**实验一(必做, 基本实验，4学时)**

**姓名：**岳宇轩 **学号：**19020011038

**专业：**19慧与  **指导老师：**纪筱鹏

**实验题目：**多项式加法乘法问题

**实验目的**：设计一个一元稀疏多项式简单计算器。

**实验内容与要求**

一元稀疏多项式简单计算器的基本功能是：

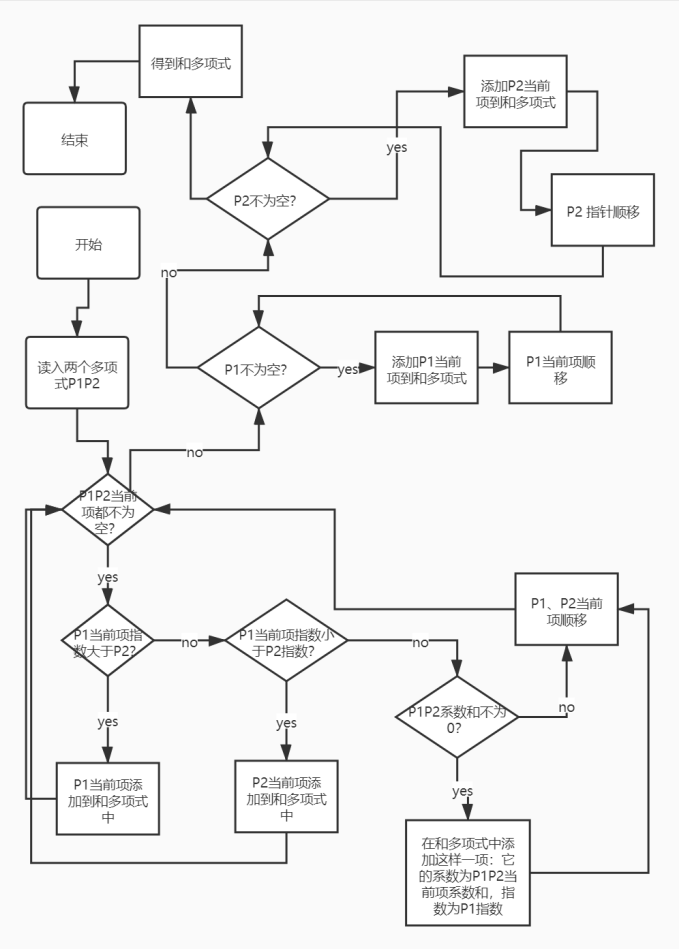
（1）输入并建立多项式；

（2）输出多项式，输出形式为整数序列：n,c1,e1,c2,e2,...,cn,en,其中n是多项式的项数，ci和ei分别是第i项的系数和指数，序列按指数降序排列。

