Larbi Ben M'hidi -Oum El Bouaghi- University
Faculty of Exact Sciences and Natural and Life Sciences
Mathematics and Computer Science Department

Academic year: 2023-2024

Level: 1st year "Computer Science & Mathematics"

Module: Algorithmic and Data Structures 2

TD n°2 "Solution"

Pedagogic objectives

→ Manipulate sub-algorithms (subroutines): procedures & functions;
→ Understand the difference between them;
→ Understand the concepts: local variable, global variable, formal parameter, effective parameter, passing parameters by value and by address.

Exercise n°1

Q1 and Q2

```
Algorithm exo1;                          Global variables
Variables X ,Y,Z : integer;
P_Test Procedure (A, B, C: integer);     Formal parameters
Variable  S: integer;                    Local variable
Begin
S←A+B+C
Write (S);
End;
Begin
Read (X,Y,Z);                            Effective parameters
P_Test (X, Y, Z);
END.
```

Q3 and Q4

```
Algorithm exo1;
Variables X ,Y,Z, R : integer;
Function F_Test  (A, B, C: integer): integer;
Var S: integer;
Begin
S← A+B+C.
F_Test←S;End;
Begin
Read (X,Y,Z);
R← F_Test (X, Y, Z);Write
(R);
END.
```

Exercise n°2:

- The effective parameters are those of the calling program (of the main algorithm): *x, y, z, t.*
- The formal parameters are those of the sub-algorithm (the procedure) and make it possible to recover the value of the real parameters: *a, b, c, d*

- *What does this program display assuming the user enters 2 in x and 3 in y ?*

```
Algorithm exo2;
Procedure ma_ procedure ( a, b, var c, d: integer)
Begin
      c ← a + b ;
      d ← a × b;
End;

Variables  x , y, z, t: integer ;
 Begin
     read (x);
     read (y);
     my_procedure (x, y, z, t);


     write (z);
     write ( t);
END
```

x	y	z	t
2	3	a + b = 2+3= **5**	

a	b	c	d
2	3		a × b = 2×3 = **6**

z	t
5	

After the call to *my_procedure* , ***z* = 5** and ***t* is worth nothing specific**. Indeed, c is a parameter passed by variable, so the modifications made to *c* in the sub-algorithm are reflected in the corresponding effective parameter *z*. On the other hand, *d* being a parameter passed by value,

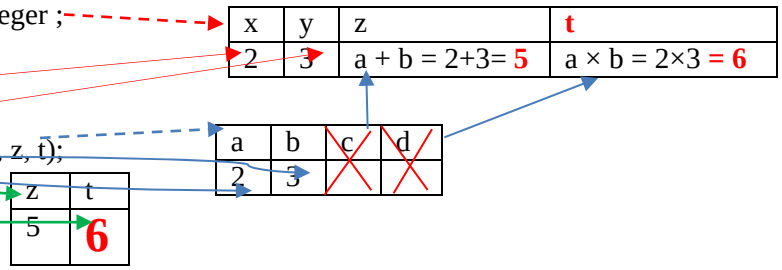
the corresponding effective parameter *t* is not affected by the changes. *t* therefore retains its value before the call to the sub-algorithm.

- *Modify the code to obtain a more logical result.*

By "transforming" *d* into a parameter by variable (address), in this case *my_procedure* takes two integers as input and returns via *c* the sum of these two integers and via *d* their product.

Algorithm exo2;

Procedure my_ procedure (a, b, **var** c, **var** d: integer)
Begin
c ← a + b ;
d ← a × b;
End;
Variables x , y, z, t: integer ;
Begin
read (x);
read (y);
my_procedure (x, y, z, t);
write (z);
write (t);
END



Exercise n° 3

1. Run the algorithm with the following array:

1	0	2	4	3	1	2	1	2	3

The algorithm displays:

19
0

2. What is the role of the procedure P1?

   It calculates the sum of the elements of an array.

3. What is the role of the procedure P2?

   It calculates the product of the elements of an array.

4. In the main algorithm, is it possible to calculate C1= a/2 and C2 = b/2?

   No, we cannot.

Justify your answer

The variables **"a",** declared to calculate the sum, and " **b",** declared to calculate the product, are **local variables.** In fact, they cannot be used in the main algorithm because the space of these local variables is freed at the end of the execution of the procedures.

