**郑 州 轻 工 业 大 学**

**实 验 报 告**

**课程名称：** 数据结构

**实验名称：** 常用的内部排序算法

**院 （系）：** 计算机与通信工程学院

**姓 名：** 原彬贺

**学 号：** 542001020223

**专业班级：** 计算机科学与技术20-02

**指导教师：** 南姣芬 王治国

2021 **年** 6 **月** 3 **日**

**一、实验目的**

1、掌握常见的内部排序算法的思想及其适用条件。

2、掌握常见的内部排序算法的程序实现。

**二、实验内容及要求**

1、任务描述

设计一个内部排序算法模拟系统，利用该系统实现常用的7种排序算法，并测试各种排序算法的性能。

通过一个简单的菜单，分别实现下列排序要求，采用几组不同数据测试各排序算法的性能（比较次数和移动次数）及稳定性。

* 实现简单选择排序、直接插入排序和冒泡排序；
* 实现折半插入排序；
* 实现希尔排序算法；
* 实现快速排序算法（递归和非递归）；
* 实现堆排序算法。

2、主要数据类型与变量

数据类型定义：

#define MAXSIZE 100 /\*参加排序元素的最大个数\*/

typedef int KeyType;

typedef struct {

KeyType key;

InfoType otherinfo; // 其他字段（自行设计）

}RedType;

typedef struct

{

RedType r[MAXSIZE+1];

