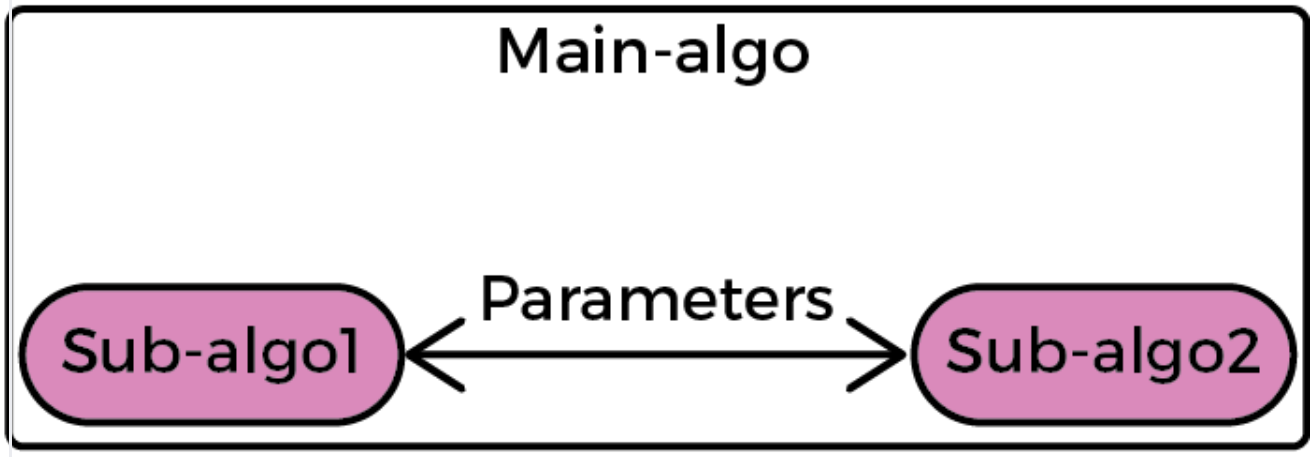
2 types of sub algorithms:

- Procedure
- Function



The sub-algorithms (procedures or functions) are called (used) inside an algorithm, the variables of this main algorithm will be the vessel that links between the sub-algorithms.

In other words, communication between two sub-algorithms will be via parameters (variables).

Each sub algorithm can have an input parameters, and can give an output parameters.

Using functions and procedures

In a computer algorithms there are often sections of the program that we want to re-use or repeat. Chunks of instructions can be given a name - they are called functions and procedures. This saves time by only having to execute (call) the function when it is required, instead of having to type out the whole instruction set.

Algorithm and even Programming languages have a set of pre-defined (also known as built-in) functions and procedures. Use some of them in our previous code such as Read(), Write(), push()....

If the programmer makes their own ones, they are custom-made or user-defined.

Procedural programming: Benefits of Cutting algorithms

Procedures or functions?

A procedure performs a task, whereas a function produces information.

Functions differ from procedures in that functions return values, unlike procedures which do not. However, arguments can be passed to both procedures and functions.

In a program for drawing shapes, the program could ask the user what shape to draw. The instructions for drawing a square could be captured in a procedure. Nothing to return here, the square is already drawn on the screen.

A function could calculate something like the VAT due on goods sold. the set of instructions of calculation the VAT could be given the name 'calculate_VAT' and would be executed by running (calling) that function, and returning the calculated value as a result.

Description of Procedure:

```
PROCEDURE procedure_name(List_parameters)
VAR
    set_declarations
BEGIN
    Procedure’s body (set_instructions)
END
```

Use of Procedure: call it directly in the code “procedure_name(List_parameters);

Description of Function:

```
FUNCTION function_name(args) : return_type
VAR
    set_declarations
BEGIN
    set_instructions
    RETURN value ;
END
```

Use of Function: call it like an affecting expression in the code “variable := function_name(args);”
Variable must be compatible with the return type of the function

Practicing with procedures and functions.

Practice Function

Previous next