Chapitre 3

Instructions de contrôle

Opérateurs de comparaison et opérateurs logiques

Les opérateurs de comparaison sont :

```
==    : égal à
>     : strictement superieur (x > y)
<     : strictement inferieur (x < y)
>=    : superieur ou égal à (x >= y)
<=    : inferieur ou égal à (x <= y)
~ =   : différent de (x ~ = y)
&     : et (x & y)
|     : ou (x | y)
~     : non (~ x)
```

Les opérateurs de comparaison et les opérateurs logiques sont utilisés essentiellement dans les instructions de contrôle.

Boucle FOR «POUR» : parcours d'un intervalle

Pour exécuter une séquence d'instructions de manière répétée consiste à effectuer une boucle pour les valeurs d'un indice (variable), incrémenté à chaque itération, variant entre deux bornes données. Ce processus est mis en oeuvre par la boucle for.

Syntaxe :

```
for indice=borne_inf:borne_sup
suite d'instructions
end
```

où

- indice est une variable appelée l'indice de la boucle
- *borne_inf* et *borne_sup* sont deux constantes (appelées paramètres de la boucle)
- suite d'instructions est le traitement à effectuer pour les valeurs d'indices variant entre *borne_inf* et *borne_sup* avec un incrément de 1. On parle du corps de la boucle.

Remarque 1 *Interprétation :*

— *Si borne_inf est <= à borne_sup, le traitement séquence d'instructions est exécuté borne_sup− borne_inf fois pour les valeurs de la variable indice égales à* $borne\_inf, borne\_inf+1, \cdots, borne\_sup$.

— *Si borne_inf est > à borne_sup, on passe à l'instruction qui suit l'instruction de fin de boucle (end)*

Remarque 2 *L'indice de boucle ne prend pas nécessairement des valeurs entières. D'autre part il n'est pas nécessaire que l'indice de la boucle apparaisse dans le corps de la boucle ; par contre il est*

interdit de modifier sa valeur s'il apparaît. Il est possible d'imbriquer des boucles mais elles ne doivent pas se recouvrir. On peut utiliser un incrément (pas) autre que 1 (valeur par défaut). La syntaxe est alors borne_inf : pas : borne_sup. Le pas peut être négatif.

Exemple 1

```
for r=1.1:-0.1:0.75
disp(['r = ', num2str(r)]);
end
```

Exemple 2

```
n = 4;
nfac = 1;
for k = 1:n
nfac = nfac*k;
end
```

Boucle WHILE : tant que . . . faire

Pour exécuter une séquence d'instructions de manière répétée consiste à effectuer une boucle tant qu'une condition reste vérifiée. On arrête de processus dès que cette condition n'est plus satisfaite. Ce processus est mis en oeuvre par la boucle while.

Syntaxe :

```
while expression logique
suite d'instructions
end
```

où

- expression logique est une expression renvoyant vrai ou faux, évaluée avant d'effectuer chaque tour de boucle
- suite d'instructions est le traitement à effectuer tant que expression logique est vraie

Remarque 3 *Interprétation :*

— *Tant que expression logique est vraie le traitement suite d'instructions est exécuté sous forme d'une boucle*

— *Lorsque expression logique devient faux, on passe à l'instruction qui suit l'instruction de fin de boucle (end).*

Remarque 4 *L'expression logique est en général le résultat d'un test. Il est impératif que le traitement de la suite d'instructions agisse sur le résultat de expression logique sans quoi on boucle indéfiniment*

Exemple 3

```
 n = 4;
k = 1; nfac = 1;
while k <= n
nfac = nfac*k;
k = k+1;
end
```

L'instruction conditionnée IF

L'instruction conditionnée la plus simple a la forme suivante :
Syntaxe :

```
if expression logique
suite d'instructions
end
```

L'instruction conditionnelle 'if condition suite d'instructions end' n'exécute l'instruction que si la condition est évaluée à true(c'est-à-dire vaut 1). La condition doit être une expression logique évaluée à vrai ou faux. Par exemple :

Exemple 4

```
if (x < 10)
disp(x);           % n'affiche x que lorsque x <10
end
if ((0 < x) & (x < 100))
disp('OK');         % affiche OK si x entre 0 et 100
end
```

Vous pouvez mettre plus d'une instruction entre le «si» et le «fin». Vous pouvez choisir entre deux plans d'action avec 'if condition suite d'instructions 1 else suite d'instructions 2 end' comme dans l'exemple suivant :

Exemple 5

```
if ((0 < x) & (x < 100))
disp('OK');
else
disp('x contient un nombre invalide');
end
```

Vous pouvez construire une chaîne de tests avec 'elseif', comme dans l'exemple suivant :

Exemple 6

```
if (n <= 0)
disp('n est négatif ou zéro');
elseif (rem(n,2)==0)
disp('n est pair');
else
disp('n est impair');
end
```

Exercice 1 *Ecrivez un programme qui demande à l'utilisateur un âge, puis classe l'âge selon le schéma suivant :*
Erreur <0 <= Bébé <1 <= Enfant <13 <= Adolescent <18 <= Adulte <60 <= Senior <120 <= Erreur

Solution 1

```
age=input('Enter un age : ');
if ((age < 0)|(age >= 120))
disp('erreur');
elseif (age < 1)
disp('Bébé');
elseif (age < 13)
disp('Enfant');
elseif (age < 18)
disp('adolescent');
elseif (age < 60)
disp('Adulte');
else
disp('Senior');
end
```

Choix multiple l'instruction switch

Une alternative à l'utilisation d'une suite d'instructions conditionnées pour effectuer un choix ventilé existe. Il s'agit de l'instruction switch.

Syntaxe :

```
switch var
case val 1,
Suite d'instructions 1
case val 2,
Suite d'instructions 2
...
case val N,
Suite d'instructions N
otherwise
Suite d'instructions par défaut
end
```

Exemple 7

```
num=input('donner un nombre eentre 1 et 4)
switch num
case 1,
A = ones(n)
case 2,
A = magic(n);
case {3,4},
A = rand(n);
otherwise
error('numero d''exemple non trouvé ...');
end
```