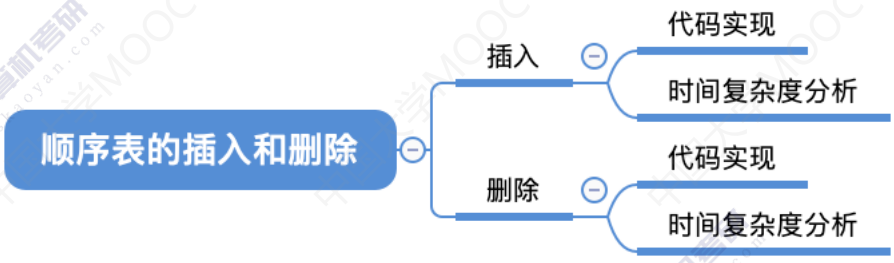
本节内容

顺序表

插入
删除

1

知识总览



2

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

位序

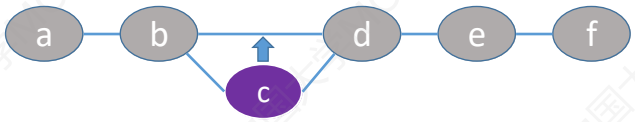
ListInsert(&L,i,e)：插入操作。在表L中的第i个位置上插入指定元素e。

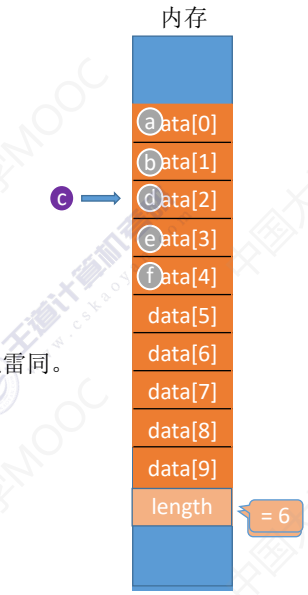
```
#define MaxSize 10      //定义最大长度
typedef struct{
    ElemType data[MaxSize];//用静态的"数组"存放数据元素
    int length;        //顺序表的当前长度
}SqList;               //顺序表的类型定义
```

用静态分配方式实现的顺序表

注：本节代码建立在顺序表的“静态分配”实现方式之上，“动态分配”也雷同。

逻辑结构：





3

顺序表的基本操作——插入

```
#define MaxSize 10      //定义最大长度
typedef struct{
    int data[MaxSize];//用静态的"数组"存放数据元素
    int length;        //顺序表的当前长度
}SqList;               //顺序表的类型定义

void ListInsert(SqList &L,int i,int e){
    for(int j=L.length;j>=i;j--)   //将第i个元素及之后的元素后移
        L.data[j]=L.data[j-1];
    L.data[i-1]=e;                 //在位置i处放入e
    L.length++;                    //长度加1
}

int main() {
    SqList L;          //声明一个顺序表
    InitList(L);       //初始化顺序表
    //...此处省略一些代码，插入几个元素
    ListInsert(L, 3, 3);
    return 0;
}
```

基本操作：在L的位序 i 处插入元素e

注意位序、数组下标的关系，并从后面的元素依次移动

ListInsert(L, 9, 3); ③

我可能就是个天才吧

基操——让自己实现的数据结构可以让别人很方便地使用

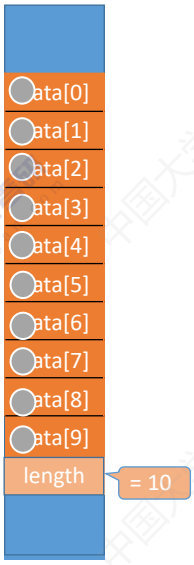
4

顺序表的基本操作——插入

```
#define MaxSize 10      //定义最大长度
typedef struct{
    int data[MaxSize];//用静态的"数组"存放数据元素
    int length;       //顺序表的当前长度
}SqList;              //顺序表的类型定义

bool ListInsert(SqList &L,int i,int e){
    if(i<1||i>L.length+1)          //判断i的范围是否有效
        return  false;
    if(L.length>=MaxSize)          //当前存储空间已满，不能插入
        return  false;
    for(int j=L.length;j>=i;j--)   //将第i个元素及之后的元素后移
        L.data[j]=L.data[j-1];
    L.data[i-1]=e;                 //在位置i处放入e
    L.length++;                    //长度加1
    return  true;
}
```

