Chapitre 3 : Les structures de contrôle en C#

3.1 Structure de choix simple

syntaxe : if (condition) {actions_condition_vraie;} else {actions_condition_fausse;}

Notes:

- la condition est entourée de parenthèses.
- chaque action est terminée par point-virgule.
- les accolades ne sont pas terminées par point-virgule.
- les accolades ne sont nécessaires que s'il y a plus d'une action.
- la clause *else* peut être absente.
- il n'y a pas de clause *then*.

L'équivalent algorithmique de cette structure est la structure *si .. alors … sinon* :

```
si condition
        alors actions_condition_vraie
        sinon
        actions_condition_fausse
finsi
```

<u>Exemple</u>

```
if (x>0) { nx=nx+1;sx=sx+x;} else dx=dx-x;
```

On peut imbriquer les structures de choix :

```
if(condition1)
        if (condition2)
        {......}
        else //condition2
        {......}
else //condition1
{.......}
```

Soit le code suivant :

```
1. using System;
2.
3. namespace Chap3 {
4. class C3       {
5. static void Main(string[] args) {
6. int n = 5;
7. if (n > 1)
8.    if (n > 6)
9.          Console.Out.WriteLine(">6");
10.   else Console.Out.WriteLine("<=6");
11.         }
12.   }
```

Dans l'exemple précédent, le *else* de la ligne 10 se rapporte à quel *if* ? La règle est qu'un *else* se rapporte toujours au *if* le plus proche : *if(n>6)*, ligne 8, dans l'exemple.

Considérons un autre exemple :

```
if (n2 > 1)   {
            if (n2 > 6) Console.Out.WriteLine(">6");
            } else Console.Out.WriteLine("<=1");
```

Ici nous voulions mettre un *else* au *if(n2>1)* et pas de *else* au *if(n2>6)*. A cause de la remarque précédente, nous sommes obligés de
mettre des accolades au *if(n2>1) {...} else ...*

3.2 Structure à choix multiple avec switch

La syntaxe est la suivante :

```
switch(expression) {
        case v1:
                actions1;
                break;
        case v2:
                actions2;
                break;
        ..
        default:
                actions_sinon;
                break;
}
```

<u>Remarques</u>

- la valeur de l'expression de contrôle du *switch* peut être un entier, un caractère, une chaîne de caractères
- l'expression de contrôle est entourée de parenthèses.
- la clause *default* peut être absente.
- les valeurs *vi* sont des valeurs possibles de l'expression. Si l'expression a pour valeur *vi* , les actions derrière la clause **case vi** sont exécutées.
- l'instruction *break* fait sortir de la structure de cas.
- chaque bloc d'instructions lié à une valeur *vi* doit se terminer par une instruction de branchement (break, goto, return, ...) sinon le compilateur signale une erreur.

<u>Exemple</u>

```
int choix = 2;
bool erreur = false;
switch (choix) {
   case 0: return;
   case 1: M1(); break;
    case 2: M2(); break;
   default: erreur = true; break;
    }
}// fin Main

static void M1() {
   Console.WriteLine("M1");
}

```

```
static void M2() {
  Console.WriteLine("M2");
}
}
```

3.3 Arrêt

La méthode E*xit* définie dans la classe *Environment* permet d'arrêter l'exécution d'un programme.
syntaxe : *void Exit(int status)*
action : *arrête le processus en cours et rend la valeur status au processus père*

Exit provoque la fin du processus en cours et rend la main au processus appelant. La valeur de *status* peut être utilisée par celui-ci.

Sous DOS, cette variable status est rendue dans la variable système **ERRORLEVEL** dont la valeur peut être testée dans un fichier batch.
Sous Unix, avec l'interpréteur de commandes Shell Bourne, c'est la variable **$?** qui récupère la valeur de *status*.
Environment.Exit(0);

3.4 Structures de répétition

3.4.1 Nombre de répétitions connu

Structure for

La syntaxe est la suivante :

```
for (i=id;i<=if;i=i+ip){
        actions;
        }
```

Remarques :

- les 3 arguments du *for* sont à l'intérieur d'une parenthèse et séparés par des points-virgules.
- chaque action du *for* est terminée par un point-virgule.
- l'accolade n'est nécessaire que s'il y a plus d'une action.
- l'accolade n'est pas suivie de point-virgule.

Structure foreach

La syntaxe est la suivante :

```
foreach (Type variable in collection){
        instructions;
        }
```

Remarques :

- *collection* est une collection d'objets énumérable. La collection d'objets énumérable que nous connaissons déjà est le tableau
- *Type* est le type des objets de la collection. Pour un tableau, ce serait le type des éléments du tableau

- *variable* est une variable locale à la boucle qui va prendre successivement pour valeur, toutes les valeurs de la collection.

Ainsi le code suivant :

```
string[] amis = { "Mohamed", "Sami", "Salima", "Mounir" };
foreach (string nom in amis) {
Console.WriteLine(nom);
}
```

afficherait :
Mohamed
Sami
Salima
Mounir

3.4.2 Nombre de répétitions inconnu

Il existe de nombreuses structures en C# pour ce cas.

Structure tantque (while)

```
while(condition){
        actions;
}
```

La boucle se répète tant que la condition est vérifiée. La boucle peut ne jamais être exécutée.

Remarques :
- la condition est entourée de parenthèses.
- chaque action est terminée par point-virgule.
- l'accolade n'est nécessaire que s'il y a plus d'une action.
- l'accolade n'est pas suivie de point-virgule.

Structure répéter jusqu'a (do while)

La syntaxe est la suivante :

```
do{
        instructions;
}while(condition);
```

La boucle se répète jusqu'à ce que la condition devienne fausse. Ici la boucle est faite au moins une fois.

Remarques :
- la condition est entourée de parenthèses.
- chaque action est terminée par point-virgule.
- l'accolade n'est nécessaire que s'il y a plus d'une action.
- l'accolade n'est pas suivie de point-virgule.

Structure pour générale (for)

La syntaxe est la suivante :
for(instructions_départ;condition;instructions_fin_boucle){
    instructions;
}
La boucle se répète tant que la condition est vraie (évaluée avant chaque tour de boucle).
Instructions_départ sont effectuées avant d'entrer dans la boucle pour la première fois.
Instructions_fin_boucle sont exécutées après chaque tour de boucle.

Les différentes instructions dans *instructions_depart* et *instructions_fin_boucle* sont séparées par des virgules.

<u>Exemples</u>

Les fragments de code suivants calculent tous la somme des 10 premiers nombres entiers.

```
int i, somme, n=10;
for (i = 1, somme = 0; i <= n; i = i + 1)
somme = somme + i;
for (i = 1, somme = 0; i <= n; somme = somme + i, i = i + 1) ;
i = 1; somme = 0;
while (i <= n) { somme += i; i++; }
i = 1; somme = 0;
do somme += i++;
while (i <= n);
```

<u>Remarques :</u>
Break : fait sortir de la boucle for, while, do ... while.
Continue : fait passer à l'itération suivante des boucles for, while, do ... while