Programming in C/C++
Week 1: Assignment 1
Tjalling Otter & Emiel Krol

********** Begin Code **********

```
// Programming in C/C++
// Week 1: Assignment 1
// Tjalling Otter & Emiel Krol

#include <iostream>
int main()
{
  std::cout << "Hello World" << '\n'; // Printing 'Hello World' to the console
}
```

********** End Code **********

Commands for compiling and linking

Compiling

```
g++ -Wall --std=c++17 -c hello.cc
```

Linking

```
g++ -Wall --std=c++17 -o hello hello.o
```

Short descriptions of object file and executable

Object file

Compiled version of the source code (contained in hello.cc). In some sense it can be considered to be an intermediate file, as it cannot run yet but is machine code.

Executable

Compiled and linked version of the source code, and is executable.

The output of the program is as follows

```
Hello World
```

File sizes

hello.cc (source file)	153 bytes
hello.o (object file)	2696 bytes
hello (executable)	6208 bytes
iostream (library)	2695 bytes

Programming in C/C++
Week 1: Assignment 2
Tjalling Otter & Emiel Krol

1

Difference between a declaration and a definition

A declaration is merely the announcement of the existence of a symbol, like a function, variable, etc, and the type thereof (e.g. int). A definition actually assigns meaning to those symbols, such as what the function does or which value the variable is assigned.

Header files

Header files are used to import, as it were, often-used functions to be able to use them in one's own program(s). As such, it precludes having to write all this functionality ourselves, and standardizes the usage of some functionality.

Header files and libraries

Header files are inserted as if they were copy pasted from their source into the source code at the place of the include statement. As such, it is used at the moment of compilation. Libraries, however, are included at the linking phase, i.e. when the executable is generated.

Libraries

A library is not an object module since the object module is created by the compiler. The object module contains the code including calls to functions but it does not know what to do with the functions yet. This information is in the libraries. In the linking phase the compiler links the object module to the library such so that functions can be used.

Why is an object module obtained after compiling a source containing int main() not an executable program?

Before it is linked it is "relocatable object code": the object's place in the executable program is not yet known. It first needs to be linked to the objects and to the libraries.

Programming in C/C++
Week 1: Assignment 3
Tjalling Otter & Emiel Krol

1. Empty program
   Yes, it is valid. It produces no visible result.

2. Nonstandard main function
   Yes, it is valid, however it is nonstandard and the third parameter should be avoided.

3. Empty return value
   No, it is not valid. The snippet says to return something, but not what. Hence, the program can be fixed by adding either a 0 or a 1. However, the line can also be removed entirely, as the program will already return a 0 by default.

4. Comparing
   Yes, when adding a semicolon to the end, the snippet is valid. However, it does 'do' anything. Yes, the size of the character c is 1, but no function is dependent on this fact. It could, for example, be used in an if-statement.

5. Argument array
   No, however the program will run. The value of argc (i.e. 1), representing the number of arguments passed to the main function, is used as the array position in the array argv[]. Since there is only one argument, and the array starts at position zero, this should be invalid or undefined.

6. End program
   It will work, however using such a function is not recommended. Instead, reordering one's code to lead to a natural ending (i.e. the end of the main function) is preferred.

7. enum definition
   Yes, that is valid. It would work either way, but it is not required to preface an enum with typedef.

8. Index evaluation sequence
   Yes, this refers to the fourth position within the third position of the array argv. It is evaluated starting from the variable itself, moving outward.

Programming in C/C++
Week 1: Assignment 4
Tjalling Otter & Emiel Krol

0

********** Begin Code **********

```
// Programming in C/C++
// Week 1: Assignment 4
// Tjalling Otter &  Emiel Krol

char const NTBs[]=

R"R(
^\\s+Encryption key:(\\w+)
^\\s+Quality=(\\d+)
^\\s+E?SSID:\"([[:print:]]+)\"
^\\s+ssid=\"([[:print:]]+)\"
)R";

/* Defining the text that needs to be printed. */


//Printing the previously defined text in the console
int main()
{
    std::cout << NTBs << '\n';
}
```

NAE : not same output

Prefer //, especially for singleline comment

********** End Code **********

********** Begin Output **********

```
^\\s+Encryption key:(\\w+)
^\\s+Quality=(\\d+)
^\\s+E?SSID:\"([[:print:]]+)\"
^\\s+ssid=\"([[:print:]]+)\"
```

********** End Output **********

Programming in C/C++
Week 1: Assignment 5
Tjalling Otter & Emiel Krol
Source: "De Programeertaal C, Frank B. Brokken, page 516"

```
\a      Adds an audible bell sound.
\b      backspace: moves the cursor back one space and removes the input of
        that space.
\f      form feed - new page: moves the cursor to the next page.
\n      line feed - new line: moves the cursor to the next line.
\r      carriage return: moves the cursor to the start of the current line
        or to the next line, which depending on the OS.
\t      horizontal tab moves the cursor to the right by one tab length.
\v      vertical tab moves the cursor down by one tab.
\0ddd   Ascii sign 'ddd': Returns an Ascii sign indicated by
        octal numeration.
\xdd    Ascii sign 'dd' : Returns an Ascii sign indicated by
        hexadecimal numeration.
```

If another character is written as an escape sequence,
for example ' \\ ', ' \ ' is returned after execution.

