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

2022

**Содержание.**

# 

[Введение 2](#_Toc270962758)

[Постановка задачи 3](#_Toc270962759)

[Руководство пользователя 4](#_Toc270962760)

[Руководство программиста 6](#_Toc270962761)

[Описание структуры библиотеки с ее структур данных 6](#_Toc270962762)

[Описание алгоритмов 2](#_Toc270962763)1

[Эксперименты 2](#_Toc270962761)2

[Заключение 2](#_Toc270962765)3

[Литература 2](#_Toc270962766)4

[Приложения 2](#_Toc270962767)5

[Приложение 1 2](#_Toc270962768)5

[Приложение 2 3](#_Toc270962769)5

# 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.

Необходимо написать классы для работы с векторами и матрицами, использовав шаблоны. Матрица должна быть наследником вектора. Классы вектора и матрицы должны быть внесены в статическую библиотеку. Необходимо продемонстрировать их работу на примере.

В программе должны быть:

• Конструкторы (по умолчанию, инициализатор, копирования);

• Деструктор;

• Доступ к защищенным полям;

• Перегруженные операции: +,-,\*,/, =, ==,[ ], потоковый ввод и вывод(перегруженные операции +,-,\*,/ должны быть реализованы для векторов (вектор +,-,\*,/ вектор), матриц(матрица +,-,\* матрица), матрично-векторные(матрица \* вектор и наоборот

**Руководство пользователя.**

Была написана библиотека векторов и матриц, в которой пользователь может работать с данными объектами, а именно задавать, удалять, инициализировать, складывать, вычитать, умножать, делить, сравнивать, присваивать, а так же производить потоковый ввод и вывод.

Классы в С++ - это абстракция описывающая методы, свойства, ещё не существующих объектов. Объекты - конкретное представление абстракции, имеющее свои свойства и методы. Чтобы воспользоваться программой необходимо создать объекты из класса «TVector» или «TMatrix». Для создания объекта применяются конструкторы. По сути конструкторы представляют специальные методы, которые называются так же как и класс, и которые вызываются при создании нового объекта и выполняют инициализацию объекта.

|  |
| --- |
| MyVector(); // Конструктор вектора по умолчанию  MyVector(const std::initializer\_list<ClassName> &list)}; // Конструктор инициализатор создает вектор заполненный передоваемыми значениями  MyVector(int Size); // Конструктор вектора длинны Size  MyMatrix() // Конструктор матрицы по умолчанию  MyMatrix(int x,int y); // Конструктор матрицы размера x \* y |

*Фрагмент кода 1. Конструкторы.*

|  |
| --- |
| MyVector<Type> test(); // Создание вектора по умолчанию  MyVector<Type> test(){a,b,c}; // Создание и Инициализция вектора типа Type со значениями a,b,c  MyVector<Type> test(n); // Создание вектора длинны n типа Type  MyMatrix<Type> test(); // Создание матрицы по умолчанию типа Type  MyMatrix<Type> test(n,m); // Создание матрицы размера n \* m типа Type |

*Фрагмент кода 2. Создание обекта.*

После создания объекта можно воспользоваться функционалом класса. Для обращения к функциональности класса - полям, методам применяется точечная нотация точки - после объекта класса ставится точка, а затем элемент класса: объект поле класса или объект.метод класса (параметры метода). Кроме этого С++ позволяет организовать перегрузку операций. Механизм перегрузки операций позволяет обеспечить более традиционную и удобную запись действий над объектами.

|  |
| --- |
| MyMatrix<int> test(2,2); // Создание матрицы 2\*2  std::cin>>test; // Потоковое заполнение матрицы  test=test\*test; // Умножение матрицы саму на себя и запись в ту же переменную |

*Фрагмент кода 3.Пример использования библиотеки.*

**Руководство програмиста.**

**Описание структуры библиотеки и ее структур данных.**

Была написана библиотека векторов и матриц, в которой пользователь может работать с данными объектами, а именно задавать, удалять, инициализировать, складывать, вычитать, умножать, делить, сравнивать, присваивать, а так же производить потоковый ввод и вывод.

