**实验报告（第 10 次）**

**实验名称： 各种排序操作的实现**

**实验时间：**

1. **实验目的**

（1）掌握各类排序方法的特点；

（2）掌握各类排序方法的算法；

（3）掌握各类排序方法的效率分析和应用场合。

1. **实验内容**

编程实现：

（1）写一个菜单程序，可选择调用任何一种排序函数。

1. **源程序及主要算法说明**

**#define MAXSIZE 50**

**#include <stdio.h>**

**#include <iostream>**

**#include <ctime>**

**#include <windows.h>**

**#include <cmath>**

**using namespace std;**

**typedef struct**

**{**

**int key;**

**}RECNODE;**

**int b,t; //b为记录交换的次数，t为记录排序的趟数**

**int MakeList(RECNODE \*r)**

**{**

**int j,k;**

**printf("请输入初始数据(每个数据以空格隔开，-1结束）： ");**

**k=0;**

**scanf("%d",&j);**

**while(j!=-1)**

**{**

**k++;**

**r[k].key=j;**

**scanf("%d",&j);**

**}**

**return k;**

**}//输入要排序的一组数据，并且返回数据的个数，以-1为结束标志**

**////////////////////////////////////////////////////////////////////////**

**void UndealoutList(RECNODE \*r,int n)**

**{**

**int i;**

**printf("未排序前的数据 ： ");**

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

**printf(" %d",r[i+1].key);**

**printf("\n");**

**}**

**void DealoutList(RECNODE\*r,int n)**

**{**

**int i;**

**printf("排序后的数据 ： ");**

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

**printf(" %d",r[i+1].key);**

**printf("\n");**

**printf("交换或比较的次数： %d\n",b);**

**printf("排序的趟数： %d\n",t);**

**}**

**////////////////////////////////////////////////////////////////////////**

**void InsertSort(RECNODE\*r,int n)//直接插入排序//**

**/\*实现"直接插入排序"可分三步进行：1．利用 "顺序查找"实现在R[1..i-1]中**

**查找R[i]的插入位置，R[1..j].key ? R[i].key < R[j+1..i-1].key；**

**2．将R[j+1..i-1]中的所有记录均后移一个位置；**

**3．将R[i] 插入(复制)到R[j+1]的位置上。\*/**

**{**

**int i,j;**

**b=0,t=0; //b为记录交换的次数，t为记录排序的趟数**

**for(i=2;i<=n;i++)**

**{**

**r[0]=r[i]; //r[0]的作用是暂存空间**

**j=i-1;**

**while(r[0].key<r[j].key) //如果目标数比源数小**

**{**

**r[j+1]=r[j]; //则让大的数从后往前依次往后挪一个位置，然后让小数插入到恰当位置**

**j--;**

**b++;**

**}**

**r[j+1]=r[0];**

**b++; t++;**

**}**

**}**

**////////////////////////////////////////////////////////////////////////**

**void BubleSort(RECNODE \*r,int n) //冒泡排序**

**{**

**int i,j; b=0,t=0; //b为记录交换的次数，t为记录排序的趟数**

**RECNODE temp; //暂寸空间**

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

**{**

**for(j=n-1;j>=i;j--)//目标数是从第一个数开始，被比较数从最后一个开始**

**if(r[j+1].key<r[j].key) //如果目标数比其下一个数要大**

**{ //则两数交换，下一趟比较则还是用原目标数与其下一个书比较**

**temp=r[j+1];**

**r[j+1]=r[j];**

**r[j]=temp;**

**b++;**

**}**

**else b++; //否则不用交换，继续和下一个数比较**

**t++;**

**}**

**}**

**////////////////////////////////////////////////////////////////////////**

**void BubleSort(double a[]) //时间数组的冒泡排序**

**{**

**int i,j; b=0,t=0;**

**double temp;**

**for(i=1;i<7;i++)**

**{**

**for(j=5;j>=i;j--)**

**if(a[j+1]<a[j])**

**{**

**temp=a[j+1];**

**a[j+1]=a[j];**

**a[j]=temp;**

**b++;**

**}**

**else b++;**

**t++;**

**}**

**}**

**////////////////////////////////////////////////////////////////////////**

**int Partition(RECNODE\*r,int\*low,int\*high)//快速排序//**

**/\*找一个记录，以它的关键字作为"枢轴"，凡其关键字小于枢轴的记录均移动至该记录之前，**

**反之，凡关键字大于枢轴的记录均移动至该记录之后。