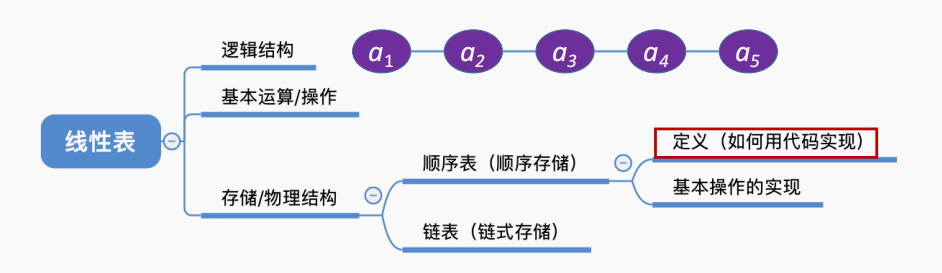
顺序表定义

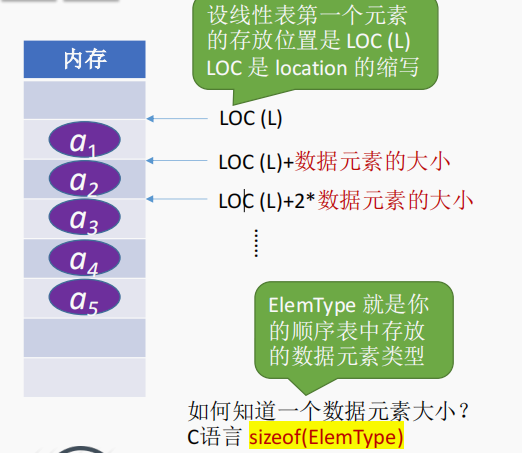


线性表是具有**相同**数据类型的n（n≥0）个**数据元素**的**有限序列**

**顺序表**--用**顺序存储**的方式实现线性表

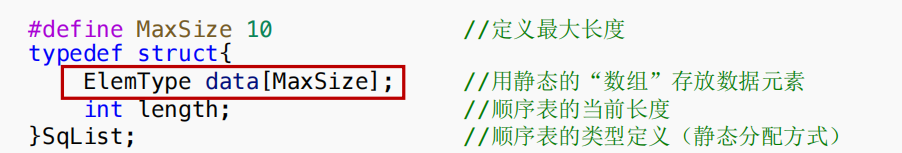
**顺序存储**：把**逻辑上相邻**的元素存储在**物理位置上也相邻**的存储单元中，元素之间的关

