

《数据结构课程实践》

**实验报告**

**实验名称：内排序**

**姓 名：杨天诏**

**学 号：20081831**

**专 业：通信工程**

**实验时间：12月5日**

**杭州电子科技大学**

**通信工程学院**

**（实验报告要求：填写三四五一共三部分内容，实验报告文件命名方式：*数据结构课程实践2022\_实验报告10-姓名-学号*，完成后上传电子版到校园网的网络教学平台，无需打印）**

1. **实验目的**

**掌握常用排序算法的原理和实现。**

1. **实验内容**
2. **（实验指导书265页InsertSort算法）对照实验六第二题使用队列排序的算法，学习使用线性表的插入排序。注意输出每一趟的排序结果。**
3. **（实验指导书269页实验10.4）冒泡排序，注意输出每一趟的排序结果。**
4. **（实验指导书270页实验10.5）快速排序算法，注意输出每一趟的排序结果。**
5. **算法设计**

给出2&3两题的程序代码。

题目二：

#pragma once

#include <stdio.h>

#define MAXL 100 //最大长度

typedef int KeyType; //定义关键字类型为int

typedef char InfoType;

typedef struct

{

KeyType key; //关键字项

InfoType data; //其他数据项，类型为InfoType

} RecType; //查找元素的类型

void swap(RecType x, RecType y) //x和y交换

{

RecType tmp = x;

x = y; y = tmp;

}

void CreateList(RecType R[], KeyType keys[], int n) //创建顺序表

{

for (int i = 0; i < n; i++) //R[0..n-1]存放排序记录

R[i].key = keys[i];

}

void DispList(RecType R[], int n) //输出顺序表

{

for (int i = 0; i < n; i++)

printf("%d ", R[i].key);

printf("\n");

}

//----以下运算针对堆排序的程序

void CreateList1(RecType R[], KeyType keys[], int n) //创建顺序表

{

for (int i = 1; i <= n; i++) //R[1..n]存放排序记录

R[i].key = keys[i - 1];

}

void DispList1(RecType R[], int n) //输出顺序表

{

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

printf("%d ", R[i].key);

printf("\n");

}

void BubbleSort(RecType R[], int n)

{

int i, j, k;

RecType tmp;

for (i = 0; i < n - 1; i++)

{

for (j = n - 1; j > i; j--) //比较,找出本趟最小关键字的记录

if (R[j].key < R[j - 1].key)

{

tmp = R[j]; //R[j]与R[j-1]进行交换,将最小关键字记录前移

R[j] = R[j - 1];

R[j - 1] = tmp;

}

printf(" i=%d: ", i);

DispList(R, n);

}

}

int main()

{

int n = 10;

RecType R[MAXL];

KeyType a[] = { 6,8,7,9,0,1,3,2,4,5 };

CreateList(R, a, n);

printf("排序前:"); DispList(R, n);

BubbleSort(R, n);

printf("排序后:"); DispList(R, n);

return 1;

}

题目三：

#pragma once

#include <stdio.h>

#define MAXL 100 //最大长度

typedef int KeyType; //定义关键字类型为int

typedef char InfoType;

typedef struct

{

KeyType key; //关键字项

InfoType data; //其他数据项，类型为InfoType

} RecType; //查找元素的类型

void swap(RecType x, RecType y) //x和y交换

{

RecType tmp = x;

x = y; y = tmp;

}

void CreateList(RecType R[], KeyType keys[], int n) //创建顺序表

{

for (int i = 0; i < n; i++) //R[0..n-1]存放排序记录

R[i].key = keys[i];

}

void DispList(RecType R[], int n) //输出顺序表

{

for (int i = 0; i < n; i++)

printf("%d ", R[i].key);

printf("\n");

}

//----以下运算针对堆排序的程序

void CreateList1(RecType R[], KeyType keys[], int n) //创建顺序表

{

for (int i = 1; i <= n; i++) //R[1..n]存放排序记录

R[i].key = keys[i - 1];

}

void DispList1(RecType R[], int n) //输出顺序表

{

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

printf("%d ", R[i].key);

printf("\n");

}

void disppart(RecType R[], int s, int t)

{

static int i = 1;

int j;

printf("第%d次划分：", i);

for (j = 0; j < s; j++)

printf(" ");

for (j = s; j < t; j++)

printf("%3d", R[j].key);

printf("\n");

i++;

}

int partition(RecType R[], int s, int t) //一趟划分

{

int i = s, j = t;

RecType tmp = R[i]; //以R[i]为基准

while (i < j) //从两端交替向中间扫描,直至i=j为止

{

while (j > i && R[j].key >= tmp.key)

j--; //从右向左扫描,找一个小于tmp.key的R[j]

R[i] = R[j]; //找到这样的R[j],放入R[i]处

while (i < j && R[i].key <= tmp.key)

i++; //从左向右扫描,找一个大于tmp.key的R[i]

R[j] = R[i]; //找到这样的R[i],放入R[j]处

}

R[i] = tmp;

disppart(R, s, t);

return i;

}

void QuickSort(RecType R[], int s, int t) //对R[s..t]的元素进行快速排序

{

int i;

if (s < t) //区间内至少存在两个元素的情况

{

i = partition(R, s, t); //调试用

QuickSort(R, s, i - 1); //对左区间递归排序

QuickSort(R, i + 1, t); //对右区间递归排序

}

}

int main()

{

int i, n = 10;

RecType R[MAXL];

KeyType a[] = { 6,8,7,9,0,1,3,2,4,5 };

CreateList(R, a, n);

printf("排序前:"); DispList(R, n);

QuickSort(R, 0, n - 1);

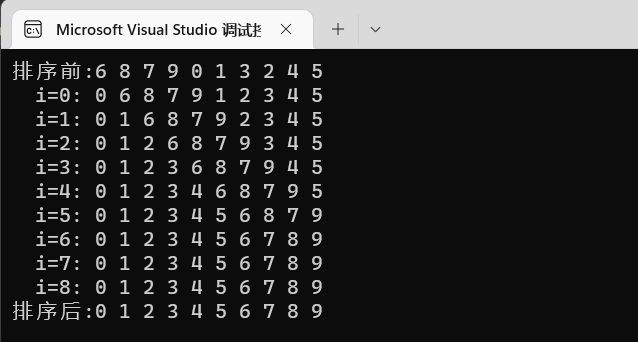
printf("排序后:"); DispList(R, n);

return 1;

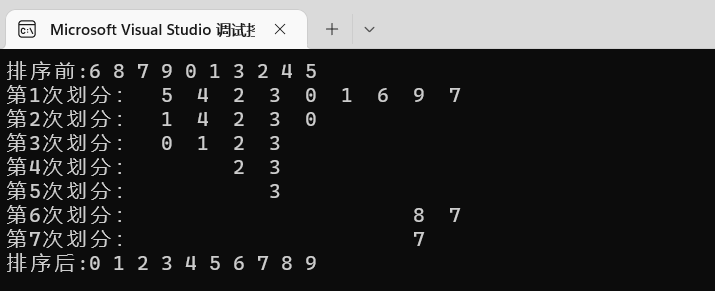
}

1. **运行结果与分析（给出运行结果截图，并加以简要说明）**

**题目二：**

****

**题目三**

**：**

1. **实验小结（记录实验过程中遇到的主要问题和心得）**

**对冒泡排序，队列排序有了进一步的了解。**