Федеральное агентство по образованию Российской Федерации

Государственное образовательное учреждение

высшего профессионального образования

Нижегородский государственный университет им. Н. И. Лобачевского

Институт информационных технологий математики и механики

Отчёт по лабораторной работе

Вектора и матрицы.

Выполнил:

студент 1-го курса ИИТММ гр. 3821Б1ПМ3

Мерзляков В.А

Проверил:

Заведующий лабораторией

суперкомпьютерных технологий и

высокопроизводительных

вычислений

Лебедев И. Г.

Нижний Новгород

2021 г.

Содержание

1.[Введение 3](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962758)

2.[Постановка задачи 4](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962759)

3.[Руководство пользователя.Вектор 5](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962760)

3.1.[Руководство пользователя.Матрица 6](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962761)

4.[Руководство программиста 7](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962760)

4.1.[Описание структуры программы.Вектор 7-9](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962762)

4.2.[Описание структуры программы.Матрица 10-12](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962763)

5.[Эксперименты 13-14](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962765)

6.[Заключение 15](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962766)

7.[Литература 16](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962767)

8.[Приложение 17](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962767)

8.1.[Приложение 1 18-19](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962767)

8.2.[Приложение 2 20-22](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962767)

8.3.[Приложение 3 23](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962767)

# 1. Введение

На сегодняшний день программирование играет ведущую роль в жизни.

Цель программирования – создание оптимизированных алгоритмов для решения различного рода задач, не обязательно связанных с компьютерами. Программы создаются с помощью специальных языков программирования, имеющих маленькое сходство с естественными языками. Они созданы для того, чтобы интегрировать команды человека в машинный код, то есть «от человека к компьютеру».

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.

**2. Постановка задачи**

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

Матрица должна наследоваться от Вектора.

Классы Вектора и Матрицы должны быть внесены в статическую библиотеку.

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

**3. Руководство пользователя. Вектор.**

Класс classVector<Type> является шаблонным, поэтому принимает все типы данных. Используемые типы: int,float,double.

Класс вектор по умолчанию инициализируется нулями и является трехмерным.

Варианты инициализации:



Рис.1 Варианты инициализации.

Результат инициализации:

1. a[0,0,0,0,0,0,0]

2. res[2.0,1.0]

3. a1[10,15]

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.

Перегруженные операторы ввода и вывода для Вектора:



Рис.2 Операторы ввода и вывода.

Все арифметические операции выполняются по правилам математики.

**3.1 Руководство пользователя. Матрица.**

Класс classMatrix<Type> является шаблонным, поэтому принимает все типы данных. Используемые типы: int,float,double.

Класс матриц по умолчанию инициализируется нулями и является матрицой второго ранга.

Варианты инициализации:

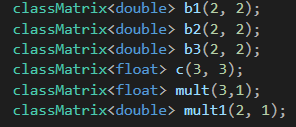


Рис.3 Инициализация матриц.

В результате инициализации получим следующее:

b1,b2,b3[0.0, 0.0]

c[0 0 0, 0 0 0, 0 0 0]

mult[0 0 0]

mult1[0 0]

Класс также 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.

Перегруженные операторы ввода и вывода для Матриц:



Рис.4 Операторы ввода и вывода.

В этом классе также все арифметические операции выполняются по правилам математики.

**4. Руководство программиста**

**4.1 Описание структуры программы. Вектор.**

Вся программа состоит из трёх подпрограмм. В первой идёт задание наследующего класса classVector<Type>, который в свою очередь является шаблонным.

Структура classVector имеет в себе защищённые поля: массив шаблона Type под названием ArrV и целое число ArrSize, обозначающее длину. В массиве ArrV находится ArrSize элементов типа Type, характеризующих вектор.

1. Создание основы шаблонного класса classVector:

|  |
| --- |
| template <class Type>  class classVector  {  protected:  Type\* ArrV;  int ArrSize = 3;  public:  Type GetArrV(int index)  {  return ArrV[index];  }  int GetSize()  {  return ArrSize;  } |

Рис.5 Фрагмент кода(1) Создание шаблона.

