

Les structures conditionnelles

A. Introduction

En programmation procédurale comme en algorithmique (qui respecte les contraintes fondamentales de la programmation!), **l'ordre des instructions est primordial.**

Le processeur exécute les instructions dans l'ordre dans lequel elles apparaissent dans le programme. On dit que l'exécution est **séquentielle.**

Une fois que le programme a fini une instruction, il passe à la suivante. Tant qu'une instruction n'est pas terminée, il attend avant de continuer. Par exemple, une instruction de saisie va attendre que l'utilisateur rentre une valeur au clavier avant de continuer. Parfois, il est nécessaire que le processeur n'exécute pas toutes les instructions, ou encore qu'il recommence plusieurs fois les mêmes instructions. Pour cela, il faudra casser la séquence. C'est le rôle des structures de contrôle.

Il existe deux grands types de structures de contrôle :

- les *structures conditionnelles* vont permettre de n'exécuter certaines instructions que sous certaines conditions ;
- les *structures répétitives*, encore appelées boucles, vont permettre de répéter des instructions un certain nombre de fois, sous certaines conditions.

B. Les structures conditionnelles

B.1. Présentation

Les structures conditionnelles permettent d'exécuter des instructions différentes en fonction de certaines conditions. Une condition (encore appelée expression conditionnelle ou logique) est évaluée, c'est à dire qu'elle est jugée vraie ou fausse. Si elle est vraie, un traitement (une ou plusieurs instructions) est réalisé; si la condition est fausse, une autre instruction va être exécutée, et ensuite le programme va continuer normalement.

Il existe 2 types principaux de structures conditionnelles :

- les *structures alternatives* (Si...Alors...Sinon) ;
- les *structures conditionnelles* au sens strict.

Dans le déroulement d'un algorithme, on doit souvent choisir entre deux actions, suivant une condition concernant la valeur de certaines données. La structure alternative va permettre d'effectuer des choix.



Supposons que nous ayons besoin, dans un programme, d'écrire un message précisant si la valeur d'une variable, nommée a, est positive ou négative. Pour cela on va utiliser la structure alternative :

```
...  
Afficher "entrez un nombre"  
Saisir n  
Si n > 0 Alors //dans le cas où l'expression n>0 est vraie  
    Afficher "valeur positive"  
Sinon //dans le cas où l'expression n>0 est fausse  
    Afficher "valeur négative ou nulle"  
Finsi  
...
```

Si la condition $n < 0$ mentionnée après le mot **Si** est vraie, on exécute ce qui figure après le mot **Alors** ; si la condition est fausse, on exécute ce qui figure après le mot **Sinon**. La syntaxe générale de cette structure est la suivante :

```
Si <condition> Alors  
    <traitement1>  
Sinon  
    <traitement2>  
Finsi
```

Pour une meilleure lisibilité du programme, on décale (on indente) le **Alors** et le **Sinon** par rapport au **Si**. On peut faire apparaître un trait vertical entre **Si** et **Finsi**.

Pour l'instant cela peut paraître superflu, mais en fait quand les programmes se compliquent, ces règles d'écriture facilitent grandement leur relecture.

Rappelons que les traitements apparaissant après les mots **Alors** et **Sinon** peuvent être constitués d'une instruction simple, comme dans notre premier exemple, mais aussi d'un ensemble d'instructions, appelé **bloc d'instruction**.

Exemple de structure alternative avec bloc d'instruction :

Nous voulons un programme qui mémorise et affiche la somme ou le produit de 2 nombres, suivant le choix de l'utilisateur. Ce programme doit saisir les deux nombres voulus ainsi que la lettre représentant l'opération à effectuer. Si la lettre est s (comme somme), il calcule et affiche la somme, et si la lettre est p (ou tout autre caractère), le programme doit calculer et afficher le produit.

```
Algorithme choix  
Variables  
nb1,nb2, res : entiers  
op : caractère  
Début  
Afficher "Entrez deux nombres"  
Saisir nb1, nb2  
Afficher "entrez la première lettre de l'opération voulue"  
Saisir op  
Si op = 's' Alors  
    res  $\leftarrow$  nb1 + nb2  
    afficher "la somme est :", res  
Sinon  
    res  $\leftarrow$  nb1 * nb2  
    Afficher "le produit est :", res  
Finsi  
Fin
```

B.2. Les expressions conditionnelles

Une expression conditionnelle (ou expression logique, ou expression booléenne) est une expression dont la valeur est soit VRAI soit FAUX. Il existe plusieurs types d'expressions conditionnelles.

