C++, week 1

Nils Brouwer & Niek Naber

September 2018

1 Exercise 1: Hello World

1.1 Program Code

```
1 #include <iostream>
2
3 main(int argc, char *argv[])
4 {
5     std::cout << "Hello World\n";                    //print hello world
6 }
```

1.2 Commands that we use

- g++ -c hello.cc
- g++ -s -o hello hello.cc
- ./hello

1.3 Programs Output

The programs output is:

```
1 Hello World
```

1.4 All Files

File Name	Size	Description
hello.cc	124	This is the source file. This file contains the code in plain text that the programmer wrote.
hello.o	2592	This is the object file. This file is compiled by the compiler, but it is not yet linked.
hello	6120	This is the executable. This file is compiled and linked, so it is ready to be executed.
iostream	2695	This is the standard library of c++. It contains input and output definitions.

2 Exercise 2: Basic Exercise

- A declaration introduces an identifier and describes only its type, name and arguments, while a definition specifies the interpretation and attributes of these identifier for the compiler; it implements the body of the function.
- A header file is a separate file which holds declarations for functions which are defined in other files.
- Header files are read by the compiler in the compilation phase of the program. Libraries are linked during the linking phase of the program.
- Yes, a library can be a single object file, but most of the times it is a package of multiple object files.
- An object file itself is not executable. It first has to go through the linker so that the object file and other libraries declared in the header files can be linked. If the object file would be executable, functions from libraries would not be useful yet.

3 Exercise 3: Is the code snippet valid?

1. Valid.
2. Valid.
3. Not valid. You could fix this by adding a number after return, so that the program returns a value. You could also remove the whole line since a program return zero by default if the return statement is not present.
4. Valid.
5. The last argument of argv is a null pointer. Since null is evaluated to be equal to zero in c++, this statement returns true.
6. It is possible to use exit(0) to terminate a program from deep inside its code. This could however result in memory leaks and other difficulties, so the use of exit(0) is not desirable. It is better to return a well designed exit status from main.
7. Yes, you can leave out the "typedef", because this is C heritage. In C++ enums can be accessed directly as types by their names.
8. Yes.

4 Exercise 4: Raw String Literals

We are not able to use the R(...)" because in that case the delimiter becomes)" which is present in the snippet we want to print. Replacing this with R"R(...)R" specifies the delimiter to)R" which is not present in the snippet itself. So you should make sure that the delimiter is not present in your text (which in the 'default' case is)"): you should change the delimiter to a random other string which is not present.

Program: RSL.cc

```
#include <iostream>

char const snippet[]=                    // define snippet as raw string variable
R"R(^\s+Encryption key:(\w+)
^\s+Quality=(\d+)
^\s+E?SSID:" ([[:print:]]+)"
^\s+ssid=" ([[:print:]]+)"
)R";

int main(int argc, char const *argv[])
{
    std::cout << snippet;                // print snippet
}
```

5 Excercise 5: Escape Sequences

An escape sequence is a sequence of characters, beginning with a backslash: \. This sequence of characters represents another character or character sequence than the characters of the sequence itself.

Table 1: The standard escape sequences in C++ and their description

Escape sequences	Description
\'	inserts a single quote
\"	insert a double quote
\?	inserts a question mark
\\	inserts a backslash
\a	alert: plays a sound when executed.
\b	backspace: deletes a single character to the left.
\f	form feed: inserts a blank page when printed.
\n	newline
\r	carriage return: moves the cursor to the begin of the line.
\t	horizontal tab: which is equal to 8 spaces.
\v	vertical tab
\nnn	inserts a character by its arbitrary octal value
\xnn	inserts a character by its arbitrary hexadecimal value
\unnnn	inserts the utf-8 character that is associated with the 'nnnn' value.
\unnnnnnnn	inserts the utf-16 character that is associated with the 'nnnnnnnn' value.

