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

Мезенцев Д.А

Проверил:

Заведующий лабораторией

суперкомпьютерных технологий и

высокопроизводительных

вычислений

Лебедев И. Г.

Нижний Новгород

2022 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)

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

4.1.[Описание структуры программы.Вектор 7-11](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962762)

4.2.[Описание структуры программы.Матрица 12-16](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962763)

5.[Эксперименты 17](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962765)

6.[Заключение 18](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962766)

7.[Литература 19](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962767)

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

8.1.[Приложение 1 21-27](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962767)

8.2.[Приложение 2 28-34](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962767)

8.3.[Приложение 3 35](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962767)

# 1. Введение

Программирование играет огромную роль во всех сферах человеческой жизни, помогает создавать программы для работы, как дома, так и для решения вычислительных, бухгалтерских, графических и других задач.

Для того чтобы быть востребованным программистом нужно постоянно совершенствоваться: изучать новые языки программирования, методы и алгоритмы.

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.**

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.

Перегрузки операторов:

1. Сложение
   1. Сложение векторов



*Рисунок 1.1 Операторы сложения векторов*

* 1. Сложение матриц



*Рисунок 1.2 Операторы сложения матриц*

1. Вычитание

2.1 Вычитание векторов



*Рисунок 2.1 Операторы вычитания векторов*

2.2 Вычитание матриц



*Рисунок 2.2 Операторы вычитания матриц*

1. Умножение

3.1 Умножение векторов



*Рисунок 3.1 Операция умножения вектора на вектор*

3.2 Умножение матриц



*Рисунок 3.2 Операция умножения матрицы на матрицы*

3.3 Умножение матрицы на вектор



*Рисунок 3.3 Операция умножения матрицы на вектор*

1. Деление

4.1 Деление векторов



*Рисунок 4. Оператор деления векторов*

1. Присваивание

5.1 Присваивание векторов



*Рисунок 5.1 Оператор присваивания векторов*

5.2 Присваивание матриц



*Рисунок 5.2 Оператор присваивания матриц*

1. Равенство

6.1 Равенство векторов



*Рисунок 6.1 Оператор равенства векторов*

6.2 Равенство матриц



*Рисунок 6.2 Оператор равенства матриц*

1. Ввод и вывод

7.1 Ввод и вывод вектора



*Рисунок 7.1 Потоковый ввод для векторов*



*Рисунок 7.2 Потоковый вывод для векторов*

7.2 Ввод и вывод матрицы



*Рисунок 7.3 Потоковый ввод для матриц*



*Рисунок 7.4 Потоковый вывод для матриц*

1. Оператор индексации []

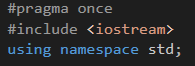


*Рисунок 8. Оператор индексации вектора*

**4. Руководство программиста**

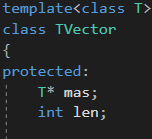
**4.1 Описание структуры программы. Вектор.**

1. Подключение библиотек и пространства имен (см. рис. 9.1)



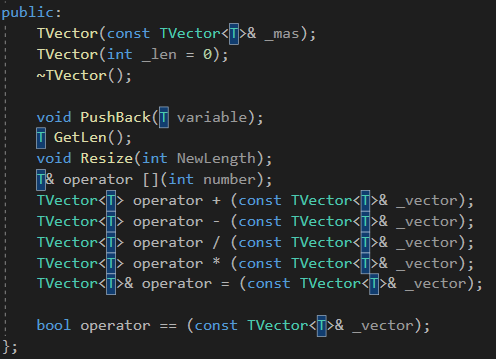
*Рисунок 9.1*

1. Объявление класса векторов и их защищенных полей (длина и массив данных). Для инициализации используем шаблон класса. (рис. 9.2)



