1. Introduction

Les programmes informatiques sont constitués d'instructions exécutées séquentiellement, du début à la fin.

La plupart des problèmes à résoudre par programmation requièrent l'éventualité d'effectuer un choix dans le programme. Une *structure conditionnelle* est une instruction permettant de spécifier des séquences d'instructions alternatives dans un programme.

Scilab offre quatre structures conditionnelles :

- La *structure if (SI)*
- La *structure if-else (SI-SINON)*
- La structure if-elseif (*SI-SINON-SI)*
- La structure select (*SÉLECTION)*

2. La *structure if (SI)*

Dans sa forme la plus simple (la structure SI), une structure conditionnelle est composée des mots réservés **SI**, **ALORS** et **FINSI**, d'une *condition* et d'une *séquence d'instructions* à exécuter lorsque la condition est vraie.

Algorithme	Script Scilab
SI *condition* **ALORS** *Séquence d'instructions* **FINSI** suite du programme	if *condition* then instructions_if end suite du programme

Les structures conditionnelles sont basées sur l'évaluation d'une condition, dont le résultat est *vrai* ou *faux*. C'est sur la base de cette condition que le flux d'exécution est déterminé.

Les conditions

Une condition est une comparaison. Cet énoncé décrit l'essentiel de ce qu'est une condition. En pratique, une *condition simple* est composée d'au moins trois éléments :

1. une première valeur,
2. un opérateur de comparaison, et
3. une seconde valeur.

Les valeurs peuvent être a priori de n'importe quel type (numériques, chaînes de caractères ou booléens) et peuvent être spécifiées explicitement sous forme de constantes ou implicitement sous forme d'expressions à évaluer. Si l'on veut que la comparaison ait un sens, il faut cependant que les deux valeurs comparées soient du même type ou de types comparables.

L'opérateur de comparaison dans une condition simple est appelé un opérateur relationnel. Ces opérateurs permettent de comparer l'envergure de deux valeurs.

Enfin, des conditions simples peuvent être regroupées en une *condition composée* à l'aide des opérateurs logiques.

Opérateurs relationnels et Opérateurslogiques

Les opérateurs relationnels permettent de comparer deux valeurs

Opérateurs relationnels	
Relations	Opérateurs Scilab
plus petit que	<
plus petit ou égal	<=
plus grand	>
plus grand ou égal	>=
égal	==
différent	<> ou ~=

Opérateurs logiques	
Opérateurs logiques	Opérateurs Scilab
Et	&
Ou	|
Non (négation)	~

Les *opérateurs logiques* permettent de relier des conditions simples en une seule « condition conposée ». Le regroupement de conditions est parfois requis pour spécifier qu'un ensemble de conditions doit être satisfait pour procéder à l'exécution d'une séquence d'instructions. Par exemple, une *condition composée* est requise pour exprimer la condition « *la valeur doit être supérieure à zéro et inférieure à 100* » ou « *la couleur doit être rouge ou verte* ». Les opérateurs logiques permettent de tels regroupements de conditions simples.

Les opérateurs relationnels permettent de comparer deux valeurs de types comparables. Des valeurs sont de types comparables lorsqu'elles peuvent être logiquement comparées.

Exemples

```
  // Lire deux valeurs
a=input("Entrez une valeur : ")
b=input("Entrez une valeur : ")

 // Identifier la plus petite de deux valeurs lues
 if a < b then
   disp (a, "Minimum = ")
end
if a >= b then
   disp (b, "Minimum = ")
end

 // Déterminer si une des deux valeurs est égale à 0
 if a*b == 0 then
      disp ( "Au moins une valeur est 0" )
end
```

Attention : les opérateurs relationnels ne peuvent pas être enchaînés. Par exemple, la condition **5<a<10** est invalide. Il faut exploiter les *opérateurs logiques* pour exprimer de telles conditions.

```
// Lire et valider une couleur
couleur = input( "Entrez une couleur: " ,'s')
if couleur == "bleu" | couleur == "blanc" | couleur == "rouge"  then
   disp( "Couleur invalide")
end
```

---

3. Structures if et if-else (SI et SI-SINON)

Algorithme	Script Scilab

SI *condition* **ALORS**
Séquence d'instructions #1
SINON
Séquence d'instructions #2
FINSI
suite du programme
	if *condition* **then**
	Séquence d'instructions #1
	else
	Séquence d'instructions #2
	end
	suite du programme

Cette structure conditionnelles est relativement claire. Lorsque le flux d'exécution atteint la structure (i.e. la ligne **if** *condition* **then** du script), **Scilab** examine la valeur de la *condition*. Si cette condition est vraie, la *Séquence d'instructions #1* est exécutée (Cette séquence d'instructions peut être très brève comme très longue, cela n'a aucune importance).
À la fin de la *Séquence d'instructions #1, Scilab saute à la fin de la structure conditionnelle. De même, si la condition est fausse, le flux d'exécution ignore la Séquence d'instructions #1 et passe directement à la première ligne située après le else afin d'exécuter la Séquence d'instructions #2*

Exemple

```
//  Structure if -else (SI-SINON)

a=input("Entrez une valeur : ")
b=input("Entrez une valeur : ")
if  a < b then

      disp(a, "Minimum = ")

 else

     disp (b, "Minimum = ")

end
```

4. Structures if-else (SI-SINON)

Il y a des situations où deux alternatives ne suffisent pas. Par exemple, un programme devant donner l'état de l'eau selon sa température doit choisir entre trois réponses possibles (solide, liquide ou gazeuse).

