**数据结构**

**——实验报告书**



姓名：樊雨晨

学号：201632110111

班级：软件工程(中外)16 班

学院：数理与信息工程学院

日期：2017年05月10日

# **实验十九 快速排序**

## **问题**

对已知的数据进行快速排序

## **实验思路**

一次排序时候先选取一个数作为pivotkey，pivotkey保存在第一位a[0]，然后从后往前寻找比pivotkey大或者小的数，进行交换，然后再从前往后找比pivotkey小或者大的数，进行交换。找到其中的low位置的数，从中间隔断，两遍递归分别进行快速排序。

## **实验步骤**

两个函数，一个是单次排序。一个while条件为low<high，内部嵌套两个while函数，分别改变high和low。然后是Qsort函数，递归将函数断开进行快速排序。

## **代码**

#**include**<iostream>

#**include**<algorithm>

#**include**<cstdio>

#**include**<string.h>

**using** **namespace** std;

#**define** maxsum 20000

**int** **Partition**(**int** L[maxsum], **int** low, **int** high) { // 算法10.6(b)

// 交换顺序表L中子序列L.r[low..high]的记录，使枢轴记录到位，

// 并返回其所在位置，此时，在它之前（后）的记录均不大（小）于它

**int** pivotkey;

L[0] = L[low]; // 用子表的第一个记录作枢轴记录

pivotkey = L[low]; // 枢轴记录关键字

**while** (low<high) { // 从表的两端交替地向中间扫描

**while** (low<high && L[high]>=pivotkey) --high;

L[low] = L[high]; // 将比枢轴记录小的记录移到低端

**while** (low<high && L[low]<=pivotkey) ++low;

L[high] = L[low]; // 将比枢轴记录大的记录移到高端

}

L[low] = L[0]; // 枢轴记录到位

**return** low; // 返回枢轴位置

} // Partition

**void** **QSort**(**int** L[maxsum], **int** low, **int** high) { //算法10.7

// 对顺序表L中的子序列L.r[low..high]进行快速排序

**int** pivotloc;

**if** (low < high) { // 长度大于1

pivotloc = Partition(L, low, high); // 将L.r[low..high]一分为二

QSort(L, low, pivotloc-1); // 对低子表递归排序，pivotloc是枢轴位置

QSort(L, pivotloc+1, high); // 对高子表递归排序

}

} // QSort

**int** **main**(){

**int** n;

**int** L[maxsum];

memset(L,0,**sizeof**(L[maxsum]));

**while**(~scanf("%d",&n)){

**for**(**int** i=1;i<=n;i++){

scanf("%d",&L[i]);

}

QSort(L,1,n);

**for**(**int** i=1;i<=n;i++)

{

printf("%d ",L[i]);

}

}

}

## **测试**

正常

## **结果**



## **总结**

在对数组初始化的过程中，用到了memset函数，但是memset函数在对int类型的数组初始化时候，只能初始化0，不能是其他的数字。因为是按照字节初始化的，一个人int类型是4个字节，所以要特别注意。另外就是不忘记在单次排序后将pivotkey中的值重新赋值到函数中L[low] = L[0]。