实验 3 顺序表

3.1 实验目的

1. 熟练掌握顺序表的类型定义和基本操作算法（以建立、插入、删除、遍历、排序和归并等操作为重点）的实现。
2. 通过实验加深对 C 语言的使用（特别是函数、数组、结构体和指针）。
3. 掌握模块化程序设计方法。

3.2 预备知识

1. 顺序表的类型定义一（静态存储）

```
typedef  struct
{
   ElemType  a[MAXSIZE];     //一维数组子域
   int  length;              //表长度子域
}SqList;   //顺序表类型
```

2. 顺序表的类型定义二（动态存储）

```
#define LIST_Init_Size  100   //线性表存储空间的初始分配量
#define LISTINCREMENT 10   //线性表存储空间的分配增量
typedef  struct
{
   ElemType    *elem;  //存储区域基地址
   int    length;       //当前有效长度
   int    listsize;     //当前分配的存储容量
} SqList; //顺序表类型
```

3. 顺序表的基本操作

⑴ 初始化顺序表 InitList(&L)

构造一个空的顺序表 L，为表分配存储空间用于存放数据元素。

⑵ 销毁顺序表 DestroyList(&L)

释放顺序表 L 占用的内存空间。

⑶ 判定是否为空表 ListEmpty(L)

返回一个值表示 L 是否为空表。若 L 为空表,则返回 1,否则返回 0。

⑷ 求顺序表的长度 ListLength(L)

返回顺序表 L 的长度。实际上只需返回 length 成员的值即可。

⑸ PriorElem(L, cur_e, &pre_e)

返回给定数据元素的前驱数据元素的值。

⑹ NextElem(L, cur_e, &next_e)

返回给定数据元素的后继数据元素的值。

⑺ 输出顺序表 DispList(L)

当表 L 不为空时，顺序输出 L 中各数据元素的值。

⑻ 求某个数据元素值 GetElem(L,i,&e)

返回 L 中第 i(1≤i≤ListLength(L))个元素的值,存放在 e 中。

⑼ 按元素值查找 LocateElem(L, e)

顺序查找第 1 个值域与 e 相等的数据元素的序号。若这样的元素不存在,则返回值为 0。

⑽ 插入数据元素 ListInsert(&L, i, e)

在顺序表 L 的第 i 个位置(1≤i≤ListLength(L)+1)上插入新的元素 e。

⑾ 删除数据元素 ListDelete(&L, i, &e)

该运算删除顺序表 L 的第 i(1≤i≤ListLength(L))个元素。

⑿ 清空顺序表 ClearList(&L)

删除顺序表 L 中的所有数据元素，但不释放已分配给顺序表的存储空间。

3.3 实例：顺序表的构建及输出

构建一个含有数据元素的顺序表。

参考代码：

```
#include <stdio.h>
#include <stdlib.h>
#define MAXSIZE 20   //数组最大界限


typedef int ElemType;  //数据元素类型
typedef  struct
{
   ElemType  a[MAXSIZE]; //一维数组
   int  length;          //表长
}SqList;   //顺序表类型（采用了定义一）
//基本操作的函数声明
void InitList(SqList &L);
void DispList(SqList L);


//顺序表基本操作的具体实现
void InitList(SqList &L) //建立顺序表
{
   int i;
   printf("\n n=? ");
```

```
    scanf("%d", &L.length);
    for(i=0; i<L.length; i++)
    {
       printf("\n data %d=? ",i);
       scanf("%d", &(L.a[i]));
    }
} //InitList

void DispList(SqList L) //输出顺序表
{
    int i;
    printf("\n 顺序表为:\n");
    if(L.length==0)
       printf("空表\n");
    else
       for(i=0; i<=L.length-1; i++)
          printf("%11d", L.a[i]);
}//DispList



//主函数，测试上述定义的类型及基本操作
void main()
{
    int  k;
    SqList  l;
    printf("\n\n==================================");
    printf("\n   1. 建立顺序表" );
    printf("\n   2. 输出顺序表");
    printf("\n   3. 结束程序运行");
    printf("\n==================================");
    do
    {
       printf("\n 请选择(1-3): ");
       scanf("%d", &k);
       switch(k)
       {
       case 1: InitList(l);  break;
       case 2: DispList(l);  break;
       } //switch
    }while(k!=3);
    printf("\n 再见! \n");
}//main
```

3.4 实验：实现顺序表的基本操作并进行测试

1. 用 C 语言实现顺序表的 ADT（包括顺序表的类型定义和基本操作）,假定数据元素的类型为整型。

2. 设计一个测试应用程序实现如下功能：

⑴建立空顺序表 L；

⑵在上述顺序表 L 中依次插入若干数据元素，如：插入 13,5,27,9,32,123,76,98,54,87；

⑶输出顺序表 L；

⑷输出顺序表 L 的长度；

⑸判断顺序表 L 是否为空；

⑹输出顺序 L 的第 3 个数据元素；

⑺输出指定数据元素（如 76）的位置；

⑻在第 4 个位置上插入一个新数据元素，如：插入 56；

⑼删除顺序表 L 的第 7 个元素；

⑽销毁顺序表 L；

以上为必做题。下面为选做题。

⑾建立两个有序的顺序表 La 和 Lb，并分别向顺序表 La 和 Lb 插入为 m 个和 n 个有序的整数；**（注意：m 和 n 的大小，具体的整数值都由自己确定）**

⑿输出顺序表 La 和 Lb；

⒀将顺序表 La 和 Lb 归并为新的有序表 Lc；

⒁输出顺序表 Lc；

⒂销毁顺序表 La、Lb 和 Lc。

3. 其他要求

将所需要的标准头文件以及一些符号常量的定义等放在 Common.h 头文件中；

顺序表类型定义（采用定义二动态存储）和基本操作函数声明放在 Sqlist.h 头文件中；

基本操作函数的实现放在 Sqlist.cpp 文件中；

测试应用程序放在 SqListTestApp.cpp 文件中。