Practical work nr. 6 – Functions

Subjects

- Using functions for program solving

Exercises

1. Create a function, `BMI(weight, height)`, to calculate the body mass index, having as inputs the weight (in kg) and height (in meters). Use it in a program that requests those parameters from the user.

2. Write a function to calculate the polynomial $p(x) = x^2 + 2x + 3$ and use it in a program to make a table of its values for x in the $[0, 2]$ interval, and using 0.1 increments in *x*. [That is, evaluate the polynomial for x=0, x=0.1, x=0.2, …, x=2.0.]

3. Create a function that allows to calculate the value of any second degree polynomial $g(x) = ax^2 + bx + c$. Notice that, besides *x*, the function has to receive the parameters *a*, *b* and *c*. Use this new function in a program to calculate the same values of the previous exercise.

4. Create a function that returns the greatest of its two parameters. For example, `max2(4,-5)` should return 4, whereas `max2(-3,-2)` should return -2.

5. Use the previous function as the base for a function `max3` that returns the greatest of its three parameters.

6. Write a function `countdown(N)` which prints a countdown starting from a positive number `N`. Test it in a program which requests the value of `N` to the user.

7. Write a function which determines how many digits has an integer positive number. Use it in a program which requests that value to the user.

8. Write a function which determines the binary representation of a positive integer number. Use it in a program which requests that value to the user.

9. Write a function which calculates the greatest common divisor between two integer positive numbers using the Euclidean algorithm (`https://en.wikipedia.org/wiki/Euclidean_algorithm#Implementations`). Use it in a program which requests that value to the user.

10. Write a function which calculate the sum of all integer numbers between two integer numbers passed as arguments (ex: `sumAll(1, 10)` should return the sum of all numbers between 1 and 10, including). Use it in a program which requests that value to the user.

11. Change the program developed in class #04 to solve problem 2, but now using functions. At least 4 functions should be implemented (think carefully in the arguments and return values of each one of them):
    a. reading with validation of the N value;
    b. reading of the keyboard values into an array;
    c. average calculation;
    d. standard deviation calculation;
    e. printing values greater than the average.

12. Change the program developed in class #04 to solve problem 4, but now with functions. A function for each option of the menu should be implemented.

13. Change the program developed on class #04 to solve problem 6, but now with functions. At least 4 functions should be implemented (the previously developed functions should also be reused):
    a. reading with validation of the N value;
    b. validated reading of the grades values into an array;
    c. counting calculation;
    d. histogram printing.

14. The Fibonacci sequence is a sequence of integers in which each element is equal to the sum of the two previous ones: $F_N = F_{N-1} + F_{N-2}$. The first values are defined as $F_0 = 0$ e $F_1 = 1$. Write a function, `Fibonacci(n)`, to calculate the n[th] Fibonacci number. Which is the value of F_{40}?

15. The number πcan be approximated by a truncated version of the Leibniz series:

$$\pi = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \frac{4}{13} - \cdots$$

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = \frac{\pi}{4}.$$

Write a function, `aproxPi(n)`, which returns the sum of the first `n` terms of this series. Test this function on a program which asks value `n` to the user. Try it with n=50000, 500000, e 5000000. Do you detect any pattern in the results obtained (compare the results you obtained with pi)?

16. Raising a number `n` to a power `p` is the same as multiplying `n` by itself `p` times. Write a function called `power()` that takes a `double` value for `n` and an `int` value for `p`, and returns the result as a `double` value. Use a default argument of 2 for `p`, so that if this argument is omitted, the number n will be squared. Write a program that gets values from the user to test this function.

17. Start with the `power()` function of previous exercise, which works only with type `double`. Create a series of overloaded functions with the same name that, in addition to `double`, also work with types `char`, `short`, `int`, `long`, and `float`. Write a program that exercises these overloaded functions with all argument types.