山东大学 计算机科学与技术 学院

数据结构与算法 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201700130009 | 姓名： 张愈博 | | 班级： 计科3 |
| 实验题目：排序算法 | | | |
| 实验学时：2 | | 实验日期： 2018.10.09 | |
| 实验目的：  掌握各种简单排序算法 | | | |
| 软件环境：  Windows 10  Dev c++ 5.11  Visual studio 2017 | | | |
| 1. 实验内容（题目内容，输入要求，输出要求）   1、创建排序类：数据含有 n 个整数，使用动态数组存储；提供操作：按名次  排序、及时终止的选择排序、及时终止的冒泡排序、插入排序；  2、键盘输入 n, 随机生成 n 个 0~1000 之间的整数建立排序实例；输出各种  排序算法的排序过程。  3、统计每一种排序所耗费的时间（即比较次数和移动次数）。   1. 数据结构与算法描述 （整体思路描述，所需要的数据结构与算法）   首先创建一个排序类，成员变量有size, array[ ]，将4中排序方法作为成员函数编入。  按名次排序算法描述：先两两比较数组中所有的元素对，让较大的元素名次++，然后根据名次大小来重新确定原数组中元素的次序。  及时终止的选择排序算法描述：把该数组分为有序段（在数组后端）和无序段（在数组前段），遍历无序段数组找出最大元素，把它和数组无序段末位元素交换，无序段长度-1。为了使循环及时终止，我们加入判断条件，当数组已经有序的时候就提前跳出循环。  及时终止的冒泡排序算法描述：从头将数组中相邻元素比较，若左边的元素大于右边的，则进行一次交换。一次冒泡结束后，最大的元素一定在最右端。那么我们进行多次冒泡过程，直到整个数组有序。为了使循环及时终止，我们加入判断条件，当数组已经有序的时候就提前跳出循环。  插入排序算法描述：把该数组分为有序段（在数组前段）和无序段（在数组后端），取无序段的第一个元素，将其插入有序段元素，使有序段仍然有序，长度+1。  输出排序过程合同及比较和移动次数较为简单，只需在排序算法过程中添加相应的代码即可。   1. 测试结果（测试输入，测试输出，结果分析）   C:\Users\张愈博\AppData\Roaming\Tencent\Users\1390994379\QQ\WinTemp\RichOle\U}520UOCOG%KYWCVWQS[FH5.png  图1 图2    图3  在测试中，指定元素个数为10，经分析输出正常，结果正常。   1. 分析与探讨（结果分析，若存在问题，探讨解决问题的途径）   在实验过程中，出现了不少错误，但是并未保存错误的实验代码，下面我指出一些过程中的错误。  1.在计算名次部分的代码中，开始时我使i=[1:size-1],j=[i:size-1]，这样写有一个小bug，i与j存在相等的情况，导致r[i]最小值是1非0，在后面赋值过程中漏掉了temp[0]，从而使结果出错。  2.在及时终止的选择排序，开始时我用了while和2个for语句组成三重循环来判断，在调试过程中我发现这种结构存在很大的缺陷，十分复杂且难以调试。还是书本上的双重循环加两个判断条件最好，简明且易调试。在以后的实验中需要多多学习书本上的代码风格和规范。   1. 附录：实现源代码（本实验的全部源程序代码，程序风格清晰易理解，有充分的注释）   #include <iostream>  #include <string.h>  #include <time.h>  #include<stdlib.h>  using namespace std;  void swap(int &a,int &b,int \*mov\_step)  {  int t=a;  a=b;  b=t;  (\*mov\_step)++;  }  class Sort  {  int size,\*array;  public:  Sort(int \*temp,int n) //构造函数  {  if(n<=0) cout<<"logic error!"<<endl;  else  {  size=n;  array=new int [n];  for(int i=0;i<n;i++)  {  array[i]=temp[i];  }  }  }  Sort() { } //默认构造函数  Sort(const Sort &X)  {  size=X.size;  array=new int [size];  for(int i=0;i<size;i++)  {  array[i]=X.array[i];  }  }  ~Sort() //析构函数释放内存  {  delete [] array;  }    Sort& operator =(const Sort S) //重载=给Sort赋值  {  array=new int [S.size];  size=S.size;  for(int i=0;i<S.size;i++)  {  array[i]=S.array[i];  }  return \*this;  }    void rank\_sort(int r[],int \*mov\_step,int \*comp\_step)  {//给数组a[0:n-1]的n个元素按名次排序  //下面是计算名次部分  \*mov\_step=0;  \*comp\_step=0;  memset(r,0,size\*sizeof(r[0]));  for(int i=1;i<size;i++) //这里比较所有元素对  {  for(int j=0;j<i;j++) //这样写会避免i和j相等的情况，从而让r[i]最小值是0非1  {  (\*comp\_step)++;  if(array[i]>array[j]) r[i]++;  else r[j]++;  }  } cout<<"这是array中元素按顺序的名次：";  for(int i=0;i<size;i++) cout<<r[i]<<" "; cout<<endl;  int \*temp=new int [size];  for(int