Structures répétitives

R1.01 – Initiation au développement

N. Gruson

IUT ANNECY
UNIVERSITÉ SAVOIE MONT BLANC

INFO | INFORMATIQUE

Pourquoi ?

- Pour recommencer un traitement

  Ex : on veut facturer les consommations en eau d'une ville : il faut traiter toutes les relèves de compteur d'eau et calculer les consommations de tous les foyers

- Pour vérifier une saisie utilisateur

  Ex : on attend une réponse Oui/Non (O/N) à une question. On redemandera à l'utilisateur tant que sa réponse ne sera pas conforme aux attentes

Structure : do….while

Ou Faire … Tant Que( )

Les instructions sont exécutées une 1ere fois, et le seront à nouveau si la condition est vraie => AU MOINS UNE FOIS !

Syntaxe:

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

Instructions indentées !

Attention : ne pas oublier ;

La condition peut être simple ou multiple.

Quand il n'y a qu'une seule instruction les { } ne sont pas obligatoires

Structure : do….while

Utile pour relancer un programme en fonction d'une réponse utilisateur.

Exemple :

```
String recommencer ;
do
 {
      // instructions
      //  …..
       Console.WriteLine(" Recommencer  (O/N) ? ");
       recommencer = Console.ReadLine();
    } while ( recommencer == "O");
```

While = if qui se répète

Structure : do … while

Utile pour relancer un programme avec un menu.

Exemple :

```
String choix ;
do
 {
    Console.WriteLine( "0 : Quitter ");
    Console. WriteLine ( "1 : Jouer ");
    Console. WriteLine( "2 : Voir les scores ");
    choix  = Console.ReadLine();
    switch ( choix )
          { …. }
} while ( choix != "0" );
```

Structure : while

Ou Tant que ( ) faire …

Les instructions peuvent ne jamais être exécutées

=> le test a lieu avant !

Syntaxe:

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

Attention : ne pas mettre;

Structure : while

Utile pour vérifier une saisie utilisateur. Exemple 1

```
int res;
bool reussi = int.TryParse(Console.ReadLine(), out res);
while ( ! reussi )
    {
      Console.WriteLine("Erreur de saisie. Entier attendu ! ");
      reussi = int.TryParse(Console.ReadLine(), out res);
     }
return res;
```

Structure : while

Utile pour vérifier une saisie utilisateur. Exemple 2

```
String  type ;
 Console.WriteLine( " votre type (F/H)  ");
 type = Console.ReadLine().ToUpper() ;
 while( ( type != "H")  && ( type != "F") )
    {
     Console. WriteLine( " Erreur de saisie.  ");
     Console. WriteLine( " votre type (F/H)  ");
     type = Console. ReadLine();
     }
```

! Le non logique

! ⇔ pas

Utile pour répéter tant que la condition n'est **pas vérifiée**.

Exemple :

```
// …
while(  !  ( ( type == "H") || ( type == "F")) )
{
    Console. WriteLine( " Erreur de saisie.  ");
    Console. WriteLine( " votre type (F/H)  ");
    type = Console.ReadLine();
 }
```

((type != "H") && (type != "F"))

Boucle infinie avec un while

Faire une boucle volontairement infinie peut être utile pour un jeu !

Exemple :

```
while (true)
 {
      // instructions du jeu…..
 }
```

Structure : for

Ou Pour

Lorsque le nombre de répétitions est connu ou calculable.

Syntaxe : pour i qui commence à debut augmente de nb jusqu'à fin

```
for (int i  = debut ; i <=  fin ; i = i + nb )
 {
     instructions;
 }
```

ou <

Syntaxe : pour i qui commence à debut diminue de nb jusqu'à fin

```
for (int i  = debut ; i >= fin ; i = i - nb )
 {
     instructions;
 }
```

ou >

On a pour coutume d'appeler la variable qui sert de compteur i

Structure : for

Exemple classique : pas de 1

Exemple :

i++
i+=1

Condition de continuité

```
for ( int i=1 ; i <= 10 ; i = i + 1 )
    { Console.WriteLine (i); }
```

1
2
…
10

Le pas n'est pas obligatoirement de 1 !

