Practice 4: Modular programming (3 sessions)

Starting from 17th october 2024

Exercise 1

Write a C module receiving a speed in km/h and returning it in m/s. Note: It is necessary to implement a C program to test the module (it will be called from the `main()`).

Test cases:

km/h	m/s
27.8	7.72
84	23.33
126.5	35.14

Exercise 2

Write a module in C that receives two integer numbers and returns a random number between the two, inclusive. Note: It is necessary to implement a C program to test the module (it will be called from the `main()`).

Example of operation

```
$ ./exercise2

Introduce the minimum and maximum values of the interval: 5 15
The random number is: 13
```

Exercise 3

Write a C program that asks the user for the average fuel consumption of a vehicle (expressed in liters of fuel per 100 kms) and the kilometers to be traveled, and returns the number of liters of fuel needed. You must implement a module that receives the two data entered and returns the number of liters needed.

Example of operation

```
$ ./exercise3

Introduce the average fuel consumption expressed in liters per 100kms:
6.8
Introduce the kilometers to be traveled: 450
The number of liters needed to travel 450 kms is: 30.6
```

Exercise 4

Write a C program that asks the user for the radius of a circumference and returns the length of the circumference and the area of the circle. For that, the following modules must be implemented:

- `calculateCircumference`: receives the radius of the circle and returns the length of the circumference corresponding to the circle.
- `calculateArea`: receives the circle radius and returns the area corresponding to the circle.

Example of operation

```
$ ./exercise4

Introduce the circle radius: 3
* Circumference length: 18.8496
* Area: 28.2743
```

Exercise 5

Write a C program that asks the user for a 7-digit integer number (to be checked). If the number is correct, the program will print the sum of the even digits and the sum of the odd digits. Three modules will be implemented for this purpose:

- `countDigits`: receives the number entered by the user, and returns true if it has 7 digits, or false otherwise.
- `addEvenFigures`: receives the number entered by the user, and returns the sum of the even digits.
- `addEvenFigures`: receives the number entered by the user, and returns the sum of the odd digits.

Example of operation

```
$ ./exercise5

Introduce a 7-digit integer number: 42
Try again. Introduce a 7-digit integer number: 4251345
Addition of even digits: 10
Addition of odd digits: 14
```

Exercise 6

Write a C program that asks the user for the length of the side of a square and returns the length of its perimeter and its area. You must implement, in addition to the main module, a single module that receives the length of the side of the square and returns the perimeter and area of the rectangle.

Example of operation

```
$ ./exercise6

Introduce the side length: 3.5
```

```
3.5-square data:
* Perimeter: 14
* Area: 12.25
```

Exercise 7

Write a C program that asks the user for three integers and returns the lowest and the greatest of them. You must implement, in addition to the main module, a single module that receives the three numbers and returns the smallest and the largest.

Example of operation

```
$ ./exercise7

Introduce three numbers separated by commas: 5.3 2.75 4
The lowest of the three numbers is 2.75 and the greatest is 5.3

$ ./exercise7

Introduce three numbers separated by commas: 34 64.3 32.5
The lowest of the three numbers is 32.5 and the greatest is 64.3
```

Exercise 8

Write a C program that calculates the roots of a second degree equation. The program will ask the user for the coefficients a, b and c, and print the roots on the screen. You must implement, in addition to the main() module, a single module that receives the coefficients and returns the roots. Note: The equation is assumed to have a real solution in all cases.

```
$ ./exercise8

Introduce the coefficients a, b and c: 1 -5 6
The roots of the equation are: 3 and 2

$ ./exercise8

Introduce the coefficients a, b and c: 1 -14 49
The roots of the equation are: 7 and 7
```

Exercise 9

Write a C module that receives two integers and returns them swapped. Note: It is necessary to implement a C program to test the module (it will be called from the `main()`).

Example of operation

```
$ ./exercise9

Introduce two integer numbers a and b: 5 -8
The numbers before exchange are a=5 and b=-8
The numbers after exchange are a=-8 and b=5
```

Exercise 10

Write a C program that allows the user to play the well-known game of seven and a half. In the game of seven and a half, the player and the banker are dealt cards at random in an attempt to add up to a maximum of seven and a half. The first player to exceed 7.5 loses. Note: The random values represented by the cards are from 1 to 10, the figures being 1 (ACE), 8 (JACK), 9 (KNIGHT) and 10 (KING). The figures add up to only half a point while the other cards (2 to 7) add up to their score. As a minimum, the following modules must be implemented:

- `getCard`. It takes no input parameters and returns a random integer between 1 and 10 inclusive.
- `showCard`. It receives the random number representative of the card and writes the card on the screen as follows: if it is a figure, it will print the number of the card in brackets and then which figure it corresponds to, for example: (8) JACK. If it is not a figure, it will only write the value of the card.
- `updateScore`. It receives the representative random number of the card and the corresponding player's score, and updates the player's score taking into account whether the card is a figure or not.

Note: The `getchar()` function stops the execution of the program and waits for a character to be entered from the keyboard. If the character pressed is the `Enter` key, the program continues to run. You can use this function to pause the program.

Example of operation

```
$ ./exercise10

Press Intro to play ...

======================
Banker card: 3 => Score: 3
Player's card: (8) JACK => Score: 0.5

Scores:
* Banker: 3
* Player: 0.5

Press Intro for more cards...

======================
Banker's card: 3 => Score: 6
Player's card: 3 => Score: 3.5

Scores:
* Banker: 6
* Player: 3.5
```

```
Press Intro for more cards...

======================
Banker's card: (9) KNIGHT => Score: 6.5
Player's card: (1) ACE => Score: 4

Scores:
* Banker: 6.5
* Player: 4

Press Intro for more cards...

======================
Banker's card: 2 => Score: 8.5
Player's card: (1) ACE => Score: 4.5

Scores:
* Banker: 8.5
* Player: 4.5

**********
GAME OUT
**********
CONGRATULATIONS!! YOU WIN!!
```