int

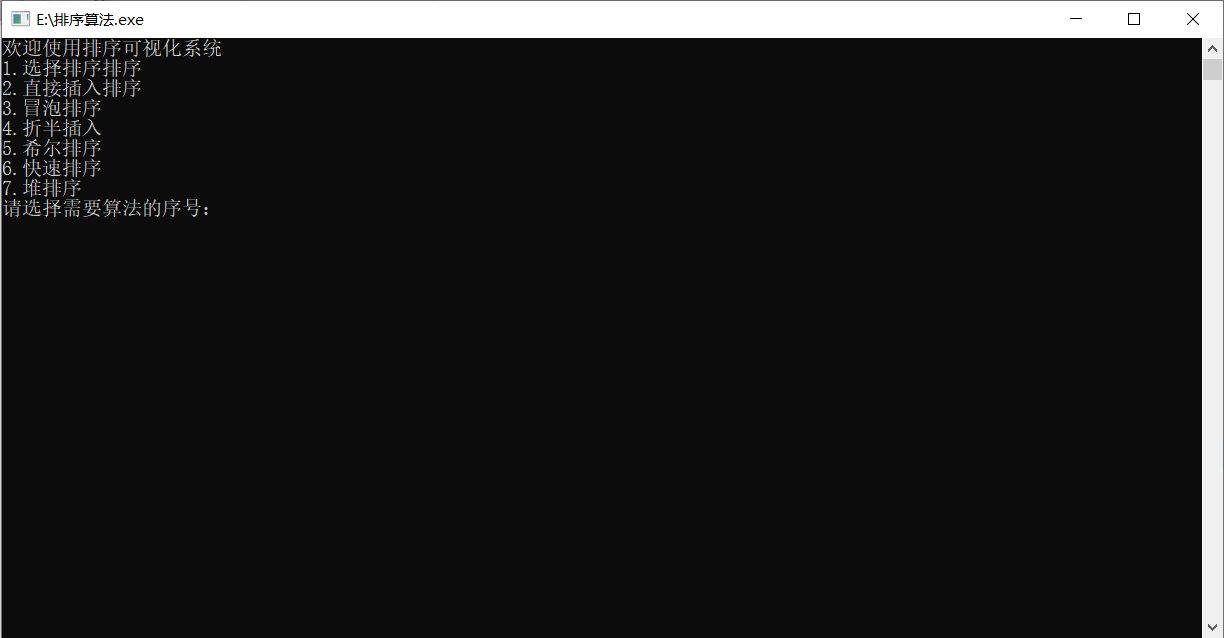
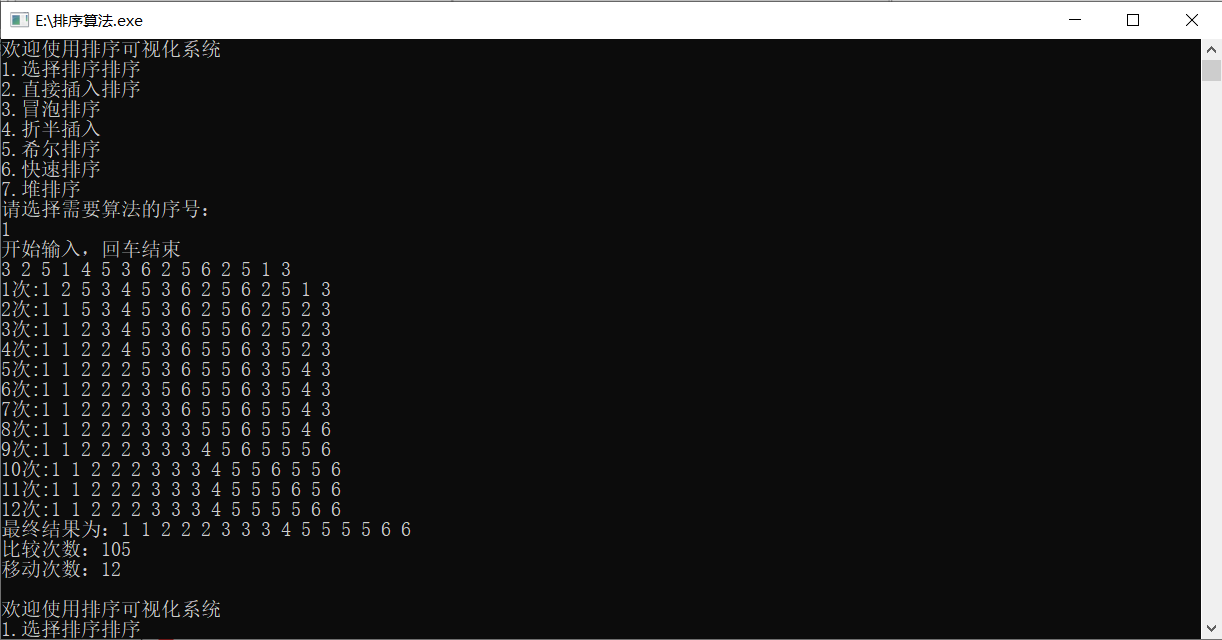
