**САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО**

**ИНСТИТУТ КОМПЬЮТЕРНЫХ НАУК И ТЕХНОЛОГИЙ**

**ВЫСШАЯ ШКОЛА ПРОГРАММНОЙ ИНЖЕНЕРИИ**

**Контрольная работа №1**

**Вариант 3**

**по предмету «Алгоритмы и структуры данных»**

**Выполнил: студент гр. 5130904/30002 Севостьянова А.В.**

**Руководитель Череповский Д.К.**

**Санкт-Петербург**

**2024 г.**

1. **Общая постановка задачи**

1. Класс двусвязный список (из целых чисел), шаблон.

2. Класс должен иметь конструктор по умолчанию и деструктор.

3. Копирование и присваивание запретить.

4. Класс должен иметь метод insert для добавления элемента в «хвост» или в «голову» списка (порядок следования элементов в списке при этом будет соответствовать порядку их добавления или будет обратным; указывается в варианте задания).

3) метод reverse, переворачивающий список в обратном порядке (данный метод не будет вызывать ошибку при пустом списке, а только выводить сообщение о невозможности своего выполнения, далее программа будет работать корректно)

1. **Код программы**

main.cpp

#include"DoubleLinkedList.h"

#include<string>

int main()

{

setlocale(LC\_ALL, "ru");

std::cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_int\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

DoubleLinkedList<int> list1;

list1.print();

list1.reverce();

list1.insert(1);

list1.insert(2);

list1.insert(3);

list1.insert(4);

std::cout << "List:\t";

list1.print();

list1.reverce();

std::cout << "отсортированнный List:\t";

list1.print();

std::cout << "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_char\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

DoubleLinkedList<char> list2;

list2.print();

list2.reverce();

list2.insert('a');

list2.insert('b');

list2.insert('c');

list2.insert('d');

std::cout << "List:\t";

list2.print();

list2.reverce();

std::cout << "отсортированнный List:\t";

list2.print();

std::cout << "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_string\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

DoubleLinkedList<std::string> list3;

list3.print();

list3.reverce();

list3.insert("apple");

list3.insert("banana");

list3.insert("cat");

list3.insert("dog");

std::cout << "List:\t";

list3.print();

list3.reverce();

std::cout << "отсортированнный List:\t";

list3.print();

return 0;

}

DoubleLinkedList.h

#pragma once

#include<iostream>

template <typename T>

class DoubleLinkedList

{

private:

struct Node

{

T key;

Node\* next;

Node\* previous;

Node(T value) : key(value), next(nullptr), previous(nullptr) {};

};

Node\* head;

Node\* tail;

public:

DoubleLinkedList() : head(nullptr), tail(nullptr) {};

~DoubleLinkedList()

{

Node\* ptr = head;

while (head)

{

head = head->next;

delete ptr;

ptr = head;

}

}

DoubleLinkedList(const DoubleLinkedList& list) = delete;

DoubleLinkedList& operator=(const DoubleLinkedList& list) = delete;

void reverce()

{

Node\* ptr = head;

Node\* temp = nullptr;

if (!head) std::cout<< "Метод reverce не может быть применен - список пуст\n";

else

{

while (ptr)

{

temp = ptr->previous;

ptr->previous = ptr->next;

ptr->next = temp;

ptr = ptr->previous;

}

if (temp != nullptr)

head = temp->previous;

}

}

void insert(const T value)

{

Node\* new\_element = new Node(value);

if (!head)

head = new\_element;

else

{

Node\* ptr = head;

while (ptr->next) ptr = ptr->next;

ptr->next = new\_element;

new\_element->previous = ptr;

}

}

void print() const

{

if (!head) std::cout << "Элементы еще не добавлены\n";

else

{

Node\* ptr = head;

while (ptr) {

std::cout << ptr->key << " ";

ptr = ptr->next;

}

std::cout << std::endl;

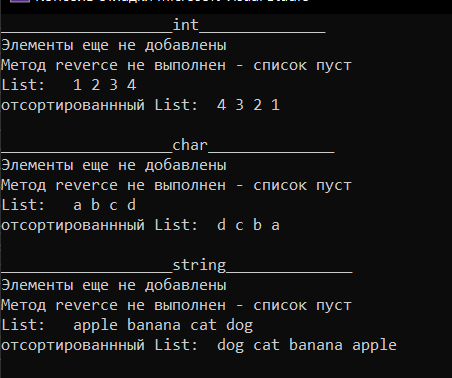
}

}

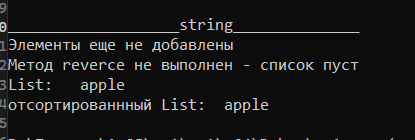
};

1. **Работа программы**

Работа методов insert и replace (элементы добавляются в конец списка, метод реверс не вызывает ошибки, на экран выводится информация о том, что метод не выполнен, т.к. список оказался пуст)

****

Если в списке один элемент

****