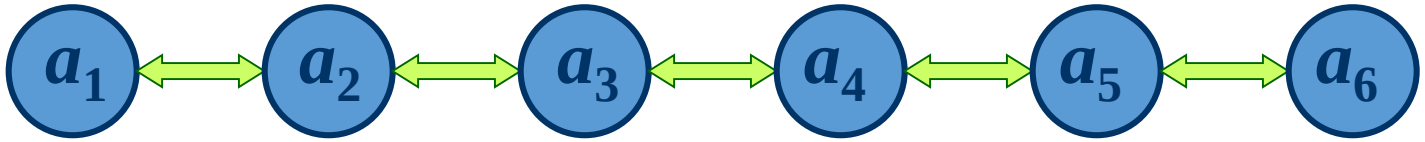
线性表 (Linear List)

- 线性表的定义和特点
  - 定义 n（$\geqslant 0$）个数据元素的有限序列，记作

    $$(a_1, a_2, \cdots, a_n)$$

    a_i是表中数据元素，n 是表长度。
  - 特点 线性排列
    - ✓ 除第一个元素外，其他每一个元素有一个且仅有一个直接前趋。
    - ✓ 除最后一个元素外，其他每一个元素有一个且仅有一个直接后继。



- 理解线性表的要点是
  a) 表中元素具有逻辑上的顺序性，在序列中各元素排列有其先后次序，有唯一的首元素和尾元素。
  b) 表中元素个数有限。
  c) 表中元素都是数据元素。即每一表元素都是原子数据，不允许“表中套表”。
  d) 表中元素的数据类型都相同。这意味着每一表元素占有相同数量的存储空间。

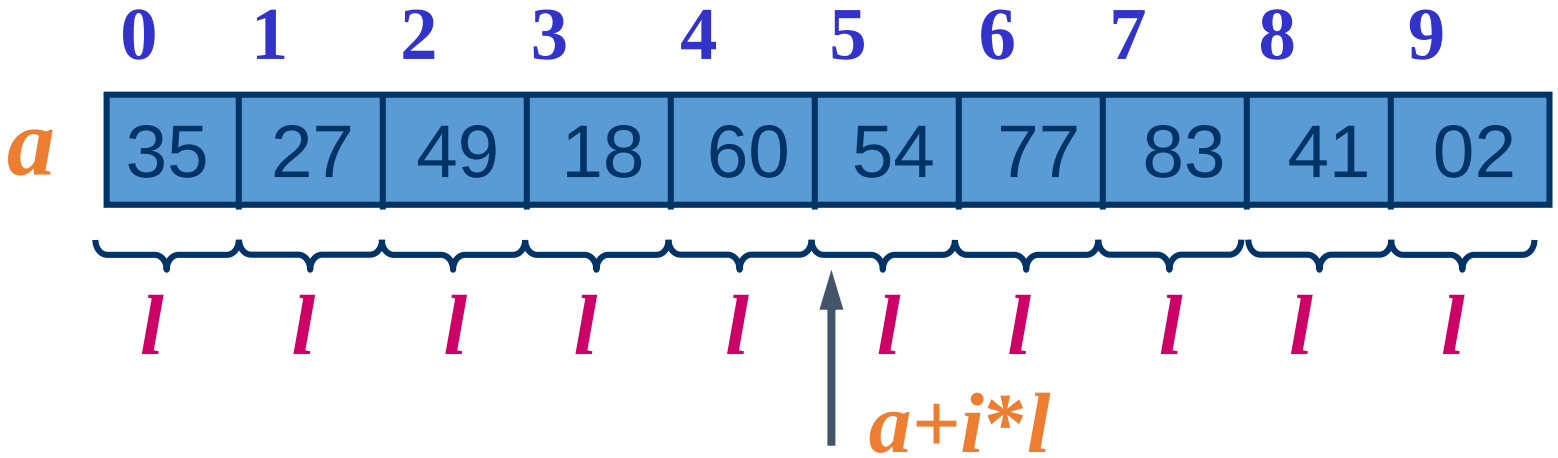
顺序表 (Sequential List)