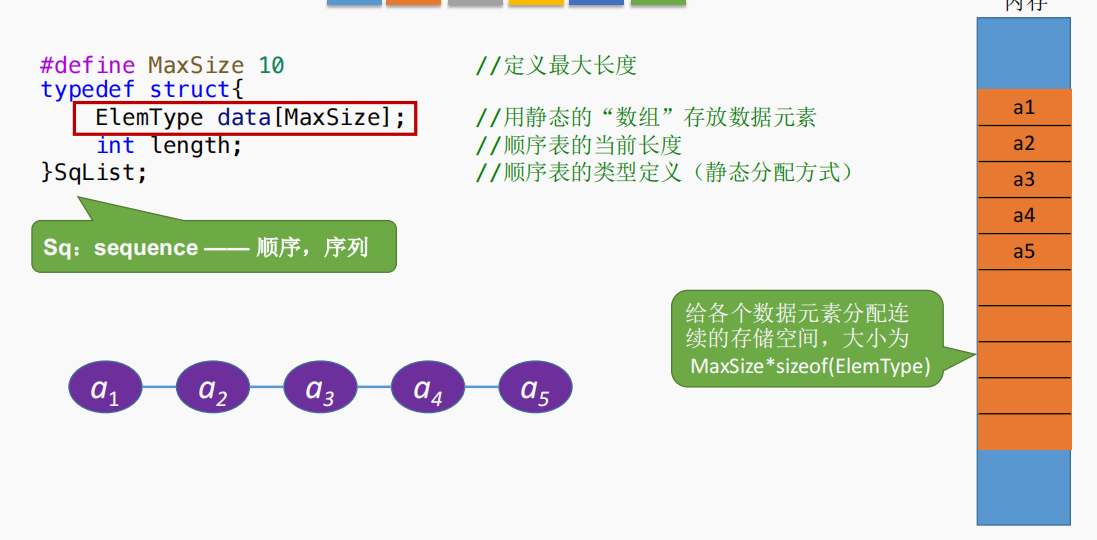
系由存储单元的邻接关系来体现。



设线性表第**1**个元素的存放位置是**loc(i)**，则第**i**个元素的存放位置为**loc(L)+(i-1)\*sizeof(Elemtype)**

顺序表的实现——**静态分配**：

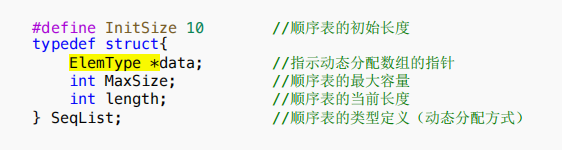




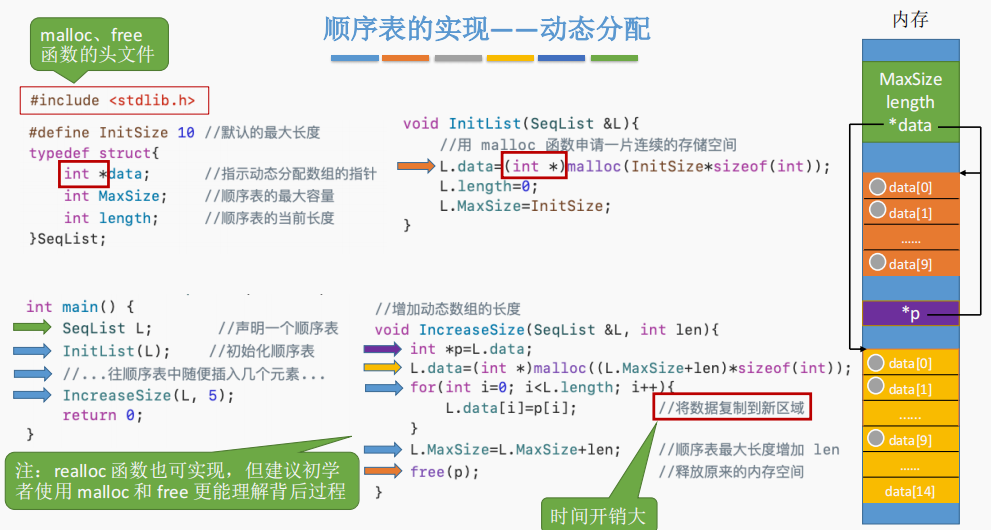
初始化：



顺序表的实现——**动态分配**：







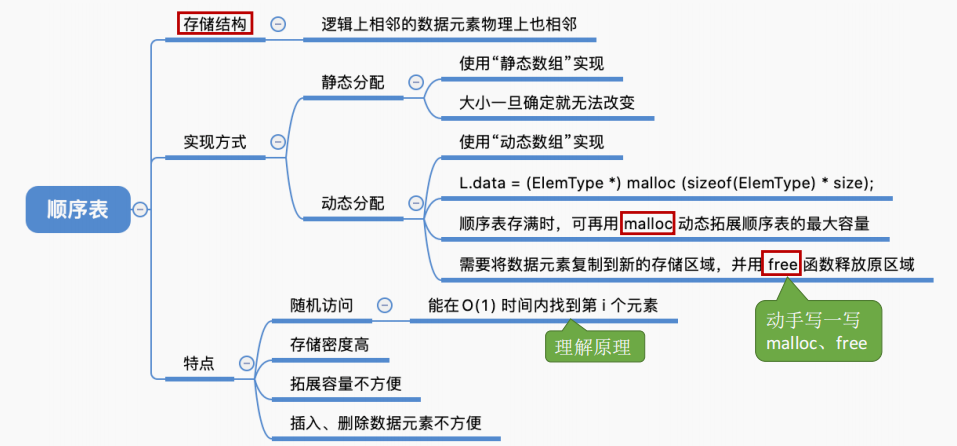
顺序表的特点：

①**随机访问**，即可以在 **O(1) 时间**内找到**第 i 个元素**。