5. Replace both procedures with functions.

```
Algorithm exo3;
Variables T : array [1..100] integer;
          N,S,P, i:integer;
Function F1 (T: array [1.. N ] integer): integer ;
Variables i , a: integer;
Begin
   a ←0 ;
For i ← 1 to N do
      a←a+ T [i] ;
End for
   F1←a ;
End;
Function F2 ( T: array [1.. N ] integer ): integer ;
Variables i , b: integer;
Begin
b←1;
For i ← 1to N do
      b←b *T[i];
Endfor
    F2←b ;
End;
Begin
Repeat
   Read(N);
Until (N>=1 and N<100)
 For i ← 1 to N do
     Read (T[i]);
 Endfor
  S← F1(T); Write (S);
  P ←F2(T); Write (P);
END
```

In this case, is it possible to calculate C1and C2 in the main algorithm?

Yes. The use of functions makes it possible to recover the result of the sum “a” in **the global variable S** and the result of the product “b” in **the global variable P** ;

Exercise n°4:

1.

```
Algorithm exo4;
Variables  T: array [1..20] real;
                val: real;
                case: integer;
                 b: boolean;
Procedure FindVal (N: Integer,T: array[1.. N ] real,val: real; Var rg : integer, found: boolean );
Variables i: integer;
Begin
found ← false;
rg ← 0 ;
i ← 1 ;
Repeat
 If (T[i] = val) then
    found ←  true;
    rg ← i;
 else
  i ← i + 1 ;
 endif
Until (found = true OR i > N)
End;
Begin
// We assume that T is already filled
Read (val);
FindVal (20, T, val, case, b);
If (b = true) then
    Write ("The value", val, "was detected at the position", case);
Else
    Write ("Value not found");
Endif
END
```

2.

```
Algorithm exo4;
Variables A1, A2: array [1.. 20 ] real;
            M1, M2: integer;
            b: boolean;
Function Compare (N1, N2: integer, T1: array [1..N1 ] real, T2: array [1..N2] real): boolean;
Variables i: integer;
            identical: boolean;
Begin
```

```
 If (N1 ≠ N2) then
     identical = false; // if the arrays are different sizes, they are not identical.
else
  i←1;
  identical ← true;
Repeat
  If (T1[ i] ≠ T2[i] )Then
     identical ← false;
 else
    i ← i+1 ;
  Endif
Until (identical = false OR i > N1)
 Compare ← identical;
End if
END
Begin
Read (M1 , M2); // M1 and M2 <= 20 We assume that A1 and A2 are already filled
    b= Compare (M1, M2, A1, A2 );
    If (b = true) then
       Write ("the two tables are identical");
    else
       Write ("the two tables are not identical") ;
    Endif
END
```

Exercise n° 5:

1.

```
 Function Som_Div (n: integer): integer;
    Variable i, sum: integer;
 Begin
   sum ←0 ;
   For i ←1 to (n div 2) do
        If (n mod i = 0) then
            sum ← sum+i ;
       Endif
    EndFor
  Som_Div← sum;
End;
```

2.

```
 Procedure P_Perfect (n: integer);
   Variable S: integer;
Begin
   S← Som_Div (n);
```

```
    If (S = n) then
        Write ("the number", n, " is perfect");
    Else
        Write ("the number", n, "is not perfect");
    Endif
End;
```

3.

```
Function F_Perfect (n: integer): boolean;
Variable S: integer;
Begin
    S= Som_Div (n);
    If (S = n) then
        F_Perfect ←true;
    Else
        F_Perfect ←false;
    Endif
End;
```

4. Main algorithm:

```
Algorithm Perfect;
Variables Bool : boolean, n: integer;
//Declaration and definition of subroutines 1, 2, 3.
Function Som_Div (n: integer): integer;
    Variable i, sum: integer;
 Begin
    sum ←0 ;
    For i ←1 to (n div 2) do
        If (n mod i = 0) then
            sum ← sum+i ;
        Endif
    EndFor
   Som_Div← sum;
End;
Procedure P_Perfect (n: integer);
  Variable S: integer;
Begin
   S← Som_Div (n);
   If (S = n) then
       Write ("the number", n, " is perfect");
    Else
        Write ("the number", n, "is not perfect");
    Endif
End;
Function F_Perfect (n: integer): boolean;
```

```
   Variable S: integer;
   Begin
        S= Som_Div (n);
        If (S = n) then
            F_Perfect ←true;
         Else
             F_Perfect ←false;
        Endif
   End;
```

```
Begin
Write ("Enter a number"); Read (n);
// Procedure call
P_Perfect (n); // here, “n” is an effective parameter.
// Function call
Bool ← F_Perfect (n); // here, “n” is an effective parameter.
If (Bool = true) then
   Write (“the number”, n, “ is perfect”);
Else
    Write (“the number”, n, “is not perfect”);
End if
END
```