- 顺序表的定义和特点
  - 定义　将线性表中的元素相继存放在一个连续的存储空间中，即构成顺序表。
  - 存储　它是线性表的顺序存储表示，可利用一维数组描述存储结构。
  - 特点　元素的逻辑顺序与物理顺序一致。
  - 访问方式　可顺序存取，可按下标直接存取。

顺序表的连续存储方式

$$LOC(i) = LOC(i-1) + l = a + i * l,$$

LOC 是元素存储位置，*l* 是元素大小



$$LOC(i) = \begin{cases} a, & i = 0 \\ LOC(i-1)+l = a + i*l, & i > 0 \end{cases}$$

顺序表的静态结构定义

```
#define maxSize 100          //最大允许长度
typedef int DataType;        //元素的数据类型

typedef struct {
    dataType data[maxSize];          //存储数组
    int n;                           //当前表元素个数
} SeqList;
```

- 顺序表静态定义，假定 L 是一个类型 SeqList 的顺序表，一般用 L.data[i] 来访问它。
- 表一旦装满，不能扩充。

顺序表的动态结构定义

```
#define initSize 100          //最大允许长度
typedef int DataType;         //元素的数据类型

typedef struct {
      DataType *data;         //存储数组
     int n;                   //当前表元素个数
      int maxSize;            //表的最大长度
} SeqList;
```

- 顺序表动态定义，它可以扩充，新的大小计入数据成员maxSize中。

顺序表基本运算的实现

- 构造一个空的顺序表

```
void InitList (SeqList& L) {
                L.data = ( DataType* ) malloc
( initSize*sizeof                    ( DataType ));
                if ( L.data == NULL)
          { printf ("存储分配失败!\n");   exit (1); }
       L.n = 0;  L.maxSize = initSize;
  }
```

- 引用型参数 & 的使用
  - 例如， void InitList (SeqList& L)
  - 引用型参数“&”是把形参 L 看作是实际变量（一个表）的别名，在函数体内对 L 的操作将直接对实际变量的操作。
  - 好处之一是可在函数体内像普通变量那样对 L 操作，使得操作简单。
  - 好处之二是可直接从实际变量得到操作结果。
  - 好处之三是不必创建实际变量的副本空间。

- 按值查找：在顺序表中从头查找结点值等于给定值 x 的结点

```
int Find ( SeqList& L, DataType x ) {
    for ( i = 0; i < L.n; i++ )
        if ( L.data[i] == x ) return i;     //查找成功
    return −1;                               //查找失败
}
```