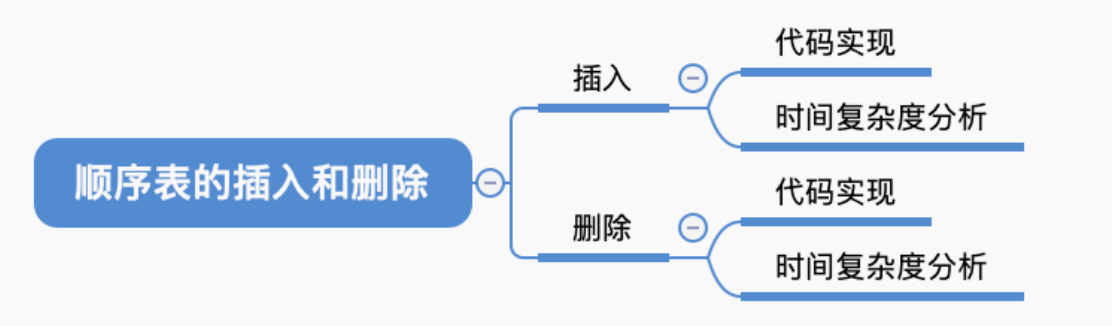
②**存储密度高**，每个节点只存储数据元素

③**拓展容量不方便**（即便采用**动态分配**的方式实现，拓展长度的**时间复杂度也比较高**）

④**插入、删除**操作**不方便**，需要**移动大量元素**



顺序表插入和删除

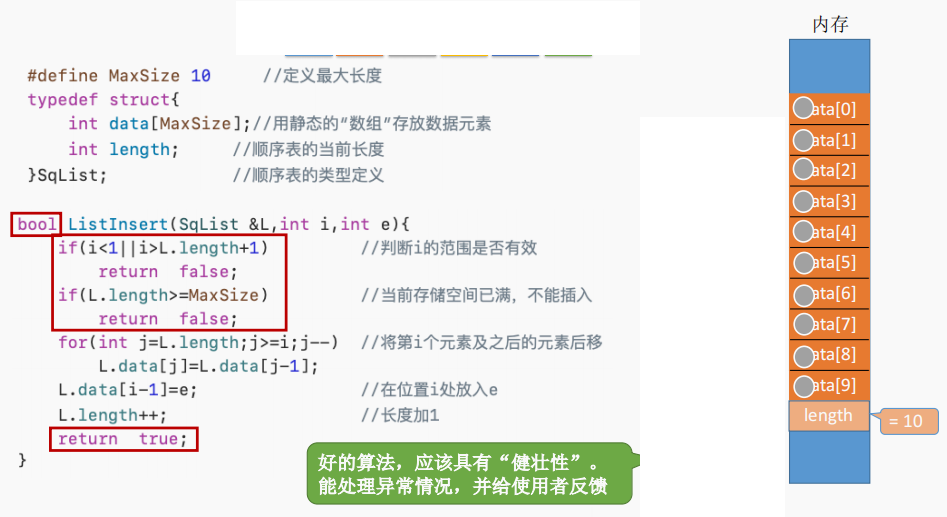


顺序表的基本操作——插入

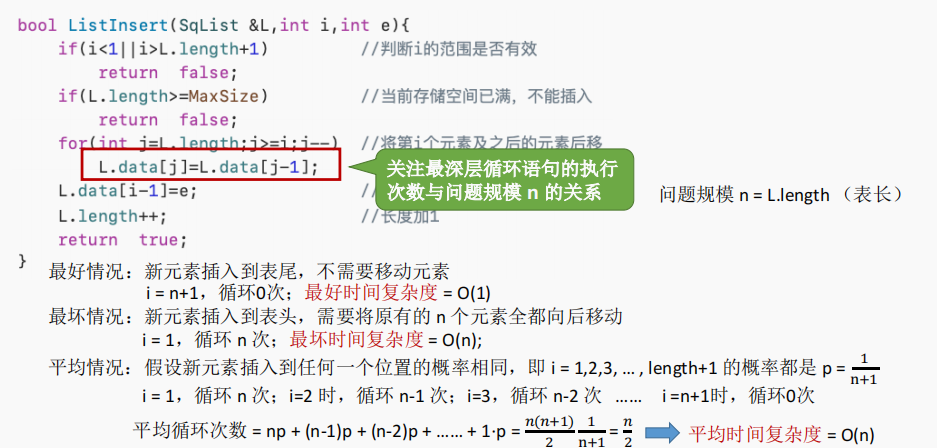
用**存储位置的相邻**来体现**数据元素之间的逻辑关系**

ListInsert(&L,i,e)：**插入**操作。在**表L中**的**第i个位置**上**插入指定元素e**。



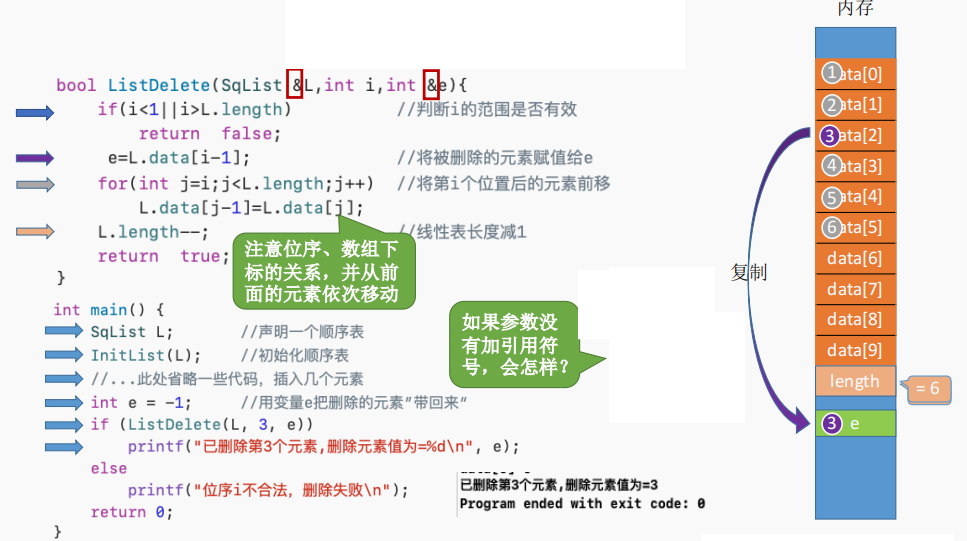


