Assignment: Market Data Processing

Introduction

Your task is to implement a handler for a (simple but not entirely unrealistic) market data feed from an imaginary stock exchange. The focus is on parsing and processing, so any actual network communication is omitted; instead, the input is provided as a file, and output is to stdout.

Language: C++. Modern C++ is encouraged. In terms of language features and standard library functionality, you may use everything supported by recent versions of popular compilers. You may also use the Boost library.

Platform: Linux or Windows (your choice) on x86-64. It is appreciated if the program can be built and run on both Linux and Windows.

Input

A sample input file is provided so you can test during development. Your final program will also be tested using other input files. You may assume that the input file is always small enough to fit into memory all at once.

The following sections describe the input format in a bottom-up manner.

Data Types

The feed uses native C++ data types to enable fast processing. All types use their native x86 representation, i.e. little endian.

Numeric Types

Numeric types used in the feed are: int8_t, int32_t, int64_t, double (64 bits).

Strings

There is only one type of string in the feed, the so-called symbol which uniquely identifies stock listings at the exchange. A symbol is an ASCII string consisting entirely of uppercase letters and decimal digits, the length is at least one and at most six.

In market data messages, symbols are always transmitted as exactly six bytes. If the actual symbol has less than six characters, it is padded with null bytes at the end.

Messages

Each message consists of a header, followed immediately by the body. The size of a message depends on its type. There is no padding between header and body, nor is there padding at the end of the message.

Header

The header consists of a single int8_t which identifies the message type.

Body

There are different message body structures for different message types. For parsing, the correct type must be chosen based on the message type indicated in the header.

Within a message body, fields appear in the order given in the definitions below. There is no padding between fields, nor is there padding at the end of the body.

Type 1: Best Bid/Ask

Messages of this type provide the current Best Bid and Best Ask prices for a given listing.

Type	Name	Description
symbol string	symbol	Identifies the listing to which this message is related.
double	best_bid	The current best bid price for the listing.
double	best_ask	The current best ask price for the listing.

Type 2: Trade

Messages of this type indicate that a trade took place for a given listing.

Type	Name	Description
symbol string	symbol	Identifies the listing to which this message is related.
int32_t	trade_id	Uniquely identifies this trade.
int64_t	timestamp	Time when the trade took place, in nanoseconds since the Unix epoch.
int32_t	quantity	The number of shares that were traded.
double	price	The price per share that was paid in this trade.

Type 3: Trade Cancellation

Messages of this type indicate that a trade has been reverted, e.g. if it has been ruled a mistrade by the stock exchange. When a trade has been cancelled, it is as if it had never existed.

Type	Name	Description
symbol string	symbol	Identifies the listing to which this message is related.
int32_t	trade_id	ID of the trade that has been cancelled.

Feed

The feed (i.e. input file) consists of a series of messages, with nothing else before, between or after the messages. The feed is in chronological order.

Expected Functionality

Part One

Your program must read and parse all messages in the input. Each message must be transformed into a native C++ struct and then printed to stdout in human-readable form.

Part Two

Once all input has been processed, the program must print this information for each listing it encountered in the input:

- latest value of best_bid
- latest value of best_ask
- per-share price paid in the last trade
- number of trades that took place
- total quantity traded (i.e. the sum of the quantities of all trades)
- total turnover (obtained by multiplying price and quantity for each trade, and then adding these up)

Deliverable

Submit your C++ source code. If any specific compiler settings are necessary to build it, these must be documented.

Evaluation Criteria

Primary criteria are:

- Completeness and correctness of functionality
- Readability and maintainability of the code

There are no particular rules regarding code style, simply keep in mind that your code will need to be read and understood by others.

Program speed and memory consumption are only of secondary concern – of course the program shouldn't waste resources gratuitously, but clarity of code is more important than saving a few bytes or microseconds.

Be prepared to discuss your code and the decisions you made during development.