C#

Instructions conditionnelles et alternatives

SOMMAIRE

Les structures conditionnelles 2
L'action conditionnée 2
La structure alternative..... 4
Le choix multiple 6
Expression de la condition 10
Exercices 12

Les structures conditionnelles

L'action conditionnée

L'action conditionnée est une instruction élémentaire ou une suite d'instructions **exécutées** en séquence **si l'état du système l'autorise**. Le(s) critère(s) à respecter pour exécuter l'action s'exprime(nt) à l'aide d'une condition (ou **prédicat**) évaluable au moment précis où l'action doit, le cas échéant, intervenir.
Lors de l'exécution du programme, le processeur est donc amené à évaluer la condition. La condition évaluée constitue alors un énoncé (ou **proposition**) vrai ou faux.

Schéma :

```
Si prédicat alors
   Instruction 1
   Instruction 2
   …
   Instruction N
Fin si
```

Exemples :

Si Température > 38 **alors**
  Écrire "Le patient a de la fièvre"
Fin si
	Si Température > 41 **et** Tension > 25 **alors**
	  Écrire "Le patient va perdre patience"
	Fin si

Si non Patient **alors**
  Écrire "Éconduire l'olibrius"
Fin si
	Si Température > 42 **ou** (Tension < 25 **et** Pouls > 180) **alors**
	  Écrire "Prévenir la famille"
	Fin si

Si Température > 40 **ou** Tension ≥ 25 **alors**
  Écrire "Hospitaliser le patient"
Fin si
	Si Patient **et** Pouls = 0 **alors**
	  Patient ← **non** Patient
	Fin si

Syntaxe C# :

```
if (prédicat)
{
   Instruction 1
   Instruction 2
   …
   Instruction N
}
```

Exemple 1: réaliser le programme qui après lecture d'un entier au clavier, affiche un message si l'entier est pair.

```
class Program
{
  static void Main(string[] args)
  {
    string strLigne;
    int entier;
    int reste;
    // saisie utilisateur
    Console.WriteLine("Saisissez un entier :");
    strLigne = Console.ReadLine();
    // conversion en entier
    entier = Int32.Parse(strLigne);
    // Reste de la division par 2
    reste = entier % 2;
    if (reste == 0)
    {
      Console.WriteLine ("Entier pair !");
    }
    // Affichage persistant
    Console.ReadLine();
  }
}
```

Notez l'opérateur relationnel == qui teste l'égalité. Tous les opérateurs relationnels seront vus dans le prochain paragraphe.
Remarquez l'indentation (retrait de paragraphe) qui permet de voir facilement la portée de la condition.
A noter : si une seule instruction est à exécuter dans le IF, les accolades ne sont pas obligatoires …

```
if (reste == 0)
  Console.WriteLine ("Entier pair !");
```

Mais le code est tellement plus lisible et évite des problèmes futurs de rajout d'instructions!
Les conditions peuvent être imbriquées :

Exemple 2: Modifier le programme précédent qui après lecture d'un entier au clavier, affiche un message si l'entier est pair ; indiquez également si l'entier est multiple de 4.

```
…
// Reste de la division par 2
if (entier % 2 == 0)
{
  Console.WriteLine ("Entier pair !");
  // Reste de la division par 4
  if (entier % 4 == 0)
  {
      Console.WriteLine("Entier multiple de 4 !");
  }
}
```

Pourquoi les conditions sont-elles imbriquées ? Tout simplement parce qu'un nombre impair ne peut pas être un multiple de 4… le traitement ne concerne donc que les nombres pairs.
Notez la notation différente du prédicat dans cet exemple … Les 2 notations sont équivalentes, l'important étant que le prédicat puisse être évalué.

Exemple 3: Et de la même façon, si on avait voulu également savoir si l'entier était également multiple de 6, on aurait pu écrire :

```
…
// Reste de la division par 2
if (entier % 2 == 0)
{
  Console.WriteLine ("Entier pair !");
  // Reste de la division par 4
  if (entier % 4 == 0)
  {
       Console.WriteLine("Entier multiple de 4 !");
  }

  // Reste de la division par 6
  if (entier % 6 == 0)
  {
       Console.WriteLine("Entier multiple de 6 !");
  }
}
```

Un entier pair peut être à la fois multiple de 4 et de 6 (12), mais peut être multiple de 4 sans être multiple de 6 (8) ou l'inverse (18).

La structure alternative

La structure alternative traduit la nécessité d'exécuter, à un instant donné, une action, élémentaire ou composée, ou une autre action, elle-même élémentaire ou composée, exclusivement l'une de l'autre.