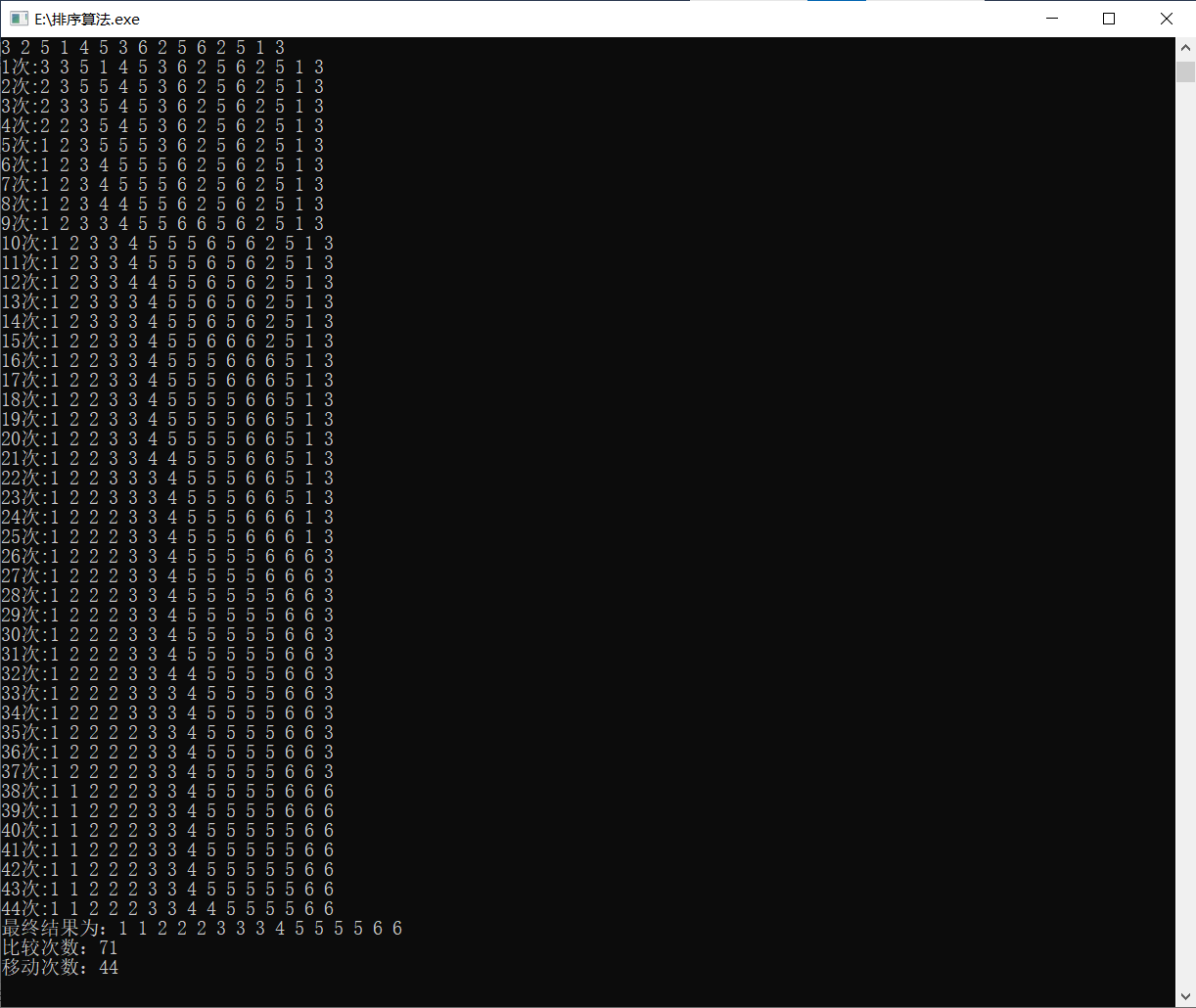