（3）多项式a与多项式b相乘，建立多项式。

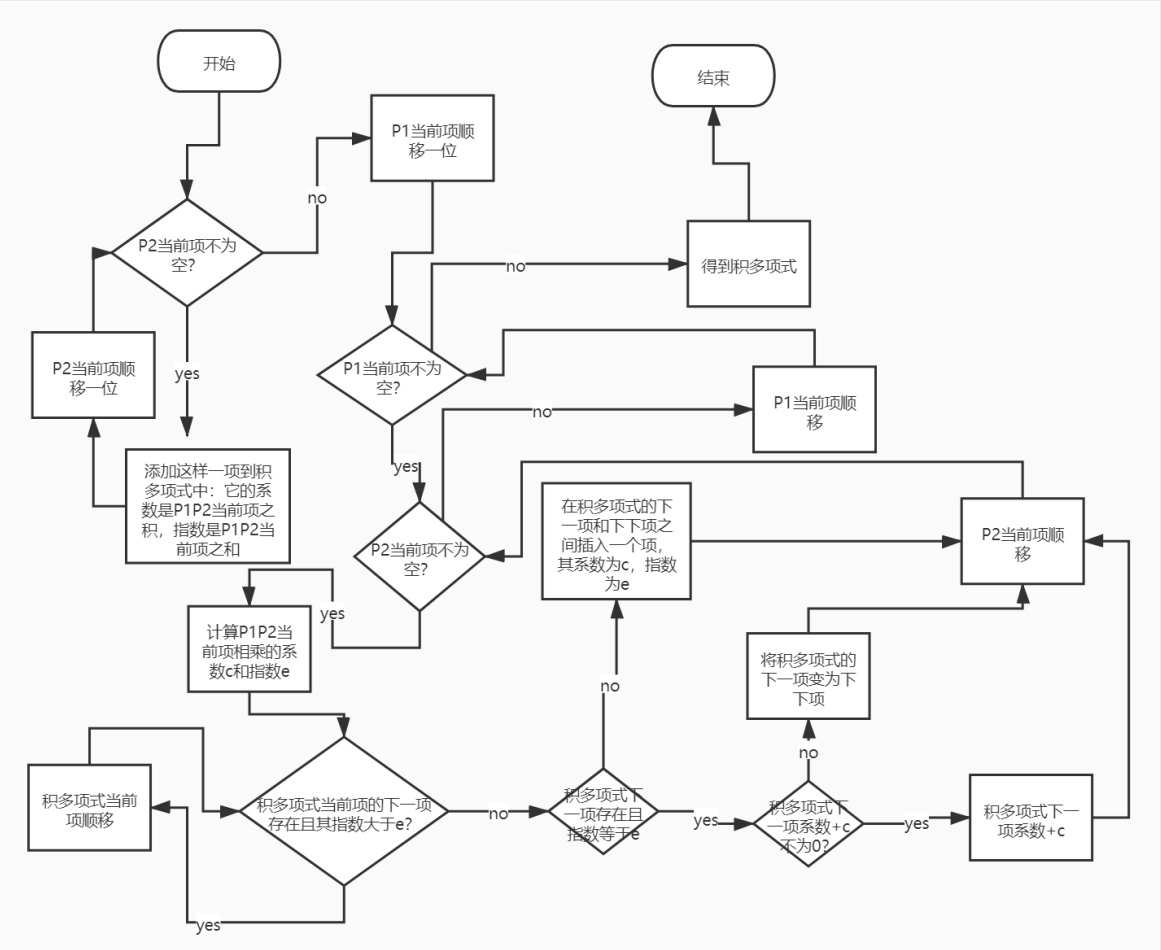
**实验内容和实验步骤：**

**加法算法流程图**



核心思想：根据P1P2指数的大小关系，按照一定的顺序将他们插入到和多项式中

**乘法算法流程图**



核心思想：先用P1第一项乘P2每一项，得到一个暂时的积多项式，然后依次用P1的每一项乘P2，对于得到的结果，在积多项式中找到一个位置，它下一个节点存在且指数≤当前两项相乘后的结果。比较结果项指数和下一项指数，指数相等且系数和不为0，则直接在下一项中加上结果项系数；指数相等且系数和为0，直接连接到下下项；指数不相等，直接在这个位置插入结果项。时间复杂度为O（n2）

**计算乘法还有另外一种方法：(没写在代码里，补充在实验报告里了，编译器测试通过，pta测试通过）**

**先求出P1的第一项与P2相乘后得到的多项式，然后依次用P1各项与P2相乘，将每次得到的结果调用PolyAdd函数相加，最后也可得到正确结果。具体过程如下：**

Polynomial Mult(Polynomial P1, Polynomial P2){  
  
 Polynomial P, Rear, t1, t2, t, head, temp;  
 int c, e;  
 if(!P1 || !P2) //  
 return NULL;  
 t1 = P1;  
 t2 = P2;  
 P = (Polynomial)malloc(sizeof (struct PolyNode));  
 Rear = (Polynomial)malloc(sizeof (struct PolyNode)); // 临时存放相乘结果  
 Rear->link = NULL;  
 head = Rear;  
 P->link = NULL;  
 while(t2){   
 Attach(t1->coef \* t2->coef, t1->expon + t2->expon, &Rear);  
 t2 = t2->link;  
 } // 先求出P1第一项与P2相乘后的结果  
 P = head->link;  
 Rear = head;  
 t1 = t1->link;   
 while(t1){ // 依次求出P1其余各项与P2相乘后的结果  
 t2 = P2;  
 while(t2){ //  
 Attach(t1->coef \* t2->coef, t1->expon + t2->expon, &Rear);  
 t2 = t2->link;  
 }  
 P = PolyAdd(P,head->link); // 每次求出一个结果，便与结果多项式进行加和  
 Rear = head;  
 t1 = t1->link;   
 }  
 return P;  
}

