinates:
1 2 3 4
Enter 3 coordinates:
1 2 3
Enter 2 coordinates:
1 2
Enter 1 coordinates:
1
Displaying matrix elements
1      2      3      4      5
      1      2      3      4
          1      2      3
              1      2
                  1
```

Рис. 2 – Проверка работы операторов ввода/вывода

С пользователя запрашивается размер матрицы, после чего запрашиваются координаты векторов, составляющих строки матрицы. С каждой последующей строкой необходимо вводить число координат, меньшее на 1, чем в предыдущей строке (таким образом, мы получаем верхнетреугольную матрицу).

## 4. Руководство программиста

### 4.1. Описание структуры программы

Для реализации алгоритмов будет использовано 2 класса:

1. Класс TVector
2. Класс TMatrix, который будет использовать класс TVector

Лабораторная работа содержит следующие модули:

#### ❖ VectorLib

Этот модуль состоит из заголовочного файла Vectorlib.h, отвечающего за определение интерфейса класса TVector и содержащего реализацию методов, и файла Vectorlib.cpp.

Реализованы следующие функции: конструкторы, деструктор для класса TVector, перегружены теоретико-множественные операторы (такие как сравнение, присваивание и т.д.), операторы ввода/вывода.

#### ❖ MatrixLib

Этот модуль состоит из заголовочного файла Matrix.h, отвечающего за определение интерфейса класса TMatrix и содержащего реализацию методов, и файла Matrix.cpp. Реализованы следующие функции: конструкторы, деструктор для класса TMatrix, перегружены теоретико-множественные операторы (такие как сравнение, присваивание и т.д.), операторы ввода/вывода.

#### ❖ Тесты

В Matriz\_tets.cpp прописаны 15 тестов, в VecTests.cpp - 14 тестов. Назначение тестов: проверить каждый метод из классов TMatrix, TVector.

#### ❖ Пример использования

*Производится проверка операторов сложения, вычитания и умножения:*

*Проверка операторов ввода-вывода:*

Запрашивается размер матрицы, после чего запрашиваются координаты векторов, составляющих строки матрицы. С каждой последующей строкой необходимо вводить число

координат, меньшее на 1, чем в предыдущей строке (таким образом, мы получаем треугольную матрицу).

#### ❖ TException

Этот модуль содержит класс исключений.

### 4.2. Описание структур данных

В данной лабораторной программе матрицы представлены следующим образом:

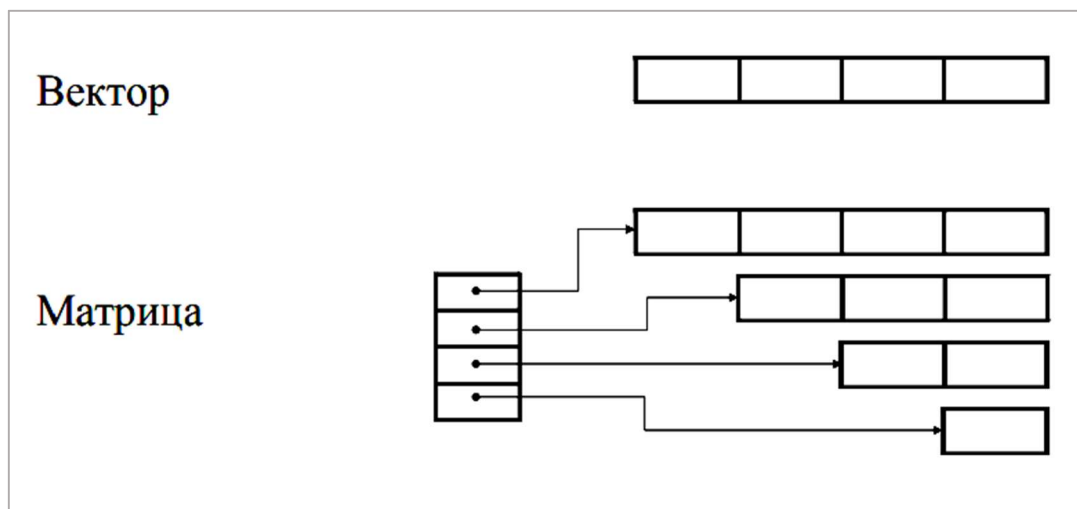


Рис. 3 – Представление матрицы в виде набора векторов

#### Класс TVector

Класс TVector является шаблонным классом.

Включены следующие два поля (protected - часть):

int dlna; - размер вектора

T \*Vector; - массив элементов вектора

Public – часть:

1) TVector<T>(int n = 0);

Конструктор инициализации, принимающий длину вектора.

2) TVector<T>(const TVector <T> &A);

Конструктор копирования, принимающий ссылку на объект класса TVector.



3) `TVector(T* s, int _dlina);`

Конструктор, принимающий массив типа T и размер массива.

4) `virtual ~TVector<T>();`

Деструктор.

5) `int getDlina() const;`

Возвращает размер вектора.