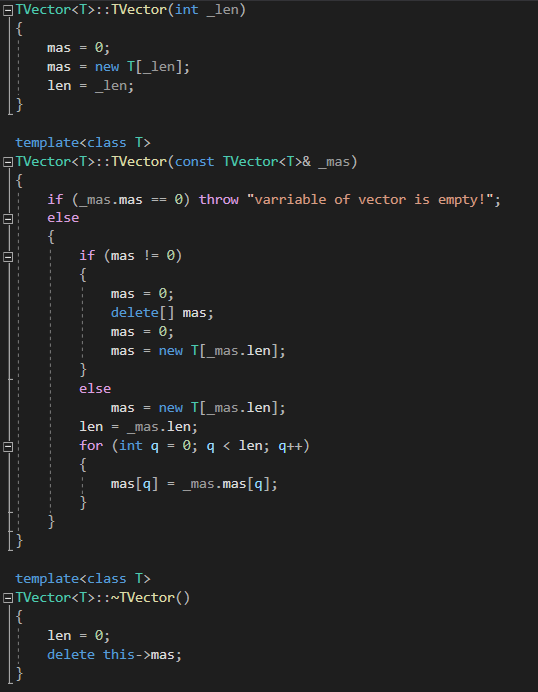
*Рисунок 9.1*

1. Объявляем в секции public конструктор по умолчанию, деструктор, конструктор копирования, методы класса и перегрузку операторов.(рис. 9.3)



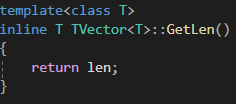
*Рисунок 9.3*

1. Реализация конструкторов (по умолчанию и создания объекта класса, копирования), деструктора.(рис. 9.4)



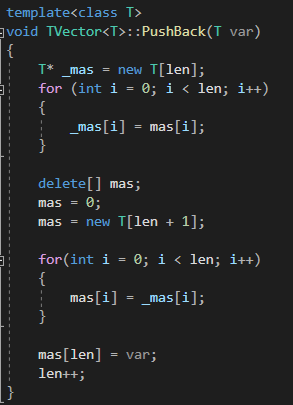
*Рисунок 9.4*

1. Реализация метода дающего доступ к защищённому полю длины вектора. (рис. 9.5)



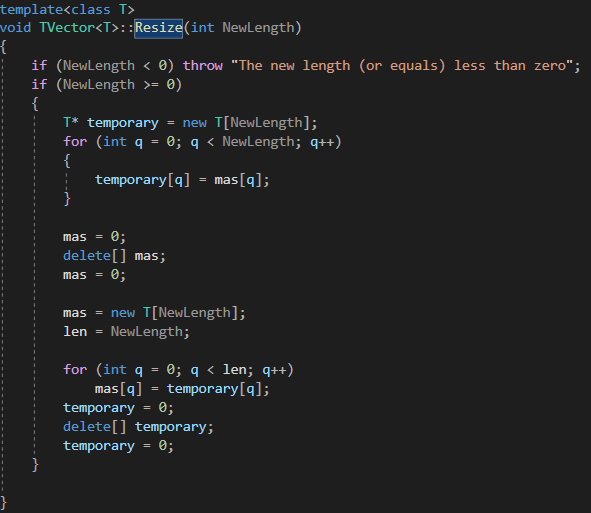
*Рисунок 9.5*

1. Реализация метода позволяющего добавить значение в конец вектора. (рис. 9.6)



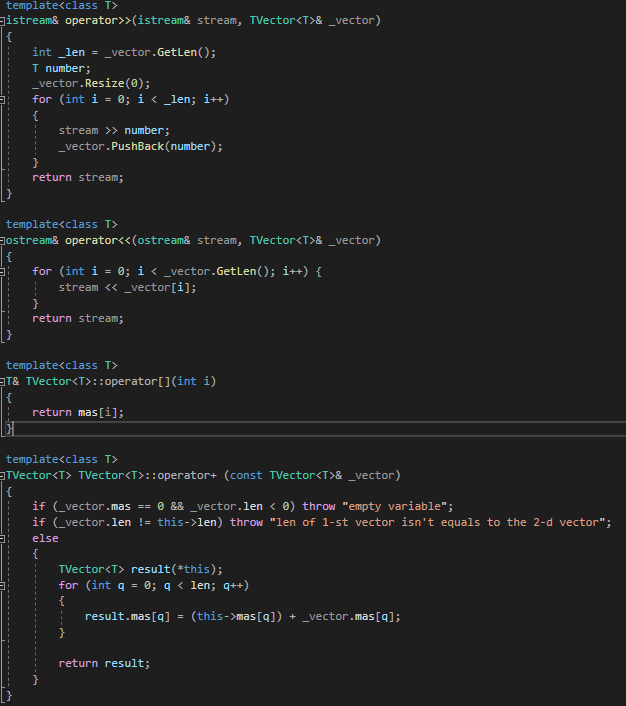
*Рисунок 9.6*

1. Реализация метода позволяющего менять размер вектора. (рис. 9.7)