Schéma :

```
Si prédicat alors
     Instructions_si_vrai;
Sinon
     Instructions_si_faux;
Fin si
```

Exemples :

Si Nombre = 0 **alors**
  Écrire "Impossible de diviser par 0"
sinon
  Moyenne = Total / Nombre
  Écrire Moyenne
Fin si
	Si C= "A" **ou** C= "E" **ou** C= "I" **ou** C= "O" **ou** C= "U" **alors**
	  Écrire C, " est une voyelle"
	sinon
	  Écrire C, " est une consonne"
	Fin si

Syntaxe C# :

```
if (prédicat)
{
    Instructions_si_vrai;
}
else
{
    Instructions_si_faux;
}
```

Exemple : Modifier le programme précédent qui après lecture d'un entier au clavier, affiche un message si l'entier est pair, pour qu'il affiche un message si l'entier est impair.

```
…
// Reste de la division par 2
if (entier % 2 == 0)
{
  Console.WriteLine ("Entier pair !");
  // Reste de la division par 4
  if (entier % 4 == 0)
  {
      Console.WriteLine("Entier multiple de 4 !");
  }

  // Reste de la division par 6
  if (entier % 6 == 0)
  {
      Console.WriteLine("Entier multiple de 6 !");
  }
}
else
{
 Console.WriteLine("Entier impair !");
}
```

Le choix multiple

La sélection (ou **choix multiple**) permet de présenter une solution claire à des problèmes où un nombre important de cas, mutuellement exclusifs, sont à envisager en fonction des valeurs prises par un nombre réduit de variables (généralement une seule). Puisque chaque action est exclusive des autres, la structure sélective correspond à une **imbrication d'alternatives.**

Cette structure vise en fait à réduire la complexité engendrée par la présence d'un nombre élevé de cas à envisager. Elle privilégie ainsi l'essentiel (la recherche des différents cas et des actions à leur associer) sans négliger l'accessoire (la lisibilité de la solution).

Exemple 1: En fonction du caractère saisi à l'écran, affichez le statut de la personne en clair.

```
namespace Test
{
 class Statut
 {
  static void Main(string[] args)
  {

   string strStatut;
   // saisie utilisateur
   Console.WriteLine("Saisissez le statut :");
   strStatut = Console.ReadLine();

   if (strStatut == "C")
   {
    Console.WriteLine("Célibataire !");
   }
   else
   {
    if (strStatut == "M")
    {
     Console.WriteLine("Marié !");
    }
    else
    {
     if (strStatut == "V")
     {
      Console.WriteLine("Veuf !");
     }
     else
     {
      if (strStatut == "D")
      {
       Console.WriteLine("Divorcé !");
      }
     }
    }
   }
```

```
    // Affichage persistant
    Console.ReadLine();
   }
  }
}
```

Traité avec une imbrication de if, la solution est peu lisible, mais peut être améliorée en utilisant le **If étendu** :

Exemple 2:

```
namespace Test
{
 class Statut
 {
  static void Main(string[] args)
  {

   string strStatut;
   // saisie utilisateur
   Console.WriteLine("Saisissez le statut :");
   strStatut = Console.ReadLine();

   if (strStatut == "C")
   {
    Console.WriteLine("Célibataire !");
   }
   else if (strStatut == "M")
   {
    Console.WriteLine("Marié !");
   }
   else if (strStatut == "V")
   {
    Console.WriteLine("Veuf !");
   }
   else if (strStatut == "D")
   {
    Console.WriteLine("Divorcé !");
   }
   // Affichage persistant
   Console.ReadLine();
  }
 }
}
```

L'utilisation de l'instruction **switch** va encore apporter une autre lisibilité :

Syntaxe C#

```
switch (expression)
{
case valeur1:
       Instructions_Valeur1;
       break;
case valeur2:
       Instructions_Valeur2;
       break;
case valeur3:
       Instructions_Valeur3;
       break;
default:
       Instructions_Defaut;
       break;
}
```

Si **expression** est égale à **valeur1** on exécute les instructions **Instructions_Valeur1**.
Si **expression** est égale à **valeur2** on exécute les instructions **Instructions_Valeur2**.
Si **expression** est égale à **valeur3** on exécute les instructions **Instructions_Valeur3**.
Si **expression** n'est égale à aucune des valeurs énumérées, on exécute **Instructions_Defaut**.

Pour éviter d'exécuter tous les pavés d'instructions en séquence à partir du moment où l'égalité a été trouvée, il est obligatoire de mettre une instruction **`break`** (rupteur) à la fin de chaque pavé : cette instruction provoque un débranchement à l'instruction suivant l'accolade fermante.