B.2.1. Les comparaisons simples

Dans nos deux exemples, les conditions que nous avons rencontrées ($n > 0$) et ($op = "s"$) sont des conditions simples. Une condition simple est une comparaison de deux expressions de même type. ($n > 0$ type entier ou réel, $op = "s"$ type caractère).

Les symboles de comparaison utilisables en algorithmique sont : $<$, $>$, $<=$, $>=$, $=$, $<\tilde{>}$

Pour les comparaisons de caractères, on utilise l'ordre ASCII, qui respecte l'ordre alphabétique. Une lettre placée avant une autre dans l'ordre alphabétique sera inférieure à l'autre : "a" est inférieur à "b", mais "s" est supérieur à "m".

Attention, une condition simple ne veut pas dire une condition courte. Une condition simple peut être la comparaison de deux expressions comme : $(a + b - 3) * c = (5 * y - 2) / 3$

Exemple :

Supposons que nous voulions afficher la valeur absolue de la différence entre deux nombres entiers. Ces nombres entiers seront notés x et y. Nous voulons donc afficher $x - y$ si x est plus grand que y et $y - x$ sinon.

Nous écrivons pour ce faire :

Si $x > y$ Alors

Afficher $x - y$

Sinon

Afficher $y - x$

Finsi

B.2.2. Les conditions complexes

Les conditions (ou expressions conditionnelles) peuvent aussi être complexes, c'est à dire formées de plusieurs conditions simples ou variables booléennes reliées entre elles par les opérateurs logiques **et**, **ou**, **non**.

Exemples :

Si $a < 0$ et $b < 0$ Alors ...

Si $(a + 3 = b$ et $c < 0)$ ou $(a = c * 2$ et $b \tilde{>} c)$ Alors ...

Et :

Une condition composée de deux conditions simples reliées par 'et' est vraie si les deux conditions sont vraies.

La condition $a < 0$ et $b < 0$ est vraie si $a < 0$ est vraie et si $b < 0$ est vraie.

Ou :

Une condition composée de deux conditions simples séparées par ou est vraie si au moins l'une des conditions simples est vraie.

La condition $a < 0$ ou $b < 0$ est vraie si $a < 0$ ou si $b < 0$ ou si a et b sont négatifs.

Non :

Une condition précédée par non est vraie si la condition simple est fautive et inversement.

La condition non $(a < 0)$ est vraie si $a \geq 0$.

L'usage des parenthèses permet de régler d'éventuels problèmes de priorités des opérateurs logiques.

B.2.3. Les variables booléennes

Les variables booléennes, comme les expressions conditionnelles, sont soit vraies, soit fausses. On peut donc affecter une expression conditionnelle à un booléen et on peut aussi trouver une variable booléenne à la place d'une expression conditionnelle.

Les variables booléennes et les expressions conditionnelles sont équivalentes. A chaque fois que l'on peut trouver une expression conditionnelle, on peut aussi trouver une variable booléenne.

