

Tema 3 Operadores y expresiones

Tema 3.- Operadores y expresiones

- 3.1. Operadores
 - a) Asignación
 - b) Aritméticos
 - c) Incrementación y decrementación
 - d) Operadores relacionales
 - e) Operadores lógicos
 - f) Operadores de manipulación de bits
- 3.2. Conversiones de tipos

3.1 Operadores

a) Operador de asignación. `variable=expresion`

Da valor a la variable, deben ser de tipos compatibles

```
main(){
    int a,b;
    char c;

    a=b;
    c='a';
    b=3.4;
}
```

Annotations:

- ¿Cuánto vale a? (points to `a=b;`)
- No sabemos porque b no tiene valor (points to `b=3.4;`)
- Da un warning (points to `b=3.4;`)
- Tipos incompatibles (points to `b=3.4;`)
- Da un warning (points to `b=3.4;`)

Importante!!
Hasta que no inicializo de alguna forma una variable, ésta no tiene valor

3.1 Operadores

Asignaciones múltiples

```
variable1=variable2= ... =expresion;
```

```
main(){
    int a,b,c;
    a=b=c=0; /*asigna valor 0 a todas la vbles*/
}
```

3.1 Operadores

Operadores de asignación adicionales

Símbolo	Uso	Descripción
<code>*=</code>	<code>a *= b</code>	multiplica a por b y lo asigna a a
<code>/=</code>	<code>a /= b</code>	divide a por b y lo asigna a a
<code>%=</code>	<code>a %= b</code>	asigna a a el resto de a/b
<code>+=</code>	<code>a += b</code>	suma a y b y lo asigna a a
<code>-=</code>	<code>a -= b</code>	resta b de a y lo asigna a a

3.1 Operadores

b) Operadores aritméticos `+ - * / %`

Operador	Enteros	Reales
<code>+</code>	suma	suma
<code>-</code>	resta	resta
<code>*</code>	producto	producto
<code>/</code>	división entera. Cociente	división
<code>%</code>	división entera. Resto	-

```
26/5
26/5.0
```

3.1 Operadores

Precedencia y asociatividad

	precedencia	asociatividad
Más ↑ prioridad	+, - (unarios)	por la izquierda
	*, /, %	por la izquierda
	+, -	por la izquierda
	=, *=, /=, %=, +=, -=	por la derecha

```
3 * 4 - 2  ⇔  (3 * 4) - 2
x - y + z  ⇔  (x - y) + z
a = b = c  ⇔  a = (b = c)
```

Importante!!
Paréntesis

3.1 Operadores

c) Incrementación y decrementación ++ --

Suman o restan 1 a su argumento

Incrementación	Decrementación
<code>++n</code>	<code>--n</code>
<code>n++</code>	<code>n--</code>
<code>n += 1</code>	<code>n -= 1</code>
<code>n = n + 1</code>	<code>n = n - 1</code>

Pueden ir sufixo y prefijo

3.1 Operadores

c) Incrementación y decrementación ++ --

Ojo!! No siempre se comporta igual como prefijo que como sufixo

```
#include<stdio.h>
main(){
  int a,b;

  a=10;
  b = ++a;
  printf("a[%d] b[%d]",a,b);
}
```

⇒ a[11] b[11]

3.1 Operadores

c) Incrementación y decrementación ++ --

Ojo!! No siempre se comporta igual como prefijo que como sufixo

```
#include<stdio.h>
main(){
  int a,b;

  a=10;
  b = a++;
  printf("a[%d] b[%d]",a,b);
}
```

⇒ a[11] b[10]

3.1 Operadores

CUIDADO!! pueden dar lugar a resultados no esperados

```
#include<stdio.h>
main(){
  int n=5,r;

  r = ++n * --n;
  printf("n=%d t=%d\n",n,r);

  printf("%d %d %d\n",++n,++n,++n);
}
```

⇒ n=5 t=25
8 7 6

3.1 Operadores

d) Operadores relacionales < > <= >= == !=

(menor, mayor, menor o igual, mayor o igual, igual y distinto)

- Trabajan con datos numéricos (también con char)
- Devuelven 0 para falso y cualquier valor distinto de 0 como verdadero

3.1 Operadores

e) Operadores lógicos && || !

Evalúan expresiones y devuelven 0 (falso) otro número distinto de 0 (verdadero)

AND			OR			NOT		
&&	1	0		1	0	!	1	0
1	1	0	1	1	1	0	0	1
0	0	0	0	1	1	0	1	0

3.1 Operadores

e) Operadores de manipulación de bits

Ejecutan operaciones lógicas sobre cada uno de los bits de los operandos.

Operador	Operación
&	AND lógica bit a bit
	OR lógica bit a bit
^	XOR (OR exclusive) lógica bit a bit
~	Complemento a 1
<<	Desplazamiento de bits a la izquierda
>>	Desplazamiento de bits a la derecha

3.1 Operadores

e) Operadores de manipulación de bits

0 & 0 == 0	0 0 == 0	0 ^ 0 == 0	~1 == 0 ~0 == 1
0 & 1 == 0	0 1 == 1	0 ^ 1 == 1	
1 & 0 == 0	1 0 == 1	1 ^ 0 == 1	
1 & 1 == 1	1 1 == 1	1 ^ 1 == 0	

Si se aplica el operador & a los números 9 y 14 se obtiene 8

9 equivale a 1 0 0 1
 &&&&
 14 equivale a 1 1 1 0
 1 0 0 0 que es 8

```
#include <stdio.h>
int a = 9, b=14,c;
main(){
    c = a & b;
    printf("c es %d\n",c);}
```

3.1 Operadores

e) Operadores de manipulación de bits

Operadores de desplazamiento de bits

valor >> númeroDeBits
 valor << númeroDeBits

Indica cuántos bits se desplazan a la dcha. o a la izq.

29 << 3 da como resultado 232

29 0 0 0 1 1 1 0 1
 1 1 1 0 1 0 0 0

```
#include <stdio.h>
main(){
    c = 29<<3;
    printf("c es %d\n",c);}
```

3.2 Conversiones de tipos

Conversiones implícitas.

Realizadas automáticamente por el compilador. C lo hace cuando:

- Se asigna aun valor de un tipo una variable de otro tipo
- Se combinan tipos mixtos en expresiones
- Se pasas argumentos a funciones

3.2 Conversiones de tipos

Conversiones implícitas.

```
#include<stdio.h>
main(){
    int i=6;
    double x=4.0;
    x = x+i;
    x = i/5;
    x = 4.0;
    x = x/5;
}
```

Convierte i a **double** antes de la suma

Hace la división entera y convierte el resultado (1) a **double**

Convierte 5 a **double** hace la división real (0.8) y lo asigna a x

3.2 Conversiones de tipos



Conversiones explícitas o *cast*.

`(tipoDato)valor`

Fuerza la conversión de **valor**
a **tipoDato**

```
int i=9;  
(float) i;  
i = (int)25.3;
```

Convierte el valor de **i** a **float**

Convierte **25.3** a **int** (25) y lo
asigna a **i**