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 ЛАБОРАТОРНОЙ РАБОТЕ**

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

**Выполнил:** студент группы 381706-2

Мышкин Андрей Александрович

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Руководитель:**

ассистент кафедры МОСТ ИИТММ,

Лебедев Илья Геннадьевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

Нижний Новгород

2018

**Содержание**

1. Введение………………………………………………………………………...3

2. Постановка задачи……………………………………………………...………4

3. Руководство пользователя…………………………………………...………...6

4. Руководство программиста…………………………………………...………..7

4.1. Описание структур данных…..……………………………………….7

4.2. Описание алгоритмов………………………………………………….8

4.3. Описание структуры программы….……………………………..…...8

6. Заключение…………………………………………………………………….10

7. Литература……………….…………………………...…………………….…11

**Введение**

Матричные обозначения широко распространены в современной математике и её приложениях. Матрица – полезный аппарат для исследования многих задач теоретической и прикладной математики. Так, одной из важнейших является задача нахождения решения систем линейных алгебраических уравнений.

Следствием разнообразия областей применения матричного аппарата в современной науке является наличие в любом из больших математических программных комплексов (Mathcad, Mathematica, Derive, Mapple) подсистем, выполняющих операции над матрицами, а также существование специальных программных библиотек (ScalaPack, PlaPack), рассчитанных на обработку огромных (десятки и сотни тысяч строк) матриц, в том числе с использованием распределенных вычислений.

Помимо матриц общего вида, для которых наиболее естественной и 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 потому более эффективных. В данном проекте мы сможем рассмотреть один из видов матриц, треугольные.

**Постановка 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 Фреймворк для разработки автоматических тестов [Google Test](https://github.com/google/googletest).

Перед выполнением был получен данный проект-шаблон, содержащий следующее:

* Интерфейсы классов Вектор и Матрица (h-файл)
* Начальный набор готовых тестов для каждого из указанных классов.
* Набор заготовок тестов для каждого из указанных классов.
* Тестовый пример использования класса Матрица

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.

**Используемые инструменты**

Система контроля версий [Git](https://git-scm.com/book/ru/v2). Рекомендуется использовать один из следующих клиентов на выбор студента:

* + [Git](https://git-scm.com/downloads)
  + [GitHub Desktop](https://desktop.github.com/)

Фреймворк для написания автоматических тестов [Google Test](https://github.com/google/googletest). Не требует установки, идет вместе с проектом-шаблоном.

Среда разработки Microsoft Visual Studio (2008 или старше).

Опционально. Утилита [CMake](http://www.cmake.org/) для генерации проектов по сборке исходных кодов. Может быть использована для генерации решения для среды разработки, отличной от Microsoft Visual Studio 2008 или 2010, предоставленных в данном проекте-шаблоне.

**Руководство пользователя**

Для проверки работы созданного проекта будет проведено тестирование. Суть его в том, чтобы создать 2 случайно сгенерированные матрицы, и произвести над ними некие действия. В нашем случае будет выполнены такие операции, как сложение и умножение двух матриц. Для простоты тестирование проводится с целыми числами в небольшом диапазоне. Значения полей первой матрицы варьируются от 0 до 20, а второй от 0 до 30.

