МИНИСТЕРСТВО НАУКИ И ВЫСШЕГООБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

**ЛАБОРАТОРНАЯ РАБОТА**

на тему:

**«Верхнетреугольные матрицы на шаблонах»**

**Выполнил:** студент группы 3822Б1ФИ2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Миронов А. И./

Подпись

**Проверил:** к.т.н, доцент каф. ВВиСП

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Кустикова В.Д./

Подпись

Нижний Новгород  
2023

**Содержание**

[Введение 3](#_Toc148449443)

[1 Постановка задачи 4](#_Toc148449444)

[2 Руководство пользователя 5](#_Toc148449445)

[2.1 Приложение для демонстрации работы вектора 5](#_Toc148449446)

[2.2 Приложение для демонстрации работы матрицы 7](#_Toc148449447)

[3 Руководство программиста 8](#_Toc148449448)

[3.1 Описание алгоритмов 8](#_Toc148449449)

[3.1.1 Вектор 8](#_Toc148449450)

[3.1.2 Матрица 9](#_Toc148449451)

[3.2 Описание программной реализации 10](#_Toc148449452)

[3.2.1 Описание класса TVector 10](#_Toc148449453)

[3.2.2 Описание класса TMatrix 14](#_Toc148449454)

[Заключение 18](#_Toc148449455)

[Литература 19](#_Toc148449456)

[Приложения 20](#_Toc148449457)

[Приложение А. Реализация класса TVector 20](#_Toc148449458)

[Приложение Б. Реализация класса TMatrix 21](#_Toc148449459)

# Введение

Лабораторная работа "Векторы и верхнетреугольные матрицы на шаблонах" направлена на изучение и практическое применение концепции шаблонов в языке программирования C++. Шаблоны позволяют создавать обобщенные типы данных, которые могут быть использованы с различными типами данных без необходимости дублирования кода. В рамках данной работы мы будем рассматривать примеры использования шаблонов для реализации векторов и верхнетреугольных матриц.

# Постановка задачи

**Цель:**

Ознакомление студентов с принципами работы шаблонов в языке C++ и их применением для создания обобщенных типов данных. В результате выполнения работы студенты должны приобрести практические навыки по созданию и использованию шаблонов для реализации векторов и верхнетреугольных матриц.

**Задачи:**

1. Изучение основных принципов работы шаблонов в языке C++.

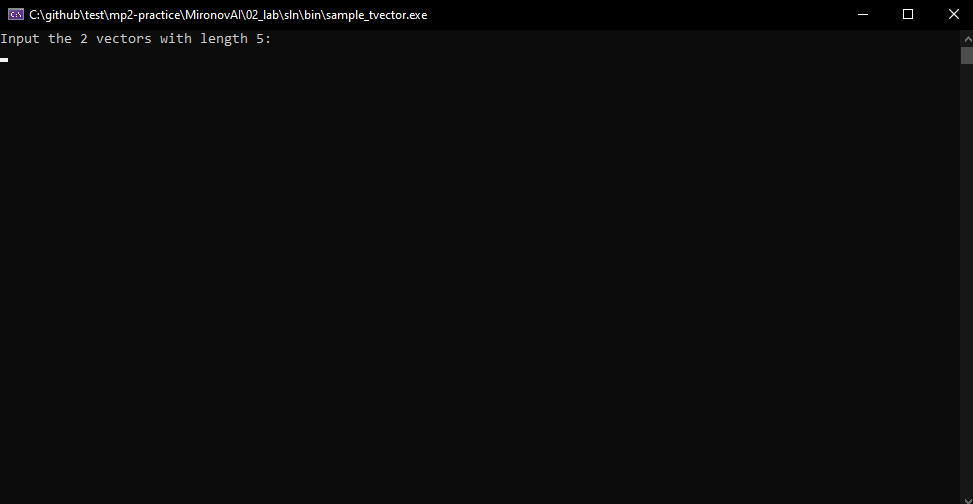
2. Разработка шаблонного класса для реализации вектора, который будет поддерживать основные операции.

3. Разработка шаблонного класса для реализации верхнетреугольной матрицы, который будет поддерживать операции сложения матриц, умножения матриц и т.д.

