Coding workshop: read a file in line by line and tokenise

In this worksheet, we will write a program that reads in a CSV file line by line, tokenises the values and converts them into appropriate data types. We will use exception handling to make this program robust.

Start with a blank main file

If you prefer, you can work on your existing project. If you are working on your existing project, you need to comment out everything in the main file so we can write a new program. Later, we will integrate our new code back into the main project.

Put some starter code into the main CPP file

Put this code into your main file:

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

std::vector<std::string> tokenise(std::string csvLine, char separator
)
{
std::vector<std::string> tokens;
signed int start, end;
std::string token;
    start = csvLine.find_first_not_of(separator, 0);
    do{
        end = csvLine.find_first_of(separator, start);
        if (start == csvLine.length() || start == end) break;
        if (end >= 0) token = csvLine.substr(start, end - start);
        else token = csvLine.substr(start, csvLine.length() - start);
        tokens.push_back(token);
    start = end + 1;
    }while(end > 0);

return tokens;
}


int main()
{
    std::string s = "one,two,three";
    std::vector<std::string> tokens = tokenise(s, ',');
```

```
    for (const std::string& t : tokens)
    {
        std::cout << t << std::endl;
    }
    return 0;
}
```

Run it and verify that it outputs the following:

```
one
two
three
```

Reading lines from a file

Now we are going to write the code to read lines in from a file. First of all, add this include, so you have access to std::ifstream:

```
#include <fstream>
```

Now let's try and open the CSV data file with the following main function:

```
int main()
{
    std::string csvFilename{"20200317.csv"};
    std::ifstream csvFile{csvFilename};
    std::string line;
    if (csvFile.is_open())
    {
        std::cout << "Opened file " << csvFilename << std::endl;
    }
    else
    {
        std::cout << "Problem opening file " << csvFilename << std::endl;
    }
    // don't forget to close it!
    csvFile.close();
    return 0;
}
```

You might have to adjust the contents of the csvFilename string, so it points to the place where you have put the CSV file. E.g. if you put it in a temp folder on your windows C drive:

```
std::string csvFilename{"C:\temp\20200317.csv"};
```

Confirm that your program can open the file.

Read a line from the file

Now we know the program can at least open the data file, let's read a line from it:

```
std::string line;
std::getline(csvFile, line);
std::cout << "Read line " << line << std::endl;
```

That should print out the first line of the CSV file, something like this:

```
Read line 2020/03/17 17:01:24.884492,ETH/BTC,bid,0.02187308,7.44564869
```

Your exact output might vary, but it should have the same set of columns.

Tokenise the line

Now we have a line read in from the file, we can use the tokenise function to break it up on the commas.

```
std::vector<std::string> tokens = tokenise(line, ',');
std::cout << "Read " << tokens.size() << " tokens " << std::endl;
```

It should read five tokens from the first line of the CSV file.

Can you write a loop to print out all the tokens to check what has been read in?

Iterate over lines in the file

Now we have read in a single line, it is time to read in all the lines.

```
std::string line;
while(getline(csvFile, line))
{
    std::vector<std::string> tokens = tokenise(line, ',');
    std::cout << "Read " << tokens.size() << " tokens " << std::endl;
}
```

That should print a lot of output. Can you write some code that counts how many lines there are and prints out that number?

Convert the tokens into appropriate data types

Now we are parsing all the lines from the file and tokenising each row into columns, we need to convert the data into the appropriate types.

Basic check: correct number of tokens

First off, let's add a check to see if we have the correct number of tokens before proceeding. This should do the trick if you place it inside the while loop:

```
if (tokens.size() != 5) continue;
```

Convert tokens into correct data types

Now to convert the tokens into appropriate data types for an OrderBookEntry. The tokens are as follows:

- 2020/03/17 17:01:24.884492 - this needs to be a std::string
- ETH/BTC - std::string
- bid - OrderBookType
- 0.02187308 - double
- 7.44564869 - double

Several fields are strings, and tokenise gives us strings, so not much to do there.

Two fields are doubles. Here is some code to convert the 4th token into a double:

```
double price = std::stod(tokens[3]);
```

Handle exceptions on string conversions

The double conversion code can throw an exception. Try this:

```
double price = std::stod("I am not a double");
```

That should crash your program. You can prevent the crash with this:

```
try
{
    double price = std::stod("I am not a double");
}
catch (const std::exception& e)
{
    continue;
}
```

This prevents the crash by catching the exception thrown by the double conversion and skipping onto the next iteration.

Can you set up your code, so it reads the two numerical tokens into local double variableS?

For further reading about exceptions, see this link https://www.cplusplus.com/doc/tutorial/exceptions/ or (degree students) the textbook covers exceptions in Chapter 15, with a try-catch example on p577.

Next steps

The next step is to deal with the OrderBookType and then to convert the data we have collected into an OrderBookEntry object. We will cover that in the following worksheet.

Conclusion

In this worksheet, we have read a CSV data file into our program and converted the data in the file into appropriate data types. We have also used exception handling to deal with invalid data.