Chapitre 3

Les structures conditionnelles

Sommaire

Objectif

L'objectif de ce chapitre est de construire des algorithmes comportant des traitements conditionnels.

I. Introduction

Une structure séquentielle est une suite d'instructions. Les instructions sont exécutées dans l'ordre l'une après l'autre.

Syntaxe :

```
Début
        Instruction 1 ;
        Instruction 2 ;
        …..
        Instruction n ;
Fin.
```

Exemple : l'algorithme qui affiche l'inverse d'un nombre réel.

```
ALGORITHME  INVERSE ;
VAR
        X, Y : REEL ;
DEBUT
        ECRIRE ("ENTREZ UNE VALEUR SVP ") ;
        LIRE (X) ;
        Y ← 1 /X ;
        ECRIRE ("L'INVERSE DE " , X, "EST", Y) ;
FIN.
```

Généralement, un algorithme ne comprend pas que des instructions séquentielles, mais il comprend aussi des instructions dites conditionnelles. L'algorithme ci-dessus calcule l'inverse d'un nombre réel. Si l'utilisateur tape la valeur 0 il y a une erreur d'exécution (division par 0), ce problème ne peut pas être résolu que par l'utilisation des structures conditionnelles.

Une instruction conditionnelle permet à un programme de modifier son traitement en fonction d'une condition. Les structures conditionnelles permettent de déterminer l'ordre dans lequel les instructions sont exécutées.

Dans les structures conditionnelles on se base sur ce qu'on appelle ***prédicat*** ou ***condition***. Ce dernier est un énoncé ou proposition qui peut être ***vrai*** ou ***faux.***

Dans ce qui suit, nous allons étudier les quatre formes d'instructions conditionnelles qui sont:

- Les structures conditionnelles simples
- Les structures conditionnelles composées
- Les structures conditionnelles imbriquées
- Les structures conditionnelles de choix multiple
- Les branchements

II. Structure conditionnelle simple : L'instruction « if..... »

Une structure de contrôle conditionnelle est dite à forme simple (la plus basique) lorsque le traitement dépend d'une condition. Si la condition est évaluée à « vrai », le traitement est exécuté. Dans cette structure, la non-satisfaction de la condition ne correspond à aucune action à faire.

Cette structure est subdivisée en deux parties : la condition et l'action.

Syntaxe :

Algorithme	langage C

SI <Condition> **ALORS**
< Bloc d'actions >
FINSI
	if (condition)
	{
	< Bloc d'actions >
	}

Où : <Condition> est une expression de condition dont l'évaluation donne une valeur logique, et le < Bloc d'actions > est un groupe d' instructions élémentaires.

- Si la condition est vérifiée (condition = vrai), les instructions du < Bloc d'actions > sont exécutées et on continue l'exécution des actions (instructions) situées après le Finsi.

- Si la condition n'est pas vérifiée (condition = faux), la partie < Bloc d'actions > à l'intérieur du Si n'est pas exécuté et on poursuit l'exécution de l'algorithme directement à partir de l'instruction qui suit le FinSi.

En langage C, si on ne met pas les accolades, le compilateur va considérer uniquement la première instruction comme instruction subordonnée à la structure if.

Exemple :

SI ((A-B) * C) = 12 **ALORS**
B ← 3;
C ← (X+2+B);
FINSI;

	<condition> = VRAI			**<condition> = FAUX**	
	Avant l'action	**Après l'action**		**Avant l'action**	**Après l'action**
A	5	5		2	2
B	2	3		2	2
C	4	8		4	4
X	3	3		5	5

Exemple en langage C:

```
if ( note >= 10)
   {
        printf (" vous êtes admis  ") ;
    }
```

Le formalisme de cette structure dans un organigramme est comme suit :