好的算法，应该具有“健壮性”。
能处理异常情况，并给使用者反馈



王道考研/CSKAOYAN.COM

5

插入操作的时间复杂度

```
bool ListInsert(SqList &L,int i,int e){
    if(i<1||i>L.length+1)          //判断i的范围是否有效
        return  false;
    if(L.length>=MaxSize)          //当前存储空间已满，不能插入
        return  false;
    for(int j=L.length;j>=i;j--)   //将第i个元素及之后的元素后移
        L.data[j]=L.data[j-1];
    L.data[i-1]=e;                 //
    L.length++;                    //长度加1
    return  true;
}
```

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）

最好情况：新元素插入到表尾，不需要移动元素
i = n+1，循环0次；最好时间复杂度 = O(1)

最坏情况：新元素插入到表头，需要将原有的 n 个元素全都向后移动
i = 1，循环 n 次；最坏时间复杂度 = O(n);

平均情况：假设新元素插入到任何一个位置的概率相同，即 i = 1,2,3, … , length+1 的概率都是 $p = \frac{1}{n+1}$
i = 1，循环 n 次；i=2 时，循环 n-1 次；i=3，循环 n-2 次 …… i =n+1时，循环0次

平均循环次数 = $np + (n-1)p + (n-2)p + \cdots\cdots + 1\cdot p = \frac{n(n+1)}{2}\frac{1}{n+1} = \frac{n}{2}$ ⇒ 平均时间复杂度 = O(n)

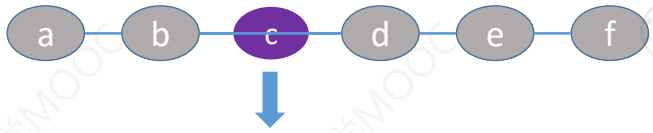
王道考研/CSKAOYAN.COM

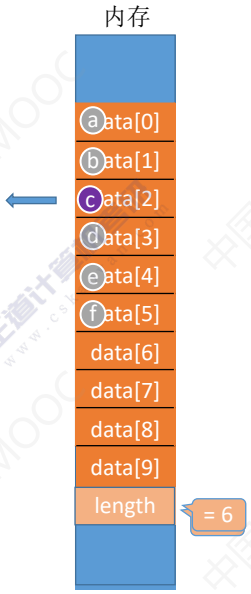
6

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。

逻辑结构：





7

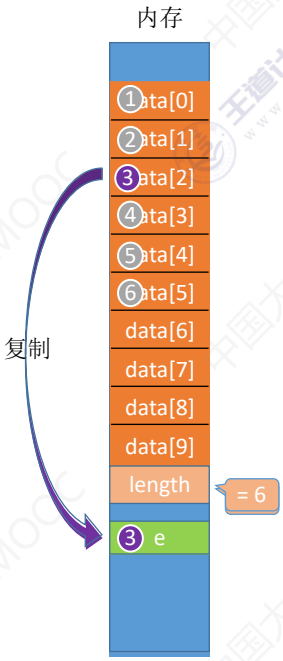
顺序表的基本操作——删除

```
bool ListDelete(SqList &L,int i,int &e){
    if(i<1||i>L.length)            //判断i的范围是否有效
        return  false;
     e=L.data[i-1];                //将被删除的元素赋值给e
    for(int j=i;j<L.length;j++)    //将第i个位置后的元素前移
        L.data[j-1]=L.data[j];
    L.length--;                    //线性表长度减1
    return  true;
}
int main() {
    SqList L;          //声明一个顺序表
    InitList(L);       //初始化顺序表
    //...此处省略一些代码，插入几个元素
    int e = -1;        //用变量e把删除的元素"带回来"
    if (ListDelete(L, 3, e))
        printf("已删除第3个元素,删除元素值为=%d\n", e);
    else
        printf("位序i不合法，删除失败\n");
    return 0;
}
```

