```c
/*
线性表及其实现
    线性表：
        由同类型数据元素构成有序序列的线性结构

    顺序存储实现：
        利用数组的连续存储空间顺序存放线性表的各个元素
*/


#include<stdio.h>
#include<malloc.h>
#define MAXSIZE 100 // MAXSIZE 定义为 Data 数组的大小

typedef int ElementType; // ElementType 可定义为任意类型
typedef struct LNode *List;
struct LNode{
   ElementType Data[MAXSIZE];
   int Last; // 线性表最后一个元素的下标，从-1开始记录
};
List L;

// 访问下标为 i 的元素：L->Data[i]
// 线性表的长度：L->Last+1

List MakeEmpty(); // 初始化顺序表

void Insert(ElementType X, int i, List L); // 在下标为 i 的地方插入 X

int Find(ElementType X, List L); // 查找 X 第一次出现的下标

void Delete(int i, List L); // 删除下标为 i 的当前值

ElementType FindKth(int K, List L); // 返回下标为 K 的当前值

int Length(List L); //返回顺序表的长度

// 初始化
List MakeEmpty()
{
    // sizeof(struct LNode) == 404，也为Last分配了空间
    List L;
    L = (List)malloc(sizeof(struct LNode));
    L->Last = -1;
    return L;
}

// 插入
void Insert(ElementType X, int i, List L)
{
    int j;
    if(L->Last == MAXSIZE-1) // 位置已满
    {
        printf("表满");
        return;
    }
    // 位置越界，设定 i 属于 [0, Last+1]
    if(i<0 || i>L->Last+1)
    {
        printf("位置不合法");
        return;
    }
    // 从后往前依次向后挪一个，给 a[i] 腾出位置
    for(j=L->Last; j>=i; j--)
        L->Data[j+1] = L->Data[j];
    L->Data[i] = X; // 新元素插入
    L->Last++; // Last仍然指向最后元素
    return;
}

// 查找值为 X 的元素，返回元素下标
int Find(ElementType X, List L)
{
```

```c
    int i = 0;
    while(i<=L->Last && L->Data[i]!=X)
        i++;
    if(i > L->Last) // 如果没找到，返回 -1
        return -1;
    else // 找到后返回下标
        return i;
}

// 删除位置 i 的元素，设定 i 属于[0, Last]
void Delete(int i, List L)
{
    int j;
    // 位置越界
    if(i<0 || i>L->Last)
    {
        printf("L->Data[%d]不存在元素", i);
        return;
    }
    // 从前往后依次向前挪一个，将 a[i] 覆盖
    for(j=i; j<L->Last; j++)
        L->Data[j] = L->Data[j+1];
    L->Last--; // Last仍然指向最后元素
    return;
}

// 查找位置 K 的元素值
ElementType FindKth(int K, List L)
{
    if(K<0 || K>L->Last)
    {  // 位置越界
        printf("L->Data[%d]不存在元素", K);
        return;
    }
    return L->Data[K];
}

// 表长
int Length(List L)
{
    return L->Last+1;
}

int main()
{
    int i = 0;
    L = MakeEmpty();
    Insert(11, 0, L);
    printf("在线性表L-Data[0]插入11\n");
    Insert(25, 0, L);
    printf("在线性表L-Data[0]插入25\n");
    Insert(33, 0, L);
    printf("在线性表L-Data[0]插入33\n");
    Insert(77, 0, L);
    printf("在线性表L-Data[0]插入77\n");
    printf("此时的线性表为: ");
    for(i=0; i<Length(L); i++)
        printf("%d ", L->Data[i]);
    printf("\n");
    printf("查找值为12的下标是: %d\n", Find(12, L));
    printf("下标为3的线性表的值是: %d\n", FindKth(3, L));
    Delete(2, L);
    printf("删除线性表中下标为2的元素\n");
    Delete(2, L);
    printf("删除线性表中下标为2的元素\n");
    printf("此时的线性表为: ");
    for(i=0; i<Length(L); i++)
        printf("%d ", L->Data[i]);
    printf("\n");
    return 0;
}

```