北京師範大學
BEIJING NORMAL UNIVERSITY

《数据结构》上机实验报告

第 6 次上机

学号：202011140104
姓名：李馨
学院：物理学系
专业：物理学
教师：郑新
日期：2022.12.9

1 实验过程

1.1 实验内容

1.1.1 问题描述

对 2006 年度全国 80 多个城市的每天空气质量状况进行查询、排序等操作。空气质量状况对象包括城市代码、城市名称、首要污染物、污染指数、污染物级别、空气状况、年、月、日。

1.1.2 实验要求

1. 普通查询：输入城市名称和城市代码，分别查询该城市每天、每周、每月、每季度和全年的空气质量状况
2. 统计查询：
   (a) 输入城市名称和城市代码，分别查询该城市每周、每月、每季度和全年的空气质量为优、良、轻微污染、轻度污染、重污染的天数
   例子：查询太原市 2006 年第 8 周的空气质量状况
   (b) 根据时间查询空气质量状况：输入周编号、月编号、季度编号或年编号，以及空气质量为优、良、轻微污染、轻度污染、重污染的天数，查找相应的城市名称
   例子：查询 2006 年 5 月，空气被轻度污染 3 天以上的城市有哪些?
3. 排序查询
   (a) 输入周编号、月编号、季度编号或年编号，查询城市空气质量的排行榜
   例子：查询 2006 年第 6 周，全国空气平均质量最好的前 20 个城市为哪些?

1.1.3 数据文件

Data.txt

1.2 实验步骤

对于城市，建立 **city** 类，在每一个 **city** 对象中，有大小为 12×31 的储存相应日期空气质量数据（**AQCondition** 对象，即空气质量状况对象）的二维数组。由数组索引可以直接读取该城市相应日期的数据，从而较为简单地实现 **1. 普通查询**和 **2. 统计查询**。

对于 **3. 排序查询**，通过访问每个城市对应的 **city** 对象中的数据，计算其在给定时间段的平均空气质量指数，再而使用插入排序算法进行排序，即可得到排行榜。

1.3 实验过程

1.3.1 读取数据（AQCondition.cpp, city.cpp）

主要涉及 **AQCondition.cpp** 中 **class AQCondition** 的定义，以及 city.cpp 中 **class city** 的定义和 **readAQCs** 函数。

```cpp
class AQCondition {
public:
    int num;
    string name;
    string primaryPollutant;
    int index;
    string degreeNum;
    string degree;
    dateIn2006 AQCdate;  // 日期

    // 构造函数
    AQCondition() {}
    AQCondition(int num, string name, string prPoll, int index,
    string degreeNum, string degree, int m, int d) : …

    // 打印
    void display() {…

};
```

图 1: AQCondition.cpp：class AQCondition

```cpp
class city {
public:
    int num;                        // 城市序号
    string name;                    // 城市名
    AQCondition AQCList[12][31];    // 该城市对应的所有空气质量状况数据组成的二维数组

    // 构造函数
    city(int a = 0, string b = "")
    : num(a), name(b) {}
```

图 2: city.cpp：class city

在 **readAQCs** 函数中，对于每行数据，读取后转换为一个 **AQCondition** 对象，然后根据数据中的城市存入相应的 **city** 对象中。最后 80 多个城市的 **city** 对象一起存入一个 **vector<city>** 对象 cityList 中，它通过引用型变量返回。

1.3.2 普通查询（city.cpp）

我将数据按城市存入了不同的 **city** 对象中，那么针对特定城市的各种查询，不如定义为 **city** 类的方法。由此，普通查询通过 city 类中 **searchWeekAQ**、**searchMonthAQ**、**searchSeasonAQ**、**searchYearAQ** 四个成员函数实现。

在这之前，输入要查询的城市后，在 **vector<city>** cityList 中首先需要顺序查找一遍找到这个城市对象。

城市对象某一日期的数据通过数组索引直接查询。以 **searchMonthAQ** 函数为例：

```cpp
// 查询该城市每月的空气质量状况
void searchMonthAQ(int m) {
    int mList[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    int max = mList[m - 1]; // 迭代次数
    for (int i = 1; i <= max; i++) {
        if (AQCList[m - 1][i - 1].AQCdate.month == 0) {
            cout << name << " 在 "; dateIn2006(m, i).display(); cout << " 没有数据！" << endl;
        }
        else {
            AQCList[m - 1][i - 1].display();
        }
    }
}
```

图 3: city.cpp：searchMonthAQ

1.3.3 统计查询：城市查询（city.cpp）

这一步与普通查询同样是**针对特定城市的查询**，唯一不同是需要统计和计数。所以同样通过在 **city** 类中定义成员函数来实现，即 **staSearchWeekAQ**、**staSearchMonthAQ**、**staSearchSeasonAQ**、**staSearchYearAQ** 函数。

