第一章　概论

考点一：数据结构

1、数据结构是研究数据元素及其之间的相互关系和数据运算的一门学科，它包含数据的逻辑结构、存储结构和数据的运算。

2、著名计算机科学家沃思曾指出：算法+ 数据结构=程序。

考点二：基本概念和常用术语

1、数据是描述客观事物的数、字符以及能输入计算机处理的符号的集合。

2、数据元素是数据的基本单位，数据项是具有独立含义的最小标识单位。

3、数据对象是具有相同性质的数据元素的集合，是数据的一个子集。

4、数据的逻辑结构分为线性结构和非线性结构。

（1）线性结构是 n 个数据元素的有序(次序)集合。

（2）非线性结构的逻辑特征是一个结点元素可能对应多个直接前驱和多个后驱。

（3）常用的线性结构有：线性表，栈，队列，双队列，数组，串。

（4）常见的非线性结构有：广义表，树(二叉树等)，图(网等)。

5、逻辑结构是反映数据元素之间逻辑关系的数据结构。逻辑结构反应数据之间的关系，常见的逻辑结构有：集合、线性结构、树形结构、图状结构。

6、四种基本存储方法：顺序存储方法、链法存储方法、索引存储方法、散列存储方法。

考点三：算法

1、算法必须满足的五个准则：

（1）输入。算法开始前必须给算法中用到的变量初始化，一个算法的输入可以包含零个或多个数据。

（2）输出。算法至少有一个或多个输出。

（3）有穷性。算法中每一条指令的执行次数都是有限的，而且每一步都在有穷时间内完成，即算法必须在执行有限步后结束。

（4）确定性。算法中每一条指令的含义都必须明确，无二义性。

（5）可行性。算法是可行的，即算法中描述的操作都可以通过有限次的基本运算来实现。

2、评价算法的优劣：算法的"正确性"是首先要考虑的。此外，主要考虑如下三点：

① 执行算法所耗费的时间，即时间复杂性；

② 执行算法所耗费的存储空间，主要是辅助空间，即空间复杂性；

③ 算法应易于理解、易于编程，易于调试等，即可读性和可操作性。

3、一个算法的时间复杂度(时间复杂性)T(n)就是该算法的时间耗费，它是该算法所求规模 n 的函数。当问题规模 n 趋近于无穷大时，我们把时间复杂度 T(n)的数量级(阶)称为算法的渐近时间复杂度，简称时间复杂度。

4、常见时间复杂性量级：

（1）常数阶：O（1）即算法的时间复杂性与输入规模 n 无关或 n 恒为常数。

(2) 对数阶：O（log_2n）

(3) 线性阶：O(n)

(4) 多项式阶：O（n^2），常见的多项式阶 O（n^2）和 O（n^2）

(5) 指数阶：O（C^n），常见的指数阶 O（2^n）

第二章　线性表

考点一：线性表的基本概念

1、线性表是最简单和最常用的一种数据元素，它是由 n（n≥0）个数据元素（结点）组成的有穷序列。其中结点个数 n 称为表长。当 n=0 时，线性表不含任何数据元素，称为空表。当 n＞0 时，线性表通常表示成（a_1，a_2，…，a_n）。

2、非空线性表的逻辑特征：

（1）有且仅有一个称为开始元素的 a_1，它没有前趋，仅有一个直接后继 a_2；

（2）有且仅有一个称为终端元素的 a_n，它没有后继，仅有一个直接前趋；

（3）其余元素 a_i(2≤i≤n-1)称为内部元素，它们都有且仅有一个直接前趋 a_i-1 和一个直接后继 a_i+1。

3、线性表的基本运算：

（1）初始化 Initiate（L）：建立一个空表 L=（）。

（2）求表长 ListLength（L）：返回线性表 L 的长度。

（3）读表元素 GetNode（L，i）：返回线性表第 i 个元素，当不满足（1≤i≤ListLength（L））时，返回一特殊值。

（4）按值查找 LocateNode（L，x）：查找线性表中数据元素等于 x 的结点序号，若有多个数据元素值与 x 相等，运算结果为这些节点中序号的最小值，若找不到该结点，则运算结果为 0。

