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

Векторы и матрицы. Создание статической библиотеки.

Выполнил:

студент группы 3821Б1ПМ3

Усов Д.В.

Проверил:

заведующий лабораторией

суперкомпьютерных технологий и

высокопроизводительных вычислений

Лебедев И.Г

Нижний Новгород

2021 г.

Содержание.

[Введение 3](#_Toc270962758)

[Постановка задачи 4](#_Toc270962759)

[Руководство пользователя 5](#_Toc270962760)

[Руководство программиста 6](#_Toc270962761)

[Описание структур данных 6](#_Toc270962762)

[Описание алгоритмов 6](#_Toc270962763)

[Описание структуры программы 6](#_Toc270962764)

[Заключение 7](#_Toc270962765)

[Литература 8](#_Toc270962766)

[Приложения 9](#_Toc270962767)

[Приложение 1 9](#_Toc270962768)

[Приложение 2 9](#_Toc270962769)

# Введение.

Программирование позволяет реализовывать получаемые людьми задачи в ЭВМ, благодаря чему людям не приходится самостоятельно решать их, позволяя производить все вычисления в электронном виде, благодаря чему единственное, что требуется от пользователя - ввести данные в программу.

В компьютерах реализованы многочисленные возможности для работы с данными различных типов, использовать которые позволяют языки программирования, включая созданные не изначальными разработчиками языка. Одними из важнейших не стандартных типов данных являются векторы и матрицы, позволяющие решать многочисленные математические задачи.

В данной работе будет создана статическая библиотека для работы с векторами и матрицами.

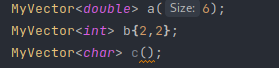
# Постановка задачи.

Библиотека должна позволять осуществлять основные математические действия (+, -, \*) над векторами и матрицами.

# Руководство пользователя.

# Вектора.

Класс MyVector<ClassName> является шаблонным,по этому принимает все типы данных.Рекомендованные (int,float,double,char).  
Класс вектор по умолчанию инициализируется нулями и является трехмерным.  
Варианты инициализации:



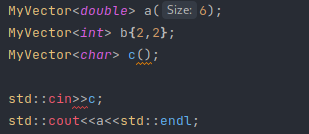
Результатом инциализации являются

a[0,0,0,0,0,0]

b[2,2]

c[“”,””,””]

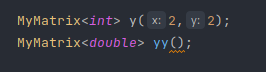
Класс имеет перегрузку операторов, сложения деления умножения вычитания, присвоения и др.Так же имеется перегрузка потокового вводы и вывода.Это позволяет удобным способом вводить и выводить значения.



Все арифметические операции выполняются по правилам математики.

# Матрицы.

Класс MyMatrix<ClassName> является шаблонным,по этому принимает все типы данных.Рекомендованные (int,float,double,char).  
Класс матриц по умолчанию инициализируется нулями и является матрицой третьего ранга.  
Варианты инициализации:

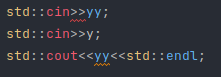


Результатом инциализации являются

y[[0,0],[0,0]]

yy[[0,0,0],[0,0,0],[0,0,0]]

Класс имеет перегрузку операторов, сложения деления умножения вычитания, присвоения и др.Так же имеется перегрузка потокового вводы и вывода.Это позволяет удобным способом вводить и выводить значения.

