Algorithmique et Programmation :
Chap 4 : Les Structures Conditionnelles

E. M. Souidi

Faculté des Sciences - Rabat
SVI4 –STU4 2013-14

Les strucrures conditionnnelles ou tests sont des éléments essentiels à tout langage de programmation.

Les strucrures conditionnnelles ou tests sont des éléments essentiels à tout langage de programmation.
Elles permettent d'exécuter une instruction ou un bloc d'instructions si une condition est vraie ou fausse.

Une condition est une expression logique. Elle est formée de deux valeurs et d'un opérateur de comparaison

Opérateurs de comparaison

Les opérateurs de comparaison numérique sont :

Opérateurs de comparaison

Les opérateurs de comparaison numérique sont :
$=$ égal,

Opérateurs de comparaison

Les opérateurs de comparaison numérique sont :
$=$ égal,
$<=$ inférieur ou égal,

Opérateurs de comparaison

Les opérateurs de comparaison numérique sont :
$=$ égal,
$<=$ inférieur ou égal,
$>=$ supérieur ou égal,

Opérateurs de comparaison

Les opérateurs de comparaison numérique sont :
$=$ égal,
$<=$ inférieur ou égal,
$>=$ supérieur ou égal,
$<$ trictement inférieur,

Opérateurs de comparaison

Les opérateurs de comparaison numérique sont :
$=$ égal,
$<=$ inférieur ou égal,
$>=$ supérieur ou égal,
$<$ trictement inférieur,
$>$ trictement supérieur,

Opérateurs de comparaison

Les opérateurs de comparaison numérique sont :
$=$ égal,
$<=$ inférieur ou égal,
$>=$ supérieur ou égal,
$<$ trictement inférieur,
$>$ trictement supérieur,
$!=$ différent,

On les note tels quels en pseudo code.

Ces opérateurs de comparaison s'emploient avec les chaines de caractères aussi.

Ces opérateurs de comparaison s'emploient avec les chaines de caractères aussi.
Car les caractères sont codés par la machine dans l'ordre alphabétique, les majuscules étant systématiquement placées avant les minuscules. Ainsi on a :

Exemple

```
"A" < "a" est Vrai
"S" < "F" est Faux
```

Opérateurs logiques

En informatique il y en a 4 : `ET`, `OU`, `XOR` et `NON`. Soient Cond1 et Cond2 deux conditions. Table de vérité de `ET` :

Cond1	Cond2	Cond1 ET Cond2
VRAI	VRAI	VRAI
FAUX	VRAI	FAUX
VRAI	FAUX	FAUX
FAUX	FAUX	FAUX

Table de vérité de `OU` :

Table de vérité de `OU` :

Cond1	Cond2	Cond1 OU Cond2
VRAI	VRAI	VRAI
FAUX	VRAI	VRAI
VRAI	FAUX	VRAI
FAUX	FAUX	FAUX

Table de vérité de `XOR` :

Table de vérité de `XOR` :

Cond1	Cond2	Cond1 XOR Cond2
VRAI	VRAI	FAUX
FAUX	VRAI	VRAI
VRAI	FAUX	VRAI
FAUX	FAUX	FAUX

Table de vérité de `XOR` :

Cond1	Cond2	Cond1 XOR Cond2
VRAI	VRAI	FAUX
FAUX	VRAI	VRAI
VRAI	FAUX	VRAI
FAUX	FAUX	FAUX

Table de `NON` :

Table de `NON` :

Cond1	NON (Cond1)
VRAI	FAUX
FAUX	VRAI

Booléen

Un booléen est une expression logique dont la valeur est VRAI ou FAUX.

Booléen

Un booléen est une expression logique dont la valeur est VRAI ou FAUX.
Une expression peut être une condition unique ou plusieurs conditions composées avec des opérateurs logiques `ET`, `OU`, `NON` ou `XOR`.

Un booléen est soit une variable de type booléen, soit une condition expression logique.

Un booléen est soit une variable de type booléen, soit une condition expression logique.
Exemple : $(3 > 7)$ et $(4 > 2)$

Un booléen est soit une variable de type booléen, soit une condition expression logique.
Exemple : $(3 > 7)$ et $(4 > 2)$
Il n'y a que deux formes possibles pour une structure conditionnelle qui s'écrivent en pseudo code.

Un booléen est soit une variable de type booléen, soit une condition expression logique.
Exemple : $(3 > 7)$ et $(4 > 2)$
Il n'y a que deux formes possibles pour une structure conditionnelle qui s'écrivent en pseudo code.

Formes conditionnelles

Forme complète	Forme simple

`Si` booléen `Alors`
Bloc1 d'instructions
`Sinon`
Bloc2 d'instructions
`Finsi`
	`Si` booléen `Alors`
	Bloc d'instructions
	`Finsi`

Dans la forme complète si booléen est `VRAI`, alors c'est Bloc1 d'instructions qui est exécuté et l'exécution passe directement aux instructions se trouvant après `Finsi`.

Dans la forme complète si booléen est `VRAI`, alors c'est Bloc1 d'instructions qui est exécuté et l'exécution passe directement aux instructions se trouvant après `Finsi`.
Si booléen est `FAUX` alors Bloc1 d'Instructions n'est pas exécuté, c'est Bloc2 d'instructions qui sera exécuté.

Dans le cas de la forme simple, si booléen est `VRAI` alors bloc d'instructions est exécuté. Par contre si booléen est `FAUX` alors l'exécution passe directement aux instructions se trouvant après `Finsi`.