i=0;i<size;i++) //根据名次在临时数组中排序  { temp[r[i]]=array[i]; (\*mov\_step)++; }  for(int i=0;i<size;i++)  { array[i]=temp[i]; (\*mov\_step)++; }    delete [] temp;  }    void select\_sort(int \*mov\_step,int \*comp\_step)  {  \*mov\_step=0;  \*comp\_step=0;  bool sorted=false; //用于查看数组是否已经有序（从小到大）  int i,j;  //这里试来试去三重循环还是不合理，还是用书本上的双重循环加两个判断条件最好  for(i=0;i<size&&sorted==false;i++) //只有当数组无序时才继续执行  { int max=0; \*comp\_step+=2;  sorted=true;  for(j=0;j<size-i;j++)  { (\*comp\_step)++;  if(array[max]<=array[j]) max=j; //从第一个元素开始向后搜索，若发现其后的元素都是递增的，那么有序  else sorted=false; //如果发现其后的一个元素比a[max]小，说明数组仍然无序（注意我这里max的元素是要放在数组最后的）  }  swap(array[max],array[size-1-i],mov\_step);  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ cout<<"第"<<i<<"次排序后的结果是：" ; output(); cout<<" "<<sorted<<endl; /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  }      }    void bubble\_sort(int \*mov\_step,int \*comp\_step)  {//从小到大  \*mov\_step=0;  \*comp\_step=0;  int i,j;  bool sorted=false;  int count=0; //计数器  for(j=0;j<size&&sorted==false;j++)  {  sorted=true;  for(i=0;i<size-1;i++)  {  (\*comp\_step)++;  if(array[i]>array[i+1]) { swap(array[i],array[i+1]); (\*mov\_step)++; sorted=false; }  }  cout<<"第"<<++count<<"次排序后的结果是："; output(); cout<<" "<<sorted<<endl;  }  }    void insert\_sort(int \*mov\_step,int \*comp\_step)  {  \*mov\_step=0;  \*comp\_step=0;  int i,j,temp,count=0;  for(i=1;i<size;i++)  {  temp=array[i];  for(j=i-1;j>=0;j--)  {  (\*comp\_step)++;  if(array[j]>temp) { array[j+1]=array[j]; (\*mov\_step)++; count++; } //当temp前面的数比它大，那么把它们后移  if(array[j]<=temp) break; //否则把temp插入，跳出循环  }  array[j+1]=temp; (\*mov\_step)++;  cout<<"第"<<i<<"次排序后的结果是："; output(); cout<<" "<<endl;  }  output();  }    void output() const  {  for(int i=0;i<size;i++)  cout<<array[i]<<" ";  cout<<endl;  }  };  int main()  {  int n,\*p,mov\_step=0,comp\_step=0;  cout<<"请输入元素的数量n"<<endl; cin>>n;  srand((unsigned)time(0)); //这个随机数种子1秒会改变一次，利用了系统时钟  p=new int[n];  for(int i=0;i<n;i++)  {  p[i]=(rand()%1001);  }  Sort A(p,n),B(p,n),C(p,n),D(p,n); //对类对象初始化  cout<<"The elements are:"; A.output();    cout<<"即时终止的选择排序："<<endl;  A.select\_sort(&mov\_step,&comp\_step); A.output();  cout<<"比较次数为："<<comp\_step<<"\t"<<"移动次数为："<<mov\_step<<endl;    cout<<"按名次排序 ："<<endl;  int \*ref=new int[n]; B.rank\_sort(ref,&mov\_step,&comp\_step); B.output();  cout<<"比较次数为："<<comp\_step<<"\t"<<"移动次数为："<<mov\_step<<endl;    cout<<"及时终止的冒泡排序排序 ："<<endl;  C.bubble\_sort(&mov\_step,&comp\_step); C.output();  cout<<"比较次数为："<<comp\_step<<"\t"<<"移动次数为："<<mov\_step<<endl;    cout<<"插入排序 ："<<endl;  D.insert\_sort(&mov\_step,&comp\_step); D.output();  cout<<"比较次数为："<<comp\_step<<"\t"<<"移动次数为："<<mov\_step<<endl;    return 0;  } | | | |