//node.h

#pragma once

class Node

{

Node \*prev;

int data;

Node \*next;

public:

Node();

Node(int);

~Node();

void setData(int);

int getData() const;

void setNext(Node \*);

Node \*getNext() const;

void setPrev(Node \*);

Node \*getPrev()const;

};

//node.cpp

#include "pch.h"

#include "node.h"

#include<iostream>

using namespace std;

Node::Node()

{

this->prev = NULL;

this->data = 0;

this->next = NULL;

}

Node::Node(int data)

{

this->prev = NULL;

this->data = data;

this->next = NULL;

}

void Node::setData(int data)

{

this->data = data;

}

int Node::getData()const

{

return data;

}

void Node::setNext(Node \* next)

{

this->next = next;

}

Node \* Node::getNext()const

{

return next;

}

void Node::setPrev(Node \*prev)

{

this->prev = prev;

}

Node \* Node::getPrev()const

{

return prev;

}

Node::~Node()

{

}

//doublylinkedlist.h

#include"node.h"

#include<iostream>

using namespace std;

class CDoublyLinkedlist

{

Node \*head;

Node \*tail;

public:

CDoublyLinkedlist();

~CDoublyLinkedlist();

bool insert(int);

void Display();

void revDisplay();

bool insertAtPosition(int,int);

bool removeByValue(int);

bool removeFromPos(int);

};

// doublylinkedlist.cpp

#include "pch.h"

#include "doublylinkedlist.h"

CDoublyLinkedlist::CDoublyLinkedlist()

{

this->head = NULL;

this->tail = NULL;

}

bool CDoublyLinkedlist::insert(int data)

{

Node \*newNode = new Node(data);

if (newNode == NULL)

return false;

if (this->head == NULL && this->tail == NULL)

{

this->head = newNode;

this->tail = newNode;

return true;

}

else

{

Node \*trav = this->head;

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

{

trav = trav->getNext();

}

trav->setNext(newNode);

newNode->setPrev(trav);

this->tail = newNode;

return true;

}

}

void CDoublyLinkedlist::Display()

{

Node \*trav = this->head;

while (trav != NULL)

{

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

trav = trav->getNext();

}

}

void CDoublyLinkedlist::revDisplay()

{

Node \*trav;

trav = this->tail;

while (trav!=NULL)

{

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

trav = trav->getPrev();

}

}

bool CDoublyLinkedlist::insertAtPosition(int pos,int data)

{

Node \*newNode = new Node(data);

Node \*trav = this->head;

if (newNode==NULL)

{

return false;

}

else

{

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

{

trav = trav->getNext();

}

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

trav->getNext()->setPrev(newNode);//trav->next->prev = newNode

trav->setNext(newNode);

newNode->setPrev(trav);

return true;

}

}

bool CDoublyLinkedlist::removeFromPos(int pos)

{

Node \*trav = this->head;

if (pos <= 0)

{

return false;

}

else

{

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

{

trav = trav->getNext();

}

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

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

ptrNode->getNext()->setPrev(trav);//ptr->next->prev = trav

delete ptrNode;

return true;

}

}

bool CDoublyLinkedlist::removeByValue(int data)

{

Node \*trav = this->head;

Node \*back = head;

while (trav->getData() != data)

{

back = trav;

trav = trav->getNext();

}

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

trav->getNext()->setPrev(back);

delete trav;

return true;

}

CDoublyLinkedlist::~CDoublyLinkedlist()

{

}

//dllmain.cpp

#include "pch.h"

#include "doublylinkedlist.h"

#include "pch.h"

#include <iostream>

using namespace std;

#include<conio.h>

int main()

{

CDoublyLinkedlist dll;

int ch,data,pos;

do

{

cout << "\n1.INSERT \n2.DISPLAY \n3.Reverse Display \n4.Insert at Pos \n5.Remove from pos \n";

cout << "6.Remobe by value \n7.EXIT";

cout << "Enter ch : ";

cin >> ch;

switch (ch)

{

case 1:

cout << "Enter data : ";

cin >> data;

dll.insert(data);

break;

case 2:dll.Display(); break;

case 3:dll.revDisplay(); break;

case 4:

cout << "Enter data : ";

cin >> data;

cout << "Enter pos : ";

cin >> pos;

dll.insertAtPosition(pos,data); break;

case 5:

cout << "Enter pos : ";

cin >> pos;

dll.removeFromPos(pos);

break;

case 6:

cout << "Enter data : ";

cin >> data;

dll.removeByValue(data);

break;

case 7:exit(0);

default:cout << "Wrong input choice ...\n";

}

} while (ch!=7);

\_getch();

return 0;

}