1.3 - Variables, Operators and Expressions

You are required to do and submit all homework assignments in each level.

Note: the following exercises are for for their respective video lecture denoted by 1.x (1.3 corresponds to video lecture 1.3)

If you have not done so yet, please read these two threads, as HW submissions and questions will only be responded to with strict adherence to these policies:

Getting started on the C++ course
Instruction on homework submission

All exercises in this Level must be coded *exclusively* in C syntax (no <iostream>, cout, cin, classes, etc.)

Exercise 1

printf() is a standard function. Each compiler will support this function. Between the () is stated what has to be printed (on screen). Create a C-program that prints the following when executed:

My first C-program
is a fact!
Good, isn't it?

Exercise 2

Write a program that calculates the surface of a triangle with one 90 degree angle. The formula is half the height multiplied by the base. The program should take an input from the user (base & height), and output the result.

Exercise 3

In the following program various operators are used to assign a value to the variable x. In this example the string that is passed to printf() has the format specification %d. This means that a decimal will be printed in place of %d. This decimal is passed to printf() as the second argument. The first argument of printf() must be a string. In this example the second argument is the variable x.

Predict what will be printed on screen (provide a code file with comments stating the output for each line).

```
/* Operators */

#include <stdio.h>

int main()
{
	int x;

	x=-3+4*5-6;
	printf("x=%d\n", x);

	x=3+4%5-6;
	printf("x=%d\n", x);

	x=-3*4%-6/5;
	printf("x=%d\n", x);

	x=(7+6)%5/2;
	printf("x=%d\n", x);

	return 0;
}
```

Exercise 4

Create a C-program that uses the fact that 0 (zero) is interpreted as FALSE and non-zero is interpreted as TRUE. The C-program can be made easier to read when this 0 (or non-zero) is assigned to a variable e.g. an int called married. Use the ?: operator to print if someone is married or not. (See if you can use a single printf)

See forum discussion on this exercise

Exercise 5

Create a C-program that clearly shows the difference between --i and i--.

Exercise 6

Write a C-program that shifts any number two places to the right. Input should be an integer. Output should be the shifted result, **as well as output an indication of whether a logical or arithmetic shift is performed** (if a 1 or 0 is shifted in at the left side) for the inputted number. For more info and example, see http://en.wikipedia.org/wiki/Logical_shift

Exercise 7

Write a C-program that efficiently multiplies a number by a factor 2 to the power n. The number to multiply and n are variables, which get a value at the start of the program.
Clue:

1 shift to the left is the same as multiplying by 2.
2 shifts to the left are the same as multiplying by 4.
3 shifts to the left are the same as multiplying by 8.

Exercise 8

The following program uses assignment-operators. Predict what will be printed on screen (provide a code file with comments stating the output for each line). The operators + and = = have a higher priority than the assignment-operators.

```
/* Assignment operators */

#include <stdio.h>

int main()
{
	int x=2;
	int y;
	int z;

	x*=3+2;
	printf("x=%d\n", x);

	x*=y=z=4;
	printf("x=%d\n", x);

	x=y==z;
	printf("x=%d\n", x);

	return 0;
}
```

Exercise 9

Predict what the following program prints on screen (provide a code file with comments stating the output for each line).

```c
/* Conditional expressions */

#include <stdio.h>

int main()
{
     int x=1;
     int y=1;
     int z=1;

     x+=y+=x;
     printf("%d\n\n", (x<y)?y:x);        // Number 1
     printf("%d\n", (x<y)?x++:y++);      // Number 2
     printf("%d\n", x);                  // Number 3
     printf("%d\n", y);                  // Number 4

     return 0;
}
```

1.4 - Flow Control

Exercise 1

Write a C-program that asks for text input from the keyboard. The output of this program should be the amount of characters, the amount of words and the amount of newlines that have been typed. Multiple consecutive spaces should*not* be counted as multiple words.

Reading keys from the keyboard is possible by using the function getchar(). The reading of characters from the keyboard can be stopped when the shutdown-code ^D (CTRL + D) is entered. ^D has the ASCII-value 4 (see forum discusson on this exercise). **Use a while loop.**

Exercise 2

Rewrite the C-program that was written for exercise 1, but use *do while* instead of *while*.

Exercise 3

Do exercise 1 again, but change your solution so that the *switch-case* statement is used instead of the *if* blocks.

Exercise 4

Create a C-program that prints two columns on the screen with the temperature in degrees Fahrenheit and the equivalent temperature in degrees Celsius.

The left column shows the temperature in Fahrenheit. The right column shows the temperature in Celsius.

Start with 0 degrees Fahrenheit and proceed until 300 degrees Fahrenheit. Take steps of 20 degrees. Print degrees Celsius with 1 position behind the comma (use “%10.1f” as format specifier). Also print a header text.

Make the program maintenance insensitive, which means that the start- and end-temperature and the step size must be easy to adjust.

The result must be obtained using a while construction!

Tip: To calculate the degrees Celsius from degrees Fahrenheit use the following formula:

$$Celsius = (5/9) * (Fahrenheit - 32)$$

Exercise 5

Create a C-program that prints two columns on the screen with the temperature in degrees Celsius in the left column and degrees Fahrenheit in the right column.

Start with 0 degrees Celsius and proceed until 19 degrees Celsius. Take steps of 1 degree. Print degrees Fahrenheit with 1 position after the comma. Also print a header text.

The result must be obtained using a for construction!

Exercise 6

Create a C-program that counts how many times each of the numbers 0-4 have been typed. Use a *switch-case* construction. Use default to count the number of other characters. The input will be halted with ^Z (EOF). EOF means End-of-File and is defined in <stdio.h>. Thus, the constant EOF can be used in a condition (test if EOF has been typed).
Print the amount of times a certain number has been typed.

Name the program freq.c

See this forum discussion

Exercise 7

Extend the program of exercise 6 in such a way that the frequency of number 3 is shown in words.
E.g.: Number three appears two times.

Only print the frequency when it is smaller than three. If the frequency is three or larger, then print: "The number three appears more than two times."

1.5 - Functions and Scope of Variables

Exercise 1

Write a C-program that calls a function *minus()*. This function receives two arguments and returns the difference (regular subtraction, not absolute). This difference should be printed on screen.

Exercise 2

Write a C-program that prints the factorials of a number.
6! (six factorial) is the same as 6 * 5 * 4 * 3 * 2 * 1

Must make use of a recursive function.

Exercise 3

Write a program that consists of two source-files. The first (Main.c) contains the *main()* function and gives the variable *i*a value. The second source-file (Print.c) multiplies *i* by 2 and prints it.

Print.c contains the function *print()* which can be called from *main()*.

Exercise 4

Write a **recursive** function *printnumber()* which gets the number to be printed. This number is an integer. The function should print the number digit by digit by using the *putchar()* function. Don't use *printf()*.

Tips: Use the modulo operator (%) to determine the digit to print. Use the division operator (/) to calculate the argument for the recursive call. Don't forget to handle negative numbers correctly.