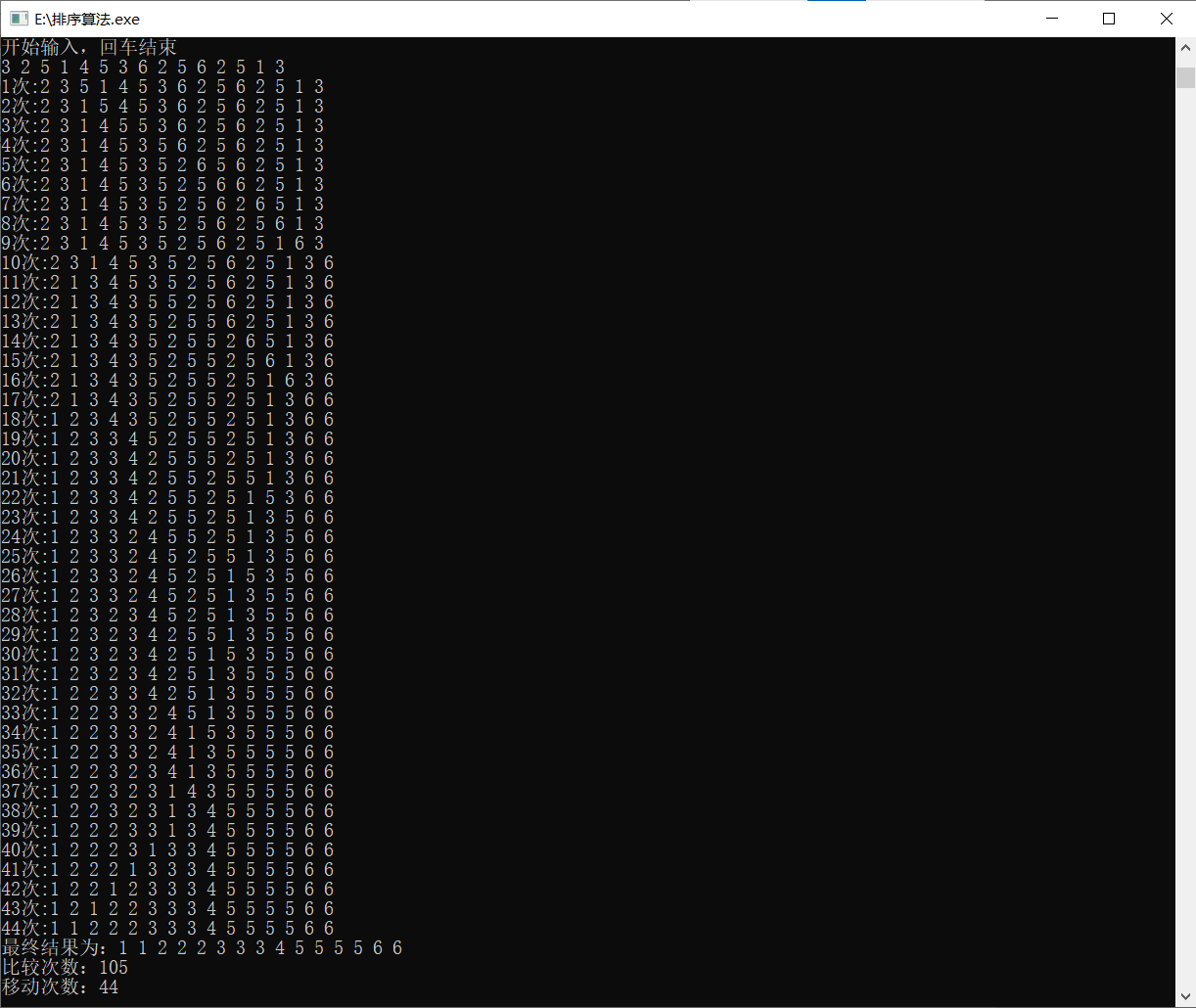
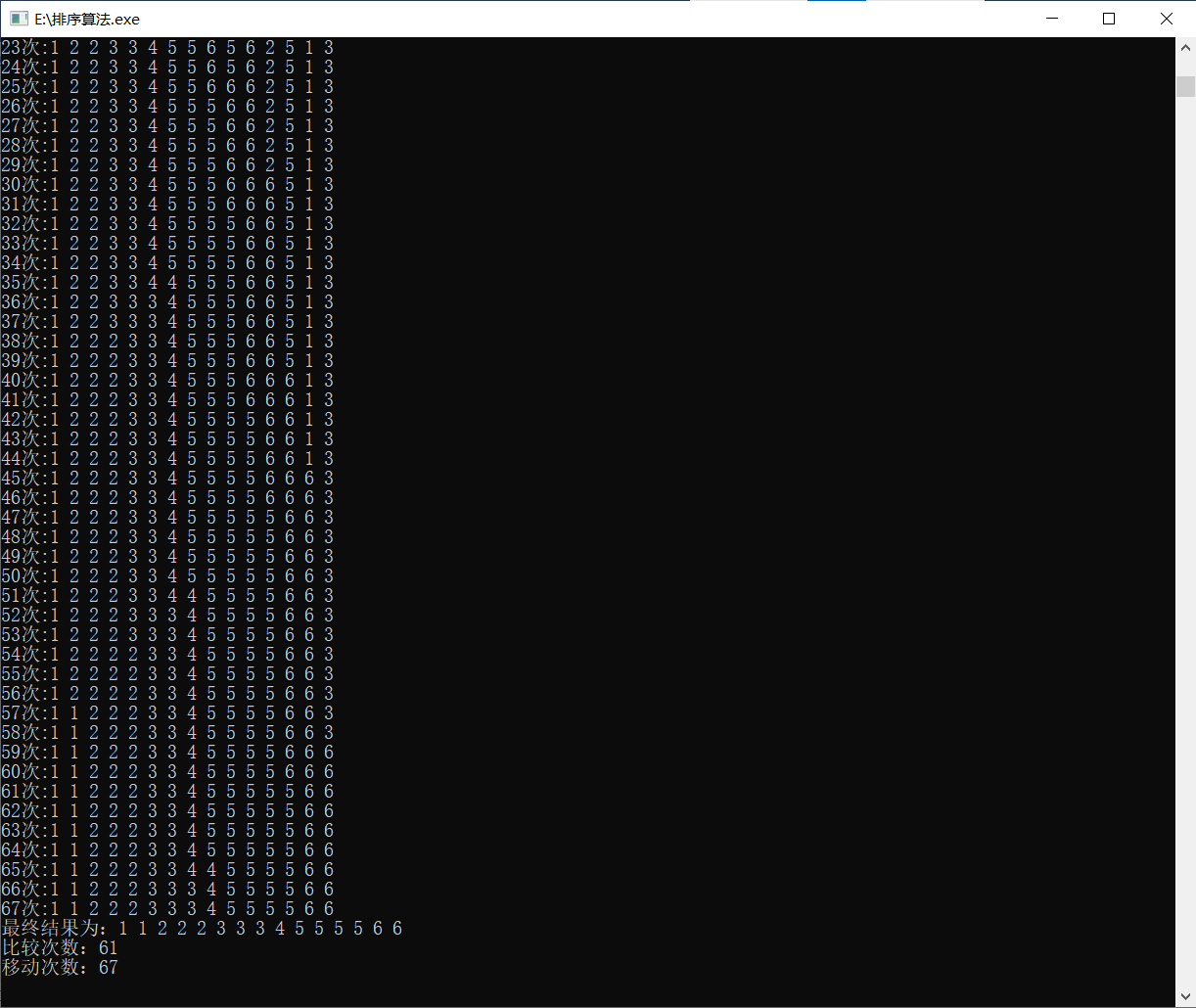
