Coding workshop: Looking at the data and modelling it

In this worksheet, we are going to familiarise ourselves with the order book data set and to look at how we can model the data using basic data types.

The order book data set

Here is a link to the dataset:

orderbook_data.csv

Download the CSV file and open it in a text editor, to see the raw data, or in a spreadsheet program, to see the data in a more easily manipulated form.

Think about what kind of data each of the columns contains. What range do the values fall into?

Numerical types in C++

Check out the textbook chapter 2. This covers the types available in C++. You will see that on pg 28, it shows two methods of declaring and initialising variables:

```
int aNumber {10};
```

and:

```
int aNumber = 10;
```

Both are equally valid, but the first one uses C++11 style braces. We'll see that style again later when we start creating objects.

Table 2.1 on page 31 shows all the integer data types. Table 2.4 on page 44 shows the floating-point data types. Table 2.5 on page 45 explains how much precision these representations have in terms of the decimal (mantissa) and the range (exponent).

Can you write out a set of variable declarations and assignments to cover the numerical columns in the dataset? Here is an example:

```
#include <iostream>

int main()
{
    double amount = 4.5;
}
```

Can you re-write your variable declarations using the C++11 style curly braces?

String data types

Some of the columns contain string data. Chapter 7 on p219 introduces the std::string data type. P223 includes a summary of the various ways you can create std::string objects in C++.

Let's declare some std::strings to store the data items in the CSV file. First off, include the string header:

```
#include <string>
```

Then in the main function, add some string declarations:

```
std::string product = "ETH/BTC";
```

Or better, in the C++11 style:

```
std::string product{"ETH/BTC"};
```

Write variable declarations for the other string types.

Categorical data types - enum class

One of the columns in the dataset only ever has one of two values, the order type. It can be "ask" or "bid". For this, we are going to define a new type using the enum class keywords.

The textbook discusses enum class in Chapter 3, p77, They use the example of days of the week. We are using the example of order types. First, we define our new type.

```
enum class OrderBookType {bid, ask};
```

Next, we can create a variable of this type:

```
OrderBookType orderType = OrderBookType::ask;
```

Trying setting the variable to different values of OrderBookType and printing them out. What values are used to actually represent these orderTypes?

Storing the data for multiple orders in a vector

Now let's see how we can store multiple orders in a vector.

Add this to the top of your cpp file, so you can use the std::vector class:

```
#include <vector>
```

Now in the main function, add several vectors, one for each column in the dataset. Here is a simple example program showing how to store one column in one vector:

```
#include <iostream>
#include <vector>
#include <string>

int main()
{
    std::vector<double> amounts;
    amounts.push_back(0.5);
    amounts.push_back(0.125);

    std::cout << "First row amount " << amounts[0] << std::endl;
    std::cout << "Second row amount " << amounts[1] << std::endl;

}
```

The textbook covers std::vector in Chapter 5, p169. Note the difference between initialising with curly braces and normal braces, covered on page 170.

Can you adapt the above program so it has five vectors, one for each column, each using an appropriate data type. Make sure you add your enum class definition so you can store the asks and bids.

Note that we are not reading the real data file in yet - we are just hard coding the values of the columns for now.

Write statements that print out a complete row of information.

Conclusion

In this worksheet, we have experimented with different data types, including numerical types, std::string and enum class. With these data types, we can represent all the data in the order book data set. We have also used std::vectors to store multiple rows of data, one vector per column.