# Chapter 3 : Conditional Instructions

## 1. Introduction

The instructions of the algorithms seen previously are executed sequentially; they are executed in the same order of their appearance, 1st, 2nd, 3rd, ... etc.

Often the problems being addressed require the study of several situations or cases that cannot be represented by the same sequences (groups) of instructions. These are the conditional instructions that differentiate between these situations based on what is called a condition.

Conditional instructions are classified into three types: simple, alternative, and multiple-choice.

### Simple Conditional Structure

**General Syntax :**

```
if (Condition) then
  IG ;
Endif ;
```

In C language, the simple conditional instruction is written as :

```
if (Condition){
   IG;
}
```

**Operation of simple conditional instruction :**

After the evaluation of the condition (a logical expression with a true or false value), if the condition is verified, the instruction group "IG" will be executed, and if the condition is not verified, execution continues with the instruction following "**EndIf**" without executing the instruction group "IG".

> **Note:**
> The Instruction Group "IG" can contain one or more instructions.

**Example:**

Let be the problem of determining the absolute value of an integer (or real) number.

**Solution:**

**Analysis of the problem :**

- Data : an integer X.
- Results : absolute value of X
- Steps of resolution:
  - Reading of the value of the integer X
  - Determination of the absolute value of X : we distinguish two cases

1 case : if the integer X is positive, its absolute value is the number itself (X).

2 case : if the integer X is strictly negative, its absolute value is its opposite (-X).

The translation of the resolution steps produces the following algorithm:

```
Algorithm absolute_value_integer ;
  Var X, abs_val_X : integer ;
Begin
   Write("Give a value of integer X : ") ;
   Read (X) ;
   if (X≥0) Then
      abs_val_X ← X ;
   Endif ;
   if (X<0) Then
      abs_val_X ← -X ;
   Endif ;
   write ("|",X, "|=",abs_val_X) ;
End.
```

The previous algorithm is translated into C language as follows:

```
# include <stdio.h>
main()
{
   int X, abs_val_X ;
   printf("Give a value of integer X : ");
   scanf("%d",&X);
   if (X>=0){
      abs_val_X = X ;
   }
   if (X<0){
      abs_val_X = -X ;
   }
   printf("| %d |= %d\n",X,abs_val_X) ;
}
```

**Remarks :**

- *If the block contains more than one instruction, then both braces { and } are required to delimit the sequence of instructions. Braces are optional if there is only one instruction.*
- *If you place a semicolon just after the condition, the compiler considers the "if" block to be composed of a single instruction (nothing...).*

## 2. Alternative conditional Instructions :

**Genaral syntax :**

```
If (condition) Then
     IG_1 ;
else
     IG_2 ;
Endif ;
```

In C language, alternative conditional instruction is written in the form :

```
if (condition){
      IG_1 ;
}
else{
      IG_2 ;
}
```

**Operation of the alternative conditional instruction :**

Like the simple conditional statement, we start by evaluating the condition. If this condition is satisfied (true), the only instruction group that will be executed is IG_1, and if the condition is not satisfied (false), only instruction group IG_2 will be executed.

By using the alternative conditional instruction, the algorithm for absolute value can be written as follows :

```
Algorithm absolute_value_integer ;
  Var X, abs_val_X : integer ;
Begin
   Write("Give a value for integer X : ");
   Read (X) ;
   If (X≥0) Then
      abs_val_X ← X ;
   else
      abs_val_X ← -X ;
   Endif ;
   write ("|",X, "|=",abs_val_X) ;
End.
```

The previous algorithm is translated into C language as follows :

```
# include <stdio.h>
main()
{
   int X, abs_val_X ;
   printf("Give a value for integer X :");
   scanf("%d",&X);
   if (X>=0) {
      abs_val_X = X ;
   }
   else{
      abs_val_X = -X ;
   }
   printf("| %d |= %d\n",X,abs_val_X) ;
}
```

### 3. Multiple Choice Conditional Instruction

The multiple-choice conditional Instruction allows to compare an object (expression) to a serie of its possible values, and to execute only one instruction group among several, depending on the effective value of the object. A default instruction group (IG_n+1) can be executed in case the object is different from all possible values.

