Operadores lógicos

Nievas Martin

4 de junio de 2019

Operadores de asignación

Operador	Ejemplo	Equivalencia
=	`x = 3`	

Operadores de asignación

Operador	Ejemplo	Equivalencia
$=$	`x = 3`	
$+=$	`prom += 3`	`prom = prom + 3`

Operadores de asignación

Operador	Ejemplo	Equivalencia
$=$	`x = 3`	
$+=$	`prom += 3`	`prom = prom + 3`
`-=`	`prom -= 3`	`prom = prom - 3`

Operadores de asignación

Operador	Ejemplo	Equivalencia
=	`x = 3`	
+=	`prom += 3`	`prom = prom + 3`
-=	`prom -= 3`	`prom = prom - 3`
*=	`prod *= 2`	`prod = prod * 2`

Operadores de asignación

Operador	Ejemplo	Equivalencia
=	`x = 3`	
+=	`prom += 3`	`prom = prom + 3`
-=	`prom -= 3`	`prom = prom - 3`
*=	`prod *= 2`	`prod = prod * 2`
/=	`prom /= 10`	`prom = prom / 10`

Operadores de asignación

Operador	Ejemplo	Equivalencia
=	`x = 3`	
+=	`prom += 3`	`prom = prom + 3`
-=	`prom -= 3`	`prom = prom - 3`
*=	`prod *= 2`	`prod = prod * 2`
/=	`prom /= 10`	`prom = prom / 10`
%=	`mod%= 10`	`mod = mod% 10`

Operadores unarios

Operador	Ejemplo	Explicación
`++`	`i++`	`Utiliza e incrementa el valor de i`

Operadores unarios

Operador	Ejemplo	Explicación
`++`	`i++`	`Utiliza e incrementa el valor de i`
`--`	`i--`	`Utiliza y decrementa el valor de i`

Operadores unarios

Operador	Ejemplo	Explicación
`++`	`i++`	`Utiliza e incrementa el valor de i`
`--`	`i--`	`Utiliza y decrementa el valor de i`
`++`	`++i`	`Incrementa y luego utiliza el valor de i`

Operadores unarios

Operador	Ejemplo	Explicación
`++`	`i++`	`Utiliza e incrementa el valor de i`
`--`	`i--`	`Utiliza y decrementa el valor de i`
`++`	`++i`	`Incrementa y luego utiliza el valor de i`
`--`	`--i`	`Decrementa y luego utiliza el valor de i`

Operadores unarios

```c
#include <stdio.h>

int main(void)
{
  int i = 0;

  printf("i: %d\n", i + 1);

  if (i == 0)
  {
    printf("i vale cero\n");
  }

  return 0;
}
```

Operadores unarios

```c
#include <stdio.h>

int main(void)
{
  int i = 0;

  printf("i: %d\n", i + 1);

  if (i == 0)
  {
    printf("i vale cero\n");
  }

  return 0;
}
```

```
i: 1
i vale cero
```

Operadores unarios

```c
#include <stdio.h>

int main(void)
{
  int i = 0;

  printf("i: %d\n", i++);

  if (i == 0)
  {
    printf("i vale cero\n");
  }

  return 0;
}
```

Operadores unarios

```c
#include <stdio.h>

int main(void)
{
  int i = 0;

  printf("i: %d\n", i++);

  if (i == 0)
  {
    printf("i vale cero\n");
  }

  return 0;
}
```

```
i: 0
```

Operadores unarios

Operador	Ejemplo	Equivalencia
=	`x = 3`	

Operadores unarios

Operador	Ejemplo	Equivalencia
$=$	`x = 3`	
$+=$	`prom += 3`	`prom = prom + 3`

Operadores unarios

Operador	Ejemplo	Equivalencia
=	`x = 3`	
+=	`prom += 3`	`prom = prom + 3`
-=	`prom -= 3`	`prom = prom - 3`

Operadores unarios

Operador	Ejemplo	Equivalencia
=	`x = 3`	
+=	`prom += 3`	`prom = prom + 3`
-=	`prom -= 3`	`prom = prom - 3`
*=	`prod *= 2`	`prod = prod * 2`

Operadores unarios

Operador	Ejemplo	Equivalencia
=	`x = 3`	
+=	`prom += 3`	`prom = prom + 3`
-=	`prom -= 3`	`prom = prom - 3`
*=	`prod *= 2`	`prod = prod * 2`
/=	`prom /= 10`	`prom = prom / 10`

Operadores unarios

Operador	Ejemplo	Equivalencia
=	`x = 3`	
+=	`prom += 3`	`prom = prom + 3`
-=	`prom -= 3`	`prom = prom - 3`
*=	`prod *= 2`	`prod = prod * 2`
/=	`prom /= 10`	`prom = prom / 10`
%=	`mod%= 10`	`mod = mod% 10`

