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

Соболева Юлия Алексеевна

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

ассистент каф. МОСТ ИИТММ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лебедев И.Г

Нижний Новгород

2018 г.

Содержание

[1. Введение 3](#_Toc531274519)

[2. Постановка задачи 4](#_Toc531274520)

[3. Руководство пользователя 5](#_Toc531274521)

[4. Руководство программиста 6](#_Toc531274522)

[4.1. Описание структуры программы 6](#_Toc531274523)

[4.2. Описание структур данных 7](#_Toc531274524)

[4.3. Описание алгоритмов 9](#_Toc531274525)

[5. Заключение 11](#_Toc531274526)

[6. Литература 12](#_Toc531274527)

# Введение

**Структура данных** - программная единица, которая определяет метод хранения и обработки различных логически связанных данных в вычислительной технике. Знание структур данных позволяет наиболее компактно и практично расположить данные в памяти компьютера. В данной работе мы рассмотрим такую структуру данных, как треугольные матрицы.

**Ма́трица —** математический объект, записываемый в виде прямоугольной таблицы элементов кольца или поля, которая представляет собой совокупность строк и столбцов, на пересечении которых находятся её элементы. Количество строк и столбцов задает размер матрицы. Матрица называется треугольной, если все элементы, расположенные под главной (или побочной) диагональю равны нулю. Такая структура хранения очень удобна в случаях, когда:

- матрица симметрична (использование треугольных матриц позволяет сократить объем используемой памяти вдвое);

- в матрице содержится большое количество нулей.

Кроме пересиленных преимуществ, треугольную матрицу можно представить в виде вектора, состоящего из векторов фиксированного размера. Именно так мы и попробуем реализовать класс треугольных матриц

**Цель данной работы**  разработка структуры данных для хранения треугольных матриц с использованием векторов, а также освоение таких инструментов разработки программного обеспечения, как система контроля версий Git и фрэймворк для разработки автоматических тестов Google Test.

# Постановка задачи

Выполнение работы предполагает решение следующих задач:

1. Реализация класса вектора TVector.
2. Реализация класса треугольной матрицы TMatrix, унаследованного от класса TVector.
3. Разработка интерфейса для данных классов.
4. Обеспечение работоспособности примера использования.
5. Реализация нескольких тестов на базе Google Test.

# Руководство пользователя

При запуске программы на экран выводится сообщение, что все тесты пройдены успешною Примерs вывода сообщений для класса TVector и TMatrix показаны на рис.1 и рис.2 соответственно.

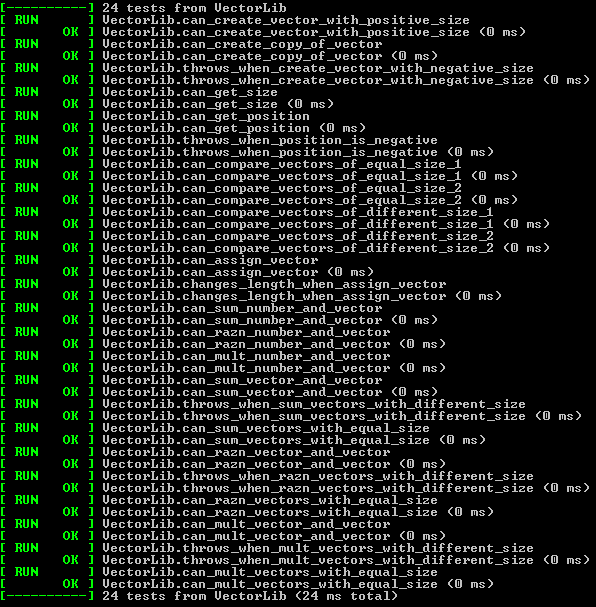


Рис. 1

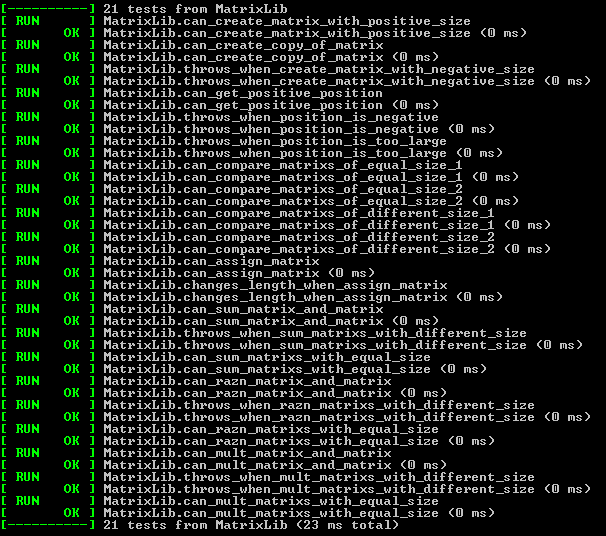


Рис. 2

# Руководство программиста

## Описание структуры программы

**Проект “Vector” состоит из следующих файлов:**

1. mainVector.cpp (в нём находится main)

