**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“调整后收盘价(Adj Close)”调整后收盘价替代；其余字段如volume、change和pct\_change填补为0。

**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，以函数值的形式返回；二是符合条件的所有记录编号，存入通过形参传递而来的数组中。关于查询条件字符串，它类似编程语言的逻辑表达式的结构，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. 为了简化，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 的含义与前面第四项任务中相同；
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%。
4. 股票收益要缴税，税率如下：
   * 1. 持有<d天，税率为2t%
     2. 持有>=d天，税率为t%
     3. 上述计算过程中，请使用double类型，并保留小数点后2位。

**【输入格式】**

第一行5个整数：X、n、g、d、t；

第二行2个字符串t1, t2, 表示时间段为[t1, t2]，格式如原始数据中的date格式；

第三行n个字符串，表示n只股票的股票代码；

**【输出格式】**

假设时段内m个交易日，则输出m行。

每行有n+1数，每2个数之间用一个空格隔开，前n个数表示当前交易日结束时，这n支股票的持有量；最后一个小数表示当前手中的现金额。

注意：按照题目要求最后一行前n个整数应该均为0，而最后一个小数就表示最终手中的资金。

【样例待更新】