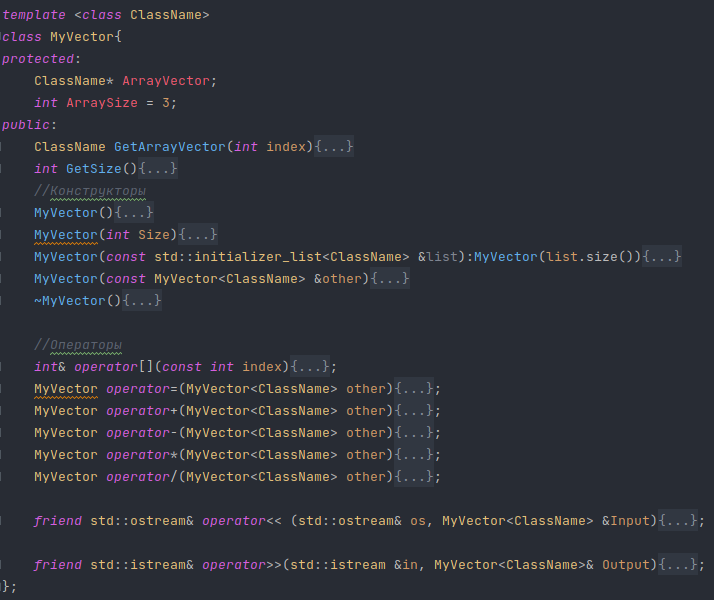


Все арифметические операции выполняются по правилам математики.

# Руководство программиста.

## Описание структуры кода программы.

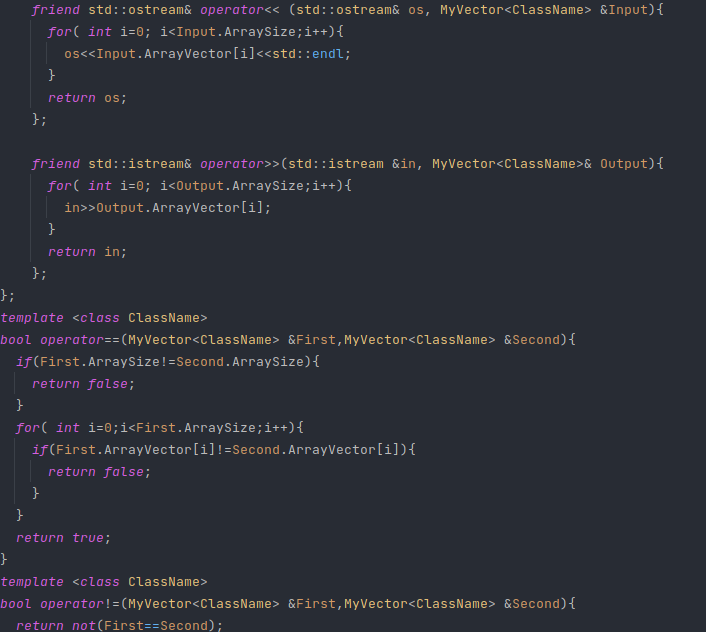
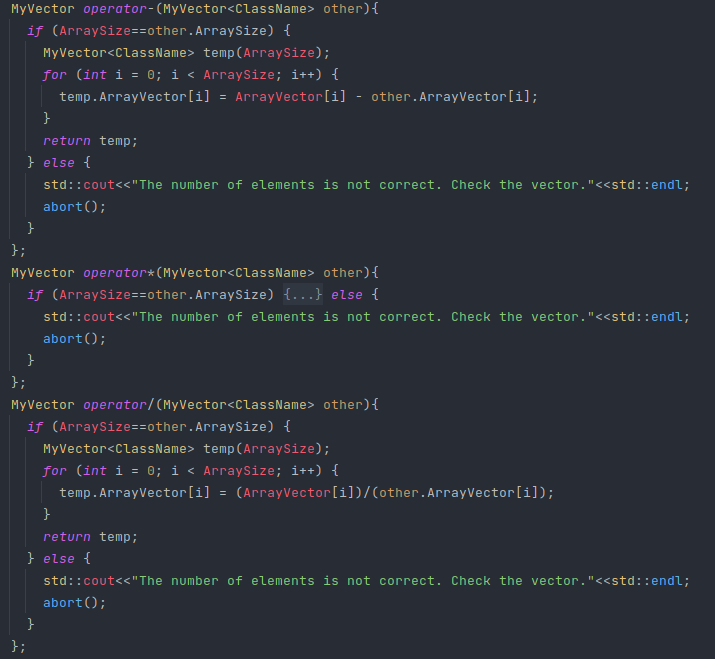
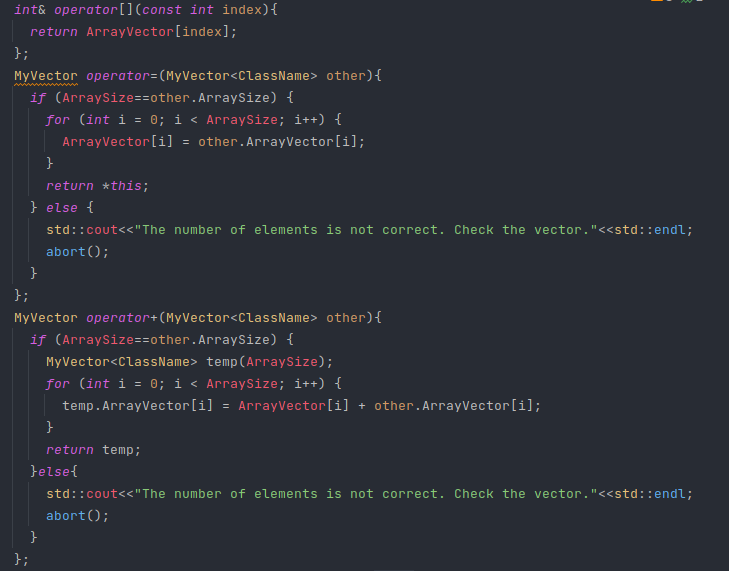
1. Вектор.
   1. Написание/создание основы шаблонного класса MyVector.



