Computing Fundamentals

Salvatore Filippone

salvatore.filippone@uniroma2.it

2013–2014

Functions

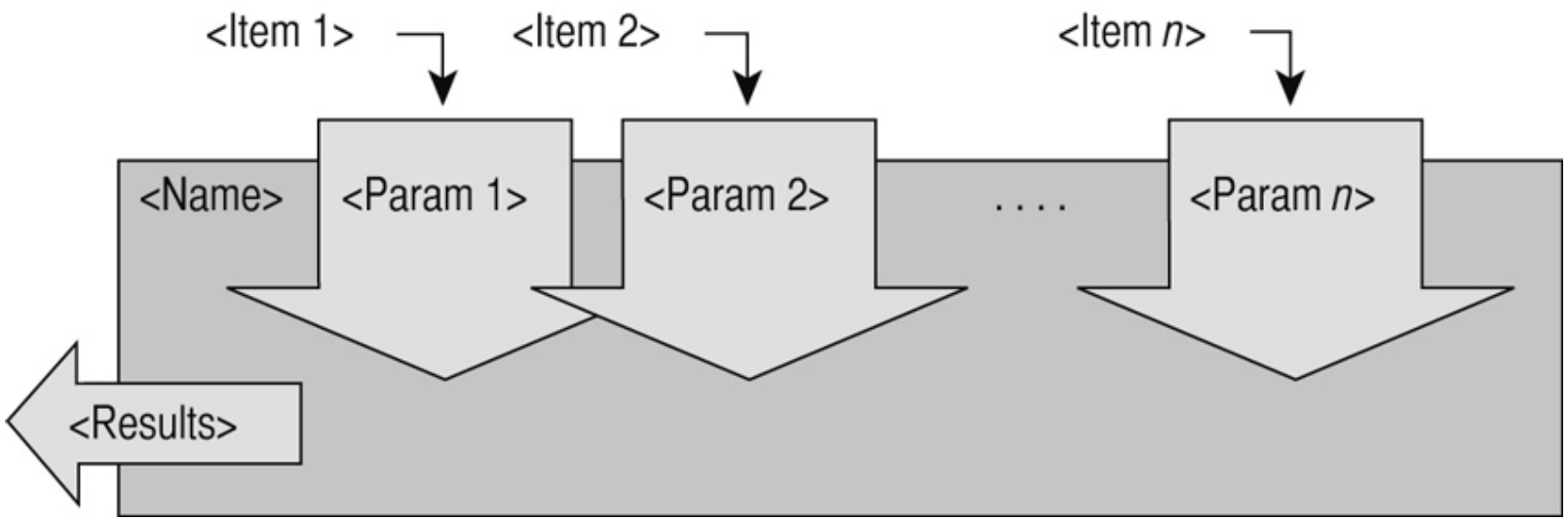
Functions are an essential ingredient of programming. They do:

1. Encapsulate a set of instructions for easy reuse;
2. Separate interface from implementation;
3. Separate variable spaces.

Functions

The black box view of functions:

A function only communicates with the rest of the world via its input arguments and its result(s)



Function template

```
function <results> = <function name> (<arguments>)
  <documentation>

    <code body>
end
```

The final `end` is optional.

Example 1

```
function    mx=amax(v)
  % Return the max absval of a vector
  mx=0;

  for i=1:length(v)
    if (abs(v(i)) >= mx)
      mx = v(i);
    end
  end
```

Trovare l'errore in questa funzione!

Example 2

```
function    [mx,ix]=iamax(v)
  % Return the max (abs) val of a vector
  % and its position
  mx=0;
  ix=0;

  for i=1:length(v)
    if (abs(v(i)) >= mx)
      mx = abs(v(i));
      ix = i;
    end
  end
```

Calling conventions

What happens when you invoke a function?

1. You (the user) type the name of the function, and specify constants and/or variables (*actual parameters*)
2. The system establishes a correspondence between the entities in the caller's workspace and the *formal arguments* in the funcion's workspace;
3. The function does its computations, and assigns values to the results;
4. The results are associated with the variables specified by the user.

The correspondence on entry/exit is established by *copying* the value; this is the *call by value* mechanism.

Calling conventions

What happens when you invoke a function?

```
function volume = cylinder1(height, radius)
% function to compute the volume of a cylinder
% volume = cylinder(height, radius)
    base = pi * radius^2;
    volume = base * height;

h=10;
r=2;
vol=cylinder1(h,r);
```

Functions

Fine points:

- The work space of the function is separated from the outer workspace; (try `whos`)
- The copy semantics means that the function is normally free to alter the input arguments;
- But copy semantics also means there is an overhead that can be quite substantial;
- It is possible to declare a `global` variable (must be done in both caller and function), but use with great care!
- A function may invoke other functions;
- Always test functions;

Functions

Fine points:

- The Matlab/Octave language is peculiar in making it easy to return multiple variables;
- The input arguments may be specified only in part: `nargin`, `nargout`;
- A function is stored in a file with the same name and an `.m` extension;
- How does Octave/Matlab search for a function?

Style points:

- Never trust the user (not even yourself!)
- Never do input/output inside a function;
- Always debug in isolation;
- Consider carefully what a function should do, and whether to partition its work.

Functions

Examples:

- Find the roots of a quadratic polynomial;
- Evaluate a polynomial;
- Compute the GCD of two numbers;
- Book exercises;