数据结构课程设计

题目十 八种排序算法的比较案例

文档

班级：软件5班

学号：1353010

姓名：薛梦迪

**一、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。

**二、项目功能：**

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。

**三、项目数据格式说明与类说明：**

1、PQueue.h

1）引用的头文件：

#include"assert.h"

#include"stdlib.h"

2）内容：最小优先级队列 PQueue 类定义

template <class T>

|  |  |  |  |
| --- | --- | --- | --- |
| **访问控制类型** | **成员名称** | **成员类型** | **成员功能** |
| public | Pqueue(PQueue(int sz = DefaultPQSize) | constructor | 构造函数，构造一个最小优先级队列 |
| Insert(const T& x) | bool | 将新元素x插入到队尾 |
| RemoveMin(T& x) | bool | 将队头元素删去 |
| getFront(T& x)const | bool | 读取队头（具最小优先权）的值 |
| makeEmpty() | void | 置优先级队列为空 |
| IsEmpty()const | bool | 判队列空否 |
| IsFull()const | bool | 判队列满否 |
| getSize()const | int | 求优先级队列中的元素个数 |
| protected | pqelements | T\* | 优先级队列数组 |
| count | int | 当前元素个数（长度） |
| maxSize | int | 队列最大可容纳元素个数 |
| adjust() | void | 队列调整 |

2、heap.h

1）引用的头文件：

#include"PQueue.h"

#include<iostream>

2）内容：最小堆MinHeap 类定义（继承自PQueue类）

template <class E>

|  |  |  |  |
| --- | --- | --- | --- |
| **访问控制类型** | **成员名称** | **成员类型** | **成员功能** |
| public | MinHeap(int sz = DefaultSize) | constructor | 构造函数：建立空堆 |
| MinHeap(E arr[], int n) | constructor | 构造函数：通过一个数组建堆 |
| Insert(const E& x) | bool | 将x插入到最小堆中 |
| RemoveMin(E& x) | bool | 删除堆顶上的最小元素 |
| IsEmpty()const | bool | 判断空否，空返回1，否则0 |
| IsFull()const | bool | 判断满否，满返回1，否则0 |
| MakeEmpty() | void | 置空堆 |
| private | heap | E \* | 存放最小堆中元素的数组 |
| currentSize | int | 最小堆中当前元素个数 |
| maxHeapSize | int | 最小堆最多允许元素个数 |
| siftDown(int start,int m) | void | 从start到m下滑调整成为最小堆 |
| siftUp(int start) | void | 上滑调整最小堆 |

3、data.h

1）引用的头文件：

#include <iostream>

#include <stdlib.h>

#include <time.h>

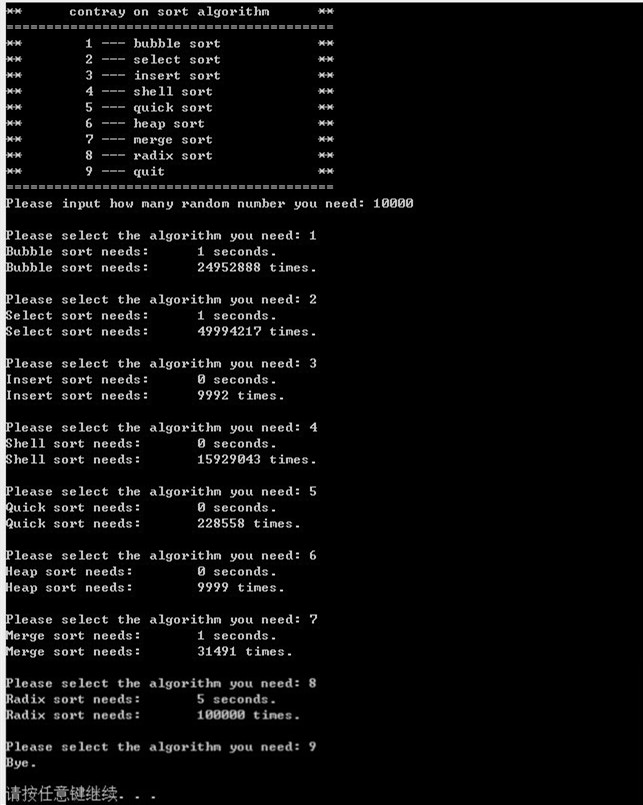
#include <random>

#include "heap.h"

2）内容：建立八种排序算法并封装

|  |  |  |  |
| --- | --- | --- | --- |
| **访问控制类型** | **成员名称** | **成员类型** | **成员功能** |
| public | getExcTime() | long | 获取交换次数 |
| getSortTime() | float | 获取排序时间 |
| datalist(int num) | constructor | 构造函数 |
| init() | void | 生成随机数 |
| clear() | void | 清除交换次数和排序时间 |
| print() | void | 打印排序后的数组 |
| getBubbleSort() | void | 打印冒泡排序的交换次数和排序时间 |
| getInsertSort() | void | 打印插入排序的交换次数和排序时间 |
| getQuickSort() | void | 打印快速排序的交换次数和排序时间 |
| getSelectSort() | void | 打印选择排序的交换次数和排序时间 |
| getHeapSort() | void | 打印栈排序的交换次数和排序时间 |
| getMergeSort() | void | 打印归并排序的交换次数和排序时间 |
| getShellSort() | void | 打印希尔排序的交换次数和排序时间 |
| getRadixSort() | void | 打印基排序的交换次数和排序时间 |
| private | sortTime | float | 存储排序所需时间 |
| exchangeTimes | long | 存储排序所需交换次数 |
| number | int | 存储随机数个数 |
| randomNums | vector<int> | 存储随机数 |
| swap(int& x,int& y) | void | 交换x和y的值 |
| print(int \*a) | void | 打印a指针为首的数组 |
| bubbleSort() | void | 冒泡排序算法 |
| selectSort() | void | 选择排序算法 |
| insertSort() | void | 插入排序算法 |
| shellSort() | void | 希尔排序算法 |
| Partition(int \*a,const int low,const int high) | int | 快速排序基础函数 |
| quick(int \*a,const int left,const int right) | void | 快速排序基础函数 |
| quickSort() | void | 快速排序算法 |
| heapSort() | void | 栈排序算法 |
| merge(int \*a,int \*b,const int l,const int m,const int n) | void | 归并排序基础函数 |
| mergeSort() | void | 归并排序算法 |
| pre\_process\_data(int array[], int length, int weight) | int | 基排序基础函数 |
| sort\_for\_basic\_number(int array[], int length, int swap[]) | void | 基排序基础函数 |
| sort\_data\_by\_basic\_number(int array[], int data[], int swap[], int length, int weight) | void | 基排序基础函数 |
| radixSort() | void | 基排序算法 |

**四、项目运行的界面：**



注：1、完成了基本操作。

2、若是选择非1-9的数会返回错误并重新选择。

3、选择9退出。

4、可以选择需要的随机数个数。

**五、项目优点和不足之处：**

本项目满足了项目的所有需求，并在输入正确的情况下能顺利执行。本项目也基本遵循了类的封装性，较好的保护了数据，更好的保存了数据，并在排序过程中不改变原容器中的值，以保证各个排序能共用一组随机数。由于类非常多，没有过多时间加入注释，这是项目的疏漏之处。总之，本项目具有较强的客观性，也难免有一些小的地方考虑不周，希望以后能更加改进。