实验一 线性表的基本操作

一、实验目的

1．熟悉数据结构上机环境。

2．会定义和使用线性表的顺序存储结构。

3．熟练掌握顺序表的一些基本操作的实现。

二、实验要求

1．认真阅读题目要求和题目中所给的程序代码。

2．按照代码块上面的注释要求补齐代码，注意所给的参数。

3．认真进行程序运行调试，保证正确性。

三、实验内容

按部分代码块上面的红色注释要求完成 C 代码补齐，每做完一个要在 main 函数进行调用和调试，保证正确之后再做下一题。你的代码用红色突出。

```
#include <stdio.h>
#include <malloc.h>
#define MAXSIZE 10 //MAXSISE是顺序表的最大容纳量//顺序表结构定义
typedef struct
{
  int a[10];
  int last;
}SeqList;

//顺序表初始化
SeqList *Init_Seqlist()
{
  SeqList* L = (SeqList*)malloc(sizeof(SeqList));
  L->last = -1;
  return L;
}
```

```
/*注释
    malloc:在内存的动态存储区中分配一个连续空间，函数的返回值是分配区域的起始地址。
            当内存不再使用时，应使用free()函数将内存块释放。
  */

  //顺序表单值插入，在表 L 的 i 位置插入一个值为 e 的元素
  void Ins_SeqList(SeqList *L,int i,int e)
  {
    int j;
    if (i < 1 || i > L->last + 2)
    {
        printf("插入位置非法！\n");
        return;
    }
    /*注释
        i>L->last+2 :i位置指常识意义上的位置，即第1个位置、第二个位置，语言上不存在第0个位置。但在数组层面上第一个位置是a[0]，下标为0。last定义为数组的下标，所以初始化时不能为0，因为数组没有内容，应把last初始化的更往前一位（-1）。当我们想往数组中添加第一个数据时，我们实际上为a[0]赋了值，赋i=1的值后，last应变为0。假若我们跳过i=1给i=2赋值，这应该是非法的。我们先允许它非法赋值，赋值后last变为1。我们可知，顺序表中每次为i=n赋值时，相应的last的值应是n-1。当跨位赋值时，i最小是n+1，而last还是n-1，即非法状态i与last的最小差距为2。故i>L->last+2
    */
    if (L->last >= MAXSIZE - 1)
    {
        printf("顺序表已满！\n");
        return;
    }
    for (j = L->last; j >= i-1; j--)
```

```
    {
        L->a[j + 1] = L->a[j];
    }
    L->a[i - 1] = e;
    L->last++;
    printf("已插入！\n");
    return;
}

//顺序表查找，查找表 L 的中值为 e 的元素位置
void Loc_SeqList(SeqList *L,int e)
{
    for (int i = 0; i <= L->last; i++)
    {
        if (L->a[i] == e)
        {
            printf("已找到！位置为：%d\n",i+1);
            return;
        }
    }
    printf("查找失败！\n");
    return;
}

//顺序表单值删除，在表 L 的 i 位置删除一个元素，x 用于记录该元素的值
void Del_SeqList(SeqList *L,int i,int *x)
{
    int n;//元素下标
    if (i<1 || i>L->last + 1)
    {
```

```
        printf("删除位置非法！\n");
        return;
    }
    if (L->last < 0) {
        printf("顺序表已为空！\n");
        return;
    }
    *x = L->a[i - 1];
    for (n = i - 1; n < L->last; n++)
    {
        L->a[n] = L->a[n + 1];
    }
    L->last--;
    printf("删除成功！\n");
    return;
}

//后面两道题为附加题，选做
//顺序表批量值插入，在表 L 的 i 位置批量插入 n 个元素，所插元素值由键盘输入
void BIns_SeqList(SeqList *L,int i,int n)
{
    int m,j;
    if (i < 1 || i > L->last + 2)
    {
        printf("插入位置非法！\n");
        return;
    }
    if (L->last >= MAXSIZE - 1)
    {
        printf("顺序表已满！\n");
```

```
        return;
    }
    for (m=0; m < n; m++)
    {
        for (j = L->last; j >= i - 1; j--)
        {
            L->a[j + 1] = L->a[j];
        }
        L->last++;
        printf("输入插入元素值：");
        scanf_s("%d", &L->a[i-1]);
        printf("\n");
        printf("已插入！第%d次\n",m+1);
    }
    printf("批量插入已完成！\n");
    return;
}

//顺序表批量删除，在表 L 的 i 位置批量删除 n 个元素
void BDel_SeqList(SeqList *L,int i,int n)
{
    int x;
    if (i<1 || i>L->last + 1)
    {
        printf("删除位置非法！\n");
        return;
    }
    if (L->last < 0) {
        printf("顺序表已为空！\n");
        return;
```

```
    }
    for (x = 0; x < n; x++)
    {
        for (n = i - 1; n < L->last; n++)
        {
            L->a[n] = L->a[n + 1];
        }
        L->last--;
        printf("删除成功！第%d次\n",x+1);
    }
    printf("批量删除已成功！");
    return;
  }

//顺序表显示
void Pri_SeqList(SeqList* L)
{
    int i;
    printf("顺序表为：");
    for (i = 0; i <= L->last; i++)
    {
        printf("%d  ", L->a[i]);
    }
    printf("\n");
    return;
  }

  //主函数
  main()
  {
```

```
    //int i, j;
    int del;
    SeqList* L1;
    L1 = Init_Seqlist();
    Ins_SeqList(L1, 1, 1);
    Pri_SeqList(L1);
    Ins_SeqList(L1, 2, 2);
    Pri_SeqList(L1);
    Ins_SeqList(L1, 3, 3);
    Pri_SeqList(L1);
    Loc_SeqList(L1, 1);
    Del_SeqList(L1, 1, &del);
    Pri_SeqList(L1);
    BIns_SeqList(L1, 1, 5);
    Pri_SeqList(L1);
    BDel_SeqList(L1, 1, 4);
    Pri_SeqList(L1);
    return 0;
}
```