

Faculté des sciences

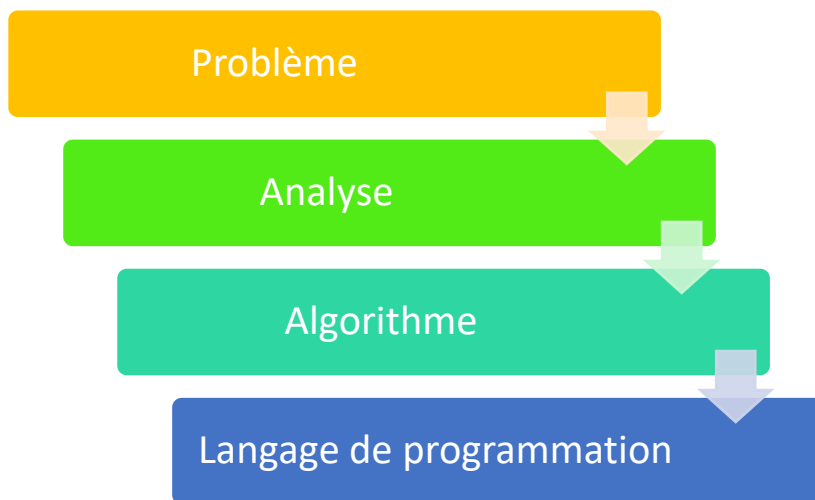
Département de mathématiques

Master 1: MF/MSS

Module : Apprentissage logiciel/LCS

6. Programmation sous MATLAB

Avant de commencer, il faut faire attention à l'étape d'**Analyse** car c'est cette étape qui fait la différence entre les programmeurs.



6.1. Les structures conditionnelles

6.1.1. L'instruction « if »

L'instruction **if** évalue une expression logique et exécute un groupe d'instruction lorsque l'expression logique est vraie.

Les instructions facultatives **elseif** et **else** permettent d'imbriquer des boucles conditionnelles supplémentaires. L'instruction **end**, clôture la structure conditionnelle **if**.

if condition 1

Instruction (si la condition 1 est vérifiée)

elseif condition 2

Instruction (si la condition 2 est vérifiée)

else

Instruction (si les conditions ne sont pas vérifiées)

end

- **Opérateurs de comparaison et opérateurs logiques**

- Les 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 :

`&` : et (`x&y`)

`|` : ou (`x|y`)

`~` : non (`~x`)

Application 1 : Ecrire un script qui permet de comparer deux nombres.

```
comparaison.m
1 % input
2 x=input('saisir le premier nombre:');
3 y=input('saisir le deuxième nombre:');
4
5 % les conditions
6 if x > y
7     disp([num2str(x), 'est supérieur à' num2str(y)]);
8 elseif x < y
9     disp([num2str(x), 'est inférieur à', num2str(y)]);
10 elseif
11     disp([num2str(x), 'égal à', num2str(y)]);
12 endif
13
```

Application 2 : Ecrire la fonction `resoleqt2degre.m` qui calcule les racines de l'équation $ax^2 + bx + c = 0$ (a, b et c réels)

```
comparaison.m  * resoleqt2degre.m
1 function [r] = resoleqt2degre(a, b, c)
2     delta = b*b-4*a*c;
3     if delta>0
4         r=[(-b+sqrt(delta))/(2*a), (-b-sqrt(delta))/(2*a)];
5     elseif delta<0
6         r=[(-b+i*sqrt(-delta))/(2*a), (-b-i*sqrt(-delta))/(2*a)];
7     else
8         r=-b/(2*a);
9     endif
10 endfunction
```

6.1.2. L'instruction « switch » - « case »

L'instruction **switch** exécute un groupe d'instructions relatives à la valeur prise par une variable.

Les instructions **case** et **otherwise** délimitent les groupes.