Рассмотрим конструкторы данной библиотеки:

1. Матрицы.
   1. Конструктор по умолчанию.

|  |
| --- |
| 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;  }  }  } |

*Фрагмент кода 4.Конструктор по умолчанию.*

* 1. Конструктор с параметрами размера.

|  |
| --- |
| 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;  }  }  } |

*Фрагмент кода 5.Конструктор с параметрами размера.*

* 1. Конструктор копирования.

|  |
| --- |
| 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];  }  }  } |

*Фрагмент кода 6.Конструктор копирования.*

* 1. Деструктор.

|  |
| --- |
| ~MyMatrix(){  for (int i = 0; i < ArraySize\_x; i++) {  delete [] ArrayMatrix[i];  }  delete [] ArrayMatrix;  } |

*Фрагмент кода 7.Деструктор.*

1. Вектора.
   1. Констркутор по умолчанию.

|  |
| --- |
| MyVector(){  ArrayVector = new ClassName[ArraySize];  for( int i=0; i<ArraySize; i++){  ArrayVector[i]=0;  }  } |

*Фрагмент кода 8.Конструктор по умолчанию.*

* 1. Конструктор с параметром размера.

|  |
| --- |
| MyVector(int Size){  ArraySize=Size;  ArrayVector = new ClassName[ArraySize];  for( int i=0; i<ArraySize; i++){  ArrayVector[i]=0;  }  } |

*Фрагмент кода 9. Коснтруктор с параметром размера.*

* 1. Конструктор с листом инициализации.

|  |
| --- |
| MyVector(const std::initializer\_list<ClassName> &list):MyVector(list.size()){  int count = 0;  for (auto &element : list)  {  ArrayVector[count] = element;  ++count;  }  } |

*Фрагмент кода 10.Конструктор с листом инициализации.*

* 1. Конструктор копирования.

|  |
| --- |
| MyVector(const MyVector<ClassName> &other){  ArraySize=other.ArraySize;  ArrayVector = new ClassName[ArraySize];  for( int i=0;i<ArraySize;i++){  ArrayVector[i]=other.ArrayVector[i];  }  } |

*Фрагмент кода 11.Конструктор копирования.*

* 1. Деструктор.

|  |
| --- |
| ~MyVector(){  delete [] ArrayVector;  } |

*Фрагмент кода 12.Деструктор.*

Рассмотрим перегрузку операторов данной библиотеки:

1. Матрицы
   1. Оператор индексации.

|  |
| --- |
| auto operator[](const int index){  return ArrayMatrix[index];  }; |

*Фрагмент кода 13.Индексация.*

* 1. Оператор присваения.

|  |
| --- |
| MyMatrix<ClassName>& 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;  }; |

*Фрагмент кода 14.Присваение.*

* 1. Оператор сложения.

|  |
| --- |
| MyMatrix<ClassName> 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();  }  }; |

Фрагмент кода 15.Сложение.

* 1. Оператор разности.

|  |
| --- |
| MyMatrix<ClassName> 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();  }  }; |

*Фрагмент кода 16.Разность.*

* 1. Оператор умножения.

|  |
| --- |
| MyMatrix<ClassName> 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();  }  }; |

*Фрагмент кода 17.Умножение.*

* 1. Оператор равенства.

|  |
| --- |
| 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;  } |

*Фрагмент кода 18.Равенство.*

* 1. Оператор неравенства.

|  |
| --- |
| template <class ClassName>  bool operator!=(MyMatrix<ClassName> &First,MyMatrix<ClassName> &Second){  return not(First==Second);  } |

*Фрагмент кода 19.Неравенство.*

* 1. Оператор >>

|  |
| --- |
| 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;  }; |

*Фрагмент кода 20.>>.*

* 1. Оператор <<

|  |
| --- |
| 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;  }; |

*Фрагмент кода 20.<<.*

1. Вектора
   1. Оператор индексации.

|  |
| --- |
| int& operator[](const int index){  return ArrayVector[index];  }; |

*Фрагмент кода 21.Индексация.*

* 1. Оператор присваения.

|  |
| --- |
| MyVector<ClassName> 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();  }  }; |

*Фрагмент кода 22.Присваение.*

* 1. Оператор сложения.

|  |
| --- |
| MyVector<ClassName> 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();  }  }; |

*Фрагмент кода 23.Сложение.*

* 1. Оператор отрицания.

|  |
| --- |
| MyVector<ClassName> 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();  }  }; |

*Фрагмент кода 24.Отрицание.*

* 1. Оператор умножения.

|  |
| --- |
| MyVector<ClassName> 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();  }  }; |