\*/**

**{**

**int i,j; static int w=0;**

**RECNODE temp;**

**i=\*low; //i指向第一个数**

**j=\*high;**

**temp=r[i]; //暂存单元，存着目标数**

**do**

**{ //当目标数小于等于要比较的数，并且目标数在左边**

**while((r[j].key>=temp.key)&&(i<j))**

**{**

**j--; //则不用换，只要让指向要比较的数的位置左移一位**

**w++;**

**}**

**if(i<j) //若遇到一个在目标数右边的小数**

**{**

**r[i]=r[j]; //则将小数赋给目标数原来所在的位置**

**i++; //要比较的数的位置右移一位**

**w++;**

**}**

**while((r[i].key<=temp.key)&&(i<j))**

**//同理比较，只是现在要比较的数在目标数的右边**

**{**

**i++;**

**w++;**

**}**

**if(i<j)**

**{**

**r[j]=r[i];**

**j--;**

**w++;**

**}**

**}while(i!=j);**

**r[i]=temp;**

**b=w; //b为记录交换的次数**

**return i; }**

**void QuickSort(RECNODE\*r,int start,int end)//快速排序**

**{**

**int i; static int q=0;**

**if(start<end)**

**{**

**i= Partition(r,&start,&end);**

**q++;**

**QuickSort(r,start,i-1); //对低子表递归排序，i为中间位置**

**QuickSort(r,i+1,end); //对高子表递归排序**

**}**

**t=q;//t为记录排序的趟数**

**}**

**////////////////////////////////////////////////////////////////////////**

**double TInsertSort(int len,RECNODE \*a,int p){**

**if(p!=7) {len=MakeList(a); UndealoutList(a,len); }**

**//若为时间效率全比较，则不用把比较的具体过程输出，只需输出排序时间**

**LARGE\_INTEGER m\_liPerfFreq={0};**

**QueryPerformanceFrequency(&m\_liPerfFreq);**

**LARGE\_INTEGER m\_liPerfStart={0};**

**QueryPerformanceCounter(&m\_liPerfStart);**

**InsertSort(a,len);**

**if(p!=7)**

**{DealoutList(a,len);}**

**LARGE\_INTEGER liPerfNow={0};**

**QueryPerformanceCounter(&liPerfNow);**

**double time=liPerfNow.QuadPart - m\_liPerfStart.QuadPart;**

**time/=m\_liPerfFreq.QuadPart;**

**cout<<"直接插入排序时间消耗："<<time<<" MS"<<"\n";**

**return (time);**

**} //直接插入排序时间测试**

**////////////////////////////////////////////////////////////////////////**

**double TBubleSort(int len,RECNODE \*a,int p){**

**if(p!=7)**

**//若为时间效率全比较，则不用把比较的具体过程输出，只需输出排序时间**

**{len=MakeList(a); UndealoutList(a,len); }**

**LARGE\_INTEGER m\_liPerfFreq={0};**

**QueryPerformanceFrequency(&m\_liPerfFreq);**

**LARGE\_INTEGER m\_liPerfStart={0};**

**QueryPerformanceCounter(&m\_liPerfStart);**

**BubleSort(a,len);**

**if(p!=7)**

**{DealoutList(a,len);}**

**LARGE\_INTEGER liPerfNow={0};**

**QueryPerformanceCounter(&liPerfNow);**

**double time=liPerfNow.QuadPart - m\_liPerfStart.QuadPart;**

**time/=m\_liPerfFreq.QuadPart;**

**cout<<"冒泡排序序时间消耗："<<time<<" MS"<<"\n";**

**return (time);**

**} //冒泡排序时间测试**

**////////////////////////////////////////////////////////////////////////**

**double TQuickSort(int len,RECNODE \*a,int p){**

**if(p!=7){len=MakeList(a); UndealoutList(a,len); }**

**//若为时间效率全比较，则不用把比较的具体过程输出，只需输出排序时间**

**LARGE\_INTEGER m\_liPerfFreq={0};**

**QueryPerformanceFrequency(&m\_liPerfFreq);**

**LARGE\_INTEGER m\_liPerfStart={0};**

**QueryPerformanceCounter(&m\_liPerfStart);**

**QuickSort(a,1,len);**

**if(p!