ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»  
(РУТ (МИИТ))

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

ОТЧЁТ  
О ПРАКТИЧЕСКОЙ РАБОТЕ № 1

По дисциплине «Языки программирования»

Выполнил: ст. гр. ТКИ – 241

Захаров Дмитрий Константинович

Проверил: к.т.н., доц.

Васильева М. А.

Москва 2023

**Задание:** написать класс вектор с использованием темплейтов

**Код задания:**

#pragma once

#include <iostream>

#include <sstream>

namespace miit::vector

{

template<typename T> class Vector;

/\*\*

@brief Опрератор сдвига.

@param os Поток вывода.

@param vector Вектор для вывода.

@return Изменённый поток вывода.

\*/

template<typename T>

std::ostream& operator<< (std::ostream& os, const Vector<T>& vector);

/\*\*

@brief Перегрузка оператора равно

@param rha Первый аргумент для сравнения

@param lha Второй аргумент для сравнения

@return false если равны и true если нет

\*/

template<typename T>

bool operator==(const Vector<T>& rha, const Vector<T>& lha) noexcept;

/\*\*

@brief Перегрузка оператора не равно

@param rha Первый аргумент для сравнения

@param lha Второй аргумент для сравнения

@return true если равны и false если нет

\*/

template<typename T>

bool operator!=(const Vector<T>& rha, const Vector<T>& lha) noexcept;

template<typename T>

class Vector final

{

public:

/\*\*

@brief Создание vector'a по initializer\_list

@param list Лист по которому строится vector

\*/

Vector(const std::initializer\_list<T> list);

/\*\*

@brief Чистит памать за vector

\*/

~Vector();

/\*\*

@brief Получение длинны vector'a

@return Количество элементов vector'a

\*/

size\_t get\_size() const;

/\*\*

@brief Добавление элемента в вектор

@param Значение элемента

\*/

void append(const T& value);

/\*\*

@brief Получение индекса элемента в списке

@param value Искомое значение

@return индекс элемента если он есть в vector и -1 если его в нем нет

\*/

bool find(const T& value) const;

/\*\*

@brief Получение информации об наличии элементов в vector

@return true если есть элементы и false если нет

\*/

bool has\_elements() const;

/\*\*

@brief Перегрузка оператора квадратные скобки

@param index Индекс элемента в векторе

@return Элемент из вектора

\*/

T& operator[](size\_t index);

/\*\*

@brief Перегрузка оператора квадратные скобки для конастантного объекта

@param index Индекс элемента в векторе

@return Элемент из вектора

\*/

const T& operator[](size\_t index) const;

/\*\*

@brief Перегрузка оператора копирования

@param Vector Вектор для копирования

@return Скопированный объект типа Vector

\*/

Vector<T>& operator=(const Vector<T>& vector);

/\*

@brief Перегрузка оператора перемещения

@param Vector Список для перемещения

@return Объект типа DLList

\*/

Vector<T>& operator=(Vector<T>&& vector) noexcept;

/\*\*

@brief Move конструктор

@param vector Вектор для перемещения

\*/

Vector(const Vector<T>& vector);

/\*\*

@brief Конструктор копирования

@param vector Вектор для копирования

\*/

Vector(Vector<T>&& vector) noexcept;

/\*\*

@brief Преобразование объета в строку

@return строка полученная из объекта

\*/

std::string to\_string() const noexcept;

private:

size\_t size;

T\* values;

};

template<typename T>

inline Vector<T>::Vector(std::initializer\_list<T> list)

:size(0)

{

for (auto& value : list)

{

this->append(value);

}

}

template<typename T>

inline Vector<T>::~Vector()

{

delete[] this->values;

this->values = nullptr;

}

template<typename T>

inline size\_t Vector<T>::get\_size() const

{

return this->size;

}

template<typename T>

void Vector<T>::append(const T& value)

{

T\* temp = new T[this->size + 1];

std::copy(this->values, this->values + this->size, temp);

temp[this->size] = value;

delete[] this->values;

std::swap(temp, this->values);

this->size++;

}

template<typename T>

Vector<T>::Vector(Vector<T>&& vector) noexcept

:size(0), values(nullptr)

{

\*this = std::move(vector);

}

template<typename T>

Vector<T>& Vector<T>::operator=(Vector<T>&& vector) noexcept

{

if (\*this != vector)

{

\*this = std::move(vector);

}

return \*this;

}

template<typename T>

Vector<T>::Vector(const Vector<T>& vector)

:size(0), values(nullptr)

{

Vector temp {};

for (size\_t i = 0; i < vector.get\_size(); i++)

{

temp.append(vector.values[i]);

}

std::swap(this->values, temp.values);

std::exchange(this->size, temp.size);

}

template<typename T>

inline Vector<T>& Vector<T>::operator=(const Vector& vector)

{

if (\*this != vector)

{

Vector temp{ vector };

std::swap(this->values, temp.values);

std::exchange(this->size, temp.size);

}

return \*this;

}

template<typename T>

bool Vector<T>::find(const T& value) const

{

for (size\_t i = 0; i < this->get\_size(); i++)

{

if (value == this->values[i])

{

return true;

