Лабораторная работа №1

Вариант 16

Салихов А.Г. гр. Б04-191-1з

1.Постановка задачи.

Однонаправленный список.

Список должен содержать следующие операции:

1. Вставка элемента в конец (целочисленное значение)

Вставка должна корректно обрабатывать случай, когда список пуст (указатель на первый узел пуст)

1. Вставка элемента перед заданным индексом
2. Получение значения элемента по индексу
3. Удаление элемента по значению

В случае, если элемент с ключом не найден, функция должна вернуть *false*, иначе - *true*

1. Печать всех элементов списка
2. Освобождение памяти от структуры данных

2.Исходный код решения

﻿#include <iostream>

using namespace std;

struct Node

{

int value;

Node\* next = NULL;

};

Node\* add(Node\* root\_, int val) {

if (root\_ == NULL) {

Node\* newElem = new Node();

newElem->value = val;

return newElem;

}

else {

Node\* head = root\_;

while (root\_->next) {

root\_ = root\_->next;

}

Node\* newElem = new Node();

newElem->value = val;

root\_->next = newElem;

return head;

}

}

bool addAfter(Node\* after, int val) {

if (after == NULL) return false;

if (after->next == NULL) {

Node\* newElem = new Node();

newElem->value = val;

after->next = newElem;

return true;

}

else {

Node\* next = after->next;

Node\* newElem = new Node();

newElem->value = val;

after->next = newElem;

newElem->next = next;

}

}

Node\* find\_elem(Node\* root, int val) {

while (root->value != val) {

root = root->next;

if (!root) return NULL;

}

return root;

}

void printList(Node\* root) {

while (root) {

cout << root->value << " -> ";

root = root->next;

}

cout << "NULL" << endl;

}

bool Delete(Node\*\* Head, Node\* root, int val) {

Node\* forDelete = find\_elem(root, val);

if (forDelete == NULL)

{

return false;

}

else

{

if (forDelete == root)

{

\*Head = root->next;

delete forDelete;

return true;

}

while (root->next != forDelete)

{

root = root->next;

}

root->next = forDelete->next;

delete forDelete;

return true;

}

}

Node\* erase(Node\* root) {

Node\* next;

while (root) {

next = root;

root = root->next;

delete next;

}

return NULL;

}

int main()

{

Node\* root = 0;

root = add(root, 1);

root = add(root, 2);

root = add(root, 3);

root = add(root, 4);

printList(root);

Node\* f = find\_elem(root, 2);

addAfter(f, 13);

printList(root);

Delete(&root, root, 4);

printList(root);

root = erase(root);

printList(root);

}

3.Контрольный пример

