实习一 线性表及其应用

题目：一元稀疏多项式的加法运算 实习时间：2016.03.30

一、需求分析

* 1. 输入并建立多项式；
  2. 多项式a与b相加，建立和多项式c；
  3. 输出多项式a,b,c。输出格式：比如多项式a为：A(x)=c1xe1+

c2xe2+…+ cmxem，其中，ci和ei分别为第i项的系数和指数，且各项按

指数的升幂排列，即0≤e1＜e2＜…＜em。多项式b,c类似输出。

4、测试数据：（1）(1+x+x2+x3+x4+x5)+(-x3-x4)=(1+x+x2+x5)

（2）(x+x100)+(x100+x200)=(x+2x100+x200)

（3）(2x+5x8-3x11)+(7-5x8+11x9)=(7+2x+11x9-3x11)

二、设计

1. 设计思想

（1）存储结构：以带头结点的单链表存储多项式。

（2）主要算法基本思想

首先定义单链表每个结点的结构体类型，设计能够生成多项式链表的函数，其功能是可根据给定的多项式来创建存储它的单链表，然后设计两个多项式相加的函数，和打印输出函数来检验多项式的运算结果以及初始的多项式链表是否正确。主函数的设计：实现输入多项式的系数和指数功能，依次调用多项式链表生成的函数、多项式相加的函数，最后输出结果。

1. 设计表示

（1）函数调用关系图

Main ListInsert Add printf

（2）函数接口规格说明

void ListInsert(LinkList \*head,count L) //采用头插法建立多项式的单链表

void Add(LinkList \*a,LinkList \*b,LinkList \*&c) //多项式a b相加等于多项式c

int print(LinkList \*head) //输出多项式单链表

1. 实现注释

void ListInsert(LinkList \*head,count L) //采用头插法建立多项式的单链表

void Add(LinkList \*a,LinkList \*b,LinkList \*&c) //多项式a b相加等于多项式c

int print(LinkList \*head) //输出多项式单链表

4、详细设计

（1）多项式的数据类型定义：

typedef struct Node

{

float coef; //系数

int exp; //指数

struct Node \*next;

}LinkList;

（2）详细设计思路：

1).输入并建立多项式L：输入多项式采用头插法的方式，输入多项式中一个项的系数和指数，就产生一个新的节点，建立起它的右指针，并用头节点指向它。程序如下

void ListInsert(LinkList \*head,count L) //链表插入操作

{

LinkList \*p,\*q;

p=head;

while(p->next!=NULL) //从系数由小到大排序的角度寻找插入结点

{

if(p->next->exp>L.exp) break;

p=p->next;

}

q=(LinkList\*)malloc(sizeof(LinkList));

q->coef=L.coef;

q->exp=L.exp;

q->next=p->next;

p->next=q;

}

2).多项式a,b相加，并付给c：它从两个多项式的头部开始，两个多项式的某一项都不为空时，如果指数相等的话，系数就应该相加；相加的和不为0的话，用头插法建立一个新的节点。p的指数小于q的指数的话，就应该复制q节点到多项式中。p的指数大于q的指数的话，就应该复制p节点到多项式中。当第二个多项式空，第一个多项式不为空时，将第一个多项式用新节点产生。当第一个多项式空，第二个多项式不为空时，将第二多项式用新节点产生，程序如下：

void Add(LinkList \*a,LinkList \*b,LinkList \*c) //多项式a b相加等于多项式c

{

LinkList \*p,\*q,\*s,\*k;

s=c;

p=a->next;

while(p!=NULL) //先将a存入c中

{

q=(LinkList\*)malloc(sizeof(LinkList));

q->coef=p->coef;

q->exp=p->exp;

q->next=NULL;

c->next=q;

c=c->next;

p=p->next;

}

c=s;

p=b->next;

while(p!=NULL) //将b存入c中

{

while((c->next!=NULL)&&(p->exp>(c->next->exp))) //寻找插入结点

{

c=c->next;

}

if(c->next==NULL) //从最后插入

{

q=(LinkList\*)malloc(sizeof(LinkList));

q->coef=p->coef;

q->exp=p->exp;

q->next=c->next;

c->next=q;

}

else if(p->exp==c->next->exp) //指数相等时系数相加

{

s=c;

c=c->next;

c->coef=c->coef+p->coef;

if(c->coef==0)

{

k=c;

c=s;

c->next=c->next->next;

free(k);

}

}

else if(p->exp>c->exp&&c->next!=NULL) //普通插入

{

q=(LinkList\*)malloc(sizeof(LinkList));

q->coef=p->coef;

q->exp=p->exp;

q->next=c->next;

c->next=q;

}

p=p->next;