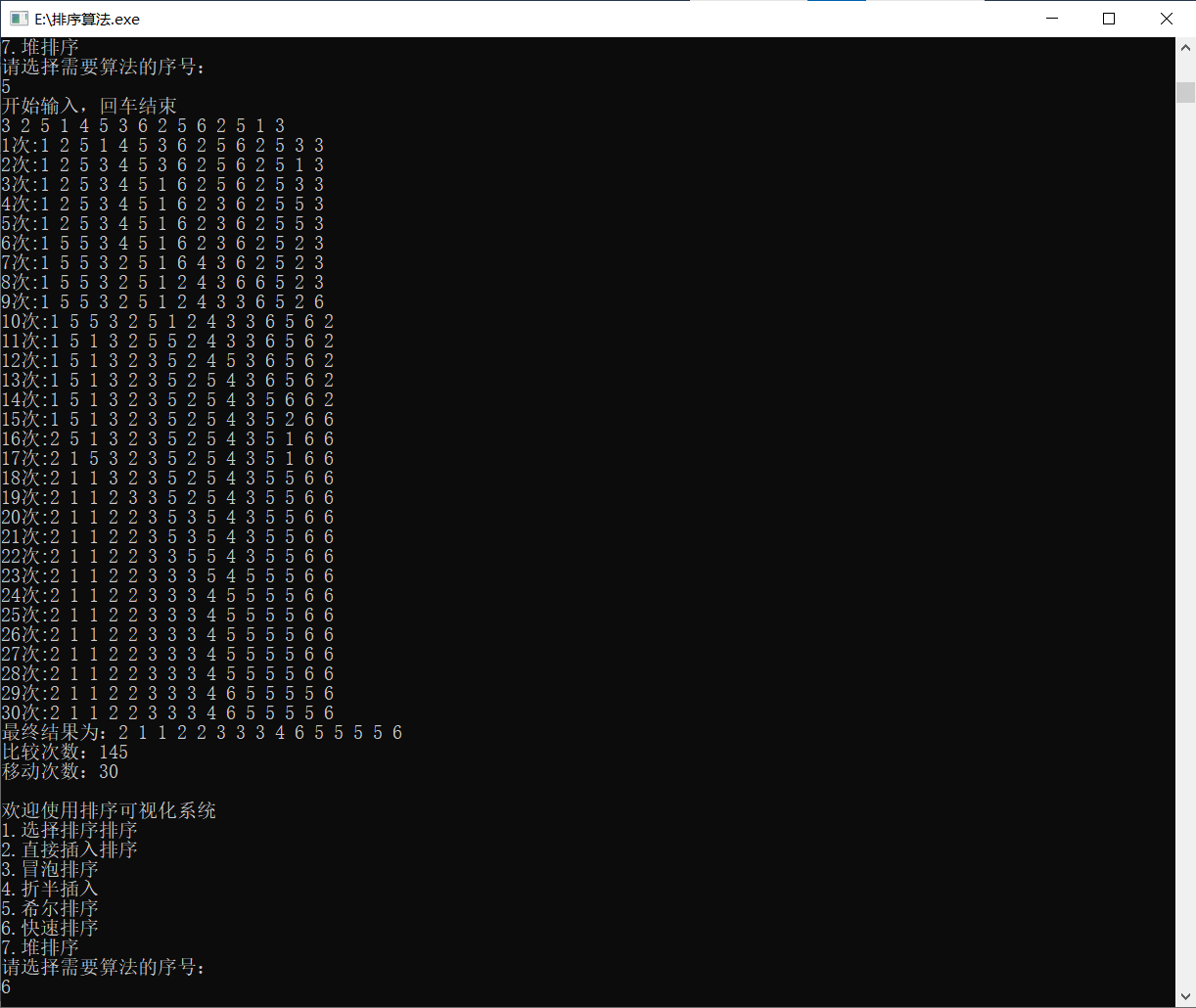
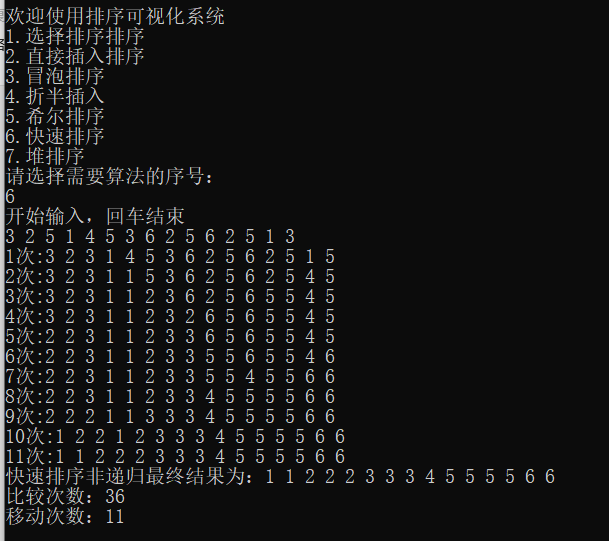
