Chapitre 3
Les conditionnelles

1. Présentation

Jusqu'à présent, les algorithmes présentés sont complètement linéaires, c'est-à-dire que les instructions sont toutes exécutées dans l'ordre. Les structures de contrôle permettent, suivant les cas, de ne pas exécuter les mêmes instructions. Pour reprendre l'exemple de l'algorithme de calcul du prix TTC, il serait possible de demander à l'utilisateur si le produit a le droit à un taux de TVA réduit et, seulement si la réponse est "oui", de demander à l'utilisateur le taux de TVA à appliquer.

2. La structure de contrôle Si (forme simple)

La première possibilité pour n'effectuer des instructions que dans certaines situations est d'utiliser la structure de contrôle `Si`. Voici la syntaxe à utiliser pour ce test :

```
Si conditionBooléenne Alors
 SuiteDInstructions
FSi
```

Après le `Si`, il faut indiquer la condition qui doit être remplie pour exécuter la suite d'instructions comprise entre les mots-clés `Alors` et `FSi`. Le mot-clé `FSi` correspond à "Fin du Si". La condition doit être de type booléen : elle doit avoir pour valeur soit vrai soit faux. Si la condition a pour valeur vrai, alors l'ensemble de la suite des instructions sont exécutées, sinon elles sont sautées et l'exécution continue avec les instructions situées après le `FSi`.

Exemple :

```
Algo Tva2
# calcule le prix TTC d'un article
Variable prixHT : réel
Variable tvaReduite : texte
Variable tva : réel <- 20/100
Début
   écrire("Prix HT de l'article ?")
   prixHT <- saisir()
   écrire("Ce produit bénéficie-t-il d'un taux de TVA réduit ?")
   tvaReduite <- saisir()
   Si tvaReduite = "oui" Alors
     écrire("Quel est le taux (%) ?")
     tva  <- saisir()/100
   FSi
   écrire("Prix TTC de l'article : " & prixHT×(1+tva) & "€")
Fin
```

Dans cet exemple, la condition est `tvaReduite = "oui"`. Cette expression a bien une valeur booléenne puisque l'opérateur de comparaison égal donne comme résultat un booléen (cf. chapitre précédent). Les instructions comprises entre `Alors` et `FSi` ne sont exécutées qu'à la condition que la variable `tvaReduite` vaille "oui". Si elle vaut une autre valeur, alors ces instructions seront sautées et l'exécution se poursuivra avec l'affichage du prix TTC de l'article.

En Java, il faut utiliser le mot-clé `if` pour effectuer l'équivalent. La condition est positionnée entre un couple de parenthèses, et les instructions qui ne sont exécutées que si la condition est vraie sont positionnées dans un bloc d'instructions. Un bloc est matérialisé en plaçant les instructions entre un couple d'accolades.

© Editions ENI - All rights reserved

Exemple :

```
import java.util.Scanner;

public class Tva2 {
 public static void main(String[] args) {
   double tva = 20.0 / 100;
   double prixHT;
   Scanner console = new Scanner(System.in);
   System.out.println("Prix HT de l'article ?");
   prixHT = console.nextDouble();
   console.nextLine();
   System.out.println(
         "Ce produit bénéficie-t-il d'un taux de TVA réduit ?");
   String tvaReduite = console.nextLine();
   if(tvaReduite.equals("oui")) {
     System.out.println("Quel est le taux (%) ?");
     tva = console.nextDouble()/100;
     console.nextLine();
   }
   System.out.printf("Prix TTC de l'article : %.2f€%n",
                     prixHT * (1 + tva));
   console.close();
 }
}
```

Une variable qui est déclarée est utilisable uniquement entre le moment où elle est déclarée et la fin du bloc dans lequel elle a été déclarée. C'est ce qui se nomme la portée d'une variable. Si vous déclarez une variable au sein du bloc d'instructions, elle n'existera plus à la fin de celui-ci.

En Java, il ne faut pas utiliser les opérateurs == et != pour tester l'égalité de deux variables de type `String`. Il faut utiliser la méthode `equals()`. Dans l'exemple précédent, la condition pour comparer la valeur saisie avec le texte "oui" s'écrit donc :

```
tvaReduite.equals("oui")
```

S'il n'y a qu'une seule instruction qui ne doit être exécutée de manière conditionnelle, il est possible d'omettre les accolades.

Exemple :

```
if(tva < 5.5)
 System.out.println ("Taux de TVA très réduit");
```

3. La structure de contrôle Si (forme double)

La structure de contrôle `Si` peut se présenter dans une forme double. Dans ce cas-là, en plus des éléments présents dans sa version simple, s'ajoute après le mot-clé `Sinon` la suite d'instructions à effectuer lorsque la condition est fausse.Voici donc la syntaxe :

```
Si conditionBooléenne Alors
 suiteDInstructions
Sinon
 autreSuiteDInstructions
FSi
```

Si la condition est vraie, alors la suite d'instructions comprises entre `Alors` et `Sinon` est exécutée et l'exécution se poursuit après le `FSi`. Si par contre la condition est fausse, la première suite d'instructions est sautée et les instructions qui sont exécutées sont celles comprises entre `Sinon` et `FSi`, puis l'exécution se poursuit après le `FSi`.

Exemple :

```
Algo Age
Variable age : entier
Début
 age <- saisir("Quel est votre âge ? ")
 Si age ≥ 18 Alors
   écrire("Vous êtes majeur !")
 Sinon
   écrire("Vous êtes mineur !")
 FSi
Fin
```

Dans cet exemple, suivant la valeur saisie par l'utilisateur, le message "Vous êtes majeur !" ou "Vous êtes mineur !" s'affiche. En aucun cas les deux messages ne peuvent s'afficher.

