股票交易数据处理

——2018 年秋季学期《程序设计 I》课程大作业

Date	Open	High	Low	Close	Adj Close	Volume	Code	Change	Pct_change
2017-01-03	9.1	9.18	9.09	9.16	9.028069	45984049	000001.SZ	0.06	0.0065934
2017-01-04	9.15	9.18	9.14	9.16	9.028069	44932953	000001.SZ	0	0
2017-01-05	9.17	9.18	9.15	9.17	9.037926	34437291	000001.SZ	0.01	0.0010917
2017-01-06	9.17	9.17	9.11	9.13	8.998502	35815420	000001.SZ	-0.04	-0.004362
2017-01-09	9.13	9.17	9.11	9.15	9.018213	36108157	000001.SZ	0.02	0.0021906
2017-01-10	9.15	9.16	9.14	9.15	9.018213	24105395	000001.SZ	0	0

如上表所示，现有某个阶段中国股市的 cvs 格式数据（数据样例如上，从左开始的列名（字段）依次是日期、开盘价、最高价、最低价、收盘价、调整后收盘价、成交量、股票代码、价格上涨、价格上涨百分比），请你编写程序对其进行分析处理并完成以下任务。

一、读入数据

实现函数 LoadStockData 加载数据并补齐缺失数据。

- 函数原型：int **LoadStockData**(char *filename);

具体要求如下：

- ✓ 加载 cvs 格式的股票数据，保存在适当的数据结构中，并给每一行记录给一个唯一的编号(index)，按照原始数据的顺序，从 0 开始编号。
- ✓ 加载成功，返回总的记录条数；否则返回-1
- ✓ 数据中有些缺失（例如，某只股票某天停盘会缺失开盘价、最高价、最低价、收盘价等信息），对于这些有缺失的数据，请将所有价格（包括开盘、收盘等等）用数据**缺失前**离这个数据最近的“调整后收盘价(Adj Close)”替代；其余字段如 volume、change 和 pct_change 填补为 0。e.g. 某股票 2017.7.1 的数据有缺失，找此日期之前最接近的数据来替补。

二、提取数据

实现对指定字段值的提取。在本例中，不同的字段的值的类型不尽相同，有字符串、浮点数（double）和整数等。现需设计一个统一的函数来提取指定记录里给定字段的值。提取成功返回 1，否则返回 0。

- 函数的原型：int **GetFieldVal**(int index, const char * name, **void** * pvalue);

其中：

1) index 表示数据记录的编号，具体见任务一描述；
2) name 表示字段名，如成交量 "Volume"。
3) pvalue 是一个空类型指针地址，它指向查询到的具体字段值。在函数中使用时，需要根据字段对应的数据类型，转换为有相应类型的指针。

应用示例：

```
int vol;  //用以保存提取的成交量
GetFieldVal(123, "Volume", &vol); //将编号 123 行记录中的交易量提取到 vol 中
```

三、记录排序

当有单条记录数据比较大，对所有记录进行排序的一种方案是：并不需要移动数据本身，而只是移动数据记录的编号，编号顺序能体现出所需的记录顺序。请你实现一个排序函数，根据给定的排序准则对记录编号数组进行排序。这里的重点是排序准则，它由一个字符串表达，相对较为灵活：

1) 由若干字段名构成，以逗号分隔，如”Code,Close,Date”;
2) 字段的顺序体现排序的优先级别，比如上例，先根据股票代码 Code 排序，代码相等的情况下，再依据收盘价 Close，若代码、收盘价都相等，再依据交易日期 Date，以此类推；
3) 默认升序排列。

- 函数的原型：void **SortRecods**(int reco_index[], int n, const char *order_by);

其中：

1) reco_index []数组代表所有须排序的数据记录的编号，运行结束后，其中是已按要求排过序的索引号；
2) n 表示记录的数量；
3) order_by 指向排序的准则，它由若干逗号分隔的字段名组成；
4) 在排序比较时，涉及到浮点数值比较时，当绝对值相差小于 0.0001，即可视为相等。

应用示例：

```
int reco_index [10000];
//假设选取分配了 10000 行数据的编号存入到了 reco_index 数组中
char * order_by = "Open,Volume";
SortRecods(reco_index, 10000, order_by);  //对查询到的结果再进行排序
```

