Universidad Autónoma de Entre Ríos
Facultad de Ciencia y Tecnología
Sede: Oro Verde

FUNDAMENTOS DE PROGRAMACIÓN

RESUMEN DE CONTENIDOS N°7

Operadores

RESUMEN DE CONTENIDOS N° 7

Operadores

Introducción

A continuación, se estudiarán algunos de los numerosos operadores que posee C++ para conformar o plantear expresiones.

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

Se debe tener en cuenta que las operaciones algebraicas tienen asociatividad de izquierda a derecha, pero respetando la precedencia de los operadores aritméticos.

Ejemplos:

10 + 7 – 4 arroja como resultado **13**. Se resuelve: **(10 + 7) – 4** porque ante igual prioridad de operadores, se asocia de izquierda a derecha.

6 + 2 * 9 arroja como resultado **24**. Se resuelve: **6 + (2 * 9)** porque el operador producto tiene prioridad (precede) al operador de la suma.

21 / 4 arroja 5 (entero)

20.0 / 4.0 arroja **5.0** (punto flotante)

21 % 4 arroja **1** (entero)

++x incrementa en **1** el valor de **x**

x++ toma el valor de **x** y luego lo incrementa en **1**

--x decrementa en **1** el valor de **x**

x-- pos decrementa en **1** el valor de **x**

Nota: obsérvese en los 2 ejemplos siguientes la diferencia entre los operadores de incremento y decremento cuando preceden a una variable y cuando la suceden.

```
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 de asignación

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;
```

La acción precedente, permite asignar simultáneamente a las tres variables (a, b y c) el valor 30. El compilador realiza la asociación del modo siguiente: **a = (b = (c = 30))** . 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

Se utilizarán expresiones relacionales y lógicas en varias estructuras de control de C++, que permitirán plantear decisiones en el programa.

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.

Otros Operadores

C++ dispone de otros operadores que se describirán más adelante, conforme se avance en el desarrollo de nuevos temas. Por ejemplo: operadores de bits, operador de desplazamiento, operador de direcciones, operador condicional, operador de ámbito de resolución, operador coma, operador ( ), operador [ ].

Precedencia de Operadores en C++

La precedencia o prioridad de un operador determina el orden de aplicación de los operadores de una expresión. En la tabla siguiente se indican los grupos de operadores según el orden de prioridad.

Prioridad y categoría	Operador	Función o significado	Asociati-vidad
1. (Prioridad más alta)	( ) [ ] → :: .	Llamada a función Subíndice de arreglos Selector indirecto de miembro Selector de ámbito de resolución Selector directo de miembro	I-D
2. Unarios	! ~ + - ++ -- & * sizeof new delete	Negación lógica Complemento a uno (bits) Más (unario) Menos (unario) Incremento Decremento Dirección Indirección Tamaño del operando en bytes Alocación dinámica en memoria Eliminación dinámica	D-I
3.Acceso a miembros	* →	Lee o modifica el valor apuntado Accede a un miembro de un objeto apuntado	I-D
4.Multiplicativos	* / %	Multiplicación División entera o flotante Resto de la división entera (módulo)	I-D
5.Aditivos	+ -	Más binario Menos binario	I-D
6.Desplazamiento	>> <<	Desplazamiento a la derecha Desplazamiento a la Izquierda	I-D
7.Relacional	< <=	Menor que Menor o igual que	I-D

	> >=	Mayor que Mayor o igual que	
8. Igualdad	== !=	Igual que Distinto que	I-D
9.	&	And (manipulación de bits)	I-D
10.	^	Xor (manipulación de bits)	I-D
11.	|	Or (manipulación de bits)	I-D
12.	&&	Conjunción lógica and	I-D
13.	||	Disyunción lógica or	I-D
14.Condicional	?:	a ? x : y (significa: if a then b, else c)	D-I
15.Asignación	= *= /= %= += -= &= ^= | = <<= >>=		D-I
16. Coma (prioridad más baja)	,	Evaluación múltiple	

Si en una expresión aparecen operadores consecutivos de igual prioridad, debe considerarse la forma de asociarlos para resolver la expresión. Por ejemplo, en aritmética: ante la presencia de operadores de suma (+) y resta (-), los cuales tienen igual prioridad, C++ asocia de izquierda a derecha como corresponde al álgebra de números. La expresión **`a+b-c`** se resuelve: **`(a+b)-c`**

Deben tenerse en cuenta las reglas siguientes para el planteo de expresiones:

- Todos los operadores de un mismo grupo tienen igual prioridad y asociatividad.
- Si dos operadores se aplican a un operando, se aplica antes el de mayor prioridad
- Asociatividad I-D significa que primero se aplica el operador de la izquierda y luego el siguiente hacia la derecha. Asociatividad D-I significa hacer lo contrario.
- La precedencia o prioridad de operadores puede alterarse con los paréntesis, quienes tienen máxima prioridad.

Sentencia compuesta o Bloque

Una sentencia compuesta o un bloque es un conjunto de sentencias, encerradas entre llaves "**{ }**". Sintácticamente, **un bloque se considera como una única sentencia**. Los bloques pueden estar anidados hasta cualquier profundidad.