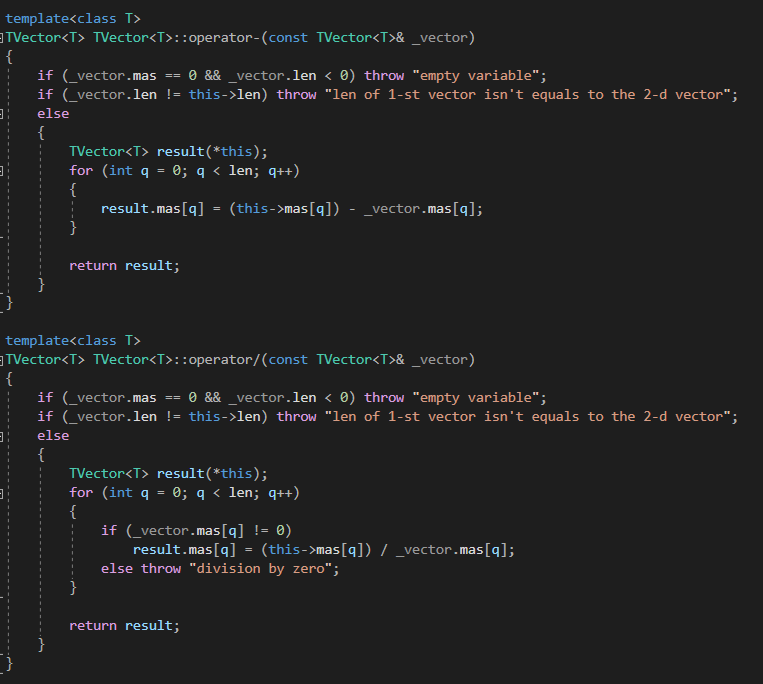


*Рисунок 9.7*

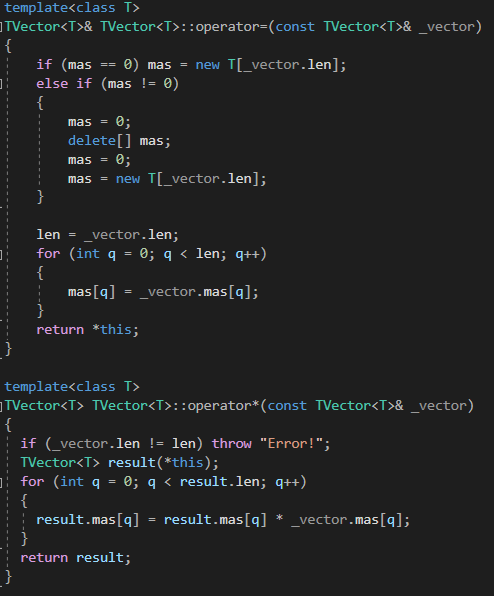
1. Реализация всех перегрузок для класса вектор (ввод, вывод, +, -, \*, /, равенства, присваивания, индексации) (рис. 9.8 – 9.11)



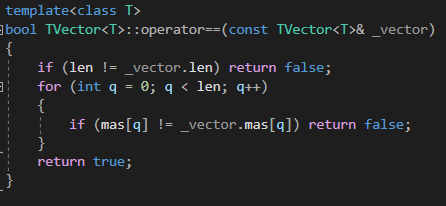
*Рисунок 9.8*



*Рисунок 9.9*



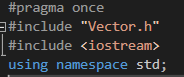
*Рисунок 9.10*



*Рисунок 9.11*

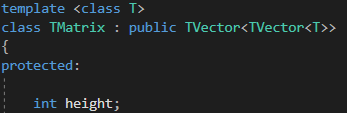
**4.2 Описание структуры программы. Матрица.**

1. Подключение библиотек и пространства имен (см. рис. 10.1)



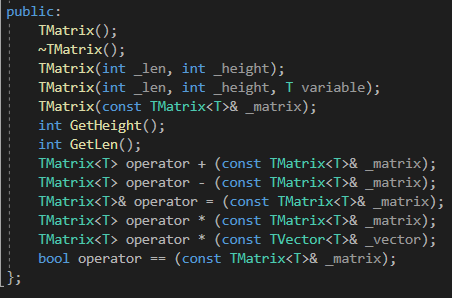
*Рисунок 10.1*

1. Объявление класса матриц как публичного наследника класса векторов. Также в данном блоке в классе матриц добавляется защищенное поле. (рис. 10.2)



