Université Ahmed Zabana de relizane
Faculté des sciences et technologies
2ème année L2-ST
Module : Méthodes Numériques (travaux pratiques)

TP 1 : Instructions de contrôle
(Boucles: for et while, Instructions: if, else et switch)

1. Opérateurs de comparaison et opérateurs logiques

a) Les opérateurs de comparaison sont:

= = : égale à (X= = Y)
> : Strictement plus grand que (X > Y)
< : Strictement plus petit que (X< Y)
>= : plus grand ou égale à (X>= Y)
<= : plus petit ou égale à (X<= Y)
~= : différent de (X~=Y)

b) Les opérateurs logiques sont:

& : et (X &Y)
| : ou (or) (X | Y)
- : Non X (-X)

2. Instructions de contrôle

Les instructions de contrôle sous matlab sont très proches de celles existant dans d'autres langages de programmation.

2.1 Les boucles (traitement répétitif)

2.1.2 La Boucle FOR : parcours d'un intervalle **(pour … faire)**

On peut répéter des actions grâce aux boucles, la boucle **for** permet de changer la valeur d'une variable de manière régulière.
La syntaxe pour la boucle **for** est la suivante :

for indice (variable) = borne_inf (valeur début) : pas : borne_sup (valeur fin)

    Séquence d'instructions
end

Exemple 1 : faire un programme Matlab qui calcule la somme suivante $s = \sum_{i=3}^{n} i$

Solution:

```
>> n = input('donner la valeur de n = ');
>> s=0;
>> for   i=3: n
      s=s+i;
end
>> disp( 'la somme s est: ' ) , s  % disp(['la somme s est: ',num2str(s)])
```

Exemple 2 : faire un programme Matlab qui calcule la somme suivante $s = \sum_{k=1}^{b} k^a$

Exemple 3 : avec la boucle **for** donner les valeurs de la variable **r** dans [1.1 : -0.1 : 0.75]
Exemple 4 : avec la boucle **for** calculer la factorielle de la variable **n** (**n!**) pour n = 5
Exemple 5 : calculer v=amp*sin(wt) avec amp=i*1.2 et wt=j*0.05 et i=1 :5 j=1 :20

2.1.2 La Boucle WHILE : (tant que . . . faire)

Une seconde possibilité pour exécuter une séquence d'instructions de manière répétée consiste à effectuer une boucle **while** jusqu'à ce qu'une condition soit vérifiée.

Syntaxe :

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

Où

- L'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 1 : exécuter le script suivant

```
>> x = 1 ;
>> while  x<=50
          x=x+1
          s = x^2 ;
end
```

Exemple 2 : calculer **n!** avec une boucle while

Solution:

```
>> n = 4;
>> k = 1;  nfac = 1;
>> while   k <= n
         nfac = nfac*k;
         k = k+1;
end
>> nfac
nfac = 24
```

Exemple 3 : faire un programme sous matlab qui calcule la somme suivante:
S=1+2/2! +3/3!+…
on arrête le calcul quand S>2.5

Solution:

```
>> clear all
>> s=1;
>> i=1;  f=1;
>> while  s<=2.5
          i=i+1
          f=f*i;
          s=s+i/f
end
```

Exemple 4 :

Faire un programme sous matlab qui calcule les deux sommes suivantes:

S1 = 1+1/2 +1/3 + 1/4 + ……. + 1/n .

S2 = 1 – 1/2 + 1/3 – 1/4 + …. + (ou -) 1/n .

2.2. Les instructions de condition :

On a parfois besoin d'exécuter une séquence d'instructions seulement dans le cas où une condition donnée est vérifiée au préalable. Différentes formes d'instructions de condition existent sous matlab.

2.2.1 L'instruction if : (si)

L'instruction de condition la plus simple a la forme suivante :

1er cas :
Sa syntaxe est:

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

Où

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

Exemple:
Faire un programme sous matlab qui résout le problème suivant:
1) y = x si x<0
2) y = x^2 si x>0
3) y = 10 si x=0

Solution

```
>> clear all
>> x= input( 'donner la valeur de x ' );
>> if  x<0
   y=x;
>> end
>> if  x>0
    y=x^2;
>> end
>> if  x==0
    y=10;
>> end
>> disp([ ' la valeur de y est: ' , num2str(y)])
```

2eme cas : if else (si sinon)
Il existe une séquence conditionnée sous forme d'alternatives.
Sa syntaxe est :

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

Où

- séquence d'instructions 1 est la séquence d'instructions à exécuter dans le cas où l'expression logique est vraie et séquence d'instructions 2 est la séquence d'instructions à exécuter dans le cas où expression logique est faux.

Exemple 1:

```
>> clear all
>>x=input('donner la valeur de x ');
>> if  x<0
    y = x;
>> else
       y = x^2;
>> end
>> disp([ 'la valeur de y est: ', num2str(y) ])
```

Exemple 2 :

Faire un programme sous matlab qui résout le problème suivant:

Y= x si x<0
Y= 10 si x=0
Y= sqrt(x) si 0<x<20
Y= x^2 si x =20
Y= x^3 si x>20

Solution

```
>> clear all
>>x=input('donner la valeur de x ');
>> if  x<0
   y=x;
>> elseif  x==0
       y=10;
>> elseif  x==20
       y=x^2;
>> elseif   x>0 & x<20
       y=sqrt(x);
>> else
       y=x^3;
>> end
>> disp([ 'la valeur de y est: ', num2str(y) ])
```

2.2.1 L'instruction switch

Une alternative à l'utilisation d'une séquence d'instructions conditionnées pour effectuer un choix en cascade existe. Il s'agit de l'instruction **switch**.

Sa syntaxe est:

```
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
```

Où
– **var** est une variable numérique ou une variable chaîne de caractères.
– **cst_1, . . ., cst_N**, sont des constantes numérique ou des constantes chaîne de caractères.
– séquence d'instructions **N** est la séquence d'instructions à exécuter si **var** == **cst-N**.
– Si l'instruction à exécuter est la même pour un ensemble de cas alors la syntaxe est :
case {**cst-1**, **cst-2**,…}

Exemples 1:

```
jj= input('donner le jour 1 :7 ');
switch  jj
case  1
      disp('samedi')
case  2
      disp('dimanche')
case  3
      disp('lundi')
case  4
      disp('mardi')
case  5
      disp('mercredi')
case  6
      disp('jeudi')
case  7
      disp('vendredi')
end
```

Exemples 2:

```
mm= input('donner le mois: ');
switch  mm
case  'Ja'
      disp('Janvier')
case  'F'
      disp('Février')
case  'M'
      disp('Mars')
case  'Av'
      disp('Avril')
otherwise
      disp('autre')
end
```