Practice 3: Control structures (3 sessions)

Weeks from 23th september to 7th october 2024

Part 1: Conditional structures

Exercise 1.1

Write a C program that asks the user for two numbers and print out on screen the maximum of them or, if they are equal, a message to indicate that situation.

Examples of operation

```
$ ./exercise1_1

Introduce an integer number: 8
Introduce another integer number: 3
The number 8 is greater than the number 3
```

```
$ ./exercise1_1

Introduce an integer number: 32
Introduce another integer number: 32
The numbers 32 and 32 are equal
```

Exercise 1.2

Write a C program that asks the user for a time, only the hour between 0 and 23. The program will display a greeting depending on the time: ‘Good morning’, for a time between 7am and 2pm; ‘Good afternoon’, for a time between 3pm and 8pm; and ‘Good evening’ for a time between 9pm and 6am.

Examples of operation

```
$ ./exercise1_2

Introduce the hour of the day (0-23): 5
Good evening
```

```
$ ./exercise1_2

Introduce the hour of the day (0-23): 13
Good morning
```

Exercise 1.3

Write a C program implementing the calculator function for the operations addition (+), subtraction (-), multiplication (*) and division (/). The program will ask the user for operand 1, operand 2 and the operation, and return the result of op1 op op op2.

Note: The operands can have decimals.

Examples of operation

```
$ ./exercise1_3

Introduce the operand 1: 5
Introduce the operand 2: 7
Introduce the operation (+, -, *, /): +
The result of the expression 5.00 + 7.00 is 12.00

$ ./Exercise1_3

Introduce the operand 1: 234.5
Introduce the operand 2: 6
Introduce the operation (+, -, *, /): /
The result of the expression 234.50 / 6.00 is 39.08
```

Exercise 1.4

Write a C program that asks the user for three integers and stores them in three different variables a, b and c. The program will then check which is the greatest of the three and store it in the variable a. The lowest of the three will be stored in the variable c and the remaining number will be stored in the variable b. Finally, you must print each variable with its value. Important: Only one auxiliary variable can be used.

Input

Three integer numbers to be assigned to the variables *a*, *b*, *c*

Output

Three integer numbers corresponding to the values of the variables *a*, *b*, *c*

Example of operation

```
$ ./exercise1_4

Introduce the value of a: 4
Introduce the value of b: 1
Introduce the value of c: 9

Values before exchange: a=4 b=1 c=9
Values after exchange: a=9 b=4 c=1
```

Exercise 1.5

Write a C program that asks the user for three integer numbers and the order to be displayed, ascending or descending. Finally, the program will print the numbers out on the screen in the order indicated by the user.

Examples of operation

```
$ ./exercise1_5

Introduce three integer numbers
* Number 1: 5
* Number 2: 2
* Number 3: 8
Order you want them displayed, ascending (A) or descending (D)? A
Numbers in ascending order: 2 5 8
```

```
$ ./exercise1_4

Introduce three integer numbers
* Number 1: 5
* Number 2: 2
* Number 3: 8
Order you want them displayed, ascending (A) or descending (D)? D
Numbers in descending order: 8 5 2
```

Exercise 1.6

Write a C program to calculate the body mass index (BMI), which is calculated as `BMI=weight/(height*height)`. The program will display a message with the BMI value and the group you are in according to the following classification:

- If BMI < 18.0, Below normal
- If 18.1<=BMI<=24.9, Normal
- If 25.0<=BMI<=29.9, Overweight
- If BMI>=30, Obese

Note: Weight is expressed in kg and height in meters.

Examples of operation

```
$ ./exercise1_6

Introduce the weight (kg): 74.5
Introduce the height (m): 1.8
Value of BMI: 22.99 - Normal
```

```
$ ./exercise1_6

Introduce the weight (kg): 91.8
Introduce the height (m): 1.74
Value of BMI: 30.32 - Obese
```

Part 2.1: Repeating or iterative structures: determined loops

Exercise 2.1.1

Write a C program that asks the user for an integer number and print out on screen its multiplication table.

Example of operation

```
$ ./exercise2_1_1

Introduce an integer number: 4
4 x 0 = 0
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40
```

Exercise 2.1.2

Write a program in C that asks the user for a positive integer and prints out the divisors it has. In addition, it will also print a message telling if the number is prime or not. Note: If the number entered is not correct an error message will be displayed to the user and nothing will be done.

Examples of operation

```
$ ./exercise2_1_2

Introduce an integer number: 12
Divisors of 12: 1, 2, 3, 4, 6, 12 => IT IS NOT PRIME

$ ./exercise2_1_2

Introduce an integer number: 7
Divisors of 7: 1, 7 => IT IS PRIME

$ ./exercise2_1_2

Introduce an integer number: 156
Divisors of 156: 1, 2, 3, 4, 6, 12, 13, 26, 39, 52, 78, 156 => IT IS NOT
PRIME
```

Exercise 2.1.3

Write a C program that asks the user for a positive integer number and shows on the screen the countdown from that number to 0 by using commas.