1.3.4 统计查询：时间查询（city.cpp）

由于要考察 **vector<city>** cityList 中的所有城市对象，显然不能再通过在城市类中定义成员函数来实现了。只好在外部进行。

遍历 **vector<city>** cityList 储存的所有城市对象，对之依次进行特定时间城市查询（调用上一小节的函数）即可，最后将满足条件的城市名存入一个 **vector<string>** 中。以函数 **timeSearchMonthAQ** 为例：

```cpp
// 查询某月某空气质量状况满足大于 min 的城市，城市名通过 result 返回。
void timeSearchMonthAQ(int m, int level, int min, vector<city> &cityList, vector<string> &result) {
    int max = cityList.size(); // 城市个数
    city p;
    int num[5] = {0};
    // 遍历城市列表
    for (int i = 0; i < max; i++) {
        for(int j = 0; j < 5; j++) {
            num[j] = 0;
        }
        p = cityList[i];
        p.staSearchMonthAQ(m, num);
        if (num[level - 1] > min) {
            result.push_back(p.name);
        }
    }
}
```

图 4: city.cpp：timeSearchMonthAQ

1.3.5 排序查询

为了储存城市和相应的平均空气质量指数，我考虑过用 **map** 容器，但是它在插入新元素时会自动排序。也可以为原来的 **city** 类新添加一个平均空气质量指数的属性。不过我最后选择新建了一个 **cityAQC** 类来处理这个问题。

排序查询主要由下图五个函数实现，其中前四个函数用于返回城市和对应的平均空气质量指数组成的未排序的 **vector<cityAQC>**，最后一个函数 **insertSort** 以每个 cityAQC 对象的平均空气质量指数 averageAQ 为排序码进行**插入排序**。

```
286   // 通过引用返回某周各城市 城市名-平均空气质量指数 组成的 cityAQC 类的 vector
287 > void weekAverageAQ(int w, vector<city> &cityList, vector<cityAQC> &result) {…
308
309   // 通过引用返回某月各城市 城市名-平均空气质量指数
310 > void monthAverageAQ(int m, vector<city> &cityList, vector<cityAQC> &result) {…
331
332   // 通过引用返回某季度各城市 城市名-平均空气质量指数
333 > void seasonAverageAQ(int s, vector<city> &cityList, vector<cityAQC> &result) {…
358
359   // 通过引用返回一整年各城市 城市名-平均空气质量指数
360 > void yearAverageAQ(vector<city> &cityList, vector<cityAQC> &result) {…
383
384   // 插入排序
385 > void insertSort(vector<cityAQC> &arr) {…
```

图 5: city.cpp：排序查询相关函数

1.4 运行结果（main.cpp）

1.4.1 普通查询

由于篇幅所限，这里展示部分运行结果。

北京的日、周、月普通查询：

```
clem@Connor:/mnt/c/Users/12879/Desktop/dataStructures/projects/project06/new$ g++ main.cpp -o main.out
clem@Connor:/mnt/c/Users/12879/Desktop/dataStructures/projects/project06/new$ ./main.out
你要进行 1普通查询/2统计查询/3排序查询？请输入序号：1
已进入普通查询。你要查询哪个城市？请输入城市名：北京
你要进行 日/周/月/年 查询？请输入：
日
请输入日期，月和日以空格隔开 (如1月25日输入1 25):2 3
北京2006-2-3的数据如下：
110000  北京    可吸入颗粒物    56      II      良              2006    2       3
```

图 6: 普通查询-日查询

```
clem@Connor:/mnt/c/Users/12879/Desktop/dataStructures/projects/project06/new$ ./main.out
你要进行 1普通查询/2统计查询/3排序查询？请输入序号：1
已进入普通查询。你要查询哪个城市？请输入城市名：北京
你要进行 日/周/月/年 查询？请输入：
周
请输入要查询的周数 (不应超过52):4
北京第 4 周的数据如下：
110000  北京    可吸入颗粒物    243     Ⅳ1     中度污染        2006    1       16
110000  北京    可吸入颗粒物    116     Ⅲ1     轻微污染        2006    1       17
110000  北京    可吸入颗粒物    110     Ⅲ1     轻微污染        2006    1       18
110000  北京    可吸入颗粒物    181     Ⅳ2     轻度污染        2006    1       19
110000  北京    可吸入颗粒物    273     Ⅳ2     中度重污染      2006    1       20
110000  北京    可吸入颗粒物    310     Ⅴ      重污染          2006    1       21
北京 在 2006-1-22 没有数据！
```

