Exercise 1.1

I'm using GCC as my compiler and Debian as my OS. Here are the file extensions GCC uses for C++ source files:

```
.cc
.cp
.cxx
.cpp
.CPP
.c++
.C
```

I wrote the first program in the `prog1.cpp` file. I compile it with `$ g++ -o prog1 prog1.cpp`. I execute the compiled program with `$ ./prog1`, nothing happen as expected. I obtain the status with the command `$ echo $?` and get `0` as expected.

```cpp
// prog1.cpp

int main()
{
    return 0;
}
```

Exercise 1.2

We change the return value to `-1` and save the modified program in the file `prog2.cpp`. We compile it and run it and nothing happen as in the first program. The status value I get after the command `$ echo $?` is `255`. It seems any value returned by `main` is stored modulo $256 = 2^8$ on my system.

```cpp
// prog2.cpp

int main()
{
    return -1;
}
```

Exercise 1.3

```cpp
// hello_world.cpp

#include <iostream>

int main()
{
```

```
    std::cout << "Hello, World" << std::endl;
    return 0;
}
```

Exercise 1.4

```
// product.cpp

#include <iostream>

int main()
{
    std::cout << "Enter two numbers:" << std::endl;
    int v1 = 0, v2 = 0;
    std::cin >> v1 >> v2;
    std::cout << "The product of " << v1 << " and " << v2
              << " is " << v1 * v2 << std::endl;
    return 0;
}
```

Exercise 1.5

```
// sum_separated_print_statements.cpp

#include <iostream>

int main()
{
    std::cout << "Enter two numbers:";
    std::cout << std::endl;
    int v1 = 0, v2 = 0;
    std::cin >> v1 >> v2;
    std::cout << "The sum of ";
    std::cout << v1;
    std::cout << " and ";
    std::cout << v2;
    std::cout << " is ";
    std::cout << v1 + v2;
    std::cout << std::endl;
    return 0;
}
```

Exercise 1.6

This program fragment is not legal because the first `<<` operator on the second line has no left operand. For information, here is the error message from my compiler (GCC):

```
illegal_program_fragment.cpp: In function ‘int main()’:
illegal_program_fragment.cpp:11:15: error: expected primary-expression before ‘<<’ token
               << " and " << v2;
               ^~
illegal_program_fragment.cpp:12:15: error: expected primary-expression before ‘<<’ token
               << " is " << v1 + v2 << std::endl;
               ^~
```

To fix this program fragment we only need to remove the first two semicolons:

```
std::cout << "The sum of " << v1
          << " and " << v2
          << " is " << v1 + v2 << std::endl;
```

Exercise 1.7

```
// incorrectly_nested_comments.cpp

#include <iostream>

int main()
{
    /*
      int main() {
      return 0; /* no error status */
}
*/
return 0;
}
```

Here is my compiler message when trying to compile this illegal program:

```
incorrectly_nested_comments.cpp:9:6: error: expected unqualified-id before ‘/’ token
     */
      ^
incorrectly_nested_comments.cpp:11:1: error: expected declaration before ‘}’ token
 }
 ^
```

Exercise 1.8

The two first statements are legal:

```
std::cout << "/*"; // legal, "/*" is treated as a string literal
std::cout << "*/"; // legal, "/*" is also treated as a string literal
```

Comment pairs cannot nest, as a result the third statement is equivalent to:

```
std::cout << " */;
```

And this statement is not legal as the second operand of << makes no sense.

Using the same property of comment pairs, the last statement is equivalent to:

```
std::cout << " /* ";
```

And this statement is legal, the second operand is a valid string literal.

Exercise 1.9

```
// sum_50_to_100.cpp

#include <iostream>

int main()
{
    int sum = 0, val = 50;
    while (val <= 100) {
        sum += val;
        ++val;
    }
    std::cout << "Sum of 50 to 100 inclusive is "
              << sum << std::endl;
    return 0;
}
```

Exercise 1.10

```
// count_ten_to_zero.cpp

#include <iostream>

int main()
{
    int n = 10;
    while (n >= 0) {
        std::cout << n << std::endl;
```

```
        --n;
    }
    return 0;
}
```

Exercise 1.11

```
// print_range.cpp

#include <iostream>

int main()
{
    std::cout << "Enter two numbers:" << std::endl;
    int v1 = 0, v2 = 0;
    std::cin >> v1 >> v2;
    while (v1 <= v2) {
        std::cout << v1 << std::endl;
        ++v1;
    }
    return 0;
}
```

Exercise 1.12

This `for` loop adds the number from -100 to 100 to the variable `sum`. The final value of `sum` is 0.

Exercise 1.13

exercise 1.9 with `for` loop:

```
// sum_50_to_100_with_for.cpp

#include <iostream>

int main()
{
    int sum = 0;
    for (int val = 50; val <= 100; ++val)
        sum += val;
    std::cout << "Sum of 50 to 100 inclusive is "
              << sum << std::endl;
```

```
    return 0;
}
```

exercise 1.10 with `for` loop:

```
// count_ten_to_zero_with_for.cpp

#include <iostream>

int main()
{
    for (int n = 10; n >= 0; --n)
        std::cout << n << std::endl;
    return 0;
}
```

exercise 1.11 with `for` loop

```
// print_range_with_for.cpp

#include <iostream>

int main()
{
    std::cout << "Enter two numbers:" << std::endl;
    int v1 = 0, v2 = 0;
    std::cin >> v1 >> v2;
    for (int v = v1; v <= v2; ++v)
        std::cout << v << std::endl;
    return 0;
}
```

Exercise 1.14

Loops that used a `for` are generally more conscise especially when we are itterating over a range of numbers. `for` loops also make it clear that a variable is only used as a loop index.

