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

Манаев 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](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962765)

6.[Заключение 14](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962766)

7.[Литература 15](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962767)

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

8.1.[Приложение 1 17-18](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962767)

8.2.[Приложение 2 19-21](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962767)

8.3.[Приложение 3 22](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 TVector<T> 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. v1[0,0,0]

2. v2[3,3,3]

3. v3[2,2,2]

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 TMatrix<T> 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:

m1[3,3,3 ; 3,3,3 ; 3, 3,3]

m2[2,2,2; 2,2,2; 2,2,2]

m3[5,5,5 ; 5,5,5 ; 5,5,5]

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.**

1. Создание основы шаблонного класса TVector:

template <class T>

class TVector

{

protected:

T\* data;

int len = 2;

public:

T GetData(int i)

{

return data[i];

}

int GetLen()

{

return len;

}

2) Написание конструкторов:

TVector()

{

data = new T[len];

for (int i = 0; i < len; i++)

{

data[i] = 0;

}

}

TVector(int \_len)

{

len = \_len;

data = new T[len];

for (int i = 0; i < len; i++)

{

data[i] = 0;

}

}

TVector(const initializer\_list<T>& list) :TVector(list.size())

{

int count = 0;

for (auto& element : list)

{

data[count] = element;

++count;

}

}

TVector(const TVector<T>& p)

{

len = p.len;

data = new T[len];

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

data[i] = p.data[i];

}

}

~TVector()

{

delete[] data;

data = 0;

}

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

int& operator[](const int i)

{

return data[i];

};

TVector operator=(TVector<T> p)

{

if (len == p.len)

{

for (int i = 0; i < len; i++)

{

data[i] = p.data[i];

}

return \*this;

}

else

{

cout << "Error." << endl;

abort();

}

};

TVector operator+(TVector<T> p)

{

if (len == p.len)

{

TVector<T> tmp(len);

for (int i = 0; i < len; i++)

{

tmp.data[i] = data[i] + p.data[i];

}

return tmp;

}

else

{

cout << "Error." << endl;

abort();

}

};

TVector operator-(TVector<T> p)

{

if (len == p.len)

{

TVector<T> tmp(len);

for (int i = 0; i < len; i++)

{

tmp.data[i] = data[i] - p.data[i];

}

return tmp;

}

else

{

cout << "Error." << endl;

abort();

}

};

TVector operator\*(TVector<T> p)

{

if (len == p.len)

{

TVector<T> tmp(len);

for (int i = 0; i < len; i++)

{

tmp.data[i] = data[i] \* p.data[i];

}

return tmp;

}

else {

cout << "Error." << endl;

abort();

}

};

TVector operator/(TVector<T> p)

{

if (len == p.len)

{

TVector<T> tmp(len);

for (int i = 0; i < len; i++)

{

tmp.data[i] = (data[i]) / (p.data[i]);

}

return tmp;

}

else

{

cout << "Error." << endl;

abort();

}

};

friend ostream& operator<< (ostream& os, TVector<T>& p)

{

for (int i = 0; i < p.len; i++)

{

os << p.data[i] << endl;

}

return os;

};

friend istream& operator>>(istream& in, TVector<T>& p)

{

for (int i = 0; i < p.len; i++)

{

in >> p.data[i];

}

return in;

};

template <class T>

bool operator==(TVector<T> &Second)

{

if (this->len != Second.len)

{

return false;

}

for (int i = 0; i < this->len; i++)

{

if (this->data[i] != Second.data[i])

{

return false;

}

}

return true;

}

**4.2 Описание структуры программы. Матрица.**

1. Создание основы шаблонного класса TMatrix:

template <class T>

class TMatrix : public TVector<T>

{

protected:

T\*\* dataM;

int len = 2;

int weidth = 2;

2) Написание конструкторов:

public:

TMatrix()

{

dataM = new T \* [len];

for (int i = 0; i < len; i++)

{

dataM[i] = new T[weidth];

}

for (int i = 0; i < len; i++)

{

for (int j = 0; j < weidth; j++)