© Editions ENI - All rights reserved

En Java, voici la syntaxe équivalente :

```
if(conditionBooléenne) {
  suiteDInstructions
} else {
 autreSuiteDInstructions
}
```

Voici donc le code Java correspondant à l'algorithme d'exemple précédemment présenté :

```
import java.util.Scanner;

public class Age {

   public static void main(String[] args) {
         Scanner console = new Scanner(System.in);
         System.out.println("Quel est votre âge ?");
         int age = console.nextInt();
         if(age>=18) {
                System.out.println("Vous êtes majeur !");
         } else {
                System.out.println("Vous êtes mineur !");
         }
         console.close();
   }
}
```

4. L'imbrication des structures de contrôle

Les structures de contrôle peuvent être imbriquées les unes dans les autres. C'est-à-dire que parmi la suite d'instructions contenues entre `Alors` et `Sinon` ou celles contenues entre `Sinon` et `FSi`, il est possible de positionner une autre structure de contrôle.

Exemple :

```
Algo Age2
Variable age : entier
Début
 age <- saisir("Quel est votre âge ? ")
 Si age < 0 Alors
   écrire("Ce n'est pas possible !")
```

```
 Sinon
   Si age ≥ 18 Alors
     écrire("Vous êtes majeur !")
   Sinon
     écrire("Vous êtes mineur !")
   FSi
 FSi
Fin
```

Dans cet exemple, il y a deux structures de contrôle imbriquées. Il y a tout d'abord la structure de contrôle `Si` englobante indiquée en gras ci-dessous et ensuite le `Si` englobé indiqué en italique ci-dessous :

```
Algo age2
Variable age : entier
Début
 age <- saisir("Quel est votre âge ? ")
 Si age < 0 Alors
   écrire("Ce n'est pas possible !")
 Sinon
   Si age ≥ 18 Alors
     écrire("Vous êtes majeur !")
   Sinon
     écrire("Vous êtes mineur !")
   FSi
 FSi
Fin
```

Si l'âge est négatif, alors le message "Ce n'est pas possible !" est affiché puis les instructions jusqu'au `FSi` en gras sont sautées. Si l'âge n'est pas négatif, alors un nouveau test est effectué pour savoir si celui-ci est plus grand ou égal à 18. Si c'est le cas, le message "Vous êtes majeur !" est affiché, sinon c'est le message "Vous êtes mineur !" qui est affiché.

En Java, il est bien évidemment possible également d'imbriquer les structures de contrôle. Voici l'algorithme précédent codé en Java :

```
import java.util.Scanner;

public class Age2 {
   public static void main(String[] args) {
         Scanner console = new Scanner(System.in);
         System.out.println("Quel est votre âge ?");
         int age = console.nextInt();
```

© Editions ENI - All rights reserved

```
        if (age < 0) {
                System.out.println("Ce n'est pas possible");
        } else {
                if (age >= 18) {
                        System.out.println("Vous êtes majeur !");
                } else {
                        System.out.println("Vous êtes mineur !");
                }
        }
        console.close();
    }
}
```

Il est à noter qu'après le premier `else`, le bloc ne contient que la seconde structure de contrôle `if`, il n'est donc pas obligatoire de mettre les accolades autour de celle-ci. Cela permet de rendre le code un peu plus lisible :

```
import java.util.Scanner;

public class Age3 {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.println("Quel est votre âge ?");
        int age = console.nextInt();
        if (age < 0) {
            System.out.println("Ce n'est pas possible");
        } else if (age >= 18) {
            System.out.println("Vous êtes majeur !");
        } else {
            System.out.println("Vous êtes mineur !");
        }
        console.close();
    }
}
```

Remarque

Dans cet exemple, une structure de contrôle `Si` a été imbriquée au sein d'une autre structure de contrôle `Si`. Pour le moment, c'est la seule structure de contrôle qui a été présentée, mais il en existe beaucoup d'autres. Il est à noter qu'il est également possible d'imbriquer n'importe quelle structure de contrôle au sein de n'importe quelle structure de contrôle.

5. La structure de contrôle Selon

La structure de contrôle `Selon` permet également de n'effectuer que certaines instructions de manière conditionnelle. `Selon` permet d'effectuer les instructions souhaitées en fonction de la valeur d'une variable. Voici la syntaxe de cette structure de contrôle :

```
Selon nomDeLaVariable
    cas premièreListeDeValeurs : premièreSuiteDInstructions
    cas deuxièmeListeDeValeurs : deuxièmeSuiteDInstructions
    cas troisièmeListeDeValeurs : troisièmeSuiteDInstructions
...
    autre : autreSuiteDInstructions
FSelon
```

Si la valeur de la variable fait partie de la première liste de valeurs, alors c'est la première suite d'instructions qui est exécutée, puis l'exécution reprend après `FSelon`. Si ce n'est pas le cas, la valeur de la variable est recherchée dans la deuxième liste de valeurs. Si elle est présente, alors la deuxième suite d'instructions est exécutée, puis l'exécution reprend après `FSelon`. Cela est réalisé jusqu'à trouver le cas contenant dans sa liste la valeur de la variable.

Le cas `autre` est facultatif. Si celui-ci est présent et si la valeur de la variable n'a été trouvée dans aucun cas, ce sont les instructions situées après `autre` qui sont exécutées.

Exemple :

```
Algo TriSelectif
Variable dechet : texte
Début
 dechet <- saisir("Que souhaitez-vous jeter ?")
 Selon dechet
   cas "papier"; "carton"; "boîte de conserve" :
     écrire("à recycler")
   cas "végétaux"; "épluchures" :
     écrire("à composter")
   autre :
     écrire("à mettre à la poubelle")
 FSelon
Fin
```

© Editions ENI - All rights reserved