

LES STRUCTURES DE CONTROLE

SANE ARNAUD

Table des matières



I - Objectifs	3
II - LES STRUCTURES CONDITIONNELLES	4
III - EVALUATION 1	8
IV - LES STRUCTURES ITERATIVES	10
V - EVALUATION 2	13



Objectifs

Etre capable de :

- Manipuler les structures conditionnelles en Visual Basic.Net
- Manipuler les structures itératives en Visual Basic.Net

LES STRUCTURES CONDITIONNELLES



Définition

Les structures de contrôle permettent d'appliquer des restrictions sur l'exécution d'un bloc d'instructions. Les structures conditionnelles font partie des structures de contrôle car elles exécutent un bloc d'instructions quand la condition est vérifiée.

Ainsi, nous distinguons plusieurs types de structures conditionnelles :

LA STRUCTURE CONDITIONNELLE SIMPLE

Définition : Elle permet d'exécuter un bloc d'instructions lorsque la condition posée est vérifiée

Syntaxe :

If condition Then

.....Bloc d'instructions

End If

Exemple :

L'exemple défini ci-dessous retourne la valeur absolue d'un nombre entier saisi par l'utilisateur.

Ceci dit, Après qu'il ait saisi le nombre entier, s'il est négatif, le programme retourne son contraire.

```

1 Module Module1
2
3     Sub Main()
4         Dim x, y As Integer
5         Console.WriteLine("Entrez un nombre entier Svp!")
6         x = Console.ReadLine()
7         y = x
8         If x < 0 Then
9             y = -1 * x
10        End If
11        Console.WriteLine("La valeur absolue de " & x & " est " & y)
12        Console.Read()
13    End Sub
14
15 End Module
16

```

LA STRUCTURE CONDITIONNELLE ALTERNATIVE

Définition : Elle permet d'exécuter un bloc d'instructions lorsque la condition posée est vérifiée, dans le cas contraire, elle exécute un autre bloc d'instructions

Syntaxe :

If condition Then

.....Bloc d'instructions1

Else

.....Bloc d'instructions2

End If

Exemple :

Dans l'exemple ci-dessous, nous reprenons le même exemple mais en utilisant cette fois, la structure alternative.

```

1 Module Module1
2
3     Sub Main()
4         Dim x, y As Integer
5         Console.WriteLine("Entrez un nombre entier Svp!")
6         x = Console.ReadLine()
7         If x < 0 Then
8             y = -1 * x
9             Console.WriteLine("La valeur absolue de " & x & " est " & y)
10        Else
11            y = x
12            Console.WriteLine("La valeur absolue de " & x & " est " & y)
13        End If
14        Console.Read()
15    End Sub
16
17 End Module
18

```

LES STRUCTURES IMBRIQUEES

Définition : Elles vont au-delà de la structure alternative. Elles sont généralement utilisées lorsque vous avez plus de deux contions à vérifier.

Syntaxe :

If condition1 Then

.....Bloc d'instructions _1

ElseIf condition2 Else

.....Bloc d'instructions _2

...

Else

.....Bloc d'instructions _n

End If

Exemple :

Dans l'exemple ci-dessous, le programme calcule la moyenne de deux notes et retourne :

- 'Bourse étrangère', lorsque la moyenne est supérieure ou égale à 18
- 'Bourse interne', lorsque la moyenne est comprise entre 15 (inclus) et 18 (non inclus)
- 'Admis', quand la moyenne est supérieure ou égale à 10 et strictement inférieure à 15
- Sinon 'Exclu'

```

1 Module Module1
2
3     Sub Main()
4         Dim n1, n2, moy As Double
5         Dim mess As String

```

```

6      Console.WriteLine("Entrez la note 1 Svp!")
7      n1 = Console.ReadLine()
8      Console.WriteLine("Entrez la note 2 Svp!")
9      n2 = Console.ReadLine()
10     moy = (n1 + n2) / 2
11     If moy >= 18 Then
12         mess = "Vous bénéficiez d'un bourse étrangère car vous avez obtenu une
moyenne de: "
13     ElseIf moy >= 15 Then
14         mess = "Vous bénéficiez d'un bourse interne car vous avez obtenu une
moyenne de: "
15     ElseIf moy >= 10 Then
16         mess = "Vous êtes admis car vous avez obtenu une moyenne de: "
17     Else
18         mess = "Vous êtes ajourné car vous avez obtenu une moyenne de: "
19     End If
20     Console.WriteLine(mess & moy)
21     Console.Read()
22 End Sub
23
24 End Module

```

LA STRUCTURE DE CHOIX

Définition : Elle a pratiquement le même fonctionnement que les structures imbriquées sauf que ses critères (ses conditions) portent sur l'égalité.

