

```

1  /*
2  线性表及其实现
3      线性表：
4          由同类型数据元素构成有序序列的线性结构
5
6      顺序存储实现：
7          利用数组的连续存储空间顺序存放线性表的各个元素
8  */
9
10
11 #include<stdio.h>
12 #include<malloc.h>
13 #define MAXSIZE 100 // MAXSIZE 定义为 Data 数组的大小
14
15 typedef int ElementType; // ElementType 可定义为任意类型
16 typedef struct LNode *List;
17 struct LNode{
18     ElementType Data[MAXSIZE];
19     int Last; // 线性表最后一个元素的下标，从-1开始记录
20 };
21 List L;
22
23 // 访问下标为 i 的元素：L->Data[i]
24 // 线性表的长度：L->Last+1
25
26 List MakeEmpty(); // 初始化顺序表
27
28 void Insert(ElementType X, int i, List L); // 在下标为 i 的地方插入 x
29
30 int Find(ElementType X, List L); // 查找 x 第一次出现的下标
31
32 void Delete(int i, List L); // 删除下标为 i 的当前值
33
34 ElementType FindKth(int K, List L); // 返回下标为 K 的当前值
35
36 int Length(List L); //返回顺序表的长度
37
38 // 初始化
39 List MakeEmpty()
40 {
41     // sizeof(struct LNode) == 404，也为Last分配了空间
42     List L;
43     L = (List)malloc(sizeof(struct LNode));
44     L->Last = -1;
45     return L;
46 }
47
48 // 插入
49 void Insert(ElementType X, int i, List L)
50 {
51     int j;
52     if(L->Last == MAXSIZE-1) // 位置已满
53     {
54         printf("表满");
55         return;
56     }
57     // 位置越界，设定 i 属于 [0, Last+1]
58     if(i<0 || i>L->Last+1)
59     {
60         printf("位置不合法");
61         return;
62     }
63     // 从后往前依次向后挪一个，给 a[i] 腾出位置
64     for(j=L->Last; j>=i; j--)
65         L->Data[j+1] = L->Data[j];
66     L->Data[i] = X; // 新元素插入
67     L->Last++; // Last仍然指向最后元素
68     return;
69 }
70
71 // 查找值为 x 的元素，返回元素下标
72 int Find(ElementType X, List L)
73 {

```

```

74     int i = 0;
75     while(i<=L->Last && L->Data[i]!=X)
76         i++;
77     if(i > L->Last) // 如果没找到, 返回 -1
78         return -1;
79     else // 找到后返回下标
80         return i;
81 }
82
83 // 删除位置 i 的元素, 设定 i 属于[0, Last]
84 void Delete(int i, List L)
85 {
86     int j;
87     // 位置越界
88     if(i<0 || i>L->Last)
89     {
90         printf("L->Data[%d]不存在元素", i);
91         return;
92     }
93     // 从前往后依次向前挪一个, 将 a[i] 覆盖
94     for(j=i; j<L->Last; j++)
95         L->Data[j] = L->Data[j+1];
96     L->Last--; // Last仍然指向最后元素
97     return;
98 }
99
100 // 查找位置 k 的元素值
101 ElementType FindKth(int K, List L)
102 {
103     if(K<0 || K>L->Last)
104     { // 位置越界
105         printf("L->Data[%d]不存在元素", K);
106         return;
107     }
108     return L->Data[K];
109 }
110
111 // 表长
112 int Length(List L)
113 {
114     return L->Last+1;
115 }
116
117 int main()
118 {
119     int i = 0;
120     L = MakeEmpty();
121     Insert(11, 0, L);
122     printf("在线性表L-Data[0]插入11\n");
123     Insert(25, 0, L);
124     printf("在线性表L-Data[0]插入25\n");
125     Insert(33, 0, L);
126     printf("在线性表L-Data[0]插入33\n");
127     Insert(77, 0, L);
128     printf("在线性表L-Data[0]插入77\n");
129     printf("此时的线性表为: ");
130     for(i=0; i<Length(L); i++)
131         printf("%d ", L->Data[i]);
132     printf("\n");
133     printf("查找值为12的下标是: %d\n", Find(12, L));
134     printf("下标为3的线性表的值是: %d\n", FindKth(3, L));
135     Delete(2, L);
136     printf("删除线性表中下标为2的元素\n");
137     Delete(2, L);
138     printf("删除线性表中下标为2的元素\n");
139     printf("此时的线性表为: ");
140     for(i=0; i<Length(L); i++)
141         printf("%d ", L->Data[i]);
142     printf("\n");
143     return 0;
144 }
145

```