一、引入

同一个问题可以用不同的存储方法来表示，常见的有数组存储和链表存储

二、什么是线性表

由同类型数据元素构成的有序序列的线性结构

特性

- 有长度
- 没有 元素时为空表
- 起始位置为表头，结束位置为表尾

三、线性表的抽象数据类型

基本操作：

1. 初始化一个线性表

（1）数组：定义一个线性表结构体，包含一个数组和表尾位置last

（2）链表：

2. 根据位序查找元素

（1）数组：按下标访问

（2）链表：帮助指针首先指向第一个元素i=1，当p!=null&&i<k时循环，如果停在i=k就找到了，否则就是没找着

3. 查找某元素第一次出现的位置

（1）数组：循环判断，i=0;while(i<=last&&data[i]!=x)

结束有两种情况，一是找到了，返回i，二是没找到返回-1

（2）链表：while(p!=null&&p->data!=tiem)

4. 在某位序前插入一个元素

（1）数组：在第i个位置插入一个元素，代表在数组中将last到i-1号元素依次向后挪一位，最后last++代表表长加1

注意：插入操作需要判断表是否已经满了 last==maxsize-1 以及判断插入的位置合不合法 i<1||i>last+2

（2）链表：先找到第i-1个节点位置的指针，此时得分类讨论

若i==1,则插在头节点前面，直接将新建节点的指针指向头节点，返回指向新建节点的新的头指针

若i!=1,且1<i<=n+1 就先找到i-1位置的节点的指针p=findkth(i-1,ptrl s)

5. 返回线性表的长度

（1）数组：list的legth=last+1

（2）链表：通过帮助指针遍历链表的方法求表长

6. 删除第i个位置的元素

（1）数组：从i-1循环到last-1，然后last--

（2）链表:先找到第i-1个节点位置的指针，此时得分类讨论

若i==1,则删除头节点，先再用一个s指向头节点，将头节点指针指向头节点下以恶搞，再free(s)

若i!=1,且1<i<=n+1 就先找到i-1位置的节点的指针p=findkth(i-1,ptrl s)

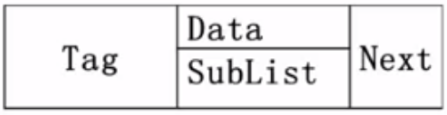
四、广义表

广义表(Generalized List)

- 广义表是线性表的推广
- 对于线性表而言，n个元素都是基本的单元素；
- 广义表中，这些元素不仅可以是单元素也可以是另一个广义表。

```
typedef struct GNode *GList;
struct GNode{
    int Tag;            /*标志域：0表示结点是单元素，1表示结点是广义表 */
    union {             /*子表指针域Sublist与单元素数据域Data复用，即共用存储空间*/
      ElementType  Data;
      GList  SubList;
    } URegion;
    GList Next;         /* 指向后继结点 */
};
```

共用体中要么是子表的指针要么是单元素

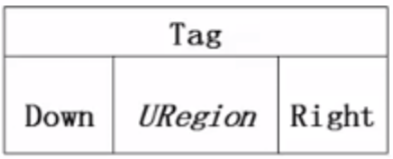


五、多重链表

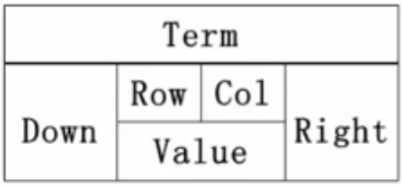
理解：每个节点的指针域有多个

实例：十字链表存储矩阵

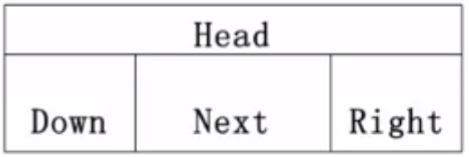
- 用一个标识域Tag来区分头结点和非0元素结点：
- 头节点的标识值为“Head”，矩阵非0元素结点的标识值为“Term”



(a) 结点的总体结构



(b) 矩阵非0元素结点



(c) 头结点

❖ 矩阵A的多重链表图