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

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), …, 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:

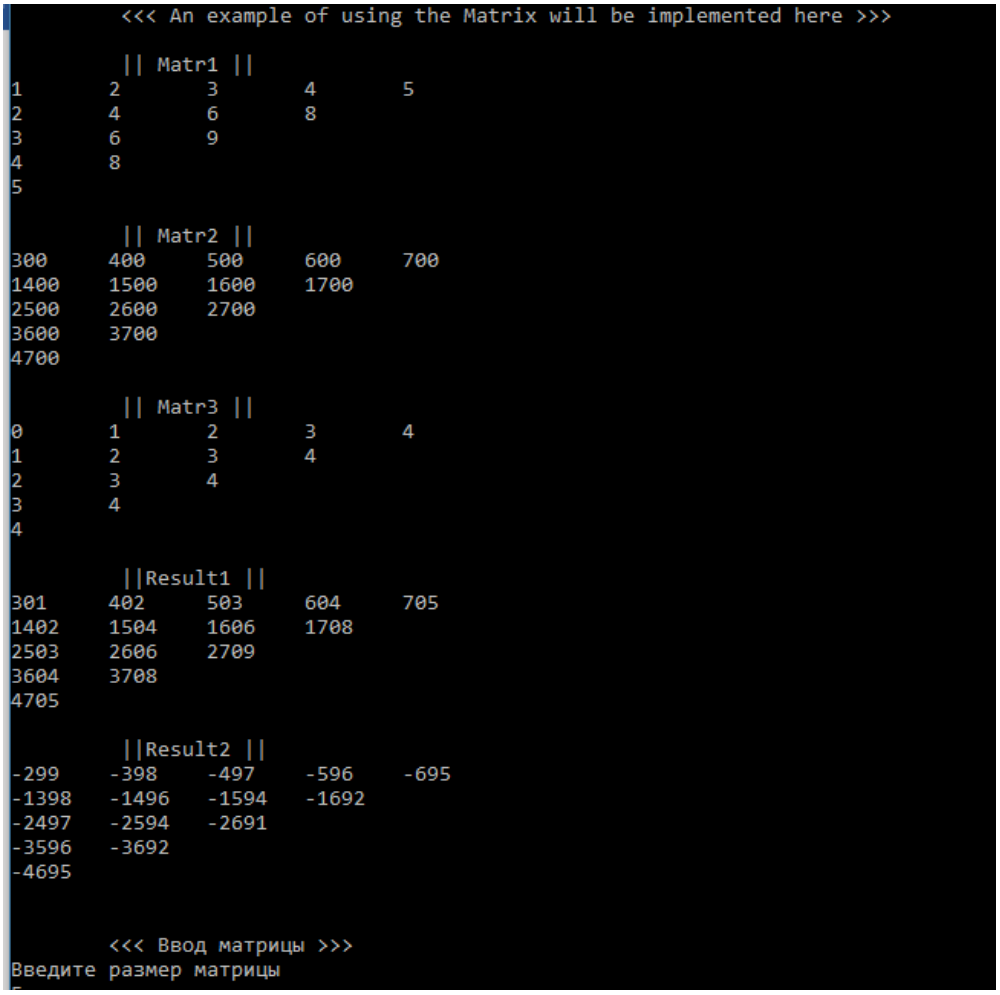

Рис. 1 – Пример использования

Сначала показывается *пример использования матриц*.

Result1 = Matr1 + Matr2;

Result2 = Matr1 - Matr2;

Result3 = Matr3 * Matr3;

Проверка 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:

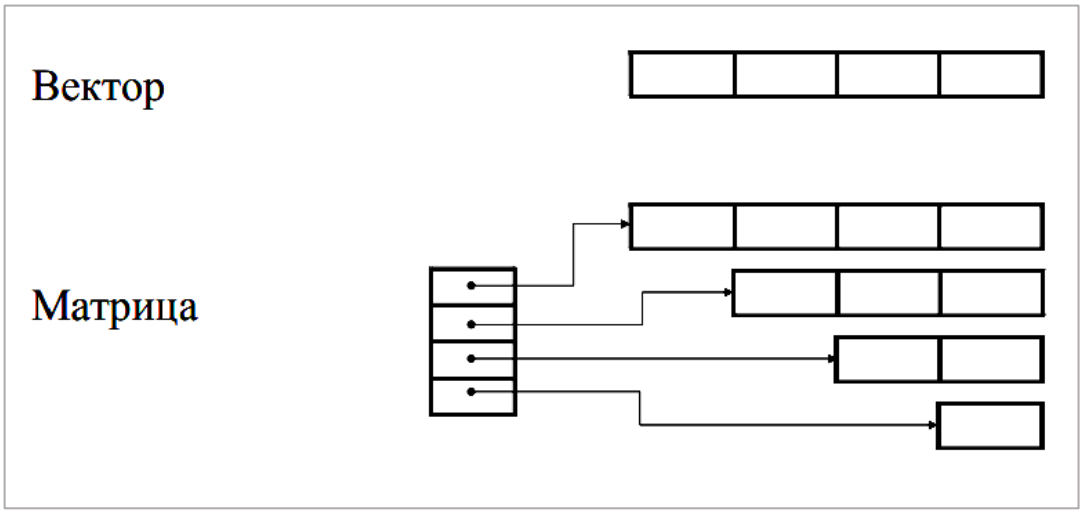

Рис. 2 – Представление матрицы в виде набора векторов

Класс TVector

Класс TVector является шаблонным классом.

Включены следующие два поля (protected - часть):

int dlina; - 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 масива.

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. Конструктор инициализации:

```
TMatrix<T>::TMatrix(int n) : TVector<TVector<T> >(n) //
{
 int a = MAX_SIZE;
 if (n < 0 || n > a)
        throw TException("Overflow");
 else
        for (int i = 0; i < n; i++)
          this-> Vector[i]= TVector <T>(n - i);
// заполняем элементы, расположенные выше побочной диагонали
}
```

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

```
TMatrix<T>::TMatrix(const TMatrix<T> &Matr) : TVector<TVector<T> >(Matr) {}

TMatrix<T>::TMatrix(const TVector<TVector<T> > &Matr) : TVector<TVector<T> >(Matr) { }

bool TMatrix<T>::operator==(const TMatrix<T> &Matr) const
{
 return TVector<TVector<T> >::operator==(Matr);
}

bool TMatrix<T>::operator!=(const TMatrix<T> &Matr) const
```

```
{
  return TVector<TVector<T> >::operator!=(Matr);
}

TMatrix<T>& TMatrix<T>::operator=(TVector<TVector<T> > &Matr)
{
  TVector<TVector<T> >::operator=(Matr);
  return *this;
}

TMatrix<T> TMatrix<T>::operator+(const TMatrix<T> &Matr)
{
  if (this -> dlina != Matr.dlina)
        throw TException("Different dimensions.");
  else
        return TVector<TVector<T> > :: operator+(Matr);
}

TMatrix<T> TMatrix<T>::operator-(const TMatrix<T> &Matr)
{
  if (this->dlina != Matr.dlina)
        throw TException("Different dimensions.");
  else
        return TVector<TVector<T> > :: operator-(Matr);
}
```

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]

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.