2) Написание конструкторов:

|  |
| --- |
| classVector() {  ArrV = new Type[ArrSize];  for (int i = 0; i < ArrSize; i++) {  ArrV[i] = 0;  }  }  classVector(int Size) {  ArrSize = Size;  ArrV = new Type[ArrSize];  for (int i = 0; i < ArrSize; i++) {  ArrV[i] = 0;  }  }  classVector(const initializer\_list<Type> &list) :classVector(list.size()) {  int count = 0;  for (auto& element : list) {  ArrV[count] = element;  ++count;  }  }  classVector(const classVector<Type>& other) {  ArrSize = other.ArrSize;  ArrV = new Type[ArrSize];  for (int i = 0; i < ArrSize; i++) {  ArrV[i] = other.ArrV[i];  }  }  ~classVector() {  delete[] ArrV;  } |

Рис.6 Фрагмент кода(2) Конструкторы.

3) Перегрузка операторов: +,-,=,/,\*,[], ==, потоковый ввод и вывод:

int& operator[](const int index) {

return ArrV[index];

};

classVector operator=(classVector<Type> other) {

if (ArrSize == other.ArrSize) {

for (int i = 0; i < ArrSize; i++) {

ArrV[i] = other.ArrV[i];

}

return \*this;

}

else {

cout << "The number of elements is not correct. Check the vector." << endl;

abort();

}

};

classVector operator+(classVector<Type> other) {

if (ArrSize == other.ArrSize) {

classVector<Type> temp(ArrSize);

for (int i = 0; i < ArrSize; i++) {

temp.ArrV[i] = ArrV[i] + other.ArrV[i];

}

return temp;

}

else {

cout << "The number of elements is not correct. Check the vector." << endl;

abort();

}

};

classVector operator-(classVector<Type> other) {

if (ArrSize == other.ArrSize) {

classVector<Type> temp(ArrSize);

for (int i = 0; i < ArrSize; i++) {

temp.ArrV[i] = ArrV[i] - other.ArrV[i];

}

return temp;

}

else {

cout << "The number of elements is not correct. Check the vector." << endl;

abort();

}

};

classVector operator\*(classVector<Type> other) {

if (ArrSize == other.ArrSize) {

classVector<Type> temp(ArrSize);

for (int i = 0; i < ArrSize; i++) {

temp.ArrV[i] = ArrV[i] \* other.ArrV[i];

}

return temp;

}

else {

cout << "The number of elements is not correct. Check the vector." << endl;

abort();

}

};

classVector operator/(classVector<Type> other) {

if (ArrSize == other.ArrSize) {

classVector<Type> temp(ArrSize);

for (int i = 0; i < ArrSize; i++) {

temp.ArrV[i] = (ArrV[i]) / (other.ArrV[i]);

}

return temp;

}

else

|  |
| --- |
| cout << "The number of elements is not correct. Check the vector." << endl;  abort();  }  };  friend ostream& operator<< (ostream& os, classVector<Type>& Input) {  for (int i = 0; i < Input.ArrSize; i++) {  os << Input.ArrV[i] << endl;  }  return os;  };  friend istream& operator>>(istream& in, classVector<Type>& Output) {  for (int i = 0; i < Output.ArrSize; i++) {  in >> Output.ArrV[i];  }  return in;  };  };  template <class Type>  bool operator==(classVector<Type>& First, classVector<Type>& Second) {  if (First.ArrSize != Second.ArrSize) {  return false;  }  for (int i = 0; i < First.ArrSize; i++) {  if (First.ArrV[i] != Second.ArrV[i]) {  return false;  }  }  return true;  }  template <class Type>  bool operator!=(classVector<Type>& First, classVector<Type>& Second)  {  return not(First == Second);  } |

Рис.7 Фрагмент кода(3) Перегрузка операторов.

**4.2 Описание структуры программы. Матрица.**

Структура classMatrix имеет в себе защищённые поля: двумерный массив ArrM шаблона Type и два поля целых чисел отвечающих за размерность матрицы ArrSize\_x и ArrSize\_y . В поле ArrM находятся массивы которые содержат ArrSize\_y элементов типа Type.

Изначальная идея: наследник шаблонного класса classVector от classVector от шаблона.

