#ifndef ELEMENT\_H

#define ELEMENT\_H

class ELEMENT

{

public:

ELEMENT();

ELEMENT(int);

virtual ~ELEMENT();

int Getdata() { return data;}

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

ELEMENT\* Getpointer() {return pointer;}

void Setpointer(ELEMENT \*val) {pointer = val;}

protected:

private:

int data;

ELEMENT \*pointer;

};

#endif // ELEMENT\_H

#include "ELEMENT.h"

ELEMENT::ELEMENT()

{

this->data = 0;

this->pointer = nullptr;

}

ELEMENT::ELEMENT(int data)

{

this->data = data;

this->pointer = nullptr;

}

ELEMENT::~ELEMENT()

{

//dtor

}

#ifndef LINKEDLIST\_H

#define LINKEDLIST\_H

#include "ELEMENT.h"

class LINKEDLIST

{

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();

void RemoveAll();

protected:

private:

ELEMENT \*head;

ELEMENT \*tail;

};

#endif // LINKEDLIST\_H

#include "LINKEDLIST.h"

#include<iostream>

using namespace std;

LINKEDLIST::LINKEDLIST()

{

this->head = nullptr;

this->tail = nullptr;

}

LINKEDLIST::~LINKEDLIST()

{

//dtor

}

void LINKEDLIST::InsertFirst(ELEMENT \*val)

{

if(this->head == nullptr)

{

this->head = this->tail = val;

}

else

{

val->Setpointer(this->head);

this->head = val;

//Sethead(val);

}

}

void LINKEDLIST::InsertTail(ELEMENT \*val)

{

if(this->head == nullptr)

{

this->head = this->tail = val;

}

else

{

this->tail->Setpointer(val);

this->tail = val;

}

}

void LINKEDLIST::InsertElementAfterNodep(ELEMENT \*p, int val)

{

// neu dslk chi co 1 phan tu thi them vao cuoi

ELEMENT \*e = new ELEMENT(val);

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

{

InsertTail(e);

}

else

{

ELEMENT \*k = this->head;

while(k != nullptr)

{

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

{

ELEMENT \*h = e;

h->Setpointer(k->Getpointer());

k->Setpointer(h);

}

k = k->Getpointer();

}

}

}

void LINKEDLIST::DeleteFirst()

{

if(this->head == nullptr)

{

return;

}

ELEMENT \*p = this->head;

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

delete p;

}

void LINKEDLIST::DeleteTail()

{

if(this->head == nullptr)

{

return;

}

if(this->head->Getpointer() == nullptr)

{

return DeleteFirst();

}

ELEMENT \*p = this->head;

while(p != nullptr)

{

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

{

delete this->tail;

p->Setpointer(nullptr);

this->tail = p;

return;

}

p = p->Getpointer();

}

}

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 \*tmp = new ELEMENT;

ELEMENT \*k = this->head;

while(k != nullptr)

{

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

{

delete k;

tmp->Setpointer(k->Getpointer());

return;

}

tmp = k;

k = k->Getpointer();

}

}

void LINKEDLIST::Travel()

{

ELEMENT \*p = this->head;

while(p != nullptr)

{

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

p = p->Getpointer();

}

}

void LINKEDLIST::RemoveAll()

{

ELEMENT \*p = this->head;

while(p != NULL)

{

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

delete p;

p = p->Getpointer();

}

}

#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();

cout << endl;

list->DeleteFirst();

list->Travel();

cout << endl;

list->InsertElementAfterNodep(new ELEMENT(10), 11);

list->InsertElementAfterNodep(new ELEMENT(8), 13);

list->Travel();

// cout << endl;

// list->DeleteNodep(new ELEMENT(13));

// list->Travel();

// cout << endl;

// list->DeleteTail();

// list->Travel();

return 0;

}