Syntaxe :

select Case variable

.....Case valeur_1

.....Instructions_1

.....Case valeur_2

.....Instructions_2

...

.....Case Else

.....Instructions_n

End Select

Exemple :

Dans l'exemple ci-dessous, le programme effectue un affichage en fonction du choix de l'utilisateur

```

1 Module Module1
2
3     Sub Main()
4         Dim choix As Integer
5         Console.WriteLine("1 - Choix N°1")
6         Console.WriteLine("2 - Choix N°2")
7         Console.WriteLine("3 - Choix N°3")
8         Console.WriteLine("4 - Choix N°4")
9         Console.WriteLine()
10        Console.WriteLine("Effectuez votre choix SVP!")
11        choix = Console.ReadLine()
12        Select Case choix
13            Case 1
14                Console.WriteLine("Exécution de l'instruction N°1")
15            Case 2

```

```
16         Console.WriteLine("Exécution de l'instruction N°2")
17     Case 3
18         Console.WriteLine("Exécution de l'instruction N°3")
19     Case 4
20         Console.WriteLine("Exécution de l'instruction N°4")
21     Case Else
22         Console.WriteLine("Désolé! erreur de saisie")
23     End Select
24     Console.Read()
25 End Sub
26
27 End Module
28
```

EVALUATION 1



Exercice 1

```
1 Module Module1
2
3     Sub Main()
4         Dim v1, v2 As Integer
5         v1 = 2
6         v2 = 5
7         If v2 Mod v1 = 2 Then
8             v2 = v2 + 3
9         Else
10            v2 = v2 - 2
11        End If
12        Console.Read()
13    End Sub
14
15 End Module
16
```

Que vaut v2 à la fin de ce programme ?

- ☐ 5
- ☐ 8
- ☐ 3

Exercice 2

```
1 Module Module1
2
3     Sub Main()
4         Dim v1, v2, v3 As Integer
5         v1 = 2
6         v2 = 5
7         If v2 Mod v1 = 1 Then
8             v2 = v2 + 3
9             v1 = v1 + 5
10        End If
11        If v2 Mod 2 = 0 Then
12            v1 = v1 - 1
13        Else
14            v1 = v1 - 3
15        End If
16        Console.Read()
17    End Sub
18 End Module
19
20
```

Que vaut *v1* à la fin du programme ?

- ☐ 7
- ☐ 6
- ☐ 4

LES STRUCTURES ITERATIVES



Définition

Les structures itératives encore appelées structures répétitives font également partie des structures de contrôle. Mais contrairement aux structures conditionnelles, les structures itératives ont la possibilité d'exécuter plusieurs fois le même bloc d'instructions tout en tenant compte de la condition posée.

Ainsi, nous en distinguons plusieurs types en Visual Basic.Net

LA STRUCTURE While

Définition

Elle correspond à la structure TANTQUE en algorithmme. Elle exécute le même bloc d'instructions tant que la condition posée est vérifiée.

Syntaxe :

While condition

.....Bloc d'instructions

End While

Exemple :

L'exemple défini ci-dessous contraint l'utilisateur à saisir un nombre pair. Dans ce programme, l'utilisateur saisi un nombre entier dans un premier temps, tant que ce nombre n'est pas pair (le reste de la division de ce nombre par 2 est différent de zéro), le programme lui demande de saisir une valeur correcte.

```

1 Module Module1
2
3     Sub Main()
4         Dim val As Integer
5         Console.WriteLine("Saisissez un nombre pair SVP!")
6         val = Console.ReadLine()
7         While val Mod 2 <> 0
8             Console.WriteLine("Désolé! Saisissez un nombre pair SVP!")
9             val = Console.ReadLine()
10        End While
11        Console.WriteLine("Vous avez saisi " & val & " qui est un nombre pair")
12        Console.Read()
13    End Sub
14
15 End Module
16

```

LA STRUCTURE Do While

Définition

Elle correspond à la structure FAIRE ..TANTQUE en algorithmique. Elle exécute le bloc d'instructions au moins une fois car ce bloc est d'abord exécuté avant que la condition ne soit vérifiée, ce qui fait qu'à la première exécution, même si la condition n'est pas vérifiée, le bloc sera quand même exécuté une fois.