（5）插入 Insert(L，X，i)：在线性表 L 的第 i 个位置上插入一个值为 x 的新数据元素，参数 i 的合法取值范围是 1≤i≤n+1。

（6）删除 Delete（L，i）：删除 L 的第 i 个结点 a_i，表 L 的长度减 1。

考点二：线性表的顺序存储

1、顺序存储方法：把线性表的数据元素按逻辑次序依次存放在一组地址连续的存储单元里的方法。

2、顺序表：用顺序存储方法存储的线性表称为顺序表。顺序表是一种随机存取结构，顺序表的特点是逻辑上相邻的结点其物理位置亦相邻。

3、线性表中第 i 个元素 a_i 的存储地址：　LOC（a_i）= LOC（a_1）+（i-1）*d　1≤i≤n，d 代表存储单元。

4、顺序表上实现的基本运算：

（1）插入：该算法的平均时间复杂度是 O(n)，即在顺序表上进行插入运算，平均要移动一半结点（n/2），在第 i 个位置插入一个结点的移动次数为 n-i+1。

（2）删除：顺序表上做删除运算，平均要移动表中约一半的结点（n-1）/2，平均时间复杂度也是 O(n)，删除第 i 个结点移动次数为 n-i。

考点三：线性表的链接存储——单链表

5、采用链式存储结构可以避免频繁移动大量元素。一个单链表可由头指针唯一确定，因此单链表可以用头指针的名字来命名。

（1）生成结点变量的标准函数　p=(ListNode *)malloc(sizeof(ListNode))；　//函数 malloc 分配一个类型为 ListNode 的结点变量的空间，并将其首地址放入指针变量 p 中

（2）释放结点变量空间的标准函数 free(p)；//释放 p 所指的结点变量空间

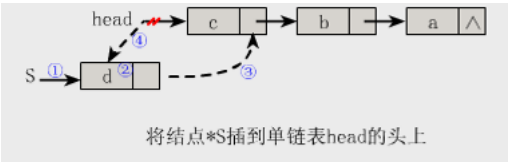
（3）p->data 和 p->next 分别表示指针的数据域变量和指针域变量。

（4）指针变量 p 和结点变量*p 的关系： 指针变量 p 的值——结点地址，结点变量*p 的值——结点内容。

6、建立单链表：

（1）头插法建表

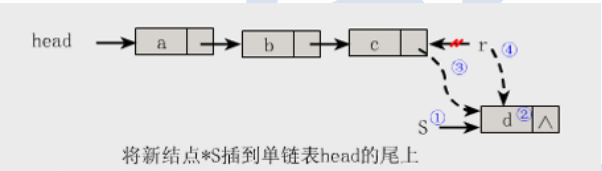
```
算法：  p=(ListNode *)malloc(sizeof(ListNode));        //①生成新结点
        p->data=ch;                                    //②数据域赋值
        p->next=head;                                  //③指针域赋值
        head=p;                                        //④头指针指向新结点
```



将结点*S插到单链表head的头上

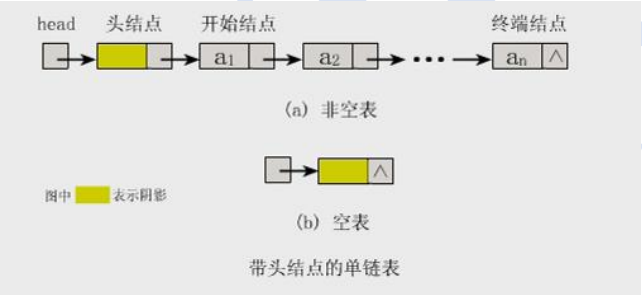
（2）尾插法建表

```
算法：  p=(ListNode *)malloc(sizeof(ListNode));    //①生成新结点
        p->data=ch;                                //②数据域赋值
        if (head==NULL) head=p;                    //新结点*p 插入空表
        else rear->next=p;                         //③将新结点*p 插入到*rear 之后
        rear=p;                                    //④表尾指针指向新的表尾结点
```



将新结点*S插到单链表head的尾上

（3） 尾插法建带头结点的单链表：在链表的开始结点之前附加一个结点并称其为头结点。



带头结点的单链表

具体算法：r=head；// 尾指针初值也指向头结点

```
        while((ch=getchar( ))!='\n'){
            p=(ListNode *)malloc(sizeof(ListNode));          //生成新结点
            p->data=ch;
        r->next=p;                    //新结点连接到尾结点之后
        r=p;                          //尾结点指向新结点
    }
```

```
        r->next=NULL;                    //终端结点的指针域置空
```

以上三个算法的时间复杂度均为 O(n)。

7、单链表上的查找：（带头结点）

（1）按结点序号查找：序号为 0 的是头结点。算法：

```
p=head->next; j=1;                  //使 p 指向第一个结点，j 置为 1
    while(p! =NULL && j<i){         //顺指针向后查找，直到 p 指向第 i 个结点或 p 为空为止
      p=p->next;          ++j;
}
    if(j==i)            return p;
    else        return NULL;
```

在等概率假设下，平均时间复杂度为：为 n/2=O(n)