Each instruction group is labeled with a corresponding value. Each group will be executed if its value is equivalent to the value of the expression. The multiple choice conditional instruction is :

```
According to (expression) do
  case val1 : IG_1 ;
  case val2 : IG_2 ;
   …
  case valn : IG_n ;
 Otherwise : IG_n+1 ;
EndAccording ;
```

**Operation of the multiple-choice conditional instruction :**

After the evaluation of the expression (integer, character or boolean) and according to its value :

- if it equals val1 (case 1) only the IG_1 instruction group will be executed,
- if it equals val2 (case 2) only the IG_2 instruction group will be executed,
- …
- if it equals valn (case n) only the IG_n instruction group will be executed,
- if it is different to all possible values (val1, val2, …, valn) only the instruction group of "otherwise" IG_n_1 will be executed

**Example**: Consider a problem of implementing a basic calculator that allows performing the four basic arithmetic operations (addition, subtraction, multiplication, and division).

**Problem Analysis :**

- **Data:**
  - Two real numbers, X and Y
  - An arithmetic operator.

- **Results :**
  - The value of the arithmetic operation on the two real numbers X and Y.

- **Resolution Steps :**
  - Read the values of X and Y.
  - Read the arithmetic operator.
  - Apply the operator to the two numbers (X and Y). we distinguish 4 cases:
    - If the operator is '+' the result is the sum of the two numbers X and Y (X+Y).
    - If the operator is '-' the result is the difference between these two numbers X and Y (X-Y).
    - If the operator is '*' the result is the product of the two numbers X and Y (X*Y).
    - If the operator is '/' and the number Y is non-zero, the result is the quotient of the division of X by Y (X/Y).

The translation of the resolution steps produces the following algorithm:

```
Algorithm basic_calculator ;
  Var X,Y,S : real ;
  Operator : char ;
Begin
   Write ("Give a two real numbers X and Y: ") ;
   Read (X,Y) ;
   Write ("Give an arithmetic operator : ") ;
   Read (Operator) ;
   According to (Operator) do
      case '+' : S ← X+Y ;
                 Write (X,"+",Y,"=",S) ;
      case '-' : S ← X-Y ;
                 Write (X,"-",Y,"=",S) ;
      case '*' : S ← X*Y ;
                 Write (X,"*",Y,"=",S) ;
      case '/' : If (Y=0) Then
                    Write ("Error : division by zero ");
                 Else
                    S← X/Y ;
                    Write (X,"/",Y,"=",S) ;
                 EndIf ;
      Otherwise
        Write ("Error : ",Operator, "it's not an arithmetic
                                                Operator ") ;
   EndAccording ;
End.
```

The previous algorithm is translated to C language as follows :

```
# include <stdio.h>
main()
{
  float X,Y,S ;
  char Operator ;

  printf ("Give a two real numbers X and Y: ") ;
  scanf ("%f%f",&X,&Y) ;
  printf ("Give an arithmetic operator : ") ;
  scanf (" %c",&Operator) ;

  switch (Operator){
    case '+' : S= X+Y ; printf ("%f+%f=%f\n",X,Y,S); break;
    case '-' : S= X-Y ; printf ("%f-%f=%f\n",X,Y,S); break;
    case '*' : S= X*Y ; printf ("%f*%f=%f\n",X,Y,S); break;
    case '/' : if (Y==0){
                printf("Error : division by zero "); break;
               }
               else{
```

```
                    S= X/Y; printf("%f/%f=%f\n",X,Y,S); break;
                  }
      Default :
            printf("Error : %c it's not an arithmetic operator ", Operator) ;
   }
}
```

**Notes :**

- In C language, the expression and values to be chosen must be of type **int** or **char**.
- The **break** instruction ends the execution of the **switch** instruction (statement) in which it appears. In case of missed **break**, the instructions of the following cases will also be executed.