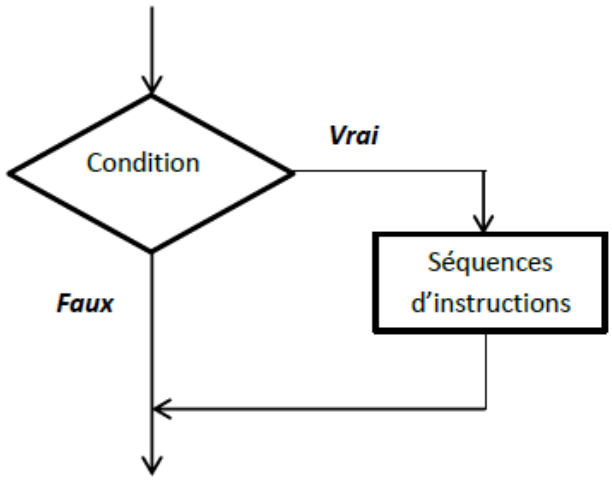


Figure 3.1 : Structure conditionnelle simple

III. Structure conditionnelle composée : L'instruction « if ... else »

Une structure de contrôle conditionnelle est dite composée ou à forme alternative lorsque le traitement dépend d'une condition à deux états: Si la condition est évaluée à « vrai », le premier traitement est exécuté, si la condition est évaluée à « faux », le second traitement est exécuté.

Syntaxe :

Algorithme	langage C

SI <Condition> **ALORS**
< Bloc d'actions 1 >
SINON
< Bloc d'actions 2 >
FINSI
	if (Condition)
	{
	Bloc d'actions 1
	}
	else
	{
	Bloc d'actions 2
	}

Elle s'exécute comme suit :

- Si la condition est vérifiée, les instructions du < Bloc d'actions1 > sont exécutées puis poursuite de l'exécution de l'algorithme à partir de l'instruction qui suit le Finsi.

- Si la condition n'est pas vérifiée, les instructions du < Bloc d'actions2 > sont exécutées puis poursuite de l'exécution de l'algorithme à partir de l'instruction qui suit le Finsi.

Exemple :

SI (A+B = 0) **ALORS**
A ⟵ (C-2);
SINON
B ⟵ (X*3);
A ⟵ (B-4);
FINSI;

	<condition> = VRAI (exécuté action(s)1)			<condition> = FAUX (exécuté Action (s)2)	
	Avant l'action	**Après l'action**		**Avant l'action**	**Après l'action**
A	3	0		5	2
B	-3	-3		2	6
C	2	2		4	4
X	1	1		2	2

Exemple en langage C:

```
int main ()
        {
        int valeur1 ;
        int valeur2 ;
        printf (" Entrez une 1ere valeur : ");
        scanf ("%d" ,& valeur1 );
        printf (" Entrez 2eme valeur : ");
        scanf ("%d" ,& valeur2 );
        if ( valeur1 < valeur2 )
                printf ("%d > %d\n",valeur2 , valeur1 );
        else
                printf ("%d >= %d\n",valeur1 , valeur2 );
        return 0;
        }
```

Le formalisme de cette structure dans un organigramme est comme suit :

