











ADT List

{

数据元素（对象）：

元素之间存在的关系：

抽象运算：

InitList(L)

输入：线性表的变量L

输出：构造一个空表

Traverse(L):

输入：线性表的变量L

输出：…

…

}ADT List;

顺序存储结构：顺序表（逻辑相邻，物理相邻）.

线性表存储结构

链式存储结构：链表（逻辑相邻，但物理可能不相邻）

逻辑结构(ADT)->设计存储结构->实际运算

问题：删除顺序表中所有值为item的元素：

（一）方法一

算法思想/步骤：

1. 判断顺序表L中是否有值为item的元素
2. 若存在，删除该元素。回到（1）
3. 若不存在，退出

算法/代码：

Void deleteAllitem(Sqlist \*&L, Elemtype item)

{

Elemtype e;

while(LocateElem(L，item)!=0) //检查L中是否有item，如果有返回的为逻辑位置，反之返回0

i = LocateElem(L，item);

ListDelete(L，i); //删除l中当前位置的元素

}

（二）方法二

算法思想/步骤：

1. 用变量k记录顺序表L中不等于item的元素个数，k初始化为0
2. 采用类似建立顺序表的思想，从前向后建立顺序表，查找值不为item 的元素
3. 若找到，则利用原表的空间记录不为item的元素，同时使得k增加1
4. 遍历结束后，顺序表前k个元素即为值不为item的元素，最后将顺序表的长度置为K

算法：

typedef struct

{ ElemType data[MaxSize]; //存放顺序表元素

int length; //存放顺序表的长度

} SqList;

void delnode1(SqList \*L， ElemType item)

{ int k=0， i; //k记录值不等于item的元素个数

for (i=0;i<L->length;i++)

if (L->data[i]!=item) //若当前元素不为item，将其插入A中

{ L->data[k]=L->data[i];

k++; //不等于x的元素增1

}

L->length=k; //顺序表L的长度等于k

}

时间复杂度为O(n)、空间复杂度为O(1)

线性表的链式存储结构：单链表

Typedef stuct LNode

{

ElemType data;

stuct LNode \* next;

}LinkNode, \*LinkList;

stuct LNode \* L;

LinkNode \* L;

LinkList L;

相关运算：

1. 基于遍历
2. 基于插入
3. 基于删除
4. 基于建表

头插法

尾插法