第2章 线性表

2.1 线性表的基本概念

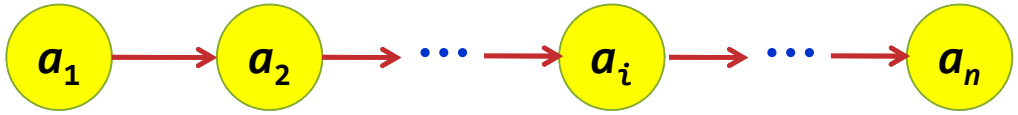
2.1.1 线性表的定义

- 线性表是由n（$n \geq 0$）个相同类型的数据元素组成的有限序列。标记为：

  $L = (a_1, a_2, \ldots, a_i, \ldots, a_n)$

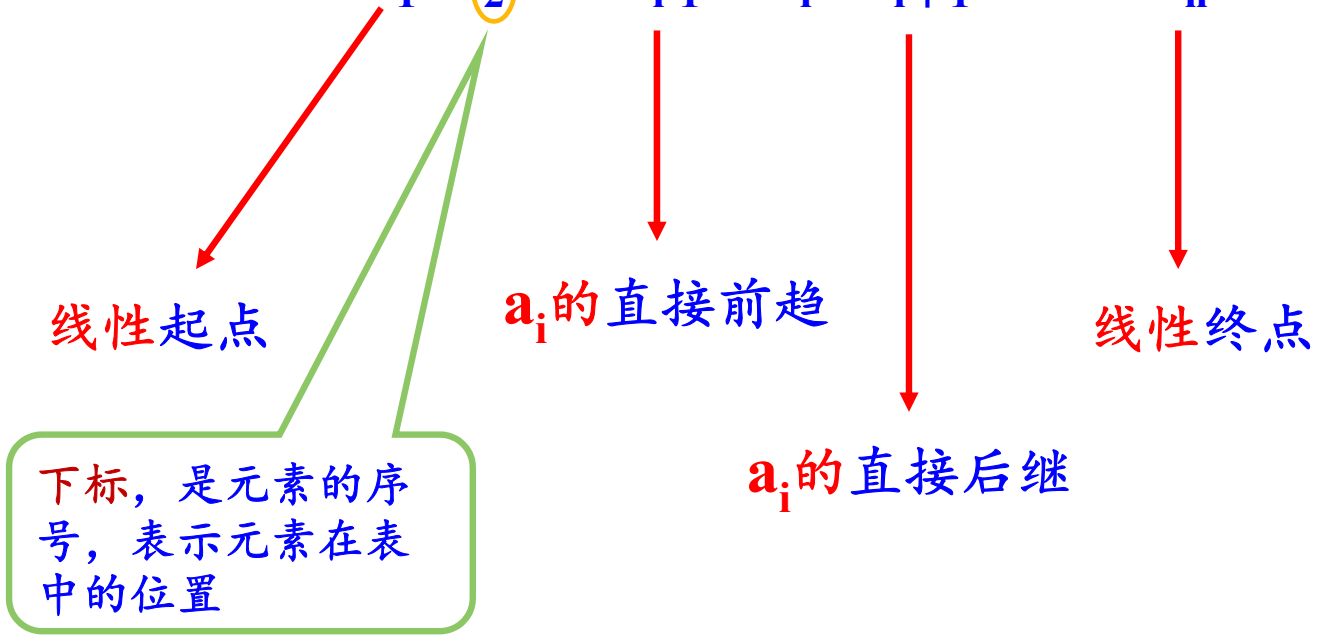
- 线性表中元素的个数n定义为线性表的长度，当$n=0$时为空表。

- 当$n>0$时，线性表的逻辑结构图如下所示。



线性表的几个概念

$L = (a_1, a_2, \cdots a_{i-1}, a_i, a_{i+1}, \cdots, a_n)$



*i*为数据元素a_i在线性表中的位序。

逻辑特征：

- 若至少含有一个元素，则只有唯一的起始元素；
- 若至少含有一个元素，则只有唯一的尾元素；
- 除了起始元素外，其他元素有且仅有一个前驱元素；
- 除了尾结点外，其他元素有且仅有一个后继元素。