Une première solution serait la suivante :

```
Temp= input("Température de l'eau ? ")    // Lecture de la température de l'eau
if  Temp  <=  0  then                     // Est-ce de la glace ?
   disp( "C'est de la glace" )
end
if Temp > 0 & Temp < 100 then             // Est-ce du liquide ?
    disp( "C'est du liquide" )
end
if  Temp >= 100 then                      // Est-ce de la vapeur ?
    disp ("C'est de la vapeur" )
end
```

Notez que ce script est assez laborieux. Les conditions se ressemblent plus ou moins, et surtout on oblige le flux d'exécution à examiner trois conditions successives alors que toutes portent sur un même thème, soit la température (la valeur de la variable **Temp**).
Il est cependant plus rationnel d'*imbriquer* les structures :

5. Structures if-elseif (SI-SINON-SI) imbriquées

Une deuxième solution à l'exemple précédent :

```
Temp = input("Température de l''eau ?")     // Lecture de la température de l'eau
if  Temp <= 0 then                          // Est-ce de la glace ?
       disp( "C''est de la glace" )
elseif   Temp > 0  &  Temp < 100 then       // Est-ce du liquide ?
       disp( "C''est du liquide" )
   else                                     // Temp >= 100 , C'est de la vapeur, c'est sûr !
      disp ("C''est de la vapeur" )
end
```

Nous avons fait des économies en termes d'instruction : au lieu de devoir taper trois conditions, dont une composée, nous n'avons plus que deux conditions simples.

Mais aussi, et surtout, nous avons fait des économies au niveau du temps d'exécution du script. En effet, si la température est inférieure à zéro, celui-ci écrit dorénavant « C'est de la glace » et le flux d'exécution passe directement après le dernier **end**, sans examiner d'autres possibilités (qui sont forcément fausses).

Cette deuxième version n'est donc pas seulement plus simple à écrire et plus lisible, elle est également plus performante à l'exécution.

Les structures conditionnelles imbriquées sont donc un outil indispensable à la simplification et à l'optimisation des scripts.

L'emploi de structures conditionnelles imbriquées engendre une économie en temps d'exécution puisque le flux d'exécution quitte la structure dès qu'une condition est satisfaite et la séquence d'instructions correspondante est exécutée.

L'avantage des structures conditionnelles imbriquées est par contre amoindri par la complexité du script lorsque plusieurs conditions sont impliquées.

Voici l'exemple précédent reformulé à l'aide d'une structure conditionnelle **if elseif**

```
Temp = input("Température de l"eau? ")
if Temp <= 0 then
      disp("C'est gelé")
elseif  Temp <= 12 then
     disp ("C"est froid" )
        elseif   Temp <= 25 then
             disp ("C"est confortable")
          elseif Temp <= 75 then
            disp ("C'est chaud")
         elseif Temp <= 100 then
                disp ("C"est très chaud")
           else
              disp ("C"est brûlant")
end
```

Dans une structure **if elseif**, la dernière séquence d'instructions

(**else** *Séquence d'instructions #n+1*) est optionnelle.

7. Structure de sélection : select

Un script peut parfois contenir une série de décisions dans laquelle une variable ou une expression est testée séparément pour chacune des valeurs qu'elle peut prendre, et où différentes séquences d'instructions sont exécutées conséquemment.
Scilab offre la structure de sélection, **select,** pour implémenter de telles prises de décisions.

La structure de sélection permet de formuler une structure **if... elseif** tout en rendant la logique du script beaucoup plus lisible.

Voici la syntaxe généralisée de la structure **select** :

select *expression*
  case *constante #1* **then** *Séquence d'instructions #1*
  case *constante #2* **then** *Séquence d'instructions #2*
  case *constante #3* **then** *Séquence d'instructions #3*
  ...
  case *constante #n* **then** *Séquence d'instructions #n*
else
  Séquence d'instructions #n+1
end

Les caractéristiques de cette structure sont :
- Chaque ***Séquence d'instructions*** peut contenir une ou plusieurs commandes ***Scilab***.

- La section **else** et sa séquence d'instructions sont optionnelles.
- Lors de l'exécution d'une structure de sélection, la valeur de l'expression est comparée successivement aux constantes, en commençant par la première (**constante #1**).
- Lorsqu'une constante correspondant à la valeur de l'expression est trouvée, la Séquence d'instructions correspondante est exécutée puis le flux d'exécution quitte la structure de sélection.
- Si aucune constante correspondant à la valeur de l'expression n'est trouvée et que la structure dispose d'une section **else**, la **Séquence d'instructions** correspondante (**Séquence d'instructions #n+1**) est exécutée.

Exemple

```
// Simulation des opérations arithmétiques

Valeur1 =  input('Donner la valeur 1 : ')
Valeur2 = input('Donner la valeur 2 : ')
Opérateur =input('Donner l''opérateur : ','s')
select Opérateur
    case  '+' then  Résultat = Valeur1+Valeur2
    case  '-' then  Résultat = Valeur1-Valeur2
    case  '*'   then  Résultat = Valeur1*Valeur2
    case 'x'   then  Résultat = Valeur1*Valeur2
    case  '/'  then  Résultat = Valeur1/Valeur2
else

    disp( "Opérateur erroné ")

end
 disp(Résultat,"Le résultat de l''opération est :  ")
```

Notons que :

Dans l'exemple ci-dessus, la valeur de la variable **Opérateur** est comparée à chaque constante spécifiée.

Si une constante testée correspond à la valeur de **Opérateur**, la séquence d'instructions associée à cette constante est exécutée.

L'expression testée doit être comparée à des **constantes de type chaîne de caractère** (ex: '+')

Si la valeur de la variable **Opérateur** ne correspond pas à une **constante**, elle est comparée à la **constante suivante**, et ainsi de suite.

La section **else** permet de prendre les mesures nécessaires lorsque la variable **Opérateur** ne correspond à aucune des constantes énumérées.