**题目5：一元多项式的减法**

设有两个一元多项式A(x),B(x)，请完成运算A(x)+B(x)、A(x)-B(x)，要求多项式采用链表进行存储。另外该任务要求具有建立多项式链表以及输出多项式到屏幕的功能。

**代码：**

#include<stdio.h>

#include<stdlib.h>

#define LEN sizeof(struct Link)

struct Link

{

int c;

int e;

struct Link \*next;

};

int main()

{

void start();

struct Link \*creat(char ch);

struct Link \*addLink(struct Link \*head,struct Link \*pre);

struct Link \*cutLink(struct Link \*head,struct Link \*pre);

void print(struct Link \*p);

struct Link \*a,\*b;

int sign=-1;

start();

while(sign!=0)

{

scanf("%d",&sign);

switch(sign)

{

case 0:

break;

case 1:

{

printf("多项式相加:\n");

a=creat('A');

printf("A(x)=");

print(a);

b=creat('B');

print(b);

printf("C(x)=A(x)+B(x)=");

a=addLink(a,b);

print(a);

sign=-1;

start();

break;

}

case 2:

{

printf("多项式相减:\n");

a=creat('A');

printf("A(x)=");

print(a);

b=creat('B');

printf("B(x)=");

print(b);

printf("C(x)=A(x)-B(x)");

a=cutLink(a,b);

print(a);

sign=-1;

start();

break;

}

default:

{

printf("输入错误！\n");

start();

break;

}

}

}

}

void start()

{

printf("请选择操作：\n");

printf("0.退出\n");

printf("1.两个一元多项式相加\n");

printf("2.两个一元多项式想减\n");

}

struct Link \*creat(char ch)

{

void insert(struct Link \*head,struct Link \*inpt);

struct Link \*head,\*inpt;

int x;

int y;

head=(struct Link\*)malloc(LEN);

head->next=NULL;

printf("Please input the %C(x):(The end:0 0)\n",ch);

scanf("%d %d",&x,&y);

while(x!=0)

{

inpt=(struct Link\*)malloc(LEN);

inpt->c=x;

inpt->e=y;

inpt->next=NULL;

insert(head,inpt);

printf("请输入%C(x)的下一项：(The end:0 0)\n",ch);

scanf("%d %d",&x,&y);

}

return head;

}

void insert(struct Link \*head,struct Link \*inpt)

{

struct Link \*pre,\*now;

int signal=0;

pre=head;

if(pre->next==NULL)

pre->next=inpt;

else

{

now=pre->next;

while(signal==0)

{

if((inpt->e)<(now->e))

{

if(now->next==NULL)

{

now->next=inpt;

signal=-1;

}

else

{

pre=now;

now=pre->next;

}

}

else

if((inpt->e)>(now->e))

{

inpt->next=now;

pre->next=inpt;

signal=1;

}

else

{

now->c=now->c+inpt->c;

signal=1;

free(inpt);

if(now->c==0)

{

pre->next=now->next;

free(now);

}

}

}

}

}

struct Link \*addLink(struct Link \*head,struct Link \*pre)

{

struct Link \*inpt;

int flag=0;

while(flag==0)

{

if(pre->next==NULL)

flag=1;

else

{

pre=pre->next;

inpt=(struct Link \*)malloc(LEN);

inpt->c=pre->c;

inpt->e=pre->e;

inpt->next=NULL;

insert(head,inpt);

}

}

return head;

}

struct Link \*cutLink(struct Link \*head,struct Link \*pre)

{

struct Link \*inpt;

int flag=0;

while(flag==0)

{

if(pre->next==NULL)

flag=1;

else

{

pre=pre->next;

inpt=(struct Link \*)malloc(LEN);

inpt->c=0-pre->c;

inpt->e=pre->e;

inpt->next=NULL;

insert(head,inpt);

}

}

return head;

}

void print(struct Link \*p)

{

struct Link \*now;

int flag=0;

now=p->next;

if(p->next==NULL)

{

printf("\n");

return;

}

while(flag==0)

{

if((now->c)>0&&(p->next)!=now)

printf("+");

if(now->c==1);

else if(now->c==-1)

printf("-");

else

printf("%d",now->c);

if(now->e!=0)

printf("x^%d",now->e);

else if((now->c==1)||(now->c==-1))

printf("1");

if(now->next==NULL)

flag=1;

else

now=now->next;

}

printf("\n");

}

**运行结果：**

