Федеральное государственное образовательное бюджетное учреждение

высшего профессионального образования

«Нижегородский Государственный Университет им.

Н.И.Лобачевского» (ННГУ)

Институт Информационных Технологий Математики и Механики

Отчёт по лабораторной работе

Классы для работы с векторами и матрицами.

Использование шаблонов

Выполнил:

студент группы 3821Б1ФИ3

Варданян Г.А.

Проверил:

заведующий лабораторией суперкомпьютерных технологий и высокопроизводительных вычислений

Лебедев И.Г

Нижний Новгород

2022г.

Содержание

[**Введение** 3](#_Toc104097330)

[**Постановка задачи** 4](#_Toc104097331)

[**Руководство пользователя** 5](#_Toc104097332)

[**Файл vector.h** 5](#_Toc104097333)

[**Файл Matrix.h** 6](#_Toc104097334)

[**Руководство программиста 8**](#_Toc104097335)

[**Файл vector.h 8**](#_Toc104097336)

[**Файл Matrix.h** 10](#_Toc104097338)

[**Файл main.cpp** 11](#_Toc104097339)

[**Описание структур данных** 13](#_Toc104097342)

[**Описание алгоритмов** 13](#_Toc104097343)

[**Заключение** 15](#_Toc104097344)

[**Список литературы** 16](#_Toc104097345)

**Введение**

Программирование — это интересный, полезный и увлекательный процесс, благодаря которому создаются программы – набор инструкций, которые приводятся в исполнение компьютерами. Одной из ключевых задач компьютера является работа с данными. В том числе и с классами, а в частности с шаблонами классов. Концепция шаблонов возникла из принципа программирования [Don't repeat yourself](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself), что переводится как “не повторяйся”. Шаблоны можно использовать для функций, структур и классов, как в сегодняшней лабораторной работе.

**Постановка задачи**

Написать классы для работы с векторами и матрицами, использовать шаблоны.

Матрица должна быть наследником вектора, классы вектора и матрицы должны быть вынесены в статическую библиотеку.

Продемонстрировать их работу (написать пример в main).

**Руководство пользователя**

**Файл vector.h**

 Сразу после запуска программы в консоль выводится два вектора, в данном случае состоящие из случайно сгенерированных чисел(см. рис1).



Рисунок 1. Векторы A и B.

Также получаем произведение этих векторов(см. рис2)



Рисунок 2. Произведение A на B.

После этого в консоль выводятся следующие 2 вектора, один из которых изначально заполнен цифрами 10, а второй прировнен к первому(см. рис3).



Рисунок 3. Векторы F и G.

Очевидно, что два вектора одинаковы, поэтому на их примере проверим работоспособность bool ==, а также разделим эти два вектора друг на друга(см. рис 4).



Рисунок 4. Проверка векторов на равенство.

Все предыдущие операции были над векторами типа int, проверим работоспособность программы с векторами типа double(см. рис 5).

Рисунок 5. Векторы D и E типа double.

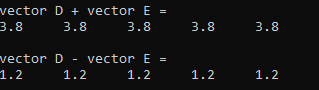
И также получаем сумму и разность двух этих векторов(см. рис6)

Рисунок 6. Сумма и разность векторов.

## Файл Matrix.h

В консоль выводятся две матрицы, заполненные случайными числами в диапазоне от 0 до 100(см.рис7).

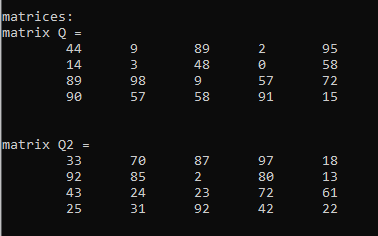


Рисунок 7. Матрицы.

После этого получаем сумму этих векторов и разность Q2 – Q(см.рис8)

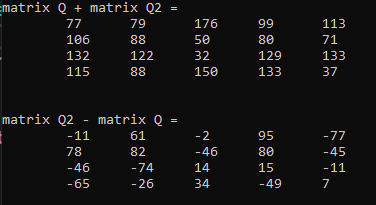
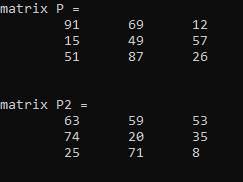


Рисунок 8. Сумма и разность матриц.

Как и в прошлом примере с векторами, сначала использовались матрицы типа int. Протестируем программу также с матрицами типа double(см. рис9).



В полученной матрице полученной в результате деления двух матриц наглядно видим, что используется именно тип double(см. рис10).

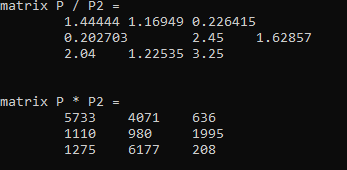


Рисунок 10. Произведение и частное матриц.