**实验用测试数据和相关结果分析：**

**（1）测试数据**

输入样例：

4 3 4 -5 2 6 1 -2 0

3 5 20 -7 4 3 1

输出样例：

15 24 -25 22 30 21 -10 20 -21 8 35 6 -33 5 14 4 -15 3 18 2 -6 1

5 20 -4 4 -5 2 9 1 -2 0

1. **结果分析**

上述样例中计算的是：

3x4-5x2+6x-2

5x20-7x4+3x

的乘法和加法。

按照多项式乘法和加法的计算规则，最后的结果应该为：

积：

15x24-25x22+30x21-10x20-21x8+35x6-33x5+14x4-15x3+18x2-6x

和：

5x20-4x4-5x2+9x-2

通过对比发现，

实验输出结果

15 24 -25 22 30 21 -10 20 -21 8 35 6 -33 5 14 4 -15 3 18 2 -6 1

5 20 -4 4 -5 2 9 1 -2 0

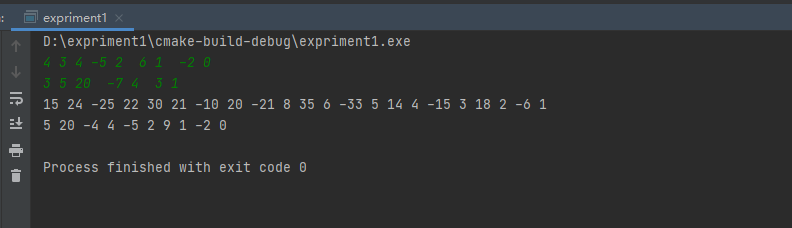
与计算结果相同

**实验总结：**

pta平台运行截图：



编译器：



CODE:

#include <stdio.h>  
#include <stdlib.h>  
  
struct PolyNode{  
 int coef; // 系数  
 int expon; // 指数  
 struct PolyNode \*link; // 指针指向下一个结点  
};  
  
  
typedef struct PolyNode \*Polynomial;  
  
  
void Attach(int c, int e, Polynomial \*pRear){  
 /\*获得参数为一个系数和一个指数，并生成一个项，把它接在多项式后\*/  
 Polynomial P;  
 P = (Polynomial)malloc(sizeof (struct PolyNode));  
 P->coef = c;  
 P->expon = e;  
 P->link = NULL; // 生成一个新节点  
 (\*pRear)->link = P; // 连接  
 \*pRear = P; // 指针下移  
}  
  
  
Polynomial PolyAdd(Polynomial P1, Polynomial P2){  
  
 /\*多项式加法的计算函数，参数为两个Polynomial类型的多项式，返回结果为Polynomial类型\*/  
  
  
 int sum;  
 Polynomial rear;  
 rear = (Polynomial)malloc(sizeof (struct PolyNode)); // 产生一个临时空节点作为多项式链表头  
 Polynomial front;  
 front = rear;  
 while (P1 && P2) // 当两个多项式都还没完全参与计算时  
 if(P1->expon > P2->expon){  
 // 如果多项式1当前项的指数大于多项式2当前项的指数，则把多项式1的当前项接入生成多项式  
 Attach(P1->coef, P1->expon, &rear);  
 P1 = P1->link;  
 } else if(P1->expon < P2->expon){  
 // 如果多项式1当前项的指数小于多项式2当前项的指数，则把多项式2的当前项接入生成多项式  
 Attach(P2->coef, P2->expon, &rear);  
 P2 = P2->link;  
 } else{  
 // 如果多项式1和2的指数相同  
 sum = P1->coef + P2->coef;  
 if(sum) // 系数相加后不为0，则生成新节点；否则，系数和为0，则只需将两指针顺移即可  
 Attach(sum, P1->expon, &rear);  
 P1 = P1->link;  
 P2 = P2->link; // 指针顺移  
 }  
 /\*将未处理完的另一个多项式的所有节点复制到生成多项式中\*/  
 for(; P1; P1 = P1->link)  
 Attach(P1->coef, P1->expon, &rear);  
 for(; P2; P2 = P2->link)  
 Attach(P2->coef, P2->expon, &rear);  
 rear->link = NULL; // 设置生成多项式最后一项的下一项指针为空  
 Polynomial temp;  
 temp = front;  
 front = front->link; // front指向第一个非空节点  
 free(temp); // 释放空节点  
 return front; // 返回生成多项式  
}  
  