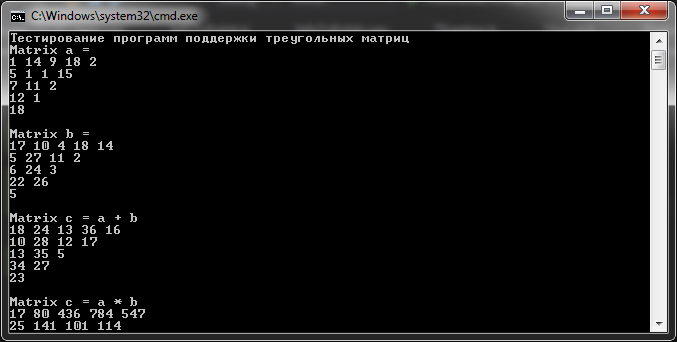


Рисунок 1. Вывод на консоль двух случайных матриц

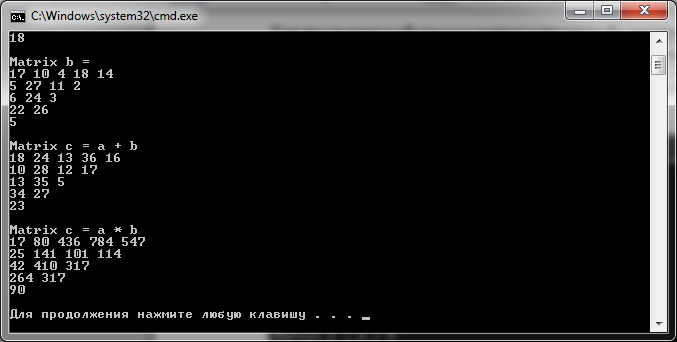


Рисунок 2. Вывод на консоль результатов сложения и умножения двух матриц

**Руководство программиста**

**Описание структур данных.**

**Класс TVector**

Поля

int size – размер вектора

int startIndex – индекс первого элемента вектора

ValType \* pVector – выделенная память под вектор

Методы

TVector() – конструктор класса

TVector(int s, int si = 0) – конструктор с параметрами

TVector(const TVector &v) – конструктор копирования

~TVector() – деструктор класса

int GetSize() – метод класса, возвращающий размер вектора

int GetStartIndex() – метод класса, возвращающий индекс первого элемента вектора

int operator==(const TVector &v) – метод класса, производящий сравнение двух векторов

TVector & operator= (const TVector &v) – метод класса, оператор присваивание вектора

TVector operator\* (const ValType &val) – метод класса, который проводит умножение вектора на скаляр

TVector operator+ (const ValType &val) – метод класса, который прибавляет к вектору скаляр

TVector operator- (const ValType &val) – метод класса, который вычитает у вектора скаляр

TVector operator+ (const TVector &v) – метод класса, который суммирует два вектора

TVector operator- (const TVector &v) – метод класса, который вычитает из данного вектора указанный в методе

ValType operator\* (const TVector &v) – метод класса, производящий скалярное произведение

**Класс TMatrix**

Данный класс является наследником класса TVector, поэтому имеет такие поля

Методы

TMatrix(int s = 10) – конструктор с параметром

TMatrix(const TMatrix &mt) – конструктор копирования

TMatrix(const TVector<TVector<ValType> > &mt) – конструктор преобразования типа

int operator==(const TMatrix &mt) – метод класса, оператор сравнения матриц

TMatrix & operator= (const TMatrix &mt) – метод класса. оператор присваивания матрицы

TMatrix operator+ (const TMatrix &mt) – метод класса, который суммирует две матрицы

TMatrix operator- (const TMatrix &mt) – метод класса, который вычитает из данной матрицы указанную в методе матрицу

TMatrix operator\* (const TMatrix &mt) – метод класса, который проводит умножение двух матриц

**Описание алгоритмов.**

Суммирование двух матриц.

Сначала проверяется размер главного вектора матриц. В случае если их векторы не равны, выходим с исключением. В случае если векторы равны, тогда создаем временную матрицу, куда положим сумму предыдущих матриц, с таким же главным вектором. Производим сложение матриц по i векторам в цикле от 0 до размера главного вектора матриц. Полученные результаты заносим во временную матрицу, изначально пустую. После завершения цикла возвращаем полученную матрицу.

**Описание структуры программы.**

vector – модуль для тестирования работы структуры данных типа вектор. Главным, файлом которого является main.cpp

vectorlib – модуль для хранения реализации класса вектор. В нем содержится заголовочный файл – TVector.h

vectortest – модуль, содержащий тесты данного класса для прохождения их с помощью использования Google C++ Testing Framework. В файле vector\_tests.cpp содержатся написанные тесты, а в test\_main.cpp код, запускающий тестирование

matrixlib – модуль для хранения реализации класса матрица. В нем содержится заголовочный файл – TMatrix.h

matrixtest – модуль, содержащий тесты данного класса для прохождения их с помощью использования Google C++ Testing Framework. В файле matrix\_tests.cpp содержатся написанные тесты, а в test\_main.cpp код, запускающий тестирование

**Заключение**

В ходе проведение данной лабораторной работы была создана и протестирована такая структура хранения данных как треугольная матрица. В классе список на массиве были реализованы все необходимые методы для комфортной работы с данной структурой. Усвоены все тонкости, с которыми пришлось столкнуться при выполнении данной лабораторной работы. Закреплены знания теории.

Также были освоены инструменты разработки программного обеспечения, как система контроля версий Git и фрэймворк для разработки автоматических тестов Google Test.

**Литература**

1. Гергель В.П. Методические материалы по курсу «Методы программирования 2», Нижний Новгород, 2015
2. Треугольные матрицы - <https://life-prog.ru/2_87934_treugolnie-matritsi.html>
3. Треугольные матрицы и характеристическое уравнение -<https://helpiks.org/1-127405.html>