



Curso Básico de C

Clase N° 2

Operadores y Expresiones

Prof. Patricia Guerrero



Lenguaje C - Operadores

Operadores:

Son signos especiales que asocian variables y/o constantes para realizar una determinada operación.

- ❖ Operadores Aritméticos (+, -, *, /, %)
- ❖ Operadores de Asignación (=, +=, -=, *=, /=)
- ❖ Operadores Relacionales (==, <, >, <=, >=, !=)
- ❖ Operadores Lógicos (&&, ||, !)

```
espacio = espacio_inicial + 0.5 * aceleracion * tiempo;
```



Lenguaje C - Operadores

Operadores Aritméticos:

En C se utilizan cinco operadores aritméticos:

- ❖ Suma: +
- ❖ Resta: -
- ❖ Multiplicación: *
- ❖ División: /
- ❖ Resto: %

Estos operadores se consideran binarios

% (resto de la división entera)
sólo se aplica a variables enteras

Su sintaxis es:

<variable1> <operador> <variable2>



Lenguaje C - Operadores

Operadores Aritméticos:

- ❖ Incremento: ++
- ❖ Decremento: --
- ❖ Cambio de signo: -

Estos operadores se consideran unarios

Su sintaxis es:

<variable> <operador>
<operador> <variable>



Lenguaje C - Operadores

Operadores Aritméticos - Ejemplo:

`/* Uso de los operadores aritméticos */`

```
main( )  
{  
    int a = 1, b = 2, c = 3, r1, r2;  
    r1 = a + b;  
    r2 = c - a;  
    b++;  
}
```



Lenguaje C - Operadores

Operadores Aritméticos - Ejemplo:

```
int i = 2, k = 4, m, n;
```

```
m = i++; // después de ejecutarse esta  
         // sentencia m = 2 e i = 3
```

```
n = ++k; // después de ejecutarse esta  
         // sentencia n = 5 y k = 5
```



Lenguaje C - Operadores

Operadores de Asignación

Atribuyen a una variable -registran en la zona de memoria correspondiente a dicha variable- el resultado de una expresión o el valor de otra variable.

❖ Operador de asignación (=):

`nombre_variable = expresión;`

Se evalúa **expresión** y el resultado se copia en `nombre_variable`

`a + b = c;` // Esta asignación es incorrecta



Lenguaje C - Operadores

Otros Operadores de Asignación:

+=	Suma
-=	Resta
*=	Multiplicación
/=	División
%=	Resto (Módulo)

variable op= expresion; // op: +, -, *, /, %



variable = variable op expresión;



Lenguaje C - Operadores

Operadores de Asignación - Ejemplo:

```
/* Uso de los operadores de asignación */  
main( )  
{  
    int a = 1, b = 2, c = 3;  
    float x = 1.3;  
    a += 5;  
    c -= 1;  
    b *= 3;  
    x *= 3.0 * b - 1.0;  
}
```

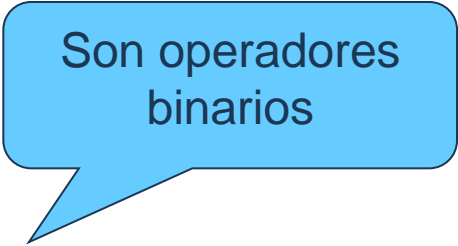


Lenguaje C - Operadores

Operadores Relacionales:

