**中国海洋大学计算机科学与技术系**

**实验报告**

**姓名：徐云哲 学号：15020031088 专业：计算机科学与技**

**实验题目：求两个一元多项式P（x）和Q（x）的乘积： P(x)\*Q(x)**

**一、实验要求：**

1.通过键盘随机输入两个多项式P（x）和Q（x）的内容。

2.输出结果要有P（x）和Q（x）以及它们的乘积P(x)\*Q(x)。

**二、实验程序：**

#include<stdio.h>

#include<stdlib.h>

//多项式中的一项的结构

typedef struct term{

double coef; // 定义系数

int expn; // 定义指数

struct term\* next; // struct关键字可要可不要

}PolyNode ,\*pPolyNode;

//创建一个保存多项式的链表，返回指向头结点的指针。多项式按指数降序排列

pPolyNode CreatePoly()

{

PolyNode \*p,\*q,\*s,\*head=NULL;

double coef;

int expn;

head=(pPolyNode)malloc( sizeof(PolyNode) );

if(head==NULL)

{

printf("内存分配失败!\n");

return NULL;

}

head->coef =0.0;

head->expn =0;

head->next =NULL;

printf("\n请输入多项式的系数和指数（输入0 0结束）:\n");

//scanf("%lf%d",&coef,&expn);

printf("请输入系数:");

scanf("%lf",&coef);

printf("请输入指数:");

scanf("%d",&expn);

while( (long)coef !=0 && expn !=0 )

{

s = (pPolyNode)malloc(sizeof(PolyNode));

s->coef = coef;

s->expn = expn;

q=head->next ;

p=head;

while(q && expn <q->expn )

{

p=q;

q=q->next;

}

if(q== NULL || expn > q->expn )

{

p->next =s;

s->next =q;

}

else

{

q->coef+=coef;

}

printf("请输入系数:");

scanf("%lf",&coef);

printf("请输入指数:");

scanf("%d",&expn);

}

return head;

}

//将多项式逆置,按升幂排列

pPolyNode Reverse(pPolyNode head)

{

PolyNode \*p,\*q,\*t;

p=NULL;

q=head->next;

while( q!=NULL )

{

t=q->next;

q->next =p;

p=q;

q=t;

}

head->next =p;

return head;

}

//两个多项式相乘

pPolyNode multiply(pPolyNode A,pPolyNode B)

{

PolyNode \*pa,\*pb,\*pc,\*u,\*head;

int k ,maxExp;

double coef;

head=(pPolyNode)malloc( sizeof (PolyNode) );

if(head==NULL)

{

printf("内存分配失败!\n");

return NULL;

}

head->coef=0.0;

head->expn =0;

head->next =NULL;

if(A->next !=NULL && B->next != NULL)

{

maxExp=(A->next) ->expn +(B->next )->expn ;

}

else

{

return head;

}

pc=head;

B=Reverse (B);

for(k=maxExp; k>=0; k--)

{

pa = A->next ;

while(pa != NULL && pa->expn >k)

pa=pa->next ;

pb = B->next ;

while( pb != NULL && pa != NULL && (pa->expn + pb->expn) < k )

pb=pb->next;

coef=0.0;

while(pa != NULL && pb != NULL )

{

if( (pa->expn +pb->expn )==k )

{

coef+=pa->coef \* pb->coef;

pa=pa->next;

pb=pb->next;

}

else

{

if(( pa->expn + pb->expn ) > k )

{

pa=pa->next;

}

else

{

pb=pb->next;

}

}

}

if( coef != 0.0 )

{

u=(pPolyNode)malloc(sizeof(PolyNode));

u->coef =coef;

u->expn =k;

u->next =pc->next;

pc->next=u;

pc=u;

}

}

B=Reverse(B);

return head;

}

void Printpoly(pPolyNode head)

{

PolyNode \*p=head->next;

while(p)

{

printf("%1.1f",p->coef);

if(p->expn )

printf("\*x^%d",p->expn );

if(p->next && p->next->coef >0)

printf("+");

p=p->next;

}

}

int main()

{

pPolyNode A,B,C;

A=CreatePoly();

printf("A(x)=");

Printpoly (A);

printf("\n");

B=CreatePoly();

printf("B(x)=");

Printpoly (B);

printf("\n");

C=multiply(A,B);

printf("C(x)=");

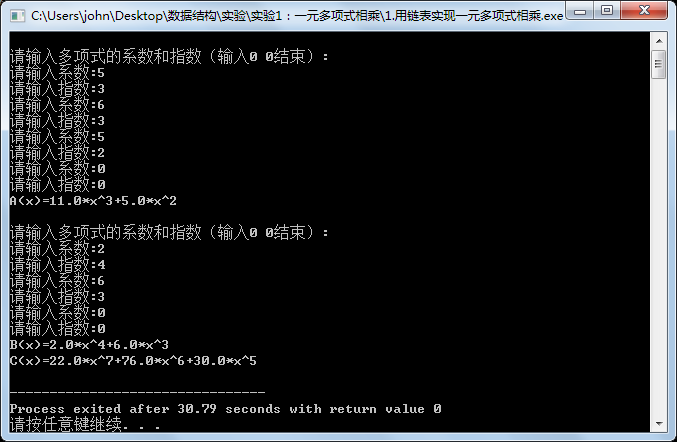
Printpoly (C);

printf("\n");

return 0;

}

1. **实验结果：**

****