Formes conditionnelles

Les tests qu'on vient de voir permettent de réaliser, au mieux, un choix parmi deux possibilités.

Formes conditionnelles

Les tests qu'on vient de voir permettent de réaliser, au mieux, un choix parmi deux possibilités.
Parfois on a plusieurs cas. Par exemple :

Formes conditionnelles

Les tests qu'on vient de voir permettent de réaliser, au mieux, un choix parmi deux possibilités.
Parfois on a plusieurs cas. Par exemple :
Si la moyenne est entre 10 et 12, attribuer la mention Passable

Si la moyenne est entre 12 et 14, attribuer la mention Assez Bien
Si la moyenne est entre 14 et 16, attribuer la mention Bien
Si la moyenne est entre 16 et 18, attribuer la mention très bien
Si la moyenne est entre 18 et 20, attribuer la mention excellent

Dans de telles cas on utilise le concept de conditions multiples qui permettent de comparer un objet à toute une série de valeurs et d'exécuter en fonction de la valeur effective de l'objet le bloc d'instruction qui convient.

Une première solution

```
Variable moy : réel
Début
Ecrire "Entrez la moyenne de l'étudiant "
Lire (moy)
Si moy >=10 et moy < 12  Alors
    Ecrire "mention est Passable "
Finsi
Si moy >=12 et moy<14 Alors
    Ecrire "mention est Assez bien"
Finsi
```

```
Si moy >=14 et  moy<16  Alors
    Ecrire "mention est Bien "
Finsi
Si moy >=16 et moy<18 Alors
    Ecrire "mention Très Bien"
Finsi
Si moy >=16 et moy<18 Alors
    Ecrire "mention est Excellent"
Finsi
Fin
```

Ce n'est pas une bonne solution, car la machine est obligée de vérifier les quatre conditions même si la première est vraie. Donc plus de temps d'exécution.

Une bonne solution

Cette solution consiste à imbriquer les tests :

Une bonne solution

Cette solution consiste à imbriquer les tests :
Au lieu d'effectuer toutes les quatre conditions de la première solution, nous avons dans cette solution aux maximum quatre conditions.

De plus si le premier test est vrai, l'exécution passe à la suite du dernier Finsi. D'où un gain de temps.

De plus si le premier test est vrai, l'exécution passe à la suite du dernier Finsi. D'où un gain de temps.

Il est donc pratique de placer en premier la condition qui a plus de probabilité pour être VRAI. Cette deuxième version n'est donc pas seulement plus simple à écrire et plus lisible, mais elle est également plus performante à l'exécution.

Les structures de tests imbriqués sont donc un outil indispensable à la simplification et à l'optimisation des algorithmes.

Simplification : Dans le cas de tests imbriqués, le `Sinon` et le `Si` peuvent être fusionnés en un `SinonSi`. On considère alors qu'il s'agit d'un seul bloc de test, conclu par un seul `FinSi`

```
Ecrire ("Entrez la moyenne : ")
Lire (moy)
Si moy>=10 et moy<12 Alors
     Ecrire ("Mention est passable")
SinonSi moy >=12 et moy <14 Alors
     Ecrire ("Mention est Assez bien")
SinonSi moy>=14 et moy <16 Alors
     Ecrire ("Mention est Bien")
```

```
Sinon
     Ecrire "Mention est Excellent "
Finsi
Fin
```

Tests en Python

Un test prend la forme générale suivante :

Tests en Python

if booléen_1 :

Tests en Python

```
if booléen_1  :
        bloc instructions1
```

Tests en Python

```
if booléen_1  :
        bloc instructions1
elif booléen_2  :
```

Tests en Python

```
if booléen_1  :
        bloc instructions1
elif booléen_2  :
        bloc instructions2
```

Tests en Python

```
if booléen_1  :
        bloc instructions1
elif booléen_2  :
        bloc instructions2
```

. . . *(autant de elif qu'il faut)*

Tests en Python

```
if booléen_1  :
        bloc instructions1
elif booléen_2  :
        bloc instructions2
... (autant de elif qu'il faut)
elif booléen_n  :
```

Tests en Python

```
if booléen_1  :
        bloc instructions1
elif booléen_2  :
        bloc instructions2
... (autant de elif qu'il faut)
elif booléen_n  :
        bloc instructionsn
```

Tests en Python

```
if booléen_1  :
        bloc instructions1
elif booléen_2  :
        bloc instructions2
... (autant de elif qu'il faut)
elif booléen_n  :
        bloc instructionsn
else  :
```

Tests en Python

```
if booléen_1  :
        bloc instructions1
elif booléen_2  :
        bloc instructions2
... (autant de elif qu'il faut)
elif booléen_n  :
        bloc instructionsn
else  :
        autre bloc instructions
```

elif est une contaraction de else if. On peut utiliser les elif ou non. else est optionnelle.

elif est une contaraction de else if. On peut utiliser les elif ou non. else est optionnelle.
Les deux points (à ne pas oublier !) indiquent que l'instruction n'est pas encore terminée. Dans ce cas la touche entrée est suivie de ... et non >>>

Le passage à la ligne après une ligne vide indique à Python la fin du bloc. C'est donc `FinSi`.

Exemple

```
x=input('introduisez un nombre : ')
y=input('introduisez un nombre : ')
if x>y :
        print x, 'est le plus grand'
elif x==y :
        print 'sont egaux'
else :
        print y, 'est le plus grand'
```