应用示例

学习目标和要求

1.能够根据问题，设计链表结构，并编程实现。

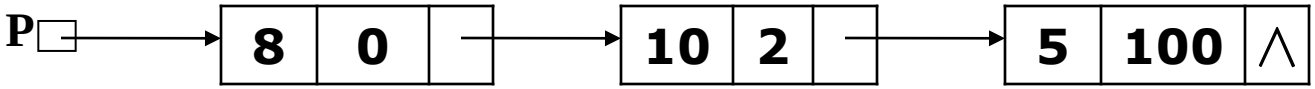
问题：稀疏多项式（在原表上）求和

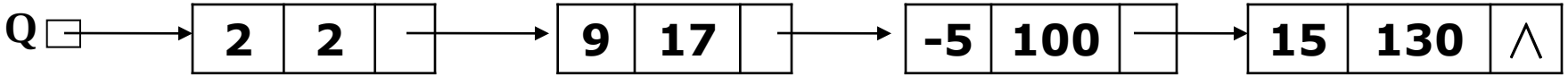
- $P=8+10x^2+5x^{100}$
- $Q=2x^2+9x^{17}-5x^{100}+15x^{130}$
- 求P+Q

- 如何存储多项式？链表。
- 如何求和？有序合并。
- 如何高效？

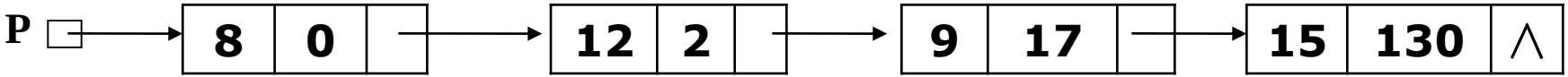
问题：稀疏多项式（在原表上）求和

- $P=8+10x^2+5x^{100}$
- $Q=2x^2+9x^{17}-5x^{100}+15x^{130}$
- 求P+Q



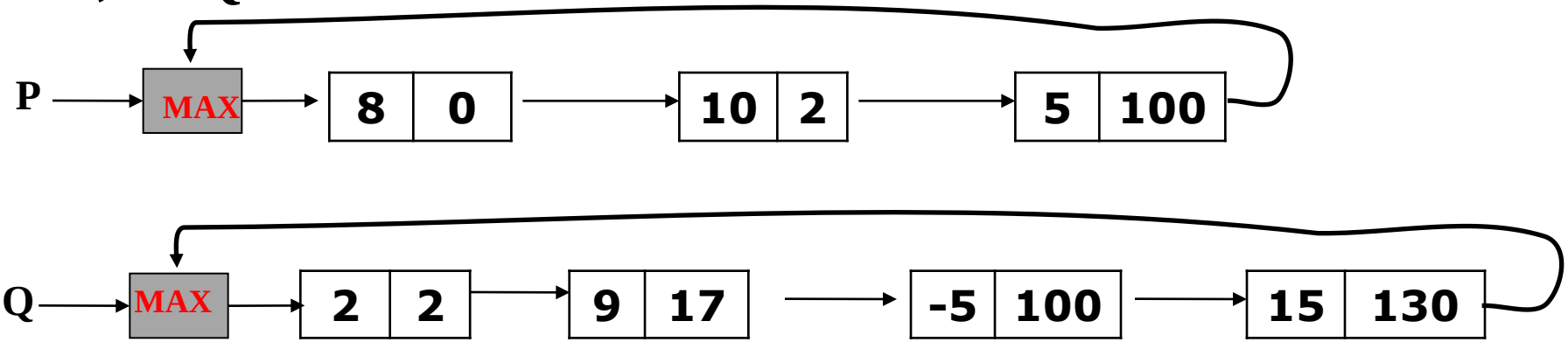


