Chapitre 4

Instructions de choix

1. Instruction conditionnelle

Rôle : permet d'exécuter une séquence d'instructions plutôt qu'une autre en fonctions de données venant de l'utilisateur, ou de résultats calculés faits dans l'algorithme.

Syntaxe :

```
Si <condition> alors
    <séquence d'instructions>
finsi
```

<condition> est évaluée. Sa valeur peut être

- vrai ➔ la séquence est exécutée
- faux ➔ l'instruction est ignorée et on passe à l'instruction qui suit le fin si

Forme complète : instruction alternative :

```
Si <condition> alors
    <séquence d'instructions 1 >
sinon
    <séquence d'instructions 2 >
finsi
```

<condition> est évaluée. Sa valeur peut être

- vrai ➔ la séquence 1 est exécutée
- faux ➔ la séquence 2 est exécutée

Remarque : il faut que la condition soit **évaluable**, c'est à dire que les variables qui la composent éventuellement aient une valeur. Ci-dessous, une situation d'erreur

```
Début
  Si x > 5 alors
    …
  finsi
Fin
```

```
Algorithme maxdeux-1
/* demande deux nombres à l'utilisateur, calcule et affiche le plus grand des
deux. */
/* Les déclarations */
Variables
/* identificateur       : type              rôle */
   nb1, nb2             : entier            /* les deux nombres */
   maxi                 : entier            /* maximum des 2 nombres */

/* Les instructions */
Début
1 écrire('donner deux entiers ')
2 lire(nb1, nb2)
3 Si nb1 > nb2 alors
4    maxi ← nb1
  sinon
5    maxi ← nb2
6 finsi
  écrire('Le plus grand des deux
       nombres ', nb1, ' et ', nb2, 'est :
       ', maxi)
Fin
```

	nb1	nb2	maxi	Ecran
2	10	15		
3 : faux				
5			15	
6				
7 :				Le plus grand des deux nombres 10 et 15 est 15s

```
Algorithme maxdeux-2
/* demande deux nombre à l'utilisateur, calcule et affiche le plus grand des
deux. */
/* Les déclarations */
Variables
/* identificateur       : type              rôle */
   nb1, nb2             : entier            /* les deux nombres */
   maxi                 : entier            /* maximum des 2 nombres */

/* Les instructions */
Début
1 écrire('donner deux entiers ')
2 lire(nb1, nb2)
3 maxi ← nb1
4 Si nb2 > maxi alors
5    maxi ← nb2
6 finsi
7 écrire('Le plus grand des deux
       nombres ', nb1, ' et ', nb2,
       'est : ', maxi
Fin
```

	nb1	nb2	maxi	Ecran
2	10	15		
3			10	
4 : vrai				
5			15	
7 :				Le plus grand des deux nombres 10 et 15 est 15s

2. Instruction à choix multiple

Exemple : algorithme qui détermine le nombre de jours du mois entré par l'utilisateur.

```
Algorithme NombreDeJours

/* Les déclarations */
Variables
/* identificateur       : type            rôle */
   mois                 : entier          /* n° du mois dans l'année */
   an                   : entier          /* millésime de l'année */
   bis                  : booléen         /* indicateur d'année bissextile */
   nbJ                  : entier          /* nombre de jours par mois */

/* Les instructions */
Début
   lire(mois, an)
   bis ← (an mod 4 = 0 et an mod 100 <> 0) ou (an mod 400 = 0)
   Si mois ∈ {1, 3, 5, 7, 8, 10, 12} alors                        (*)
     nbJ ← 31
   sinon
     Si mois ∈ {4, 6, 9, 11} alors                                (*)
       nbJ ← 30
     sinon
       Si mois = 2 alors
         Si bis alors
           nbJ ← 29
         sinon
           nbJ ← 28
         finsi
       sinon
         nbJ ← 0
       finsi
     finsi
   finsi
   Si nbJ = 0 alors
     écrire('le nombre saisi pour le mois doit être compris entre 1 et 12')
   sinon
     écrire(nbJ, 'jours')
   finsi
Fin
```

(*) s'écrit encore :

```
mois = 1 ou mois = 3 ou mois = 5 ou mois = 7 ou mois = 8 ou mois = 10 ou mois = 12
```

On peut écrire un algorithme équivalent à la partie grisée avec l'instruction suivante :