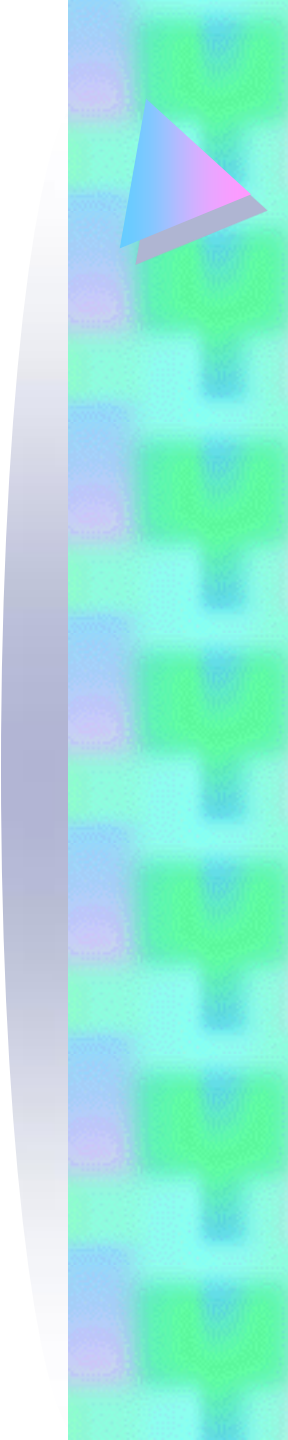
Permiten estudiar si se cumplen una o más condiciones.

>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
==	Igual que
!=	Distinto que



Son operadores binarios

Sintaxis:
expresion1 op expresion2



Lenguaje C - Operadores

Operadores Relacionales:

>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
==	Igual que
!=	Distinto que

Sintaxis:

expresion1 op expresion2

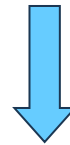
Se evalúan **expresion1** y **expresion2**, y se comparan los valores resultantes. *Si la condición representada por el operador relacional se cumple, el resultado es 1; en caso contrario será 0*



Lenguaje C - Operadores

Operadores Relacionales:

En el lenguaje natural, existen varias palabras o formas de indicar si se cumple o no una determinada condición:



(si/no), (yes/no), (on/off), (true/false)

En C el valor **0** representa la condición **false** y cualquier número distinto de **0** equivale a la condición **true** (por defecto el valor **1**).



Lenguaje C - Operadores

Operadores Relacionales - Ejemplo:

`(2 == 1) // resultado = 0, la condición no se cumple`

`(3 <= 3) // resultado = 1, la condición se cumple`

`(3 < 3) // resultado = 0, la condición no se cumple`

`(1 != 1) // resultado = 0, la condición no se cumple`



Lenguaje C - Operadores

Operadores Lógicos:

Permiten combinar los resultados de los operadores relacionales, comprobando si se cumplen varias condiciones de manera simultanea.

&&	AND
 	OR
!	NOT

&& y || son
operadores binarios.

! Es unario

expresion1 op expresion2



Lenguaje C - Operadores

Operadores Lógicos - Ejemplo:

`(2 == 1) || (-1 == -1)`

`(2 == 2) && (3 == -1)`

`((2 == 2) && (3 == 3)) || (4 == 0)`

`((6 == 6) || (8 == 0)) && ((5 == 5) && (3 == 2))`

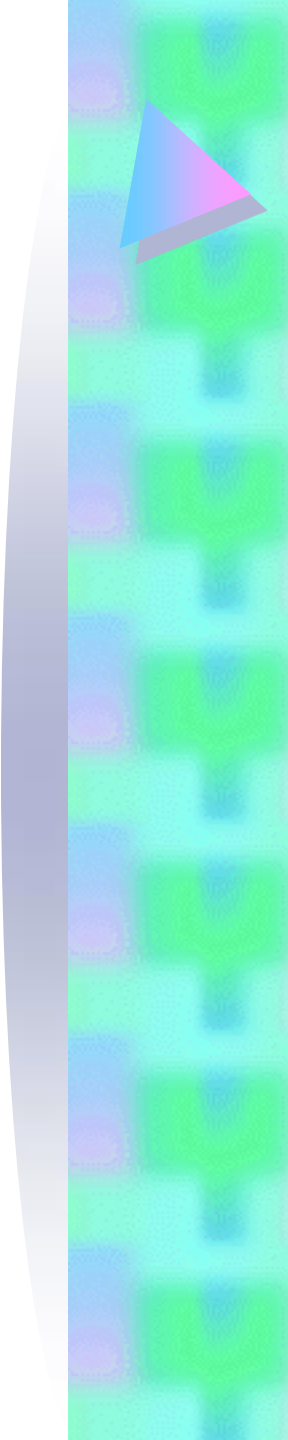
Lenguaje C - Operadores

Jerarquía de Operadores

Las operaciones con mayor precedencia se realizan antes que las de menor precedencia.