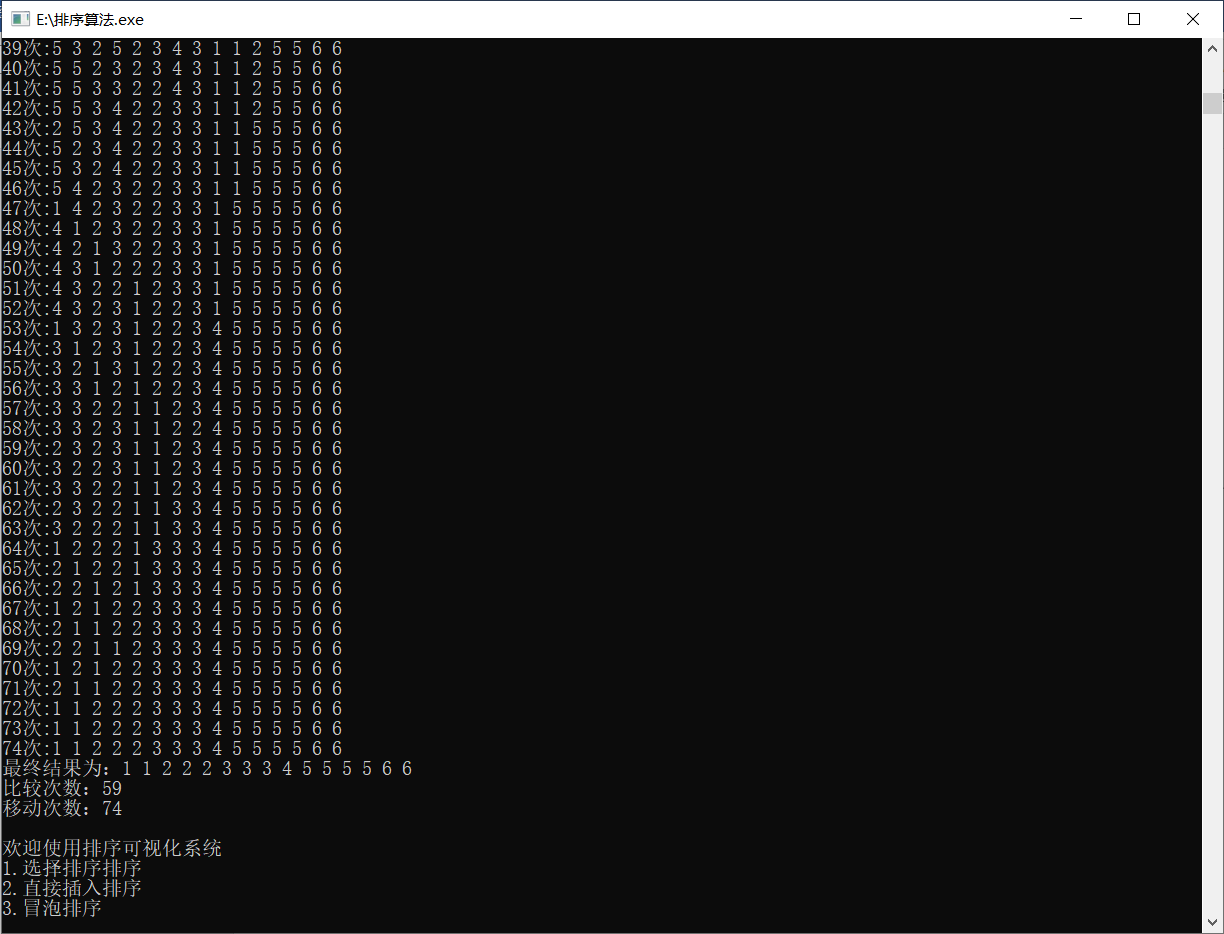
3、算法或程序模块