四、查询类任务

实现数据记录查询选择函数 select，该函数根据**表达查询条件的字符串**，对加载的数据进行查询。其结果包含两部分，一是符合条件的记录总数，以变量引用的方式，更新保存在调用实参中；二是符合条件的所有记录编号，通过动态数组地址的方式，作为函数值返回。关于查询条件字符串，它类似编程语言的逻辑表达式结构，其构成**可包含**：

1) 表示关系运算的字符： "<", "<=", ">", ">=", "==", "!=";
2) 表示逻辑运算的关键字："AND", "OR", "NOT"；
3) 可改变运算优先级别的小括号 "()"，若没有括号两类运算符的优先次序与 C/C++ 中相应的保持一致；
4) 字段名，表示对当前记录行指定字段值的引用，如 "Open >= 9.15" 表达的条件就是开盘价大于等于 9.15；
5) 为简化，约定字段名在关系运算符的左边，标准格式下运算符前后各一个空格。

通过以上内容，可以灵活写出简单或复合的查询条件。比如 “Open >= 9.15 AND Date

== 2017-1-3”表达了查找 2017 年 1 月 3 日开盘价大于等于 9.15 的股票交易记录。其中，表达日期时，采取“年-月-日”短横线连接的固定格式。

提示：你需**再设计一个解析查询条件字符串的方案**，以判断一条记录是否符合需要。

- 函数的原型：int* **Select**(const char *condition, int& n); 其中：

1) condition 为查询的条件表达式，具体规则如上所述；
2) n 为结果的数量，这里是以引用的形式传入。
3) 返回值为整数类型的指针存放数据编号的数组，应考虑将它创建为一个**动态数组空间**，因为查询结果的具体数目事先不可预知。

应用示例：

```
char * query_condtion = "( Open > 9.0 OR Close > 9.15) AND Date == 2017-1-15";
int count;                                          //初始分配 100 个空间，
int * reco_index = Select(query_condtion, count);//Select 函数可根据需要调整空间
//…….
free(reco_index);
```

五、分组统计

在有大量数据记录的情况下，分组统计是常见的需求。分组规则是某一个字段，该字段值相等记录视为一组。分组统计即在按字段分组的基础上，得到每组的统计量，常见统计量有**最大值、平均值、累计求和、计数**等。现需实现一个函数，根据统计指令和分组规则，进行统计，并输出分组情况及所得统计值。统计指令有固定的格式，分组依据由字符串表达，类似上述的排序准则。

1) 只考虑四个统计指令：MAX、AVG、SUM 和 COUNT，它们分别与某一字段相结合如 "MAX(Open)" 表示每组的最大开盘价，"AVG(Close) "表示每组的平均收盘价，对于 COUNT 指令，无需指定特别的字段，以"COUNT(*)"表达；
2) 分组依据，根据本例数据只考虑一个字段，比如按日期分组，或按股票代码分组。

- 函数的原型：

```
int Aggregate (int reco_index[], int n, const char *cmd, const char *group_by);
```

其中：

1) reco_index 和 n 的含义与前面第三项任务中相同；
2) cmd 表示统计指令，如"COUNT(*)"；
3) group_by 为分组依据，如"Date"

函数的返回值为所得的分组数。所得的各组情况及指定的统计值由本函数负责输出。

输出格式：

1) 标题行，第一行为标题行，包含分组字段名 和 统计指令；
2) 此后每行包含分组字段对应的取值 和 具体的统计值，是小数的，保留 4 位；
3) 按分组字段的值从小到大的顺序，输出分组结果；
4) 输出格式参照下面的示例

应用示例：

Aggregate (reco_index, n, "AVG(Open)", "Code"); //redoIndex, n 指明待分组的数据子集，
//如经查询在某一日期范围内的数据

//输出结果，类似如下的形式

Code,AVG(Open)

000001.SZ,9.1522

000002.SZ,15.5578

六、数据输出

在第三和第四子问题中，可根据得到的数据记录编号数组，用一个统一的输出函数，输出相应的原始数据到指定文件。内容和格式说明如下：

1) 第 1 行为标题行，与原始数据中的保持一致；
2) 从第 2 行起，按标题行的字段顺序，输出相应的值；
3) 小数保留 2 位；
4) 输出格式：数据项之间用逗号分隔，参照原始数据格式。

