

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

Выполнил:
студент группы 361706-1
Резанцев Сергей Алексеевич
_____ Подпись

Научный руководитель:
ассистент каф. МОСТ ИИТММ
_____ Лебедев И. Г.

Нижний Новгород
2018 г.

Содержание	
Введение	3
Постановка задачи	4
Руководство пользователя	5
Руководство программиста	6
Описание структуры программы	6
Описание структур данных	6
Описание алгоритмов	8
Оценка сложности некоторых алгоритмов	8
Заключение	10
Литература	11

1. Введение

Целью данной лабораторной работы является рассмотрение треугольных матриц и их представления в виде вектора, состоящего из векторов, и для реализации этой программы понадобится создать классы с шаблонами, различными функциями и перегрузить арифметические операторы.

Треугольная матрица — в линейной алгебре квадратная матрица, у которой все элементы, стоящие ниже (или выше) главной диагонали, равны нулю. Служат для более компактного хранения данных.

2. Постановка задачи

Реализовать классы TMatrix и TVector для работы с матрицами. Поля классов должны быть закрыты. TMatrix наследуется от TVector.

В каждом классе обязательно должны присутствовать методы:

- ☐ инициализации значений полей объектов класса;
- ☐ доступа к полям класса на чтение и запись;
- ☐ ввода значений объектов с клавиатуры и вывода на консоль output;

Должны быть перегружены операции:

- ☐ объединения и пересечения и отрицания;
- ☐ арифметические – сложение, вычитание, умножение и деление;
- ☐ оператор присваивания, сравнения;
- ☐ операции ввода/вывода в поток (в классе эти функции объявлены как дружественные для доступа к закрытым полям класса);

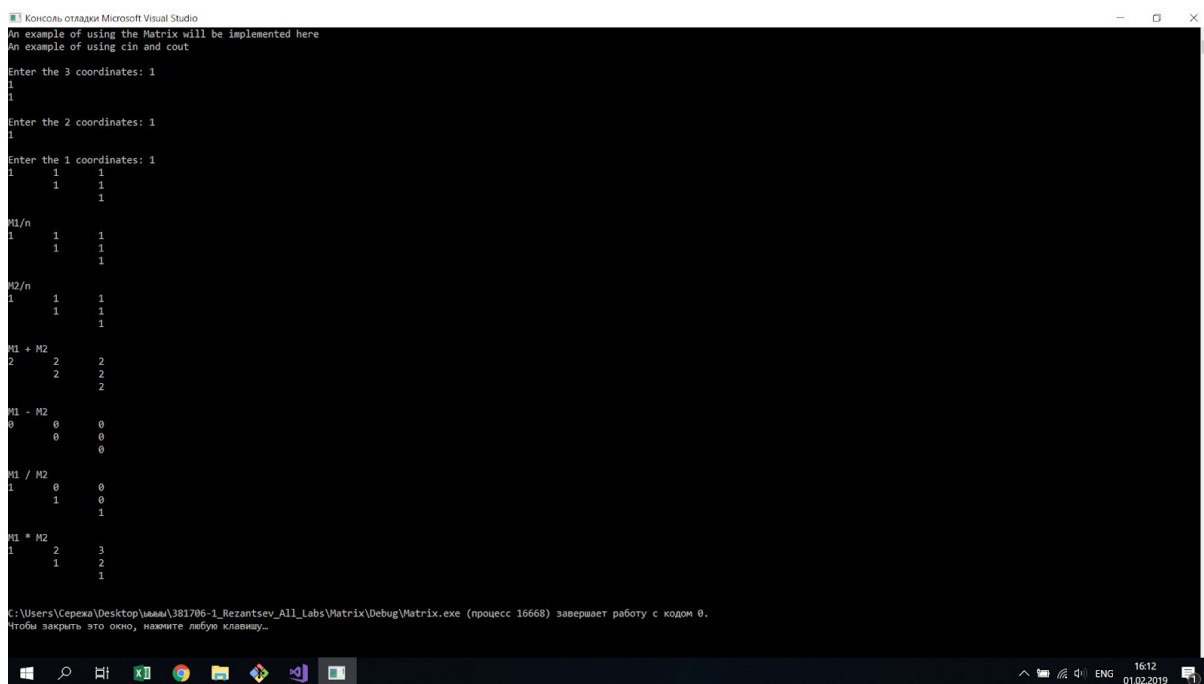
Должны быть реализованы конструкторы: по умолчанию, копирования и инициализатор.

Предоставить пример использования и обеспечить работоспособность тестов, покрывающих все методы классов TMatrix и TVector.

3. Руководство пользователя

Чтобы начать работу с программой запустите приложение Matrix.

