Task 9 Lab: File Input and Output

Summary

The aim of this lab is to improve (or refresh) your file input/output skills in C++. The lab in divided into three parts.

We take a quick look at reading and writing to binary files in Part 1, then read from a simple text file (splitting lines) in Part 2, and finally make use of the common JSON file format in Part 3. There are many other file I/O related topics to explore, and some extension ideas are included that you may wish to explore in and include as "extension" work in your final portfolio. There are questions in green throughout that you should respond to in a submission to canvas.

Task Description

1. **Part 1: Binary file Output / Input**
    a. Write a C++ program that:
        i. Has a basic struct or class ("compound type") that has at least 3 simple variables: char, int, float;
        ii. Create an instance of your compound type
        iii. Set the value of each variable in your instance, to something other than a zero value.
        iv. Write a reusable routine (function) to print/show the values to screen.
    b. Extend the program to
        i. Write the values to a text file (as text!)
        ii. Read the values from a text file
        iii. Display the values to screen
    c. Further extend the program to:
        i. Open a binary file in "write" mode (such as "test1.bin"), then
        ii. Write the three different values to the binary file, and finally
        iii. Close the file.

    Q: There are different file open modes: What are they?
    Q: What happens if you don't "close" the file? Is it something we need to worry about?

    d. Compile and run.
    e. Confirm that a file was created.
    f. Using an appropriate viewing program (hex viewer or similar), inspect the file and confirm that something is being written to the file.

    Q: How many bytes are in the file? Is this expected based on the size of the variable types?

    g. Further extend the program to :
        i. Open the file you saved earlier, in binary read (only) mode,
        ii. Read the values from the file (in correct order), then
        iii. Display the values to screen

2. **Part 2: Simple Text File Input with Split**
    a. With a text editor, create and save a simple text file (such as "test2.txt") that contains three lines similar to the following, with the last line being the actual line of useful data and using a full-colon separator character:

```
# This is a comment line. The next line is deliberately blank.

12:string value:13.57
```

    b. Create a new program that is able to
        i. Open the file (text mode, read only),
        ii. Print each line to screen, one at a time
    c. Modify your code so that it can (**required**!)
        i. Ignore any blank line ("strip" whitespace first?),
        ii. Ignore a line that starts with the hash "#" character (treats it as a single line comment),
        iii. Splits all other lines by ":", checking that is has the appropriate number of tokens, and

iv. Prints each split line to screen, one token at a time.

3. **Part 3: Reading JSON Files**
   There are many good reasons for using established file formats, and to also use well-tested code. Now we get some experience with the JSON format in C++ using a popular JSON library.
   a. With a text editor create a basic JSON text file (such as "test3.json") with the following content (or similar) for player character details:

```
{ "exp": 12345, "health": 100, "jsonType": "player", "level": 42, "name": "Fred",
"uuid": "123456" }
```

   b. Go to https://github.com/nlohmann/json, read and download the JSON library. We suggest the "Trivial integration" version (search the README.md) which is a single hpp file in plain C++11, but it's up to you. You could use another JSON library if you want, but this one is popular and well designed.
   c. Create a simple JSON test program in C++ that, using the JSON library, opens your JSON file and then print the contents to screen. (Simple to write – but might take you a bit of effort. Use similar steps to the ones earlier in this lab.)

Expected Output

Repository

1. Code for Part 1
2. Code for Part 2
3. Code for Part 3

Canvas

1. Responses to the questions in green
2. Commit logs