Exercise 1.15

Syntax error

```
// syntax_error.cpp

#include <iostream>
```

```cpp
int main()
{
    // error: missing quote
    std::cout << "Hello, World << std::endl;
    return 0;
}
```

Error message:

```
syntax_error.cpp:8:18: warning: missing terminating " character
     std::cout << "Hello, World << std::endl;
                  ^
syntax_error.cpp:8:18: error: missing terminating " character
     std::cout << "Hello, World << std::endl;
                  ^~~~~~~~~~~~~~~~~~~~~~~~~~
syntax_error.cpp: In function ‘int main()’:
syntax_error.cpp:9:5: error: expected primary-expression before ‘return’
     return 0;
     ^~~~~~
```

Type error

```cpp
// type_error.cpp

int main()
{
    // error: returned value should be of int type
    return "0";
}
```

The error message from my compiler:

```
type_error.cpp: In function ‘int main()’:
type_error.cpp:6:12: error: invalid conversion from ‘const char*’ to ‘int’ [-fpermissive]
     return "0";
            ^~~
```

Declaration error

```cpp
// declaration_error.cpp

#include <iostream>

int main()
{
    std::cout << "Hello, World" << endl;
```

```
    return 0;
}
```

Error message associated:

```
declaration_error.cpp:7:36: error: ‘endl’ was not declared in this scope
     std::cout << "Hello, World" << endl;
                                  ^~~~
declaration_error.cpp:7:36: note: suggested alternative:
In file included from /usr/include/c++/8/iostream:39,
                 from declaration_error.cpp:3:
/usr/include/c++/8/ostream:590:5: note:   ‘std::endl’
     endl(basic_ostream<_CharT, _Traits>& __os)
     ^~~~
```

Exercise 1.16

```
// sum_integers_cin.cpp

#include <iostream>

int main()
{
    int sum = 0, val = 0;
    std::cout << "Enter numbers:" << std::endl;
    while (std::cin >> val)
        sum += val;
    std:cout << "The sum of these numbers is " << sum << std::endl;
    return 0;
}
```

Exercise 1.17

If the input values are all equal, the program will print one line with number of times we entered the value. If there are no duplicated values the program will output one line for each number entered with a count of 1 each time.

Exercise 1.18

This exercise verify our answer from the previous exercise.

If the input is `2 2 2 2 2 2` the program prints `2 occurs 6 times`.

If the input is `1 2 3 4 5 6` the program prints:

```
1 occurs 1 times
```

```
2 occurs 1 times
3 occurs 1 times
4 occurs 1 times
5 occurs 1 times
6 occurs 1 times
```

Exercise 1.19

```cpp
// print_range_revised.cpp

#include <iostream>

int main()
{
    std::cout << "Enter two numbers:" << std::endl;
    int v1 = 0, v2 = 0;
    std::cin >> v1 >> v2;
    if (v1 < v2) {
        while (v1 <= v2) {
            std::cout << v1 << std::endl;
            ++v1;
        }
    } else {
        while (v1 >= v2) {
            std::cout << v1 << std::endl;
            --v1;
        }
    }
    return 0;
}
```

Exercise 1.20

```cpp
// book_transactions_print.cpp

#include <iostream>
#include "Sales_item.hpp"

int main()
{
    std::cout << "Enter book sales transactions:" << std::endl;
    Sales_item book;
    while (std::cin >> book)
        std::cout << book << std::endl;
```

```
    return 0;
}
```

Exercise 1.21

```
// sum_two_sales_item.cpp

#include <iostream>
#include "Sales_item.hpp"

int main()
{
    Sales_item item1, item2;

    std::cin >> item1 >> item2;
    std::cout << item1 + item2 << std::endl;

    return 0;
}
```

Exercise 1.22

```
// sum_sales_item.cpp

#include <iostream>
#include "Sales_item.hpp"

int main()
{
    Sales_item sum_item, item;

    if (std::cin >> sum_item) {
        while (std::cin >> item)
            sum_item += item;
        std::cout << sum_item << std::endl;
        return 0;
    }
    return -1;
}
```

Exercise 1.23

```
// consecutive_count_ISBN.cpp
```

```cpp
#include <iostream>
#include "Sales_item.hpp"

int main()
{
    Sales_item curr_item, item;
    if (std::cin >> curr_item) {
        int cnt = 1;
        while (std::cin >> item) {
            if (item.isbn() == curr_item.isbn())
                ++cnt;
            else {
                std::cout << curr_item.isbn() << " ISBN occurs "
                          << cnt << " times" << std::endl;
                curr_item = item;
                cnt = 1;
            }
        }
        std::cout << curr_item.isbn() << " ISBN occurs "
                  << cnt << " times" << std::endl;
    }
    return 0;
}
```

Exercise 1.24

Here is my transactions used as input for the previous program:

```
0-201-78345-X 3 20.00
0-201-78345-X 2 25.00
0-201-70353-X 4 24.99
0-201-70353-X 2 10.99
0-201-70353-X 1 40.00
```

The program outputs:

```
0-201-78345-X ISBN occurs 2 times
0-201-70353-X ISBN occurs 3 times
```

Exercise 1.25

#+INCLUDE "bookstore.cpp" src cpp

Using the same input as in the previous exercise we get:

```
0-201-78345-X 5 110 22
0-201-70353-X 7 161.94 23.1343
```