多项式的加法运算实现

主要思路：相同指数的项系数相加，其余部分进行拷贝。

采用不带头结点的**单向链表**，按照**指数递减**的顺序排列各项。

算法思路：两个指针P1，P2分别指向这两个多项式的第一个结点，不断循环：

- P1->expon==P2->expon：系数相加，若结果不为0，则作为结果多项式对应项的系数。同时P1，P2都分别指向下一项；
- P1->expon>P2->expon：将P1的当前项存入结果多项式，并使P1指向下一项；
- P1->expon<P2->expon：将P2的当前项存入结果多项式，并使P2指向下一项。

当某一多项式处理完时，将另一多项式的所有结点依次复制到结果多项式中。

多项式加法运算

```c
struct PolyNode
{
    int coef; //系数
    int expon; //指数
    struct PolyNode *link; //指向下一个结点的指针
};

typedef struct PolyNode *Polynomial;
Polynomial P1,P2;

Polynomial PolyAdd(Polynomial P1,Polynomial P2)
{
    Polynomial front,rear,temp;
    int sum;
    rear=(Polynomial)malloc(sizeof(struct PolyNode));
    front=rear; //由front记录结果多项式链表头结点
    while(P1&&P2) //当两个多项式都有非零项待处理时
    {
        switch (Compare(P1->expon,P2->expon))
        {
        case 1:
            Attach(P1->coef,P1->expon,&rear);
            P1=P1->link;
            break;
        case -1:
            Attach(P2->coef,P2->expon,&rear)
            P2=P2->link;
            break;
        case 0:
            sum=P1->coef+P2->coef;
            if(sum)
            {
                Attach(sum,P1->expon,&rear);
            }
            P1=P1->link;
            P2=P2->link;
            break;
        default:
            break;
        }
    }
    //将未处理完的多项式复制到结果多项式里面去
    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;
    temp=front;
    front=front->link; //令front指向结果多项式第一个非零项
    free(temp); //释放临时空表头结点，free掉temp指针所指向的这个空间
    return front;
}

void Attach(int c,int e,Polynomial *pRear)
{
    Polynomial P;
    P=(Polynomial)malloc(sizeof(struct PolyNode));
    P->coef=c; //对新结点幅值
    P->expon=e;
    P->link=NULL;
    (*pRear)->link=P;
    *pRear=P; //修改pRear值
}
```