=7){DealoutList(a,len); }**

**//若为时间效率全比较，则不用把比较的具体过程输出，只需输出排序时间**

**LARGE\_INTEGER liPerfNow={0};**

**QueryPerformanceCounter(&liPerfNow);**

**double time=liPerfNow.QuadPart - m\_liPerfStart.QuadPart;**

**time/=m\_liPerfFreq.QuadPart;**

**cout<<"快速排序时间消耗："<<time<<" MS"<<"\n";**

**return (time);**

**} //快速排序时间测试**

**int main( )**

**{**

**double TIMES[7],TIMES1[7];**

**RECNODE a[MAXSIZE];**

**int len,p;**

**do**

**{**

**system("cls");//清屏**

**printf("★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆\n");**

**printf("☆★ ☆★\n");**

**printf("★☆\*\*\*\*\*\*\*\*\*\*\*\*\*\* 欢迎来到综合排序系统! \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*★☆\n");**

**printf("☆★ ☆★\n");**

**printf("☆★ ※※※※※※※ ☆★\n");**

**printf("★☆ ※ 菜 单 ※ ★☆\n");**

**printf("☆★ ※※※※※※※ ☆★\n");**

**printf("☆★ ☆★\n");**

**printf("★☆\*\*\*\*\*\*\*\*\*\*\*\*\*\* (1)---直接插入排序 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*★☆\n");**

**printf("☆★\*\*\*\*\*\*\*\*\*\*\*\*\*\* (2)---冒泡排序 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*☆★\n");**

**printf("★☆\*\*\*\*\*\*\*\*\*\*\*\*\*\* (3)---快速排序 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*★☆\n");**

**printf("★☆\*\*\*\*\*\*\*\*\*\*\*\*\*\* (7)---时间效率比较 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*★☆\n");**

**printf("☆★\*\*\*\*\*\*\*\*\*\*\*\*\*\* (0)---退出 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*☆★\n");**

**printf("★☆\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*★☆\n");**

**printf("☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★\n");**

**printf("\n请在上述序号中选择一个并输入： ");**

**scanf("%d",&p);**

**switch(p)**

**{**

**case 1:printf("直接插入排序\n");**

**TInsertSort(len,a,p);**

**break; //直接插入排序**

**case 2:printf("冒泡排序\n");**

**TBubleSort(len,a,p);**

**break; //冒泡排序**

**case 3:printf("快速排序\n");**

**TQuickSort(len,a,p);**

**break; //快速排序**

**break; //希尔排序**

**case 7:printf("时间效率比较\n");**

**{**

**len=MakeList(a);**

**UndealoutList(a,len);**

**TIMES1[1]=TIMES[1]=TInsertSort(len,a,p);**

**TIMES1[2]=TIMES[2]=TBubleSort(len,a,p);**

**TIMES1[3]=TIMES[3]=TQuickSort(len,a,p);**

**BubleSort(TIMES);**

**printf("\n");**

**if(TIMES[1]==TIMES1[1]) printf("排序这组数据用直接插入排序时间最短!\n");**

**if(TIMES[1]==TIMES1[2]) printf("排序这组数据用冒泡排序时间最短!\n");**

**if(TIMES[1]==TIMES1[3]) printf("排序这组数据用快速排序时间最短!\n");**

**if(TIMES[1]==TIMES1[4]) printf("排序这组数据用直接选择排序时间最短!\n");**

**if(TIMES[1]==TIMES1[5]) printf("排序这组数据用堆排序时间最短!\n");**

**if(TIMES[1]==TIMES1[6]) printf("排序这组数据用希尔排序时间最短!\n");**

**break;**

**} //时间效率全比较**

**case 0: break; //退出**

**default:printf("输入错误!请重新输入!\n");**

**break;**

**}**

**system("pause");//系统暂停，提示按任意键继续。。。。**

**}**

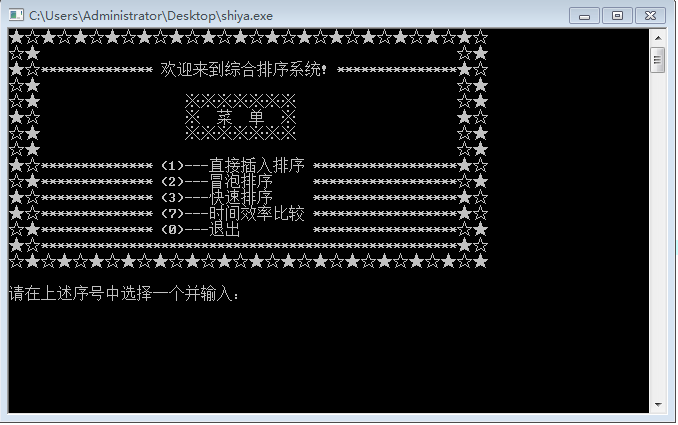
**while(p!=0); //for do**

**}**

1. **主要问题和解决方案**

**还是不太熟悉，要多练多看看**

1. **测试数据及结果**



1. **心得体会与自我评价**

**多练练就熟练了**

1. **教师评分**