注意位序、数组下标的关系，并从前面的元素依次移动

如果参数没有加引用符号，会怎样？


```
已删除第3个元素,删除元素值为=3
Program ended with exit code: 0
```



8

删除操作的时间复杂度

```
bool ListDelete(SqList &L,int i,int &e){
    if(i<1||i>L.length)          //判断i的范围是否有效
        return  false;
     e=L.data[i-1];              //将被删除的元素赋值给e
    for(int j=i;j<L.length;j++)  //将第i个位置后的元素前移
        L.data[j-1]=L.data[j];
    L.length--;                  //
    return  true;
}
```

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）

最好情况：删除表尾元素，不需要移动其他元素
i = n，循环 0 次；最好时间复杂度 = O(1)

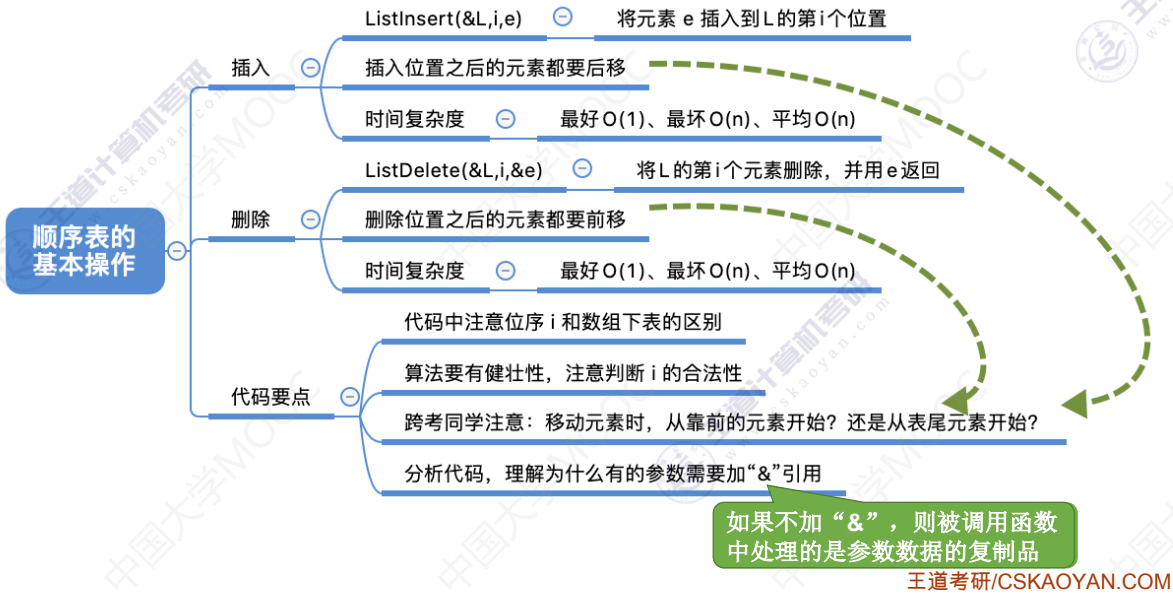
最坏情况：删除表头元素，需要将后续的 n-1 个元素全都向前移动
i = 1，循环 n-1 次；最坏时间复杂度 = O(n);

平均情况：假设删除任何一个元素的概率相同，即 i = 1,2,3, … , length 的概率都是 $p=\frac{1}{n}$
i = 1，循环 n-1 次；i=2 时，循环 n-2 次；i=3，循环 n-3 次 …… i =n 时，循环0次

平均循环次数 = (n-1)p + (n-2)p + …… + 1·p = $\frac{n(n-1)}{2}\frac{1}{n}=\frac{n-1}{2}$ ⇒ 平均时间复杂度 = O(n)

9

知识回顾与重要考点



10


bilibili
@王道论坛
新浪微博 weibo.com
@王道计算机考研备考
@王道咸鱼老师-计算机考研
@王道楼楼老师-计算机考研
抖音 短视频
@王道计算机考研
等撩
等撩
知乎
@王道计算机考研
微信视频号
@王道计算机考研
微信公众平台
@王道在线

11