一. 基本顺序表和元素外置顺序表

基本顺序表：

Loc(ei) = loc(e0) + c\*i。数据元素本身连续存储，每个元素所占的存储单元大小固定相同，元素的下标是其逻辑地址，而元素存储的物理地址(实际内存地址) 可以通过存储区的初始地址loc(e0) 加上逻辑地址(第i个元素) 与存储单元大小(c) 的乘积计算而获得。故访问指定元素时无需从头遍历，通过计算便可获得对应地址。其时间复杂度为O(1)。

元素外置：

元素大小不统一，将实际数据元素另行存储，而顺序表中各单元位置保存对应元素的地址信息，指针。

二. 顺序表的一体式结构与分离式结构

一体式：表头和数据连续存放

分离式：表头加上第三项数据区链接地址（其他两项为容量和现有元素个数）指向数据区，分离开来

元素数据区替换：超出原设定的容量，需要重新申请新的空间。

对于一体式：li地址改变，表头和数据区全部用新的连续空间，地址全改变。

对于分离式：li地址不变，表头部分只改第三项指向地址链接，指向新的地址。

三. 顺序表数据区替换与扩充

1. 每次固定数目，操作次数多

2. 每次加倍扩充，以空间换时间

四. 顺序表添加删除元素的实现

添加：

1. 尾端加入： O(1)

2. 保存：插入后，后面元素顺序后移O(n)

3.非保存：插入替换位置元素，并把原元素放到尾部 O(1)

删除：

1. 尾端删除： O(1)

2. 保存： O(n)

3.非保存： O(1)

Python中的list分离式元素外置，初始8个元素存储区，区满扩充四倍大，阈值50000之后采用一倍扩充。

五. 访问越界问题

Int arr[3] = {0};

For(; i<=3; i++){}

i=3时a[3]访问越界，c语言中，只要不是访问受限的内存，所有的内存空间都可以自由访问。a[3]也会被定位到某块不属于数组的内存地址上，而这个地址正好时存储变量i的内存地址，那么a[3] = 0相当于 i = 0。代码无限循环。