Exemple :

Algorithme intervalles

Variables

appartient : booléen

nb : réel

Début

Afficher "veuillez entrer un nombre réel"

Saisir nb

appartient = (nb < 10 ET nb > 5) OU (nb > 15 ET nb < 20)

Si appartient Alors

Afficher "Le nombre appartient aux intervalles définies"

Sinon

Afficher "Le nombre n'appartient pas aux intervalles définies"

Finsi

Fin

Ce programme saisit un nombre et affiche si ce nombre est compris dans les intervalles 5-10 ou 15-20

Application 1 :

Évaluer les expressions booléennes suivantes :

M	M
2	1

M > 2		
M <= 2		

M	R
2	1

M	R
3	-2

M > 2 et R = 1	
M > 2 ou R = 1	
M <= 3 et R <> 1	
M < 3 et R >= 0	

C. La structure Si...Alors (conditionnelle)

Cette structure est utilisée si on veut exécuter une instruction seulement si une condition est vraie et ne rien faire si la condition est fausse. Elle évite d'écrire **Sinon** rien. La syntaxe d'une structure conditionnelle est la suivante :

Si <condition> Alors

<traitement>

Finsi

Exemple :

Dans un programme de calcul d'une facture, on veut effectuer une remise de 1% si le montant de la facture dépasse 1000F. Supposons que la variable qui contient le montant de la facture s'appelle mont. On veut écrire l'algorithme qui affiche le montant à payer.

Si le montant est inférieur à 1000F, on veut juste afficher le montant tel quel. Mais si le montant est supérieur à 1000F, il faut prendre en compte la remise et calculer le nouveau montant.

Le morceau d'algorithme concerné est :

```
...
Si mont > 1000 Alors
    mont = mont * 0.9
Finsi
Afficher mont
```

Le programme effectue la réduction seulement si le montant est supérieur à 1000F. Sinon, il ne fait aucun traitement particulier et passe à l'instruction suivante. Dans tous les cas, le montant est affiché.

Application n°1 :

Écrivez le reste de l'algorithme donné en exemple ci-dessus.

Application 2 :

On considère l'algorithme suivant :

```
Algo trucchose
Variables
A, B, C, X : Entier
Début
Afficher "Entrer 1er nombre"
Saisir A
Afficher "Entrer 2nd nombre"
Saisir B
Afficher "Entrer 3ème nombre"
Saisir C
Si A>B Alors
    X ⇔ A
Sinon
    X ⇔ B
Finsi
Si C>X Alors
    X ⇔ C
Finsi
Fin
```

Terminer les exécutions de cet algorithme et en déduire son objectif.

Instructions :	Mémoire :	Expression booléenne								
	<table border="1" style="display: inline-table; margin-right: 10px;"> <tr><td style="text-align: center;">A</td></tr> <tr><td style="text-align: center;">15</td></tr> </table> <table border="1" style="display: inline-table; margin-right: 10px;"> <tr><td style="text-align: center;">B</td></tr> <tr><td style="text-align: center;">18</td></tr> </table> <table border="1" style="display: inline-table; margin-right: 10px;"> <tr><td style="text-align: center;">C</td></tr> <tr><td style="text-align: center;">3</td></tr> </table> <table border="1" style="display: inline-table;"> <tr><td style="text-align: center;">X</td></tr> <tr><td style="text-align: center;"> </td></tr> </table>	A	15	B	18	C	3	X		
A										
15										
B										
18										
C										
3										
X										
<u>Si</u> A>B <u>Alors</u>										

Instructions :	Mémoire :	Expression booléenne								
	<table style="display: inline-table; border: none;"> <tr> <td style="border: none; padding: 0 10px;">A</td> <td style="border: none; padding: 0 10px;">B</td> <td style="border: none; padding: 0 10px;">C</td> <td style="border: none; padding: 0 10px;">X</td> </tr> <tr> <td style="border: 1px solid black; text-align: center; width: 40px; height: 30px;">12</td> <td style="border: 1px solid black; text-align: center; width: 40px; height: 30px;">5</td> <td style="border: 1px solid black; text-align: center; width: 40px; height: 30px;">17</td> <td style="border: 1px solid black; width: 40px; height: 30px;"></td> </tr> </table>	A	B	C	X	12	5	17		
A	B	C	X							
12	5	17								
<u>Si A>B Alors</u>										

Application 3 :

L'entreprise Ferdo accorde une prime d'ancienneté à chaque employé dont l'ancienneté est supérieure ou égale à 5 ans. Cette prime est calculée de la manière suivante :

- Si l'ancienneté est comprise entre 5 et 10 ans on lui accorde une prime dont le taux est de 5% de son salaire de base.
- Si l'ancienneté est supérieure ou égale à 10 ans, l'employé perçoit une prime dont le taux s'élève à 10% de son salaire de base.

