main.c

```c
#include <stdio.h>
#include "SeqList.h"
#include "FunctionImpl.h"

int main() {

    /* 定义链表 */
    LinkList linklist;

    //初始化链表
    if (InitLinkList(&linklist)) {
        puts("初始化单链表成功！\n");
    } else {
        puts("初始化单链表失败！\n");
    }

    //插入
    ElementType e;
    puts("输入要插入的元素：\n");
    for (int i = 0; i < 5; i ++) {
        scanf("%d", &(e.id));
        InsertElemLinkList(&linklist, e);
    }

    //输出
    puts("输出单链表\n");
    ViewElemLinkList(linklist);

    //按位置查找单链表的元素】
    int post;
    printf("请输入要查询元素的位置：\n");
    scanf("%d", &post);
    FindPostElemLinkList(linklist, post);

    int post1;
    ElementType type;
    printf("请输入要添加元素的位置：\n");
    scanf("%d", &post1);
    printf("请输入要添加的元素：\n");
    scanf("%d", &(type.id));
    InsertPostElemLinkList(&linklist, post1, type);

    puts("输出单链表\n");
    ViewElemLinkList(linklist);

    int post2;
    ElementType e1;
    printf("请输入要修改元素的位置：\n");
    scanf("%d", &post2);
    printf("请输入要修改的元素：\n");
    scanf("%d", &(e1.id));
    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;
```

```
//    (*L)->next = p;
//    return true;

    //尾插法
    LinkList p, s;
    p = *L;
    while (p->next != NULL)
    {
        p = p->next;
    }
    s = (LinkList) malloc (sizeof (Node));
    s->data = e;
    s->next = NULL;
    p->next = s;

    return true;
}

boolean ViewElemLinkList(LinkList L)
{

    LinkList p;
    p = L->next;
    while ( p != NULL )
    {
        printf("%d", p->data.id);
        p = p->next;
    }
    puts("\n");

    return true;
}

boolean FindPostElemLinkList(LinkList L, int post)
{
    int j;
    LinkList p;
    p = L->next;
    j = 1;
    while (p != NULL && j < post) {
        p = p->next;
        ++j;
    }
    if (!p || j > post) {
        puts("post指向的元素不存在\n");
        return false;
    }
    printf("post位置的元素为：%d\n", p->data.id);
    return true;
}

boolean InsertPostElemLinkList(LinkList *L, int post, ElementType e)
{
    int j = 1;
    LinkList p, s;
    p = *L;
    while ( p && j < post)
    {
```

```
        p = p->next;
        ++ j;
    }
    if ( !p || j > post)
    {
        puts("post位置不存在");
        return false;
    }
    s = (LinkList) malloc (sizeof (Node));
    s->data = e;
    s->next = p->next;
    p->next = s;

    return true;
}

boolean UpdateLinkListElem(LinkList *L, int post, ElementType e)
{
    int i = 0;
    LinkList p;
    p = *L;
    while ( p && i < post)
    {
        p = p->next;
        ++ i;
    }
    p->data = e;

    return true;
}

boolean ArrayElemLinkList(LinkList *L)
{
    LinkList p, q;
    p = *L;
    while (p)
    {
        q = p;
        p = p->next;
        q->next = NULL;
        free(q);
    }

    return true;
}

boolean DeletePostElemLinkList(LinkList *L, int post)
{
    int j;
    LinkList p, q;
    p = *L;
    j = 1;
    while (p->next && j < post)
    {
        p = p->next;
        ++ j;
    }
    if (!(p->next) || j > post)
```

```
    {
        puts("post位置的元素不存在");
        return false;
    }
    q = p->next;
    p->next = q->next;
    printf("post位置的元素为：%d\n", q->data.id);
    free(q);

    return true;
}

#endif //DEMO08_FUNCTIONAL_H
```

SeqList.h

```
//
// Created by 14815 on 2021/11/4.
//

#ifndef DEMO08_SEQLIST_H
#define DEMO08_SEQLIST_H

/**
 * 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
 * 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
 */

typedef enum
{
    false = 0,
    true = 1
} boolean ;

typedef struct
{
    int id;
}ElementType;

typedef struct Node
{

    ElementType data;
    struct Node *next;

} Node;

//定义链表的数据类型
typedef struct Node *LinkList;

/**
 * 初始化链表
 * 头插法和尾插法
 */
boolean InitLinkList(LinkList *L);
boolean InsertElemLinkList(LinkList *L, ElementType e);

```

```
/**
 * 插入数据
 */
boolean InsertPostElemLinkList(LinkList *L, int post, ElementType e);

/**
 * 删除数据
 * 位置删除和全部删除
 */
boolean DeletePostElemLinkList(LinkList *L, int post);
boolean ArrayElemLinkList(LinkList *L);

 /**
  * 查询数据
  * 按位置查询和遍历输出
  */
 boolean ViewElemLinkList(LinkList L);
 boolean FindPostElemLinkList(LinkList L, int post);

 /**
  * 按位置修改LinkList的元素
  */
 boolean UpdateLinkListElem(LinkList *L, int post, ElementType e);

#endif //DEMO08_SETLIST_H
```