* 1. Написание конструкторов и деструктора.



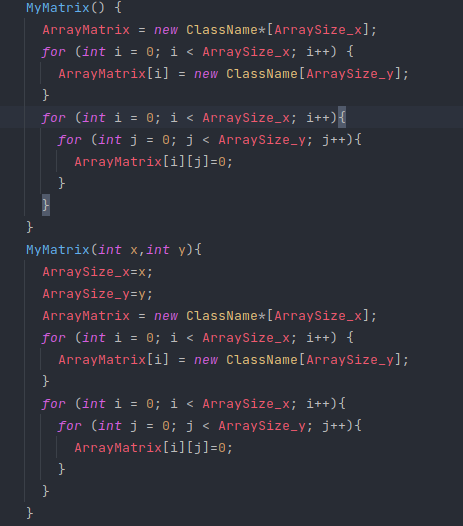
* 1. Написание операторов +, -, \*, /, =, ==, >>, <<, индексатора.

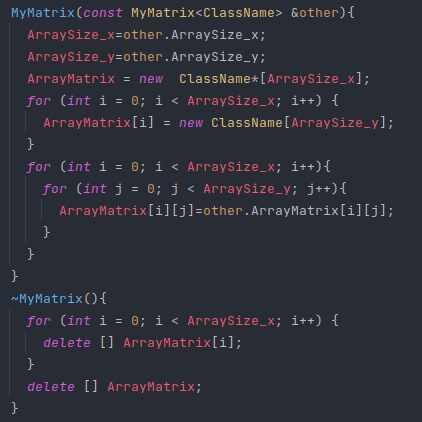


1. Матрица. Изначальная идея: наследник шаблонного класса MyVector от MyVector от шаблона.
   1. Основа класса.

