# 摩根项目介绍

516030910435

汤志彪

## Sort

**1.Orders.h**

Class order：

用于存储每条从文件中读取的order，其中每条order包括order id, side, symbol, price 和 quantity。Quantity为public, 主要目的是为了交易的时候改变剩余的数量。因此，order传递的时候只能拷贝。

**2.Quotes.h**

Class quote：

用于存储每条从文件中读取的quote，其中每条quote包括exchange, side, symbol, price 和 size。其中size 的作用同order中的quantity。

**3.Report.h**

Class report：

用于存取每条交易记录。每条交易记录都包含order中的order id & side, quote中的exchange, 及交易过程中的交易价格price, 品牌symbol, 数量quantity。

**4.Some\_functions.h**

Str\_to\_dou():

将字符串转变为double类型的数字，用于读取文件。

Str\_to\_int():

将字符串转变为int类型的数字，用于读取文件。

Keep\_window\_open():

程序运行结束后保持窗口，输入一个字符后窗口关闭。

**5.Sort.cpp**

get\_order():

用于读取文件中的order，把每一条读到的order添加到一个向量中，并返回这个向量。

Order文件的地址本来要用户输入（读取文件部分注释中的代码），为了自己测试方便，已把文件地址固定为自己计算机里的地址。

get\_quote()

用于读取文件中的quote，把每一条读到的quote添加到一个向量中，并返回这个向量。

Aggressive\_take()：

实现了homework2 部分的要求，对每读一条quote，对vector<order>中的order进行遍历。若匹配，则进行交易，将交易内容report写入vector<report>中；否则跳过该条quote，继续读取下一条quote。

交易结束之后，建立了四个map。两个分别存储每个exchange中的总成交量和成交总价，另两个分别储存每条order的总成交量和成交总价。通过遍历vector<report>来填充map中的数据，最后输出每个交易所及每条order的总成交量和平均成交价格。

## 未完成部分

**1.Level1\_book.h:**

Class book:

讲不满足order的quote存入其中。该类中只包含了Symbol，Side， NYSE\_size = 0，NYSE\_price = 0，NASDAQ\_size，NASDAQ\_price，IEX\_size，IEX\_price，所以如果要存取全部类型的数据，其实真正的level1\_book包含了4个book类（在注释的Passive\_placement函数中）：

Book Bid\_AAPL("AAPL", "Bid");

Book Ask\_AAPL("AAPL", "Ask");

Book Bid\_GOOG("GOOG", "Bid");

Book Ask\_GOOG("GOOG", "Ask");

2. Passive\_placement（在sort.cpp的注释中）：

未完成，基本思路：通过读取每条order遍历quote，将不匹配的quote放在level1\_book中。