实验 10　经典排序算法的实现

一、实验目标

通过本实验，你将能够在以下几个方面得到锻炼：

1. 用 C 语言熟练编写程序和调试程序；
2. 掌握和运用 C 语言的高级语法特性，包括：结构体、指针、函数指针、typedef、动态内存分配和递归函数，等。
3. 掌握从分析问题到编写伪代码、编写程序、调试程序和测试程序的思路；
4. 能够根据给定的 ADT 规格说明熟练的编写代码；
5. 理解并熟练掌握经典排序算法实现的 C 语言描述；
6. 熟练掌握常用排序算法的实现。

二、ADT规格说明

排序表的 ADT 规格说明请参考教材 264 页（本实验略作改动）。

三、实验要求

用顺序结构的排序表实现基本的排序算法。

四、实验步骤

1. 建立本次实验的文件夹 Lab10。
2. 从Moodle网络课堂下载实验文件Lab10.rar，包括五个代码文件：ElemType.h、ElemType.cpp、SqList.h、SqList.cpp和Lab.cpp。各代码文件的作用如下表所示：

文件	作用
ElemType.h	定义 ElemType 数据类型
ElemType.cpp	实现 ElemType 数据类型的基本操作
SqList.h	排序表的顺序表示 ADT 的定义
SqList.cpp	排序表的顺序表示 ADT 的实现

Lab.cpp	测试主程序

3. ElemType.h 文件定义了 ElemType 类型，ElemType.cpp 文件针对 ElemType 类型实现了 Compare 和 visit 函数，这样排序表的顺序存储结构中就可以存放 ElemType 这种类型的数据了。这两个文件的代码如下。

ElemType.h

```
#ifndef ELEMTYPE_H
#define ELEMTYPE_H

typedef int ElemType;

int  compare(ElemType x, ElemType y);
void visit(ElemType e);

#endif /* ELEMTYPE_H */
```

ElemType.cpp

```
#include <stdio.h>
#include "ElemType.h"

int compare(ElemType x, ElemType y)
{
   return(x-y);
}

void visit(ElemType e)
{
   printf("%d\t", e);
}
```

4. SqList.h 文件是排序表的顺序表示的数据结构存储定义和基本操作的声明。SqList.cpp 是排序表的顺序表示的基本操作的实现。**本次实验的主要内容就是编写排序表的顺序存储形式下经典排序算法的实现代码，也就是完成 SqList.cpp 代码文件的编写。**注意每个操作函数前面都有函数头注释，包括：操作目的、初始条件、操作结果、函数参数和返回值。这些信息对于保证函数代码的逻辑正确性以及对函数进行测试是非常有用的。这两个文件的代码如下。

SqList.h

```
/***
*SqList.h – 排序表的定义
*
****/
```

```h
#ifndef SqList_H
#define SqList_H

#include "ElemType.h"

/*----------------------------------------------------------
// 定义排序表结构
----------------------------------------------------------*/
typedef struct {  // 结点定义
   ElemType   r[10+1];   // 关键字数组，r[0]不用
   int   length;         // 数组中当前的关键字个数
} SqList;

/*----------------------------------------------------------
// 排序表的基本操作
----------------------------------------------------------*/
void InitSqList(SqList *ll);
void QSort(SqList *L,int low,int high);
void HeapSort(SqList *L);
void InsertSort(SqList *ll);
void ShellSort(SqList *ll, int dlta[],int t);
#endif /* SqList_H */
```

SqList.cpp

```cpp
/***
*SqList- 排序表
*
*
*题目：实验10 排序表
*
*班级：
*
*姓名：
*
*学号：
*
****/
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <memory.h>
#include <assert.h>
#include "SqList.h"
/*----------------------------------------------------------
操作目的： 初始化排序表
```

```
初始条件： 无
操作结果： 构造一个符合你输入长度的排序表（排序表的内容就是你输入的各个值）
函数参数：
      SqList *ll  待初始化的SqList *ll
返回值：
      无
-----------------------------------------------------------*/
void InitSqList(SqList *ll){

}
/*-----------------------------------------------------------
操作目的： 对排序表的自low至high部分进行划分
初始条件： 排序表L已经存在,下标low与high值合法
操作结果： 保证比枢轴(pivot)所对应的元素小的元素都放在它的前面,比它大的都放在它后面。我们暂时选择首元素为枢轴
函数参数：
      SqList *L   待划分的排序表
      int low     下标
      int high    上标
返回值：
      bool     操作是否成功

-----------------------------------------------------------*/
int Partion(SqList *L,int low,int high)
{

}

/*-----------------------------------------------------------
操作目的： 对排序表进行快速排序
初始条件： 排序表L已存在
操作结果： 得到自小至大的排序表
函数参数：
      SqList *L   待排序的排序表
      int low     排序表的下界
       int high    排序表的上界
返回值：
      无
-----------------------------------------------------------*/
void QSort(SqList *L,int low,int high)
{

}
/*-----------------------------------------------------------
```

```
操作目的：将排序表调成一个大根堆
初始条件：排序表L已存在
操作结果：构造一个大根堆L
函数参数：
      SqList *L   待调整的排序表
      int s       堆的下界
      int m       堆的上界
返回值：
      无
-------------------------------------------------------------*/
//调整堆
void HeapAdjust(SqList *L,int s,int m){

}
/*-------------------------------------------------------------
操作目的：对排序表进行堆排序
初始条件：排序表L已存在
操作结果：整个排序表有序
函数参数：
      SqList *L   待排序的排序表
返回值：
      无
-------------------------------------------------------------*/
//堆排序
void HeapSort(SqList *L){

}
/*-------------------------------------------------------------
操作目的：对排序表进行直接插入排序
初始条件：排序表ll已存在
操作结果：得到自小至大的排序表
函数参数：
      SqList *ll  待排序的排序表
返回值：
      无
-------------------------------------------------------------*/
//插入排序
void InsertSort(SqList *ll){

}
/*-------------------------------------------------------------
操作目的：对排序表进行增量为dk的希尔排序
初始条件：排序表ll已存在
操作结果：排序表增量有序
函数参数：
```

```
      SqList *ll  待排序的排序表
返回值:
      无
-------------------------------------------------------------*/
//1趟希尔排序
void ShellInsert(SqList *ll,int dk){

}
/*-------------------------------------------------------------
操作目的:  对排序表进行希尔排序
初始条件:  排序表L已存在
操作结果:  得到自小至大的排序表
函数参数:
      SqList *ll  待排序的排序表
      int dlta[]  递减增量数组
       int t      递减增量数组长度
返回值:
      无
-------------------------------------------------------------*/
//希尔排序
void ShellSort(SqList *ll, int dlta[],int t){

}
```

5. Lab.cpp 文件包含程序入口函数 main 用来声明和定义图的对象，并使用图 ADT 提供的操作。在这里主要写测试代码，用来保证图 ADT 实现的正确性。初始代码如下。

Lab.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include "SqList.h"

int main()
{
   // TODO: Place your test code here

   return 0;
}
```

6. 使用 VC++ 6.0 建立一个空的控制台工程，把上面的五个代码文件加入到工程中，进行代码的编写和调试。

五、思考问题

1. 本实验的排序表和教材中的排序表的区别是什么？如果排序表中的元素为结构体怎

么办，这时如何进行关键字的比较。

2. 本实验是扩展性极差的一种排序表，如何让其健壮？
3. 对于待排序的排序表，为什么每个都需要指针作为形式参数？
4. 实验中排序表都采用顺序存储结构，排序表能采用链式结构吗？