**Проект “VectorLib ” состоит из следующих файлов:**

1. VectorLib.h (описание класса “TVector” реализация методов класса “ TVector” и перегрузка операций )
2. VectorLib.cpp

**Проект “Matrix” состоит из следующих файлов:**

1. mainVector.cpp (в нём находится main)

**Проект “MatrixLib ” состоит из следующих файлов:**

1. MatrixLib.h (описание класса “TMatrix” реализация методов класса “ TMatrix” и перегрузка операций )
2. MatrixLib.cpp

**Проект “ Test” состоит из следующих файлов:**

1. test\_vectorlib.cpp (реализация тестов для класса TVector)
2. test\_matrixlib.cpp (реализация тестов для класса TMatrix)

## Описание структур данных

***Класс TVector:***

*Поля:*

* \*vector - массив для представления вектора
* size - количество элементов вектора

*Конструкторы и деструктор:*

* TVector<T> (int s = 2);
* TVector<T> (const TVector<T> &v);
* ~TVector<T> ();

*Методы:*

* int GetSize(); - возвращает размер вектора
* T & operator[] (int pos); - возвращает значение i-ого элемента вектора

*Перегруженные операторы:*

* bool operator==(const TVector &v); - сравнение
* bool operator!=(const TVector &v);
* TVector<T> & operator= (const TVector &v);
* TVector<T> operator+ (const T a); - сложение с числом
* TVector<T> operator- (const T a); - вычитание числа
* TVector<T> operator\* (const T a); - умножение на число
* TVector<T> operator+ (const TVector &v); -сложение с вектором
* TVector<T> operator- (const TVector &v);-вычитание вектора
* TVector<T> operator\* (const TVector &v); - умножение векторов
* friend istream & operator>>( istream &in, TVector<T> &v); - ввод вектора
* friend ostream & operator<<( ostream &out, const TVector<T> &v); - вывод вектора

***Класс TMatrix:***

*Поля:*

Нет полей.

*Конструкторы:*

* TMatrix<T> (int s = 5);
* TMatrix<T> (const TMatrix<T> &mt);
* ~TMatrix<T>();

*Методы*:

* TVector<T>& operator [] (int i); - возвращает i-ый вектор
* - int GetMaxPower(void) const; - максимальная мощность множества
* void InsElem(const int Elem); - включить элемент в множество
* void DelElem(const int Elem); - удалить элемент из множества
* int IsMember(const int Elem) const; - проверить наличие элемента в множестве

*Перегруженные операторы:*

* bool operator == (const TMatrix<T> &mt); - сравнение
* bool operator != (const TMatrix<T> &mt); - сравнение
* TMatrix<T>& operator= (const TMatrix<T> &mt); - присваивание
* TMatrix<T> operator+ (const TMatrix<T> &mt); - сложение
* TMatrix<T> operator- (const TMatrix<T> &mt); - вычитание
* TMatrix<T> operator\* (const TMatrix<T> &mt); - умножение
* friend istream & operator>>( istream &in, TMatrix<T1> &mt); - ввод вектора
* friend ostream & operator<<( ostream &out, const TMatrix<T1> &mt); - вывод вектора

## Описание алгоритмов

***Формирование вектора:***

На входе мы имеем множество чисел. Его можно представить в виде вектора, где каждое число есть i-ый элемент. Элементы вектора удобнее всего хранить в массиве.

***Формирование массива:***

На входе мы имеем множество чисел. Его можно представить в виде множества(вектора) векторов, где каждый i-ый вектор имеет размер n-i. Элементы вектора удобнее всего хранить в массиве. Нумерация векторов идет с нуля.

Кроме того в программе были реализованы такие алгоритмы, как:

1. Чтобы получить каждый i-ый элемент вектора применяется алгоритм:

|  |  |
| --- | --- |
|  |  |

1. Чтобы очистить бит используем побитовое «И» между битами элемента массива pMem, в котором находится бит , и между дополнением его битовой маски:

|  |  |
| --- | --- |
|  |  |

1. Чтобы получить значение бита n используем побитовое «И» между битами элемента массива pMem, в котором находится бит , и между его битовой маской:

|  |  |
| --- | --- |
|  |  |

# 

# Заключение

В ходе выполнения лабораторной работы я смогла реализовать такую структуру данных, как битовое поле. Вместе с ней, согласно заданному интерфейсу, я реализовала классы TBitField и TSet. Написание нескольких своих тестов в дополнение к уже имеющимся, а так же достижение одной из главных целей – того, чтобы все тесты были пройдены, помогло мне разобраться с системой автоматических тестов Google Test.

В результате проделанной работы у меня получилось

1. Реализовать класс битового поля TBitField и класс множества TSet
2. Разработать интерфейс для данных классов.
3. Обеспеченить работоспособность примера использования.
4. Реализовать некоторые тесты на базе Google Test.

Таким образом, данная лабораторная работа отвечает всем поставленным задачам. Это означает, что цель работы была достигнута.

# Литература

* Книги

1. A.O. Грудзинский. Методы программирования, Издательство Нижегородского госуниверситета, 2006.
2. Васильев А.Н. Самоучитель С++ с примерами и задачами. -СПб.: Наука и Техника, 2016. -480с.

* Ссылки в Internet

1. 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.