This document contains all the non-coding exercises of Chapter 2 of C++ Primer.

Exercise 2.1

The different integer types vary in their minimum bit sizes and so their minimum value range:

`short` 16 bits gives a value range of -32768 – 32767.

`int` 16 bits gives a value range of -32768 – 32767.

`long` 32 bits gives a value range of -2147483648 – 2147483647.

`long long` 64 bits vies a value range of -9223372036854775808 – 9223372036854775807.

`unsigned` types do not include negative values, that is, values ≥ 0, for example if we consider `unsigned int` the value range is 0 – 65536. Of course a `signed` type takes negative values.

Lastly, both `float` and `double` represent decimal precision values. The difference is in the accuracy of precision, a `float` is precise to 6 significant digits and `double` is precise to 10 significant digits.

Exercise 2.2

They should all be `double` as they all involved decimal figures.

Exercise 2.3

Assuming the machine is 32 bits for integers the outputs are (in order):

- 32
- 4294967264
- 32
- -32
- 0
- 0

Exercise 2.4

See ex24.cpp.

Exercise 2.5

(a) In order: character literal; wide character literal; string literal; wide character string literal.

(b) In order: integer; unsigned integer; long integer; unsigned long integer; octal integer; hexadecimal integer.

(c) In order: double; float; long double.

(d) In order: integer; unsigned integer; double; double.

Exercise 2.6

In the first statement the definitions use integer literals so the values are what we expect, that is, `month = 9` and `day = 7`. In the second statement we try to define integers using octals however there is an error as 09 is not an octal and will throw an error.

Exercise 2.7

(a) This is a string literal which represents "Who goes with Fergus?"

(b) The value is 3.1 and is a long double.

(c) The value is 1024 and is a float.

(d) The value is 3.14 and is a long double.

Exercise 2.8

See ex28.cpp.

Exercise 2.9

(a) Cannot use variable before it has been declared. To correct:
```
int input_value;
std::cin >> input_value;
```

(b) Cannot use double in list initialisation since narrowing value may lose data. To correct:
```
double i = { 3.14 };
```

(c) Cannot use variable to initialise another variable before it has been initialised. To correct:
```
double wage = 9999.99;
double salary = wage;
```

(d) This will initialise `i` to be an `int` but it will truncate it's value from 3.14 to 3.

Exercise 2.10

- `global_str` will be an empty string.
- `global_int` will be 0.
- `local_int` will be undefined.
- `local_str` will be an empty string.

Exercise 2.11

(a) This is a definition since the variable is initialised and so memory is allocated for the variable.

(b) This is a definition as the variable is defined locally and so memory is allocated for it.

(c) This is a declaration of a variable as it specifies the variable is external without initialising its value.

Exercise 2.12

`double` is a reserved C++ keyword and so is invalid, `catch-22` does not include valid characters in an identifier and so is invalid and lastly, `1_or_2` is invalid as it begins with a number.

Exercise 2.13

Since `i` is defined globally and locally, within the block scope its value is that derived from initialisation within the block. So `j` will have value 100.

Exercise 2.14

The program is legal since variables can be redefined in an inner scope (here this is the `for` loop). The program prints `100 45` since the value of `sum` is the sum of the numbers 0–9.