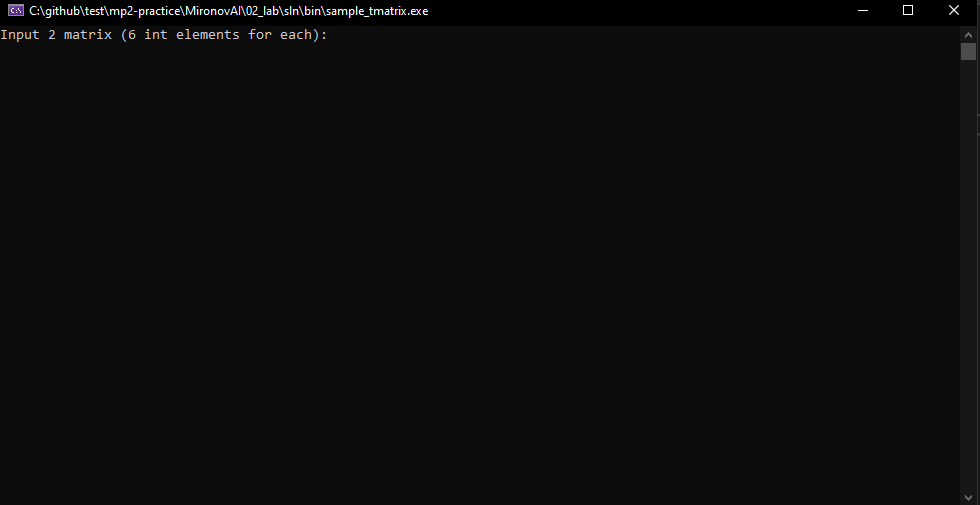
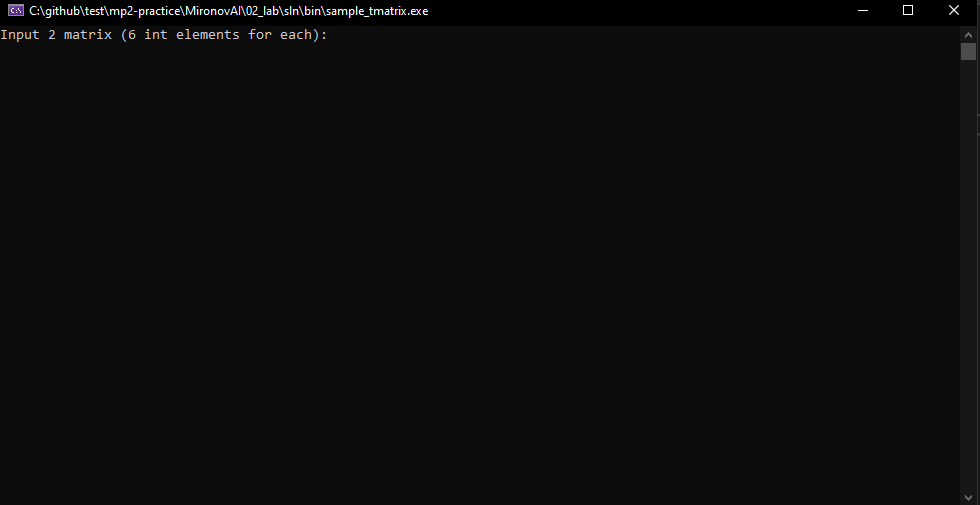
4. Проведение тестирования разработанных шаблонных классов на различных наборах данных для проверки их корректности и эффективности.

# Руководство пользователя

## Приложение для демонстрации работы вектора

1. Запустите приложение с названием sample\_tvector.exe. В результате появится окно, показанное ниже и вам будет предложено ввести 2 целочисленных вектора длины 5 (рис. 1).
2. Основное окно программы 
3. После ввода будет выведены результаты соответствующих операций и функций (рис 2).
4. Результат тестирования функций класса TVector 