{

dataM[i][j] = 0;

}

}

}

TMatrix(int x, int y, T v)

{

len = x;

weidth = y;

dataM = new T \* [len];

for (int i = 0; i < len; i++)

{

dataM[i] = new T[weidth];

}

for (int i = 0; i < len; i++)

{

for (int j = 0; j < weidth; j++)

{

dataM[i][j] = v;

}

}

}

TMatrix(const TMatrix<T>& p)

{

dataM = new T \* [p.len];

for (int i = 0; i < p.len; i++)

{

dataM[i] = new T[p.weidth];

}

for (int i = 0; i < p.len; i++)

{

for (int j = 0; j < p.weidth; j++)

{

dataM[i][j] = p.dataM[i][j];

}

}

}

~TMatrix() {

for (int i = 0; i < len; i++)

{

delete[] dataM[i];

}

delete[] dataM;

}

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

auto operator[](const int i)

{

return dataM[i];

};

TMatrix& operator = (TMatrix<T> p)

{

for (int i = 0; i < len; i++)

{

for (int j = 0; j < weidth; j++)

{

dataM[i][j] = p.dataM[i][j];

}

}

return \*this;

};

TMatrix operator+(TMatrix<T>& p)

{

if ((len == p.len) && (weidth == p.weidth))

{

TMatrix<T> tmp(len, weidth, 0);

for (int i = 0; i < len; i++)

{

for (int j = 0; j < weidth; j++)

{

tmp.dataM[i][j] = dataM[i][j] + p.dataM[i][j];

}

}

return tmp;

}

else

{

cout << "Error." << endl;

abort();

}

};

TMatrix operator-(TMatrix<T>& p)

{

if ((len == p.len) && (weidth == p.weidth))

{

TMatrix<T> tmp(len, weidth, 0);

for (int i = 0; i < len; i++)

{

for (int j = 0; j < weidth; j++)

{

tmp.dataM[i][j] = dataM[i][j] - p.dataM[i][j];

}

}

return tmp;

}

else

{

cout << "Error." << endl;

abort();

}

};

TMatrix operator\*(TMatrix<T>& p)

{

if (weidth == p.len)

{

TMatrix<T> tmp(len, p.weidth, 0);

for (int i = 0; i < len; i++)

{

for (int j = 0; j < p.len; j++)

{

for (int k = 0; k < weidth; k++)

{

tmp.dataM[i][j] += dataM[i][k] \* p.dataM[k][j];

}

}

}

return tmp;

}

else

{

cout << "Error." << endl;

abort();

}

};

TMatrix operator\*(TVector<T>& p)

{

if (weidth == p.GetLen())

{

TMatrix<T> tmp(p.GetLen(), 1, 0);

for (int i = 0; i < tmp.len; i++)

{

for (int j = 0; j < 1; j++)

{

for (int k = 0; k < p.GetLen(); k++)

{

tmp.dataM[i][j] += dataM[i][k] \* p.GetData(k);

}

}

}

cout << tmp << endl;

return tmp;

}

else

{

cout << "Error." << endl;

abort();

}

};

friend ostream& operator<< (ostream& os, TMatrix<T>& p)

{

for (int i = 0; i < p.len; i++)

{

for (int j = 0; j < p.weidth; j++)

{

os << p.dataM[i][j] << " ";

}

os << endl;

}

return os;

};

friend istream& operator>>(istream& in, TMatrix<T>& p)

{

for (int i = 0; i < p.len; i++)

{

for (int j = 0; j < p.weidth; j++)

{

in >> p.dataM[i][j];

}

}

return in;

};

template <class T>

bool operator==(TMatrix<T>& Second)

{

if ((this->len != Second.len) || (this->weidth != Second.weidth))

{

return false;

}

for (int i = 0; i < this->len; i++)

{

for (int j = 0; j < this->weidth; j++)

{

if (this->dataM[i][j] != Second.dataM[i][j])

{

return false;

}

}

}

return true;

}

**5. 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++.

**6. 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>