}

}

3).输出多项式

int Print(LinkList \*head) //输出

{

LinkList \*p;

p=head->next;

if(p==NULL) //多项式为0时直接输出

{

printf("0\n");

exit(0);

}

while(p->coef==1&&p->exp) //指数不为零时系数为1时的输出

{

printf("X^%d",p->exp);

p=p->next;

if(p==NULL) return 0;

else printf("+");

}

while(p->next)

{

printf("%gx^%d",p->coef,p->exp);

p=p->next;

if(p->coef>0) printf("+");

}

printf("%gx^%d",p->coef,p->exp);

return 1;

}

1. 调试分析

1.）调试中遇到的问题及解决方法：

 1.1遇到问题：

 a.在输入多项式时，输入了实型的系数，忽略了程序中指定的类型是整型，从而产生了错误的输出结果。

 b.程序运行的结果出现了形如“0\*3”项，不符合习惯。

 c.输出的表达是不符合数学上的表达方式。

 1.2解决方法：

a.输入整型的系数，也可考虑修改程序中的系数类型，但是因为时间原因，没有修改。

b.在多项式加法函数中，对于两多项式中指数相加的项数系数相加时，要考虑系数和为0的情况，此时应删除结点，不要将其作为和多项式中的一项。

 c.在输出函数的设计中，考虑以下几点： 多项式的第一项，系数为正时不显示“+”号，其他项要显示“+”“-”号。   指数为零时，不能输出指数部分;指数为1时，不输出指数1; 系数为1时，不输出系数.

2.）算法的时间复杂度：

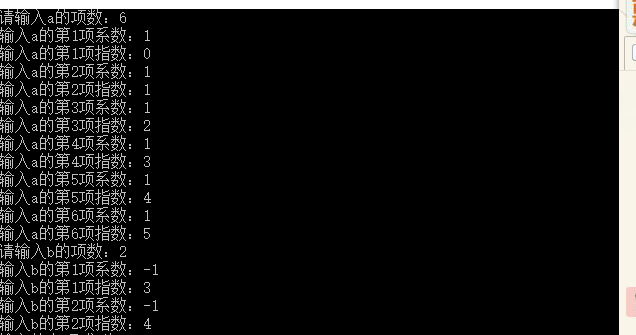
   假设两个多项式分别有m、n项，那么建立俩个多项式的时间复杂度为O（m）和O(n),两个多项式相加的时间复杂度为O(m+n),输出含有n个子项的多项式的时间复杂度为O（n），因而算法的总时间复杂度为O（m+n）.

1. 用户手册

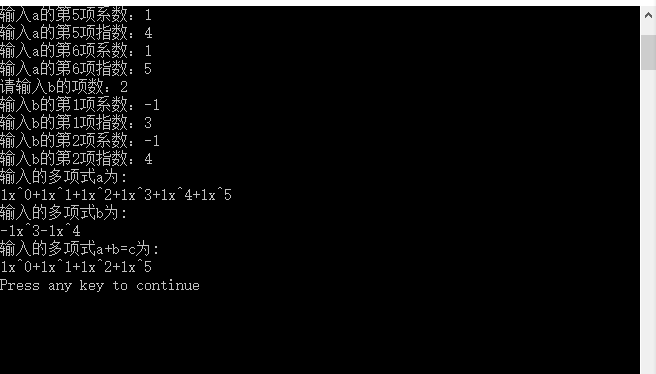
根据屏幕输出的提示首先输入多项式a的项数，然后依次根据屏幕输出的提示输入各项系数和指数，接着输入多项式b，重复a的步骤。输入结束后回车可显示运行结果。