设计一个内部排序算法模拟系统，利用该系统实现简单选择排序、直接插入排序、冒泡排序、折半插入排序、希尔排序算法、快速排序算法（递归和非递归）、堆排序算法。采用几组不同数据测试各排序算法的性能。

**三、测试**

* 1. 方案

设计一个内部排序算法模拟系统，利用该系统实现简单选择排序、直接插入排序、冒泡排序、折半插入排序、希尔排序算法、快速排序算法（递归和非递归）、堆排序算法。将代码写入程序中，输入相关的数据，并选择合适的算法来观察其结果的优劣。

* 1. 结果
     1. 主界面
        1. 
     2. 简单选择排序
        1. 
     3. 直接插入排序
        1. 
     4. 冒泡排序
        1. 
     5. 折半插入排序
        1. 
     6. 希尔排序
        1. 
     7. 快速排序
        1. 
     8. 堆排序
        1. 

**四、总结与讨论**

通过这次对多种不同的排序算法进行编写实现，在完成后并用实际数据对其进行检测各种算法的优劣，使我不单单掌握了这些算法的具体操作，还使我直观的通过实验结果了解到不同算法适用的情况，比如对于快速排序的确能减少很大的时间，而对于其他算法，在输入规模较小时，性能很不错，但是当输入规模很大时，其执行很慢，这时候希尔排序，快速排序等算法的优点便凸显出来。在这次实验中，我碰到了许多困难，但是通过我对相关算法进行网络寻找资料，然后最终解决了这些困难，这次实验给我的帮助很大。

**附：程序的源代码**

#include <stdio.h>

#include <iostream>

#include <vector>

#include <stack>

#include <math.h>

using namespace std;

void xuanze(vector<int> &data);

void test();

void input(vector<int> &data);

void output(vector<int>& data);

void duipai(vector<int> &data);

void charu(vector<int> &data);

void maopao(vector<int> &data);

void xier(vector<int> &data);

void kuai(vector<int> data);

void kuaisu(vector<int>data);

void kp(int left,int right,vector<int>& data,int &cmp,int&move);

void pai(int pos,int n,vector<int> &data,int& cmp,int &move);

void fin(int low,int hight,int s,vector<int> &data,int &cmp,int &move);

void dui(vector<int>&data,int i,int&cmp,int &move);

void inser(vector<int>&data,int s,int pos,int end,int &cmp,int &move);

void zheban(vector<int> data);

void mane(int& a,vector<int>& );

void swap(int &a,int&b);

int main()

{

test();

return 0;

}

void test()

{

int a;

vector<int> data;

mane(a,data);

while(a)

{

mane(a,data);

}

}

void mane(int& a,vector<int> &data )

{

cout<<"欢迎使用排序可视化系统"<<endl;

cout<<"1.选择排序排序"<<endl;

cout<<"2.直接插入排序"<<endl;

cout<<"3.冒泡排序"<<endl;

cout<<"4.折半插入"<<endl;

cout<<"5.希尔排序"<<endl;

cout<<"6.快速排序"<<endl;

cout<<"7.堆排序"<<endl;

cout<<"请选择需要算法的序号："<<endl;

cin>>a;

if(a==0)

{

return ;

}

input(data);

if(a==1)

{

xuanze(data);

}

else if(a==2)

{

charu(data);

}

else if(a==3)

{

maopao(data);

}

else if(a==4)

{

zheban(data);

}

else if(a==5)

{

xier(data);

}

else if(a==6)

{

kuaisu(data);

kuai(data);

}

else if(a==7)

{

duipai(data);

}

else

{

cout<<"输入错误，请重新输入"<<endl;

}

return ;

}

void input(vector<int>& data)

{

int a;

data.clear();

cout<<"开始输入，回车结束"<<endl;

while(cin>>a)

{

data.push\_back(a);

if(cin.get()=='\n')

{

break;

}

}

return ;

}

void output(vector<int>& data)

{

for(int em:data)

{

cout<<em<<" ";

}

cout<<endl;

return;

}

void xuanze(vector<int> &data)

{

int a,post,cmp=0,move=0;

for(int i=0;i<data.size();i++)

{

a=data[i];

post=i;

for(int j=i+1;j<data.size();j++)

{

cmp++;

if(a>data[j])

{

a=data[j];

post=j;

}

}

if(i!=post)

{

move++;

data[post]=data[i];

data[i]=a;

cout<<move<<"次:";

output(data);

}

}

cout<<"最终结果为：";

output(data);

cout<<"比较次数："<<cmp<<endl ;

cout<<"移动次数："<<move<<endl;

cout<<endl;

}

void duipai(vector<int>& data)

{

int a,post,cmp=0,move=0;

int len=data.size();

for(int i=len;i>=0;i--)

{

dui(data,i,cmp,move);

}

cout<<"最终结果为：";

output(data);

cout<<"比较次数："<<cmp<<endl ;

cout<<"移动次数："<<move<<endl;

cout<<endl;

return ;

}

void dui(vector<int>&data,int i,int&cmp,int &move)

{

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

{

int da=j;

while(data[da]>data[da/2])

{

cmp++;

move++;

swap(data[da],data[da/2]);

da=da/2;

cout<<move<<"次:";

output(data);

}

}

if(i!=0)

{

swap(data[0],data[i-1]);

move++;

cout<<move<<"次:";

output(data);

}

}

void zheban(vector<int> data)

{

int a,post,cmp=0,move=0;

int mid=0,low=0,hight=0;

for(int i=1;i<data.size();i++)

{

fin(low,hight,data[i],data,cmp,move);

hight++;

}

cout<<"最终结果为：";

output(data);

cout<<"比较次数："<<cmp<<endl ;

cout<<"移动次数："<<move<<endl;

cout<<endl;

return;

}

void fin(int low,int hight,int s,vector<int> &data,int &cmp,int &move)

{

int h=hight+1;

int mid;

while(low<hight)

{

mid=(low+hight)/2;

if(data[mid]>s)

{

cmp++;

hight=mid-1;

}

else if(data[mid]<s)

{

cmp=cmp+2;

low=mid+1;

}

else

{

cmp=cmp+3;

inser(data,s,mid,h,cmp,move);

return ;

}

}

if(s<data[low])

{

cmp++;

inser(data,s,low,h,cmp,move);

}

else

{

cmp=cmp+2;

inser(data,s,low+1,h,cmp,move);