Pour quelles valeurs de Ancien, l'algorithme suivant donne-t-il un taux correct ?

Algorithme ancienneté

Variables

Ancien, Taux : Entier

Début

Afficher "Ancienneté ?"

Saisir Ancien

Si Ancien <5 Alors

Taux \leftarrow 0

Finsi

Si Ancien <10 Alors

Taux \leftarrow 5

Finsi

Si Ancien \geq 10 Alors

Taux \leftarrow 10

Finsi

Fin

Le fait de permuter les deux premières alternatives suffit-il à le rendre correct ?

Application 4 :

Une variable est dite booléenne lorsqu'elle peut prendre seulement deux états : 0 ou 1. Le complémentaire de X est noté \bar{X} est donné dans la table de vérité suivante :

X	\bar{X}
0	1
1	0

L'algorithme suivant réalise t-il le complémentaire de la case A ?



Olivier Mondet
<http://unidentified-one.net>

Algo compl
Variables
 A : Entier
Début
Saisir A
Si A=0 Alors
 A \leftarrow 1
Finsi
Si A=1 Alors
 A \leftarrow 0
Finsi
Fin

Application 5 :

Écrire l'algorithme permettant de résoudre l'équation : $ax + b = 0$

Les données d'entrée seront placées dans deux cases A et B. Le résultat sera rangé dans X (B est supposé différent de 0).

D. L'imbrication des Structures Si

Les structures conditionnelles peuvent être imbriquées (c'est-à-dire incluses les unes dans les autres).

Voilà un programme qui indique le taux de remise selon le montant d'un achat :

Algo taux
Variables
 montant, taux : **entiers**
Début
Si montant < 1000 Alors // Structure 1
 taux \leftarrow 1
Sinon
 Si montant < 3000 Alors // Structure 2
 taux \leftarrow 2
 Sinon
 Si montant < 10000 Alors // Structure 3
 taux \leftarrow 3
 Sinon
 taux \leftarrow 4
 Finsi //(3)
 Finsi //(2)
Finsi //(1)
 Montant \leftarrow montant - montant * taux/100
Afficher montant
Fin

Si le montant est inférieur à 1000, la valeur 1 est affectée au taux et le programme passe à l'instruction suivante. Attention, l'instruction suivante n'est pas celle qui suit **Sinon**.

Si la condition est vérifiée, tout ce qui suit le **Sinon** est ignoré, y compris les **Si** qui sont imbriqués dans le **Sinon**. Donc le programme passe à l'instruction suivante qui est montant \leftarrow montant - montant * taux/100.

En ce qui concerne la présentation, nous avons décalé vers la gauche les structures imbriquées (indentation). C'est une convention d'écriture qu'on retrouve pour toutes les structures, et qui a pour but de rendre l'algorithme plus lisible.

Les structures imbriquées sont emboîtées telles des poupées russes. Il est impossible qu'elles se chevauchent. Cela est valable pour les structures conditionnelles et pour toutes les autres structures. Dans ce cas, le premier **FinSi** rencontré dans l'algorithme indique toujours la fin de la structure la plus imbriquée. Il est donc inutile de numéroter les structures pour reconnaître où elles se terminent.

La plupart du temps, la structure Selon (cf. plus bas) va permettre d'éviter l'imbrication des structures Si. Pourtant, il est indispensable de savoir les utiliser pour deux raisons :

- dans les langages C++ et dans d'autres langages, la Structure équivalente à Selon ne permet pas d'utiliser des intervalles de valeur.
- il est parfois impossible d'utiliser la structure Selon, en particulier lorsque les conditions portent sur plusieurs variables.