Example of operation

```
$ ./exercise2_1_3

Introduce a positive integer number: 7
Countdown from 7 to 0: 7, 6, 5, 4, 3, 2, 1, 0
```

Exercise 2.1.4

Write a C program that displays on screen a menu that allows the user to choose between five figures to draw, asking the user for the figure size (minimum 4 and maximum 15). The menu will have an additional option to exit the program. As long as the chosen option is not correct, the menu will be shown again.

Example of operation

```
$ ./exercise2_1_4

MENU
------------
1 - Figure 1
2 - Figure 2
3 - Figure 3
4 - Figure 4
5 - Figure 5
0 - Exit

Choose an option: 3
Introduce the figure size (4 - 15): 5
```

Figure 1	Figure 2	Figure 3	Figure 4	Figure 5
`*`	`*********`	`*       *`	`    *****`	`    *`
`**`	` *******`	`**     **`	`   *****`	`   ***`
`***`	`  *****`	`***   ***`	`  *****`	`  *****`
`****`	`   ***`	`**** ****`	` *****`	` *******`
`*****`	`    *`	`*********`	`*********`	`*********`
			` *****`	` *******`
			`  *****`	`  *****`
			`   *****`	`   ***`
			`    *****`	`    *`

Exercise 2.1.5

Write a C program that asks the user for an even number. In case the number is not correct, the program will inform the user and do nothing. In case the number is correct, the program will print on the screen the following draw:

```
1
3 1
5 3 1
7 5 3 1
9 7 5 3 1
```

Draw obtained with a value of 10

Part 2.2: Repeating or iterative structures: indetermined loops

Exercise 2.2.1

Write a C program that asks the user an integer number between 50 and 100. If the number is not correct, a message will be shown and nothing will be done. If the number is correct, all the perfect squares lower than or equal to the input number will be shown.

Examples of operation

```
$ ./exercise2_2_1

Introduce an integer number between 50 and 100: 78
Perfect squares lower than 78: 1 (1 * 1), 4 (2 x 2), 9 (3 x 3), 16 (4 x 4),
25 (5 x 5), 36 (6 x 6), 49 (7 x 7), 64 (8 x 8)
```

```
$ ./exercise2_2_1

Introduce an integer number between 50 and 100: 34
ERROR: The introduced number (34) is not between 50 and 100
```

Exercise 2.2.2

Write a C program that generates a random number between 1 and 100. Then, the program will ask the user for numbers until the introduced number is equal to the generated one. At each attempt that the user does not guess correctly, the program will display a message saying whether the number to be guessed is higher or lower than the one entered. At the end of the program the number of attempts made by the user will be printed.

Note: To generate random numbers in C, two libraries must be included `<stdlib.h>` and `<time.h>`:

```c
#include<stdio.h>
#include<stdlib.h> // to generate the random number
#include<time.h> // to set the seed of random numbers

int main(){
  srand(time(NULL)); // initializing the random seed
  int numero = 20 + rand() % 11; // random number between 20 and 30
  // rand() % n  => generates a random number in the interval [0, n)
  ...
  return 0;
}
```

Example of operation

```
$ ./exercise2_2_2

Guess the hidden number in as few tries as possible
Introduce a number: 54
The hidden number is lower
Introduce a number: 45
The hidden number is lower
Introduce a number: 32
The hidden number is greater
Introduce a number: 37
The hidden number is lower
Introduce a number: 35
CONGRATS!!! You guess the hidden number in 6 attempts
```

Exercise 2.2.3

Write a C program that randomly generates three different numbers between 1 and 5, and displays them on the screen. At the end you should print the total number of numbers generated until you get the three distinct numbers.

Examples of operation

```
$ ./exercise2_2_3

The different generated numbers are: 5, 3, 4
Total generated numbers: 3
```

```
$ ./exercise2_2_3

The different generated numbers are: 5, 2, 4
Total generated numbers: 6
```

Exercise 2.2.4

Write a C program that asks the user for a positive real number until the value 0 is entered. Then, the program will display how many numbers have been entered and the arithmetic mean of all of them.

Example of operation

```
$ ./exercise2_2_4

Introduce a positive number (0 to finish): 4.5
Introduce a positive number (0 to finish): 6
Introduce a positive number (0 to finish): 3.75
Introduce a positive number (0 to finish): 0
# numbers: 3
Arithmetic mean: 4.75
```

Exercise 2.2.5

Write a C program that asks the user for an integer number and, then, prints out on screen the following information about the number: the number of digits, the number of even digits and the number of odd digits.

Example of operation

```
$ ./exercise2_2_5

Introduce an integer number: 42765
# digits: 5
# even digits: 3
# odd digits: 2
```