()	Mayor precedencia
++ --	
* / %	
+ -	Menor precedencia

Si en una operación encontramos signos del mismo nivel de precedencia, dicha operación se realiza de izquierda a derecha



Lenguaje C - Operadores

Jerarquía de Operadores

Precedencia	Asociatividad
() []	izda a dcha
++ -- ! +(unario) -(unario)	dcha a izda
* / %	izda a dcha
+ -	izda a dcha
< <= > >=	izda a dcha
== !=	izda a dcha
&&	izda a dcha
	izda a dcha
= += -= *= /=	dcha a izda



Lenguaje C - Expresiones

- ❖ Una **expresión** es una combinación de variables (constantes) y operadores.
- ❖ Una **expresión** es equivalente al **resultado** que proporciona el aplicar los operadores a los operandos.
- ❖ Una **expresión** puede estar formada por otras **expresiones más sencillas**, y puede contener **paréntesis de varios niveles** agrupando distintos términos.



Lenguaje C - Expresiones

Expresiones - Ejemplo:

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

```
x = (-b + sqrt((b * b) - (4 * a * c))) / (2 * a);
```

```
a = ((b > c) && (c > d)) || ((c == e) || (e == b));
```

```
(a - b * 2.0) && (c != d)
```

```
a = (b = (c = 1));      a = b = c = 1;
```



Lenguaje C - Evaluación

1. De la siguiente lista seleccione los identificadores válidos:

compañia

carlos

corazón

cot_2

luz*claro

float

ttttUUU

6.8

clave

2. Liste los datos primitivos de C.



Lenguaje C - Evaluación

3. Dado el siguiente bloque de código estudie la visibilidad de las variables:

```
float g = 6.5;
{
    int k = 1, m = 2; // g es visible?
    ...
    {
        float k = 6; // k = 1 entera es visible?
        ...          // r es visible?
    }
    int g = 7, r = 5; // g = 6.5 float es visible?
    // cuál variable k es visible?
}
```



Lenguaje C - Evaluación

4. Dados los siguiente tipos enumerados, indique el número entero asociado a cada valor:

```
enum letras {a = -2, b, c = 1, d, e = 5, m};
```

```
enum nombres {maria, pedro = -10, luis};
```

```
enum booleano {true, false};
```



Lenguaje C - Evaluación

5. Defina un programa que declare tres variables a, b y c (char, float e int), asigne a 'a' una constante, a 'b' el resultado de una división entera y a 'c' el resto de dicha división.
6. Agrupe la siguiente expresión según las reglas de precedencia:

`count * num + 88 / val - 198 % count`



Lenguaje C - Evaluación

7. Dado el siguiente bloque de código indique el resultado de la función printf:

```
int i, j, k;
```

```
i = 37;
```

```
j = i++;
```

```
k = --i;
```

```
++i;
```

```
printf("Los valores son i=%d j=%d k=%d", i, j, k);
```




Lenguaje C - Evaluación

8. Conociendo que `var = 4`, `max = 100` y `elem = -7`, evalúe el resultado de la siguiente expresión:

```
var > max || !(max == 100) && (0 <= elem)
```

9. ¿Cuáles son los valores de `count` y `max` luego de ejecutarse las siguientes sentencias?

```
count = 5;
```

```
max = 1;
```

```
count *= 10 * max++;
```



Lenguaje C - Evaluación

10. ¿Es cierta esta expresión?

`!(10 == 9)`

11. ¿Producen el mismo resultado estas dos expresiones?

`0 && 1 || 1`

`0 && (1 || 1)`