Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**Отчет по лабораторной работе №11.2**

Дисциплина: «Основы алгоритмизации и программирования»

Тема : "Двунаправленный список"

Выполнил работу:

Студент группы РИС-22-1Б

Батин В. В.

Проверил:

Доцент кафедры ИТАС

Полякова О.А.

Г. Пермь-2023

**Постановка задачи.**

Создать однонаправленный список, добавить в него k элементов с заданного номера, удалить из него k элементов с заданного номера, вывести список.

**Анализ задачи.**

Структура Node представляет узел списка, который содержит значение ключа (поле) и указатели на предыдущий и следующий узлы (поля prev и next соответственно).

Класс DoublyLinkedList представляет сам список и содержит методы для добавления, удаления и вставки элементов, а также для вывода списка на экран.

Метод add добавляет новый элемент в конец списка. Если список пустой, то новый элемент становится его головой (head), иначе происходит добавление нового элемента после последнего элемента и обновление указателя на хвост (tail).

Метод remove удаляет элементы из списка, начиная с заданного номера (start) и до количества элементов (count). Если start больше или равен размеру списка, то метод ничего не делает. Если count равен нулю, то метод также ничего не делает.

Метод insert добавляет новые элементы в список, начиная с заданного номера (start) и до количества (count). Если start больше размера списка, то метод ничего не делает.

Метод print выводит элементы списка на экран.

В функции main создается объект класса DoublyLinkedList, заполняется 10 элементами с помощью метода add, затем выводится список на экран.

**Код.**

#include <iostream>

struct Node {

int key;

Node\* prev;

Node\* next;

Node(int key) : key(key), prev(nullptr), next(nullptr) {}

};

class DoublyLinkedList {

public:

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

// Добавление элемента в конец списка

void add(int key) {

Node\* newNode = new Node(key);

if (head == nullptr) {

head = newNode;

}

else {

tail->next = newNode;

newNode->prev = tail;

}

tail = newNode;

size++;

}

// Удаление элементов из списка, начиная с заданного номера

void remove(int start, int count) {

if (start >= size || count == 0) {

return;

}

Node\* curr = head;

for (int i = 0; i < start; i++) {

curr = curr->next;

}

for (int i = 0; i < count && curr != nullptr; i++) {

Node\* next = curr->next;

delete curr;

curr = next;

size--;

}

if (curr != nullptr) {

curr->prev = curr->prev->prev;

if (curr->prev == nullptr) {

head = curr;

}

else {

curr->prev->next = curr;

}

}

else {

tail = tail->prev;

if (tail == nullptr) {

head = nullptr;

}

else {

tail->next = nullptr;

}

}

}

// Добавление элементов в список, начиная с заданного номера

void insert(int start, int count) {

if (start > size) {

return;

}

Node\* curr = head;

for (int i = 0; i < start; i++) {

curr = curr->next;

}

Node\* prev = curr->prev;

for (int i = 0; i < count; i++) {

Node\* newNode = new Node(rand() % 100);

newNode->prev = prev;

newNode->next = curr;

if (prev == nullptr) {

head = newNode;

}

else {

prev->next = newNode;

}

curr->prev = newNode;

prev = newNode;

size++;

}

}

// Вывод элементов списка на экран

void print() const {

Node\* curr = head;

while (curr != nullptr) {

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

curr = curr->next;

}

std::cout << std::endl;

}

private:

Node\* head;

Node\* tail;

int size;

};

int main() {

DoublyLinkedList list;

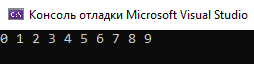
for (int i = 0; i < 10; i++) {

list.add(i);

}

list.print(); //

**Результат выполнения.**

****