Начальная часть класса:

|  |
| --- |
| template <class Type>  class classMatrix :public classVector<Type> {  protected:  Type\*\* ArrM;  int ArrSize\_x = 2;  int ArrSize\_y = 2; |

Рис.8 Фрагмент кода(4) Создание массива и полей.

Далее идут конструкторы (в их числе конструктор копирования от classVector<classVector<Type>>, где Type – шаблон):

|  |
| --- |
| classMatrix() {  ArrM = new Type \* [ArrSize\_x];  for (int i = 0; i < ArrSize\_x; i++) {  ArrM[i] = new Type[ArrSize\_y];  }  for (int i = 0; i < ArrSize\_x; i++) {  for (int j = 0; j < ArrSize\_y; j++) {  ArrM[i][j] = 0;  }  }  }  classMatrix(int x, int y) {  ArrSize\_x = x;  ArrSize\_y = y;  ArrM = new Type \* [ArrSize\_x];  for (int i = 0; i < ArrSize\_x; i++) {  ArrM[i] = new Type[ArrSize\_y];  }  for (int i = 0; i < ArrSize\_x; i++) {  for (int j = 0; j < ArrSize\_y; j++) {  ArrM[i][j] = 0;  }  }  }  classMatrix(const classMatrix<Type>& other) {  ArrM = new Type \* [other.ArrSize\_x];  for (int i = 0; i < other.ArrSize\_x; i++) {  ArrM[i] = new Type[other.ArrSize\_y];  }  for (int i = 0; i < other.ArrSize\_x; i++) {  for (int j = 0; j < other.ArrSize\_y; j++) {  ArrM[i][j] = other.ArrM[i][j];  }  }  }  ~classMatrix() {  for (int i = 0; i < ArrSize\_x; i++) {  delete[] ArrM[i];  }  delete[] ArrM;} |

Затем следуют перегрузки операторов(+,-,\*,/,=,==,[], потоковый ввод и вывод):

|  |  |
| --- | --- |
| auto operator[](const int index) {  return ArrM[index];  };  classMatrix& operator = (classMatrix<Type> other) {  for (int i = 0; i < ArrSize\_x; i++) {  for (int j = 0; j < ArrSize\_y; j++) {  ArrM[i][j] = other.ArrM[i][j];  }  }  return \*this;  };  classMatrix operator+(classMatrix<Type>& other) {  if ((ArrSize\_x == other.ArrSize\_x) && (ArrSize\_y == other.ArrSize\_y)) {  classMatrix<Type> temp(ArrSize\_x, ArrSize\_y);  for (int i = 0; i < ArrSize\_x; i++) {  for (int j = 0; j < ArrSize\_y; j++) {  temp.ArrM[i][j] = ArrM[i][j] + other.ArrM[i][j];  }  }  return temp;  }  else {  cout << "The number of rows does not match the number of columns in the matrices. Check the matrices." << endl;  abort();  }  };  classMatrix operator-(classMatrix<Type>& other) {  if ((ArrSize\_x == other.ArrSize\_x) && (ArrSize\_y == other.ArrSize\_y)) {  classMatrix<Type> temp(ArrSize\_x, ArrSize\_y);  for (int i = 0; i < ArrSize\_x; i++) {  for (int j = 0; j < ArrSize\_y; j++) {  temp.ArrM[i][j] = ArrM[i][j] - other.ArrM[i][j];  }  }  return temp;  }  else {  cout << "The number of rows does not match the number of columns in the matrices. Check the matrices." << endl;  abort();  }  };  classMatrix operator\*(classMatrix<Type>& other) {  if (ArrSize\_y == other.ArrSize\_x) {  classMatrix<Type> temp(ArrSize\_x, other.ArrSize\_y);  for (int i = 0; i < ArrSize\_x; i++) {  for (int j = 0; j < other.ArrSize\_x; j++) {  for (int k = 0; k < ArrSize\_y; k++) {  temp.ArrM[i][j] += ArrM[i][k] \* other.ArrM[k][j];  }  }  }  return temp;  }  else {  cout << "The number of rows does not match the number of columns in the matrices. Check the matrices." << endl;  abort();  }  }; | |
| classMatrix operator\*(classVector<Type>& other) {  if (ArrSize\_y == other.GetSize()) {  classMatrix<Type> temp(other.GetSize(), 1);  for (int i = 0; i < temp.ArrSize\_x; i++) {  for (int j = 0; j < 1; j++) {  for (int k = 0; k < other.GetSize(); k++) {  temp.ArrM[i][j] += ArrM[i][k] \* other.GetArrV(k);  }  }  }  cout << temp << endl;  return temp;  }  else {  cout << "The number of rows does not match the number of columns in the matrices. Check the matrices." << endl;  abort();  }  };  friend ostream& operator<< (ostream& os, classMatrix<Type>& Input) {  for (int i = 0; i < Input.ArrSize\_x; i++) {  for (int j = 0; j < Input.ArrSize\_y; j++) {  os << Input.ArrM[i][j] << " ";  }  os << endl;  }  return os;  };  friend istream& operator>>(istream& in, classMatrix<Type>& Output) {  for (int i = 0; i < Output.ArrSize\_x; i++) {  for (int j = 0; j < Output.ArrSize\_y; j++) {  in >> Output.ArrM[i][j];  }  }  return in;  };  };  template <class Type>  bool operator==(classMatrix<Type>& First, classMatrix<Type>& Second) {  if ((First.ArrSize\_x != Second.ArrSize\_x) || (First.ArrSize\_y != Second.ArrSize\_y)) {  return false;  }  for (int i = 0; i < First.ArrSize\_x; i++) {  for (int j = 0; j < First.ArrSize\_y; j++) {  if (First.ArrM[i][j] != Second.ArrM[i][j]) {  return false;  }  }  }  return true;  }  template <class Type>  bool operator!=(classMatrix<Type>& First, classMatrix<Type>& Second) {  return not(First == Second);  } | |

