#include "Element.h"

NodeDaThuc::NodeDaThuc()

{

//ctor

this->heSo=0;

this->soMu=0;

this->next=nullptr;

}

NodeDaThuc::NodeDaThuc(int heSo, int soMu)

{

//ctor

this->heSo = heSo;

this->soMu = soMu;

this->next=nullptr;

}

NodeDaThuc::~NodeDaThuc()

{

//dtor

}

#ifndef ELEMENT\_H

#define ELEMENT\_H

class NodeDaThuc

{

private:

int heSo;

int soMu;

NodeDaThuc \*next;

public:

NodeDaThuc();

NodeDaThuc(int, int);

virtual ~NodeDaThuc();

int GetHeSo() { return heSo; }

void SetHeSo(int val) { heSo = val; }

int GetSoMu() { return soMu; }

void SetSoMu(int val) { soMu = val; }

NodeDaThuc\* GetNext() { return next; }

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

protected:

};

#endif // ELEMENT\_H

#ifndef POLYNOMIAL\_H

#define POLYNOMIAL\_H

#include "Element.h"

class Polynomial

{

public:

Polynomial();

virtual ~Polynomial();

NodeDaThuc\* Gethead() { return head; }

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

NodeDaThuc\* Gettail() { return tail; }

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

void InsertFirst(NodeDaThuc\*);

void InsertTail(NodeDaThuc\*);

void Input();

void PrintPoly();

void RutGon();

int Tinhfx(int x);

Polynomial operator+ (Polynomial);

Polynomial operator\* (Polynomial);

protected:

private:

NodeDaThuc \*head;

NodeDaThuc \*tail;

};

#endif // POLYNOMIAL\_H

#include "Polynomial.h"

#include<iostream>

#include<math.h>

using namespace std;

Polynomial::Polynomial()

{

this->head = nullptr;

this->tail = nullptr;

}

Polynomial::~Polynomial()

{

//dtor

}

void Polynomial::InsertFirst(NodeDaThuc \*val)

{

if(this->head == nullptr)

{

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

}

else

{

val->SetNext(this->head);

this->head = val;

}

}

void Polynomial::InsertTail(NodeDaThuc \*val)

{

if(this->head == nullptr)

{

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

}

else

{

this->tail->SetNext(val);

this->tail = val;

}

}

void Polynomial::Input()

{

int bac;

cout << "\nNhap bac cua da thuc: "; cin >> bac;

NodeDaThuc \*Donthuc;

while(bac--)

{

int HeSo;

cout << "\n\tNhap he so bac " << bac+1 << ":" ; cin >> HeSo;

if(HeSo == 0) continue;

Donthuc = new NodeDaThuc(HeSo, bac +1);

InsertTail(Donthuc);

}

int HeSoTuDo;

cout << "\n\tNhap he so tu do: "; cin >> HeSoTuDo;

if(HeSoTuDo != 0)

{

Donthuc = new NodeDaThuc(HeSoTuDo, 0);

InsertTail(Donthuc);

}

}

void Polynomial::PrintPoly()

{

NodeDaThuc\* p = this->head;

while (p->GetNext() != NULL) {

if (p->GetHeSo() != 0) {

cout << p->GetHeSo() << "x^" << p->GetSoMu() << ((p->GetNext()->GetHeSo() > 0) ? "+" : "");

}

p = p->GetNext();

}

cout << p->GetHeSo() << "x^" << p->GetSoMu() << endl;

}

int gcd(int a, int b)

{

if(b == 0) return a;

return gcd(b, a%b);

}

void Polynomial::RutGon()

{

int UCLN = this->head->GetHeSo();

NodeDaThuc \*p = this->head->GetNext();

while(p != nullptr)

{

UCLN = gcd(UCLN, p->GetHeSo());

p = p->GetNext();

}

NodeDaThuc \*k = this->head;

while(k != nullptr)

{

k->SetHeSo( k->GetHeSo() / UCLN);

k = k->GetNext();

}

}

int Polynomial::Tinhfx(int x)

{

int fx = 0;

NodeDaThuc \*p = this->head;

while(p != nullptr)

{

fx += p->GetHeSo() \* pow(x, p->GetSoMu());

p = p->GetNext();

}

return fx;

}

// nctt +

Polynomial Polynomial::operator+ (Polynomial d)

{

Polynomial \*kq = new Polynomial();

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

else if(d.head == nullptr ) return \*this;

NodeDaThuc \*p = this->head;

NodeDaThuc \*k = d.head;

while(p !=nullptr || k != nullptr){ // phu dinh p == null && k== null => dung demorgan

NodeDaThuc \*res = new NodeDaThuc();

if(p->GetSoMu()==k->GetSoMu()){

res->SetHeSo(p->GetHeSo()+k->GetHeSo());

res->SetSoMu(p->GetSoMu());

p = p->GetNext();

k = k->GetNext();

} else if(p->GetSoMu()>k->GetSoMu() || k==nullptr) {

res->SetHeSo(p->GetHeSo());

res->SetSoMu(p->GetSoMu());

p = p->GetNext();

} else {

res->SetHeSo(k->GetHeSo());

res->SetSoMu(k->GetSoMu());

k = k->GetNext();

}

kq->InsertTail(res);

}

return \*kq;

}

//nctt \*

Polynomial Polynomial::operator\* (Polynomial d)

{

Polynomial kq;

NodeDaThuc \*p = this->head;

while(p != nullptr )

{

Polynomial \*tmp = new Polynomial();

NodeDaThuc \*k = d.head;

while(k != nullptr)

{

NodeDaThuc \*res = new NodeDaThuc();

res->SetHeSo(p->GetHeSo() \* k->GetHeSo());

res->SetSoMu(p->GetSoMu() + k->GetSoMu());

tmp->InsertTail(res);

k = k->GetNext();

}

kq = kq + \*tmp;

p = p->GetNext();

}

return kq;

}

//Polynomial Polynomial::operator\*(Polynomial d) {

// Polynomial kq;

// NodeDaThuc \*p = this->head;

//

// while (p != nullptr) {

// Polynomial tmp;

// NodeDaThuc \*k = d.head;

// while (k != nullptr) {

// NodeDaThuc \*res = new NodeDaThuc();

// res->SetHeSo(p->GetHeSo() \* k->GetHeSo());

// res->SetSoMu(p->GetSoMu() + k->GetSoMu());

// tmp.InsertTail(res);

// k = k->GetNext();

// }

// kq = kq + tmp;

// p = p->GetNext();

// }

// return kq;

//}

#include <iostream>

#include "Polynomial.h"

using namespace std;

int main()

{

Polynomial \*DaThuc1 = new Polynomial();

Polynomial \*DaThuc2 = new Polynomial();

DaThuc1->Input();

DaThuc2->Input();

Polynomial \*DaThuc3 = new Polynomial();

\*DaThuc3 = \*DaThuc1 + \*DaThuc2;

//

// cout << "\nDa thuc sau khi rut gon: \n";

// DaThuc1->RutGon();

//

DaThuc3->PrintPoly();

//// DaThuc2->PrintPoly();

// cout << endl;

// cout << DaThuc1->Tinhfx(1);

return 0;

}