6) `T& operator[](int i);`

Метод, возвращающий элемент, расположенный на i-ой позиции.

7) `bool operator==(const TVector<T> &A) const;`

Сравнение. Метод, принимающий ссылку на объект класса TVector и проверяющий два вектора на равенство.

8) `bool operator!=(const TVector &v) const;`

Сравнение. Метод, принимающий ссылку на объект класса TVector и проверяющий два вектора на неравенство.

9) `TVector& operator=(const TVector<T> &A);`

Присваивание. Метод, принимающий ссылку на объект класса TVector и приравнивающий исходный вектор к пришедшему.

10) `TVector operator++();`

Инкремент. Возвращает вектор, увеличенный на 1.

11) `TVector operator++(int);`

12) `TVector operator--();`

Декремент. Возвращает вектор, уменьшенный на 1.

13) `TVector operator--(int);`

14) `TVector operator+() const;`

Возвращает вектор, равный исходному.

15) `TVector operator-() const;`

Возвращает вектор, полученный умножением исходного на (-1).

16) `TVector operator+(const TVector<T> &A);`

Принимает ссылку на объект класса TVector, возвращает вектор, полученный сложением исходного и пришедшего векторов.

17) TVector operator-(const TVector<T> &A);

Принимает ссылку на объект класса TVector, возвращает вектор, полученный вычитанием пришедшего вектора из исходного.

18) T operator\*(const TVector<T> &A);

Умножение двух векторов. Принимает ссылку на объект класса TVector, возвращает число, равное произведению векторов.

19) TVector operator\*(T A);

Умножение на число. Возвращает вектор, равный произведению исходного вектора и числа A.

20) template <class FriendT> friend TVector<FriendT> operator\*(FriendT a, const TVector<FriendT> &A);

*Ввод-вывод:*

21) template <class FriendT> friend istream& operator>>(istream &in, TVector<FriendT> &A);

22) template <class FriendT> friend ostream& operator<<(ostream &out, const TVector<FriendT> &AV);

### **Класс TMatrix**

Класс TMatrix является шаблонным классом.

Public – часть:

1) TMatrix(int n = 10);

Конструктор инициализации. Принимает размер матрицы (кол-во векторов).

2) TMatrix(const TMatrix &Matr);

Конструктор копирования, принимающий ссылку на объект класса TMatrix.

3) TMatrix(const TVector<TVector<T> > &Matr);

Конструктор копирования, принимающий ссылку на объект класса TVector<TVector<T>.

4) `bool operator==(const TMatrix &Matr);`

Сравнение. Метод, принимающий ссылку на объект класса TMatrix и проверяющий две матрицы на равенство.

5) `bool operator!=(const TMatrix &Matr) const;`

Сравнение. Метод, принимающий ссылку на объект класса TMatrix и проверяющий две матрицы на неравенство.

6) `TMatrix& operator= (TVector<TVector<T> > &Matr);`

Присваивание. Метод, принимающий ссылку на объект класса TVector<TVector<T> > и приравнивающий исходный вектор к пришедшему.

7) `TMatrix operator+ (const TMatrix &Matr);`

Принимает ссылку на объект класса TVector, возвращает вектор, полученный сложением исходного и пришедшего векторов.

8) `TMatrix operator- (const TMatrix &Matr);`

Принимает ссылку на объект класса TVector, возвращает вектор, полученный вычитанием пришедшего вектора из исходного.

9) `TMatrix operator*(TMatrix<T> &A);`

Принимает ссылку на объект класса TVector, возвращает вектор, полученный перемножением исходного и пришедшего векторов.

*Ввод/вывод:*

10) `template <class FriendT> friend istream& operator>>(istream &istr, TMatrix<FriendT> &Matr);`

11) `template <class FriendT> friend ostream & operator<<(ostream &ostr, const TMatrix<FriendT> &Matr);`

#### 4.3. Описание алгоритмов

В данном разделе не будут рассматриваться тривиальные методы.

В классе `TVector` все методы довольно простые, поэтому сразу перейдем к классу `TMatrix`.

1. Так как мы уже реализовали методы класса `TVector`, то методы `TMatrix` написать не составляет труда, ведь `TMatrix` – потомок `TVector`.

Именно поэтому большая часть функций работает благодаря вызову соответствующего метода из класса `TVector`.

При сравнении матриц, например, нам необходимо сравнить вектор векторов, это можно осуществить с помощью представления матрицы как набора векторов :  $\langle T \rangle$  в данном случае будет являться  $\langle TVector \rangle$ .

Для реализации сложения, вычитания достаточно вызвать аналогичные функции класса-родителя.

## 2. Умножение матриц.

Определение: Произведением матрицы  $A$  на матрицу  $B$  называется матрица  $C$  такая, что элемент матрицы  $C$ , стоящий в  $i$ -ой строке и  $j$ -ом столбце, т.е. элемент  $c_{ij}$ , равен сумме произведений элементов  $i$ -ой строки матрицы  $A$  на соответствующие элементы  $j$ -ого столбца матрицы  $B$ .

