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**

оставь надежды всяк сие читающий (карин т.а.)

**Отчет по учебной практике**

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

**Выполнил:**

студент группы 381706­1

Карин Тимофей Андреевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Проверил:**

ассистент каф. МОСТ ИИТММ

Лебедев Илья Геннадьевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc536219868)

[3. Руководство пользователя 5](#_Toc536219869)

[4. Руководство программиста 6](#_Toc536219870)

[4.1. Описание структуры программы 6](#_Toc536219871)

[4.2 Описание структур данных 6](#_Toc536219872)

[4.3 Описание алгоритмов 6](#_Toc536219873)

[5. Заключение 7](#_Toc536219874)

[6. Список литературы 8](#_Toc536219875)

1. Введение

Матрица – математический объект, записываемый в виде прямоугольной таблицы элементов. Есть несколько способов представления матриц в памяти компьютера. Обычно она хранится в виде двумерного массива. Но в данной лабораторной работе будет рассмотрен отдельный тип матриц, а именно, треугольные матрицы. Эти матрицы отличаются от обычных тем, что треугольные матрицы квадратные и что выше (верхнетреугольные) или ниже (нижнетреугольные) главной диагонали они заполнены нулями. В связи с этим, можно сделать вывод, что представлять такие матрицы в виде двумерного массива не рационально (хоть и очень удобно).

В данной работе мы будем рассматривать нижнетреугольные матрицы, и, так как для нас самым дорогим ресурсом является память, мы будем хранить только те элементы, которые находятся ниже главной диагонали. Делать это мы будем с помощью векторов. Каждая строка матрицы представляет из себя вектор определённой длины (конкретно длина равна номеру строки, увеличенному на единицу).

2. Постановка задачи

Основной целью данной работы является создание библиотеки для работы с треугольными матрицами, то есть работая только с элементами, находящимися ниже главной диагонали. Для этого необходимо выполнить следующие задачи:

* Реализовать класс для работы с векторами;
* Реализовать класс для работы с матрицами, в котором матрица представлена как вектор векторов, размер которых увеличивается от единицы до размера матрицы.
* Проверить работоспособность данной библиотеки с помощью фреймворка для написания тестов Google Test.

3. Руководство пользователя

4. Руководство программиста

4.1. Описание структуры программы

Программа состоит из 3 модулей:

* *VectorLib* – содежит библиотеку для работы с векторами и треугольными матрицами. Состоит из файлов *Vector.h* и *Matrix.h*, в которых хранятся шаблонные классы *TVector* и *TMatrix*.
* *Vector* – пример использования программы. Описан в главе 3.
* *VectorTest* – предназначен для тестирования библиотеки *VectorLib*. Состоит из файлов *VectorTest.cpp* и *MatrixTest.cpp*.

Кроме этих модулей в программе используется класс исключений *TExeption* и фреймворк Google Test.

4.2 Описание структур данных

Рассмотрим класс *TVector*. Класс шаблонный, *T –* шаблонный класс, присваиваемый элементам вектора. Рассмотрим поля и методы класса:

Со спецификатором доступа protected:

* *int size* – длина вектора (размерность);
* *T\* mas* – массив, в котором хранятся элементы вектора;

Со спецификатором доступа public:

* *TVector (int n = 0)* – конструктор;
* *TVector (TVector <T> &A)* – конструктор копирования;
* *TVector<T>& operator=(TVector<T> &A)* – оператор присваивания;
* *bool operator==(TVector &A)* – проверка на равенство;
* *bool operator!=(TVector &A)* – проверка на неравенство;
* *T& operator[](int i)* – оператор индексирования;
* *TVector<T> operator+(TVector<T> &A)* – сложение двух векторов;
* *TVector<T> operator-(TVector<T> &A)* – вычитание векторов;
* *T operator\*(TVector<T> &A)* – скалярное умножение векторов;
* *virtual ~TVector()* – деструктор;
* *friend ostream& operator<< (ostream& ostr, const TVector<T1> &A)* – оператор вывода (*T1* то же самое, что и *T*);
* *friend istream& operator>> (istream& istr, TVector<T1> &A) –* оператор ввода (*T1* то же самое, что и *T*);

Рассмотрим шаблонный класс *TMatrix*. Наследуется от TVector<TVector<T>> со спецификатором доступа public. *T* – шаблонный класс, присваиваемый элементам матрицы. Поля отсутствуют, все методы имеют спецификатор доступа public. Рассмотрим их:

* *TMatrix (int n=0) –* конструктор;
* *TMatrix (TMatrix<T>& A) –* конструктор копирования;
* *TMatrix<T> operator+(TMatrix<T>& A)* – сложение матриц;
* *TMatrix<T> operator-(TMatrix<T>& A) –* вычитание матриц;
* *TMatrix<T> operator\*(TMatrix<T>& A) –* умножение матриц;
* *TMatrix<T> operator/(TMatrix<T>& A) –* деление матриц;
* *TMatrix(TVector<TVector<T>> &A) –* конструктор приведения типа;
* *TMatrix<T>& operator=(TMatrix<T>& A) –* оператор присваивания;
* *bool operator==(TMatrix<T>& A) –* проверка на равенство;
* *bool operator!=(TMatrix<T>& A) –* проверка на неравенство;
* *friend ostream& operator<<(ostream& ostr, TMatrix<T1> &A) -* оператор вывода (*T1* то же самое, что и *T*);
* *friend istream& operator>>(istream& istr, TMatrix<T1> &A) -* оператор вывода (*T1* то же самое, что и *T*);

4.3 Описание алгоритмов

5. Заключение

В ходе проделанной работы были достигнуты следующие результаты:

* Реализован класс TVector, содержащий основные функции для работы с векторами;
* Написан класс TMatrix, предназначенный для работы с нижнетреугольными матрицами;
* Обработаны основные исключительные ситуации;
* Реализован пример использования программы;
* Написаны тесты, проверяющие все основные методы.

6. Список литературы

1. Ахо Альфред В, Хопкрофт Джон Э и Ульман Джеффри Д Структуры данных и алгоритмы [Книга]. - [б.м.] : Вильямс, 2003.
2. Лафоре Роберт Структуры данных и алгоритмы в Java [Книга]. - СПб : Питер, 2013. - 2 : стр. 704.
3. Павловская Т. А. C/C++ Программирование на языке высокого уровня [Книга]. - СПб : Питер, 2003.
4. Страуструп Бьерн Язык программирования C++ Бином, 2004.