Le **end** est obligatoire à la fin de la structure.

```
Switch variable_de_choix
case constante 1
    séquence d'instruction 1
case constante 2
    séquence d'instruction 2
...
case constante N
    séquence d'instruction N
otherwise
    instructions par défaut
end
```

Remarque : on utilise switch case au lieu de refaire if else plusieurs fois à condition de l'utiliser dans le cas de valeur exact, on ne peut pas l'utiliser pour une comparaison ou une condition.

Application : écrire un script pour saisir le numéro du jour dans une semaine (1-7) et affichez le nom du jour à l'aide de switch case.

```
comparaison.m x resoleqt2degre.m x semaine.m x
1 nbr = input('donner un nombre entre 1 et 7:');
2 switch nbr
3     case 1
4         disp('dimanche');
5     case 2
6         disp('lundi');
7     case 3
8         disp('mardi');
9     case 4
10        disp('mercredi');
11     case 5
12        disp('jeudi');
13     case 6
14        disp('vendredi');
15     case 7
16        disp('samedi');
17 otherwise
18     disp('erreur, saisir un nombre entre 1 et 7');
19 endswitch
```

6.2. Les structures répétitives

6.2.1. L'instruction « for »

La boucle **for** permet de répéter un nombre de fois prédéterminé une séquence d'instruction.

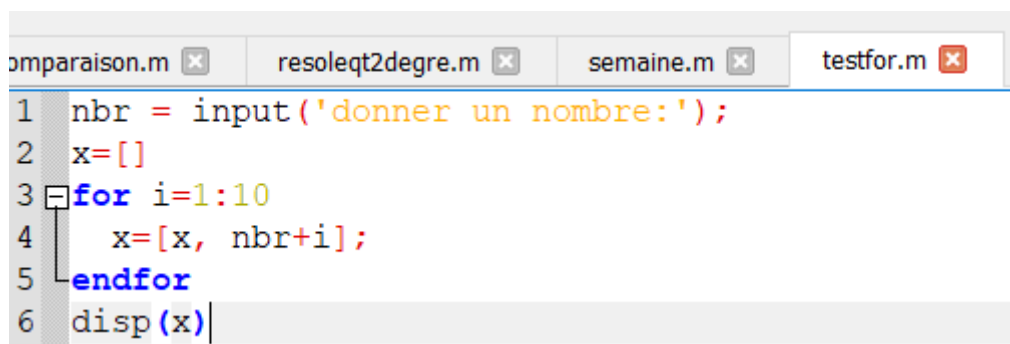
Le **end** est obligatoire à la fin de la structure.

