EZ Language – Les opérateurs

Opérateurs logiques

Syntaxe EZ	Equivalent C++	Description
and	&&	a AND b vaut VRAI si a est vrai et b est vrai
or	||	a OR b vaut VRAI si a est vrai ou b est vrai ou les deux sont vrais en même temps
xor	^	a OR b vaut VRAI si a est vrai ou b est vrai, mais faux si les deux sont vrais en même temps
not	!	NOT a vaut VRAI si a est faux, mais vaut FAUX si a est vrai

Opérateurs arithmétiques

L'opérateur de division `/` retourne un réel sauf si les 2 opérandes sont des entiers et que leur division est exacte (c'est-à-dire a pour reste 0), auquel cas une valeur entière sera retournée.
Pour pouvoir récupérer la partie entière lors d'une division entre deux entiers dont le modulo n'est pas nul, il faudra faire un `cast` (voir partie sur les opérations de cast).
Les opérandes du modulo sont convertis en entiers (en supprimant la partie décimale) avant exécution.
Le résultat de l'opération `mod` a le même signe que le premier opérande, ainsi le résultat de `a mod b` aura le signe de `a`.

Syntaxe EZ	Equivalent C++	Nom	Description
-a	-a	Négation	Opposé de a
a + b	a + b	Addition	Somme de a et b
a - b	a - b	Soustraction	Différence de a et b
a * b	a * b	Multiplication	Produit de a et b
a / b	a / b	Division	Quotient de a et b
a mod b	a % b	Modulo	Reste de a divisé par b
a pow b	pow (a, b)	Puissance	Résultat de l'élévation de a à la puissance b

Opérateurs d'affectation

Syntaxe EZ	Equivalent C++	Description
=	=	L'opérande de gauche se voit attribuer la valeur de l'expression qui est à droite du signe =
+=	+=	L'opérande de gauche se voit additionner la valeur de l'expression qui est à droite du signe =
-=	-=	L'opérande de gauche se voit soustraire la valeur de l'expression qui est à droite du signe =
*=	*=	L'opérande de gauche se voit multiplier par la valeur de l'expression qui est à droite du signe =
/=	/=	L'opérande de gauche se voit diviser par la valeur de l'expression qui est à droite du signe =

Opérateurs de comparaison

Les deux opérandes comparés doivent être de même type. Si ce n'est pas le cas, cela produira une erreur de compilation.
Pour comparer deux opérandes de types différents, il faudra utiliser une opération de cast.

Syntaxe EZ	Equivalent C++	Nom	Description
a == b	a == b	Egal	VRAI si a est égal à b
a != b	a != b	Différent	VRAI si a est égal à b
a < b	a < b	Inférieur à	VRAI si a strictement inférieur à b
a > b	a > b	Supérieur à	VRAI si a strictement supérieur à b
a <= b	a <= b	Inférieur ou égal à	VRAI si a inférieur ou égal à b
a >= b	a >= b	Supérieur ou égal à	VRAI si a supérieur ou égal à b

Opérateurs d'incrémentation et de décrémentation

Syntaxe EZ	Equivalent C++	Description
a++	a++	Post-incrémentation
++a	++a	Pré-incrémentation
a--	a--	Post-décrémentation
--a	--a	Pré-décrémentation

Opérateurs de chaînes

Les opérateurs suivants seront définis pour les chaînes de caractères :

- L'opérateur d'affectation `=` qui affectera une nouvelle valeur à la chaîne de caractères, remplaçant son contenu actuel.
- L'opérateur d'accès `[]` qui retournera une référence sur le caractère à la position spécifiée. On pourra parcourir une chaîne de la même manière qu'un tableau.
- L'opérateur de concaténation `+` qui permettra de concaténer la chaîne en partie gauche et la chaîne en partie droite.
- L'opérateur `+=` qui ajoutera les caractères à droite de l'opérateur à la fin de la chaîne à gauche de l'opérateur.
- L'opérateur de comparaison `==` qui retournera `true` si la chaîne à gauche de l'opérateur est égale à la chaîne à droite de l'opérateur, `false` sinon.

Priorité des opérateurs

La priorité des opérateurs spécifie l'ordre dans lequel les valeurs doivent être analysées et les opérations effectuées. Ci-dessous, la table de priorité des opérateurs, le plus prioritaire placé en haut :

Niveau de priorité	Opérateur	Description
1	++ -- - . [] () not	Incrémentation suffixe/postfixe Décrémentation suffixe/postfixe Moins unaire Sélection membre par référence Accès dans un tableau Appel de fonction NON logique
2	* / mod pow	Multiplication Division Reste Puissance
3	+ -	Addition Soustraction
4	< <= > >=	Plus petit que Plus petit ou égal Plus grand que Plus grand ou égal
5	== !=	Egalité Différent de
6	and or xor	ET logique OU inclusif OU exclusif
7	= += -= *= /=	Affectation Affectation par somme Affectation par soustraction Affectation par produit Affectation par division

Lorsque les opérateurs ont une priorité égale, leur association décide la façon dont les opérateurs sont groupés. Par exemple, `-` est une association par la gauche, ainsi `1 - 2 - 3` est groupé comme ceci `(1 - 2) - 3` et sera évalué à `-4`. D'un autre côté, `=` est une association par la droite, ainsi, `a = b = c` est groupé comme ceci `a = (b =c)`.
Les opérateurs de priorité égale qui ne sont pas associatifs, ne peuvent pas être utilisés entre eux, par exemple, `1 < 2 > 1` est interdit. L'expression `1 <= 1 == 1` par contre, est autorisée, car l'opérateur `==` a une priorité inférieure que l'opérateur `<=`.

Opération de cast

Afin de réaliser un cast, on utilise la syntaxe suivante :

A is type_de_depart

B is type_de_destination

B = (type_de_destination) A