Cliquez ici pour voir le cours en entier

Les Expressions Logiques

I - Variable booléenne

Le type booléen est un type de variable, tout comme le type entier ou chaine de caractères.

Une variable de ce type sert à représenter une information qui peut être vraie ou fausse.

Par exemple dans un programme de gestion de marchandises, on pourrait avoir une variable **RuptureDeStock** qui nous permettrait de savoir si le stock de marchandises est épuisé.

De manière générale :

> Une **variable booléenne** est une variable qui peut prendre deux valeurs : **true** (vrai) et **false** (faux).

Il n'y a donc que deux littéraux de type booléen.

En Pascal, le type booléen se note **Boolean** et une variable booléenne se déclare de la manière suivante:

```
Var nom de la variable : Boolean ;
```

On pourrait par exemple écrire :

```
Var RuptureDeStock : Boolean;
RuptureDeStock := True;
```

La première instruction déclare une variable booléenne nommée **RuptureDeStock**. La deuxième lui affecte la valeur **True**.

II - Opérateurs logiques

Nous avons déjà vu quelques opérateurs: les opérateurs arithmétiques qui agissent sur des nombres ainsi que l'opérateur de concaténation qui agit sur des chaines de caractères.

De la même manière, il existe des opérateurs logiques. Ceux-ci permettent de calculer une valeur logique (donc vrai ou faux) à partir d'autres valeurs logiques.

Il existe principalement trois opérateurs logiques : la **négation** , la **conjonction** et la **disjonction**.

II-1 La négation

En logique, la **négation** permet de représenter le contraire d'une proposition.

Si quelque chose est vrai, alors son contraire est faux et réciproquement.

En Pascal, l'opérateur de négation se note **Not**.

La table suivante (que l'on appelle **table de vérité**) résume l'effet de l'opérateur de négation

X	**Not** X
True	False

False	True

Cliquez ici pour voir le cours en entier

Par exemple, avec deux variables booléennes **ArticleDisponible** et **RuptureDeStock**, on pourrait écrire :

```
ArticleDisponible := Not RuptureDeStock;
```

Si **RuptureDeStock** vaut **true** avant cette affectation, alors **ArticleDisponible** vaudra **false** après cette affectation.

Réciproquement, si **RuptureDeStock** vaut **False** avant cette affectation, alors **ArticleDisponible** vaudra **true** après cette affectation.

II-2 La conjonction

La **conjonction**, représentée en Pascal par l'opérateur **And**, permet d'exprimer le fait que deux choses sont vraies simultanément.

Voilà la table de vérité de l'opérateur **And** :

X	Y	X **And** Y
False	False	False
False	True	False
True	False	False
True	True	True

Prenons un exemple.

La variable booléenne **AmpouleFonctionne** représente le fait qu'une ampoule électrique fonctionne.

Une autre variable booléenne **InterrupteurOn** est vraie si et seulement si l'interrupteur est en position On.

La variable booléenne **LumiereAllumee** représente le fait que la lumière est allumée.

Après l'affectation suivante :

```
  LumiereAllumee := AmpouleFonctionne And InterrupteurOn;
```

la variable **LumiereAllumee** ne vaudra **True** que si les variables **AmpouleFonctionne** et **InterrupteurOn** valaient également **True** juste avant l'affectation.

II - 3 La disjonction

La **disjonction** de deux propositions est vraie si et seulement si au moins une de ces deux propositions est vraie.

Autrement dit, la disjonction de deux propositions n'est fausse que lorsque ces deux propositions sont fausses.

Par exemple, la proposition "Il est bête **ou** il ne comprend pas" ne sera fausse que si l'individu en question n'est pas bête et a très bien compris.

Attention ! le ou logique n'est pas un ou exclusif. La disjonction de deux propositions est vraie lorsque les deux propositions sont vraies. Par exemple, l'affirmation "Il pleut ou il y a du soleil" est vraie lorsqu'il pleut et qu'il y a du soleil en même temps.

L'opérateur de disjonction est représenté en Pascal par le mot clé **Or**. Cliquez ici pour voir le cours en entier

X	Y	X **Or** Y
False	False	False
False	True	True
True	False	True
True	True	True

Prenons un exemple.

