Exercices sur les Expression Logiques

E. Thirion - 04/07/2015

Les corrigés des exercices suivants sont disponibles par téléchargement. Pour savoir comment y accéder cliquez ici.

D'autre part, ce document fait partie d'un ensemble de cours du même auteur (programmation procédurale, programmation objet, programmation web, bases de données) auxquels vous pouvez accéder en cliquant ici.

Exercice 1

Dans un programme Pascal, **a** et **b** sont déclarées de la manière suivante :

Var a, b : **Boolean ;**

Indiquez les valeurs des expressions logiques suivantes selon les valeurs de a et b :

a	b	a **And** (**Not** b)	b **And** (**Not** a)	(a **And** (**Not** b)) **Or** (b **And** (**Not** a))
False	False			
False	True			
True	False			
True	True			

Exercice 2

Dans un programmePascal, **m** et **n** sont déclarées de la manière suivante :

Var m, n : **Integer;**

Les deux instructions suivantes sont exécutées : n := 1; m := n+1 ;

Indiquez les valeurs des expressions logiques suivantes :

Expression	**Valeur**
n > m	
n <> m	
n >= m	
n+1 >= m	
n+2 > m+1	
2 * n = m	
n <= m	
n < m	
(n+1) < m	
n - m <= 0	

Exercice 3

Dans un programme Pascal, les variables **m** , **n** et **a** sont déclarées de la manière suivante :

Var m, n : **Integer ;** a **: Boolean ;**

Les trois instructions suivantes sont exécutées :

n := 1 ; m = 2 * (n+1) - 1; a := **True ;**

Indiquez les valeurs des expressions logiques suivantes :

Expression	Valeur
Not (n+2 > m)	
n+1 <= m	
2 * n > m	
(n+1 <=m) **Or** (2 * n > m)	
(2 * n > m) **And** a	
3 * n <> m	
(2 * n > m) **Or** (3 * n <> m)	
a **Or** (2 * n > m) **Or** (3 * n <> m)	
(n+1 <= m) **And** (a **Or** (2 * n > m) **Or** (3 * n <> m))	

Exercice 4

n et **m** sont des variables de type entier. **a** et **b** sont des variables booléennes. Indiquez les valeurs de ces variables après l'exécution des instructions indiquées.

Instruction	n	m	a	b
n := 1 ;				
m := 2 ;				
a := (n > m) ;				
b := (n <> m) ;				
a := a **Or** b ;				
n := n + 1 ;				
a := (n = m) ;				
b := **Not** b ;				
a := b **And** (Not b) ;				
n := n + 1 ;				
a := (n >= m) ;				
b := (n <= m) **Or** a ;				

Exercice 5: Machine à Café

Dans un programme simulant une machine à café on a déclaré trois variables globales de type entier: **N10M**, **N20M** et **N50M**. Ces trois variables représentent respectivement le nombre de pièces de 10cts, 20cts et 50cts contenues dans la machine.

Question 1

Ce programme contient une fonction booléenne **PeutRendrePiece** qui retourne la valeur **True** si et seulement si la machine peut rendre p10 pièce de 10 cts, p20 pièces de 20 cts et p50 pièces de 50 cts.

Voici son code partiel:

```
Function PeutRendrePiece(p10, p20 As Integer,p50: Integer) : Boolean;
begin
   PeutRendrePiece =
End;
```

Quelle expression doit figurer à droite du signe := ?

Question 2

Le programme contient six autres fonctions, nommée **PeutRendre10Cts, PeutRendre20Cts**, ... **PeutRendre60Cts** qui retournent la valeur **True** si et seulement si la machine à café peut respectivement rendre 10, 20, ... 60 cents.

Le code partiel de ces six fonctions est donné ci-dessous. On vous demande de les complèter, sachant qu'il manque simplement une expression à droite du signe **:=**.

Indication: dans ces expressions figurent l'opérateur logique **Or** et des appels de la fonction **PeutRendrePiece**.

```
Function PeutRendre10Cts() : Boolean ;
Begin
  PeutRendre10Cts :=
End;
```

```
Function PeutRendre20Cts() : Boolean ;

  PeutRendre20Cts :=
End;
```

```
Function PeutRendre30Cts() : Boolean ;

  PeutRendre30Cts :=
End;
```

```
Function PeutRendre40Cts() : Boolean ;
Begin
  PeutRendre40Cts :=


End;
```

```
Function PeutRendre50Cts(): Boolean;
Begin
  PeutRendre50Cts =



End Function
```

```
Function PeutRendre60Cts() : Boolean;

  PeutRendre60Cts :=




End Function
```