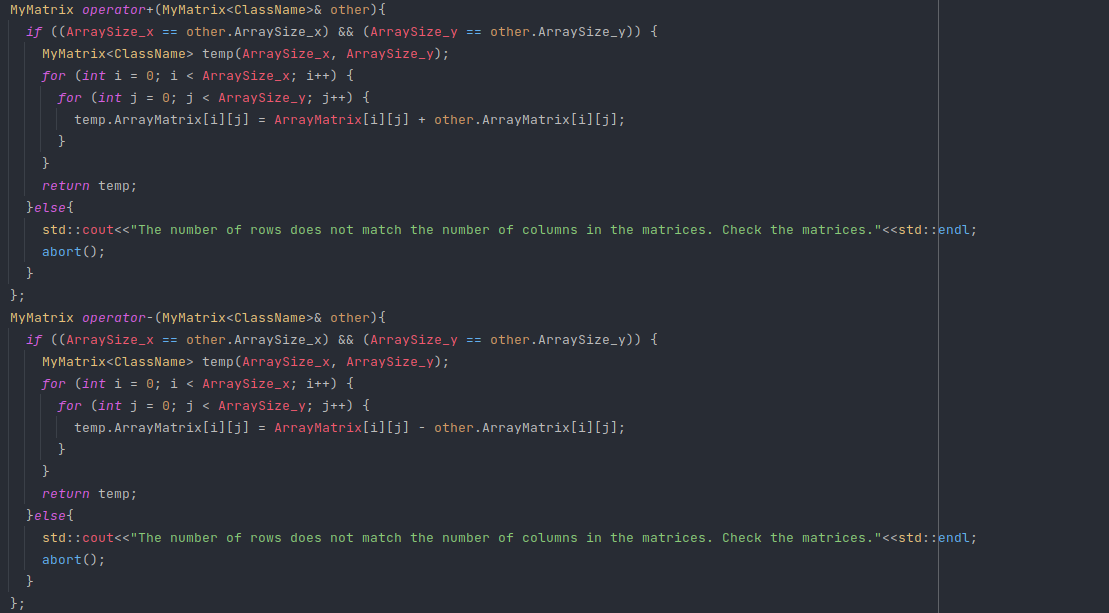


* 1. Конструкторы (в т.ч. копирования от MyVector<MyVector<ClassName>>, где ClassName - шаблон).

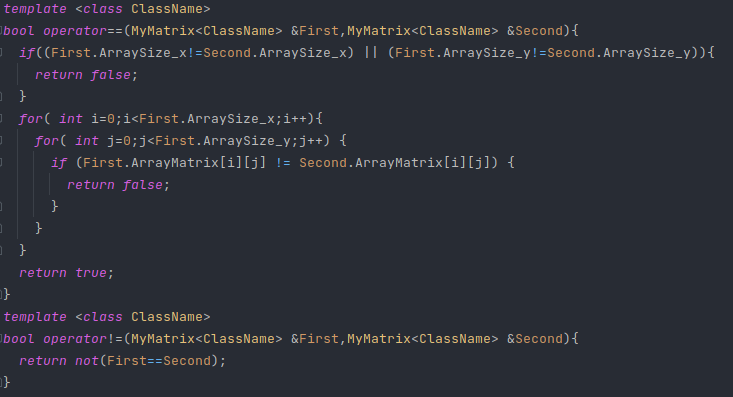
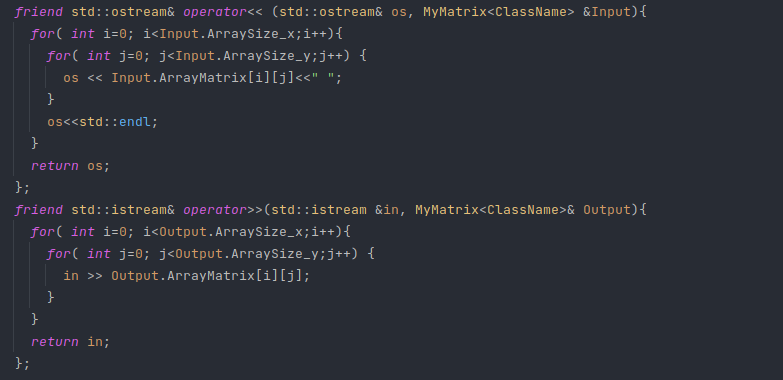




* 1. Операторы +, -, \*, =, <<, >>,==,!=индексатора.







## Описание структур данных.

* Структура MyVector имеет в себе защищённые поля: массив шаблона ClassName под названием ArrayVector и целое число ArraySize, обозначающее длину. В массиве ArrayVector находится ArraySize элементов типа ClassName, характеризующих вектор.
* Структура MyMatrix имеет в себе защищённые поля: двумерный массив ArrayMatrix шаблона ClassName и два поля целых чисел отвечающих за размерность матрицы ArraySize\_x и ArraySize\_y . В поле ArrayMatrix находятся массивы которые содержат ArraySize\_y эллемментов типа ClassName.