# Руководство программиста

## Файл vector.h

1. Подключение библиотек, с которыми предстоит работать.

|  |
| --- |
| #pragma once  #include <iostream>  using namespace std; |

1. Создание класса и его protected поле.

|  |
| --- |
| template<class T>  class TVector {  protected:  T\* data;  int len; |

1. Public поле класса.

|  |
| --- |
| TVector(int n, T v);  TVector(int n= 1);  TVector(const TVector<T>& p);  ~TVector();  int GetLen();  T& operator [] (int i);  TVector<T>& operator = (const TVector<T>& p);  TVector<T> operator + ( TVector<T>& p);  TVector<T> operator - (TVector<T>& p);  TVector<T> operator \* (TVector<T>& p);  TVector<T> operator / (TVector<T>& p);  friend ostream& operator << (ostream&, const TVector<T>&);  friend istream& operator >> (istream&, const TVector<T>&);  bool operator == (const TVector<T>& other); |

1. Описание конструкторов и деструктора.

|  |
| --- |
| template<class T>  inline TVector<T>::TVector(int n, T v)  {  ...  }  template<class T>  inline TVector<T>::TVector(int n)  {  ...  }  template<class T>  inline TVector<T>::TVector(const TVector<T>& p) {...}  template<class T>  inline TVector<T>::~TVector()  {  ...  } |

1. Описание GetLen и операторов класса.

|  |
| --- |
| template<class T>  inline int TVector<T>::GetLen()  {  return len;  }  template<class T>  inline T& TVector<T>::operator[](int i)  {  return data[i];  }  template<class T>  inline TVector<T>& TVector<T>::operator=(const TVector<T>& p)  {  ...  }  template<class T>  inline TVector<T> TVector<T>::operator+(TVector<T>& p)  {  ...  }  template<class T>  inline TVector<T> TVector<T>::operator-(TVector<T>& p)  {  ...  }  template<class T>  inline TVector<T> TVector<T>::operator\*(TVector<T>& p)  {  ...  }  template<class T>  inline TVector<T> TVector<T>::operator/(TVector<T>& p)  {  ...  }  template<class T>  inline bool TVector<T>::operator==(const TVector<T>& other)  {  ...  }  template<class T>  ostream& operator<<(ostream& o, TVector<T> & p) {.. |

**Файл Matrix.h**

1. Подключение библиотек, с которыми предстоит работать.

|  |
| --- |
| #pragma once  #include "Vector.h" |

1. Создание класса и его protected поле.

|  |
| --- |
| template <class A>  class TMatrix : public TVector<TVector<A>>  {  protected:  int len2; |

1. Public поле класса.

|  |
| --- |
| public:  TMatrix(int n = 1);  TMatrix(int n1, int n2);  ~TMatrix();  using TVector::operator=;    }; |

1. Конструкторы и деструктор.

|  |
| --- |
| template<class A>  inline TMatrix<A>::TMatrix(int n) : TVector<TVector<A>>::TVector(n)  {  ...  }  template<class A>  inline TMatrix<A>::TMatrix(int n1, int n2) : TVector<TVector<A>>::TVector(n1)  {  ...  }  template<class A>  inline TMatrix<A>::~TMatrix() {...} |

**Файл main.cpp**

1. Подключение библиотек, с которыми предстоит работать.

|  |
| --- |
| #include "Vector.h"  #include <iostream>  #include "Matrix.h"  using namespace std; |

1. Функция main.

|  |
| --- |
| int main()  {  cout << "vectors:" << endl;  TVector<int> A(5);  TVector<int> B(5);  TVector<int> C;  TVector<int> F(5, 10);  TVector<int> G;  TVector<int> H;  C = A \* B;  G = F;  H = F / G;  cout << "vector A = " << A << "vector B = " << B << endl;  cout << "vector A \* vector B = "<<"\n" << C << endl;  cout << "vector F = " << F << "vector G = " << G << endl;  bool equal = F == G;  cout << "vector F / vector G = " << "\n" << H << endl;  cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << "\n"<< endl;  TVector<double> D(5, 2.5);  TVector<double> E(5, 1.3);  TVector<double> D\_E\_sum;  TVector<double> D\_E\_sub;  D\_E\_sum = D + E;  D\_E\_sub = D - E;  cout << "vector D = " << D << "vector E =" << E << endl;  cout << "vector D + vector E = " << "\n" << D\_E\_sum << endl;  cout << "vector D - vector E = " << "\n" << D\_E\_sub << endl;  cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << "\n"<< endl;  cout << "matrices: " << endl;  TMatrix<int> Q(4,5);  TMatrix<int> Q2(4, 5);  TMatrix<int> QQ2(4, 5);  TMatrix<int> Q3(4, 5);  QQ2 = Q + Q2;  Q3 = Q2 - Q;  cout << "matrix Q = " << endl;  cout << "\t" << Q << endl;  cout << "matrix Q2 = " << endl;  cout << "\t" << Q2 << endl;  cout << "matrix Q + matrix Q2 = " << endl;  cout << "\t" << QQ2 << endl;  cout << "matrix Q2 - matrix Q = " << endl;  cout << "\t" << Q3 << endl;  TMatrix<double> P(3, 3);  TMatrix<double> P2(3, 3);  TMatrix<double> P3;  TMatrix<double> P4;  P3 = P / P2;  P4 = P \* P2;  cout << "matrix P = " << endl;  cout << "\t" << P << endl;  cout << "matrix P2 = " << endl;  cout << "\t" << P2 << endl;  cout << "matrix P / P2 = " << endl;  cout << "\t" << P3 << endl;  cout << "matrix P \* P2 = " << endl;  cout << "\t" << P4 << endl;  } |

**Описание структур данных**

При написании кода главного файла main.cpp использовались стандартная <iostream> и собственные заголовочные файлы “Vector.h” и “Matrix.h”

**Описание алгоритмов**

Прежде всего, создается шаблон класса и сам класс TVector предназначенный для работы с векторами, после создается класс TMatrix, который является наследником класса TVector.

С помощью подключения заголовочных файлов в main.cpp приводятся примеры работоспособности программы.

# Заключение

В ходе работы на языке С++ была написана программа использующая шаблоны классов, предназначенная для работы с векторами и матрицами.

# 

# Список литературы

1. Лекции по программированию <https://github.com/LebedevIlyaG/3821>