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 № 3

По дисциплине «Языки программирования»

Выполнил: ст. гр. ТКИ – 241

Белавинская М.А.

Проверил: к.т.н., доц.

Васильева М. А.

Москва 2023

Написать линейный односвязный список с использованием умных указателей

**Код задания:**

#include <memory>

#include <iostream>

#include <sstream>

template<typename T>

class List

{

private:

template<typename T>

struct Node

{

T data;

std::unique\_ptr<Node> next;

};

size\_t size;

std::unique\_ptr<Node> head;

public:

/\*

@brief Создание пустого объекта

\*/

List()

: head(nullptr), size(0) {}

/\*

@brief Создание деструктора для чистки памяти при удалении объекта

\*/

~List();

/\*

@brief Добавление элемента в список

@param data Элемент, который добавляем

\*/

void pushBack(T data);

/\*

@brief Преобразуем список в строку

\*/

std::string toString() const noexcept;

};

template<typename T>

List<T>::~List()

{

while (head==nullptr)

{

auto next = std::move(head->next\_node);

head = std::move(next);

}

}

template<typename T>

void List<T>::pushBack(T data)

{

head = std::unique\_ptr<Node<T>>(new Node{ data, std::move(head) });

}

template<typename T>

std::string List<T>::toString() const noexcept

{

std::stringstream buffer;

auto current = std::move(head);

while (current != nullptr)

{

buffer << current->data << ' ';

current = std::move(current->next);

}

return buffer.str();

}

#include "List.h"

#include "../List2/List.h"

template<typename T>

int main()

{

List<T> list1;

list1.pushBack(4);

list1.pushBack(2);

list1.pushBack(3);

list1.pushBack(6);

std::cout << list1.toString() << "\n" << std::endl;

return 0;

}

**Тесты:**

#include "pch.h"

#include "CppUnitTest.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest1

{

TEST\_CLASS(ListTests)

{

public:

TEST\_METHOD(ToString\_ValidData\_Success)

{

//arrange

List list{ 1,2,3,4,5,6 };

//assert

Assert::IsTrue(list.to\_string() == "[ 1 2 3 4 5 6 ]");

}

TEST\_METHOD(Push\_Data\_Success)

{

//arrange

List list{ 1,2,3,4,5 };

List expected{ 6,1,2,3,4,5 };

//act

list.push(6);

//assert

Assert::IsTrue(list == expected);

}

TEST\_METHOD(Pop\_Data\_Success)

{

//arrange

List list{ 1,2,3,4,5 };

List expected{ 2,3,4,5 };

//act

list.pop();

//assert

Assert::IsTrue(list == expected);

}

};

};

**UML Диаграмма:**

**Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание**

Рисунок 1 – UML диаграмма

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Рисунок 2 – Approve