实验一 顺序表的实现和应用

实验目的：

⑴ 熟悉线性表的定义和基本操作；

⑵ 掌握线性表的顺序存储结构设计与基本操作的实现。

实验任务与要求：

⑴ 定义线性表的顺序存储表示；

⑵ 基于所设计的存储结构实现线性表的基本操作；

⑶ 编写一个主程序对所实现的线性表进行测试；

⑷ 线性表的应用：编程实现顺序表的合并

实验内容：

1、 运行样例程序，理解静态分配顺序存储结构的线性表的下列基本操作。

（1）初始化顺序表　　（2）创建顺序表　　（3）判断空表　　（4）求顺序表长度

（5）输出顺序表　　（6）取顺序表位置 i 的元素值　　（7）在顺序表中查找值为 e 的元素位置

（8）向顺序表中插入一个元素　　（9）从顺序表中删除一个元素

样例程序如下：

```
#include <stdio.h>
#define MaxSize 50
typedef int ElemType;
typedef struct
{    ElemType data[MaxSize];        /*存放顺序表元素*/
     int length;                    /*存放顺序表的长度*/
}SqList;

//初始化顺序表
void InitList(SqList &L)
{
     L.length=0;
}

//创建 n 个元素的顺序表
void CreateList(SqList &L,int n)
{
     int i;
     printf("输入%d 个元素：\n",n);
     for(i=0;i<n;i++)
          scanf("%d",&L.data[i]);
     printf("\n");
     L.length=n;
}

//判断空表
int ListEmpty(SqList L)
{
     return(L.length==0);
}
```

```
//求顺序表长度
int ListLength(SqList L)
{
    return(L.length);
}
//输出顺序表
void DispList(SqList L)
{
    int i;
    if (L.length==0) return;
    for(i=0;i<L.length;i++)
        printf("%d ",L.data[i]);
    printf("\n");
}

//取顺序表位置 i 的元素值
int GetElem(SqList L,int i,ElemType &e)
{
    if (i<1 || i>L.length) return 0;
    e=L.data[i-1];
    return 1;
}

//在顺序表中查找值为 e 的元素位置
int LocateElem(SqList L,ElemType e)
{
    int i=1;
    while(i<=L.length && L.data[i-1]!=e) i++;
    if (i<=L.length)
        return i;
    else
        return 0;
}

//向顺序表中插入一个元素
int ListInsert(SqList &L, int i,ElemType e)
{
    int j;
    if (i<1 || i>L.length+1) return 0;
    for(j=L.length-1;j>=i-1;j--)          /*将 data[i-1]及后面元素后移一个位置*/
        L.data[j+1]=L.data[j];
    L.data[i-1]=e;                        /*插入元素 e*/
    ++L.length;                           /*顺序表长度增 1*/
    return 1;
}
```

```
//从顺序表中删除一个元素
int ListDelete(SqList &L,int i,ElemType &e)
{
    int j;
    if (i<1 || i>L.length) return 0;
    e=L.data[i-1];
    for(j=i-1;j<L.length-1;j++)                    /*将 data[i]及后面元素前移一个位置*/
        L.data[j]=L.data[j+1];
    --L.length;                                    /*顺序表长度减 1*/
    return 1;
}

void main()
{
    ElemType dd,a,b;
    SqList L;
    InitList(L);
    if (ListEmpty(L))
        printf("顺序表为空！ \n");
    printf("创建顺序表！ ");
    CreateList(L,5);
    printf("输出顺序表所有元素！ \n");
    DispList(L);
    printf("输出顺序表长度！ \n");
    printf("ListLength(L)=%d\n",ListLength(L));
    printf("判断顺序表是否为空！ \n");
    printf("ListEmpty(L)=%d\n",ListEmpty(L));
    printf("输出顺序表第 3 个位置元素到 dd！ \n");
    GetElem(L,3,dd);
    printf("dd=%d\n",dd);
    printf("查找元素 a: ");
    scanf("%d",&a);
    printf("元素%d 在顺序表的位置为: %d\n",a,LocateElem(L,a));
    printf("插入元素 b: ");
    scanf("%d",&b);
    printf("在顺序表第 4 个位置插入%d！ \n",b);
    ListInsert(L, 4,b);
    printf("输出插入操作后顺序表所有元素！ \n");
    DispList(L);
    printf("删除顺序表第 3 个位置的元素！ \n");
    ListDelete(L,3,dd);
    printf("输出删除操作后顺序表所有元素！ \n");
    DispList(L);
}
```

2、采用书上第 22 页定义的线性表动态分配顺序存储结构，编程实现书中算法 2.3、算法 2.4 和算法 2.5。
提示：要实现算法 2.4 和 2.5，必须先创建 n 个数据元素的顺序表，另外输出顺序表的操作也是必要的。

3、采用线性表动态分配顺序存储结构，实现顺序表的合并操作：①设有线性表 La 和 Lb，试设计算法将 La 和 Lb 归并为新的线性表 Lc；②设线性表 La 和 Lb 中的数据元素为整数，且均已按值非递减有序排列，要求 Lc 中的数据元素也按值非递减有序排列。