Chapter 2

Conditional structures (in algorithmic language and C)

The aim of this chapter is to present conditional action in its three forms: simple conditional action (reduced and alternating), generalized conditional action ('embedded') and choice conditional action, as well as a translation of these forms into the C language. The chapter ends with a series of exercises for you to solve and practise at home.

Problems often require the study of several situations that cannot be dealt with by simple sequences of actions. For example, to solve a second-degree equation, after calculating the determinant we need to test whether it is positive, negative or zero. The instructions read(), write() and assign ⟵ cannot indicate this sign; other instructions are therefore needed ***: the conditional structures.***

Conditional structures are instructions that allow you to test whether a condition (see page 16) is true or not.

1. Definition

The conditional control structure allows a program to modify its processing according to a condition. There are three types of conditional instructions:

- **Simple form**
- **Generalized (nested or embedded) form**
- **Choice form**

1.1. Simple conditional control structure

It is distinguished by two forms: reduced and alternating

1.1.1. The reduced form

a. Definition :

A conditional control structure is called reduced simple form when processing depends on **a condition**. If the condition evaluates to "**true", processing is executed**.

b. Vocabulary and syntax :

Algorithm	**C code**
If (condition) **Then**	**If** (condition)
Instruction 1	Instruction 1 ;
End if	
If (condition) **Then**	**If** (condition)
Instruction 1	{
Instruction 2	Instruction 1 ;
………………..	Instruction 2 ;
Instruction n	………………..
End if	Instruction n ;
	}

Note : Instruction can be any action used in the algorithm (←, read, write, if, else, for, while , repeat).

1.1.2. The alternative form :

a. Definition:

A conditional control structure is considered an alternative form when processing depends on a condition **with two states**: if the condition is evaluated as "**true**", the **first process is executed**; if the condition is evaluated as "**false**", the **second process is executed**.

b. Vocabulary and syntax

Algorithm	**C code**
If (condition) **Then**	**If** (condition)
Instruction 1	Instruction 1 ;
Else	**else**
Instruction 2	Instruction 2 ;
End if	
	If (condition)
If (condition) **Then**	{
Instruction 1	Instruction 1 ;
Instruction 2	Instruction 2 ;
Else	}
Instruction 3	**else**
Instruction 4	{
End if	Instruction 3 ;
	Instruction 4 ;
	}

1.2. The generalized conditional control structure

a. Definition :

A conditional control structure is said to be generalised when it can solve problems involving more than two processes depending on the conditions. Execution of one process automatically causes the other processes not to be executed.

b. Vocabulary and syntax :

Algorithm	**C code**
If (condition 1) **Then**	**If** (condition)
Traitement 1	Traitement1 ;
Else	**else if (condition 2)**
If (condition 2) **Then**	Traitement2 ;
Traitement 2	**else if** (condition 3)
Else	Traitement3 ;
If (condition 3) **Then**	**else if**
Traitement 3	……………
Else	**else if** (condition n-1)
………….	Traitement n-1 ;
Else	**else**
If (condition n-1) **Then**	Traitement n ;
Traitement n-1	
Else	
Traitement n	
End if	
End if	
End if	
End if	
End if	

Remarks :

- It is preferable to put the most possible events first.
- Each treatment can include one or more instructions.

1.3. The conditional choice control structure

a. Definition:

A conditional control structure is said to be choice when the processing depends on the value that a selector will take. This selector is of integer, character or boolean type.

b. Vocabulary and syntax :

Algorithm	C code

According to selector **Do**
Value 1 : Action 1
Value 2 : Action 2-1
        Action 2-2
        Action 2-n
Value 3, Value 4: Action 4
Value 5 .. Value 7 : Action 6
…..
Value N : Action N
Else
Action R
End According to
	switch (selector)
	{
	case Value 1 : Action 1 ; break ;
	case Value 2 : Action 2-1 ;
	            Action 2-2 ;
	            Action 2-n ; break ;
	case Value 3 :
	case Value 4 : Action4 ; break ;
	case Value 5 :
	case Value 6
	case Value 7 : Action 5 ; break ;
	……..
	case Value N : Action N ; break ;
	default :
	Action R ;
	}

2. The Condition

The condition is a simple Boolean expression or a sequence of Boolean expressions (compound condition).
A simple condition is made up of a single comparison operator (=, ≠, <, ≤, >,≥).
A compound condition is a condition made up of several simple conditions linked by logical operators: AND, OR, exclusive OR (XOR) and NOT.

Examples:

- x between 2 and 6: (x >= 2) AND (x < =6)
- n divisible by 3 or 2: (n%3=0) OR (n%2=0)
- x is true and y is false: (x=true) and (not y)
- Two values and only two are identical among a, b and c: (a=b) XOR (a=c) XOR (b=c)

A compound condition is evaluated according to rules generally presented in **truth tables**.

Home training

Exercise 1

Write the algorithms that allow you to :
1- Solve a first-degree equation: ax+b=0, where a and b are real numbers given by the user.
2- Solve a second-degree equation: $ax^2+bx+c=0$, where a, b and c are real numbers given by the user.

Exercise 2

1) Write the algorithm that reads an angle value in degrees, converts it into radians and displays the result.
2) Also write the algorithm that reads an angle value in radians, converts it to degrees and displays the result.

Exercise 3

Write an algorithm that reads a certain time in seconds and then transforms it into hours, minutes and seconds.

Exercise 4

Write an algorithm that calculates a student's average for seven marks with coefficients 2,1,4,3, 1,4,2. Then display whether the student has been admitted or adjourned on the basis of this average.

Exercise 5

```
Algorithm exo5_1
Variable
a,b,c : integer
Begin
    Read(a,b)
    c←a+b
    If (c < 0) alors
    |   c← -(a+b)
    End If
    Write(c)
End
```

```
Algorithm exo5_2
Variable
a,b,c : integer
Begin
    Read(a,b)
    C ←a+b
    If (c < 0) alors
      Write( -c )
    End If
    If c ≥ 0 alors
    | Write( c )
    End If
End
```

```
Algorithm exo5_3
Variable
a,b,c : integer
Begin
    Read(a,b)
    c←a+b
    If (c < 0) alors
      Write( -c )
    Else
      Write( c )
    End If
End
```

1. Execute (give the execution trace) the algorithms exo5_1 , exo5_2 and exo5_3 for :
   - a= -10 , b = 5
   - a = 10, b = -29
   - a = 5, b = -2
   - a = 8 , b = 6
   - a = -4 , b = 9
2. What do these algorithms do?
3. What is the best solution (the best algorithm)?

Exercise 6

The inhabitants of a town pay tax according to the following rules:

- men over the age of 20 pay tax
- women pay tax if they are aged between 18 and 35
- all others pay no tax

Write an algorithm that asks the age and sex ('M' or 'F') of a resident and displays whether he or she is taxable. Test the algorithm with a 20-year-old woman, an 18-year-old man and a 35-year-old man.