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 ЛАБОРАТОРНОЙ РАБОТЕ №2

По дисциплине «ООП»

ВАРИАНТ 2

Выполнил: ст. гр. ТКИ-141

Буланый Сергей Евгеньевич

Проверил: к.т.н., доц. Васильева М. А.

(Проверил: к.т.н, доц. Балакина Е. П.)

Москва 2024

Оглавление

[1 Решение задачи 3](#_Toc151483877)

[1.1 Формулировка задачи 3](#_Toc151483878)

[1.2 UML диаграмма алгоритма задание 4](#_Toc151483879)

[1.3 Текст программы на языке С++ задание 5](#_Toc151483880)

[1.4 Результаты выполнения программы 11](#_Toc151483881)

[1.5 Выполнение тестовых примеров задание 12](#_Toc151483881)

[1.6 Отметка о выполнении задания в веб-хостинге системы контроля версий задание 14](#_Toc151483882)

1. Решение задачи

**1.1 Формулировка** **задачи**

**Вариант 2. Линейный односвязный список**

Создать решение (Solution), которое минимально содержит три проекта (Projects): исполняемый (Console Application), библиотеку классов (Library), и модульные тесты (Tests). Разработать библиотеку классов по заданному варианту. Важно! Библиотека классов не должна зависеть от потоков ввода-вывода. Каждый класс необходимо размещать в отдельных двух файлах, снабжённых «говорящим именем» и специальными расширениями: \*.h для заголовочных файлов (Header), содержащих API класса, и \*.cpp для компилируемых (Source), содержащих реализацию класса. В запускаемом проекте требуется создать файл main.cpp, содержащий точку входа в демонстрационную программу – главную функцию (main). В рамках данной функции показать работу с коллекцией.

Реализовать библиотеку заданной (по вариантам) структуры данных целых чисел. Предусмотреть конструктор со списком инициализации (https://en.cppreference.com/w/cpp/utility/initializer\_list). Реализовать все конструкторы, создаваемые компилятором по умолчанию, реализовать деструктор. Предусмотреть методы вывода в строку содержимого коллекции. Переопределить операторы присваивания, сдвига влево и сдвига вправо. Предусмотреть метод определения наличия элементов коллекции (пустая ли коллекция?).

1.2 UML диаграмма алгоритма задание

UML диаграмма представлена ниже (Рисунок 1).

Изображение выглядит как текст, чек, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1– UML диаграмма

1.3 Текст программы на языке С++ задание

// Файл MyLinkedList.h

#include <iostream>

#include <string>

/\*\*

\* @brief - Создаём класс MyLinkedList

\*/

class MyLinkedList

{

private:

struct Node

{

int data;

Node\* next = nullptr;

};

Node\* head;

public:

/\*\*

\* @brief - Конструктор класса MyLinkedList

\*/

MyLinkedList();

/\*\*

\* @brief - Конструктор, который создает список и заполняет её значениями

\* @param initList - Значения для инициализации

\*/

MyLinkedList(std::initializer\_list<int> initList);

/\*\*

\* @brief - Конструктор копирования для класса MyLinkedList

\* @param other - Ссылка на другой объект MyLinkedList, который будет скопирован

\*/

MyLinkedList(const MyLinkedList& other);

/\*\*

\* @brief - Деструктор для очистки объекта списка

\*/

~MyLinkedList();

/\*\*

\* @brief - Конструктор перемещения

\* @other - Ссылка на другой объект MyLinkedList, который будет перемещён

\*/

MyLinkedList(MyLinkedList&& other) noexcept;

/\*\*

\* @brief - Оператор перемещения

\* @param other - Ссылка на другой объект MyLinkedList, который будет перемещён

\* @return Ссылка на элемент списка, который был перемещён

\*/

MyLinkedList& operator=(const MyLinkedList& other);

/\*\*

\* @brief - Проверка списка

\*/

bool isEmpty() const;

std::string toString() const;

/\*\*

\* @brief - Функция добавления элемента в конец списка

\* @param value - Значение, которое будет добавлено в список

\*/

void push\_back(int value);

/\*\*

\* @brief - Функция добавления элемента в начало списка

\* @param value - Значение, которое будет добавлено в список

\*/

void push\_front(int value);

/\*\*

\* @brief - Функция удаления последнего элемента из списка

\*/

void pop\_back();

/\*\*

\* @brief - Функция удаления первого элемента из списка

\*/

void pop\_front();

/\*\*

\* @brief - Функция вставки элемента по указанному индексу

\* @param idx - Индекс элемента в списке

\* @param elem - Элемент, который нужно найти

\*/

void insert(int idx, int elem);

/\*\*

\* @brief - Функция удаления элемента по указанному индексу

\* @param idx - Индекс элемента в списке

\*/

void remove(int idx);