Рис.9 Фрагмент кода(6) Перегрузки операторов.

**5. Эксперименты.**

1) Умножение матрицы 2х2 на матрицу 2х2.

Умножим матрицу [1, 2] на матрицу [5, 6]

[3, 4] [7, 8]

В ответе должна получиться матрица [19, 22]

[43, 50]

Смотрим на результат работы программы:

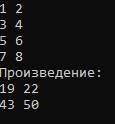


Рис.10 Результат умножения.

Программа успешно и правильно посчитала исходные данные.

2) Умножение матрицы 3х3 на вектор 3х1.

Умножим матрицу [1, 2, 3] на вектор [10, 11, 12]

[4, 5, 6]

[7, 8, 9]

В ответе должен получиться вектор-столбец 3х1 [68, 167, 266].

Результат работы программы:

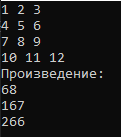


Рис.11 Результат умножения.

3) Умножение матрицы 2х2 на вектор 2х1.

Умножим матрицу [10, 20] на вектор [50.1, 60.2]

[30, 40]

В результате должен получиться вектор-столбец 2х1 [1705, 3911]

Результат работы программы:

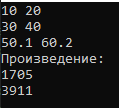


Рис.12 Результат умножения.

4) Сравним время для операции сложения. При размерах матриц 1000x1000 время равно 0.126 сек. А при размерах 10000x10000 6.274 сек. Так как сложность алгоритма сложения равна o(n^2), то при увеличении данных в 10 раз, получаем увеличение времени в 100 раз. Мы получили больше.

https://sun9-81.userapi.com/s/v1/ig2/9yHX4qQIjCxeorSKAJVJSrFyYXES-c9zOPyCL2PwNc6PdBMGClCEy506uGLJDq9QMD8PYqEly0ZeU2vHXIFjJfxJ.jpg?size=353x62&quality=96&type=album https://sun9-83.userapi.com/s/v1/ig2/3xFfuYRiuoTxZREiGRg_1NkJMJxylTCBTBi3mHkDmcYtiklqWFYgNEN9_xdCVujDKqPzl_IwaVJOsDmNCNlsiYPO.jpg?size=303x63&quality=96&type=album

Рис.13 Результат сложения №1 Рис.14 Результат сложения №2

5) Сравним время для операции умножения. При размерах матриц 100x100 время будет такое. А при размерах 10000x10000 другое. Так как сложность алгоритма сложения равна o(n^3), то при увеличении данных в 10 раз, получаем увеличение времени в 1000 раз. Мы получили немного меньше:

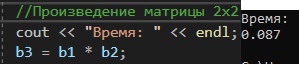
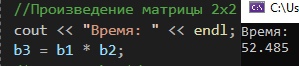
 

Рис.15 Результат умножения №1 Рис.16 Результат умножения №2

**6. Заключение**

В ходе лабораторной работы была написана статическая библиотека С++, которая реализует вектор (класс classVector) и матрицу (класс classMatrix). 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++.