## Приложение для демонстрации работы матрицы

1. Запустите приложение с названием sample\_tmatrix.exe. В результате появится окно, показанное ниже, вам будет предложено ввести 2 целочисленных верхнетреугольных матрицы 3 х 3 (вам нужно ввести 12 чисел) (рис 3).
2. Основное окно программы 
3. После ввода матриц будет выведены результаты соответствующих операций и функций (рис 4).
4. Результат тестирования функций класса TMatrix

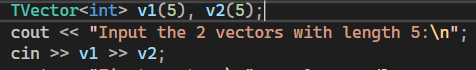
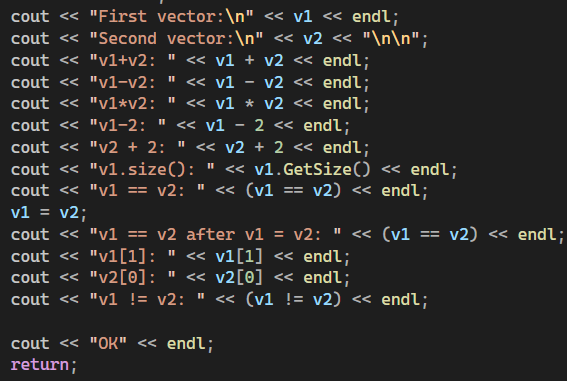
# Руководство программиста

## Описание алгоритмов

### Вектор

1. Начало работы указано в пункте [2.1 этого документа](#_Приложение_для_демонстрации)
2. Описание методов и полей класса в пункте [3.2.1 этого документа](#_Описание_класса_TVector)

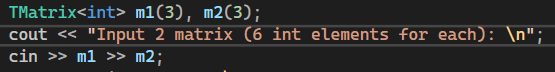
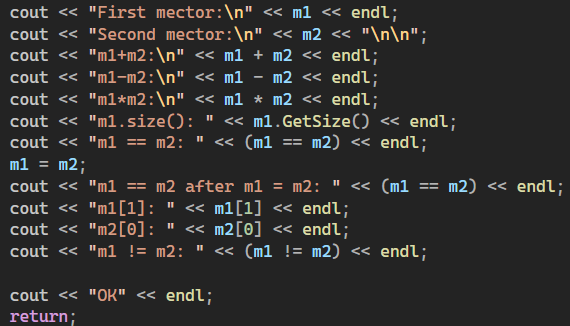
Программа алгоритм состоит из единственной функции void test\_tvector(), которая сразу вызывается из основной функции int main(). Изначально необходимо алгоритм запрашивает пользователя ввести 2 вектора поля длинной 5 для демонстрации использования экземпляров класса (рис 5)

1. Ввод данных алгоритма 
2. После чего алгоритм последовательно выведет результат вызова соответствующих операций и методов класса (рис 6).
3. Вывод результата работы алгоритма 

### Матрица

1. Начало работы указано в пункте [2.2 этого документа.](#_Приложение_для_демонстрации_1)
2. Описание методов и полей класса в пункте [3.2.2 этого документа](#_Описание_класса_TMatrix)

Программа алгоритм состоит из единственной функции void test\_tmatrix(), которая сразу вызывается из основной функции int main(). Изначально необходимо алгоритм запрашивает пользователя ввести 2 верхнетреугольных матрицы, элементы которых должны быть целочисленным целым числом, для демонстрации использования экземпляров класса (рис 7)

1. Ввод данных алгоритма 
2. После чего алгоритм последовательно выведет результат вызова соответствующих операций и методов класса (рис 8).
3. Результат работы алгоритма 

## Описание программной реализации

### Описание класса TVector

template <class ValType>

class TVector

{

private:

int StartIndex;

protected:

ValType \*pVector;

int Size;

//int StartIndex;

public:

TVector(int s = 10, int si = 0);

TVector(const TVector& v);

~TVector();

int GetSize() { return Size; }

int GetStartIndex() { return StartIndex; }

ValType& operator[](int pos);

bool operator==(const TVector<ValType> &v) const;

bool operator!=(const TVector<ValType> &v) const;

const TVector<ValType>& operator=(const TVector<ValType> &v);

// скалярные операции

TVector<ValType> operator+(const ValType &val);

TVector<ValType> operator-(const ValType &val);

TVector<ValType> operator\*(const ValType &val);

