Parameters

Parameters allow us to pass information or instructions into functions and procedures. They are useful for numerical information such as stating the size of an object. The values passed in parameters are called arguments.

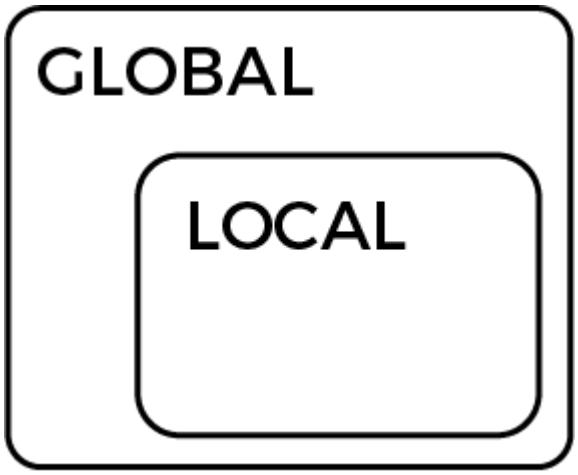
Getting back to our example of draw a square, we can create a procedure for that- but for it to be useful we need to also be able to specify how large it should be. The following example creates a procedure called 'draw_square'.

```
PROCEDURE draw_square(side_length : FLOAT)
.
END
```

In the line where we define the name for this procedure, we have included a variable called 'side_length' inside the brackets. Side_length is a parameter - it allows us to pass a value into the procedure for it to use.

Parameters

Scope refers to where a function, procedure, variable or constant can be used. They can be local or global.



A global variable is declared in a way that makes it available to every other part of the program, meaning we can call it in any function/procedure without passing as parameter. Whereas a local variable is declared in a way that makes it only available within a specific function or procedure, they are the variables declared in the VAR part of the function, the procedure, or the main algorithm itself.

We do not have a global keyword to declare global variables. However, variables declared outside functions, procedures and main algorithm have "file scope," meaning they are visible within the file, unless they are shadowed by a like-named variable in a local scope. For example: if we have a global variable called 'X' and inside a function we declared a local variable 'X', we cannot use the global 'X' inside that function anymore.

Parameters

Calling the previous procedure to create a square of size 50 for each side will be with this instruction :

```
draw_square(50);
```

The 50 here is called the argument - because it is a value passed in.

You can add more parameters to a function or procedure by separating the variable names with commas. For example:

```
PROCEDURE draw_rectangle(height : FLOAT, width : FLOAT, color : STRING[10])
```

Or you can optimize if two parameters with same type follow each other :

```
PROCEDURE draw_rectangle(height, width : FLOAT, color : STRING[10])
```

Note: You must respect the same order of the parameters when you call a function or procedure.

‹ Previous next ›