Introduction

The topic of this Skill is going to be the procedural programming. We are going to see:

- What's the interest of procedural programming?
- what is the theory behind procedure and function?
- What are the parameters?
- How to pass parameters?

Interest of Procedural Programming

Solving a complex problem can generate thousands of lines of code.

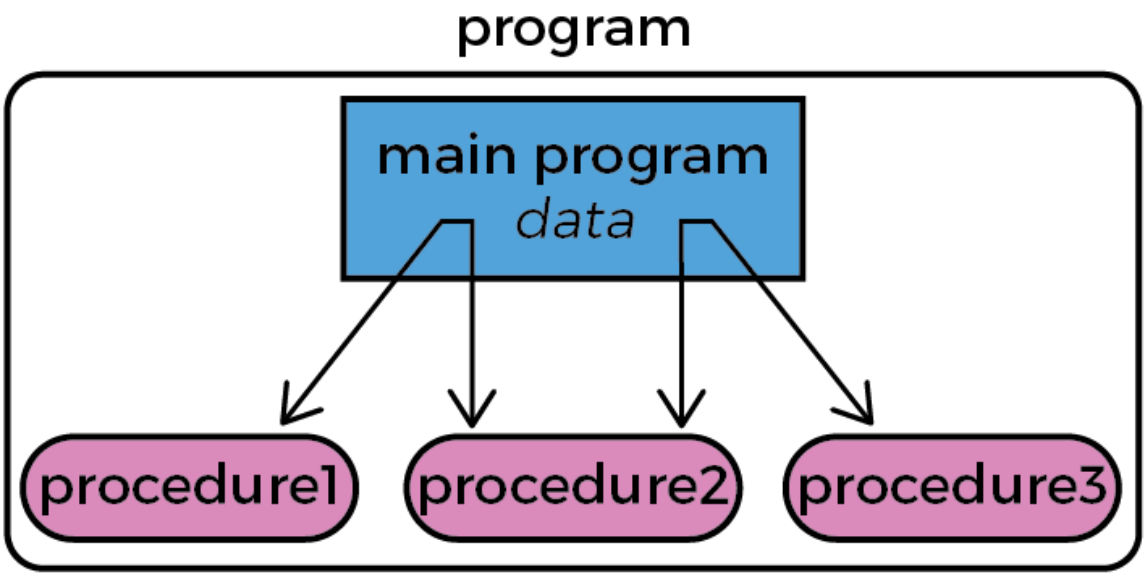
Long algorithms, difficult to write, interpret, and maintain.

Solution:

Using a new resolution methodology: Procedural Programming

Idea:

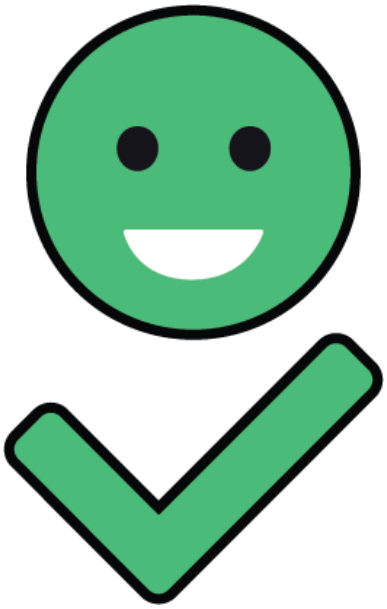
Divide the complex problem into less complex sub-problems and write each one of them separately within its proper environment.



Benefits:

- Clarity of the algorithm
- A more readable algorithm

- Ease of maintenance
- Reuse of sub-algorithms



More about benefits

As we have seen in the previous slides, the main key benefits of procedural programming are:

The clarity and readability of the algorithm,

The reuse of code,

The facility of maintaining it.

Let's create an algorithm that calculates the sum of the successive elements.

The code below is our first try:

```
ALGORITHM array_sum_two_element
VAR
  tab : ARRAY_OF type[50];
  i,n : INTEGER;
BEGIN
   // Read the size
   Read(n);
   // Read elements
   FOR i FROM 0 TO n-1 STEP 1  DO
       Read(tab[i]);
   END_FOR
   // Add successive element
```

```
    FOR i FROM 0 TO n-2 STEP 1  DO
    tab[i] := tab[i]+tab[i+1];
    END_FOR
    // Display the array
    FOR i FROM 0 TO n-1 STEP 1  DO
        Write(tab[i])
    END_FOR
END
```

This algorithm works fine, but we can optimize it more.

As we can notice, the main idea is to create an algorithm that calculates the sum of successive elements.

Even though they are necessary, the two pieces of code do not fit in the solution

They are, in fact, the reading of the array elements and the displaying of the result.

If we want to take advantage of procedural programming, we can divide the previous algorithm into three sub-algorithm (each one of them will resolve a sub-problem). Our final version will look like:

```
ALGORITHM array_sum_two_element
VAR
  tab : ARRAY_OF type[50];
  i,n : INTEGER;
BEGIN
   // Read the size
   Read(n);
   // Read elements
   /* *** *** */
  // Sub-algorithm reading elements of array
  /* *** *** */
   // Add successive element
  // Sub-algorithm calculating successive elements of array
  /* *** *** */
   // Dispalay the array
  // Sub-algorithm displaying elements of array
```

```
    /* *** *** */

END
```

Don’t worry, this is just a draft, we will see how to create a procedure and function with more details in the next slides.

Previous next