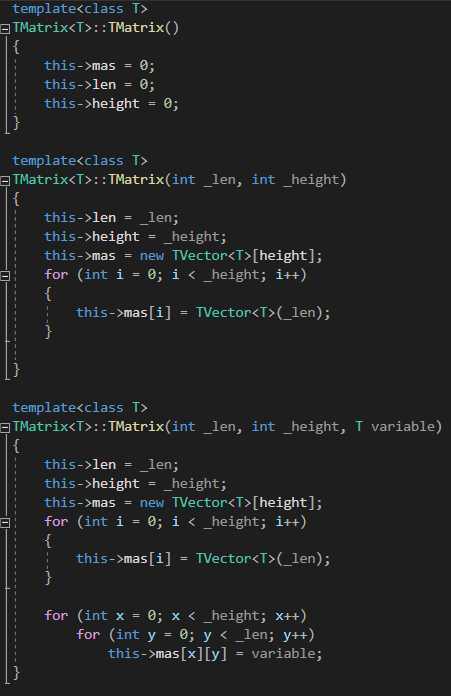
*Рисунок 10.2*

1. Объявляем в секции public конструктор по умолчанию, деструктор, конструктор копирования, методы класса и перегрузку операторов.(рис. 10.3)

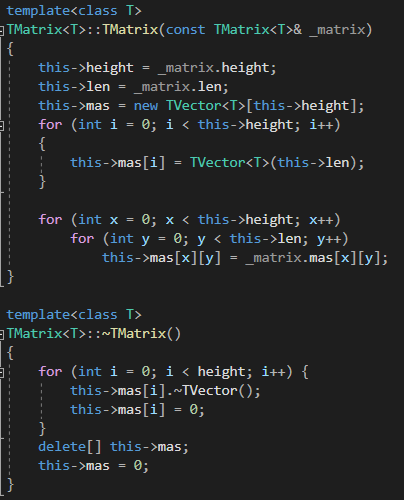


*Рисунок 10.3*

1. Реализация конструкторов (по умолчанию и создания объекта класса, копирования), деструктора.(рис. 10.4 – 10.5)

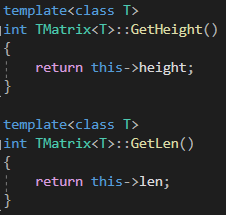


*Рисунок 10.4*



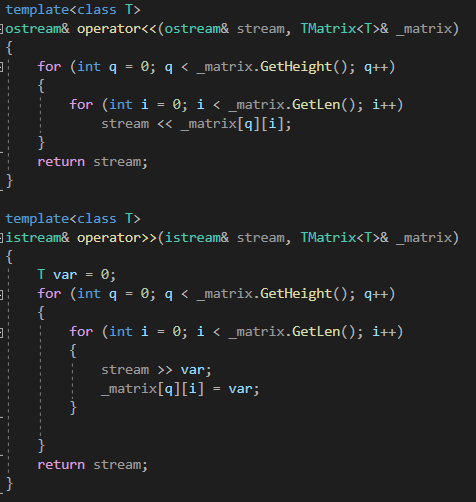
*Рисунок 10.5*

1. Реализация метода дающего доступ к защищённому полю длины и ширины матрицы. (рис. 10.5)