Exemple :

```
for ( int i=2 ; i <= 10 ; i = i + 2 )
    { Console. WriteLine (i); }
```

2
4
…
10

Structure : for

La valeur de début de boucle commence souvent à 0 ou 1, mais ce n'est pas une obligation.

Exemple :

```
for (int i = 20 ; i <= 30 ; i ++)
 { Console.WriteLine(i); }
```

20
21
…
30

Le pas peut être négatif.

Exemple :

i-=2;

```
for ( int i = 6; i >= 0 ; i = i -2 )
 { Console. WriteLine (i); }
```

6
4
…
0

Structure : for

Les valeurs de début, de fin et de pas peuvent être :

- **des variables, des expressions .**

Exemple :

```
int  nb1,nb2 ;
// instructions pour initialiser nb1, nb2

for ( int i = nb1 ; i <= nb2 ; i ++)
 { Console. WriteLine(i); }

for (int  i = nb1 ; i <= nb1 +10  ; i ++)
 { Console. WriteLine(i); }
```

Structure : for

- Les variables de départ peuvent être multiples.
- La condition de continuité peut être multiple assemblée par des opérateurs logiques.

Exemple : on donne 3 chances à un enfant pour donner un chiffre pair

```
int i; bool pair;

for( int  i = 1, pair = false     ;   i<=3 && pair == false   ;  i++     )
 {
   int val ;
   Console.WriteLine ("entre une valeur paire ");
   val  = int.Parse(Console.ReadLine());
   if(val%2==0)
    { Console. WriteLine ("Bien");
      pair = true;
     }
   else
     Console. WriteLine ("Non, " +val +" n'est pas une valeur paire ." );
 }
```

For, while, do … while

Souvent, les structures sont interchangeables …. Cependant, il y en a toujours une plus adaptée …

```
i = 1;
do
{
   Console.WL(" Prénom de l'étudiant "+ i);
   prenom = Console.RL();
   Console.WL(" Bonjour " + prenom ) ;
   i ++ ;
} while ( i <=30);
```

```
i = 1 ;
while ( i <=30)
{
   Console.WL(" Prénom de l'étudiant  "+ i);
   prenom = Console.RL();
   Console.WL(" Bonjour "+ prenom ) ;
   i ++ ;
}
```

```
for ( i =1 ; i <= 30 ; i++ )
{
        Console.WL(" Prénom de l'étudiant "+ i);
      prenom = Console.RL();
      Console.WL(" Bonjour "+ prenom ) ;
 }
```

Plus adaptée dans ce cas !

Règles de bon sens

- Utilisez la structure la plus adaptée !
- Veillez à ne pas écrire de boucle infinie sans le vouloir :
  - Vérifiez que la variable, utilisée pour la condition, évolue !
  - Plus généralement, vérifiez que la condition peut devenir fausse !

Exemple incorrect:

```
Console.WL( " Genre :  F ou H :  ");
type = Console.RL();
while( ( type != "H")  && ( type != "F") )
  {
      Console.WL( " Erreur de saisie .    ");
  }
```

type n'évolue pas dans la boucle

Exemple incorrect:

```
i = 0 ;
while ( i != 9 )
 i = i + 2 ;
```

i sera toujours != 9

Imbriquer des boucles

Pourquoi pas ? On peut tout imbriquer !

Exemple :

```
char type, recommence ;
do
{
 Console.WL( " votre type (F/H)  ");
 type = Console.RL();
 while( ( type != "H")  && ( type != "F") )
        {
            Console.WL( " Erreur de saisie . F ou H :  ");
             type = Console.RL();
         }
 Console.WL (" recommencez (O/N) \n") ;
 recomence = Console.RL();
} while (recommence = = "O")  ;
```

Il est préférable de déclarer les variables en dehors de tout bloc

A vous de jouer

Que vont afficher ces instructions ?

```
for( i = 1 ; i <= 2 ; i++ )
   Console.WL ( "A droite " + i + " fois " );

for( j = 1 ; j <= 3 ; j++ )
 Console.WL ( "A gauche " + j + " fois "  );
```

