稀疏多项式代码

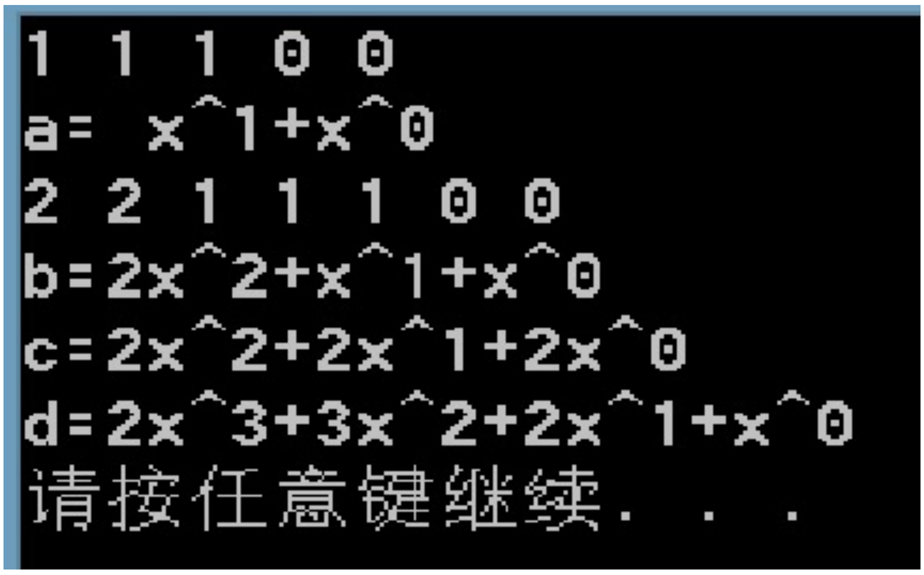
```
#include<iostream>
using namespace std;

//约定稀疏多项式输入升幂排列
template<class T>class Polynomial;
template<class T>ostream& operator<<(ostream& out, Polynomial<T>&a);

template<class T>
class Polynomial
{
private:
class Term
{
public:
T xishu;
int mi;
};
class Node
{
public:
Term data;
Node* next;
Node(T x = 0, unsigned y = 0, Node* p = 0)
{
data.xishu = x;
data.mi = y;
next = p;//修改
}
};
public:
Polynomial();
void myfree();//释放链表所有各节点
Polynomial<T>& operator=(const Polynomial<T>&);
Polynomial<T> operator+(const Polynomial<T>&);
Polynomial<T> operator*(const Polynomial<T>&);
void initial();
friend ostream& operator<< <T> (ostream& out, Polynomial<T>& a);

private:
int highestdigit;
Node* first;
int num;//用来判断多项式有几项,即链表节点数,在乘法运算里也要用到
};
template<class T>
Polynomial<T>::Polynomial()
```

```
{
first = new Node;
num = 0;
highestdigit = 0;
}

template<class T>
void Polynomial<T>::myfree() //释放链表所有节点
{
Node* p = first, *q;
while (p)
{
q = p->next;
delete p;
p = q;
}
first=NULL;//修改
num = 0; highestdigit = 0;//修改
}

template<class T>
Polynomial<T>& Polynomial<T>::operator=(const Polynomial<T>& second)
{
myfree();
highestdigit = second.highestdigit;
num = second.num;
first = second.first;

return *this;
}

template<class T>
Polynomial<T> Polynomial<T>::operator+(const Polynomial<T>& second)
{
Polynomial<T> sum;
Node* pa = first->next, *pb = second.first->next, *pc = sum.first;
while (pa&&pb)
{
if (pa->data.mi == pb->data.mi) // 指数相等时
{
T x = pa->data.xishu + pb->data.xishu;
if (x) // 相加完的系数不为0时
{
pc->next = new Node(x, pa->data.mi);
pc = pc->next;
++sum.num;
if(sum.highestdigit<pa->data.mi) sum.highestdigit = pa->data.mi;
}
```

```
pa = pa->next;
pb = pb->next;
}
else if (pa->data.mi > pb->data.mi){
pc->next = new Node(pa->data.xishu, pa->data.mi);
if(sum.highestdigit<pa->data.mi) sum.highestdigit = pa->data.mi;
++sum.num;
pc = pc->next;
pa = pa->next;
}
else{
pc->next = new Node(pb->data.xishu, pb->data.mi);
++sum.num;
if(sum.highestdigit<pb->data.mi) sum.highestdigit = pb->data.mi;
pc = pc->next;
pb = pb->next;
}
}
while (pa){
pc->next = new Node(pa->data.xishu, pa->data.mi);
++sum.num;
if(sum.highestdigit<pa->data.mi) sum.highestdigit = pa->data.mi;
pc = pc->next;
pa = pa->next;
}
while (pb){
pc->next = new Node(pb->data.xishu, pb->data.mi);