Polynomial Mult(Polynomial P1, Polynomial P2){  
 /\*计算两个多项式的乘法\*/  
 Polynomial P, Rear, t1, t2, t;  
 int c, e;  
 if(!P1 || !P2) // 如果两个多项式中任意一个为空，则返回NULL  
 return NULL;  
 t1 = P1;  
 t2 = P2;  
 P = (Polynomial)malloc(sizeof (struct PolyNode));  
 P->link = NULL;  
 Rear = P;  
 while(t2){ // 用P1的第一项与P2的每一项做乘法，即系数相乘、指数相加，得到P  
 Attach(t1->coef \* t2->coef, t1->expon + t2->expon, &Rear);  
 t2 = t2->link;  
 }  
 t1 = t1->link; // t1指向P1的第二项  
 while(t1){ // 循环计算P1的每一项与P2相乘的结果  
 t2 = P2;  
 Rear = P;  
 while(t2){ // 循环计算P1在当前循环中的项与P2在当前循环中的项相乘的结果  
 e = t1->expon + t2->expon;  
 c = t1->coef \* t2->coef;  
 while(Rear->link && Rear->link->expon > e) // 找到这样一个节点，它下一个节点存在且指数≤当前两项相乘后的结果  
 Rear = Rear->link;  
 if(Rear->link && Rear->link->expon == e){ // 指数相等的情况  
 if(Rear->link->coef + c) // 系数相加不为0，则直接在原系数上做加法  
 Rear->link->coef += c;  
 else{ // 系数相加为0  
 t = Rear->link;  
 Rear->link = t->link; // 下个节点相加后系数为0，连接至下下个节点  
 free(t); // 释放相加后系数为0的节点  
 }  
 } else{ // Rear指针下一项的指数小于当前两项相乘后结果的指数  
 t = (Polynomial)malloc(sizeof (struct PolyNode));  
 t->coef = c;  
 t->expon = e; // 生成一个新节点  
 t->link = Rear->link;  
 Rear->link = t; // 在Rear指针的当前项和下一项之间插入新节点  
 Rear = Rear->link; // 指针顺移  
 }  
 t2 = t2->link;  
 }  
 t1 = t1->link;  
 }  
 t2 = P;  
 P = P->link;  
 free(t2); // 释放空表头  
 return P;  
}  
  
Polynomial ReadPoly(){  
 /\*从键盘读入一个多项式\*/  
 Polynomial P, Rear, t;  
 int c, e, N;  
 scanf("%d", &N); // 输入多项式项的个数  
 P = (Polynomial)malloc(sizeof (struct PolyNode));  
 P->link = NULL;  
 Rear = P;  
 while(N--){ // 循环输入每项的系数和指数，添加新项  
 scanf("%d %d", &c, &e);  
 Attach(c, e, &Rear);  
 }  
 t = P;  
 P = P->link;  
 free(t); // 释放空表头  
 return P;  
}  
  
  
void PrintPoly(Polynomial P){  
 /\*输出一个多项式\*/  
 int flag = 0; // 作为控制输出格式的标志，每输出一项就输出一个空格，但在输出最后一项后不输出空格  
 if(!P){ // 0 多项式输出0 0  
 printf("0 0\n");  
 return;  
 }  
 while(P){  
 if(!flag)  
 flag = 1;  
 else  
 printf(" ");  
 printf("%d %d", P->coef, P->expon);  
 P = P->link;  
 }  
 printf("\n");  
}  
int main() {  
 Polynomial P1,P2,PP,PS;  
 P1 = ReadPoly();  
 P2 = ReadPoly();  
 PP = Mult(P1, P2);  
 PrintPoly(PP);  
 PS = PolyAdd(P1, P2);  
 PrintPoly(PS);  
 return 0;  
}