Syntaxe :

selon <expression entière ou caractère> dans
<liste de valeurs n° 1 > : < séquence n°1 >
[<liste de valeurs n° 2 > : < séquence n°2 >
…
<liste de valeurs n° n > : < séquence n°n >
[autrement : <séquence autre>]]
finselon

la clause autrement est facultative

```
selon mois dans
  1, 3, 5, 7, 8, 10, 12   : nbJ ← 31
  4, 6, 9, 11             : nbJ ← 30
  2                       : Si bis alors
                              nbJ ← 29
                            sinon
                              nbJ ← 28
                            finsi
  autrement               : nbJ ← 0
finselon
```

Mécanisme :

- Evaluation de l'expression (le sélecteur), le plus souvent c'est une variable
- Recherche de cette valeur dans les listes, en parcourant celles-ci dans l'ordre où elles sont écrites
- Dès que la valeur est trouvée dans une liste
  - la séquence correspondante est exécutée
  - l'algorithme passe à l'instruction qui suit le fin selon
- Si la valeur du sélecteur n'est trouvée dans aucune liste, c'est la séquence qui correspond à *autrement* qui est exécutée

Remarques :

- les listes doivent être disjointes
- le sélecteur doit être d'un type énumérable, c'est à dire **entier** ou **caractère** (mais pas chaîne).

3. Instruction si … sinonSi … finsi

Dans l'exemple précédent, les différents cas possibles sont relatifs à un test d'égalité d'une variable par rapport à une valeur, ou à un test d'appartenance à un ensemble de valeurs. Examinons le cas d'un tarif dégressif suivant la quantité de produit commandée.

quantité	prix au kg
entre 1 et 5 kg (exclus)	100
entre 5 et 10 kg (exclus)	90
entre 10 et 30 kg (exclus)	80
entre 30 et 60 kg (exclus)	70
à partie de 60 kg	55

L'algorithme suivant répond à la question.

```
Algorithme Tarifs

/* Les déclarations */
Constantes
/* identificateur       = valeur          rôle */
   BORNE_1              = 5
   BORNE_2              = 10
   BORNE_3              = 30
   BORNE_4              = 60
   PRIX_1               = 100
   PRIX_2               = 90
   PRIX_3               = 80
   PRIX_4               = 70
   PRIX_5               = 55

Variables
/* identificateur       : type            rôle */
   quantite             : réel            /* quantité de produit en kg */
   prix                 : réel            /* prix au kg */

/* Les instructions */
Début
   lire(quantite)
   Si quantite > BORNE_4 alors
     prix ← PRIX_5
   sinon
     Si quantite > BORNE_3 alors
       prix ← PRIX_4
     sinon
       Si quantite > BORNE_2 alors
         prix ← PRIX_3
       sinon
         Si quantite > BORNE_1 alors
           prix ← PRIX_2
         sinon
           prix ← PRIX_1
         finsi
       finsi
     finsi
   finsi
   écrire(prix)
Fin
```

Il est visible que la mise en forme de ce type d'algorithme, où il y a beaucoup de si imbriqués et où les cas s'excluent mutuellement, peut être assez fastidieuse. On peut lui préférer la structure conditionnelle suivante :

```
/* Les instructions */
Début
   lire(quantite)
   Si quantite > BORNE_4 alors
     prix ← PRIX_5
   sinonSi quantite > BORNE_3 alors
     prix ← PRIX_4
   sinonSi quantite > BORNE_2 alors
     prix ← PRIX_3
   sinonSi quantite > BORNE_1 alors
     prix ← PRIX_2
   sinon
     prix ← PRIX_1
   finsi
   écrire(prix)
Fin
```

Syntaxe :

```
si <expression booléenne1> alors
    < séquence n°1 >
sinonSi <expression booléenne2> alors
    < séquence n°2 >
[sinonSi <expression booléenne3> alors
    < séquence n°3 >
 …
 sinonSi <expression booléenneN> alors
    < séquence n°N >]
 sinon
    < séquence n°N+1 >
finsi
```

Mécanisme :

- A la première expression booléenne vraie, la séquence correspondante est exécutée. et l'algorithme passe à l'instruction qui suit le fin selon
- Si aucune expression booléenne n'est vraie, la séquence correspondant à l'instruction sinon est exécutée.