A droite 1 fois

A droite 2 fois

A gauche 1 fois

A gauche 2 fois

A gauche 3 fois

A vous de jouer

Que vont afficher ces instructions ?

```
for( i = 1 ; i <= 2 ; i++ )
{
   Console.WL ( "A droite " + i + " fois"  );
   for( j = 1 ; j <= 3 ; j++ )
      Console.WL ( "A gauche " + j +  " fois" );
}
```

A droite 1 fois
A gauche 1 fois
A gauche 2 fois
A gauche 3 fois

A droite 2 fois
A gauche 1 fois
A gauche 2 fois
A gauche 3 fois

A vous de jouer

Que vont afficher ces instructions ?

```
for( i = 1 ; i <= 2 ; i++ )
{
      for( j = 1 ; j <= 3 ; j++ )
       {
          Console.WL ( "A droite " + i + " fois " );
          Console.WL ( "A gauche " + j + " fois "  );
       }
}
```

A droite 1 fois
A gauche 1 fois
A droite 1 fois
A gauche 2 fois
A droite 1 fois
A gauche 3 fois

A droite 2 fois
A gauche 1 fois
A droite 2 fois
A gauche 2 fois
A droite 2 fois
A gauche 3 fois

Vérifier une saisie avec des expressions régulières

Il est possible de :
- créer un « format personnalisé » : expressions régulières
- tester si une chaîne respecte le « format personnalisé »

Dans la classe Regex, 2 surcharges de IsMatch
- public static bool IsMatch (string input, string pattern);
- public bool IsMatch (string input);

```
Regex formatDate = new Regex("^[0-9]{2}/[0-9]{2}/[0-9]{4}$");
Console.WriteLine("Date Naissance?");
String date = Console.ReadLine();
while ( ! formatDate.IsMatch(date))
   {
    Console.WriteLine("Erreur. Format attendu dd/mm/yyyy ?");
                date = Console.ReadLine();
          }
```

Expressions régulières

- Attention : chaque caractère employé aura soit un sens particulier, soit obligation d'être présent dans la chaine.
- Caractères spécifiques pour les expressions régulières :
  - ^ signifie début de chaîne
  - $ signifie fin de chaîne

```
Regex commenceParA = new Regex("^A");
Regex finitParA = new Regex("A$");
Regex A = new Regex("^A$");
```

Expressions régulières

[ ] : permet de lister les caractères autorisés. Ex [abc] : seuls abc autorisés

On peut utiliser le symbole - pour exprimer une liste .

[a-z] : tous les caractères de a à z sont autorisés.

[A-Z] : tous les caractères de A à Z sont autorisés.

[0-9] : tous les caractères numériques

Il existe des listes prédéfinies. Ex [:alpha:] ⇔ A-Za-z

```
Regex commenceParUneMaj = new Regex("^[A-Z]");
Regex unChiffre = new Regex("^[0-9]$");
```

Expressions régulières

On peut quantifier :

* pour 0 ou plusieurs .
  Ex : [0-9]* : chiffres ou rien
+ pour 1 ou plusieurs.
  Ex : [0-9]+ : au moins 1 chiffre
{nb} pour exactement nb.
  Ex : [0-9]{2} : 2 chiffres
{min,max} .
  Ex : [0-9]{2,4} : de 2 à 4 chiffres
? 0 ou 1.
  Ex : [0-9] ? : : 1 chiffre ou rien

```
Regex commencePar2chiffres = new Regex("^[0-9]{2} ");
Regex auMoinsUneLettre = new Regex("^[a-z]+$");
```

Expressions régulières

- pour exprimer n'importe quel caractère.
  Ex : ^.{3}$ chaine de 3 caractères
  Ex : ^.-.$ 2 caractères séparés par un tiret

| pour exprimer ou.
  Ex : ^Le| ^La chaine qui commence par Le ou La

( ) pour regrouper

\ pour annuler un symbole et utiliser le caractère.
  Ex : [0-9]\.[0-9]$ 2 chiffres séparés par un point