Лабораторная работа №2

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 {

//TODO: Esli znachenie uzhe est Node\* head = root\_;

while (root\_->next) { // Нахождение последнего элемента в списке

root\_ = root\_->next;

}

Node\* newElem = new Node(); // Создание нового элемента newElem->value = val;

root\_->next = newElem; // Последний элемент списка указывает на новый созданный return head; // возвращаем голову списка

}

}

bool addAfter(Node\* after, int val) { if (after == NULL) return false;// если узла нет то false if (after->next == NULL) { //если элемент последний

Node\* newElem = new Node(); // Создание нового элемента newElem->value = val;

after->next = newElem;//текущий элемент указывает на новый return true;//возвращает true

}

else {

Node\* next = after->next;// присваеваем next след значение

Node\* newElem = new Node(); // Создание нового элемента newElem->value = val;

after->next = newElem;//присваеваем текущему элементу указатель на новый

элемент newElem->next = next;//присваеваем к новому элементу на указатель предыдущего

}

}

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

while (root->value != val) {// пока значение текущего не равно значению которое мы ищем root = root->next;// переход к след элементу if (!root) return NULL;//если не нышла -> выходим

}

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); //присваеаем forDelete найденный элемент if (forDelete == NULL)// пройдется по листу и если нет этого элемента вернет NULL

{

return false;//вернет ложь

}

else

{

if (forDelete == root) // если forDelete первый

{

\*Head = root->next;//присваеваем голове указатель на след элемент delete forDelete; return true;

}

while (root->next != forDelete)

{

root = root->next; //находим предыдущий элемент удаляемого

}

root->next = forDelete->next;// связываем предыдущий и следующий элемент удаляемого

delete forDelete; return true;

}

}

Node\* erase(Node\* root) {

Node\* next; while (root) {

next = root;// присваиваем next указатель на текущий элемент root = root->next; //присваиваем root значение следующего

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