// векторные операции

TVector<ValType> operator+(const TVector<ValType> &v);

TVector<ValType> operator-(const TVector<ValType>&v);

ValType operator\*(const TVector<ValType> &v);

// ввод-вывод

friend istream& operator>>(istream &in, TVector<ValType> &v);

friend ostream& operator<<(ostream &out,TVector<ValType> &v);

};

Назначение: представление вектора

Поля:

StartIndex - индекс первого необходимого элемента вектора

\*pVector – память для представления элементов вектора

Size – количество нужных элементов вектора

Методы:

TVector(int s = 10, int si = 0);

Назначение: конструктор по умолчанию и конструктор с параметрами

Входные параметры:

s – длинна вектора (по умолчанию 10)

si – стартовый индекс (по умолчанию 0)

Выходные параметры: -

TVector(const Tvector<ValType>& v);

Назначение: конструктор копирования

Входные параметры:

v – экземпляр класса, на основе которого создаем новый объект

Выходные параметры: -

~TVector();

Назначение: деструктор

Входные параметры: -

Выходные параметры: -

int GetSize();

Назначение: получение размера вектора

Входные параметры: -

Выходные параметры: размер вектора ( количество элементов)

int GetStartIndex();

Назначение: получение стартового индекса

Входные параметры: -

Выходные параметры: стартовый индекс

ValType& operator[](int pos);

Назначение: перегрузка оператора индексации

Входные параметры:

pos – позиция (индекс) элемента

Выходные параметры: элемент, который находится на pos позиции

bool operator==(const TVector<ValType> &v) const;

Назначение: оператор сравнения

Входные параметры:

v – экземпляр класса, с которым сравниваем

Выходные параметры:

True (1), если они равны, иначе false(0).

bool operator!=(const TVector<ValType> &v) const;

Назначение: оператор сравнения

Входные параметры:

v – экземпляр класса, с которым сравниваем

Выходные параметры:

True (1), если они не равны, иначе false(0).

const TVector<ValType>& operator=(const TVector<ValType> &v);

Назначение: оператор присваивания

Входные параметры:

v – экземпляр класса, который присваиваем

Выходные параметры:

Ссылка на (\*this) , уже присвоенный экземпляр класса

TVector<ValType> operator+(const ValType &val);

Назначение: оператор суммирования вектора и значения

Входные параметры:

Val – элемент, с которым суммируем

Выходные параметры:

Экземпляр класса, элементы которого на val больше

TVector<ValType> operator-(const ValType &val);

Назначение: оператор вычитания вектора и значения

Входные параметры:

Val – элемент, который вычитаем

Выходные параметры:

Экземпляр класса, элементы которого на val меньше

TVector<ValType> operator\*(const ValType &val);

Назначение: оператор умножения вектора на значение

Входные параметры:

Val – элемент, на который умножаем вектор

Выходные параметры:

Экземпляр класса, элементы которого в val раз больше

TVector<ValType> operator+(const TVector<ValType> &v);

Назначение: оператор суммирования векторов

Входные параметры:

V – вектор, который суммируем

Выходные параметры:

Экземпляр класса, равный сумме двух векторов

TVector<ValType> operator-(const TVector<ValType> &v);

Назначение: оператор вычитания векторов

Входные параметры:

V – вектор, который вычитаем

Выходные параметры:

Экземпляр класса, равный разности двух векторов

TVector<ValType> operator\*(const TVector<ValType> &v);

Назначение: оператор умножения векторов

Входные параметры:

V – вектор, на который умножаем

Выходные параметры:

Значение, равное скалярному произведению двух векторов

friend istream& operator>>(istream &in, TVector<ValType> &v);

Назначение: оператор ввода вектора

Входные параметры:

In – ссылка на буфер, из которого вводим вектор

V – ссылка на вектор, который вводим

Выходные данные:

In – ссылка буфер

friend ostream& operator<<(ostream &out,TVector<ValType> &v);

Назначение: оператор вывода вектора

Входные параметры:

In – ссылка на буфер, из которого выводим вектор

V – ссылка на вектор, который выводим

Выходные данные:

In – ссылка буфер

### Описание класса TMatrix

template <class ValType>

class TMatrix : public TVector<TVector<ValType>>

{

public:

