**一、**【实验标题】

**实验2 一元多项式相加**

**二、【实验目的】**

1、了解链式存储结构的基本知识；

2、掌握算法思想和数据结构的描述；

3、结合一元多项式相加的运算规则。

**三、【实验环境】**

硬件：InterCore(TM)i5-6500 CPU3.20GHz,8GB内存

软件：Windows 7 64位、Mircrosoft Visual Studio 2010

1. **【实验内容】**

结合书上第41页的例子，采用链式存储结构，将两个线性链表表示的一元多项式相加，并输出。此一元多项式遵循多项式相加运算规则：对于两个一元多项式中存在指数相同的项时，其对应系数相加。合并后系数和为零时，删除“和多项式”中此项；合并后系数和不为零时，则构成“和多项式”中的一项。对于两个一元多项式中存在的指数不相同的项，则分别复抄到“和多项式”中去，原多项式保持不变。

**五、【实验程序】**

#include<stdio.h>

#include<stdlib.h>

struct node{ //储存每一项

double c; //系数coefficient

int e; //指数exponent

};

typedef struct Polynomial{ //多项式

struct node elem; //多项式元素

struct Polynomial \*next; //指向下一个多项式的结构体指针

}\*PolyList;

//创建一元多项式链表

void CreatePolyn(PolyList \*ha,int n)

{

double c1; //coe表示系数

int e1; //x2表示指数

PolyList p; //创建临时多项式链表

\*ha=NULL; //首先为空时ha指向空

while(n--){

scanf("%lf%d",&c1,&e1);

p=(PolyList)malloc(sizeof(struct Polynomial)); //创建多项式

p->elem.c=c1; //赋值新项的系数

p->elem.e=e1; //赋值新项的指数

p->next=\*ha; //新项p指向\*ha项，链的建立

\*ha=p; //指针的赋值，使得\*ha始终指向新创建的项

}

}

//输出一元多项式

void ShowSqList(PolyList ha,char op)

{

if(ha==NULL)

printf("多项式%c: F(x)=0",op);

else

{

printf("多项式%c: F(x)=%.0fX^(%d)",op,ha->elem.c,ha->elem.e);

ha=ha->next; //下一个

while(ha!=NULL)

{

if(ha->elem.c>0)

printf("+");

printf("%.0fX^(%d)",ha->elem.c,ha->elem.e);

ha=ha->next;//下一个

}

}

printf("\n");

}

//冒泡排序链表

void SortPolyn(PolyList head,int n,int option) //option==1 递减序 option==2 递增序

{

int i,j;

struct node temp; //建立临时比较项变量

PolyList p1,p2; //用两个差位多项式指针比较前后指数大小

for(p1=head,i=0;i<n-1;i++,p1=p1->next)

for(p2=p1->next,j=0;j<n-i-1;j++,p2=p2->next)

{

if(option==1){

if(p1->elem.e < p2->elem.e) //按指数从大到小排序

{

temp=p2->elem; //交换项元素，指针遍历不动

p2->elem=p1->elem;

p1->elem=temp;

}

}

else if(option==2){

if(p1->elem.e > p2->elem.e) //按指数从小到大排序

{

temp=p2->elem; //交换项元素，指针遍历不动

p2->elem=p1->elem;

p1->elem=temp;

}

}

}

}

//多项式相加

PolyList add(PolyList ha,PolyList hb) //对已排序的多项式ha,hb进行相加

{

int t=0; //t为系数和

int ptot=0; //hc总项数

PolyList p,hc=NULL; //多项式p,hc hc为返回值

while(ha&&hb) //ha,hb同时不为空

{

if(ha->elem.e==hb->elem.e)

{//两指数相同时，系数相加

t=ha->elem.c+hb->elem.c;

if(t)

{//系数和不为零，则插入和多项式hc链表中

p=(PolyList)malloc(sizeof(struct Polynomial)); //新建项

p->elem.c=t; //把系数和赋给新项系数p->elem.coeff

p->elem.e=ha->elem.e; //把指数赋给新项指数p->elem.expn

p->next=hc; //新项p指向hc项，链的建立

hc=p; //指针的赋值，使hc始终指向新创建的项

ha=ha->next; //遍历ha的单次操作

hb=hb->next; //遍历hb的单次操作

ptot++; //完成本次后hc总项数++

}

else ha=ha->next,hb=hb->next; //系数和为0

}

else if(ha->elem.e>hb->elem.e)

{//ha的指数大则先行插入hc

p=(PolyList)malloc(sizeof(struct Polynomial)); //新建项

p->elem=ha->elem; //把ha项元素赋给新项

p->next=hc; //新项p指向hc项，链的建立

hc=p; //指针的赋值，使得hc始终指向新创建的项

ha=ha->next; //遍历ha的单次操作

ptot++; //完成本次后hc总项数++

}

else

{//hb的指数大则先行插入hc

p=(PolyList)malloc(sizeof(struct Polynomial));

p->elem=hb->elem; //把ha项元素赋给新项

p->next=hc; //新项p指向hc项，链的建立

hc=p; //指针的赋值，使得hc始终指向新创建的项

hb=hb->next; //遍历hb的单次操作

ptot++; //完成本次后hc总项数++

}

}

while(ha)