Operadores lógicos

Operador	Explicación
!	`Negación`

Operadores lógicos

Operador	Explicación
!	`Negación`
&&	`AND lógico, ambas condiciones deben ser verdaderas`

Operadores lógicos

Operador	Explicación
!	`Negación`
&&	`AND lógico, ambas condiciones deben ser verdaderas`
||	`OR lógico, al menos una condición debe cumplirse`

Operadores lógicos

```c
#include <stdio.h>

int main(void)
{
  int num_1 = 10;
  int num_2 = 20;

  if ( num_1 >  5 && num_2 < 100)
  {
    printf("Se cumple la condición\n");
  }

  return 0;
}
```

Operadores lógicos

```c
#include <stdio.h>

int main(void)
{
  int num_1 = 10;
  int num_2 = 20;

  if ( num_1 >  5 && num_2 < 100)
  {
    printf("Se cumple la condición\n");
  }

  return 0;
}
```

```
Se cumple la condición
```

Operadores lógicos

```c
#include <stdio.h>

int main(void)
{
  int num_1 = 10;
  int num_2 = 20;

  if ( num_1 >  5 || num_2 < 100)
  {
    printf("Se cumple la condición\n");
  }

  return 0;
}
```

Operadores lógicos

```c
#include <stdio.h>

int main(void)
{
  int num_1 = 10;
  int num_2 = 20;

  if ( num_1 >  5 || num_2 < 100)
  {
    printf("Se cumple la condición\n");
  }

  return 0;
}
```

```
Se cumple la condición
```

Operadores lógicos

```c
#include <stdio.h>

int main(void)
{
  int num_1 = 10;
  int num_2 = 200;

  if ( num_1 >  5 || num_2 < 100)
  {
    printf("Se cumple la condición\n");
  }

  return 0;
}
```

Operadores lógicos

```c
#include <stdio.h>

int main(void)
{
  int num_1 = 10;
  int num_2 = 200;

  if ( num_1 >  5 || num_2 < 100)
  {
    printf("Se cumple la condición\n");
  }

  return 0;
}
```

```
Se cumple la condición
```

Operadores lógicos

```c
#include <stdio.h>

int main(void)
{
  int num_1 = 10;
  int num_2 = 20;

  if ( num_2 <  100 || num_1 > 5)
  {
    printf("Se cumple la condición\n");
  }

  return 0;
}
```

Operadores lógicos

```c
#include <stdio.h>

int main(void)
{
  int num_1 = 10;
  int num_2 = 20;

  if ( num_2 <  100 || num_1 > 5)
  {
    printf("Se cumple la condición\n");
  }

  return 0;
}
```

```
Se cumple la condición
```

Operadores lógicos

```c
#include <stdio.h>

int main(void)
{
  int num_1 = 10;
  int num_2 = 5;

  if (num_1 && num_2)
  {
    printf("Se cumple la condición\n");
  }
  else
  {
    printf("No se cumple la condición\n");
  }

  return 0;
}
```

Operadores lógicos

```c
#include <stdio.h>

int main(void)
{
  int num_1 = 10;
  int num_2 = 5;

  if (num_1 && num_2)
  {
    printf("Se cumple la condición\n");
  }
  else
  {
    printf("No se cumple la condición\n");
  }

  return 0;
}
```

```
Se cumple la condición
```

Operadores lógicos

```
expresion1 && expresion2
```

```
expresion1 || expresion2
```

Operadores lógicos

expresion1	expresion2	expresion1 && expresion2

Operadores lógicos

`expresion1`	`expresion2`	`expresion1 && expresion2`
0	0	0

Operadores lógicos

expresion1	expresion2	expresion1 && expresion2
0	0	0
0	distinto de 0	0

Operadores lógicos

expresion1	expresion2	expresion1 && expresion2
0	0	0
0	distinto de 0	0
distinto de 0	0	0

Operadores lógicos

expresion1	expresion2	expresion1 && expresion2
0	0	0
0	distinto de 0	0
distinto de 0	0	0
distinto de 0	distinto de 0	1

Operadores lógicos

expresion1	expresion2	expresion1 && expresion2
0	0	0
0	distinto de 0	0
distinto de 0	0	0
distinto de 0	distinto de 0	1

expresion1	expresion2	expresion1 || expresion2

Operadores lógicos

expresion1	expresion2	expresion1 && expresion2
0	0	0
0	distinto de 0	0
distinto de 0	0	0
distinto de 0	distinto de 0	1

expresion1	expresion2	expresion1 || expresion2
0	0	0

Operadores lógicos

expresion1	expresion2	expresion1 && expresion2
0	0	0
0	distinto de 0	0
distinto de 0	0	0
distinto de 0	distinto de 0	1

expresion1	expresion2	expresion1 || expresion2
0	0	0
0	distinto de 0	1

Operadores lógicos

expresion1	expresion2	expresion1 && expresion2
0	0	0
0	distinto de 0	0
distinto de 0	0	0
distinto de 0	distinto de 0	1

expresion1	expresion2	expresion1 || expresion2
0	0	0
0	distinto de 0	1
distinto de 0	0	1

