iomanip

- std::setprecision(int) – sets number of significant digits for decimal numbers when using scientific notation; set number of places to the right of the decimal for fixed decimal real numbers

iomanip

- std::setiosflags – sets input/output stream flags for formatting
- std::ios::showpoint – display decimal point for floating point values, even if zeros follow it
- std::ios::fixed – format real numbers in fixed decimal format

iomanip

- std::setw(int) – set the width of the field for the very next output

Functions – Part III

Separate Compilation

- In order to create truly reusable functions, the function implementations need to be written in a file that does not contain a main function.
- Placing the function’s prototype in a header file (.h) allows its inclusion for use by other C++ files.

Separate Compilation

- Header files may be included by multiple files in a project.
- To ensure that the definitions within the header file are set exactly once, we'll surround our code with the following header guards:

```
#ifndef PROJECT_PATH_FILENAME_H_
#define PROJECT_PATH_FILENAME_H_
// prototypes here
#endif  // PROJECT_PATH_FILENAME_H_
```

Separate Compilation

- Header files (.h) cannot be compiled
- We can place the function's implementation in a source file (.cc) that can be compiled
- In the source file, we'll include the header file that contains the function prototype

Separate Compilation

- Compiling the source file (g++ -c) creates the function's object code
- The object code can be linked as needed when building an executable (g++ *file1.o … filen.o*)

Unit Testing

- The smallest units of code should be individually, rigorously tested prior to use in production.
- A driver program is a source file with a main whose purpose is to test code. We will write drivers to test our individual functions.