南京信息工程大学 数据结构 I 实验(实习)报告

实验(实习)名称 **快速排序** 日期 2024.12. 得分 指导教师

学院 计院 专业 计科

一、实验目的

1、掌握快速排序的基本思想；
2、掌握用快速排序实现对无序序列的排序；
3、掌握快速排序的时间性能。

二、实验内容与步骤

1、针对从键盘输入无序关键字序列：23，5，17，12，26，31，13，4，6，17，采用快速排序完成对该序列由小到大进行排列。

报告要求：

（1）写出快速排序的基本思想和程序设计步骤；

基本思想：快速排序是一种基于分治思想的排序算法。通过选取一个"基准值"（pivot），将序列分为两部分：一部分比基准值小，另一部分比基准值大。然后对这两部分递归进行排序。

程序设计步骤：

1. 选取基准值：从序列中选取第一个元素作为基准值。
2. 划分序列：将序列列为两部分，左边部分的所有元素小于等于基准值，右边部分的所有元素大于基准值。
3. 递归排序：对左边和右边两部分分别递归调用快速排序。
4. 合并结果：递归结束时，这个序列有序。

（2）完整的实现代码（文本）和测试结果（图片）。

实现代码：

```
#include <stdio.h>

#define MAXSIZE 100

typedef struct{
    int key;
}Element;
typedef struct{
    Element r[MAXSIZE+1];   // r[0]用于零时存储
    int length;
}SqList;
// 分区函数
int Partition(SqList *L,int low,int high){
    L->r[0]=L->r[low];
    int pivotkey=L->r[low].key;
    while(low<high){
        // 从右向左扫描，找到第一个小于枢轴的元素
```

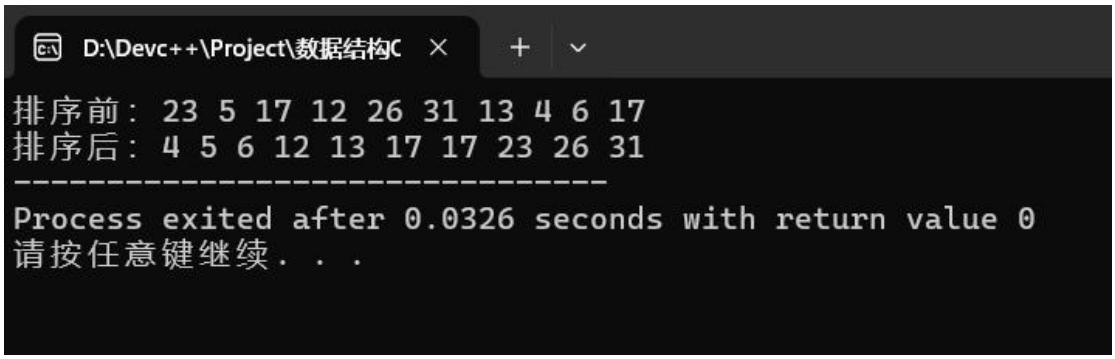
```
            while(low<high&&L->r[high].key>=pivotkey) --high;
            L->r[low]=L->r[high];
            // 从左向右扫描，找到第一个大于枢轴的元素
            while(low<high&&L->r[low].key<=pivotkey) ++low;
            L->r[high]=L->r[low];
        }
        L->r[low]=L->r[0];
        return low;
}
// 快速排序函数
void QSort(SqList *L,int low,int high){
        if(low<high){
            int pivotloc=Partition(L,low,high);
            QSort(L,low,pivotloc-1);
            QSort(L,pivotloc+1,high);
        }
}

void QuickSort(SqList *L){
        QSort(L,1,L->length);
}

int main(){
        int i;
        SqList L={{{0},{23}, {5}, {17}, {12}, {26}, {31}, {13}, {4}, {6},{17}}, 10};
        printf("排序前：");
        for(i=1;i<=L.length;i++){
            printf("%d ",L.r[i].key);
        }
        printf("\n");

        QuickSort(&L);
        printf("排序后：");
        for(i=1;i<=L.length;i++){
            printf("%d ",L.r[i].key);
        }
        return 0;
}
```

测试结果：



三、实验心得（必写）

通过本次实验，我加深了对快速排序算法的理解，特别是分治思想的应用。分治思想十分强大，能通过递归将复杂问题拆分为小问题，快速排序作为其一种应用展示了分治算法的思想。在此次实验中，我深入理解了 C 语言中参数传递的特性，并解决了由于错误使用 C++ 引用传递语法（&）而导致的编译错误问题。C 语言不支持引用传递，只能通过值传递或指针传递来实现函数间的数据修改。因此，我将函数参数修改为指针形式（如 SqList *L），并使用 -> 运算符访问结构体成员，同时在调用时通过传递结构体地址（如 &L）解决了该问题。

最后，快速排序虽然是经典的排序算法，但它的实现过程中仍有许多细节值得深入探索。通过这次实验，我不仅加深了对排序算法的理解，还提升了编写高效、清晰代码的能力。同时，也让我进一步体会到 C++在算法实现中的灵活性和高效性。