#ifndef ELEMENT\_H

#define ELEMENT\_H

class element

{

private:

int data;

element \*next;

element \*prev;

public:

element();

element(int);

virtual ~element();

int Getdata() { return data; }

void Setdata(int val) { data = val; }

element \* GetNext() { return next; }

void SetNext(element\* val) { next = val; }

element \* GetPrev() { return prev; }

void SetPrev(element\* val) { prev = val; }

protected:

};

#endif // ELEMENT\_H

#include "element.h"

element::element()

{

//ctor

this->data=0;

this->next = nullptr;

this->prev = nullptr;

}

element::element(int data)

{

//ctor

this->data=data;

this->next=nullptr;

this->prev = nullptr;

}

element::~element()

{

//dtor

}

#ifndef LINKEDLIST\_H

#define LINKEDLIST\_H

#include "element.h"

class linkedlist

{

private:

element\* head;

element\* tail;

public:

linkedlist();

virtual ~linkedlist();

element\* Gethead() { return head; }

void Sethead(element\* val) { head = val; }

element\* Gettail() { return tail; }

void Settail(element\* val) { tail = val; }

void InsertFirst(element\*);

void InsertTail(element\*);

void InsertElementAfterNodep(element\*, int);

void DeleteFirst();

void DeleteTail();

void DeleteNodep(element\*);

void Travel();

protected:

};

#endif // LINKEDLIST\_H

#include "linkedlist.h"

#include <iostream>

using namespace std;

linkedlist::linkedlist()

{

//ctor

this->head=nullptr;

this->tail=nullptr;

}

linkedlist::~linkedlist()

{

//dtor

}

void linkedlist::InsertFirst(element\* e){

if(this->head==nullptr)

this->head=this->tail=e;

else{

e->SetNext(this->head);

this->head->SetPrev(e);

this->head=e;

}

}

void linkedlist::InsertTail(element\*e){

if(this->head==nullptr)

this->head=this->tail=e;

else{

this->tail->SetNext(e);

e->SetPrev(this->tail);

this->tail=e;

}

}

void linkedlist::InsertElementAfterNodep(element \*e, int val)

{

element \*p = new element(val);

if(e->Getdata() == this->head->Getdata() && this->head->GetNext() == nullptr)

{

return InsertTail(p);

}

if(e->Getdata() == this->tail->Getdata() && this->tail->GetNext() == nullptr)

{

return InsertTail(p);

}

element \*k = this->head;

while(k != nullptr)

{

if(k->Getdata() == e->Getdata())

{

p->SetNext(k->GetNext());

k->GetNext()->SetPrev(p);

k->SetNext(p);

p->SetPrev(k);

}

k = k->GetNext();

}

}

void linkedlist::Travel(){

element\* p=this->head;

while(p!=nullptr){

cout<<p->Getdata()<<"\t";

p=p->GetNext();

}

}

void linkedlist::DeleteFirst(){

if(this->head == nullptr) return;

else

{

this->head=this->head->GetNext();

this->head->SetPrev(nullptr);

}

}

void linkedlist::DeleteTail()

{

if(this->head == nullptr) return;

if(this->head->GetNext() == nullptr) return DeleteFirst();

this->tail = this->tail->GetPrev();

this->tail->SetNext(nullptr);

}

void linkedlist::DeleteNodep(element \*p)

{

if(this->head == nullptr)

{

return;

}

if(this->head->Getdata() == p->Getdata())

{

DeleteFirst();

return;

}

if(this->tail->Getdata() == p->Getdata())

{

DeleteTail();

return;

}

element \*k = this->head;

while(k != nullptr)

{

if(p->Getdata() == k->Getdata())

{

k->GetPrev()->SetNext(k->GetNext());

k->GetNext()->SetPrev(k->GetPrev());

return;

}

k = k->GetNext();

}

}

#include <iostream>

#include "linkedlist.h"

using namespace std;

int main()

{

linkedlist \*list\_=new linkedlist();

element \*e;

e=new element(9);

list\_->InsertTail(e);

e=new element(10);

list\_->InsertTail(e);

e=new element(8);

list\_->InsertTail(e);

list\_->Travel();

// list\_->DeleteTail();

list\_->InsertElementAfterNodep(new element(8), 11);

cout<<"\n";

list\_->Travel();

cout<<"\n";

list\_->DeleteNodep(new element(10));

list\_->Travel();

delete list\_;

delete e;

return 0;

}