**Singly Linked List**

**Heading class**

#pragma once

#include "iostream"

#ifndef INT\_LINKED\_LIST

#define INT\_LINKED\_LIST

class LLNode {

public:

LLNode() {

next = 0;

}

LLNode(int el, LLNode \*p = 0) {

info = el;

next = p;

}

int info;

LLNode \* next;

};

class LinkedList {

public:

LinkedList() {

head = tail = 0;

}

~LinkedList();

int isEmpty() {

return head == 0;

}

int getValue(int n);

void addToHead(int el);

void addToTail(int el);

void addAfter(int el, int n);

int deleteFromHead();

int deleteFromTail();

void deleteNode(int n);

bool isInList(int n) const;

private:

LLNode \*head, \*tail;

};

#endif

**CPP class**

#include "iostream"

#include "sstream"

#include "stdafx.h"

#include "CreateList.h"

#include "string"

using namespace std;

LinkedList::~LinkedList() {

for (LLNode \*p; !isEmpty();) {

p = head->next;

delete head;

head = p;

}

}

int LinkedList::getValue(int n) {

int index = 0;

LLNode \*node = head;

while (node != NULL) {

if (index == n) {

break;

}

else {

index++;

node = node->next;

}

}

return node->info;

}

void LinkedList::addToHead(int el) {

head = new LLNode(el, head);

if (tail == 0)

tail = head;

}

void LinkedList::addToTail(int el) {

if (tail != 0) {

tail->next = new LLNode(el, head);

tail = tail->next;

}

else head = tail = new LLNode(el);

}

//insert into linked list - homework question

void LinkedList::addAfter(int el, int n) {

LLNode \*node = head;

LLNode \*nNode = head->next;

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

node = node->next;

nNode = nNode->next;

if (i >= n) {

LLNode \*selected = nNode;

selected->info = el;

node->next = selected;

selected->next = nNode;

n++;

}

}

}

int LinkedList::deleteFromHead() {

int el = head->info;

LLNode \*tmp = head;

if (head == tail) {

head = tail = 0;

}

else {

head = head->next;

delete tmp;

return el;

}

}

int LinkedList::deleteFromTail() {

int el = tail->info;

if (head == tail) {

delete head;

head = tail = 0;

}

else {

LLNode \*tmp;

for (tmp = head; tmp->next != tail; tmp = tmp->next);

delete tmp;

tail->next = 0;

}

return el;

}

//delete the nth node - homework question

void LinkedList::deleteNode(int el) {

if (head != 0) {

if (head == tail && el == head->info) {

delete head;

head = tail = 0;

}

else if (el == head->info) {

LLNode \*tmp = head;

head = head->next;

delete tmp;

}

else {

LLNode \*pred, \*tmp;

for (pred = head; tmp = head->next; tmp != 0 && !(tmp->info == el)){

if (tmp->info == el) {

pred->next = tmp->next;

delete tmp;

}

}

}

}

}

bool LinkedList::isInList(int el) const {

LLNode \*tmp;

for (tmp = head; tmp != 0 && !(tmp->info == el); tmp = tmp->next) {

std::cout << "this contains "<< el << "\n";

return tmp != 0;

}

}

//start main method

int main()

{

LinkedList list;

int n = 0;

int el = 0;

int delElement = 0;

int count = 6;

int count2 = 3;

int lcCount = 0;

string answer;

list.addToTail(1);

list.addToTail(2);

list.addToTail(3);

list.addToTail(4);

list.addToTail(5);

list.addToTail(6);

cout << "The default list is." << "\n";

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

cout << "the value at location " << i << " in list1 is " << list.getValue(i) << "\n";

}

//Append to new linked list destructively - homework question 3

cout << "\n" << "list2" << "\n";

LinkedList list2;

list2.addToTail(3);

list2.addToTail(2);

list2.addToTail(1);

for (int h = 0; h < count2; h++) {

cout << "the value for list2 at " << h << " is " << list2.getValue(h) << "\n";

}

list2.deleteFromHead();

list2.addToHead(4);

cout << "after appending 4 to the head of the list" << "\n";

for (int m = 0; m < count2; m++) {

cout << "the value for list2 at " << m << " is " << list2.getValue(m) << "\n";

}

//Copy list - homework question 4

LinkedList listCopy;

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

listCopy.addToTail(list.getValue(k));

lcCount++;

}

cout << "\n" << "listcopy" << "\n";

for (int l = 0; l < lcCount; l++) {

cout << "the value for listCopy at " << l << " is " << listCopy.getValue(l) << "\n";

}

//reverse a list

LinkedList reverse;

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

reverse.addToHead(list.getValue(x));

}

cout << "\n" << "reversed list" << "\n";

for (int y = 0; y < lcCount; y++) {

cout << "the value for reverse at " << y << " is " << reverse.getValue(y) << "\n";

}

//insert into ith node - homework question 2

cout << "What number do your want to add?";

cin >> el;

cout << "Where would you like to insert your integer?";

cin >> n;

list.addAfter(el, n);

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

cout << "the value at location " << i << " is "<< list.getValue(i) << "\n";

}

//delete ith node - homework question 1

cout << "Deleting value 1 from list.";

list.deleteNode(delElement);

count--;

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

cout << "the value at location " << i << list.getValue(i) << "\n";

}

return 0;

}

**Doubly Linked List**

**Circular Linked List**