2.4 一元多项式的表示和相加

一元多项式：

$P_n(x)=p_0+p_1x+p_2x^2+\ldots+p_nx^n$

在计算机中，可以用一个线性表来表示：

$P=(p_0, p_1, \ldots, p_n)$

但是对于形如 $S(x)=1+3x^{10000}-2x^{20000}$ 的多项式，上述表示方法是否合适？

一般情况下的**一元稀疏多项式**可写成

$$P_n(x) = p_1x^{e1} + p_2x^{e2} + \cdots + p_mx^{em}$$

其中：p_i 是指数为e_i 的项的非零系数，

$$0 \le e_1 < e_2 < \cdots < e_m = n$$

可以下列线性表表示：

$$((p_1, e_1), (p_2, e_2), \ldots, (p_m, e_m))$$

例如: 多项式 $P_{999}(x) = 7x^3 - 2x^{12} - 8x^{999}$

可用线性表 （（7, 3），（-2, 12），（-8, 999））表示

抽象数据类型一元多项式的定义如下：

ADT Polynomial {

数据对象：

D = { a_i | a_i ∈TermSet, i=1,2,...,m, m≥0

TermSet 中的每个元素包含一个

表示系数的实数和表示指数的整数 }

数据关系：

R1 = { <a_{i-1} ,a_i >|a_{i-1} ，a_i∈D， i=2,...,n

且a_{i-1}中的指数值 < a_i中的指数值 }

基本操作：多项式的有关操作

} ADT Polynomial

一元多项式的实现：

```
typedef  OrderedLinkList   polynomial ;
   // 用带表头结点的有序链表表示多项式
```

结点的数据元素类型定义为：

```
typedef struct {      // 项的表示
    float  coef ;          // 系数
    int    expn ;          // 指数
} term, ElemType ;
```

```
void CreatPolyn (polynomail &p, int m);  {
    //输入m项的系数和指数，建立表示一元多项式的有序链表P
    InitList (P)；   h = GetHead (P)；
    e.coef = 0.0；   e.expn = -1；  SetCurElem (h, e)；
                                          //设置头结点的数据元素
    for ( i=1; i<=m; ++i) {             //依次输入m个非零项
        scanf (e.coef, e.expn)；
        if (!LocateElem (P, e, q,(* cmp)())) {
                                        //当前链表中不存在该指数项
            if (MakeNode (s, e))   InsFirst (q, s)；
                                        //生成结点并插入链表
        }
     }
  }//CreatPolyn
```

算法：2.22

```
void AddPolyn (polynomial &Pa, polynomial &Pb) {
   // 多项式加法：Pc = Pa + Pb，利用两个多项式的结
//点构成“和多项式”
  ha = GetHead (La)；  hb = GetHead (Lb)；
                    //ha和hb分别指向La和Lb的头结点
  pa = NextPos ( La, ha)；   pb = NextPos ( Lb, hb)；
                   //pa和pb分别指向La和Lb中当前结点
  while ( pa && pb ){                  // La和Lb均非空
    a = GetCurElem ( pa )；  b = GetCurElem ( pb )；
                          //a和b为两表中当前比较元素
```

```
switch (*cmp(e1, e2)) {
    case -1： {       // 多项式PA中当前结点的指数值小
ha=qa； qa = NextPos ( Pa, qa)； break ；
   case 0： {             // 两者的指数值相等
      sum= a.coef + b.coef ;
      if ( sum != 0.0 ) {
                    //修改多项式PA中当前结点的系数值
         SetCurElem (qa, sum)； ha=qa ；}
      else {                //删除多项式PA中当前结点
         DelFirst (ha, qa)；     FreeNode (qa)；}
      DelFirst(hb, qb)； FreeNode(qb)；qb=NextPos
      ( Pb, qb)； qa = NextPos ( Pa, qa)； break ；
```

```
        case 1: {     //多项式PB中当前结点的指数值小
          DelFirst (hb, qb)；  InsFirst (ha, qb)；
                  qb =NextPos ( Pb, qb)；
                  ha = NextPos ( Pa, qa)；    break；
         }//switch
      }//while
      if (!ListEmpty (Pb))  Append (Pa, qb)；
                                  //链接Pb中剩余结点
       FreeNode (hb)；              // 释放Pb的头结点
} // AddPolyn
```

算法：2.23

本章小结

1、了解线性表的逻辑结构特性，以及线性表的顺序存储结构和链式存储结构。熟悉这两种结构的特点及适用场合。

2、熟练掌握这两类存储结构的描述方法，以及线性表的各种基本操作的实现。

3、能够从时间和空间复杂度的角度综合比较线性表两种存储结构的不同特点及其适用场合。