1. 运行结果

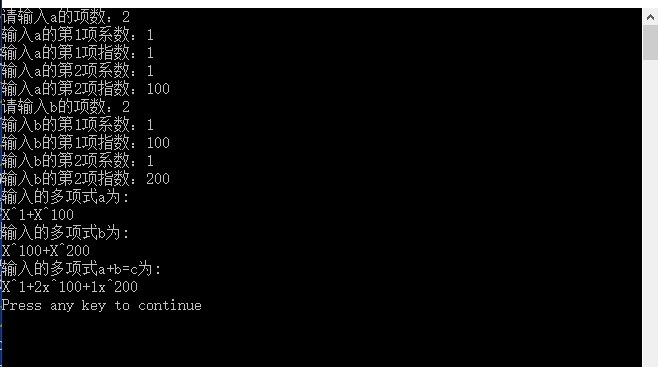
输入数据（1）：



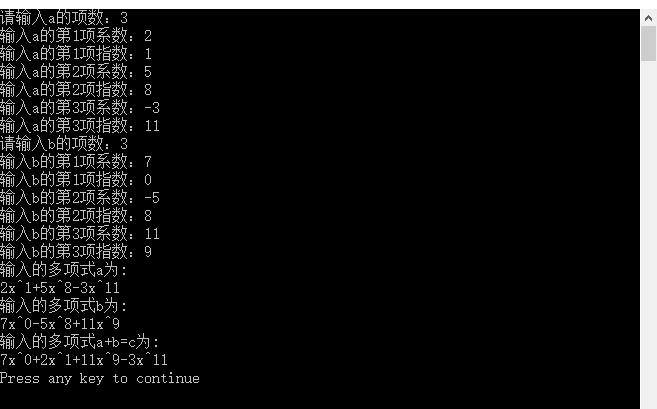
输出结果（1）：



输出结果（2）：



输出结果（3）；



六、源程序清单

#include<stdio.h>

#include<malloc.h>

#include<stdlib.h>

typedef struct

{

float coef;

int exp;

}count;

typedef struct Node

{

float coef; //系数

int exp; //指数

struct Node \*next;

}LinkList;

void ListInitiate(LinkList\*\*head) //链表初始化

{

\*head=(LinkList\*)malloc(sizeof(LinkList));

(\*head)->next=NULL;

}

void ListInsert(LinkList \*head,count L) //链表插入操作

{

LinkList \*p,\*q;

p=head;

while(p->next!=NULL) //从系数由小到大排序的角度寻找插入结点

{

if(p->next->exp>L.exp) break;

p=p->next;

}

q=(LinkList\*)malloc(sizeof(LinkList));

q->coef=L.coef;

q->exp=L.exp;

q->next=p->next;

p->next=q;

}

int Print(LinkList \*head) //输出

{

LinkList \*p;

p=head->next;

if(p==NULL) //多项式为0时直接输出

{

printf("0\n");

exit(0);

}

while(p->coef==1&&p->exp) //指数不为零时系数为1时的输出

{

printf("X^%d",p->exp);

p=p->next;

if(p==NULL) return 0;

else printf("+");

}

while(p->next)

{

printf("%gx^%d",p->coef,p->exp);

p=p->next;

if(p->coef>0) printf("+");

}

printf("%gx^%d",p->coef,p->exp);

return 1;

}

void Add(LinkList \*a,LinkList \*b,LinkList \*c) //多项式a b相加等于多项式c

{

LinkList \*p,\*q,\*s,\*k;

s=c;

p=a->next;

while(p!=NULL) //先将a存入c中

{

q=(LinkList\*)malloc(sizeof(LinkList));

q->coef=p->coef;

q->exp=p->exp;

q->next=NULL;

c->next=q;

c=c->next;

p=p->next;

}

c=s;

p=b->next;

while(p!=NULL) //将b存入c中

{

while((c->next!=NULL)&&(p->exp>(c->next->exp))) //寻找插入结点

{

c=c->next;

}

if(c->next==NULL) //从最后插入

{

q=(LinkList\*)malloc(sizeof(LinkList));

q->coef=p->coef;

q->exp=p->exp;

q->next=c->next;

c->next=q;

}

else if(p->exp==c->next->exp) //指数相等时系数相加

{

s=c;

c=c->next;

c->coef=c->coef+p->coef;

if(c->coef==0)

{

k=c;

c=s;

c->next=c->next->next;

free(k);

}

}

else if(p->exp>c->exp&&c->next!=NULL) //普通插入

{

q=(LinkList\*)malloc(sizeof(LinkList));

q->coef=p->coef;

q->exp=p->exp;

q->next=c->next;

c->next=q;

}

p=p->next;

}

}

void main() //主函数

{

LinkList \*a,\*b,\*c;

count L;

int i,j;

ListInitiate(&a);

ListInitiate(&b);

ListInitiate(&c);

printf("请输入a的项数：");

scanf("%d",&j);

for(i=0;i<j;i++)

{

printf("输入a的第%d项系数：",i+1);

scanf("%g",&L.coef);

printf("输入a的第%d项指数：",i+1);

scanf("%d",&L.exp);

ListInsert(a,L);

}

printf("请输入b的项数：");

scanf("%d",&j);

for(i=0;i<j;i++)

{

printf("输入b的第%d项系数：",i+1);

scanf("%g",&L.coef);

printf("输入b的第%d项指数：",i+1);

scanf("%d",&L.exp);

ListInsert(b,L);

}

Add(a,b,c);

printf("输入的多项式a为:\n");

Print(a);printf("\n");

printf("输入的多项式b为:\n");

Print(b);printf("\n");

printf("输入的多项式a+b=c为:\n");

Print(c);printf("\n");

}