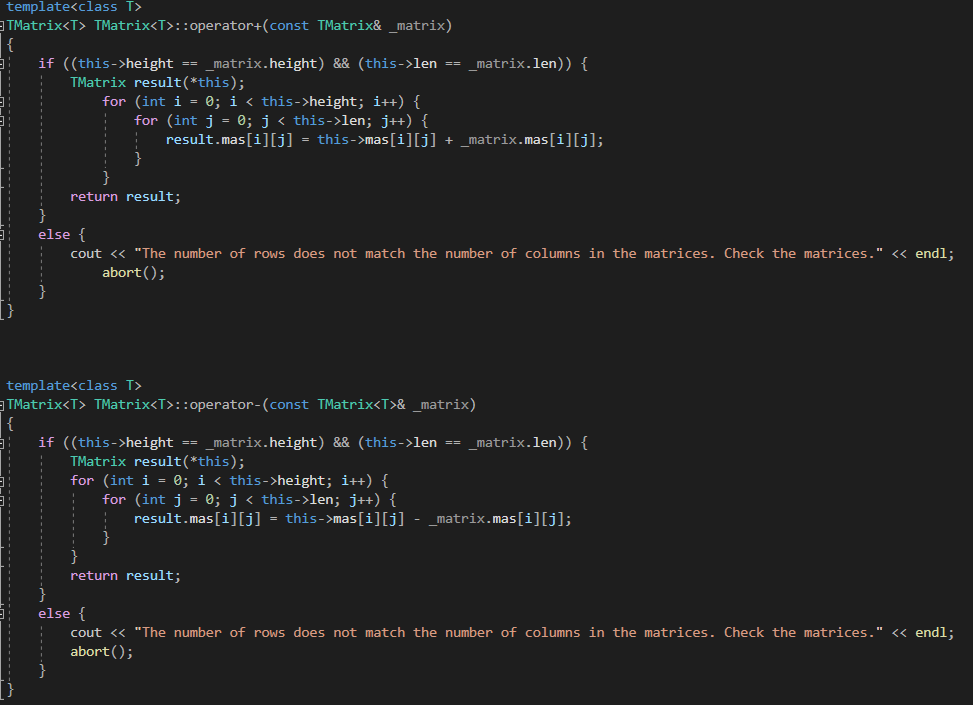


*Рисунок 10.6*

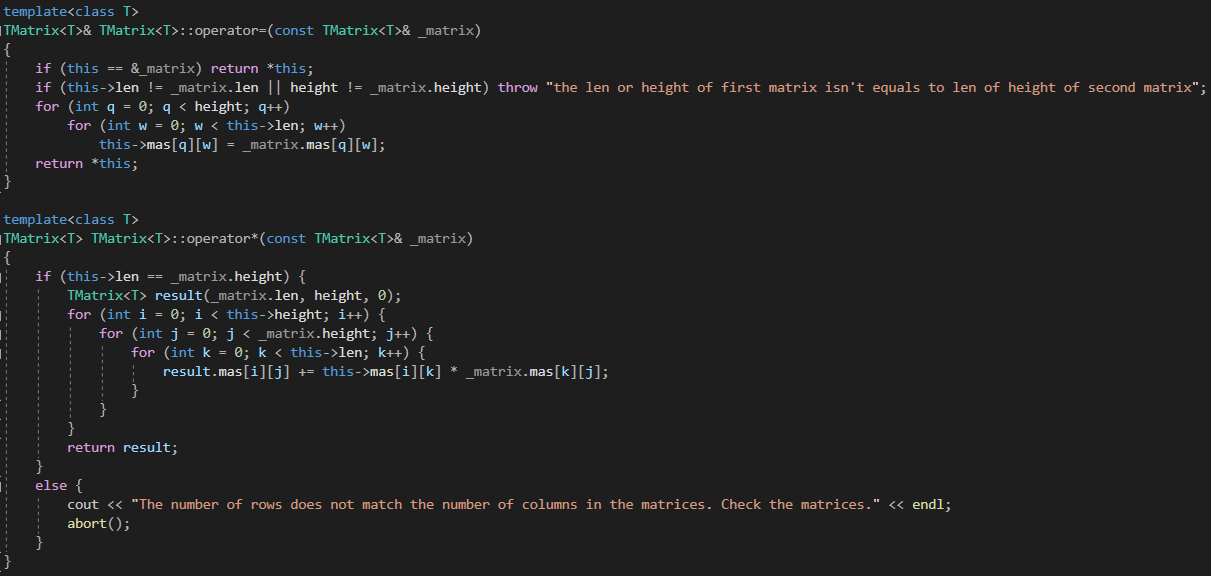
1. Реализация всех перегрузок для класса матриц (ввод, вывод, +, -, \* (матрица на матрицу и матрица на вектор), /, равенства, присваивания) (рис. 10.7 – 10.11)