На экране появится следующее:



```
Консоль отладки Microsoft Visual Studio
An example of using the Matrix will be implemented here
An example of using cin and cout
Enter the 3 coordinates: 1
1
Enter the 2 coordinates: 1
1
Enter the 1 coordinates: 1
1 1 1
1 1 1
1 1 1

M1/n
1 1 1
1 1 1
1 1 1

M2/n
1 1 1
1 1 1
1 1 1

M1 + M2
2 2 2
2 2 2
2 2 2

M1 - M2
0 0 0
0 0 0
0 0 0

M1 / M2
1 0 0
0 1 0
0 0 1

M1 * M2
1 2 3
2 1 2
1 1 1

C:\Users\Сергей\Desktop\Учебный\381706-1_Rezantsev_All_Labs\Matrix\Debug\Matrix.exe (процесс 16668) завершает работу с кодом 0.
Чтобы закрыть это окно, нажмите любую клавишу...
```

Результат программы, выведенный на консоль.

Затем программа завершится.

4. Руководство программиста

4.1. Описание структуры программы

Для реализации алгоритмов будут использованы классы TMatrix и TVector.

Лабораторная работа состоит из следующих модулей:

VectorLib

Библиотека, содержащая заголовочный файл TVector.h, в котором содержится класс TVector и реализация его методов, и файл TVector.cpp

MatrixLib

Библиотека, содержащая заголовочный файл Matrix.h, в котором содержится класс TMatrix - наследник TVector, и реализация его методов, и файл TMatrix.cpp

Matrix

Пример использования программы.

test

В файле test_matrix.cpp прописаны тесты, покрывающие каждый метод класса TMatrix. В файле test_vector.cpp прописаны тесты, покрывающие каждый метод класса TVector .

4.2. Описание структур данных

В программе описаны классы:

TVector

В нем 2 поля:

int size;

T *vector;

И реализованы следующие методы:

TVector<T>(int n = 0);

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

virtual ~TVector<T>();

```

int GetSize() const;

T& operator[](int i);

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

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

TVector operator++();

TVector operator++(int);

TVector operator--();

TVector operator--(int);

TVector operator+() const;

TVector operator-() const;

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

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

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

TVector operator*(T A);

```

TMatrix: Наследник класса TVector. Не имеет полей.

Реализованы следующие методы:

```

TMatrix(int n = 10);

TMatrix(const TMatrix &B);

TMatrix(const TVector<TVector<T>> &B);

bool operator==(const TMatrix &B) const;

bool operator!=(const TMatrix &B) const;

TMatrix& operator= (TVector<TVector<T>> &B);

TMatrix operator+ (const TMatrix &B);

TMatrix operator- (const TMatrix &B);

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

TMatrix operator/(TMatrix<T> &A);

```

TVector<T>& operator [] (int i);

4.3. Описание алгоритмов

В данном разделе не будут рассматриваться тривиальные методы и методы из класса TVector, так все они довольно простые. И большинство функций класса TMatrix работает с помощью вызова соответствующего метода из TVector.

1. Умножение матриц

Опр. Произведением матрицы A на матрицу B называется такая матрица C , что элемент матрицы c , стоящий в i -ой строке j -ого столбца, является произведением элементов i -ой строки на соответствующие элементы j -го столбца.

Для перегрузки данного оператора использовались три цикла:

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

2. 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 .

Находим обратную матрицу с помощью метода Гаусса (в нашем случае она получится треугольной) и умножаем матрицу A на B^{-1} .

4.4. Оценка сложности некоторых алгоритмов

Характеристики компьютера:

Intel Core i5

8 GB DDR3 L Memory

128 GB SSD + 1000 GB HDD

Размер матрицы	Т Сложения	Т Умножения
10	0	0.001
100	0.004	0.013
500	0.35	1.363

1000	2.897	10.851
------	-------	--------

Сложение: Сложность алгоритма $O(n^2)$

Умножение: Сложность алгоритма $O(n^3)$;

5. Заключение

В данном курсовом проекте при разработке программы были рассмотрены треугольные матрицы и их реализация с помощью векторов. И провели вычисления с матрицами. Классы и перегрузки, наследование оказались очень полезными и удобными и упростили работу с программой. Также была закреплена техника составления тестов на базе GoogleTest.

6. Литература

1. Васильев А.Н. Самоучитель С++ с примерами и задачами. -СПб.: Наука и Техника, 2016. -480с.
2. Т. А. Павловская С/С++ Программирование на языке высокого уровня. - СПб.:Питер, 2011. - 461 с.
- 3.Крапенко С. Н. и др. Методы объектно-ориентированного программирования. <http://e-learning.unn.ru/course/view.php?id=251>.
- 4.Страуструп. Б. Курс «Язык программирование С++ для профессионалов» <http://www.intuit.ru/studies/courses/98/98/info>
- 5.Гергель В.П. Методические материалы по курсу “Методы программирования 2”: [<http://www.itmm.unn.ru/files/2018/10/Primer-1.1.-Struktury-hraneniya-mnozhestva.pdf>], 2015.
6. <http://ru.solverbook.com/spravochnik/matricy/>