++sum.num;
if(sum.highestdigit<pb->data.mi) sum.highestdigit = pb->data.mi;
pc = pc->next;
pb = pb->next;
}
return sum;
}
template<class T>
Polynomial<T> Polynomial<T>::operator*(const Polynomial<T>& second){
Polynomial<T> multi, temp;
Node* pa = first->next, *pb = second.first->next, * pt;

if (num >= second.num){ //拿项数多的那个链表乘以项数少的那个
while(pb){ //pb是项数少的那个,
pt = temp.first;
while (pa){ //项数多的开始遍历
pt->next = new Node(pa->data.xishu*pb->data.xishu, pa->data.mi+pb->data.mi);//修改
temp.num++;//修改
if( temp.highestdigit<pt->next->data.mi ) temp.highestdigit=pt->next->data.mi;//修改
pt = pt->next;
pa = pa->next;
}
multi= temp+multi;
temp.myfree();
temp.first = new Node();
pa = first->next;
pb = pb->next;
}
}
else{
while(pa){
pt = temp.first;
while (pb){
pt->next = new Node(pa->data.xishu*pb->data.xishu, pa->data.mi+pb->data.mi);//修改
temp.num++;//修改
if( temp.highestdigit<pt->next->data.mi ) temp.highestdigit=pt->next->data.mi;//修改
pt = pt->next;
pb = pb->next;
}
multi = temp+multi;
temp.myfree();
temp.first = new Node();
pb = second.first->next;//修改
pa = pa->next;
}
}
return multi;
}
template<class T>
void Polynomial<T>::initial()
{
//释放链表，保留头节点
Node* p = first->next, *q;
while (p)
{
q = p->next;
delete p;
p = q;
}
first->next=NULL;//修改
p = first;

T a;
int b;
num = 0; highestdigit = 0;//修改
while (cin >> a&&a != 0 && cin >> b){ //输入a=0时结束输入
if (b >= highestdigit) highestdigit = b;
```

```
++num;
p->next = new Node(a, b);
p = p->next;
}
fflush(stdin);//修改
}

template<class T>
ostream& operator<<(ostream& out, Polynomial<T>& a)
{
Polynomial<T>::Node *p = a.first->next;

if (p == NULL) return out;//修改

if (p->data.xishu == 0)cout << endl;//修改
else{
if (p->data.xishu != 1)
out << p->data.xishu << "x^" << p->data.mi;
else out << "x^" << p->data.mi;
}
p = p->next;
while (p)
{
out << '+';
if (p->data.xishu != 1)
out << p->data.xishu << "x^" << p->data.mi;
else out << "x^" << p->data.mi;
p = p->next;
}
return out;
}

int main(){
Polynomial<int> a, b, c,d;
a.initial();
cout << "a= " << a << endl;
b.initial();
cout << "b=" << b << endl;
c = a + b;
cout << "c=" << c << endl;
d=a*b;
cout<<"d="<<d<<endl;
a.myfree();
b.myfree();
c.myfree();
d.myfree();
return 0;
}
```

运行结果：



```
1 1 1 0 0
a= x^1+x^0
2 2 1 1 1 0 0
b=2x^2+x^1+x^0
c=2x^2+2x^1+2x^0
d=2x^3+3x^2+2x^1+x^0
请按任意键继续. . .
```