//Singly Linked List

//node.h

#pragma once

class CNode

{

private:

int data;

CNode \*next;

public:

CNode();

CNode(int);

void setData(int);

int getData()const;

void setNext(CNode\*);

CNode\* getNext()const;

//~CNode();

};

//node.cpp

#include "pch.h"

#include "node.h"

#include<iostream>

using namespace std;

CNode::CNode()

{

this->data = 0;

this->next = NULL;

}

CNode::CNode(int data)

{

this->data = data;

this->next = NULL;

}

void CNode::setData(int data)

{

this->data = data;

}

int CNode::getData()const

{

return data;

}

void CNode::setNext(CNode \*temp)

{

this->next = temp;

}

CNode\* CNode::getNext()const

{

return this->next;

}

//mylinklist.h

#include"node.h"

class CMyinkedList

{

private:

CNode \*head;

public:

CMyinkedList();

int isEmpty();

bool addFirst(int);

bool insert(int);

bool insertNode(int, int);

bool removefromposition(int);

bool removeByValue(int);

void reversLinkdlist();

void Display();

~CMyinkedList();

};

//mylinklist.cpp

#include "pch.h"

#include "myinkedList.h"

#include"node.h"

#include<iostream>

using namespace std;

CMyinkedList::CMyinkedList()

{

this->head = NULL;

}

int CMyinkedList::isEmpty()

{

if (this->head == NULL)

return 0;

else

return 1;

}

bool CMyinkedList::insert(int data)

{

CNode \*newNode = new CNode(data); //dyanamic memory allocation

if (newNode == NULL)

return false;

if (this->head == NULL)

{

this->head = newNode;

return true;

}

else

{

CNode \* temp;

temp = this->head;

while (temp->getNext() != NULL)

{

temp = temp->getNext();

}

temp->setNext(newNode);

return true;

}

}

bool CMyinkedList::insertNode(int data, int position)

{

if (position <= 0)

return false;

CNode \*newNode = new CNode(data);

CNode \*trav = this->head;

for (int i = 1; i < position - 1; i++)

{

if (trav == NULL)

{

delete newNode;

return false;

}

else

{

trav = trav->getNext();

}

}

newNode->setNext(trav->getNext());

trav->setNext(newNode);

return true;

}

bool CMyinkedList::removefromposition(int position)

{

if (position > 1)

{

CNode \*trav = this->head;

for (int i = 1; i < position - 1; i++)

{

trav = trav->getNext();

if (trav == NULL)//its important to check

{

return false;

}

}

if (trav->getNext() == NULL)//it is important to check

return false;

CNode \*ptrNode = trav->getNext();

trav->setNext(ptrNode->getNext());

delete ptrNode;

return true;

}

}

void CMyinkedList::reversLinkdlist()

{

CNode \*current = this->head;

CNode \*prev = NULL;

CNode \*next = current->getNext();

while (current != NULL)

{

next = current->getNext();

current->setNext(prev);

prev = current;

current = next;

}

this->head = prev;

}

bool CMyinkedList::addFirst(int data)

{

if (this->head == NULL)

return false;

else

{

CNode \*newNode = new CNode(data);

newNode->setNext(this->head);

this->head = newNode;

return true;

}

}

void CMyinkedList::Display()

{

CNode \* temp = this->head;

while (temp != NULL)

{

cout << temp->getData()<<" ";

temp = temp->getNext();

}

}

bool CMyinkedList::removeByValue(int value)

{

CNode \*temp; CNode \*prev;

prev=temp =this->head;

while (temp->getData() != value)

{

prev = temp;

temp = temp->getNext();

if (temp == NULL)

{

return false;

}

}

prev->setNext(temp->getNext());

delete temp;

return true;

}

CMyinkedList::~CMyinkedList()

{

CNode \*del;

while (head != NULL)

{

del = this->head;

head = head->getNext();

delete del;

}

}

//linklistmain.cpp

#include "pch.h"

#include"myinkedList.h"

#include <iostream>

using namespace std;

#include<conio.h>

int main()

{

CMyinkedList list;

int choice,data,position,val;

do

{

cout << "\n1.Insert \n2.InsertData \n3.Remove from position \n4.Display \n5.Reverse Linklist \n6.Add Fiirst \n7.Remove by value \n8.EXIT";

cout << "\nEnter the choice :";

cin >> choice;

switch (choice)

{

case 1:

cout << "\nEnter the data :";

cin >> data;

list.insert(data);

break;

case 2:

cout << "\nEnter the data :";

cin >> data;

cout << "\nEnter the position :";

cin >> position;

list.insertNode(data,position);

break;

case 3:

cout << "\nEnter the position..";

cin >> position;

list.removefromposition(position);

break;

case 4:

list.Display();

break;

case 5:list.reversLinkdlist(); break;

case 6:

cout << "\nEnter the data :";

cin >> data;

list.addFirst(data); break;

case 7:

cout << "Enter value to delete : ";

cin >> val;

list.removeByValue(val); break;

case 8:exit(0);

default:cout << "Wrong input....";

}

} while (choice != 8);

\_getch();

return 0;

}