# Эксперименты.

# 1)

# Входные данные: [[1,2],[3,4]] Входные данные: [[1,2],[3,4]]

Результат:[[7,10],[15,20]]

# 2)

# Входные данные: [[1,2,3],[4,5,6,],[7,8,9]] Входные данные: [5,10,6]]

Результат:[43,106,69]

# 3)

# Входные данные: [[1,2,3],[4,5,6,],[7,8,9]] Входные данные: [5,129,76.6]]

Результат:[492.8,1124.6,1756.4]

**Эксперементы показали верный результат.**

# Заключение.

В ходе лабораторной работы была написана статическая библиотека С++, которая реализует вектор (класс MyVector) и матрицу (класс MyMatrix). В этих классах реализованы основные математические операторы (+, -, \*, /).

В ходе выполнения данной работы я улучшил свои навыки написания классов, реализации функций классов и отработал подключение внешней статической библиотеки в языке C++.

Выполнение подобных работ позволяет проверить свои знания в области выбранного языка, отточить умение написания программ или даже найти более эффективные методы реализация различных функций.

Список литературы.

1. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. — СПб.: Питер, 2003. —461 с: ил.
2. Сборник задач по программированию/Д.М. Златопольский — 3-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2011. — 304 с.

# Приложение 1.

MyVector.h

|  |
| --- |
| *#pragma* once  *template* <*class* ClassName> *class* MyVector{ *protected*:  ClassName\* ArrayVector;  *int* ArraySize = 3; *public*:  ClassName GetArrayVector(*int* index){  *return* ArrayVector[index];  }  *int* GetSize(){  *return* ArraySize;  }  *//Конструкторы* MyVector(){  ArrayVector = *new* ClassName[ArraySize];  *for*( *int* i=0; i<ArraySize; i++){  ArrayVector[i]=0;  }  }  MyVector(*int* Size){  ArraySize=Size;  ArrayVector = *new* ClassName[ArraySize];  *for*( *int* i=0; i<ArraySize; i++){  ArrayVector[i]=0;  }  }  MyVector(*const* std::initializer\_list<ClassName> &list):MyVector(list.size()){  *int* count = 0;  *for* (*auto* &element : list)  {  ArrayVector[count] = element;  ++count;  }  }  MyVector(*const* MyVector<ClassName> &other){  ArraySize=other.ArraySize;  ArrayVector = *new* ClassName[ArraySize];  *for*( *int* i=0;i<ArraySize;i++){  ArrayVector[i]=other.ArrayVector[i];  }  }  ~MyVector(){  *delete* [] ArrayVector;  }   *//Операторы  int*& *operator*[](*const int* index){  *return* ArrayVector[index];  };  MyVector *operator*=(MyVector<ClassName> other){  *if* (ArraySize==other.ArraySize) {  *for* (*int* i = 0; i < ArraySize; i++) {  ArrayVector[i] = other.ArrayVector[i];  }  *return* \**this*;  } *else* {  std::cout<<"The number of elements is not correct. Check the vector."<<std::endl;  abort();  }  };  MyVector *operator*+(MyVector<ClassName> other){  *if* (ArraySize==other.ArraySize) {  MyVector<ClassName> temp(ArraySize);  *for* (*int* i = 0; i < ArraySize; i++) {  temp.ArrayVector[i] = ArrayVector[i] + other.ArrayVector[i];  }  *return* temp;  }*else*{  std::cout<<"The number of elements is not correct. Check the vector."<<std::endl;  abort();  }  };  MyVector *operator*-(MyVector<ClassName> other){  *if* (ArraySize==other.ArraySize) {  MyVector<ClassName> temp(ArraySize);  *for* (*int* i = 0; i < ArraySize; i++) {  temp.ArrayVector[i] = ArrayVector[i] - other.ArrayVector[i];  }  *return* temp;  } *else* {  std::cout<<"The number of elements is not correct. Check the vector."<<std::endl;  abort();  }  };  MyVector *operator*\*(MyVector<ClassName> other){  *if* (ArraySize==other.ArraySize) {  MyVector<ClassName> temp(ArraySize);  *for* (*int* i = 0; i < ArraySize; i++) {  temp.ArrayVector[i] = ArrayVector[i] \* other.ArrayVector[i];  }  *return* temp;  } *else* {  std::cout<<"The number of elements is not correct. Check the vector."<<std::endl;  abort();  }  };  MyVector *operator*/(MyVector<ClassName> other){  *if* (ArraySize==other.ArraySize) {  MyVector<ClassName> temp(ArraySize);  *for* (*int* i = 0; i < ArraySize; i++) {  temp.ArrayVector[i] = (ArrayVector[i])/(other.ArrayVector[i]);  }  *return* temp;  } *else* {  std::cout<<"The number of elements is not correct. Check the vector."<<std::endl;  abort();  }  };   *friend* std::ostream& *operator*<< (std::ostream& os, MyVector<ClassName> &Input){  *for*( *int* i=0; i<Input.ArraySize;i++){  os<<Input.ArrayVector[i]<<std::endl;  }  *return* os;  };   *friend* std::istream& *operator*>>(std::istream &in, MyVector<ClassName>& Output){  *for*( *int* i=0; i<Output.ArraySize;i++){  in>>Output.ArrayVector[i];  }  *return* in;  }; }; *template* <*class* ClassName> *bool operator*==(MyVector<ClassName> &First,MyVector<ClassName> &Second){  *if*(First.ArraySize!=Second.ArraySize){  *return false*;  }  *for*( *int* i=0;i<First.ArraySize;i++){  *if*(First.ArrayVector[i]!=Second.ArrayVector[i]){  *return false*;  }  }  *return true*; } *template* <*class* ClassName> *bool operator*!=(MyVector<ClassName> &First,MyVector<ClassName> &Second){  *return not*(First==Second); } |