TMatrix(int s = 10);

TMatrix(const TMatrix &mt);

TMatrix(const TVector<TVector<ValType> > &mt);

bool operator==(const TMatrix<ValType>&mt) const;

bool operator!=(const TMatrix<ValType>&mt) const;

const TMatrix& operator=(const TMatrix<ValType> &mt);

TMatrix operator+(const TMatrix<ValType> &mt);

TMatrix operator-(const TMatrix<ValType> &mt);

TMatrix operator\*(const TMatrix<ValType> &mt);

// ввод / вывод

friend istream& operator>>(istream &in, TMatrix<ValType>&mt);

friend ostream & operator<<( ostream &out, const);

};

Класс наследуется от класса TVector<TVector<ValType>>.

Назначение: представление матрицы как вектор векторов

Поля:

StartIndex - индекс первого необходимого элемента

\*pVector – память для представления элементов матрицы

Size – размерность матрицы

Методы:

TMatrix(int s = 10);

Назначение: конструктор по умолчанию и конструктор с параметрами

Входные параметры:

s – длинна вектора (по умолчанию 10)

Выходные параметры: -

TMatrix(const TMatrix &mt);

Назначение: конструктор копирования

Входные параметры:

mt – экземпляр класса, на основе которого создаем новый объект

Выходные параметры: -

TMatrix(const TVector<TVector<ValType> > &mt);

Назначение: Конструктор преобразования типов

Входные параметры:

mt – ссылка на TVector<TVector<ValType>> - на объект, который преобразуем

Выходные данные: -

bool operator==(const TMatrix<ValType>&mt) const;

Назначение: оператор сравнения

Входные параметры:

mt – экземпляр класса, с которым сравниваем

Выходные параметры:

True (1), если они равны, иначе false(0).

bool operator!=(const TMatrix<ValType>&mt) const;

Назначение: оператор сравнения

Входные параметры:

mt – экземпляр класса, с которым сравниваем

Выходные параметры:

True (1), если они не равны, иначе false(0).

const TMatrix& operator=(const TMatrix<ValType> &mt);

Назначение: оператор присваивания

Входные параметры:

mt – экземпляр класса, который присваиваем

Выходные параметры:

Ссылка на (\*this) , уже присвоенный экземпляр класса

TMatrix operator+(const TMatrix<ValType> &mt);

Назначение: оператор суммирования матриц

Входные параметры:

mt – ссылка на матрицу, которую суммируем

Выходные параметры:

Экземпляр класса, равный сумме двух матриц

TMatrix operator-(const TMatrix<ValType> &mt);

Назначение: оператор вычитания матриц

Входные параметры:

mt – ссылка на матрицу, которую вычитаем

Выходные параметры:

Экземпляр класса, равный разности двух матриц

TMatrix operator\*(const TMatrix<ValType> &mt);

Назначение: оператор умножения матриц

Входные параметры:

mt – ссылка на матрицу, которую умножаем

Выходные параметры:

Экземпляр класса, равный произведению двух матриц

friend istream& operator>>(istream &in, TMatrix<ValType>&mt);

Назначение: оператор ввода матрицы

Входные параметры:

In – ссылка на буфер, из которого вводим матрицу

V – ссылка на матрицу, которую вводим

Выходные данные:

In – ссылка буфер

friend ostream& operator<<(ostream &out, TMatrix<ValType>&mt);

Назначение: оператор вывода матрицы

Входные параметры:

Out – ссылка на буфер, из которого выводим матрицу

V – ссылка на матрицу, который выводим

Выходные данные:

Out – ссылка буфер

# Заключение

В ходе выполнения лабораторной работы мы изучили и практически применили концепцию шаблонов в языке программирования C++. Шаблоны позволяют создавать обобщенные типы данных, которые могут быть использованы с различными типами данных без необходимости дублирования кода.

