八种排序算法比较

1452764　何冬怡

目录

项目简介

随机函数产生 10000 个随机数，用快速排序，直接插入排序，冒泡排序，选择排序的排序方法排序，并统计每种排序所花费的排序时间和交换次数。其中，随机数的个数由用户定义，系统产生随机数，并且显示他们的比较次数，排序算法包括冒泡排序，选择排序，直接插入排序，希尔排序，快速排序，堆排序，归并排序和基排序。

使用手册

1. 打开 exe 文件后，出现用户界面，可通过输入对应数字进入不同的工作模式。

2. 输入 1 建立随机生成数据列表，输入想要生成的数字量。

```
==========================================
       1  -  Generate Datalist
       2  -  Bubble Sort
       3  -  Insert Sort
       4  -  Select Sort
       5  -  Shell Sort
       6  -  Quick Sort
       7  -  Heap Sort
       8  -  Merge Sort
       9  -  Radix Sort
       10 -  Reset Datalist
       11 -  Show Origin Datalist
       12 -  Quit
==========================================

please input your operation: 1
Please input the size of the datalist:1000
------------------------------------------
```

3. 输入 2 输出冒泡排序结果以及排序性能

```
Exchange Times:  252094
Sort Time:       0.015
Compare Times:   961038

------------------------------------------
please input your operation:
```

4. 输入 3 输出直接插入排序结果及排序性能
5. 输入 4 输出选择排序结果及排序性能。
6. 输入 5 输出希尔排序结果及排序性能。
7. 输入 6 输出快速排序结果及排序性能。
8. 输入 7 输出堆排序结果及排序性能。
9. 输入 8 输出归并排序结果及排序性能。
10. 输入 9 输出直接插入排序结果及排序性能。
11. 输入 10 重置数据表格。

```
please input your operation: 10
Please input the size of the datalist:10000
----------------------------------------
please input your operation:
```

12. 输入 11 输出未排序的原数据表。
13. 输入 12 退出系统。

程序概述

1. 数据结构

使用了一个 DataList 类存放数据表，将各类排序封装为成员函数。

2. 算法思路

利用 time.h 库函数进行排序计时，random 库生成随机数。

分别使用冒泡排序、选择排序、直接插入排序、希尔排序、快速排序、堆排序、归并排序和基数排序经典算法进行排序。

3. 文件目录

可执行文件10_1452764_hedongyi.exe	类定义声明10_1452764_hedongyi.h
主文件 10_1452764_hedongyi.cpp	项目文档 10_1452764_hedongyi.pdf

成员变量/函数接口

成员变量名	数据类型	功能说明	
datalist	vector<int>*	数据列表	
num	int	数据量	
sortTime	double	排序时长	
exchangeTimes	int	交换次数	
compareTimes	int	比较次数	
成员函数名	**功能**	**参数**	**返回值**
DataList(int num)	构造要求数据量的数据表	int 数据量	DataList
bubbleSort()	冒泡排序	空	int*排序结果
selectSort()	选择排序	空	int*排序结果
insertSort()	直接插入排序	空	int*排序结果
shellSort()	希尔排序	空	int*排序结果
quicksort()	快速排序	空	int*排序结果
heapsort()	堆排序	空	int*排序结果
mergeSort()	归并排序	空	int*排序结果
radixSort()	基数排序	空	int*排序结果
copy()	产生数据表副本	空	int*副本
print(int*)	打印输出排序结果及排序性能	int* 要输出的排序结果	void
printOrigin()	打印输出原数据表	空	void
partition(int *, const int , const int)	快速排序基础函数	int* 待排序数组 int low 待排序低位 int high 待排序高位	int
quick(int*,const int, const int)	对目标区间快速排序	int* 待排序数组 int left 待排序低位 int right 待排序高位	void
heapAdjust(int *heap, int current, int end)	调整为最大堆	int* 目标最大堆 int current 当前待调整节点 int end 最后一个待调整节点	void
merge(int*, int, int)	归并排序基础函数	int* 待排序序列 int begin 待排序低位 int end 待排序高位	void
swap(int &a, int &b)	交换函数（内置计数）	int&a,&b 待交换数	void
clear()	重置排序性能	空	void

表 1　DataList 类接口