Exemple 3:

```
namespace Test
{
 class Statut
 {
  static void Main(string[] args)
  {

   string strStatut;
   // saisie utilisateur
   Console.WriteLine("Saisissez le statut :");
   strStatut = Console.ReadLine();
   switch (strStatut)
   {
     case "C":
        Console.WriteLine("Célibataire !");
        break;
     case "M":
        Console.WriteLine("Marié !");
```

```
            break;
        case "V":
            Console.WriteLine("Veuf !");
            break;
        case "D":
            Console.WriteLine("Divorcé !");
            break;
        default :
            Console.WriteLine("Saisie incorrecte !");
            break;
    }
    // Affichage persistant
    Console.ReadLine();
  }
 }

}
```

Expression de la condition

Opérateurs relationnels

Si les opérandes peuvent être des expressions arithmétiques quelconques, le résultat de ces opérateurs est de type **`bool`**.

`==`	Egalité	**`a == b`**
`!=`	Différence	**`a != b`**
`<`	Inférieur	**`a < b`**
`>`	Supérieur	**`a > b`**
`<=`	Inférieur ou égal	**`a <= b`**
`>=`	Supérieur ou égal	**`a >= b`**

Ce résultat peut, bien sur, être utilisé comme opérande des opérateurs booléens

Opérateurs booléens

Les opérateurs booléens ne reçoivent que des opérandes de type **`bool`**.Ces opérandes peuvent avoir la valeur vraie (**`true`**)ou faux (**`false`**).

`&`	ET logique Le résultat donne vrai (valeur **`true`**) si les deux opérandes ont pour valeur vrai Le résultat donne faux (valeur **`false`**) si l'un des deux opérandes a pour valeur faux.	**`a > b & a < c`**
`&&`	ET logique Cet opérateur fonctionne comme le précédent, à la différence que le deuxième opérande (à droite) n'est pas évalué (calculé) si le premier a pour valeur faux. Car quelle que soit la valeur du deuxième opérande, le résultat est forcément faux.	**`a > b && a < c`**
`|`	OU logique Le résultat donne vrai (valeur **`true`**) si l'un des deux opérandes a pour valeur vrai. Le résultat donne faux (valeur **`false`**) si les deux opérandes ont pour valeur faux.	**`a == b | a == c`**
`||`	OU logique Cet opérateur fonctionne comme le précédent, à la différence que le deuxième opérande (à droite) n'est pas évalué (calculé) si le premier a pour valeur vrai. Car quelle que soit la valeur du deuxième opérande, le résultat est forcément vrai.	**`a == b || a == c`**
`!`	NON logique Le résultat est faux si l'expression est vraie et inversement	**`!(a <= b)`**

Autre opérateur

Il existe, en C#, un opérateur à trois opérandes :

? :	Structure alternative.	**n = k>3 ? 5 : 6;**

L'exemple sera interprété :
Si **k** est supérieur à **3**, **n** vaudra **5** sinon **6**.

Exemple :

```
if (i ==1 && j !=2)
{
      …
}
```

Signifiera si i vaut 1 et j est différent de 2, alors …

Exercices

Parité

Ecrivez un programme qui demande un nombre à l'utilisateur puis qui test si ce nombre est pair. Le programme doit afficher le résultat « nombre pair » ou « nombre impair ». Vous devez utiliser l'opérateur modulo « % » qui donne le reste d'une division. a%2 donne le reste de la division de a par 2, si ce reste est égale à zéro, a est divisible par 2.

Age

Ecrivez un programme qui demande l'année de naissance à l'utilisateur.
En réponse votre programme doit afficher l'âge de l'utilisateur et indiquer si l'utilisateur est majeur ou mineur.

Calculette

Faire la saisie de 2 nombres entiers, puis la saisie d'un opérateur '+', '-', '*' ou '/'.Si l'utilisateur entre un opérateur erroné, le programme affichera un message d'erreur. Dans le cas contraire, le programme effectuera l'opération demandée (en prévoyant le cas d'erreur "division par 0"), puis affichera le résultat.

Remise

A partir de la saisie du prix unitaire d'un produit PU et de la quantité commandée QTECOM, afficher le prix à payer PAP, en détaillant le port PORT et la remise REM, sachant que :

- le port est gratuit si le prix des produits TOT est supérieur à 500 €. Dans le cas contraire, le port est de 2% de TOT.
- la valeur minimale du port à payer est de 6 €.
- la remise est de 5% si TOT est compris entre 100 et 200 € et de 10% au delà.

La participation

Un patron décide de calculer le montant de sa participation au prix du repas de ses employés de la façon suivante :

si il est célibataire	participation de 20%
si il est marié	participation de 25%
si il a des enfants	participation de 10% supplémentaires par enfant

La participation est plafonnée à 50%
Si le salaire mensuel est inférieur à 1200 € la participation est majorée de 10%

Ecrire le programme qui lit les informations au clavier et affiche pour chaque salarié, la participation à laquelle il a droit.