求和结果（放在P链表中）：



问题：稀疏多项式（在原表上）求和

- $P=8+10x^2+5x^{100}$
- $Q=2x^2+9x^{17}-5x^{100}+15x^{130}$
- 求P+Q



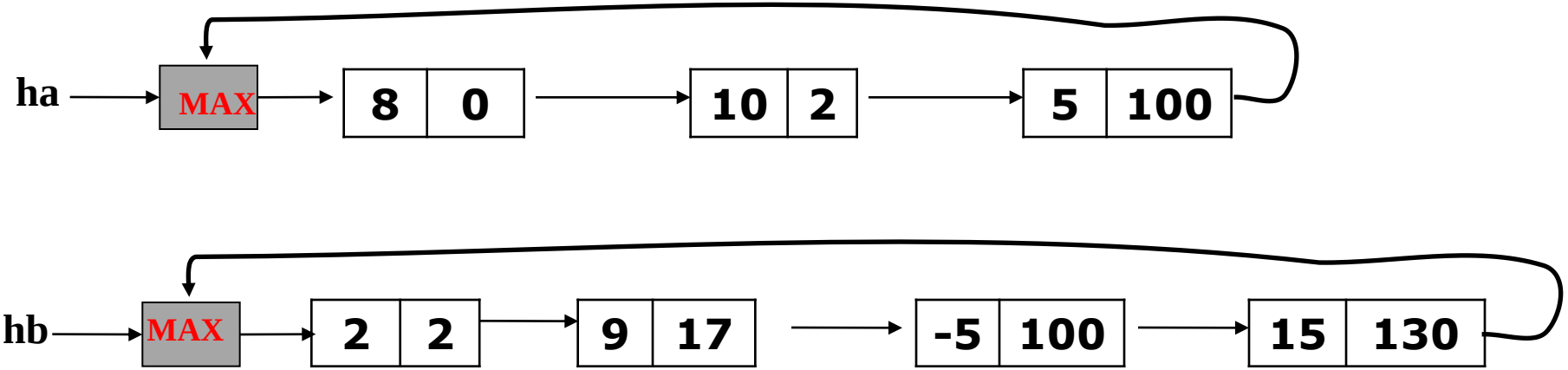
结点结构和主调程序

```
typedef struct node
{
    int  ceof;   //系数域
    int  exp;    //指数域
    struct node *next;
}snode,*ptr;
```

```
void main()
{  ……
   ha=creatlinkedBC(); //创建多项式
   hb=creatlinkedBC(); //创建多项式
   ha=add_n(ha,hb); //多项式求和
   ……
}
```