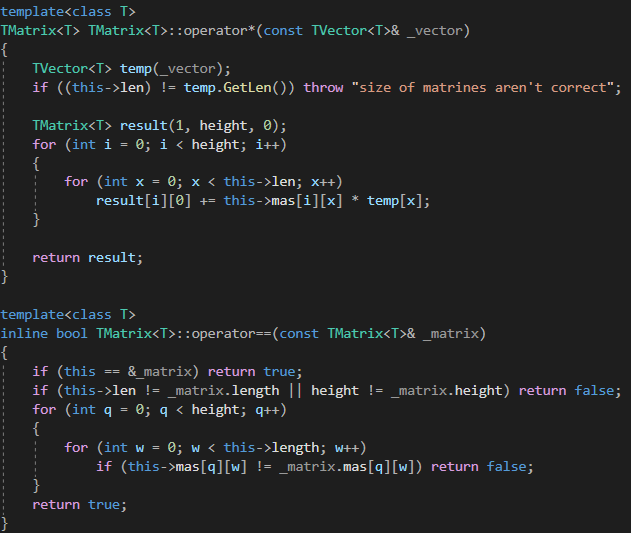
*Рисунок 10.7*



*Рисунок 10.8*



*Рисунок 10.9*



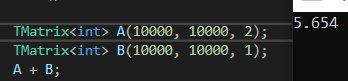
*Рисунок 10.10*

**5. Эксперименты.**

1) Сравним время для операции сложения. При размерах матриц 1000x1000 время равно 0.062 сек. (см. рис. 11.1). А при размерах 10000x10000 5.654 сек. (см. рис. 11.2) Так как сложность алгоритма сложения равна o(n^2), то при увеличении данных в 10 раз, получаем увеличение времени в 100 раз. Мы получили немного меньше.



*Рисунок 11.1*

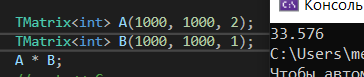


*Рисунок 11.2*

2) Сравним время для операции умножения. При размерах матриц 100x100 время будет такое (см. рис. 11.3). А при размерах 10000x10000 такое. (см. рис. 11.4) Так как сложность алгоритма сложения равна o(n^3), то при увеличении данных в 10 раз, получаем увеличение времени в 1000 раз. Что мы и получили.



*Рисунок 11.3*



*Рисунок 11.4*

Ожидания по времени работы программы оправдались.

**6. Заключение**

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. Подобные задания с программами важны, потому что они улучшают понимание работы с компьютером и средой разработки, а так же помогают закрепить уже ранее изученный материал.

**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. Приложение**

***8.1 Приложение 1(Вектор)***

#pragma once

#include <iostream>

using namespace std;

template<class T>

class TVector

{

protected:

T\* mas;

int len;

public:

TVector(const TVector<T>& \_mas);

TVector(int \_len = 0);

~TVector();

void PushBack(T variable);

T GetLen();

void Resize(int NewLength);

T& operator [](int number);

TVector<T> operator + (const TVector<T>& \_vector);

TVector<T> operator - (const TVector<T>& \_vector);

TVector<T> operator / (const TVector<T>& \_vector);

TVector<T> operator \* (const TVector<T>& \_vector);

TVector<T>& operator = (const TVector<T>& \_vector);

bool operator == (const TVector<T>& \_vector);

};

template<class T>

TVector<T>::TVector(int \_len)

{

mas = 0;

mas = new T[\_len];

len = \_len;

}

template<class T>

TVector<T>::TVector(const TVector<T>& \_mas)

{

if (\_mas.mas == 0) throw "varriable of vector is empty!";

else

{

if (mas != 0)

{

mas = 0;

delete[] mas;

mas = 0;

mas = new T[\_mas.len];

}

else

mas = new T[\_mas.len];

len = \_mas.len;

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

{

mas[q] = \_mas.mas[q];

}

}

}

template<class T>

TVector<T>::~TVector()

{

len = 0;

delete this->mas;

}

template<class T>

inline T TVector<T>::GetLen()

{

return len;

}

template<class T>

void TVector<T>::PushBack(T var)

{

T\* \_mas = new T[len];

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

{

\_mas[i] = mas[i];

}

delete[] mas;

mas = 0;

mas = new T[len + 1];

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

{

mas[i] = \_mas[i];

}

mas[len] = var;

len++;

}

template<class T>

void TVector<T>::Resize(int NewLength)

{

if (NewLength < 0) throw "The new length (or equals) less than zero";

if (NewLength >= 0)

{

T\* temporary = new T[NewLength];

for (int q = 0; q < NewLength; q++)

{

temporary[q] = mas[q];

}

mas = 0;

delete[] mas;

mas = 0;

mas = new T[NewLength];

len = NewLength;

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

mas[q] = temporary[q];

temporary = 0;

delete[] temporary;

temporary = 0;

}

}

template<class T>

istream& operator>>(istream& stream, TVector<T>& \_vector)

{

int \_len = \_vector.GetLen();

T number;

\_vector.Resize(0);

for (int i = 0; i < \_len; i++)

{

stream >> number;

\_vector.PushBack(number);

}

return stream;

}

template<class T>

ostream& operator<<(ostream& stream, TVector<T>& \_vector)

{

for (int i = 0; i < \_vector.GetLen(); i++) {

stream << \_vector[i];