Данная лабораторная работа помогла мне лучше понять суть шаблонов. Они действительно в разы упрощают написание кода программистам. Научился находить более эффективные методы реализации для тех или иных функций, да и в целом лучше понимать язык С++.

**7. Список литературы**

1. Т.А. Павловская Учебник по программированию на языках высокого  
уровня(С/С++) – Режим доступа: <http://cph.phys.spbu.ru/documents/First/books/7.pdf>  
2. Бьерн Страуструп. Язык программирования С++ - Режим доступа:  
<http://8361.ru/6sem/books/Straustrup-Yazyk_programmirovaniya_c.pdf>

**8. Приложения**

***8.1 Приложение 1(Вектор)***

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

**8. Приложения**

***8.2 Приложение 2(Матрица)***

|  |  |
| --- | --- |
| #pragma once  #include "ClassVector.h"  #include<iostream>  using namespace std;  template <class Type>  class classMatrix :public classVector<Type> {  protected:  Type\*\* ArrM;  int ArrSize\_x = 2;  int ArrSize\_y = 2;  public:  //Конструкторы: по умолчанию, инициализации, копирования, доступ к защ. полям, деструктор.  classMatrix() {  ArrM = new Type \* [ArrSize\_x];  for (int i = 0; i < ArrSize\_x; i++) {  ArrM[i] = new Type[ArrSize\_y];  }  for (int i = 0; i < ArrSize\_x; i++) {  for (int j = 0; j < ArrSize\_y; j++) {  ArrM[i][j] = 0;  }  }  }  classMatrix(int x, int y) {  ArrSize\_x = x;  ArrSize\_y = y;  ArrM = new Type \* [ArrSize\_x];  for (int i = 0; i < ArrSize\_x; i++) {  ArrM[i] = new Type[ArrSize\_y];  }  for (int i = 0; i < ArrSize\_x; i++) {  for (int j = 0; j < ArrSize\_y; j++) {  ArrM[i][j] = 0;  }  }  }  classMatrix(const classMatrix<Type>& other) {  ArrM = new Type \* [other.ArrSize\_x];  for (int i = 0; i < other.ArrSize\_x; i++) {  ArrM[i] = new Type[other.ArrSize\_y];  }  for (int i = 0; i < other.ArrSize\_x; i++) {  for (int j = 0; j < other.ArrSize\_y; j++) {  ArrM[i][j] = other.ArrM[i][j];  }  }  }  ~classMatrix() {  for (int i = 0; i < ArrSize\_x; i++) {  delete[] ArrM[i];  }  delete[] ArrM;  }  //Перегрузки операторов  auto operator[](const int index) {  return ArrM[index]; | |
| };  classMatrix& operator = (classMatrix<Type> other) {  for (int i = 0; i < ArrSize\_x; i++) {  for (int j = 0; j < ArrSize\_y; j++) {  ArrM[i][j] = other.ArrM[i][j];  }  }  return \*this;  };  classMatrix operator+(classMatrix<Type>& other) {  if ((ArrSize\_x == other.ArrSize\_x) && (ArrSize\_y == other.ArrSize\_y)) {  classMatrix<Type> temp(ArrSize\_x, ArrSize\_y);  for (int i = 0; i < ArrSize\_x; i++) {  for (int j = 0; j < ArrSize\_y; j++) {  temp.ArrM[i][j] = ArrM[i][j] + other.ArrM[i][j];  }  }  return temp;  }  else {  cout << "The number of rows does not match the number of columns in the matrices. Check the matrices." << endl;  abort();  }  };  classMatrix operator-(classMatrix<Type>& other) {  if ((ArrSize\_x == other.ArrSize\_x) && (ArrSize\_y == other.ArrSize\_y)) {  classMatrix<Type> temp(ArrSize\_x, ArrSize\_y);  for (int i = 0; i < ArrSize\_x; i++) {  for (int j = 0; j < ArrSize\_y; j++) {  