本小节内容

顺序表的删除及查询实战

顺序表的删除及查询实战

因为是实战，自然是全程手撸代码，下面直接上代码，我们的顺序表的删除及查询实战，是在原来的插入基础之上来添加，因此咱们先把 10.3 小节的项目代码复制一遍，在其基础之上进行添加：

```
#include <stdio.h>

#define MaxSize 50
typedef int ElemType;
//静态分配
typedef struct{
	ElemType data[MaxSize];
	int length;//当前顺序表中有多少个元素
}SqList;
//动态分配
#define InitSize 100
typedef struct{
	ElemType *data;
	int capacity;//动态数组的最大容量
	int length;
}SeqList;
//i 代表插入的位置，从 1 开始，e 要插入的元素
bool ListInsert(SqList &L,int i,ElemType e)
{
	if(i<1||i>L.length+1)//判断要插入的位置是否合法
		return false;
	if(L.length>=MaxSize)//超出空间了
		return false;
	for(int j=L.length;j>=i;j--)//移动顺序表中的元素
		L.data[j]=L.data[j-1];
	L.data[i-1]=e;//数组下标从零开始，插入第一个位置，访问的下标为 0
	L.length+=1;
	return true;
}
//删除使用元素 e 的引用的目的是拿出对应的值
bool ListDelete(SqList &L,int i,ElemType &e)
{
```

```
    if(i<1||i>L.length)//如果删除的位置是不合法
        return false;
    e=L.data[i-1];//获取顺序表中对应的元素，赋值给 e
    for(int j=i;j<L.length;j++)
        L.data[j-1]=L.data[j];
    L.length--=1;//删除一个元素，顺序表长度减 1
    return true;
}
//查找成功，返回位置，位置从 1 开始，查找失败，返回 0
int LocateElem(SqList L,ElemType e)
{
    int i;
    for(i=0;i<L.length;i++)
        if(L.data[i]==e)
            return i+1;//加 1 就是元素在顺序表中的位置
    return 0;
}
//打印顺序表元素
void PrintList(SqList L)
{
    for(int i=0;i<L.length;i++)
    {
        printf("%4d",L.data[i]);
    }
    printf("\n");
}
int main()
{
    SqList L;//顺序表的名称
    bool ret;//查看返回值，布尔型是 True,或者 False
    ElemType del;//要删除的元素
    //首先手动在顺序表中赋值
    L.data[0]=1;
    L.data[1]=2;
    L.data[2]=3;
    L.length=3;//总计三个元素
    ret=ListInsert(L,2,60);
    if(ret)
    {
        printf("插入成功\n");
        PrintList(L);
    }else{
        printf("插入失败\n");
    }
```

```
    ret=ListDelete(L,1,del);
    if(ret)
    {
        printf("删除成功\n");
        printf("删除元素值为 %d\n",del);
        PrintList(L);
    }else{
        printf("删除失败\n");
    }
    ret=LocateElem(L,60);
    if(ret)
    {
        printf("查找成功\n");
        printf("元素位置为 %d\n",ret);
    }else{
        printf("查找失败\n");
    }
    return 0;
}
```