图 7: 普通查询-周查询

```
clem@Connor:/mnt/c/Users/12879/Desktop/dataStructures/projects/project06/new$ ./main.out
你要进行 1普通查询/2统计查询/3排序查询？请输入序号：1
已进入普通查询。你要查询哪个城市？请输入城市名：北京
你要进行 日/周/月/年 查询？请输入：
月
请输入要查询的月份:3
北京3 月的数据如下：
110000  北京    可吸入颗粒物    62      II      良              2006    3       1
110000  北京    --              49      I       优              2006    3       2
110000  北京    可吸入颗粒物    89      II      良              2006    3       3
110000  北京    可吸入颗粒物    154     III2    轻度污染        2006    3       4
110000  北京    可吸入颗粒物    99      II      良              2006    3       5
110000  北京    可吸入颗粒物    80      II      良              2006    3       6
110000  北京    可吸入颗粒物    90      II      良              2006    3       7
110000  北京    可吸入颗粒物    98      II      良              2006    3       8
110000  北京    可吸入颗粒物    91      II      良              2006    3       9
110000  北京    可吸入颗粒物    408     V       重污染          2006    3       10
110000  北京    可吸入颗粒物    95      II      良              2006    3       11
110000  北京    可吸入颗粒物    85      II      良              2006    3       12
110000  北京    可吸入颗粒物    90      II      良              2006    3       13
110000  北京    可吸入颗粒物    88      II      良              2006    3       14
110000  北京    可吸入颗粒物    109     III1    轻微污染        2006    3       15
110000  北京    可吸入颗粒物    91      II      良              2006    3       16
110000  北京    可吸入颗粒物    135     III1    轻微污染        2006    3       17
110000  北京    可吸入颗粒物    256     IV2     中度重污染      2006    3       18
110000  北京    可吸入颗粒物    84      II      良              2006    3       19
110000  北京    可吸入颗粒物    226     IV1     中度污染        2006    3       20
110000  北京    可吸入颗粒物    197     III2    轻度污染        2006    3       21
110000  北京    可吸入颗粒物    181     III2    轻度污染        2006    3       22
110000  北京    可吸入颗粒物    66      II      良              2006    3       23
110000  北京    可吸入颗粒物    97      II      良              2006    3       24
110000  北京    可吸入颗粒物    206     IV1     中度污染        2006    3       25
110000  北京    可吸入颗粒物    99      II      良              2006    3       26
110000  北京    可吸入颗粒物    347     V       重污染          2006    3       27
110000  北京    可吸入颗粒物    98      II      良              2006    3       28
110000  北京    可吸入颗粒物    124     III1    轻微污染        2006    3       29
110000  北京    可吸入颗粒物    92      II      良              2006    3       30
110000  北京    可吸入颗粒物    96      II      良              2006    3       31
```

图 8: 普通查询-月查询

1.4.2 统计查询

```
clem@Connor:/mnt/c/Users/12879/Desktop/dataStructures/projects/project06/new$ ./main.out
你要进行 1普通查询/2统计查询/3排序查询？请输入序号：2
已进入统计查询。接下来你要进行 1城市查询/2时间查询？请输入序号：1
你要查询哪个城市？请输入城市名：哈尔滨
你要进行 周/月/年 查询？请输入：
年
哈尔滨2006年空气质量天数
优              良              轻微污染        轻度污染        重污染
22              243             40              5               0
```

图 9: 统计查询-城市查询-年查询

```
clem@Connor:/mnt/c/Users/12879/Desktop/dataStructures/projects/project06/new$ ./main.out
你要进行 1普通查询/2统计查询/3排序查询？请输入序号：2
已进入统计查询。接下来你要进行 1城市查询/2时间查询？请输入序号： 2
你要进行 周/月/年 查询？请输入：
年
你要查询 1优/2良/3轻微污染/4轻度污染/5重污染 天数在几天以上的城市？请依次输入要查询的 类型序号 和 天数
 并以空格隔开（例："1 4"表示空气质量为优4天以上）：
1 30
2006年空气质量为优在30天以上的城市有：
南通 秦皇岛 呼和浩特 大连 牡丹江 上海 南京 苏州 连云港 扬州 镇江 杭州 宁波 温州 湖州 绍兴 合肥 芜湖 福
州 厦门 泉州 南昌 青岛 烟台 济宁 泰安 日照 荆州 常德 张家界 广州 韶关 深圳 珠海 汕头 湛江 南宁 桂林 柳
州 北海 海口 重庆 成都 南充 贵阳 玉溪 拉萨 银川 石嘴山 乌鲁木齐 克拉玛依
```