}

return stream;

}

template<class T>

T& TVector<T>::operator[](int i)

{

return mas[i];

}

template<class T>

TVector<T> TVector<T>::operator+ (const TVector<T>& \_vector)

{

if (\_vector.mas == 0 && \_vector.len < 0) throw "empty variable";

if (\_vector.len != this->len) throw "len of 1-st vector isn't equals to the 2-d vector";

else

{

TVector<T> result(\*this);

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

{

result.mas[q] = (this->mas[q]) + \_vector.mas[q];

}

return result;

}

}

template<class T>

TVector<T> TVector<T>::operator-(const TVector<T>& \_vector)

{

if (\_vector.mas == 0 && \_vector.len < 0) throw "empty variable";

if (\_vector.len != this->len) throw "len of 1-st vector isn't equals to the 2-d vector";

else

{

TVector<T> result(\*this);

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

{

result.mas[q] = (this->mas[q]) - \_vector.mas[q];

}

return result;

}

}

template<class T>

TVector<T> TVector<T>::operator/(const TVector<T>& \_vector)

{

if (\_vector.mas == 0 && \_vector.len < 0) throw "empty variable";

if (\_vector.len != this->len) throw "len of 1-st vector isn't equals to the 2-d vector";

else

{

TVector<T> result(\*this);

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

{

if (\_vector.mas[q] != 0)

result.mas[q] = (this->mas[q]) / \_vector.mas[q];

else throw "division by zero";

}

return result;

}

}

template<class T>

TVector<T>& TVector<T>::operator=(const TVector<T>& \_vector)

{

if (mas == 0) mas = new T[\_vector.len];

else if (mas != 0)

{

mas = 0;

delete[] mas;

mas = 0;

mas = new T[\_vector.len];

}

len = \_vector.len;

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

{

mas[q] = \_vector.mas[q];

}

return \*this;

}

template<class T>

TVector<T> TVector<T>::operator\*(const TVector<T>& \_vector)

{

if (\_vector.len != len) throw "Error!";

TVector<T> result(\*this);

for (int q = 0; q < result.len; q++)

{

result.mas[q] = result.mas[q] \* \_vector.mas[q];

}

return result;

}

template<class T>

bool TVector<T>::operator==(const TVector<T>& \_vector)

{

if (len != \_vector.len) return false;

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

{

if (mas[q] != \_vector.mas[q]) return false;

}

return true;

}

***8.2 Приложение 2(Матрица)***

#pragma once

#include "Vector.h"

#include <iostream>

using namespace std;

template <class T>

class TMatrix : public TVector<TVector<T>>

{

protected:

int height;

public:

TMatrix();

~TMatrix();

TMatrix(int \_len, int \_height);

TMatrix(int \_len, int \_height, T variable);

TMatrix(const TMatrix<T>& \_matrix);

int GetHeight();

int GetLen();

TMatrix<T> operator + (const TMatrix<T>& \_matrix);

TMatrix<T> operator - (const TMatrix<T>& \_matrix);

TMatrix<T>& operator = (const TMatrix<T>& \_matrix);

TMatrix<T> operator \* (const TMatrix<T>& \_matrix);

TMatrix<T> operator \* (const TVector<T>& \_vector);

bool operator == (const TMatrix<T>& \_matrix);

};

template<class T>

TMatrix<T>::TMatrix()

{

this->mas = 0;

this->len = 0;

this->height = 0;

}

template<class T>

TMatrix<T>::TMatrix(int \_len, int \_height)

{

this->len = \_len;

this->height = \_height;

this->mas = new TVector<T>[height];

for (int i = 0; i < \_height; i++)

{

this->mas[i] = TVector<T>(\_len);

}

}

template<class T>

TMatrix<T>::TMatrix(int \_len, int \_height, T variable)

{

this->len = \_len;

this->height = \_height;

this->mas = new TVector<T>[height];

for (int i = 0; i < \_height; i++)

{

this->mas[i] = TVector<T>(\_len);

}

for (int x = 0; x < \_height; x++)

for (int y = 0; y < \_len; y++)

this->mas[x][y] = variable;

}

template<class T>

TMatrix<T>::TMatrix(const TMatrix<T>& \_matrix)

{

this->height = \_matrix.height;

this->len = \_matrix.len;

this->mas = new TVector<T>[this->height];

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

{

this->mas[i] = TVector<T>(this->len);

}

for (int x = 0; x < this->height; x++)