- 线性表中的每个元素有唯一的序号（逻辑序号），同一个线性表中可以存在值相同的多个元素，但它们的序号是不同的。

2.1.2 线性表的基本运算

线性表L的基本运算如下：

- 初始化**InitList(L)**。其作用是建立一个空表**L**（即建立线性表的构架，但不含任何数据元素）。
- 销毁线性表**DestroyList(L)**。其作用是释放线性表**L**的内存空间。
- 求线性表的长度**ListLength(L)**。其作用是返回线性表**L**的长度。
- 求线性表中第***i***个元素**GetElem(L,*i*,*e*)**。其作用是返回线性表**L**的第***i***个数据元素。

- 按值查找**LocateElem(L,*x*)**。若**L**中存在一个或多个值与***x***相等的元素，则其作用是返回第一个值为***x***的元素的逻辑序号。
- 插入元素**ListInsert(L,*i*,*x*)**。其作用是在线性表**L**的第***i***个位置上增加一个以***x***为值的新元素
- 删除元素**ListDelete(L,*i*)**。其作用是删除线性表**L**的第***i***个元素a_i。
- 输出元素值**DispList(L)**。其作用是按前后次序输出线性表**L**的所有元素值。

线性表抽象数据类型List：

ADT LIST {

线性表中元素的逻辑结构;

基本运算定义;

}

- 对上述定义的抽象数据类型线性表，还可进行一些更复杂的操作。例如，将两个或两个以上的线性表合并成一个线性表;把一个线性表拆开成两个或两个以上的线性表;重新复制一个线性表等。

【示例2-1】利用两个线性表LA和LB分别表示两个集合A和B，现要求一个新的集合A=A∪B。

代码实现：

```
void Union(SqList &La, SqList Lb)
{
    ElemType e;
    int La_len, Lb_len;
    int i;
    La_len = ListLength(La);        // 求线性表的长度
    Lb_len = ListLength(Lb);
    for(i = 1; i <= Lb_len; i++) {
        GetElem(Lb, i, e);          // 取Lb中第i个数据元素赋给e
        // La中不存在和e相同的元素，则插入之
        if(!LocateElem(La, e, equal))
            ListInsert(La, ++La_len, e);
    }
}
```

【示例2-2】线性表LA和LB，其元素均按非递减有序排列，编写一个算法将它们合并成一个线性表LC，且LC的元素也是按非递减有序排列。

算法思路：依次扫描LA和LB的元素，比较当前的元素的值，将较小值的元素赋给LC，如此直到一个线性表扫描完毕，然后将未完的那个顺序表中余下部分赋给LC即可。LC的容量要能够容纳LA、LB两个线性表相加的长度。

代码实现

```
void MergeList(SqList La, SqList Lb, SqList &Lc)
{
    int i = 1, j = 1, k = 0;
    int La_len, Lb_len;
    ElemType ai, bj;
    InitList(Lc);                       // 创建空表Lc
    La_len = ListLength(La);
    Lb_len = ListLength(Lb);
    while(i <= La_len && j <= Lb_len) {  // 表La和表Lb均非空
        GetElem(La, i, ai);
        GetElem(Lb, j, bj);
        if(ai <= bj) {
            ListInsert(Lc, ++k, ai);
            ++i;
        }
```

```
        else {
            ListInsert(Lc, ++k, bj);
            ++j;
        }
    }       // 以下两个while循环只会有一个被执行
    while(i <= La_len) {                // 表La非空且表Lb空
        GetElem(La, i++, ai);
        ListInsert(Lc, ++k, ai);
    }
    while(j <= Lb_len) {                // 表Lb非空且表La空
        GetElem(Lb, j++, bj);
        ListInsert(Lc, ++k, bj);
    }
}
```

END