图 10: 统计查询-时间查询-年查询

1.4.3 排序查询

```
clem@Connor:/mnt/c/Users/12879/Desktop/dataStructures/projects/project06/new$ ./main.out
你要进行 1普通查询/2统计查询/3排序查询？请输入序号：3
已进入排序查询。
你要进行 周/月/年 查询？请输入：
月
请输入要查询的月份:5
2006年第5月的平均空气质量排行榜（前20名）如下：
珠海            27.966667
北海            34.266667
海口            34.333333
桂林            40.333333
广州            42.266667
深圳            43.366667
烟台            44.733333
汕头            45.000000
湛江            47.233333
拉萨            48.766667
玉溪            49.866667
泰安            50.866667
厦门            51.333333
南宁            51.566667
日照            51.633333
牡丹江          53.433333
克拉玛依        53.966667
泉州            54.700000
张家界          55.733333
南昌            58.266667
```

图 11: 统计查询-排序查询-月查询

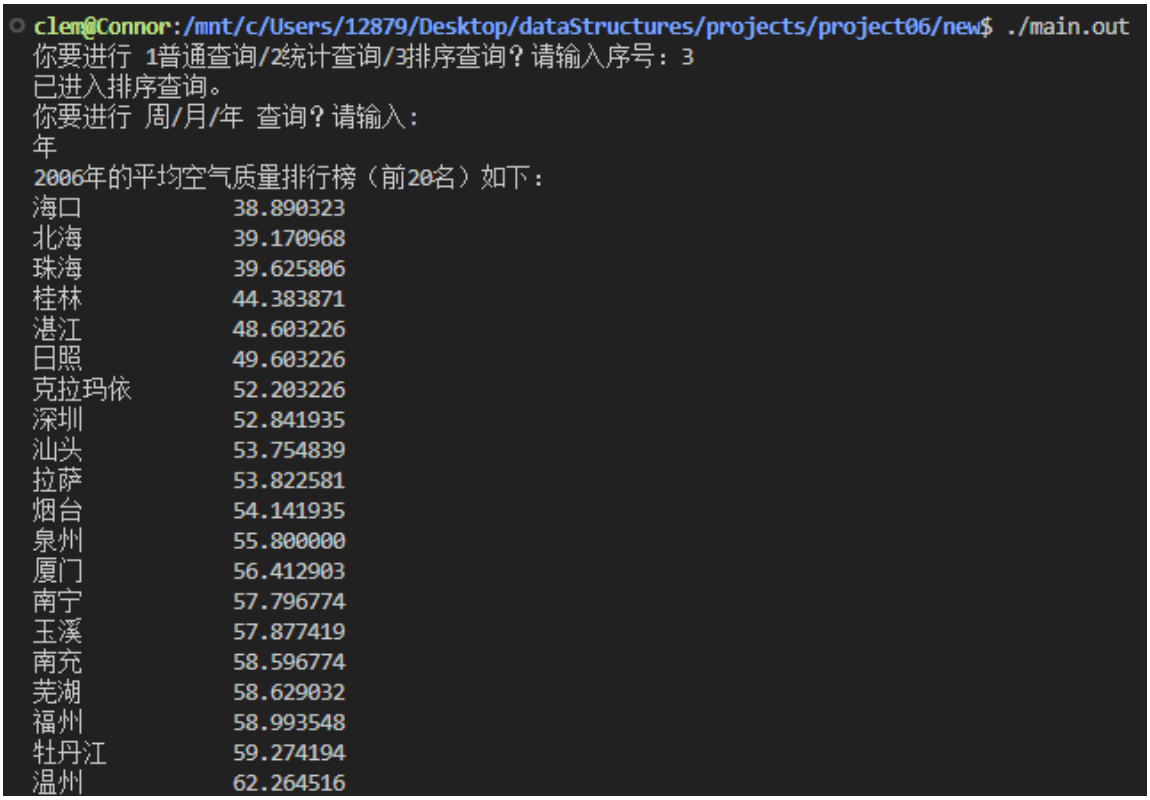

图 12: 统计查询-排序查询-年查询

2 总结

我觉得本次实验中很令人难以下手的问题是如何存储这些数据，能使得各种查询的效率更高一些，而且尽量减少内存的占用。排序则是相对简单的与存储结构关系不那么大的步骤。

由于城市的数量（约 80 个）比日期的数量（约 365 天）少一些，我最后选择把约 365 天的数据存入约 80 个城市中。查询时，给定城市通过顺序查找，给定日期通过二维数组索引直接访问。我也想了一些在此基础上可能可以进一步优化的道路：

1. 查找城市的时候，也可将所有城市以城市代码为关键码进行有序化，再而折半查找等，可略微优化查找效率。

2. 如果数据量庞大，不希望把所有数据都存在内存里，可以让 city 类的 AQCList[12][31] 属性只储存给定日期数据在 txt 文件对应的行数，然后写一个给定行数提取文件数据的函数，每次查询时都重新读取一遍文件。但感觉会大幅拖慢效率。