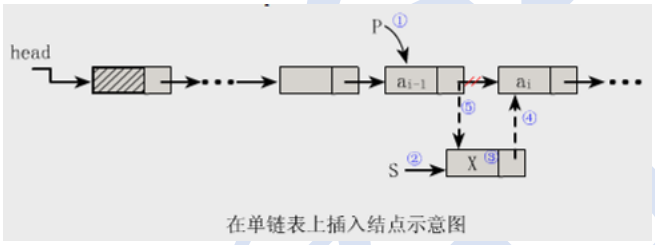
（2）按结点值查找：

具体算法：

```
ListNode *p=head->next;                 //p 指向开始结点
    while(p&&p->data!=k)                //循环直到 p 为 NULL 或 p->data 为 k 为止
        p=p->next;                      //指针指向下一结点
    return p;                                   //若找到值为 k，则指向该结点，否则 p 为 null
```

循环最坏情况下执行 n 次，因此时间复杂度为：O(n)。

8、插入运算：插入运算是将值为 x 的新结点插入到表的第 i 个结点的位置上，即插入到 a_{i-1} 与 a_i 之间。

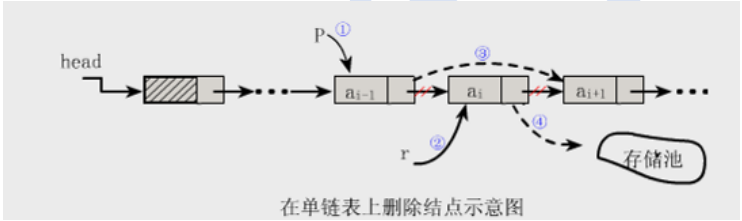


在单链表上插入结点示意图

③s->data=x;　　④s->next=p->next;　　⑤p->next=s;

算法的时间主要耗费在查找结点上，故时间复杂度亦为 O(n)。

9、删除运算



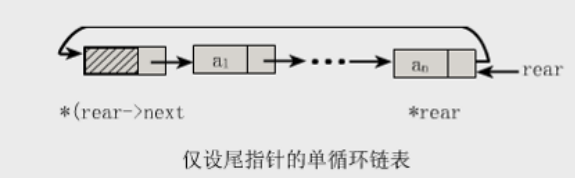
在单链表上删除结点示意图

②r=p->next;　　③p->next=r->next;　　④free(r);

算法的时间复杂度也是 O（n）。链表上实现的插入和删除运算，无须移动结点，仅需修改指针。

10、单循环链表—在单链表中，将终端结点的指针域 NULL 改为指向表头结点或开始结点即可。判断空链表的条件是 head==head->next;

11、仅设尾指针的单循环链表： 用尾指针 rear 表示的单循环链表对开始结点 a_1 和终端 结点 a_n 查找时间都是 O(1)。而表的操作常常是在表的首尾位置上进行，因此，实用中多采用尾指针表示单循环链表。判断空链表的条件为 rear==rear->next。



仅设尾指针的单循环链表

考点四：循环链表

1、循环链表的特点是单链表中最后一个结点（终端点）的指针域不为空，而是指向链表的头结点，使整个链表构成一个环。

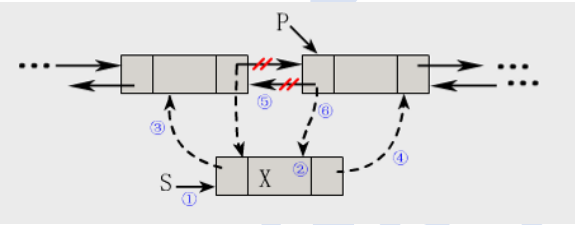
2、循环结束的条件不再是 p 或 p->next 是否为空，而是它们是否等于头指针。

考点五：双向循环链表

1、双向链表： 双（向）链表中有两条方向不同的链，即每个结点中除 next 域存放后继结点地址外，还增加一个指向其直接前趋的指针域 prior。

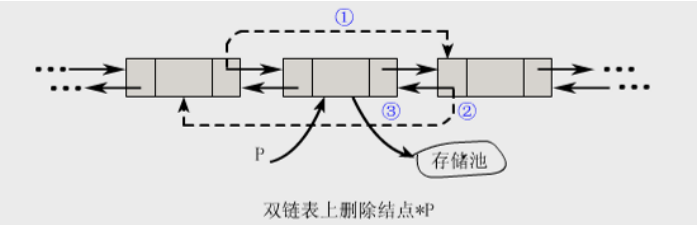
2、双向链表的插入和删除本结点操作

（1）双链表的插入操作




```
void DLInsert(DLNode *p，DataType x)
{    //将值为 x 的新结点插入*p 之前
     DListNode *s=malloc(sizeof(DListNode))；//①
          s->data=x;                  //②
     s->prior=p->prior;          //③
     s->next=p;                  //④
     p->prior->next=s;                //⑤
          p->prior=s;                 //⑥
}
```

（2）双链表上删除结点*p 自身的操作



双链表上删除结点*P

```
void DDeleteNode(DListNode *p)
{    //在带头结点的双链表中，删除结点*p，设*p 为非终端结点
     p->prior->next=p->next;          //①
        p->next->prior=p->prior;          //②
        free(p);                          //③
}
```