|  |  |  |  |
| --- | --- | --- | --- |
| **学生学号** | 175064 | **实验课成绩** |  |

*河北工业大学*

**学 生 实 验 报 告 书**

|  |  |
| --- | --- |
| **实验课程名称** | 数据结构 |
| **开 课 学 院** | 国际教育学院 |
| **指导教师姓名** | 宛艳萍 |
| **学 生 姓 名** | 张心宇 |
| **学生专业班级** | 中法计172 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2018 | -- | 2019 | 学年 | 第 | 二 | 学期 |

# 实验内容

【**问题描述**】

各种内部排序算法的时间复杂度分析结果只给出了算法执行时间的阶，或大概执行时间。试通过随机的数据比较各算法的关键字比较次数和关键字移动次数，以取得直观感受

【基本要求】

（1） 对以下6种常用的内部排序算法进行比较：起泡排序、直接插入排序、简单选择排序、快速排序、希尔排序、堆排序。

（2） 待排序的表长不小于100；其中的数据要用伪随机数产生程序产生；至少要用5组不同的输入数据作比较；比较的指标为有关键字参加的比较次数和关键字的移动次数（关键字交换计为3次移动）。

（3） 最后要对结果作出简单分析，包括对各组数据得出结果波动大小的解释。

【测试数据】

由随机数产生器生成。

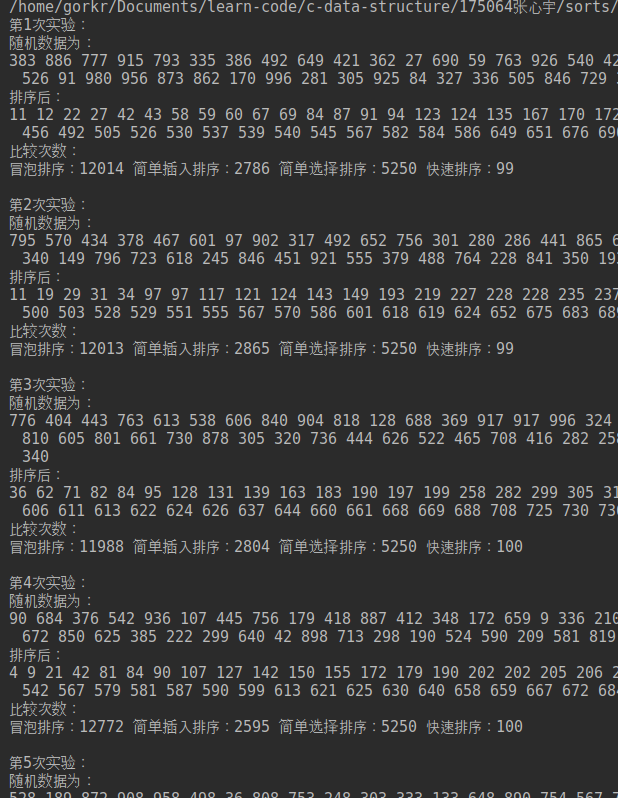
# 实验目的

掌握多种排序方法的基本思想，如直接插入、起泡、简单选择、快速、堆、希尔排序等排序方法，并能够用高级语言实现。

# 代码清单

// main.cpp  
  
#include <iostream>  
#include <ctime>  
#include <cstdlib>  
  
const size\_t LEN = 100;  
  
  
  
template <typename T>  
int bubbleSort(T data[], size\_t len){  
 int num=0;  
 auto sort\_len = len-1; // 一趟排序中最后应该比较的下标  
 while(sort\_len>0){  
 size\_t last\_exchange = 0;  
 for(size\_t j=0; j<sort\_len; j++){  
 num++;  
 if(data[j]<data[j+1]){  
 std::swap(data[j], data[j+1]);  
 last\_exchange = j;  
 num+=3;  
 }  
 }  
 sort\_len = last\_exchange;  
 }  
 return num;  
}  
  
  
template <typename T>  
int insertionSort(T data[], size\_t len){  
 int j=0, num=0;  
 auto sort\_len = len-1;  
 T key;  
 for(size\_t i=1; i<=sort\_len; ++i){  
 num++;  
 if(data[i]<data[i-1]){  
 key = data[i]; num++;  
 data[i] = data[i-1];  
 for(j=i-1;key<data[j]&&j>=0;j--){  
 data[j+1] = data[j];  
 num++;  
 }  
 data[j+1] = key; num++;  
 }  
 }  
 return num;  
}  
  
  
template <typename T>  
int simpleSelectionSort(T data[], int len){  
 auto num = 0;  
 for(auto i=0; i<len; ++i){  
 auto min = i;  
 for(auto j=i+1; j<len; ++j){  
 num++;  
 if(data[j]<data[min])min=j;  
 }  
 std::swap(data[i],data[min]);  
 num+=3;  
 }  
 return num;  
}  
  
template <typename T>  
int quickSort(T data[], int low, int high){  
 int static num=0;  
 if(low<high){  
 auto pivot\_key=0;  
 auto l=low, h=high;  
 pivot\_key = data[l];  
 while(l<h){  
 while(l<h&&data[h]>=pivot\_key){  
 h--;  
 num++;  
 } // 找到第一个小于pivot\_key的下标  
 data[l]=data[h];  
 while(l<h&&data[l]<=pivot\_key){  
 l++;  
 num++;  
 }  
 data[h]=data[l];  
 }  
 data[l] = pivot\_key;  
 quickSort(data, low, l-1);  
 quickSort(data, l+1, high);  
 }  
}  
  
  
void testSorts(){  
 // 产生随机数据  
  
 int data[LEN] , bubble\_data[LEN], insertion\_data[LEN], selection\_data[LEN] ,  
 quick\_data[LEN];  
 std::cout<<"随机数据为： "<<std::endl;  
 for(size\_t i=0;i<LEN;i++){  
 data[i] = bubble\_data[i] = insertion\_data[i] = selection\_data[i] =  
 quick\_data[i] = rand()%1000;  
 std::cout<<data[i]<<' ';  
 }  
 std::cout<<std::endl;  
  
 auto bubble\_num = bubbleSort(bubble\_data, LEN);  
 auto insertion\_num = insertionSort(insertion\_data, LEN);  
 auto selection\_num = simpleSelectionSort(selection\_data, LEN);  
 auto quick\_num = quickSort(quick\_data, 0, LEN-1);  
  
 std::cout<<"排序后： "<<std::endl;  
 for(size\_t i=0;i<LEN;i++){  
 //std::cout<<bubble\_data[i]<<' ';  
 //std::cout<<insertion\_data[i]<<' ';  
 //std::cout<<selection\_data[i]<<' ';  
 std::cout<<quick\_data[i]<<' ';  
  
 }  
  
 std::cout<<"比较次数：\n";  
 std::cout<<"冒泡排序："<<bubble\_num<<' '<<"简单插入排序："<<insertion\_num<<' '<<"简单选择排序："<<selection\_num<<' '<<"快速排序："<<quick\_num<<' '<<std::endl;  
 std::cout<<'\n';  
  
}  
  
  
int main() {  
 for(auto i=0; i<5; i++){  
 std::cout<<"第"<<i+1<<"次实验：\n";  
 testSorts();  
 }  
  
 return 0;  
}

# 运行结果



# 分析与思考

根据实际情况选用合适的排序算法。