Example:

```
**************** Begin Code ********************
#include <iostream>

 //Printing text and adding a "\" on the next line Since it follows " '\n' "


int main()
{
    std::cout << "Example text" << '\n' << '\\';

}
**************** End   Code ********************
```

√ options!

```cpp
// Programming in C/C++
// Week 1: Assignment 6
// Tjalling Otter & Emiel Krol

#include <iostream>
using namespace std;

int main()
{
  unsigned int value;

  cout << "Please type an integer." << '\n';
  cin >> value;

  cout << (value % 2 ? "odd" : "even") << '\n';
  // The modulo operator, using the number 2, checks whether a value is fully
  // divisble (i.e. no remainder) by 2, and if so, returns 1 as the remainder
  // (true), else 0 (false).

  cout << (value & 1 ? "odd" : "even") << '\n';
  // Bitwise AND outputs a binary number equivalent to wherever both a and b have
  // the same bits set. Hence, performing bitwise AND on a number 'a' and '1' will
  // only output one if the last binary digit of 'a' is a one, making it uneven.

  cout << ((value ^ 1) == (value + 1) ? "even" : "odd") << '\n';
  // Bitwise XOR outputs a binary number that represents the bits that are set in
  // only one of the two numbers. Hence, performing this operation on a number 'a'
  // and '1' will increment the number only if 'a' is even (i.e. its binary value
  // ends with a 0).

  cout << ((value | 1) == (value + 1) ? "even" : "odd") << '\n';
  // Bitwise OR works, in this case, much like XOR. The number is incremented only
  // if the last digit of the binary representation of 'a' is zero, otherwise it
  // will remain the same (i.e. if it ends with a one, and is thus uneven).

  cout << ((value / 2) * 2 == value ? "even" : "odd") << '\n';
  // For integral divisions, any fractional part is discarded. Hence, any number
  // divisible by two will equal the initial value when multiplied by two again,
  // others won´t.

  cout << (((value >> 1) << 1) == value ? "odd" : "even") << '\n';
  // The right shift operator will truncate the right-most digit of a binary
  // number.Shifting this resulting number to the left and adding a zero at
  // the end (i.e. left shift) will return the original value once more only
  // if the last digit was a zero to begin with; hence, it was even.
}
```

```cpp
// Programming in C/C++
// Week 1: Assignment 7
// Tjalling Otter & Emiel Krol

#include <iostream>
using namespace std;

int main()
{
  unsigned int value;

  cout << "Please type an integer." << '\n';
  cin >> value;

  // The bitwise AND operator will return the bits that are set in both values.
  // Hence, for a number that is a power of two, it will share exactly none of its
  // bits with that number minus one, thus returning zero (i.e. false). For
  // example, 8 (1000) & 7 (0111) = 0.
  cout << "the value " << value << " is " << ((value &(value - 1)) ? "not " : "")
       << "an exact power of two" << '\n';
}
```

1

Hard on translation to other (human) languages. Bad – 0

Programming in C/C++
Week 1: Assignment 8
Tjalling Otter & Emiel Krol

Redirection:
When changing code in a program to write a file, one can use redirection in order to append the code in the already existing file, rather than erasing and rewriting the entire file. This saves time.

In the code " program <in> out " the executable "program" uses "in" as input file and produces file "out" as output file.

Piping:
Piping passes information from one program to another. One program can pass a parameter such as the output into the pipe, where it is stored. The other program can then use the information in the pipe as, for example, input. Pipes can be one way or two way.

In the code " code | less" the output of the program "code" is piped to the program "less". less allows one to view the output of program "code" one screen-full at a time.

```cpp
// Programming in C/C++
// Week 1: Assignment 10
// Tjalling Otter & Emiel Krol

#include <iostream>
#include <string>

using namespace std;

int main(int argc, char *argv[])
{
  size_t value = stoul(argv[1], 0, 16);          // initialize hexadecimal value
  size_t nibble = stoul(argv[2]);                // nibble to replace
  size_t replacement = stoul(argv[3]) % 16;      // new nibble (= 0 .. 15)

  replacement <<= (4 * nibble);                  // Shift new nibble to desired spot
  size_t offsetMask = 0b1111 << (4 * nibble);    // Mask the nibble to be replaced
  value = (value & ~offsetMask) | replacement;   // Zero and replace the nibble

  cout << hex << value << '\n';                  // show the new value
}
```