- 函数原型：**OutputToFile** (int reco_index[], int n, FILE *pf); 其中，pf 为指向输出文件的指针。

七、寻找最大收益（独立程序）

你有一笔闲置资金 X 万元，可以用于投资股市。现给定 n 只股票，以及这些股票某个时间段内的交易信息，在这个时间段内，你可以在某一天从这些股票中选择一支或者几支购买，并在你认为合适的时间内出售股票。时间段的最后一天要求你将所有的股票兑现。请问，兑现后你手中所有的资金最多是多少元？

注意：

1) 为方便计算，买卖股票统一按照某天的收盘价进行。
2) 股票购买时，只能购买整数股。
3) 股票售出时会收取一定的手续费，费用总成交金额的 g%。

【输入格式】

第一行三个整数 X、n、g 表示开始时共有 X 万元，一共有 n 支股票可供选择，每卖出一支股票，需缴纳成交额的百分之 g 的手续费；

第二行两个字符串，表示开始时间和结束时间，格式同股票数据；

接下来的 n 行，每行一个股票代码，保证这些股票代码在给定的时间段内有数据。

【输出格式】

输出共 m*2 行，m 表示开始时间到结束时间内的可交易日的天数，例如 2000-01-03 到 2000-01-04 为两天，2000-01-03 到 2000-03-03 为 46 天。假设时段内 m 个交易日，则输出 m*2 行。

每 2 行表示一个交易日的操作。其中第 1 行表示卖出股票的信息，包括一个整数 m（卖出 m 支股票），之后跟 m 组数据，每组数据包括一支卖出股票的代码和相应卖出的股数。如果没有卖出任何股票，此行只输出 0；第 2 行表示买入股票的信息，包括一个

整数 k（买入 k 支股票），之后跟 k 组数据，每组数据包括一支买入股票的代码，相应的买入数量，如果没有买入任何股票，只输出 0。数据项之间用空格隔开。

【注意】

1. 买入卖出的股数都必须是整数，而且必须能够进行操作，比如手中的现金必须大于买股票花的钱，卖出的股票股数必须小于等于自己当前持有的该股票的股数，如果输出不合法，则判定程序出错，计 0 分。
2. 测试点共十组数据，每个人的程序会跑十组数据，根据个人的决策会有一个收益，该测试点的得分为你的收益除以参考程序的收益乘 10，每个测试点最高分 10 分。
3. 提交一个完整的 cpp 文件，以你的学号命名，例如 20182001234.cpp，该程序从当前目录加载数据（数据文件的文件名同给定的完整股票数据），从标准输入读取输入内容，并将结果输出到屏幕上。

【样例输入】

```
10000 5 2 0 0
2000-01-03 2000-03-03
000018.SZ
000016.SZ
000014.SZ
000005.SZ
000007.SZ
```

【样例输出】

```
0
1 000018.SZ 4507
0
0
0
0
0
0
0
0
1 000018.SZ 4507
0
0
0
0
0
0
1 000005.SZ 3866
0
0
0
```

0
1 000005.SZ 3866
1 000018.SZ 5528
1 000018.SZ 5528
1 000005.SZ 4119
1 000005.SZ 4119
1 000018.SZ 5766
0
0
1 000018.SZ 5766
1 000014.SZ 8049
0
0
1 000014.SZ 8049
1 000007.SZ 4641
1 000007.SZ 4641
1 000005.SZ 4550
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
1 000005.SZ 4550
1 000014.SZ 9483
1 000014.SZ 9483
1 000018.SZ 7045
0

0
1 000018.SZ 7045
1 000007.SZ 4989
0
0
1 000007.SZ 4989
0
0
0
0
1 000007.SZ 5966
1 000007.SZ 5966
1 000014.SZ 10834
0
0
0
0
0
0
1 000014.SZ 10834
1 000007.SZ 7104
1 000007.SZ 7104
1 000016.SZ 11116
1 000016.SZ 11116
0
0
0

样例说明：上述方式，最终钱数为 35830.855459，收益为 25830.855459