RESUMEN DE CONTENIDOS N° 7

OPERADORES DE C++

Toda expresión en un conjunto de operandos ligados por operadores.
Las expresiones se utilizan para efectuar cálculos, relaciones, asignaciones, etc

OPERADORES

Operadores Aritméticos

Como su nombre lo indica, los operadores aritméticos permiten efectuar cálculos aritméticos. La *jerarquía o precedencia* de estos operadores es idéntica a la utilizada en el álgebra de números. Esta jerarquía se puede alterar empleando paréntesis.

Operador	En tipos Enteros	En tipos Reales
+	símbolo + unario	símbolo + unario
-	símbolo – unario	símbolo – unario
+	suma	suma
-	resta	resta
*****	producto	producto
/	división entera	división en punto flotante
%	resto de la división entera	NA (no aplicable)
++	incremento	NA
--	decremento	NA

Ejemplos de operadores de Incremento y Decremento

```
Int c=0;
…
C++;
```

Es lo mismo que hacer:
c=0;
…
c=c+1

```
int  n=2;
cout<< n++  ;   /* Se visualiza un 2. C++ envía el contenido
de n a la salida a través de cout y luego incrementa en 1 a
n */

int  n=2;
cout << ++n ;   /* Se visualiza un 3. C++ incrementa en 1 a
n y luego muestra el nuevo valor de n */
```

Operadores Aritméticos

El operador **=** permite asignar el resultado de la expresión de la derecha a la variable de la izquierda.

X=130;

Debe observarse que este operador es *"asociativo por la derecha"*, por lo cual pueden efectuarse asignaciones múltiples o simultáneas.

a=b=c=30;

El compilador realiza la asociación del modo siguiente: **a = (b = (c = 30))** .

Operadores Aritméticos

Debe observarse que para C++ la proposición **c = 30** tiene doble sentido:

1) se trata de una asignación, y
2) se trata de una expresión que arroja el resultado **30**.

En C++, es válido realizar una asignación o una expresión en una acción de salida. Con lo cual, la siguiente acción es totalmente válida.

```
cout << (n=5) ;   /* asigna 5 a la variable n y visualiza 5 (resultado
                     de la expresión)*/
```

Operadores relativos de asignación

C++ dispone de operadores relativos, que permiten hacer más eficiente el código ejecutable resultante de la compilación

Operador	Asignación abreviada	Asignación no abreviada
+ =	x + = y	x = x + y
- =	x - = y	x = x - y
*** =**	x * = y	x = x * y
/ =	x / = y	x = x / y
%=	x % = y	x = x % y

Operadores Relacionales

C++ dispone de operadores relacionales, cuyo concepto es idéntico al que poseen en el álgebra de números: relacionar (comparar) operandos de tipos compatibles. Su simbología también es similar, a excepción de los operadores ***igual que*** y ***distinto que*** .

Los operadores relacionales en C++ son:

Operador	Significado	Ejemplo
==	Igual que	a == b
!=	Distinto que	a != b
<	Menor que	a < b
>	Mayor que	a > b
<=	Menor o igual que	a <= b
>=	Mayor o igual que	a >= b

Estos operadores permitirán plantear expresiones relacionales, las cuales, al ser evaluadas, arrojarán un valor de verdad: **verdadero** o **falso**. C++ dispone del valor **int cero (0)** para representar al valor **falso** y de un valor **int distinto de cero** para **verdadero**.

Valor de verdad	**Representación en C++**
Falso	Cero
Verdadero	Distinto de Cero

Los operadores relacionales se asocian de izquierda a derecha y tienen menor prioridad que los operadores aritméticos por lo tanto una expresión del tipo:

`a+b<10*c` equivale a **`(a+b)<(10*c)`**

Es posible asignar el resultado de una expresión relacional a una variable:

```
int m=(12+3<=10);      // asigna cero (falso) a la variable entera m
```

Operadores Lógicos

Los operadores lógicos de C++ son la **conjunción** o **and** (**&&**), la **disyunción** u **or** (**||**) y la **negación** o **not** (**!**). Conceptualmente funcionan de igual forma que en la lógica algebraica.

La conjunción (**&&**) arroja un resultado verdadero (**distinto de cero**) sólo si ambos operandos son verdaderos; la disyunción (**||**) sólo es falsa si ambos operandos son falsos; y la negación (**!**) es un operador unario que invierte el valor de verdad del operador afectado.

Operador	Significado	Ejemplo
!	Negación (no)	! a <= b
&&	Conjunción (y)	(a < b) && (n==100)
||	Disyunción (o)	(x == 10) || (a != c)

Evaluación en cortocircuito

C++ dispone la evaluación de una expresión lógica de izquierda a derecha.

Si el operando de la izquierda es suficiente para determinar el resultado de la proposición, no se evalúa el operando de la derecha.
Por ejemplo:

6 < 3 && z == 4

el operando **z == 4** no llegará a evaluarse pues **6 < 3** ya decidió el resultado **cero** (falso) de toda la proposición