

```

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      线性表的链式存储实现
7          不要求逻辑上相邻的两个元素物理上也相邻
8          通过链建立起数据元素之间的逻辑关系
9
10     广义表：
11         广义表是线性表的推广
12         对于线性表而言，n个元素都是基本的单元素
13         广义表中，这些元素不仅可以是单元素，也可以是另一个广义表
14
15     typedef struct GNode *GList;
16     struct GNode
17     {
18         int Tag; // 标志域，0表示结点是单元素，1表示结点是广义表
19         union
20         {
21             ElementType Data;
22             GList SubList;
23         } URegion; // 子表Sublist与单元素数据域Data复用，共用存储空间
24         GList Next; // 指向后继结点
25     }
26
27     多重链表：
28         多重链表中结点的指针域会有多个，且链表中的结点可能属于多个链
29         包含两个指针域的链表不一定是多重链表，如双向链表不是多重链表
30         在树、图中会广泛的用到多重链表
31
32     十字链表：
33         存储稀疏矩阵
34         结点的数据域：行坐标、列坐标、数值
35         每个结点通过两个指针域，把同行、同列串起来
36         行指针：Right
37         列指针：Down
38 */
39
40 #include <stdio.h>
41 #include <malloc.h>
42
43 typedef int ElementType; // ElementType 可定义为任意类型
44 typedef struct LNode *List;
45 struct LNode{
46     ElementType Data; // 数据域
47     List Next; // 下一个链表的地址
48 };
49 List L;
50
51 List MakeEmpty(); // 初始化链表
52
53 int Length(List L); // 以遍历链表的方法求链表长度
54
55 List FindKth(int K, List L); // 查找链表中第 K 个元素的值
56
57 List Find(ElementType X, List L); // 查找链表中值为 x 的结点
58
59 List Insert(ElementType X, int i, List L); // 将 x 插入到第 i-1(i>0) 个结点之后
60
61 List Delete(int i, List L); // 删除第 i(i>0) 个结点
62
63 void Print(List L); // 输出链表元素
64
65 // 初始化链表
66 List MakeEmpty()
67 {
68     List L = (List)malloc(sizeof(struct LNode));
69     L = NULL; // L 是一个指向结构体的指针
70     return L;
71 }
72
73 // 求表长

```

```

74  int Length(List L)
75  {
76      List p = L;
77      int len = 0;
78      while(p) // 当 p 不为空
79      {
80          len++;
81          p = p->Next;
82      }
83      return len;
84  }
85
86  // 按序查找, 设定 K 属于 [1, ...]
87  List FindKth(int K, List L)
88  {
89      List p = L;
90      int i = 1; // 从第 1 个元素开始查找
91      while(p && i<K)
92      {
93          p = p->Next;
94          i++;
95      }
96      if(i == K) // 找到了
97          return p;
98      else // 未找到
99          return NULL;
100 }
101
102 // 查找值为 x 的元素
103 List Find(ElementType X, List L)
104 {
105     List p = L;
106     while(p && p->Data!=X)
107         p = p->Next;
108     // 找到了, 返回 p
109     // 未找到, 返回 NULL, 此时 p 等于 NULL
110     return p;
111 }
112
113 /*
114 插入, 将 x 插在位置 i, 设定 i 属于 [1, length+1]
115 1. 用 s 指向一个新的结点
116 2. 用 p 指向链表的第 i-1 个结点
117 3. s->Next = p->Next, 将 s 的下一个结点指向 p 的下一个结点
118 4. p->Next = s, 将 p 的下一结点改为 s
119 */
120 List Insert(ElementType X, int i, List L)
121 {
122     List p, s;
123     if(i == 1) // 新结点插入在表头, 单独处理
124     {
125         s = (List)malloc(sizeof(struct LNode));
126         s->Data = X;
127         s->Next = L;
128         return s;
129     }
130     p = FindKth(i-1, L); // 找到第 i-1 个结点
131     if(!p) // 第 i-1 个结点不存在
132     {
133         printf("结点错误");
134         return NULL;
135     }
136     else
137     {
138         s = (List)malloc(sizeof(struct LNode));
139         s->Data = X;
140         s->Next = p->Next; // 将 s 的下一个结点指向 p 的下一个结点
141         p->Next = s; // 将 p 的下一结点改为 s
142         return L;
143     }
144 }
145
146 /*

```

```

147 删除第 i 个结点, 设定 i 属于 [1, n]
148 1. 用 p 指向链表的第 i-1 个结点
149 2. 用 s 指向要被删除的的第 i 个结点
150 3. p->Next = s->Next
151 4. free(s), 释放空间
152 */
153 List Delete(int i, List L)
154 {
155     List p, s;
156     if(i==1) // 删除头结点, 单独处理
157     {
158         s = L;
159         if(L) // 如果不为空
160             L = L->Next;
161         else
162             return NULL;
163         free(s); // 释放被删除结点
164         return L;
165     }
166     p = FindKth(i-1, L); // 查找第 i-1 个结点
167     if(!p || !(p->Next)) // 第 i-1 个或第 i 个结点不存在
168     {
169         printf("结点错误");
170         return NULL;
171     }
172     else
173     {
174         s = p->Next; // s 指向第 i 个结点
175         p->Next = s->Next; // 从链表删除
176         free(s); // 释放被删除结点
177         return L;
178     }
179 }
180
181 // 输出链表元素
182 void Print(List L)
183 {
184     List t;
185     int flag = 1; // 如果链表空, flag == 1
186     printf("当前链表为: ");
187     for(t=L; t; t=t->Next)
188     {
189         printf("%d ", t->Data);
190         flag = 0;
191     }
192     if(flag)
193         printf("NULL");
194     printf("\n");
195 }
196
197 int main()
198 {
199     L = MakeEmpty();
200     Print(L);
201     L = Insert(11, 1, L);
202     L = Insert(25, 1, L);
203     L = Insert(33, 2, L);
204     L = Insert(77, 3, L);
205     Print(L);
206     printf("当前链表长度为: %d\n", Length(L));
207     printf("此时链表中第二个结点的值是: %d\n", FindKth(2, L)->Data);
208     printf("查找22是否在该链表中: ");
209     if(Find(22, L))
210         printf("是! \n");
211     else
212         printf("否! \n");
213     printf("查找33是否在该链表中: ");
214     if(Find(33, L))
215         printf("是! \n");
216     else
217         printf("否! \n");
218     L = Delete(1, L);
219     L = Delete(3, L);

```

```
220     printf("-----删除后-----\n");
221     Print(L);
222     return 0;
223 }
224
```