}

}

return false;

}

template<typename T>

inline bool Vector<T>::has\_elements() const

{

return this->size != 0;

}

template<typename T>

inline T& Vector<T>::operator[](size\_t index)

{

if (!this->has\_elements() || index >= this->size)

{

throw std::out\_of\_range("Incorrect Index");

}

return this->values[index];

}

template<typename T>

inline const T& Vector<T>::operator[](size\_t index) const

{

if (!this->has\_elements() || index >= this->size)

{

throw std::out\_of\_range("Incorrect Index");

}

return this->values[index];

}

template<typename T>

std::string Vector<T>::to\_string() const noexcept

{

std::stringstream result;

result << "[ ";

for (size\_t i = 0; i < this->size; i++)

{

result << this->values[i] << " ";

}

result << "]";

return result.str();

}

template<typename T>

bool operator==(const Vector<T>& rha, const Vector<T>& lha) noexcept

{

return (rha.to\_string() == lha.to\_string());

}

template<typename T>

bool operator!=(const Vector<T>& rha, const Vector<T>& lha) noexcept

{

return !(rha == lha);

}

template<typename T>

std::ostream& operator<<(std::ostream& os, const Vector<T>& vector)

{

return os << vector.to\_string();

}

};

#include "Vector.h"

**Тесты:**

#include "pch.h"

#include "CppUnitTest.h"

#include "../Vector/Vector.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

using namespace miit::vector;

namespace VectorTest

{

TEST\_CLASS(VectorTest)

{

public:

TEST\_METHOD(IndexOperator\_IncorrectIndex\_ExeptionExpected)

{

Assert::ExpectException<std::out\_of\_range>([]()

{auto x = Vector({ 12, 33,44,55,11 }); x[10]; });

}

TEST\_METHOD(ToString\_ValidData\_Success)

{

Vector vector\_1{ 12, 33,44,55,11 };

Assert::IsTrue(vector\_1.to\_string() == "[ 12 33 44 55 11 ]");

}

TEST\_METHOD(EqualityOperator\_ValidData\_Success)

{

Vector vector\_1{ 12, 33,44,55,11 };

Vector vector\_2{ 12, 33,44,55,11 };

Assert::IsTrue(vector\_1 == vector\_1);

}

TEST\_METHOD(GetSize\_ValidData\_Success)

{

Vector vector\_1{ 12, 33,44,55,11 };

Assert::IsTrue(vector\_1.get\_size() == 5);

}

TEST\_METHOD(Append\_ValidData\_Success)

{

Vector vector\_1{ 12, 33,44,55,11 };

vector\_1.append(12);

Vector expected\_vector{ 12,33,44,55,11,12 };

Assert::IsTrue(vector\_1 == expected\_vector);

}

TEST\_METHOD(Find\_ValidData\_Success)

{

Vector vector\_1{ 12, 33,44,55,11 };

Assert::IsTrue(vector\_1.find(33) == 1);

}

TEST\_METHOD(HasElements\_ValidData\_Success)

{

Vector vector\_1 { 12, 33,44,55,11 };

Assert::IsTrue(vector\_1.has\_elements());

}

TEST\_METHOD(OutputOperator\_ValidData\_Success)

{

std::stringstream sstring;

Vector vector = Vector({ 10,11,12,13 });

sstring << vector;

Assert::AreEqual(sstring.str(), vector.to\_string());

}

TEST\_METHOD(CopyOperator\_ValidData\_Success)

{

Vector vector\_1{ 12, 33,44,55,11 };

Vector vector\_2{ 33, 22 ,44, 55 };

vector\_1 = vector\_2;

Assert::IsTrue(vector\_1 == vector\_2);

}

TEST\_METHOD(MoveOperator\_ValidData\_Success)

{

Vector vector\_1{ 12, 33,44,55,11 };

Vector vector\_2{ 33, 22 ,44, 55 };

vector\_1 = std::move(vector\_2);

Vector expected\_vector{ 33, 22 ,44, 55 };

Assert::IsTrue(vector\_1 == expected\_vector);

}

};

}

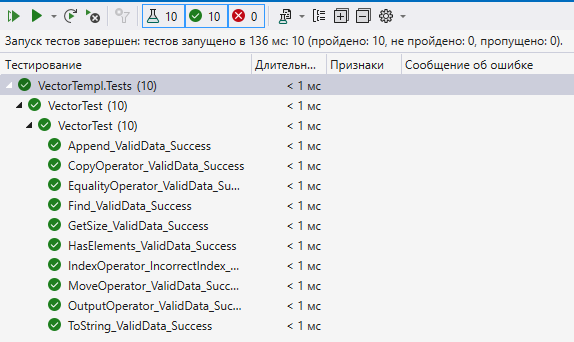
****

Рисунок 1 – прохождение тестов

**UML Диаграмма:**

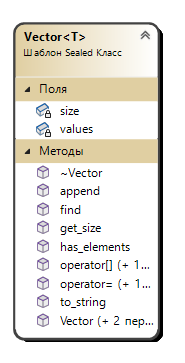


Рисунок 2 – UML диаграмма

****

Рисунок 3 – Approve