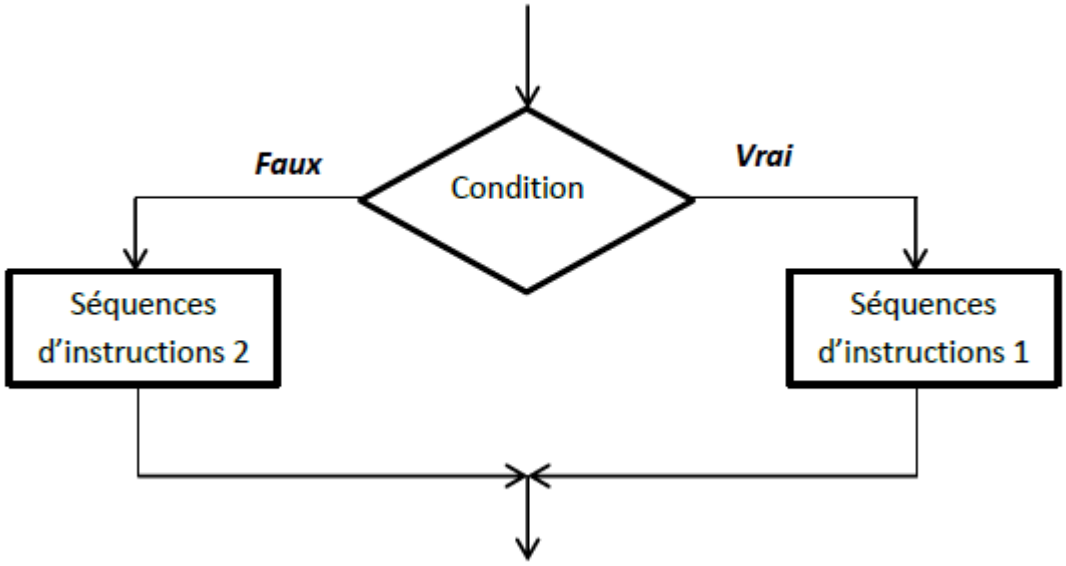


Figure 3.2 Structure conditionnelle composée

IV. Structures conditionnelles imbriquées

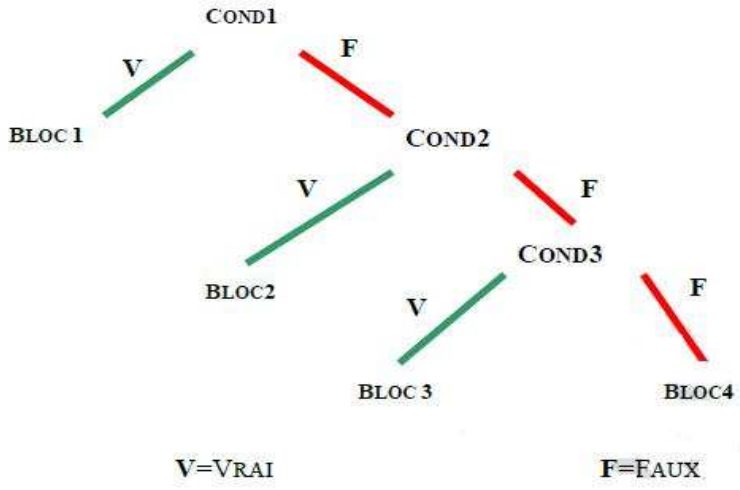
Il faut bien comprendre que les blocs d'instructions de ***(SI)*** et de ***(SINON)*** sont des séquences d'instructions. Ces blocs peuvent donc comporter des structures conditionnelles.

Par exemple, la deuxième branche d'une structure conditionnelle peut déboucher sur une nouvelle structure à deux branches, créant ainsi une structure conditionnelle à trois branches.

Syntaxe :

SI <Cond 1 > **ALORS**
< Bloc d'actions 1 >
SINON SI <Cond 2 > **ALORS**
< Bloc d'actions 2 >
SINON SI <Cond 3 > **ALORS**
< Bloc d'actions 3 >
SINON
< Bloc d'actions 4 >
FINSI
FINSI
FINSI

Fonctionnement :



Exemple en langage C:

```
int main ()
 {
int a = 0;
printf (" Saisie de a : ");
scanf ("%d" ,&a);
if (a < 0)   /* Strictement negative ? */
      printf ("la variable a est negative .\n");
else
{
if (a > 0) /* Strictement positive ? */
      printf ("la variable a est positive \n");
else /* Sinon a est nulle */
      printf ("la variable a est nulle \n");
}
return 0;
 }
```

V. Structure conditionnelle de choix multiple : L'instruction « switch »

C'est une extension du si ... alors ... sinon. Elle permet une programmation plus claire en évitant une trop grande imbrication de Si successifs. Une structure de contrôle conditionnelle est dite à choix multiple lorsque le traitement dépend de la valeur que prendra le sélecteur (variable).