La variable booléenne **AmpouleFouttue** représente le fait qu'une ampoule électrique ne fonctionne pas.

Une autre variable booléenne **InterrupteurOff** est vraie si et seulement si l'interrupteur est en position Off.

La variable booléenne **LumiereEteinte** représente le fait que la lumière est éteinte.

Après l'affectation suivante :

```
LumiereEteinte := AmpouleFouttue Or InterrupteurOff ;
```

la variable **LumiereEteinte** ne vaudra **True** que si la variable **AmpouleFouttue** et/ou la variable **InterrupteurOff** avaient la valeur **True** juste avant l'affectation.

III - Expressions logiques

Une **expression logique** est une expression de type booléen, c'est à dire une expression pouvant prendre la valeur vrai ou faux.

III-1 Les variables booléennes

Vous connaissez déjà un cas particulièrement simple d'expression logique : ce sont les noms de variables booléennes.

Par exemple, si la variable **LumiereEteinte** est déclarée comme une variable booléenne, le simple nom de variable **LumiereEteinte** peut être considéré comme une expression logique.

III-2 Utilisation des opérateurs logiques

On peut également construire des expressions logiques en combinant d'autres expressions logiques avec les opérateurs **Not**, **And**, **Or**.

Supposons par exemple que **x**, **y** et **z** soient des variables booléennes. Alors **Not x**, **x And y**, **x Or y** sont des expressions logiques. Tout comme **z Or (x And y)**, **(Not z And x) et (x Or y)**, ...

De manière générale:

- Si **E** est une expression logique alors **Not E** est une expression logique
- Si **E** et **F** sont des expressions logiques, alors **E And F** est une expression logique
- Si **E** et **F** sont des expressions logiques, alors **E Or F** est une expression logique
- Si **E** est une expression logique alors **(E)** est une expression logique

Cliquez ici pour voir le cours en entier

III-3 Utilisation des opérateurs de comparaison

Il existe encore un autre moyen de construire une expression logique : en utilisant des **opérateurs de comparaison**.

> Un **opérateur de comparaison** est un opérateur qui permet de comparer les valeurs de deux expressions de même type.

III-3-1 Les six opérateurs

Il existe six opérateurs de comparaison :

Symbole mathématiques	Signification
=	**Egal à**
≠	**Différent de**
<	**Strictement inférieur à**
>	**Strictement supérieur à**
≤	**Inférieur ou égal à**
≥	**Supérieur ou égal à**

La notation des opérateurs de comparaison varie selon les langages:

Symbole mathématiques	Pascal	Langage C
=	**=**	**==**
≠	**<>**	**!=**
<	**<**	**<**
>	**>**	**>**
≤	**<=**	**<=**
≥	**>=**	**>=**

III-3-2 Comparaison d'expressions numériques

L'utilisation la plus fréquente des opérateurs de comparaison est la comparaison d'expressions numériques.

Le plus simple est la comparaison directe de deux variables numériques. Exemple:

x	y	x = y	x <> y	x < y	x >= y	x > y	x <= y
1	1	True	False	False	True	False	True

Cliquez ici pour voir le cours en entier

1	2	False	True	True	False	False	True
2	1	False	True	False	True	True	False

Mais on peut bien sur comparer des expressions numériques quelconques. Par exemple, pour x=1 et y = 2 on aurait :

Expression logique	**Valeur**
(x+1) = y	True
(x+1)*(y+1) = 6	True
x+1 <= 2	True
x-1 < -y	False

III-3-3 Comparaison de chaines de caractères

En Pascal, les six opérateurs de comparaison peuvent être utilisés avec des chaines de caractères, en se basant sur l'ordre alphabétique. Exemple:

x	**y**	**x = y**	**x <> y**	**x < y**	**x >= y**	**x > y**	**x <= y**
'Trac'	'Trac'	True	False	False	True	False	True
'Trac'	'Truc'	False	True	True	False	False	True
'Truc'	'Trac'	False	True	False	True	True	False