插入操作的时间复杂度--**平均时间复杂度=O(n)**

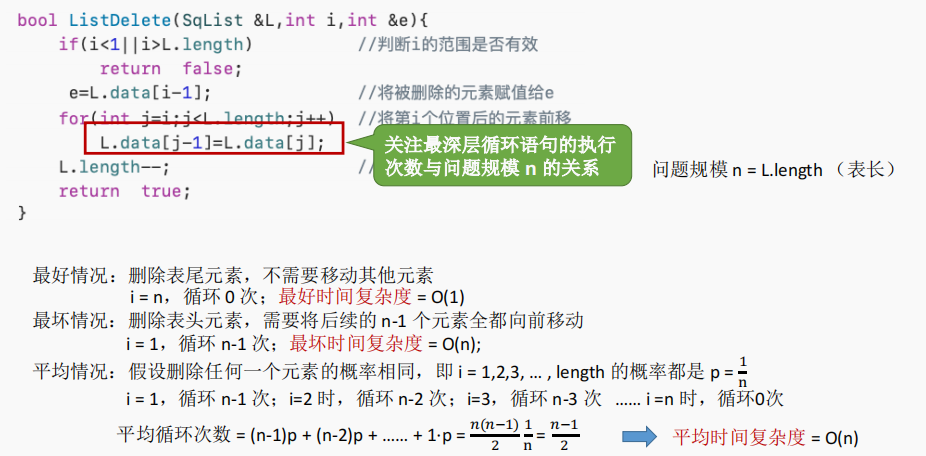


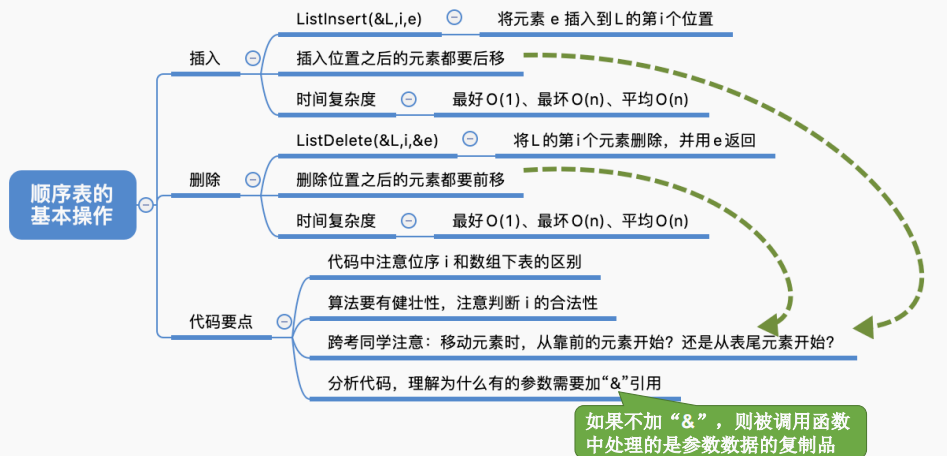
顺序表的基本操作——删除

ListDelete(&L,i,&e)：删除操作。**删除表L**中**第i个位置**的元素，并用**e返回删除元素的值**。

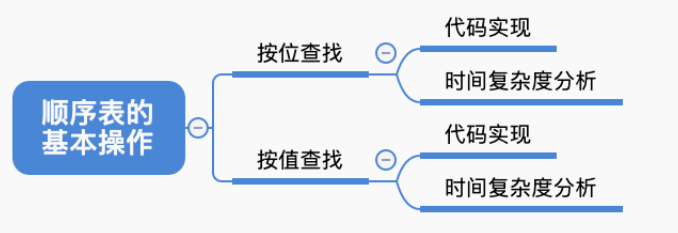


删除操作的时间复杂度--**平均时间复杂度=O(n)**





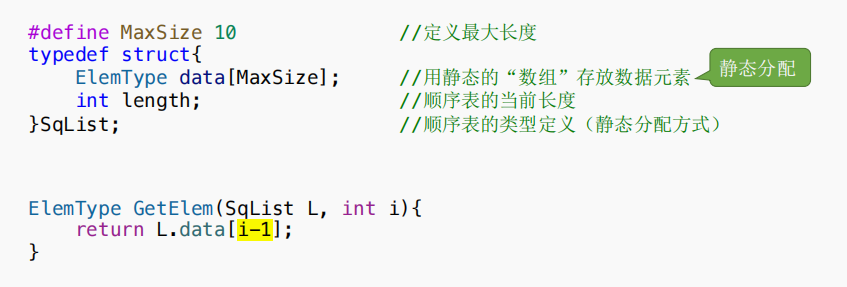
顺序表查找



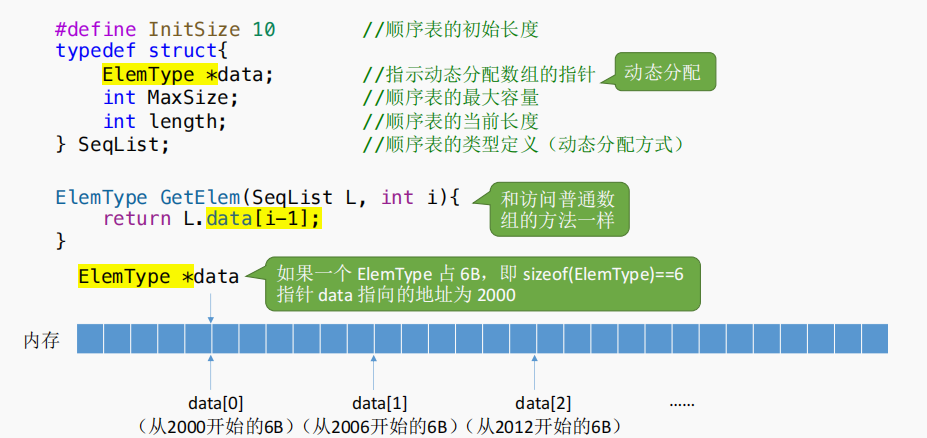
顺序表的按位查找：

**GetElem(L,i)**：**按位查找**操作。获取表L中第i个位置的元素的值。

顺序表实现:



链表实现：

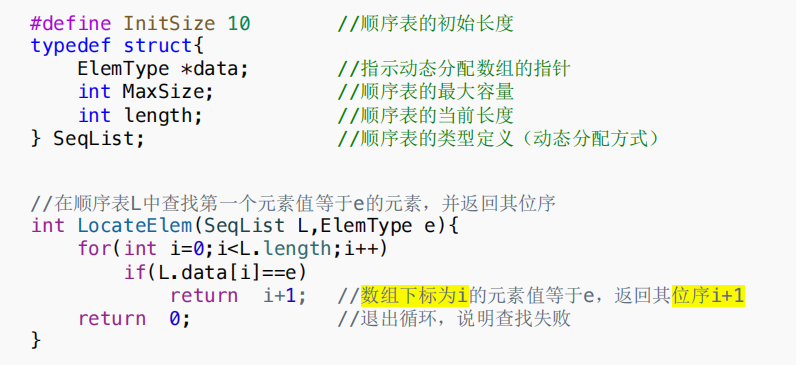


时间复杂度：O(1)

由于顺序表的各个数据元素在内存中**连续存放**，因此可以根据**起始地址**和**数据元素大小**立即找到**第i个元素**——“**随机存取**”特性

顺序表的按值查找：

**LocateElem(L,e)**：**按值查找**操作。在表L中查找具有给定关键字值的元素



1. **基本数据类型**：**int、char、double、float** 等可以直接用运算符“**==**”比较
2. C语言中，**结构体**的比较**不能**直接用 “**==** ”

按值查找的**时间复杂度**--关注最深层循环语句的执行次数与问题规模n的关系

问题规模 n = L.length （表长）