- 注意，如果表中元素序号从1开始，则第 *i* 个元素存储于第 *i*−1 个数组元素位置，函数返回位置比元素序号小1。

查找算法性能分析

- 查找成功的平均比较次数

$$ACN = \sum_{i=0}^{n-1} p_i \times c_i$$

- 若查找概率相等，则

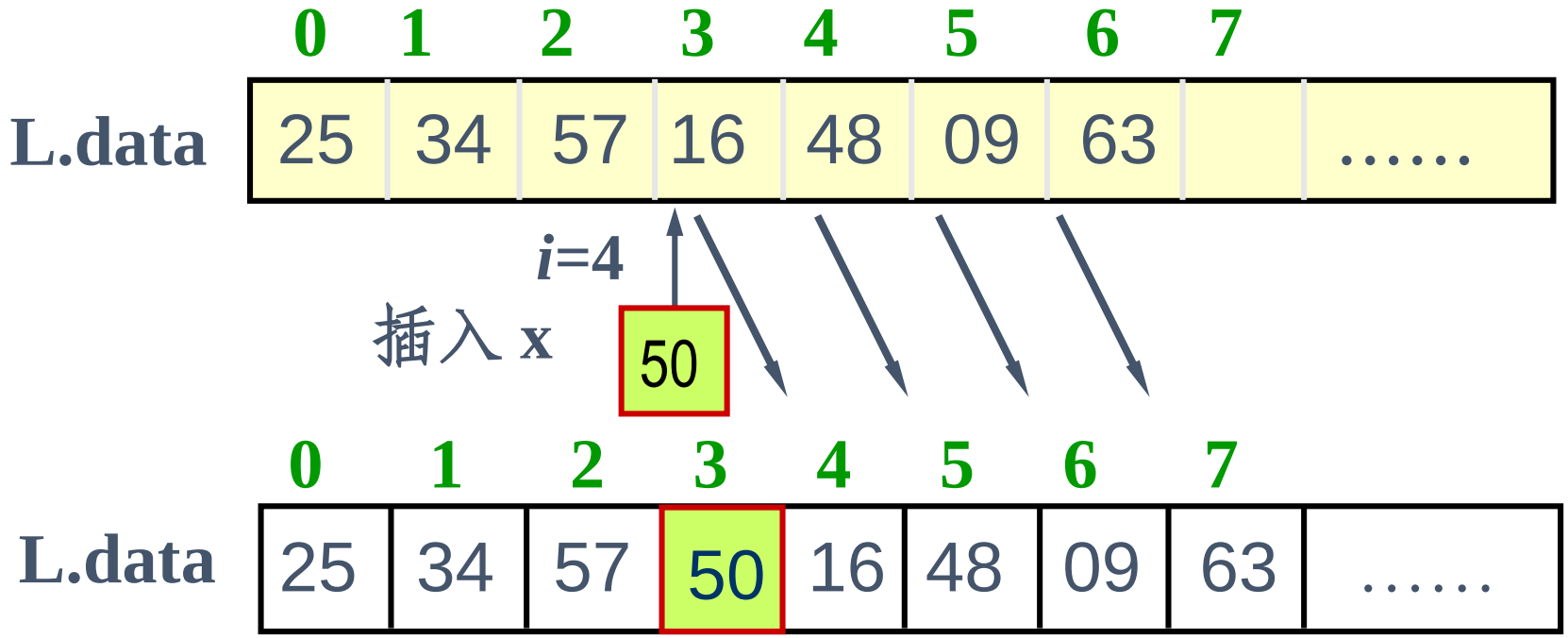
$$ACN = \frac{1}{n}\sum_{i=0}^{n-1}(i+1) = \frac{1}{n}(1+2+\cdots+n) =$$

$$= \frac{1}{n} * \frac{(1+n)*n}{2} = \frac{1+n}{2}$$

- 查找不成功　数据比较 n 次。

插入新元素

```
bool Insert ( SeqList& L, DataType x, int i ) {
    //在表中第 i (1≤i≤n+1) 个位置插入新元素 x
     if ( L.n == L.maxSize ) return false;
    if ( i < 1 || i > L.n+1 ) return false;
     for ( int j = L.n−1; j >= i−1; j−−)
           L.data[j+1] = L.data[j];
     L.data[i−1] = x;          //实际插在第i−1个位置
     L.n++;  return true;     //插入成功
}
```



- 插入时平均移动元素个数AMN

$$AMN = \frac{1}{n+1}\sum_{i=1}^{n+1}(n-i+1) = \frac{1}{n+1}\frac{n(n+1)}{2} = \frac{n}{2}$$

删除表元素

```
bool Remove ( SeqList& L, int i, DataType& x ) {
//在表中删除第 i 个元素，通过 x 返回其值
    if ( L.n > 0 && i > 0 && i <= L.n ) {
        x = L.data[i−1];
         for (int j = i; j < L.n; j++)
              L.data[j−1] = L.data[j];
                   L.n−−;  return true;        //删除成功
    }
    else return false;                  //删除失败
}
```

表项的删除

- 删除时平均移动元素个数AMN

$$AMN = \frac{1}{n}\sum_{i=1}^{n}(n-i) = \frac{1}{n}\frac{(n-1)n}{2} = \frac{n-1}{2}$$