PPP: Chapter Review Questions

# Chapter 2 – Hello, World!

1. The purpose of the “Hello, World!” program is to introduce the key concepts of a new programming language (the tool) to any new programmer. This includes using an IDE, compiling, linking, and running the program.
2. There are four parts to a function: the return type (e.g. *int*); the function name (e.g. *main()*); the parameters (e.g. **int argc**); and the body (e.g. **{cout << “Hello, World!\n”;}**)**.**
3. The **main()** function must appear in every C++ program.
4. The purpose of the **return 0** statement is to indicate to the system that the program exited without failure.
5. The purpose of the compiler is to take the human-readable *source code* and ‘translate’ it to machine readable *machine code.*
6. The purpose of the **#include** directive is to ‘include’ another source code file (a header file).
7. The **.h** file suffix at the end of a file name in C++ signifies that the file is a *header* file.
8. The linker takes the files of object code created by the compiler and ‘links’ them together into an executable program.
9. The difference between a source code file and an object code file is: a source code file is human-readable, whereas an object file is machine-readable.
10. An Integrated Development Environment (IDE) is a graphical software program that makes it easier to write, compile, link, and debug programs.
11. It is necessary to practice programming as programming is a practical activity and one can only improve from experience.

# Chapter 3 – Objects, Types, and Values

1. A *prompt* is a message or character that indicates to the user that they must enter some information.
2. The input operator **>>** is used to read into a variable.
3. The following lines of code ask the user to input a value into the variable **number**  
   **cout << “Please enter a number\n>”;**  
   **int num;**  
   **cin >> num;**
4. The **\n** character is called the new line character. It outputs a new line (line break).
5. Whitespace terminates input into a string.
6. Whitespace terminates or a new line character input into an int.
7. The following code  
   **cout << “Hello, ”;**  
   **cout << first\_name;**  
   **cout << “!\n”**  
   could be written as  
   **cout << “Hello, “ << first\_name << “!\n”;**
8. An *object* is some memory that holds a value of a given type.
9. A literal is a character or number that doesn’t mean anything else. String literals are denoted by “” (e.g. “Hello!”); single character literals are denoted by ‘’ (e.g. ‘a’); integers are typed as numbers (e.g. 1); and doubles are denoted using a decimal point (e.g. 4.5).
10. There are 4 types of literal: string, character (char), integer (int), and a real numbers (a double).
11. A variable is a named object.
12. Typical sizes of built-in types:  
    **char** 1 byte (8 bits)  
    **int** 4 bytes (32 bits)  
    **double** 8 bytes (64 bits)
13. Bits and bytes are used to measure the size of small entities in memory, such as **int**s and **string**s.
14. The assignment (and initialisation) operator (**=**) assigns a new value to a variable, whereas the equality operator (**==**) compares the values of two variables and returns **true** if they are the same.
15. A definition is a declaration that sets aside memory for an object
16. Initialisation is the declaration and immediate assignment of a variable: it sets the *initial* value. Universal and uniform initialisation notation (**{**value**}**) should be used to ensure type safety. On the contrary, assignment changes the value of an already declared variable: it gives it a *new* value.
17. String concatenation is the joining together of one or more **string**s. The syntax is **string\_1 + string\_2** or **string\_1 + “literal value”**.
18. The following are examples of legal and illegal names:  
    This\_little\_pig legal  
    latest thing illegal (spaces aren’t allowed)  
    MiniMineMine legal  
    This\_1\_is fine illegal (spaces aren’t allowed)  
    the\_$12\_method illegal (non-alphanumeric characters aren’t allowed)  
    number legal  
    2\_For\_1\_special illegal (names must begin with letters)  
    \_this\_is\_ok legal (but names shouldn’t start with an underscore)  
    correct? illegal (non-alphanumeric characters aren’t allowed)
19. These are five examples of legal names that you shouldn’t choose because they are likely to cause confusion:  
    WVB  
    \_size  
    a\_very\_long\_variable\_name  
    foo, f00  
    string
20. Good rules for choosing names:  
    keep them short and meaningful  
    don’t begin them with underscores  
    use underscores to separate words  
    don’t use names that could be easily confused (e.g. foo and f00)  
    don’t use names that are used in the standard library  
    use a capital letter when defining a new type
21. Type safety is when all objects are accessed through their respective types or are assigned to types that don’t cause a narrowing conversion (where data might be lost). It is important as it can cause information to be lost.
22. Conversion from a **double** to an **int** can be a bad thing as information after the decimal point of the **double** is truncated: it’s lost.
23. Safe conversion rule: a conversion is sage if the lvalue is stored with more bytes of memory than the rvalue.

# Chapter 4 - Computation