If you would use a non-existent escape sequence in your program, the compiler will give you a warning that the escape sequence is unknown. If you then execute the program, the sequence will just be printed literally, but without the backslash.

6 6: Get some experience in using various operators

Program: operator.cc

```
#include <iostream>
using namespace std;

main(int argc, char *argv[])
{
    size_t value;                           // declare value to be an integer
    cin >> value;                           // ask for user input and store in value

    // Test if value modulo 2 is zero, if true value is even.
    cout << (value % 2 == 0 ? "even" : "odd") << '\n';

    // if value + 1 divided by 2 is the same as original value divided by 2, number is even
    cout << (value / 2 == (value + 1) / 2 ? "even" : "odd") << '\n';

    // shift bits to the right, then to the left. If value doesnt change, the number is even
    .
    cout << (value >> 1 << 1 == value ? "even" : "odd") << '\n';

    // check if value still the same after bitwise or with int value 1.
    cout << ((value | 1) == value ? "odd" : "even") << '\n';

    // Check if least significant bit is 0, if true value is even.
    cout << ((value & 1) == 0 ? "even" : "odd") << '\n';
}
```

7 7: Learn to use bit-wise operators

Program: bitwise.cc

```
#include <iostream>
using namespace std;

main(int argc, char const *argv[])
{
    int value;                  // declare value to be an integer
    cin >> value;               // ask for user input and store in value

    // power of 2 tested by bitwise and, then printed
    cout << ((value & (value - 1)) == 0 ? "The value x is an exact power of two" : "the
    value x is not an exact power of two") << '\n';
}
```

8 8: Understand piping and redirection

Piping is used to write output to another program or utility whereas redirection is used to write output to another file or stream. This becomes more clear when discussing the examples given in the assignment. The first example is a form of redirecting: program ¡ in ¿ out. In this case *in* will be passed to the a stream of of the executable *program*. This program will write its output in *out*. The second example is a form of piping: program — less. In this case *program* will send its output to the program *less*. *less* can then do something with the just received output of *program*.

9 9: Simplify bit-field operations

Program: bitFields.cc

```
1 #include <iostream>
2 using namespace std;
3
4 union {
5     size_t value : 40;
6 } val;
7
8 main(int argc, char *argv[])
9 {
10
11     val.value = 0;
12     val.value |= (7 << 1);
13     val.value |= (15 << 4);
14     val.value |= (10 << 8);
15     val.value |= (6 << 13);
16     val.value |= (7 << 17);
17     val.value |= (15 << 21);
18     val.value |= (15 << 25);
19     val.value |= (7 << 29);
20     val.value |= ((size_t) 0 << 33);
21     val.value |= ((size_t) 3 << 38);
22
23     cout << hex << val.value << '\n';
24 }
```

Output of the program

```
1 ffffeecafe
```

10 10: Manipulate bits inside variables

Program: nibble.cc

```
1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 int main(int argc, char *argv[])
7 {
8     size_t value = stoul(argv[1], 0, 16);   // initialize hexadecimal value
9     size_t nibble = stoul(argv[2]);         // nibble to replace
10    size_t replacement = stoul(argv[3]) % 16;   // new nibble (= 0 .. 15)
11
12    size_t orgVal = value;              // make copy of value for later
13    size_t remain = 0;                  //initialize remain at zero
14
15    value >>= ((nibble+1)*4);           // shift value so that last bits fall of
16    value <<= 4;                        // get 4 bits back, initialized at zero
17    value |= replacement;               // take bitwise or of replacement and value
18    value <<= nibble * 4;               // shift value back to original length
19
20    for(int i = 0; i < (nibble); i++)           // construct the remainder
21    {
22        remain |= ((orgVal & 15) << (i * 4));   //get last 4 bits and shift to right place
23        orgVal >>= 4;                           // shift the original value to next 4 bits
24    }
25
```

```
26      value |= remain;                    // bitwise or of value and the remainder that has
        fallen of
27
28      cout << hex << value << '\n';       // show the new value
29  }
```