for **compteur** = **valeur_Debut** : **pas** : **valeur_fin**

Instructions

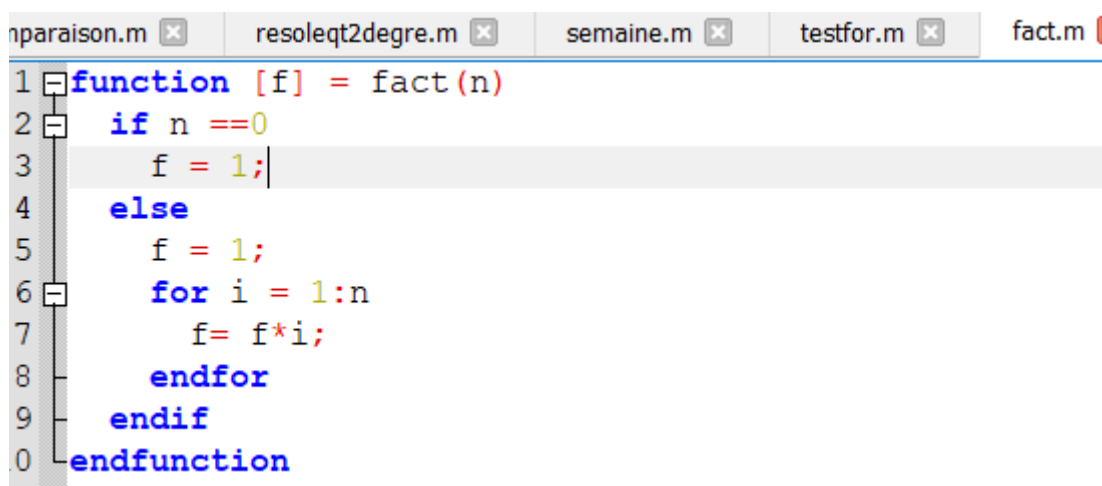
end

Application 1 : écrire un script qui demande un nombre de départ, et qui ensuite affiche les dix nombres suivants sous forme d'un vecteur ligne.



```
comparaison.m x resoleqt2degre.m x semaine.m x testfor.m x
1 nbr = input('donner un nombre:');
2 x=[]
3 for i=1:10
4     x=[x, nbr+i];
5 endfor
6 disp(x)
```

Application 2 : écrire une fonction fact qui demande un nombre de départ, et qui calcule son factorielle.



```
comparaison.m x resoleqt2degre.m x semaine.m x testfor.m x fact.m
1 function [f] = fact(n)
2     if n ==0
3         f = 1;
4     else
5         f = 1;
6         for i = 1:n
7             f= f*i;
8         endfor
9     endif
10 endfunction
```

6.2.2. L'instruction « while »

La boucle **while** permet de répéter une séquence d'instructions tant qu'une condition est vérifiée

Le **end** est obligatoire à la fin de la structure.

While conditions

Instructions

end

Application 1 : Ecrire un script qui demande à l'utilisateur un nombre compris entre 1 et 10 jusqu'à ce que la réponse convienne.

```
1 nbr = input('donner un nombre:');  
2 while (nbr < 1 | nbr > 10)  
3     nbr = input('réponse fausse, entrer un autre nombre:');  
4 endwhile  
5 disp ('réponse vraie ');
```

Application 2 : Ecrire une fonction somme_carres(n) qui renvoie le plus grand entier k vérifiant

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

```
1 function [r] = somme_carres(n)  
2     i=0;  
3     somme = 0;  
4     while somme < n  
5         i=i+1;  
6         somme = somme + i^2;  
7     endwhile  
8     r = i-1;  
9 endfunction
```

6.3. « break », « continue », « return »

6.3.1. break

L'instruction **break** permet de sortir d'une boucle **for** ou d'une boucle **while**. L'exécution se poursuit alors séquentiellement à partir de l'instruction suivant le mot clé **end** fermant la boucle. En cas de boucles imbriquées, on interrompt seulement l'exécution de la boucle intérieure contenant l'instruction break.

Exemple :

```
1 function x = ncarre_br(n)
2     x = [];
3     for i = 1:n
4         if i==5
5             break;
6         endif
7         x = [x, i^2];
8     endfor
9 endfunction
```

```
>> x = ncarre_br(10)
```

```
X=
```

```
1 4 9 16
```

6.3.2. « continue »

L'instruction **continue** permet de sauter à l'itération suivante dans une boucle **for** ou **while**.

Exemple :

```
1 function x = ncarre_con(n)
2     x=[];
3     for i = 1:n
4         if i==5
5             continue;
6         endif
7         x = [x, i^2];
8     endfor
9 endfunction
```

```
>> x = ncarre_con(10)
```

```
X=
```

```
1 4 9 16 36 49 64 81 100
```

6.3.3. « return »

L'instruction **return** provoque un retour au programme appelant (ou au clavier). Les instructions suivant le return ne sont donc pas exécutées. L'instruction return est souvent utilisée conjointement avec une instruction conditionnée par exemple pour tester dans le corps d'une fonction si les paramètres d'entrée ont les valeurs attendues.

Exemple :

```
1 function x = ncarre_ret(n)
2     x = [];
3     for i = 1:n
4         if i==5
5             return;
6         endif
7         x = [x, i^2];
8     endfor
9 endfunction
```

```
>> x = ncarre_ret(10)
```

```
X=
```

```
1  4  9 16
```