/\*\*

\* @brief - Оператор сдвига

\*/

friend std::ostream& operator<<(std::ostream& os, const MyLinkedList& list);

};

// Файл MyLinkedList.cpp

#include "MyLinkedList.h"

#include <sstream>

MyLinkedList::MyLinkedList() : head(nullptr) {}

MyLinkedList::MyLinkedList(std::initializer\_list<int> initList) : MyLinkedList()

{

for (auto& value : initList)

{

push\_back(value);

}

}

MyLinkedList::MyLinkedList(const MyLinkedList& other) : MyLinkedList()

{

Node\* temp = other.head;

while (temp != nullptr)

{

push\_back(temp->data);

temp = temp->next;

}

}

MyLinkedList& MyLinkedList::operator=(const MyLinkedList& other)

{

MyLinkedList temp(other);

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

return \*this;

}

MyLinkedList::~MyLinkedList()

{

while (head != nullptr)

{

Node\* temp = head;

head = head->next;

delete temp;

}

}

MyLinkedList::MyLinkedList(MyLinkedList&& other) noexcept

{

head = other.head;

other.head = nullptr;

}

bool MyLinkedList::isEmpty() const

{

return head == nullptr;

}

std::string MyLinkedList::toString() const

{

std::stringstream result;

Node\* temp = head;

while (temp != nullptr)

{

result << temp->data << " ";

temp = temp->next;

}

return result.str();

}

void MyLinkedList::push\_back(int value)

{

Node\* newNode = new Node{ value, nullptr };

if (head == nullptr)

{

head = newNode;

}

else

{

Node\* temp = head;

while (temp->next != nullptr)

{

temp = temp->next;

}

temp->next = newNode;

}

}

void MyLinkedList::push\_front(int value)

{

Node\* newNode = new Node{ value, head };

head = newNode;

}

void MyLinkedList::pop\_back()

{

if (head == nullptr)

{

return;

}

if (head->next == nullptr)

{

delete head;

head = nullptr;

return;

}

Node\* temp = head;

while (temp->next->next != nullptr)

{

temp = temp->next;

}

delete temp->next;

temp->next = nullptr;

}

void MyLinkedList::pop\_front()

{

if (head == nullptr)

{

return;

}

Node\* temp = head;

head = head->next;

delete temp;

}

std::ostream& operator<<(std::ostream& os, const MyLinkedList& list)

{

os << list.toString();

return os;

}

void MyLinkedList::insert(int idx, int elem)

{

if (idx < 0) throw;

size\_t index = idx;

Node\* current = head;

size\_t curr\_index = 0;

while (curr\_index - 1 < index)

{

current = current->next;

curr\_index++;

}

Node\* tmp = current->next;

current->next = new Node();

current = current->next;

current->data = elem;

current->next = tmp;

}

void MyLinkedList::remove(int idx)

{

if (idx < 0) throw;

size\_t index = idx;

Node\* current = head;

size\_t curr\_index = 0;

while (curr\_index - 1 < index)

{

current = current->next;

curr\_index++;

}

Node\* tmp = current->next->next;

delete current->next;

current->next = tmp;

}

// Файл main.cpp

#include <iostream>

#include "..//Solver/MyLinkedList.h"

/\*\*

\* @brief Точка входа в программу

\* @return 0 в случае успеха

\*/

int main()

{

std::setlocale(NULL, "ru\_RU.UTF-8");

MyLinkedList list = { 1, 2, 3, 4 };

std::cout << "Первоначальный список: " << list << std::endl;

list.push\_back(5);

std::cout << "Список после push\_back(5): " << list << std::endl;

list.pop\_front();

std::cout << "Список после pop\_front(): " << list << std::endl;

list.pop\_back();

std::cout << "Список после pop\_back(): " << list << std::endl;

list.insert(0, 8);

std::cout << "Список после insert(0, 8): " << list << std::endl;

list.remove(3);

std::cout << "Список после remove(3): " << list << std::endl;

return 0;

}

1.4 Результаты выполнения программы

Результаты выполнения программы в C++ представлена ниже (Рисунок 2).

Изображение выглядит как текст, снимок экрана, дизайн

Автоматически созданное описание

Рисунок 2 – Результаты выполнения программы

1.5 Выполнение тестовых примеров задание

Выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 3–6).

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 3 – Результаты выполнения тестов

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 4 – Результаты выполнения тестов

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 5 – Результаты выполнения тестов

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 6 – Результаты выполнения тестов

1.6 Отметка о выполнении задания в веб-хостинге системы контроля версий задание

Ниже представлено доказательство того, что задание было принято. (Рисунок 7)

Изображение выглядит как текст, снимок экрана, линия, Шрифт

Автоматически созданное описание

Рисунок 7 – Approve задачи