*Фрагмент кода 25.Умножения.*

* 1. Оператор равенства.

|  |
| --- |
| 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;  } |

*Фрагмент кода 26.Равенства.*

* 1. Оператор неравенства.

|  |
| --- |
| template <class ClassName>  bool operator!=(MyVector<ClassName> &First,MyVector<ClassName> &Second){  return not(First==Second);  } |

*Фрагмент кода 27.Неравенства.*

* 1. Оператор >>

|  |
| --- |
| friend std::istream& operator>>(std::istream &in, MyVector<ClassName>& Output){  for( int i=0; i<Output.ArraySize;i++){  in>>Output.ArrayVector[i];  }  return in;  }; |

*Фрагмент кода 28.>>.*

* 1. Оператор <<

|  |
| --- |
| 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;  }; |

*Фрагмент кода 29.<<.*

В классе векторов есть два защищенных поля ArrayVector и ArraySize. ArrayVector представляет собой одномерный массив длинны ArraySize.

В классе матриц есть три зищещенных поля ArrayMatrix, ArraySize\_x, ArraySize\_y.ArrayMatrix представляет собой двумерный масив длинны ArraySize\_x и ширины ArraySize\_y.

**Описание алгоритмов.**

Едиснтвенным сложным алгоритмом является умножение матриц.Поскольку [умножение матриц](https://ru.wikipedia.org/wiki/%D0%A3%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86" \o "Умножение матриц) является центральной операцией во многих [численных алгоритмах](https://ru.wikipedia.org/wiki/%D0%A7%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D1%8B" \o "Численные методы), много усилий было вложено в повышение эффективности алгоритма умножения матриц. Приложения алгоритма умножения матриц в вычислительных задачах найдены во многих областях, включая [научные вычисления](https://ru.wikipedia.org/w/index.php?title=%D0%9D%D0%B0%D1%83%D1%87%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F&action=edit&redlink=1" \o "Научные вычисления (страница отсутствует)) и [распознавания образов](https://ru.wikipedia.org/w/index.php?title=%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D1%80%D0%B0%D1%81%D0%BF%D0%BE%D0%B7%D0%BD%D0%B0%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2&action=edit&redlink=1" \o "Теория распознавание образов (страница отсутствует)), а также во вроде бы не имеющих отношение к матрицам задачах, таких как подсчёт путей через [граф](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)" \o ").

[По определению умножения матриц](https://ru.wikipedia.org/wiki/%D0%A3%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86" \o "Умножение матриц) для n\*m матрицы A и по m\*p матрицы В произведением C = AB является матрица n\*p, состоящая из эллементов:

cij=aikbkjОтсюда я создал алгоритм умножения матрицы на матрицу.

**Эксперименты.**

Замерим время работы программы для некоторых арифметических действий. Для этого воспользуемся библиотекой «time.h». Будем замерять время через тактовую частоту (что является неточным способом). Для замеров воспользуемся командами: clock\_t start = clock(); clock\_t end = clock(); double seconds = (double)(end - start) / CLOCKS\_PER\_SEC; std::cout << "seconds = " << seconds << std::endl;

Сначала измерим оператор умножения матрицу на матрицу. Сложность данного алгоритма o(n^3).

Проведём первый замер для матриц 100x100, которые заполнены однозначными числами. Получили, что время равно 0.002. Теперь зададим матрицу 1000x1000. Данные больше в 10 раз => время должно различаться в 1000 раз. Получили время, равное 3,528. Разница в 1000 раз. 1,5 секунды погрешность, возможно, из-за тактовой частоты, но порядок одинаков.

Теперь сравним время для операции сложения. Зададим матрицы 1000x1000. Получим время равное 0.002. Зададим матрицы 10000x10000. Получим время, равное 0,218. Так как сложность алгоритма сложения равна o(n^2), то при увеличении данных в 10 раз, получаем увеличение времени в 100 раз. Что мы и получили.

Ожидания по времени работы программы оправдались.

# 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.

# Приложения.

# **Приложение 1.**

|  |
| --- |
| #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<ClassName>& 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<ClassName> 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<ClassName> 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<ClassName> 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<ClassName> 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);  } |

# **Приложение 2.**

|  |
| --- |
| #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<ClassName> 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<ClassName> 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<ClassName> 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<ClassName> 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<ClassName> 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);  } |