Application 1 :

On considère l'algorithme suivant :

Algo gogo

Variables

Nb1, Nb2 : Entier

R : Chaîne de caractères

Début

Afficher "1er nombre ?"

Saisir Nb1

Afficher "2nd nombre ?"

Saisir Nb2

Si Nb1 < 0 Alors

Si Nb2 < 0 Alors

 R ← "+"

Sinon

 R ← "-"

Finsi

Sinon

Si Nb2 > 0 Alors

 R ← "+"

Sinon

 R ← "-" (a)

Finsi

Finsi

Afficher R

Fin

Écrire l'expression booléenne qui doit être vérifiée pour que l'exécution de l'instruction (a) se produise.

Que représente R ?



Olivier Mondet
<http://unidentified-one.net>

Application 2 :

Soit l'algorithme ci-après :

Algo Idorack

Variables

X, Y, Z : Entier

Début

Saisir X

Saisir Y

Si $X \geq Y$ et $X > 0$ Alors

 Z \leftarrow 5

Sinon

 Z \leftarrow -5

Finsi

Si $X > Z$ Alors

Si $Y \leq Z$ Alors

 Z \leftarrow Z + X

Sinon

 Z \leftarrow Z - X

Finsi

 Z \leftarrow Z + 2

Finsi

Afficher Z

Fin

Complétez le tableau suivant :

X	25	0	5	-4	0	10
Y	8	-1	4	-6	0	10
Z						

Application 3 :

La compagnie aérienne "Omega Air Lines" asperge les récoltes pour contrôler certains fléaux. Ses tarifs dépendent du produit utilisé et de la superficie couverte, et sont fixés comme suit :

Catégorie n°1 : Traitement des mauvaises herbes, 15 € l'hectare ;

Catégorie n°2 : Traitement des sauterelles, 30 € l'hectare ;

Catégorie n°3 : Traitement des vers, 45 € l'hectare ;

Catégorie n°4 : Traitement combiné, 80 € l'hectare.

Pour un traitement de plus de 1000 hectares, le fermier reçoit un rabais de 5% du prix total. De plus, si le montant de la facture excède 2500 €, une remise de 10% est accordée. Si les deux réductions s'appliquent on calcule d'abord le rabais.

Écrire l'algorithme qui fournit le prix à payer ; les données d'entrée sont :

- Le numéro de la catégorie de traitement,
- Le nombre d'hectares à traiter.

Faire la trace de l'algorithme pour les valeurs suivantes : catégorie n°4 pour 1200 hectares.

E. La structure Selon...Faire (de choix)

La structure Selon permet de choisir le traitement à effectuer en fonction de la valeur ou de l'intervalle de valeur d'une variable ou d'une expression. Cette structure permet de remplacer avantageusement une succession de structures Si...Alors.

La syntaxe de cette structure est :

Selon *expression* Faire

valeur 1 de l'expression : *traitement 1*

valeur 2 de l'expression : *traitement 2*

valeur 3 de l'expression : *traitement 3*

...

[Sinon *traitement par défaut*]

Finselon

Exemple1 : Les différents cas sont des valeurs littérales.

Voilà l'algorithme qui affiche le mois en toute lettre selon son numéro. Le numéro du mois est mémorisé dans la variable mois.

...

Selon mois Faire

1 : Afficher "Janvier"

2 : Afficher "Février"

3 : Afficher "Mars"

4 : Afficher "Avril"

...

11: Afficher "Novembre"

12: Afficher "Décembre"

Sinon Afficher "Un numéro de mois doit être compris entre 1 et 12"

Finselon

Exemple 2 : Les différents cas possibles sont décrits par des intervalles de valeurs (taux de remise différent selon le montant d'achat)

...

Selon montant Faire

<1000 : taux ⇔ 1

>=1000 et < 3000 : taux ⇔ 2

>=3000 et < 10000 : taux ⇔ 3

>=10000 : taux ⇔ 4

FinSelon

montant ⇔ montant * (1 - taux/100)

...



Olivier Mondet
<http://unidentified-one.net>