稀疏多项式结构图

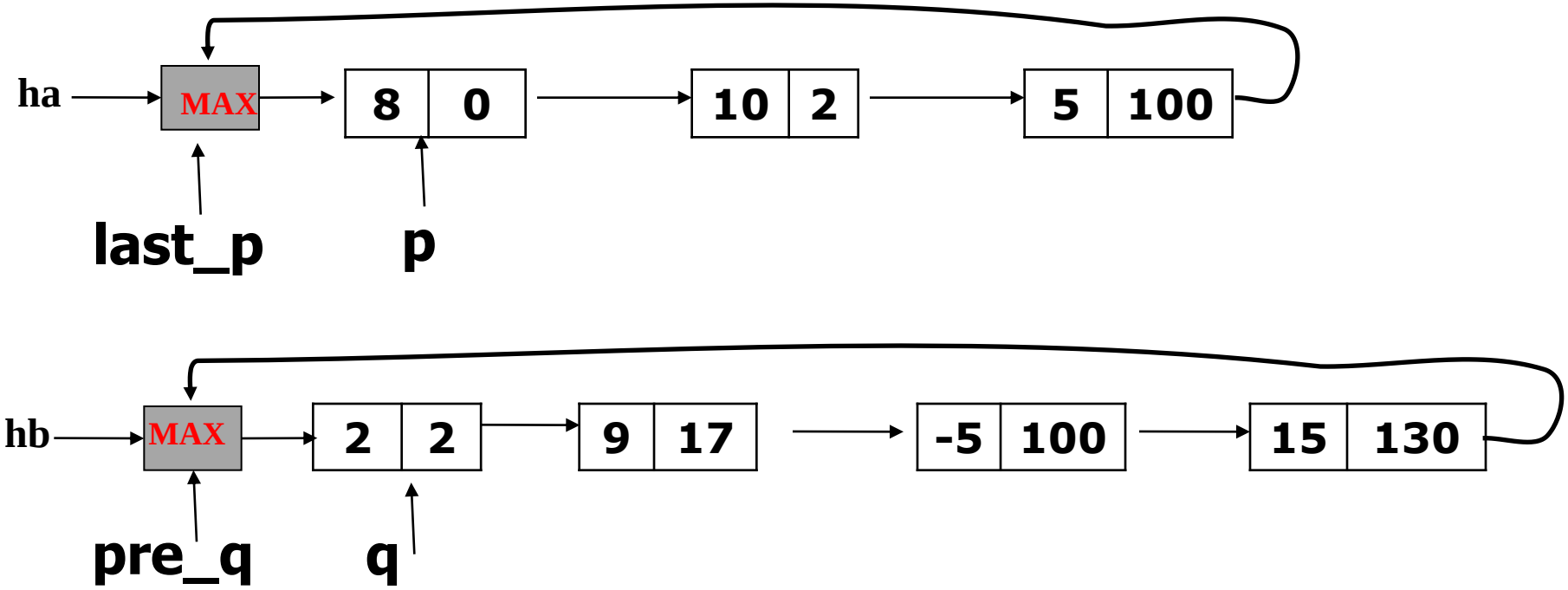
```
ha=creatlinkedBC(); //创建多项式
hb=creatlinkedBC(); //创建多项式
```



构造多项式算法（有序插入）

```
ptr  creatlinkedBC( )
{ ptr  head, f,s,p;   int  coef,exp ;
  head=new snode;  head->exp=MAX; head->next=head;
  scanf("%d %d", &coef,&exp); //输入系数和指数
  while (exp!=-1)
  {  p=new snode;   p->ceof=coef;   p->exp=exp;//建新结点
    f=head, s=f->next;                  //置搜索指针初值
    while(s->exp<exp)f=s,s=s->next;   //有序搜索
    f->next=p, p->next=s;                    //有序插入
    scanf("%d %d", &coef,&exp);        //读入下一个元素
  }
  return(head);
}
```

加头有序循环链表合并（hb并入ha）



加头有序循环链表合并算法

```
ptr add_n(ptr ha,ptr hb)
{ptr p,q,last_p,pre_q;  int val;
  p=ha->next;   q=hb->next;
  last_p=ha; pre_q=hb;
  while(p!=ha && q!=hb)
  { if(p->exp>q->exp)   //第一种情况
    {  pre_q->next=q->next;  // 从hb链表删除q结点
       last_p->next=q;   //插入ha链表
       q->next=p;
       last_p=last_p->next;  //调整前驱指针位置
       q=pre_q->next;   //在hb中取一个新结点
     }
     else
      if(p->exp<q->exp)  //第二种情况
```

加头有序循环链表合并算法

```
if(p->exp<q->exp) //第二种情况
 {last_p=p;  p=p->next;  }
 else   //第三种，指数相同的合并
  {  val=p->ceof+q->ceof;
    if(val!=0)//插入一个，删除一个
    {   p->ceof=val;
       pre_q->next=q->next;
       free(q);
       last_p=p;
       p=p->next;
       q=pre_q->next;
      }
    else//删除系数相加后为0的结点
     {  last_p->next=p->next;
        pre_q->next=q->next;
                free(p);
                free(q);
                p=last_p->next;
                q=pre_q->next;
          }
        }
  }
  if(q!=hb)   //插入hb中剩余的链表
  { last_p->next=q;
    while (q->next!=hb) q=q->next;
    q->next=p;
   }
   free(hb);//释放hb表头监督元
   return ha;
}
```