temp.ArrM[i][j] = ArrM[i][j] - other.ArrM[i][j];  }  }  return temp;  }  else {  cout << "The number of rows does not match the number of columns in the matrices. Check the matrices." << endl;  abort();  }  };  classMatrix operator\*(classMatrix<Type>& other) {  if (ArrSize\_y == other.ArrSize\_x) {  classMatrix<Type> temp(ArrSize\_x, other.ArrSize\_y);  for (int i = 0; i < ArrSize\_x; i++) {  for (int j = 0; j < other.ArrSize\_x; j++) {  for (int k = 0; k < ArrSize\_y; k++) {  temp.ArrM[i][j] += ArrM[i][k] \* other.ArrM[k][j];  }  }  }  return temp;  }  else {  cout << "The number of rows does not match the number of columns in the matrices. Check the matrices." << endl;  abort();  }  };  classMatrix operator\*(classVector<Type>& other) {  if (ArrSize\_y == other.GetSize()) {  classMatrix<Type> temp(other.GetSize(), 1); | |
| for (int i = 0; i < temp.ArrSize\_x; i++) {  for (int j = 0; j < 1; j++) {  for (int k = 0; k < other.GetSize(); k++) {  temp.ArrM[i][j] += ArrM[i][k] \* other.GetArrV(k);  }  }  }  cout << temp << endl;  return temp;  }  else {  cout << "The number of rows does not match the number of columns in the matrices. Check the matrices." << endl;  abort();  }  };  friend ostream& operator<< (ostream& os, classMatrix<Type>& Input) {  for (int i = 0; i < Input.ArrSize\_x; i++) {  for (int j = 0; j < Input.ArrSize\_y; j++) {  os << Input.ArrM[i][j] << " ";  }  os << endl;  }  return os;  };  friend istream& operator>>(istream& in, classMatrix<Type>& Output) {  for (int i = 0; i < Output.ArrSize\_x; i++) {  for (int j = 0; j < Output.ArrSize\_y; j++) {  in >> Output.ArrM[i][j];  }  }  return in;  };  };  template <class Type>  bool operator==(classMatrix<Type>& First, classMatrix<Type>& Second) {  if ((First.ArrSize\_x != Second.ArrSize\_x) || (First.ArrSize\_y != Second.ArrSize\_y)) {  return false;  }  for (int i = 0; i < First.ArrSize\_x; i++) {  for (int j = 0; j < First.ArrSize\_y; j++) {  if (First.ArrM[i][j] != Second.ArrM[i][j]) {  return false;  }  }  }  return true;  }  template <class Type>  bool operator!=(classMatrix<Type>& First, classMatrix<Type>& Second) {  return not(First == Second); } | | |

**8. Приложения**

***8.3 Приложение 3(main.cpp)***

|  |
| --- |
| #include"ClassMatrix.h"  #include"ClassVector.h"  #include<iostream>  using namespace std;  int main()  {  setlocale(LC\_ALL,"ru");  classVector<float> a{7, 10, 22.0};  classVector<double> res{2,1};  classVector<int> a1{10,15};  classMatrix<double> b1(2, 2);  classMatrix<double> b2(2, 2);  classMatrix<double> b3(2, 2);  classMatrix<float> c(3, 3);  classMatrix<float> mult(3,1);  classMatrix<double> mult1(2, 1);  //Произведение матрицы 2х2 на матрицу 2х2  /\*cin >> b1;  cin >> b2;  cout << "Произведение: " << endl;  b3 = b1 \* b2;  cout << b3;\*/  //Произведение матрицы 3х3 на вектор 3х1  //cin >> c;  //cin >> a;  //cout << "Произведение: " << endl;  //mult = c \* a;  //Произведение матрицы 2х2 на вектор 2х1  cin >> b1;  cin >> res;  cout << "Произведение: " << endl;  mult1 = b1 \* res;  return 0;  } |