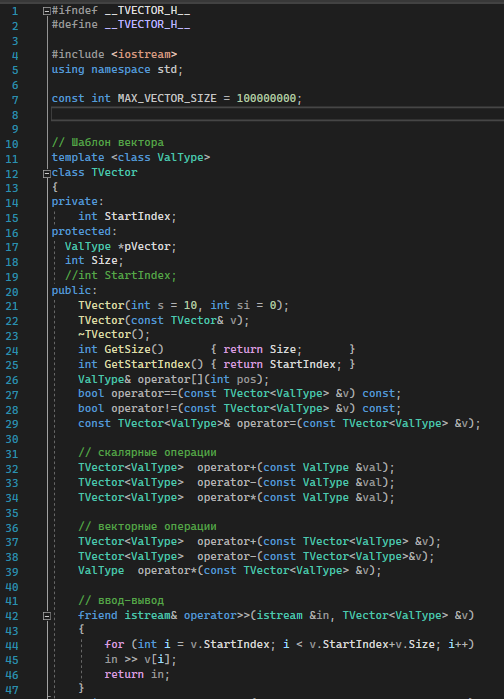
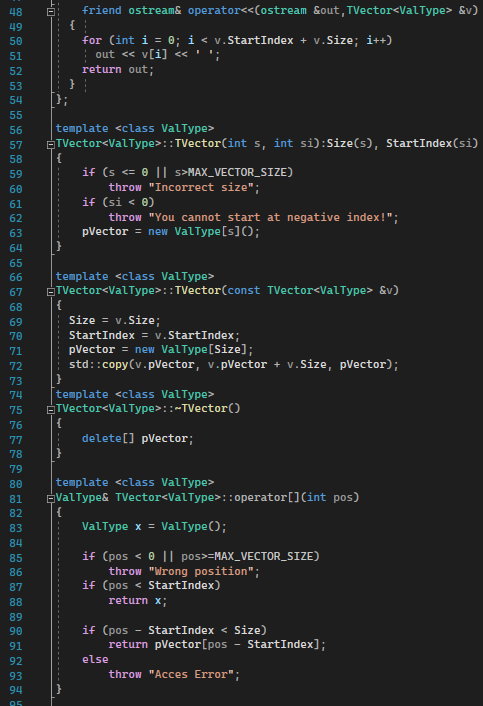
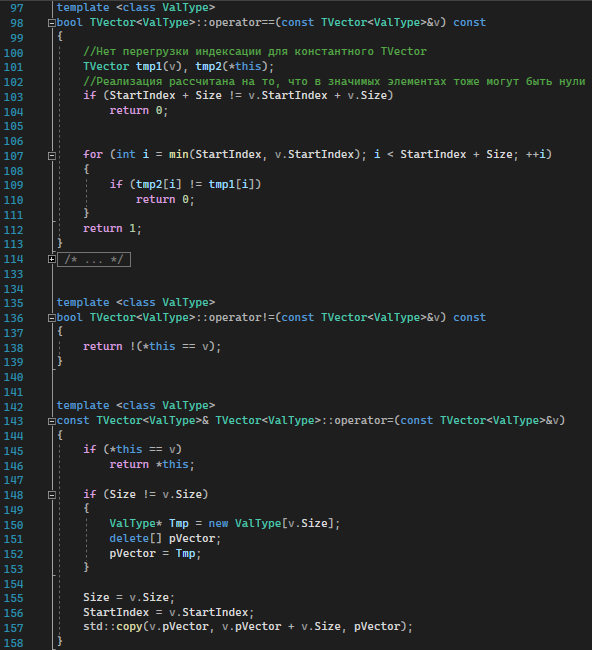
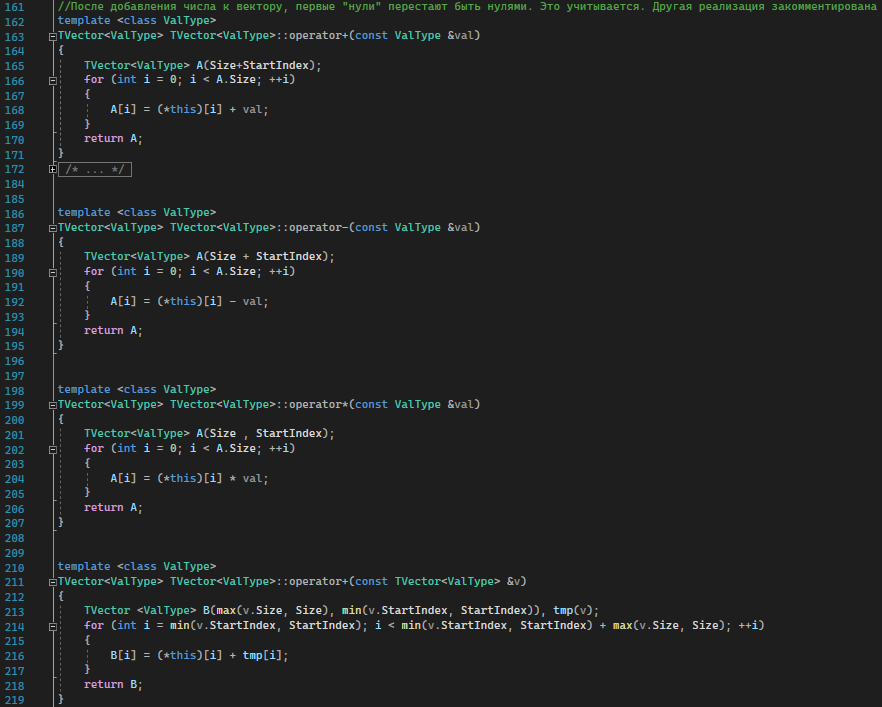
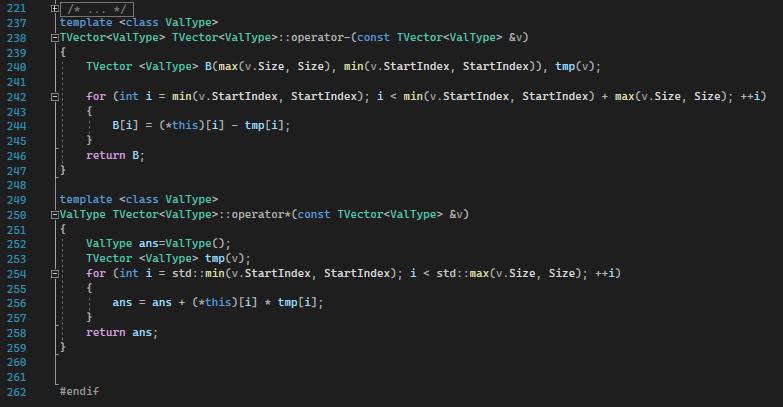
В рамках работы мы разработали шаблонный класс для реализации вектора, который поддерживает основные операции, такие как добавление элемента, удаление элемента, доступ к элементу по индексу и другие. Также мы разработали шаблонный класс для реализации верхнетреугольной матрицы, который поддерживает операции сложения матриц, умножения матрицы на матрицу и другие