}

return ;

}

void inser(vector<int>&data,int s,int pos,int end,int &cmp,int &move)

{

for(int i=end;i>pos;i--)

{

data[i]=data[i-1];

move++;

cout<<move<<"次:";

output(data);

}

data[pos]=s;

move++;

cout<<move<<"次:";

output(data);

return;

}

void charu(vector<int> &data)

{

int a,post,cmp=0,move=0;

for(int i=0;i<data.size();i++)

{

a=data[i];

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

{

cmp++;

if(a<data[j])

{

for(int k=i;k>j;k--)

{

data[k]=data[k-1];

move++;

cout<<move<<"次:";

output(data);

}

data[j]=a;

break;

}

}

}

cout<<"最终结果为：";

output(data);

cout<<"比较次数："<<cmp<<endl ;

cout<<"移动次数："<<move<<endl;

cout<<endl;

return ;

}

void maopao(vector<int>& data)

{

int a,post,cmp=0,move=0;

for(int i=0;i<data.size();i++)

{

a=data[i];

post=i;

for(int j=0;j<data.size()-i-1;j++)

{

cmp++;

if(data[j]>data[j+1])

{

a=data[j];

data[j]=data[j+1];

data[j+1]=a;

move++;

cout<<move<<"次:";

output(data);

}

}

}

cout<<"最终结果为：";

output(data);

cout<<"比较次数："<<cmp<<endl ;

cout<<"移动次数："<<move<<endl;

cout<<endl;

return ;

}

void xier(vector<int>& data)

{

int a,b,post,cmp=0,move=0;

int s=1,an=2;

int ans;

stack <int>nums;

while(s<data.size()/2)

{

nums.push(s);

an=an\*2;

s=an-1;

}

while(!nums.empty())

{

ans=nums.top();

nums.pop();

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

{

pai(i,ans,data,cmp,move);

}

}

cout<<"最终结果为：";

output(data);

cout<<"比较次数："<<cmp<<endl ;

cout<<"移动次数："<<move<<endl;

cout<<endl;

return ;

}

void pai(int pos,int n,vector<int> &data,int& cmp,int &move)

{

int min,post;

for(int i=pos;i<data.size();i=i+n)

{

min=data[i];

for(int j=i+1;j<data.size();j=j+n)

{

cmp++;

if(min>data[j])

{

min=data[j];

post=j;

}

}

swap(data[i],data[post]);

move++;

cout<<move<<"次:";

output(data);

}

}

void kuaisu(vector<int> data)

{

int a,b,post,cmp=0,move=0,right,left;

int i=0,j=data.size()-1;

stack <int> nums;

nums.push(0);

nums.push(data.size()-1);

while(!nums.empty())

{

int r,l;

right=nums.top();

nums.pop();

left=nums.top();

nums.pop();

r=right;

l=left;

a=data[left];

if(right>left)

{

left++;

while(right>left)

{

cmp++;

cmp++;

if(data[left]<=a)

{

left++;

}

if(data[right]>a)

{

right--;

}

if(data[right]<=a&&data[left]>a)

{

move++;

swap(data[right],data[left]);

cout<<move<<"次:";

output(data);

right--;

left++;

}

}

if(a>data[right])

{

move++;

swap(data[l],data[right]);

cout<<move<<"次:";

output(data);

}

nums.push(l);

nums.push(right-1);

nums.push(right+1);

nums.push(r);

}

}

cout<<"快速排序非递归最终结果为：";

output(data);

cout<<"比较次数："<<cmp<<endl ;

cout<<"移动次数："<<move<<endl;

cout<<endl;

return ;

}

void kuai(vector<int> data)

{

int a,post,cmp=0,move=0;

int left=0,right=data.size()-1;

kp(left,right,data,cmp,move);

cout<<"快速排序递归最终结果为：";

output(data);

cout<<"比较次数："<<cmp<<endl ;

cout<<"移动次数："<<move<<endl;

cout<<endl;

}

void kp(int left,int right,vector<int>& data,int &cmp,int&move)

{

if(left<right)

{

int a=data[left];

int l=left;

int r=right;

left++;

while(right>left)

{

cmp++;

cmp++;

if(data[left]<=a)

{

left++;

}

if(data[right]>a)

{

right--;

}

if(data[right]<=a&&data[left]>a)

{

move++;

swap(data[right],data[left]);

cout<<move<<"次:";

output(data);

right--;

left++;

}

}

if(a>data[right])

{

move++;

swap(data[l],data[right]);

cout<<move<<"次:";

output(data);

}

kp(l,right-1,data,cmp,move);

kp(right+1,r,data,cmp,move);

}

}

void swap(int &a,int&b)

{

int c;

c=a;

a=b;

b=c;

return ;

}