MyMatrix.h

|  |
| --- |
| *#pragma* once *#include* "MyVector.h"  *template* <*class* ClassName> *class* MyMatrix :*public* MyVector<ClassName>{ *protected*:  ClassName \*\*ArrayMatrix;  *int* ArraySize\_x = 3;  *int* ArraySize\_y = 3; *public*:  *//Конструкторы* MyMatrix() {  ArrayMatrix = *new* ClassName\*[ArraySize\_x];  *for* (*int* i = 0; i < ArraySize\_x; i++) {  ArrayMatrix[i] = *new* ClassName[ArraySize\_y];  }  *for* (*int* i = 0; i < ArraySize\_x; i++){  *for* (*int* j = 0; j < ArraySize\_y; j++){  ArrayMatrix[i][j]=0;  }  }  }  MyMatrix(*int* x,*int* y){  ArraySize\_x=x;  ArraySize\_y=y;  ArrayMatrix = *new* ClassName\*[ArraySize\_x];  *for* (*int* i = 0; i < ArraySize\_x; i++) {  ArrayMatrix[i] = *new* ClassName[ArraySize\_y];  }  *for* (*int* i = 0; i < ArraySize\_x; i++){  *for* (*int* j = 0; j < ArraySize\_y; j++){  ArrayMatrix[i][j]=0;  }  }  }  MyMatrix(*const* MyMatrix<ClassName> &other){  ArrayMatrix = *new* ClassName\*[other.ArraySize\_x];  *for* (*int* i = 0; i < other.ArraySize\_x; i++) {  ArrayMatrix[i] = *new* ClassName[other.ArraySize\_y];  }  *for* (*int* i = 0; i < other.ArraySize\_x; i++){  *for* (*int* j = 0; j < other.ArraySize\_y; j++){  ArrayMatrix[i][j]=other.ArrayMatrix[i][j];  }  }  }  ~MyMatrix(){  *for* (*int* i = 0; i < ArraySize\_x; i++) {  *delete* [] ArrayMatrix[i];  }  *delete* [] ArrayMatrix;  }   *//Операторы  auto operator*[](*const int* index){  *return* ArrayMatrix[index];  };  MyMatrix& *operator* = (MyMatrix<ClassName> other) {  *for* (*int* i = 0; i < ArraySize\_x; i++) {  *for* (*int* j = 0; j < ArraySize\_y; j++) {  ArrayMatrix[i][j] = other.ArrayMatrix[i][j];  }  }  *return* \**this*;  };  MyMatrix *operator*+(MyMatrix<ClassName>& other){  *if* ((ArraySize\_x == other.ArraySize\_x) && (ArraySize\_y == other.ArraySize\_y)) {  MyMatrix<ClassName> temp(ArraySize\_x, ArraySize\_y);  *for* (*int* i = 0; i < ArraySize\_x; i++) {  *for* (*int* j = 0; j < ArraySize\_y; j++) {  temp.ArrayMatrix[i][j] = ArrayMatrix[i][j] + other.ArrayMatrix[i][j];  }  }  *return* temp;  }*else*{  std::cout<<"The number of rows does not match the number of columns in the matrices. Check the matrices."<<std::endl;  abort();  }  };  MyMatrix *operator*-(MyMatrix<ClassName>& other){  *if* ((ArraySize\_x == other.ArraySize\_x) && (ArraySize\_y == other.ArraySize\_y)) {  MyMatrix<ClassName> temp(ArraySize\_x, ArraySize\_y);  *for* (*int* i = 0; i < ArraySize\_x; i++) {  *for* (*int* j = 0; j < ArraySize\_y; j++) {  temp.ArrayMatrix[i][j] = ArrayMatrix[i][j] - other.ArrayMatrix[i][j];  }  }  *return* temp;  }*else*{  std::cout<<"The number of rows does not match the number of columns in the matrices. Check the matrices."