# Литература

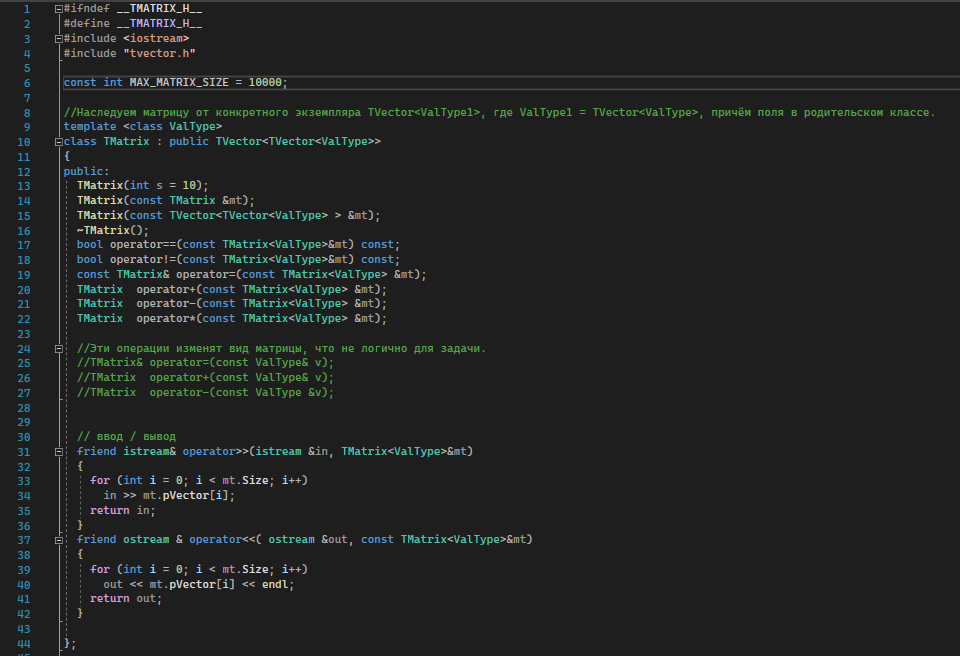
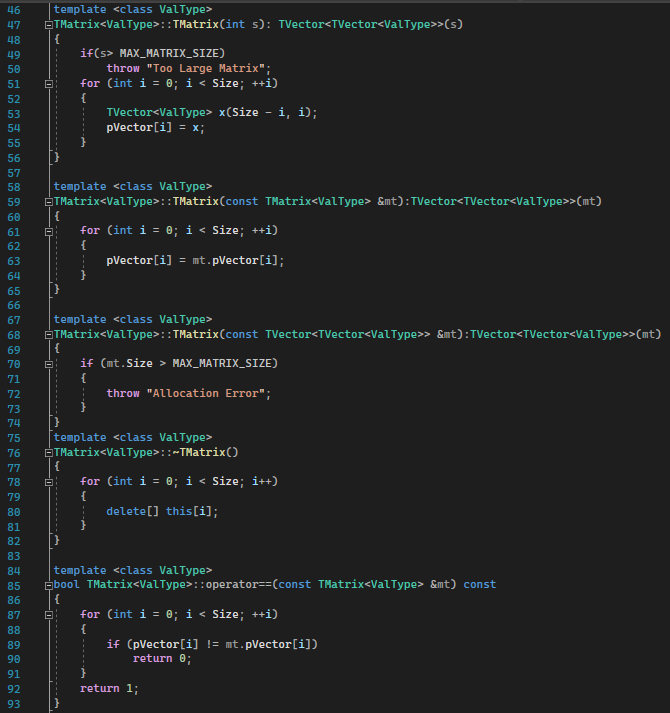
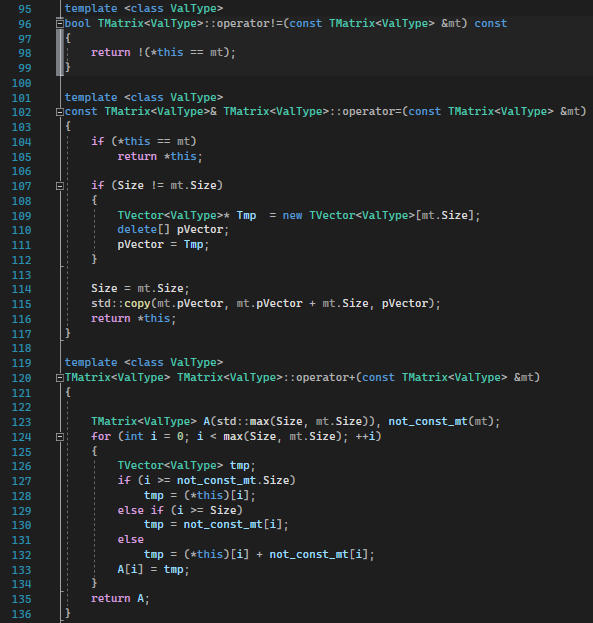
1. Википедия [https://ru.wikipedia.org/wiki/Треугольная\_матрица].

# Приложения

## Приложение А. Реализация класса TVector

1. Реализация класса TVector 1
2. Реализация класса TVector 2
3. Реализация класса TVector 3
4. Реализация класса TVector 4
5. Реализация класса TVector 5

## Приложение Б. Реализация класса TMatrix

1. Реализация класса TMatrix 1
2. Реализация класса TMatrix 2
3. Реализация класса TMatrix 3
4. Реализация класса TMatrix 4