{//ha不为空，则将当前ha剩余项全部插入hc

p=(PolyList)malloc(sizeof(struct Polynomial)); //新建项

p->elem=ha->elem; //把ha项元素赋给新项

p->next=hc; //新项p指向hc项，链的建立

hc=p; //指针的赋值，使得hc始终指向新创建的项

ha=ha->next; //遍历ha的单次操作

ptot++; //完成本次后hc总项数++

}

while(hb)

{//hb不为空，则将当前hb剩余项全部插入hc

p=(PolyList)malloc(sizeof(struct Polynomial)); //新建项

p->elem=hb->elem; //把hb项元素赋给新项

p->next=hc; //新项p指向hc项，链的建立

hc=p; //指针的赋值，使得hc始终指向新创建的项

hb=hb->next; //遍历ha的单次操作

ptot++; //完成本次后hc总项数++

}

SortPolyn(hc,ptot,2);//排序hc

return hc;

}

int main ()

{

PolyList ha,hb,hc,hd; //多项式ha,hb,hc

int m,n; //多项式ha,hb的项数

printf("请输入多项式A和多项式B的项数和对应的系数,指数:\n");

scanf("%d",&m);CreatePolyn(&ha,m); //输入并创建ha,hb

scanf("%d",&n);CreatePolyn(&hb,n);

printf("\n");

printf("原始两个多项式A 和 B\n"); //输出原始多项式

ShowSqList(ha,'A');ShowSqList(hb,'B');

printf("\n");

SortPolyn(ha,m,2);

SortPolyn(hb,n,2); //排序ha,hb

printf("按指数从小到大排序后两个多项式\n");

ShowSqList(ha,'A');

ShowSqList(hb,'B');

printf("\n");

hc=add(ha,hb); //多项式相加

printf("和多项式\n");

ShowSqList(hc,'C');

printf("\n");

printf("两个多项式\n");

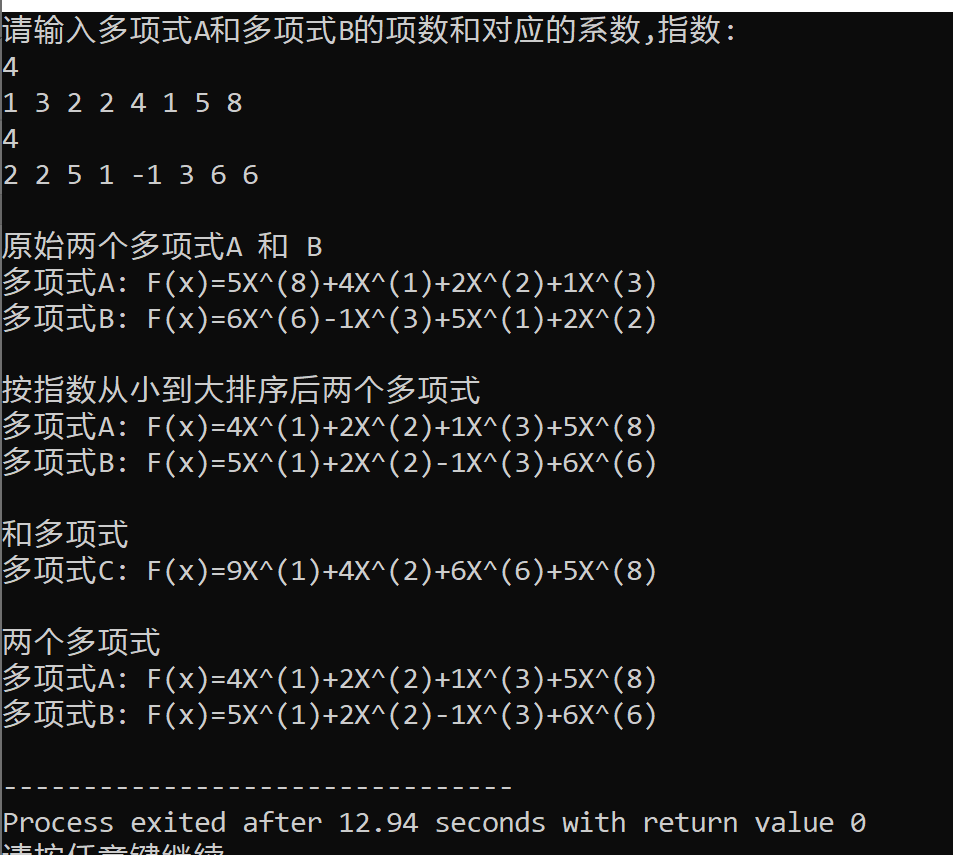
ShowSqList(ha,'A');

ShowSqList(hb,'B'); //输出原多项式(已排序)

return 0;

}

**六、【实验运行结果】**



**七、【实验小结】**

**本实验我复习了并应用了教材上的线性表的链式储存结构和排序算法，链表中的每个结点表示多项式中的一项。先利用冒泡排序对每个链表代表的多项式按照指数从小到大排序，然后根据一元多项式相加的运算法则，将指数相同的项，对应系数相加，如果系数和不为0，则插入到新链表末尾，最终生成的链表记为两个多项式之和。**

**在生成多项式的结果时，我对于多项式输出的格式有了更深的理解，对于系数为0的数要消掉，否则按照系数大小进行输出。**

**通过本次实验我学习到了链表的创建和遍历，对链表的相关的操作有了更进一步的理解，并且学会了通过链表来表示多项式，进行多项式的加法，并且对于指针的运用有了更深刻的认识。**