#### Коротко о протоколе

Протокол FIX — это протокол обмена биржевой информацией. Это один из протоколов, позволяющих получить прямой доступ к торгам на Московской бирже. В данной статье идет речь прежде всего о версии 4.4, приведенной в соответствие с требованиями МосБиржи. Моей целью не было в деталях реализовать весь протокол, потому я ограничился только клиентской частью и минимальным набором функций, необходимых для работы, но, если это будет необходимо, программа будет дорабатываться. Под минимальным набором функций подразумеваются функции снятия, выставления и изменения одиночных заявок, инициализация и поддержание сессии.

Сообщения FIX представляют из себя строку, состоящую из элементов типа ''id=value'', разделенных SOH символом (код 1 в ASCII), где id — идентификатор поля, value — значение

Например, типичное Logon сообщение выглядит следующим образом:

8=FIX.4.4[]9=93[]35=A[]49=A[]56=B[]34=1[]52=20200113-23:57:12.000000[]108=2[]141=Y[]554=7317[]98=0[]10=091[]

*Квадратные скобки выступают в роли разделителя.*

Каждое сообщение вне зависимости от типа содержит три группы полей: Header, Body, Trailer.

Группа Header содержит в себе информацию о типе, порядковом номере сообщения, времени отправления, идентификаторы отправителя и получателя.

Группа Body содержит поля самого запроса.

Группа Trailer содержит единственное поле — контрольную сумму.

Помимо этих трех основных групп, FIX предусматривает множество других, которые, как правило, помещаются внутри группы Body и предоставляют информацию об инструменте, сторонах сделки, комиссиях и т.д.

Алгоритм работы протокола следующий:

1. Устанавливается TCP соединение клиента и сервера
2. Клиент отправляет сообщение типа Logon серверу
3. При удачном выполнении клиент получает ответное сообщение Logon, в противном случае соединение разрывается.
4. Для проверки состояния соединения во время сессии сервер периодически посылает сообщения типа Heartbeat и Test Request, клиент, в свою очередь, посылает встречные Heartbeat сообщения, в противном случае соединение разрывается.
5. Для завершения работы клиент посылает Logout, в случае корректного завершения сервер посылает ответное Logout сообщение.

При реализации протокола стоит иметь в виду несколько вещей:

* Контрольная сумма считается как сумма кодов всех символов в строке по модулю 256
* Длина сообщени BodyLength(9) считается как длина оставшейся части заголовка + длина Body — разделители тоже считаются
* Чаще всего нет разницы, в каком порядке поля следуют друг за другом в пределах одной группы, но иногда встречаются исключения, поэтому все же рекомендую использовать именно тот порядок, которому они следуют в спецификации.
* Поля могут быть обязательными, необязательными и условными.  
  К примеру, поле Text(58) в Logout сообщении может передавать, а может и не передавать пояснение причины завершения сессии, в то время как поле Price(44) в NewOrder-Single передается в зависимости от значения OrdType(40) (для рыночных и средневзвешенных заявок 44=0 либо отсутствует вовсе, иначе заявка не будет выставлена).

#### Реализация

Исходная задача разбита на несколько подзадач, решение которых реализуется множеством классов. Количество связей я попытался сделать минимальным.

* MessageFormatter — реализует методы, связанные с форматированием и созданием уже готовой строки запроса.
* InputProcessor – обработчик входящих сообщений, содержит объекты классов SystemInputProcessor и ClientInputProcessor, одному из которых, в зависимости от типа сообщения, оно направляется.
* OutputProcessor – занимается формированием и передачей сообщения объекту класса TCPClient, который, в свою очередь, предназначен для осуществления TCP-соединения и передачи/получения сообщений(на уровне сокетов).
* OrderManager – обрабатывает ответы на запросы на выставление/снятие заявок.
* FIX44 – класс, выступающий интерфейсом, с помощью которого пользователь может обращаться к методам остальных классов.

#### Описание процесса работы программы

После создания объекта класса *FIX44* происходит инициализация сессии (прежде всего это означает создание TCP соединения и инициализацию атрибутов) посредством вызова метода *FIX44Init*, которому необходимо передать ID отправителя (логин), ID, IP и порт принимающей стороны.

Чтобы залогиниться и начать обмен сообщениями, вызывается метод Logon, которому передаются пароль, а также значения полей *HeartBtInt* и *ResetSeqNum* (по умолчанию они имеют значения ''2'' и ''Y'' соответственно).

При вызове Logon происходят две важные вещи: во-первых, посылается соответствующее сообщение принимающей стороне, во вторых, создается вспомогательный поток, который, работая в бесконечном цикле, проверяет буфер на наличие входящих сообщений и при получении направляет их обработчику.

Теперь наша программа может ''слышать''. А значит, с периодичностью *HeartBtInt* она будет получать сообщения типа *HeartBeat* и *TestRequest*, на последние отвечая встречным *HeartBeat* сообщением.

Для выставления и снятия заявок нужно использовать методы *NewOrderSingle* и *OrderCancelRequest*, которые принимают в качестве аргументов указатели на структуры *NewOrder\_sngl* и *OrderCancel* соответственно. В обоих случаях, при удачном выполнении операции, в ответ будет получено сообщение типа Execution Report, которое парсится и записывается в структуру *FIX44::OrdMan.LastChangedOrder* (метод *LastChangedOrder()* возвращает копию), либо, если запрос на снятие заявки не был принят, будет получено сообщение типа *OrderCancelReject* и записана структура

*FIX44::OrdMan.LastRejectedCancel* (*LastRejectedCancel()* возвращает копию). Следует обратить внимание, что в памяти хранятся только последние принятые ответы, и в целом класс *OrderManager* необходимо доработать в соответствии с практическими нуждами.

Для корректного завершения сессии следует использовать метод *Logout*, который посылает соответствующее сообщение и останавливает слушающий поток.

Помимо прочего, у пользователя есть возможность логировать входящие и исходящие сообщения, для этого необходимо создать файл с именем FIX\_vars.inf в той же директории, в которой находится исполняемая программа, и задать переменные Log\_flag = true, Log\_path = [путь, в котором будет находиться лог]. После каждого параметра необходимо ставить «;»