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

this->mas[x][y] = \_matrix.mas[x][y];

}

template<class T>

TMatrix<T>::~TMatrix()

{

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

this->mas[i].~TVector();

this->mas[i] = 0;

}

delete[] this->mas;

this->mas = 0;

}

template<class T>

int TMatrix<T>::GetHeight()

{

return this->height;

}

template<class T>

int TMatrix<T>::GetLen()

{

return this->len;

}

template<class T>

ostream& operator<<(ostream& stream, TMatrix<T>& \_matrix)

{

for (int q = 0; q < \_matrix.GetHeight(); q++)

{

for (int i = 0; i < \_matrix.GetLen(); i++)

stream << \_matrix[q][i];

}

return stream;

}

template<class T>

istream& operator>>(istream& stream, TMatrix<T>& \_matrix)

{

T var = 0;

for (int q = 0; q < \_matrix.GetHeight(); q++)

{

for (int i = 0; i < \_matrix.GetLen(); i++)

{

stream >> var;

\_matrix[q][i] = var;

}

}

return stream;

}

template<class T>

TMatrix<T> TMatrix<T>::operator+(const TMatrix& \_matrix)

{

if ((this->height == \_matrix.height) && (this->len == \_matrix.len)) {

TMatrix result(\*this);

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

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

result.mas[i][j] = this->mas[i][j] + \_matrix.mas[i][j];

}

}

return result;

}

else {

cout << "The number of rows does not match the number of columns in the matrices. Check the matrices." << endl;

abort();

}

}

template<class T>

TMatrix<T> TMatrix<T>::operator-(const TMatrix<T>& \_matrix)

{

if ((this->height == \_matrix.height) && (this->len == \_matrix.len)) {

TMatrix result(\*this);

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

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

result.mas[i][j] = this->mas[i][j] - \_matrix.mas[i][j];

}

}

return result;

}

else {

cout << "The number of rows does not match the number of columns in the matrices. Check the matrices." << endl;

abort();

}

}

template<class T>

TMatrix<T>& TMatrix<T>::operator=(const TMatrix<T>& \_matrix)

{

if (this == &\_matrix) return \*this;

if (this->len != \_matrix.len || height != \_matrix.height) throw "the len or height of first matrix isn't equals to len of height of second matrix";

for (int q = 0; q < height; q++)

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

this->mas[q][w] = \_matrix.mas[q][w];

return \*this;

}

template<class T>

TMatrix<T> TMatrix<T>::operator\*(const TMatrix<T>& \_matrix)

{

if (this->len == \_matrix.height) {

TMatrix<T> result(\_matrix.len, height, 0);

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

for (int j = 0; j < \_matrix.height; j++) {

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

result.mas[i][j] += this->mas[i][k] \* \_matrix.mas[k][j];

}

}

}

return result;

}

else {

cout << "The number of rows does not match the number of columns in the matrices. Check the matrices." << endl;

abort();

}

}

template<class T>

TMatrix<T> TMatrix<T>::operator\*(const TVector<T>& \_vector)

{

TVector<T> temp(\_vector);

if ((this->len) != temp.GetLen()) throw "size of matrines aren't correct";

TMatrix<T> result(1, height, 0);

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

{

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

result[i][0] += this->mas[i][x] \* temp[x];

}

return result;

}

template<class T>

inline bool TMatrix<T>::operator==(const TMatrix<T>& \_matrix)

{

if (this == &\_matrix) return true;

if (this->len != \_matrix.length || height != \_matrix.height) return false;

for (int q = 0; q < height; q++)

{

for (int w = 0; w < this->length; w++)

if (this->mas[q][w] != \_matrix.mas[q][w]) return false;

}

return true;

}

***8.3 Приложение 3(main.cpp)***

#include<iostream>

#include"Vector.h"

#include"Matrix.h"

using namespace std;

int main()

{

TMatrix<int> A(3, 3, 2);

TMatrix<int> G(3, 3);

TVector<int> B(3);

TVector<int> Q(3);

TVector<int> E;

cin >> B;

cout << "B = ";

for (int i = 0; i < 3; i++)

{

cout << B[i] << " ";

}

cout << endl;

cin >> Q;

cout << "Q = ";

for (int i = 0; i < 3; i++)

{

cout << Q[i] << " ";

}

G = A + A;

TVector<int> R = B;

TMatrix<int> C = A \* B;

cout << C;

cout << A;

return 0;

}