PROGRAMMER SOUS MATLAB

V. INSTRUCTIONS DE CONTRÔLE :

V.1. Operateurs De Comparaison Et Operateurs Logiques :

Ils sont utilisés essentiellement dans les instructions de contrôle :

Opérateurs de comparaison :

== : égal à (x==y)
> : strictement plus grand que (x>y)
< : strictement plus petit que (x<y)
>= : plus grand ou égal à (x>y)
<= : plus petit ou égal à (x<y)
~= : différent de (x~=y)

Les opérateurs logiques sont :

& : et (x&y)
| : ou (x|y)
~ : non (~x)

V.2. l'instruction conditionnée IF:

Permet d'exécuter une séquence d'instructions seulement dans le cas où une condition donnée est vérifiée au préalable.

Syntaxe :

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

ou

```
if expression logique
      séquence d'instructions 1
else
      séquence d'instructions 2
end
```

ou encore

```
if expression logique 1
      séquence d'instructions 1
elseif expression logique 2
      séquence d'instructions 2
      .
      .
elseif expression logique N
      séquence d'instructions N
else
      séquence d'instructions par défaut
end
```

V.3. Choix ventilé, l'instruction switch:

Permet l'utilisation d'une séquence d'instructions conditionnées pour effectuer un choix en cascade.

Syntaxe :

```
switch var
      case cst_1
      séquence d'instructions 1
      case cst_2
      séquence d'instructions 2
      .
      .
      case cst_N
      séquence d'instructions N
otherwise
      séquence d'instructions par défaut
end
```

V.4. Boucle FOR (pour … faire):

Permet d'exécuter une séquence d'instructions de manière répétée pour les valeurs d'un indice, incrémenté à chaque itération, variant entre deux bornes données.

Syntaxe :

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

avec :

- `indice` est une variable appelée l'indice de la boucle ;
- `borne_inf` et `borne_sup` sont deux constantes réelles (paramètres de la boucle) ;
- `séquence d'instructions` est le traitement à effectuer pour les valeurs d'indice variant entre `borne_inf` et `borne_sup` avec un incrément de 1 (le corps de la boucle).

Remarque :
Pour utiliser un incrément (pas) autre que 1, on écrit : **`borne_inf:pas:borne_sup`**

Exemple :

Calculer n! (i=1,..,4)

```
>>n=4 ;
>> nfact=1
>>for i=1 :n
      nfact=nfact*i ;
    end
>>nfact
nfact=
24
>>
```

V.5. Boucle WHILE (tant que … faire):

Consiste à effectuer une boucle tant qu'une condition reste vérifiée. On arrête la boucle dès que cette condition n'est plus satisfaite.
Syntaxe :

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

avec :

- `expression logique` est une expression dont le résultat peut être vrai ou faux ;
- `séquence d'instructions` est le traitement à effectuer tant que `expression logique` est vraie.

Exemple :

Calculer n! (i=1,..,4)

```
>>n=4 ;
>> i=1 ;nfact=1
>>while i<=n
      nfact=nfact*i ;
      i=i+1 ;
    end
>>nfact
nfact=
24
>>
```

VI. SCRIPTS ET FONCTIONS :

MATLAB offre la possibilité d'enregistrer une séquence d'instructions dans un fichier, appelé **M-file**, et de les faire exécuter. Ce fichier doit avoir une extension de la forme **.m**. on distingue deux types de M-files :

VI.1. Les fichiers de scripts :

Un script est un ensemble d'instructions MATLAB qui joue le rôle de programme principal. Si le script est écrit dans le fichier nommé NOM.m on l'exécute dans la fenêtre MATLAB en tapant NOM.

VI.2. Les fichiers de fonctions :

Ils permettent de définir des fonctions qui ne figurent pas parmi les fonctions MATLAB incorporées et de les utiliser de la même manière que ces dernières. On définit la fonction **fonc** comme suit :

function [vars1,…,varsn]=fonc(vare1,…,varem)
séquence d'instructions

avec :
- vars1,…,varsn : variables de sortie de la fonction ;
- vare1,…,varem : variables d'entrée de la fonction ;
- séquence d'instructions est le corps de la fonction.

Remarques :

1. Le fichier doit impérativement commencer par le mot-clé **function**.
2. Si la fonction ne possède qu'une seule variable de sortie, les crochets sont inutiles.
3. Il est impératif que la fonction ayant pour nom **fonc** soit enregistrée dans un fichier nommé **fonc.m**

Exemple :

```
   function h=pyt(a,b)
% PYT hypoténuse d'un triangle rectangle par le théorème
% de pythagore. Entrée : les cotés du triangle.
h=sqrt(a^2+b^2) ;
```

Après avoir enregistrer le fichier pyt.m, on saisie :

```
>> pyt(3,4)
     ans=
     5
```