Syntaxe :

Do

.....Bloc d'instructions

Loop While condition

Exemple :

Cet exemple reprend l'exercice précédent mais cette fois, avec la structure Do...While

```

1 Module Module1
2
3     Sub Main()
4         Dim val As Integer
5         Do
6             Console.WriteLine("Saisissez un nombre pair SVP!")
7             val = Console.ReadLine()
8             Loop While val Mod 2 <> 0
9             Console.WriteLine("Vous avez saisi " & val & " qui est un nombre pair")
10            Console.Read()
11        End Sub
12
13 End Module
14

```

LA STRUCTURE Do...UNTIL

Définition

Elle correspond à la structure REPETER ... JUSQUA en algorithmique. Elle exécute le bloc d'instructions tant que la condition posée n'est pas vraie, et ce, jusqu'à ce qu'elle soit vérifiée. Ici, le bloc d'instructions est également exécuté une fois au moins.

Syntaxe :

Do

.....Bloc d'instructions

Loop Until condition

Exemple :

Cet exemple reprend l'exercice précédent mais cette fois, avec la structure Do...Until

```

1 Module Module1
2
3     Sub Main()
4         Dim val As Integer
5         Do
6             Console.WriteLine("Saisissez un nombre pair SVP!")
7             val = Console.ReadLine()
8             Loop Until val Mod 2 = 0
9             Console.WriteLine("Vous avez saisi " & val & " qui est un nombre pair")
10            Console.Read()
11        End Sub
12
13 End Module
14

```

LA STRUCTURE For

Définition

Elle correspond à la structure POUR en algorithmique. Elle exécute également plusieurs fois le bloc d'instructions mais cette fois, en connaissant au départ le nombre d'itérations à effectuer.

Syntaxe :

For initialisation_variable To valeur_finale

.....Bloc d'instructions

Next

Exemple :

Cet exemple affiche les dix premiers nombres entiers positifs

```

1 Module Module1
2
3     Sub Main()
4         Dim val As Integer
5         For val = 1 To 10
6             Console.WriteLine(val)
7         Next
8         Console.Read()
9     End Sub
10
11 End Module
12

```



Remarque

Avec la structure For, il est possible d'évoluer par pas différent de 1. Pour le faire, il suffit de définir le mot clé *Step*.

Par Exemple, nous voulons afficher les nombres pairs compris entre 1 et 10, ceci donne :

```

1 Module Module1
2
3     Sub Main()
4         Dim val As Integer
5         For val = 2 To 10 Step 2
6             Console.WriteLine(val)
7         Next
8         Console.Read()
9     End Sub
10
11 End Module
12

```

EVALUATION 2

IV

Exercice 1

```
1 Module Module1
2
3     Sub Main()
4         Dim v1, v2 As Integer
5         Do
6             v1 = 9
7             v2 = 5
8             v1 = v1 + 1
9             If v1 < 10 Then
10                Console.WriteLine(v1)
11            Else
12                Console.WriteLine(v2)
13            End If
14        Loop While v1 < 10
15        Console.Read()
16    End Sub
17
18 End Module
```

Que fait ce programme ?

- ☐ Il affiche 9 et 10
- ☐ Il affiche 5 une seule fois
- ☐ Il affiche 5 deux fois

Exercice 2

```
1 Module Module1
2
3     Sub Main()
4         Dim v1 As Integer
5         Do
6             Console.WriteLine("Saisissez un nombre impair SVP!")
7             v1 = Console.ReadLine()
8             Loop ... v1 Mod 2 <> 0
9             Console.WriteLine("Vous avez saisi " & v1 & " qui est un nombre impair")
10            Console.Read()
11        End Sub
12
13 End Module
14
```

Ce programme contraint l'utilisateur à saisir un nombre impair, quel mot clé faut-il remplacer par les trois points de suspension ?

- ☐ Until
- ☐ While
- ☐ Next