

## main.c

```
1  #include <stdio.h>
2  #include "SeqList.h"
3  #include "FunctionImpl.h"
4
5  int main() {
6
7      /* 定义链表 */
8      LinkList linklist;
9
10     //初始化链表
11     if (InitLinkList(&linklist)) {
12         puts("初始化单链表成功! \n");
13     } else {
14         puts("初始化单链表失败! \n");
15     }
16
17     //插入
18     ElementType e;
19     puts("输入要插入的元素: \n");
20     for (int i = 0; i < 5; i++) {
21         scanf("%d", &(e.id));
22         InsertElemLinkList(&linklist, e);
23     }
24
25     //输出
26     puts("输出单链表\n");
27     viewElemLinkList(linklist);
28
29     //按位置查找单链表的元素】
30     int post;
31     printf("请输入要查询元素的位置: \n");
32     scanf("%d", &post);
33     FindPostElemLinkList(linklist, post);
34
35     int post1;
36     ElementType type;
37     printf("请输入要添加元素的位置: \n");
38     scanf("%d", &post1);
39     printf("请输入要添加的元素: \n");
40     scanf("%d", &(type.id));
41     InsertPostElemLinkList(&linklist, post1, type);
42
43     puts("输出单链表\n");
44     viewElemLinkList(linklist);
45
46     int post2;
47     ElementType e1;
48     printf("请输入要修改元素的位置: \n");
49     scanf("%d", &post2);
50     printf("请输入要修改的元素: \n");
51     scanf("%d", &(e1.id));
52     updateLinkListElem(&linklist, post2, e1);
```

```

53
54     puts("输出单链表\n");
55     viewElemLinkList(linklist);
56
57     //以元素位置删除
58     int post3;
59     printf("请输入要删除元素的位置: \n");
60     scanf("%d", &post3);
61     DeletePostElemLinkList(&linklist, post3);
62
63     puts("输出单链表\n");
64     viewElemLinkList(linklist);
65
66     //删除全部
67     ArrayElemLinkList(&linklist);
68
69     puts("输出单链表\n");
70     viewElemLinkList(linklist);
71
72     return 0;
73 }

```

## FunctionImpl.h

```

1  //
2  // Created by 14815 on 2021/11/4.
3  //
4
5  #ifndef DEMO08_FUNCTIONIMPL_H
6  #define DEMO08_FUNCTIONIMPL_H
7
8  #include <malloc.h>
9
10 boolean InitLinkList(LinkList *L)
11 {
12     *L = (LinkList) malloc(sizeof(Node));
13     if (*L != NULL)
14     {
15         (*L)->next = NULL;
16         puts("初始化分配空间成功!");
17         return true;
18     } else {
19         return false;
20     }
21 }
22
23 boolean InsertElemLinkList(LinkList *L, ElementType e)
24 {
25     //头插法
26     // LinkList p;
27     // p = (LinkList) malloc (sizeof(Node));
28     // if ( p == NULL )
29     // {
30     //     return false;
31     // }
32     // p->data = e;
33     // p->next = (*L)->next;

```

```

34 // (*L)->next = p;
35 // return true;
36
37 //尾插法
38 LinkList p, s;
39 p = *L;
40 while (p->next != NULL)
41 {
42     p = p->next;
43 }
44 s = (LinkList) malloc (sizeof (Node));
45 s->data = e;
46 s->next = NULL;
47 p->next = s;
48
49 return true;
50 }
51
52 boolean viewElemLinkList(LinkList L)
53 {
54
55     LinkList p;
56     p = L->next;
57     while ( p != NULL )
58     {
59         printf("%d", p->data.id);
60         p = p->next;
61     }
62     puts("\n");
63
64     return true;
65 }
66
67 boolean FindPostElemLinkList(LinkList L, int post)
68 {
69     int j;
70     LinkList p;
71     p = L->next;
72     j = 1;
73     while (p != NULL && j < post) {
74         p = p->next;
75         ++j;
76     }
77     if (!p || j > post) {
78         puts("post指向的元素不存在\n");
79         return false;
80     }
81     printf("post位置的元素为: %d\n", p->data.id);
82     return true;
83 }
84
85 boolean InsertPostElemLinkList(LinkList *L, int post, ElementType e)
86 {
87     int j = 1;
88     LinkList p, s;
89     p = *L;
90     while ( p && j < post)
91     {

```

```

92     p = p->next;
93     ++ j;
94 }
95 if ( !p || j > post)
96 {
97     puts("post位置不存在");
98     return false;
99 }
100 s = (LinkedList) malloc (sizeof (Node));
101 s->data = e;
102 s->next = p->next;
103 p->next = s;
104
105     return true;
106 }
107
108 boolean UpdateLinkedListElem(LinkedList *L, int post, ElementType e)
109 {
110     int i = 0;
111     LinkedList p;
112     p = *L;
113     while ( p && i < post)
114     {
115         p = p->next;
116         ++ i;
117     }
118     p->data = e;
119
120     return true;
121 }
122
123 boolean ArrayElemLinkedList(LinkedList *L)
124 {
125     LinkedList p, q;
126     p = *L;
127     while (p)
128     {
129         q = p;
130         p = p->next;
131         q->next = NULL;
132         free(q);
133     }
134
135     return true;
136 }
137
138 boolean DeletePostElemLinkedList(LinkedList *L, int post)
139 {
140     int j;
141     LinkedList p, q;
142     p = *L;
143     j = 1;
144     while (p->next && j < post)
145     {
146         p = p->next;
147         ++ j;
148     }
149     if (!(p->next) || j > post)

```

```

150     {
151         puts("post位置的元素不存在");
152         return false;
153     }
154     q = p->next;
155     p->next = q->next;
156     printf("post位置的元素为: %d\n", q->data.id);
157     free(q);
158
159     return true;
160 }
161
162 #endif //DEMO08_FUNCTIONAL_H

```

## SeqList.h

```

1  //
2  // Created by 14815 on 2021/11/4.
3  //
4
5  #ifndef DEMO08_SEQLIST_H
6  #define DEMO08_SEQLIST_H
7
8  /**
9   * 链表在修改数据的时候，需要传入一个包含首地址的指针，二级指针
10  * 在遍历输出的时候不需要修改他的元素 所以传入首地址的指针即可
11  */
12
13  typedef enum
14  {
15      false = 0,
16      true = 1
17  } boolean ;
18
19  typedef struct
20  {
21      int id;
22  } ElementType;
23
24  typedef struct Node
25  {
26
27      ElementType data;
28      struct Node *next;
29
30  } Node;
31
32  //定义链表的数据类型
33  typedef struct Node *LinkList;
34
35  /**
36   * 初始化链表
37   * 头插法和尾插法
38   */
39  boolean InitLinkList(LinkList *L);
40  boolean InsertElemLinkList(LinkList *L, ElementType e);
41

```

```
42  /**
43   * 插入数据
44   */
45  boolean InsertPostElemLinkList(LinkList *L, int post, ElementType e);
46
47  /**
48   * 删除数据
49   * 位置删除和全部删除
50   */
51  boolean DeletePostElemLinkList(LinkList *L, int post);
52  boolean ArrayElemLinkList(LinkList *L);
53
54  /**
55   * 查询数据
56   * 按位置查询和遍历输出
57   */
58  boolean ViewElemLinkList(LinkList L);
59  boolean FindPostElemLinkList(LinkList L, int post);
60
61  /**
62   * 按位置修改LinkList的元素
63   */
64  boolean UpdateLinkListElem(LinkList *L, int post, ElementType e);
65
66  #endif //DEMO08_SETLIST_H
```