Cette structure conditionnelle est appelée aussi ***sélective*** car elle sélectionne entre plusieurs choix à la fois, et non entre deux choix alternatifs (permet de faire plusieurs conditions sur une même variable). La structure globale est la suivante :

Syntaxe :

Algorithme	langage C
SELON Variable **FAIRE** **Cas val 1 :** liste d'instructions 1 **Cas val 2 :** liste d'instructions 2 ..... **Cas val n :** liste d'instructions n **SINON** traitement par défaut ; **FIN SELON**	**switch (** Variable **) {** **case Valeur1 : // Liste d' instructions ;** **break ;** **case Valeur2 :// Liste d' instructions ;** **break ;** **.....** **case Valeurs n ... : // Liste d' instructions;** **break ;** **default : // Liste d'instructions ;** **break ;** **}**

Le sélecteur doit être de type scalaire (dont le type est un entier, un caractère, un booléen, mais pas un réel ni une chaîne de caractères).

Les **CAS X** sont des constantes ordinales du même type que le sélecteur (sont des valeurs qui peuvent être prise par la variable sélecteur).

CAS X peut être :

- une valeur
- une suite de valeurs : 1, 3, 5, 7, 9
- une fourchette : 0 à 9
- une plage : >= 10

Si la valeur de « Variable » est égale à l'une des **CAS X**, la liste d'instructions correspondant est exécutée puis le programme sort de la structure. Sinon la liste d'instructions correspondant à « Sinon ou default » est exécutée. Au cas où l'action sinon ou default n'existe pas alors aucune action n'est exécutée.

Exemple en langage C:

```
void main ()
{
int choixMenu ;
printf (" ===== Menu =====\ n\n");
printf ("1. Royal Cheese \n");
printf ("2. Big Burger \n");
printf ("3. Complet Poulet \n");
printf ("4. Panini Thon \n");
printf ("\n Votre choix ? ");
scanf ("%d", & choixMenu );    /* Saisie du choix de l' utilisateur */
printf ("\n");
switch ( choixMenu ) /* Tester le choix de l' utilisateur */
{
case 1: printf (" Vous avez choisi un Royal Cheese !");
break ;
case 2: printf (" Vous avez choisi un Big Burger !");
break ;
case 3: printf (" Vous avez choisi un Complet Poulet !");
break ;
case 4: printf (" Vous avez choisi un Panini Thon !");
break ;
default : printf (" Choix incorrect . Vous ne mangerez rien !");
break ;
}
return 0;
}
```

Remarques :

- L'instruction default est facultative.
- Le mot clé break indique la sortie de la structure conditionnelle. N'oubliez pas d'insérer des instructions break entre chaque test, ce genre d'oubli est difficile à détecter car aucune erreur n'est signalée
- Ceci peut être utilisé judicieusement afin de faire exécuter les mêmes instructions pour différentes valeurs consécutives

VI. Une façon plus courte de faire un test :

Il est possible de faire un test avec une structure beaucoup moins lourde en langage C grâce à la structure suivante :

(condition) ? instruction si vrai : instruction si faux

Remarques :

- la condition doit être entre des parenthèses
- Lorsque la condition est vraie, l'instruction de gauche est exécutée
- Lorsque la condition est fausse, l'instruction de droite est exécutée
- En plus d'être exécutée, la structure ? : renvoie la valeur résultant de

Exemple :

```
( moyenne >=10) ? printf (" Admis ") : printf (" Ajourne ");

admis = (( moyenne >=10) ? 1 : 0);
```

Conclusion

Dans ce chapitre, nous avons présenté les structures conditionnelles, qui vont nous permettre de faire des tests et d'exécuter une ou plusieurs instructions dans un cas, d'autres instructions dans un autre cas. Selon la situation et le positionnement de problème, le développeur peut choisir entre plusieurs structures conditionnelles à savoir : les structures simples, les structures composées, les structures imbriquées, les structures de choix multiples ou les branchements.