Рассмотрим схему (Рис. 4)

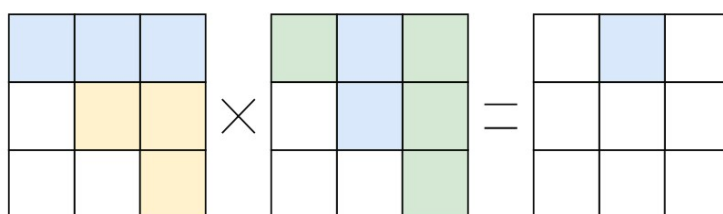


Рис. 4 – перемножение двух квадратных матриц

Для вычисления произведения двух матриц каждая строка первой почленно умножается на каждый столбец второй. Затем подсчитывается сумма таких произведений и записывается в соответствующую клетку результата.

Для перегрузки данного оператора использованы три цикла:

- 1) по строкам матрицы  $A$
- 2) по столбцам матрицы  $B$
- 3) по элементам матрицы  $C$

### 3. Деление матриц

В теории матриц нет понятия «деления матрицы», матрицы можно только умножать. Если нужно разделить матрицу на некоторое число  $k$ , то используется термин умножить матрицу на дробь  $\frac{1}{k}$ . А вместо «разделить матрицу  $A$  на матрицу  $B$ » говорят, что нужно умножить матрицу  $A$  на матрицу  $B^{-1}$ , где  $B^{-1}$  – обратная матрица к матрице  $B$ .

Таким образом, исходная задача сводится к поиску обратной матрицы и дальнейшему умножению на нее исходной матрицы.

Обратную матрицу можно найти, применив метод Гаусса. Суть этого метода состоит в следующем:

1) построить вспомогательную матрицу  $M$ , приписав к столбцам матрицы  $A$  справа столбцы единичной матрицы того же порядка, что и матрица  $A$ :

$$M = \left( \begin{array}{cccc|cccc} a_{11} & a_{12} & \dots & a_{1n} & 1 & 0 & \dots & 0 \\ a_{21} & a_{22} & \dots & a_{2n} & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} & 0 & 0 & \dots & 1 \end{array} \right)$$

Рис. 5 Первый шаг метода Гаусса

2) элементарными преобразованиями строк привести матрицу  $M$  к матрице, в левой части которой стоит единичная матрица:

$$N = \left( \begin{array}{cccc|cccc} 1 & 0 & \dots & 0 & b_{11} & b_{12} & \dots & b_{1n} \\ 0 & 1 & \dots & 0 & b_{21} & b_{22} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & b_{n1} & b_{n2} & \dots & b_{nn} \end{array} \right)$$

Рис. 6 Второй шаг метода Гаусса

3) матрица, стоящая в правой части полученной матрицы  $N$  и будет обратной матрицей

$$A^{-1} = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots \\ b_{n1} & b_{n2} & \dots & b_{nn} \end{pmatrix}$$

Рис. 7 Третий шаг метода Гаусса

В нашем случае матрица получится треугольной.

#### 4.4 Оценка сложности некоторых алгоритмов

В этом разделе уделим внимание лишь двум алгоритмам: сложения и умножение двух матриц  $N \times N$  (т.к. деление и вычитание являются частными случаями применения этих алгоритмов).

Эксперимент:

Размер матрицы	Т сложения	Т умножения
10	0	0.001
100	0.007	0.026
600	0.068	5.332
1000	0.237	25.412
1200	0.268	46.77
1500	0.505	91.267
200	1.073	281.501

Таблица 1. Результаты эксперимента.

1. Сложение матриц  
Сложность алгоритма равна  $O(n^2)$ .
2. Умножение матриц  
Сложность алгоритма равна  $O(n^3)$ .

## 5. Заключение

В данной лабораторной работе была выполнена задача разработки программы, поддерживающей эффективное хранение треугольных матриц и выполнение основных операций над ними.

Результат: завершена реализация классов TVector и TMatrix, разработана структура хранения матриц как набора векторов разной длины, освоена техника составления *тестов* путем самостоятельного составления нескольких из них на базе GT.

## 6. Литература

### *Интернет-ресурсы:*

1. Глухих Михаил Игоревич, к.т.н., доц. Лекция 5. Отношения между объектами, наследование [[http://kspt.icc.spbstu.ru/media/files/2011/course/cpp/slides2/05\\_Inheritance\\_v1.pdf](http://kspt.icc.spbstu.ru/media/files/2011/course/cpp/slides2/05_Inheritance_v1.pdf)]

2. Г.И. Радченко, Е.А. Захаров «Объектно-ориентированное программирование», Челябинск Издательский центр ЮУрГУ 2013 [<https://glebradchenko.susu.ru/courses/bachelor/oop/OOP-PrePrint.pdf>]

### *Книги:*

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