Definitions

PROGRAMMING AND INFORMATION SYSTEMS - 1º THEAT THE
GPSI TECHNICAL PROFESSIONAL COURSE

"If your projects are for a year, sow the grain. If it's for ten years, plant a tree. If it's for a hundred years, educate the people"

(Chinese proverb)

Module 02

structured programming

Introduction

Decomposing a problem is a very important aspect of solving any problem. Subprograms are an important tool that make it possible to divide a complex algorithm into a certain number of components, each component being implemented as a subprogram.

Two types of subprograms will be analyzed:

- **Roles** (produces result)
- **Procedures** (does not produce result)

The functions defined by the programmer have a similar structure to the procedures, with only the obligation to return a certain result.

In the Python language, only function definitions are allowed.

This will imply that the way a function is declared and called in an algorithm is slightly different than in a procedure.

The parameters in the procedures can be input (input) or output (output), but they are not always exclusively input or output. They often serve to transfer information to the subprogram, working as an input parameter, but they receive a new value from the subprogram, which they return to the calling point, then serving as an output parameter. Such parameters are called input-output parameters.

The difference between a function and a procedure is also reflected in the way each of these two types of subprograms is called, namely:

- when it comes to a procedure, you only have to write the respective identifier, with any arguments that it requires;
- when it comes to a function, it becomes necessary to include it in a writing or assignment instruction.

A very important advantage of subprograms is that they can be called as many times as we like within an algorithm. We can thus perform the same operation or set of operations at different points in an algorithm.

But the advantage of subprograms goes further: we may want to perform, at different points in an algorithm, the same type of operations (the same subprogram) but with different data. There, we have new elements that subprograms can use - parameters.

Parameters are elements similar to variables, but which are inserted in the headers of subprograms and which are then used in calls to those same subprograms. Values given in place of parameters, when calling a subprogram, are called arguments.

A subprogram can contain more than one parameter (inserted within parentheses). When the subprogram is called, the order of the arguments is taken into account, as well as the data type to which they belong, and as many arguments as the parameters are needed. The arguments used in subprogram calls can be not only direct values, but also variables and expressions, as long as the values are compatible with the corresponding parameters.

Advantages of using subprograms:

- the complexity of the global solution is limited at the expense of the combination of sub-programs;
- it contributes to the clarity and readability of the programs, as the code that solves each task is separate;
- it is easier to identify the data needed and the results produced by each task;
- the existence of redundant (repeated) code is avoided, since by bringing together the code common to different tasks, it can then be reused at different points in the program.

Activity proposal

1. Write a function named sum_squares that takes a positive integer, n, and its value is the sum of the square of all numbers up to n.

```
> > > sum_squares(3)
14
> > > sum_squares(5)
55
```

2. Write a function that determines and displays the first n multiples of an integer m on the monitor (they shouldn't even be input parameters to the function). Implement the subprogram in Python language, building a main program that takes care of I/ O tasks and calling the created function.

3. Make a function that, given as an input parameter an integer, calculates and displays on the monitor all its multiples less than 100. Implement the program taking into account the requirements (relating to the structure of the program) defined in 1.

4. Write a function that displays on the monitor all the even numbers between two integers nor (n<m). The values of nor must be passed as parameters to the function and in the case of (m<n) the message “Invalid Values!” must be displayed.

5. Implement a function that allows you to exchange the value of two variables, whose value was specified by the user. Test it in a small program where the values are introduced in the main program and the change takes place in the function.

6. Write a function that receives as input parameters the measurements of the two legs of a right triangle, and returns the measurement of the hypotenuse.

7. If you had to return two values, would the solution used in the previous question be usable? Justify. Show the alternative so that the function returns the hypotenuse and the triangle perimeter.

8. Design a program that determines the square of an integer n. The number n must be asked to the user and, through a function, return its square.

9. Write a program that allows you to convert temperatures in degrees Centigrade to degrees Fahrenheit and vice versa. To do this, you must write two functions, one to perform the conversion to Celsius and the other to perform the conversion to Fahrenheit. Each of these functions must receive the value of the temperature to be converted as a parameter and display the converted temperature on the monitor (C = 5*(F – 32)/9; F = (9*C/5)+32).

10. Write a program that uses a function called *multiple*, which receives two integer values and returns *true* or *false* whether one of the values is a multiple (or not) of the other. The main program should accept two numbers and say whether or not the numbers entered are multiples of each other.

11. Implement a function called factorial that calculates n! (factorial of n, where n is a positive integer). The value of n must be a function parameter and the calculated value must be sent abroad.

12. Add to the above program a function that calls the factorial function to show the factorial of numbers 1 to 10.

GOOD WORK! YOU ARE ABLE! BUILD YOUR KNOWLEDGE...

The subject teachers, Andreia
Backyard | Carlos Almeida