
TP n° 3 - Structures conditionnelles

Tous les sujets et les corrigés sont disponibles aux adresses suivantes :

<http://www.lif.univ-mrs.fr/~vpoupet/enseignement/matlab09.php>

<http://www.lif.univ-mrs.fr/~pvanier/?q=cours>

Exercice 1.

Taxacées

Nous avons vu comment les scripts permettaient de demander à Matlab d'exécuter une séquence d'instructions, et comment une boucle `for` permettait de demander qu'une même instruction soit répétée un certain nombre de fois (en changeant un paramètre si l'on veut).

Nous allons maintenant voir comment on peut demander à un script de changer son comportement en fonction de la situation, à l'aide du mot-clé `if`.

Définissez une fonction `test(x)` en recopiant le code suivant dans un script nommé `test.m` :

```
function reponse = test(x)
if x == 42
    reponse = 'C''est vrai !' ;
elseif x == 41
    reponse = 'Presque...' ;
else
    reponse = 'C''est faux !' ;
end
```

1. Le double symbole égal et la double apostrophe dans le script sont tous les deux importants... À votre avis à quoi servent-ils ?

Réponse : Le double symbole égal sert à tester l'égalité (et non à l'affectation, contrairement au symbole égal), la double apostrophe sert à afficher une apostrophe à l'intérieur d'une chaîne de caractères.

2. Que renvoie la fonction si on l'appelle avec l'argument 12 ? Et avec 42 ? Et avec 41 ? Avez-vous compris la signification des mots-clés `if`, `elseif` et `else` ?

Réponse : Les mots clés `if`, `elseif` et `else` servent à faire des disjonctions de cas en fonction des résultats du test en argument du `if`. Le code juste après le `if` est exécuté uniquement si la condition est vraie, le code après le `elseif` (qui est facultatif et que l'on peut répéter autant de fois que l'on veut) est exécuté uniquement si la condition après celui-ci est vraie et que les précédentes ne l'étaient pas. Le code après le `else` n'est exécuté que si aucune des conditions le précédent n'a été satisfaite.

3. Écrivez une fonction `pair(x)` qui prend en argument un entier `x` et qui renvoie `vrai` si l'entier est pair et `faux` sinon.

Indication : Utilisez la fonction `mod(a, b)` qui renvoie la valeur de `a` modulo `b` pour obtenir la parité de `x`.

Réponse :

```
function r=pair(x)
if mod(x,2)==0
    r='vrai';
else
    r='faux';
end
```

On peut également effectuer des tests un peu plus complexes :

- $a \sim b$ signifie que a est différent de b ;
- $a < b$ (resp. $a \leq b$) signifie que a est inférieur (resp. inférieur ou égal) à b (ne fonctionne que si les éléments sont comparables...);
- $(a == b) \ \&\& \ (a == c)$ signifie que les deux conditions $(a == b)$ et $(a == c)$ sont vraies à la fois;
- $(a == b) \ || \ (a == c)$ signifie que l'une des deux conditions est vraie (pas nécessairement les deux).


4. Écrivez une fonction `binome(a, b, c)` qui renvoie les racines du polynôme $aX^2 + bX + c$.

Indications :

- Suivez l'algorithme habituel. Calculez $\Delta = b^2 - 4ac$ puis en fonction du signe de δ renvoyez les racines réelles ou complexes;
- Vous pouvez utiliser la variable `i` en *Matlab* pour désigner la racine carrée de -1 , et la fonction `sqrt(x)` pour obtenir la racine carrée d'un nombre positif;
- Pour renvoyer plusieurs valeurs, mettez les dans un tableau et renvoyez le tableau.

Réponse :

```
function r=binome(a,b,c)
delta=b*b-4*a*c;
if delta >= 0
    r=[(-b+sqrt(delta))/(2*a), (-b-sqrt(delta))/(2*a)];
else
    r=[(-b+i*sqrt(-delta))/(2*a), (-b-i*sqrt(-delta))/(2*a)];
end
```

 **Bonus.** Quel rapport avec le titre de l'exercice ?

Exercice 2.

Tant qu'il y aura des sommes

Nous avons déjà vu comment utiliser une boucle `for` pour répéter plusieurs instructions. Cependant il fallait connaître à l'avance le nombre de fois que l'on voulait répéter la boucle.

Le mot-clé `while` sert à exécuter une suite d'instructions *jusqu'à* ce qu'une condition soit vérifiée.

1. Exécutez le script suivant :

```
x = 1;
while x < 1000
    x
    x = 2*x;
end
```

Que fait-il ? Avez-vous compris la signification du mot-clé `while` ?

Réponse : Ce programme calcule la plus petite puissance de deux supérieure à mille.

2. Écrivez une fonction `cube(n)` qui renvoie le plus grand cube inférieur ou égal à n .

Réponse :

```
function r=cube(n)
i=0;
while i^3<=n
    i=i+1;
end
r=(i-1)^3;
```

3. Écrivez une fonction `somme_carres(n)` qui renvoie le plus grand entier k vérifiant

$$\sum_{i=1}^k i^2 \leq n$$

Réponse :

```
function r=somme_carres(n)
i=0;
S=0;
while S<=n
    i=i+1;
    S=S+i^2;
end
r=i-1;
```

Exercice 3.

« - chotomie ! »

Le jeu *dichotomie* est un jeu qui se joue à deux joueurs (et qui n'est pas très amusant en fait).

Le premier joueur choisit un nombre entre 1 et 100, et le second joueur essaie de deviner ce nombre. À chaque essai du second joueur, le premier indique si le nombre est plus grand ou plus petit que le nombre à deviner (ou égal, et dans ce cas le jeu est terminé).

1. Écrivez un script qui permet de jouer à *dichotomie* contre l'ordinateur (le script choisit un nombre, et le joueur essaie de le deviner).

Indications :

- Utilisez la fonction `randi(n)` qui renvoie un nombre entier tiré aléatoirement entre 1 et n pour choisir le nombre ;
- Utilisez l'instruction `a = input('Entrez un nombre :')` qui demande à l'utilisateur d'entrer un nombre et enregistre la valeur dans la variable `a`.

Réponse :

```
k=randi(100);
a=k-1;
while a~=k
    a=input('Entrez un nombre :');
    if a>k
        disp('Le nombre est plus grand');
    elseif a<k
        disp('Le nombre est plus petit');
    end
end
disp('Vous avez gagné');
```