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

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

По дисциплине «Системы управления базами данных»

Выполнил: ст. гр. ТКИ – 441,

Тешев В.А.

Проверил: к.т.н., доц.,

Васильева М. А.

Москва 2022

ОГЛАВЛЕНИЕ

[ОГЛАВЛЕНИЕ 2](#_Toc117015075)

[ЗАДАНИЕ 3](#_Toc117015076)

[1 ЛИСТИНГ ПРОГРАММЫ 4](#_Toc117015077)

[1.1 Файл «CircularLinkedList.h» 4](#_Toc117015078)

[1.2 Файл «CircularLinkedList.cpp» 7](#_Toc117015079)

[1.3 Файл «CircularLinkedListTest.cpp» 13](#_Toc117015080)

[2 ДИАГРАММА КЛАССОВ 17](#_Toc117015081)

[3 РЕЗУЛЬТАТЫ ПРОХОЖДЕНИЙ ТЕСТОВ 18](#_Toc117015082)

ЗАДАНИЕ

1. Для структуры «ЦОС» разработать API;
2. На языке «C++» написать конструктор и деструктор класса разрабатываемой структуры для типа «int»;
3. Разработать метод вывода структуры в строку;
4. Реализовать CRUD;
5. Разработать тесты.

# ЛИСТИНГ ПРОГРАММЫ

## Файл «CircularLinkedList.h»

#pragma once

#include <iostream>

#include <sstream>

using namespace std;

/\*\*

\* \brief Структура узла

\*/

struct Node

{

private:

/\*\*

\* \brief Данные узла

\*/

int data;

/\*\*

\* \brief Указатель на следующий узел в списке

\*/

Node\* next;

friend class CircularLinkedList; // Для возможности классу ЦОС обращаться к privite полям Node

public:

/\*\*

\* \brief Консруктор с параметрами

\*/

Node(const int data\_) noexcept;

/\*\*

\* \brief Деструктор

\*/

~Node() noexcept;

};

/\*\*

\* \brief Класс ЦОС

\*/

class CircularLinkedList

{

private:

/\*\*

\* \brief Указатель на голову списка

\*/

Node\* head;

/\*\*

\* \brief Удаление узла с ЦОС

\*/

Node\* delete\_head(Node \*node);

public:

/\*\*

\* \brief Консруктор по умолчанию

\*/

CircularLinkedList() noexcept;

/\*\*

\* \brief Деструктор

\*/

~CircularLinkedList() noexcept;

/\*\*

\* \brief Добавление узла в ЦОС

\*/

void add\_node(const int data\_) noexcept;

/\*\*

\* \brief Удаление узла с ЦОС

\*/

void remove() noexcept;

/\*\*

\* \brief Поиск узла в ЦОС

\*/

bool find\_node(const int data\_) const noexcept;

/\*\*

\* \brief Печать списка

\* \return Строка, содержащая узлы списка

\*/

string print\_list() const;

};

## Файл «CircularLinkedList.cpp»

#include "CircularLinkedList.h"

Node::Node(const int data\_) noexcept : data(data\_) {}

Node::~Node() noexcept

{

this->next = nullptr;

}

CircularLinkedList::CircularLinkedList() noexcept : head(nullptr) {}

CircularLinkedList::~CircularLinkedList() noexcept

{

Node\* currentNode = this->head;

Node\* headPtr = currentNode;

while (currentNode)

{

Node\* nextNode = currentNode->next;

if (currentNode != headPtr) {

delete currentNode;

}

if (nextNode == headPtr) {

delete headPtr;

currentNode = nullptr;

}

else {

currentNode = nextNode;

}

}

head = nullptr;

}

void CircularLinkedList::add\_node(const int data\_) noexcept

{

auto node = new Node(data\_);

if (head == nullptr)

{

head = node;

node->next = head;

}

else

{

Node\* ptr = head;

while (ptr->next != head)

{

ptr = ptr->next;

}

ptr->next = node;

node->next = head;

head = node;

}

}

Node\* CircularLinkedList::delete\_head(Node \*node)

{

Node\* temp = node->next;

Node\* p = head;

if (head == nullptr)

return nullptr;

if (p->next == head)

{

delete head;

return nullptr;

}

while (p->next != head)

{

p = p->next;

}

p->next = temp;

delete head;

return temp;

}

void CircularLinkedList::remove() noexcept

{

head = delete\_head(head);

}

bool CircularLinkedList::find\_node(const int data\_) const noexcept

{

if (head == nullptr)

{

throw invalid\_argument("Не существует узла с данным значением!");

}

Node\* current = head->next;

while (current != head)

{

if (current->data == data\_)

return current != nullptr;

else

current = current->next;

}

return current == nullptr;

}

string CircularLinkedList::print\_list() const

{

ostringstream tmpStream;

if (head == nullptr)

{

throw invalid\_argument("Список пуст!");

}

else

{

Node\* temp = head;

do {

tmpStream << temp->data << ",";

temp = temp->next;

} while (temp != head);

}

return tmpStream.str();

}

## Файл «CircularLinkedListTest.cpp»

#include "pch.h"

#include "CppUnitTest.h"

#include "..\Teshev\CircularLinkedList.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace CircularLinkedListTest

{

TEST\_CLASS(CircularLinkedListTest)

{

public:

TEST\_METHOD(First\_node\_insert)

{

// Arrange

std::string expected = "1,";

// Act

CircularLinkedList list;

std::ostringstream temp\_act;

list.add\_node(1);

temp\_act << list.print\_list();

std::string actual = temp\_act.str();

// Assert

Assert::AreEqual(expected, actual);

}

TEST\_METHOD(Second\_node\_insert)

{

// Arrange

std::string expected = "2,1,";

// Act

CircularLinkedList list;

std::ostringstream temp\_act;

list.add\_node(1);

list.add\_node(2);

temp\_act << list.print\_list();

std::string actual = temp\_act.str();

// Assert

Assert::AreEqual(expected, actual);

}

TEST\_METHOD(Delete\_node)

{

// Arrange

std::string expected = "1,";

// Act

CircularLinkedList list;

std::ostringstream temp\_act;

list.add\_node(1);

list.add\_node(2);

list.remove();

temp\_act << list.print\_list();

std::string actual = temp\_act.str();

// Assert

Assert::AreEqual(expected, actual);

}

TEST\_METHOD(Success\_node\_find)

{

// Arrange

std::string expected = "1";

// Act

CircularLinkedList list;

std::ostringstream temp\_act;

list.add\_node(1);

list.add\_node(2);

temp\_act << list.find\_node(1);

std::string actual = temp\_act.str();

// Assert

Assert::AreEqual(expected, actual);

}

TEST\_METHOD(Unsuccessful\_node\_find)

{

// Arrange

std::string expected = "0";

// Act

CircularLinkedList list;

std::ostringstream temp\_act;

list.add\_node(1);

list.add\_node(2);

temp\_act << list.find\_node(3);

std::string actual = temp\_act.str();

// Assert

Assert::AreEqual(expected, actual);

}

};

}

# ДИАГРАММА КЛАССОВ

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 - Диаграмма классов

# РЕЗУЛЬТАТЫ ПРОХОЖДЕНИЙ ТЕСТОВ

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 - Результаты прохождения тестов