<<std::endl;  abort();  }  };  MyMatrix *operator*\*(MyMatrix<ClassName>& other){  *if*(ArraySize\_y==other.ArraySize\_x) {  MyMatrix<ClassName> temp(ArraySize\_x, other.ArraySize\_y);  *for* (*int* i = 0; i < ArraySize\_x; i++) {  *for* (*int* j = 0; j < other.ArraySize\_x; j++) {  *for* (*int* k = 0; k < ArraySize\_y; k++) {  temp.ArrayMatrix[i][j] += ArrayMatrix[i][k] \* other.ArrayMatrix[k][j];  }  }  }  *return* temp;  }  *else*{  std::cout<<"The number of rows does not match the number of columns in the matrices. Check the matrices."<<std::endl;  abort();  }  };   MyMatrix *operator*\*(MyVector<ClassName>& other){  *if*(ArraySize\_y==other.GetSize()) {  MyMatrix<ClassName> temp(other.GetSize(),1);  *for* (*int* i = 0; i < temp.ArraySize\_x; i++) {  *for* (*int* j = 0; j < 1; j++) {  *for* (*int* k = 0; k < other.GetSize(); k++) {  temp.ArrayMatrix[i][j] += ArrayMatrix[i][k] \* other.GetArrayVector(k);  }  }  }  std::cout<<temp<<std::endl;  *return* temp;  }  *else*{  std::cout<<"The number of rows does not match the number of columns in the matrices. Check the matrices."<<std::endl;  abort();  }  };       *friend* std::ostream& *operator*<< (std::ostream& os, MyMatrix<ClassName> &Input){  *for*( *int* i=0; i<Input.ArraySize\_x;i++){  *for*( *int* j=0; j<Input.ArraySize\_y;j++) {  os << Input.ArrayMatrix[i][j]<<" ";  }  os<<std::endl;  }  *return* os;  };  *friend* std::istream& *operator*>>(std::istream &in, MyMatrix<ClassName>& Output){  *for*( *int* i=0; i<Output.ArraySize\_x;i++){  *for*( *int* j=0; j<Output.ArraySize\_y;j++) {  in >> Output.ArrayMatrix[i][j];  }  }  *return* in;  };  }; *template* <*class* ClassName> *bool operator*==(MyMatrix<ClassName> &First,MyMatrix<ClassName> &Second){  *if*((First.ArraySize\_x!=Second.ArraySize\_x) || (First.ArraySize\_y!=Second.ArraySize\_y)){  *return false*;  }  *for*( *int* i=0;i<First.ArraySize\_x;i++){  *for*( *int* j=0;j<First.ArraySize\_y;j++) {  *if* (First.ArrayMatrix[i][j] != Second.ArrayMatrix[i][j]) {  *return false*;  }  }  }  *return true*; } *template* <*class* ClassName> *bool operator*!=(MyMatrix<ClassName> &First,MyMatrix<ClassName> &Second){  *return not*(First==Second); }  } |