Operadores lógicos

expresion1	expresion2	expresion1 && expresion2
0	0	0
0	distinto de 0	0
distinto de 0	0	0
distinto de 0	distinto de 0	1

expresion1	expresion2	expresion1 || expresion2
0	0	0
0	distinto de 0	1
distinto de 0	0	1
distinto de 0	distinto de 0	1

Precedencia de operadores

Operadores						Asociatividad
`++`	`--`	`+`	`-`	`!`	`(tipo)`	De izquierda a derecha

Precedencia de operadores

Operadores						Asociatividad
`++`	`--`	`+`	`-`	`!`	`(tipo)`	De izquierda a derecha
`*`	`/`	`%`				Izquierda a derecha

Precedencia de operadores

Operadores						Asociatividad
`++`	`--`	`+`	`-`	`!`	`(tipo)`	De izquierda a derecha
`*`	`/`	`%`				Izquierda a derecha
`<`	`<=`	`>`	`>=`			Izquierda a derecha

Precedencia de operadores

Operadores						Asociatividad
`++`	`--`	`+`	`-`	`!`	`(tipo)`	De izquierda a derecha
`*`	`/`	`%`				Izquierda a derecha
`<`	`<=`	`>`	`>=`			Izquierda a derecha
`==`	`!=`					Izquierda a derecha

Precedencia de operadores

Operadores						Asociatividad
`++`	`--`	`+`	`-`	`!`	`(tipo)`	De izquierda a derecha
`*`	`/`	`%`				Izquierda a derecha
`<`	`<=`	`>`	`>=`			Izquierda a derecha
`==`	`!=`					Izquierda a derecha
`&&`	`||`					Izquierda a derecha

Precedencia de operadores

Operadores						Asociatividad
`++`	`--`	`+`	`-`	`!`	`(tipo)`	De izquierda a derecha
`*`	`/`	`%`				Izquierda a derecha
`<`	`<=`	`>`	`>=`			Izquierda a derecha
`==`	`!=`					Izquierda a derecha
`&&`	`||`					Izquierda a derecha
`=`	`+=`	`-=`	`*=`	`/=`	`%=`	Derecha a Izquierda

Precedencia de operadores

Operadores						Asociatividad
`++`	`--`	`+`	`-`	`!`	`(tipo)`	De izquierda a derecha
`*`	`/`	`%`				Izquierda a derecha
`<`	`<=`	`>`	`>=`			Izquierda a derecha
`==`	`!=`					Izquierda a derecha
`&&`	`||`					Izquierda a derecha
`=`	`+=`	`-=`	`*=`	`/=`	`%=`	Derecha a Izquierda
`,`						Izquierda a Derecha

```c
#include <stdio.h>

int main(void)
{
  int a = 0, b = 0, c = 0;

  a = b = c = 7;

  printf("a=%d b=%d c=%d\n", a, b, c);

  return 0;
}
```

```c
#include <stdio.h>

int main(void)
{
  int a = 0, b = 0, c = 0;

  a = b = c = 7;

  printf("a=%d b=%d c=%d\n", a, b, c);

  return 0;
}
```

```
a=7 b=7 c=7
```

```c
#include <stdio.h>

int main(void)
{
  int a = 0, b = 0, c = 0;

  a = b = c == 7;

  printf("a=%d b=%d c=%d\n", a, b, c);

  return 0;
}
```

```c
#include <stdio.h>

int main(void)
{
  int a = 0, b = 0, c = 0;

  a = b = c == 7;

  printf("a=%d b=%d c=%d\n", a, b, c);

  return 0;
}
```

```
a=0 b=0 c=0
```

```c
#include <stdio.h>

int main(void)
{
  /** Estado: */
  /**   0 -> Apagado */
  /**   1 -> Encendido */
  int estado = 0;

  if (estado = 0 )
  {
    printf("Está apagado!\n");
  }

  if (estado = 1)
  {
    printf("Está encendido\n");
  }


  return 0;
}
```

```c
#include <stdio.h>

int main(void)
{
  /** Estado: */
  /**   0 -> Apagado */
  /**   1 -> Encendido */
  int estado = 0;

  if (estado = 0 )
  {
    printf("Está apagado!\n");
  }

  if (estado = 1)
  {
    printf("Está encendido\n");
  }


  return 0;
}
```

```
Está encendido
```

```c
#include <stdio.h>

int main(void)
{
  /** Estado: */
  /**   0 -> Apagado */
  /**   1 -> Encendido */
  int estado = 0;

  if (estado == 0 )
  {
    printf("Está apagado!\n");
  }

  if (estado == 1)
  {
    printf("Está encendido\n");
  }


  return 0;
}
```

```c
#include <stdio.h>

int main(void)
{
  /** Estado: */
  /**   0 -> Apagado */
  /**   1 -> Encendido */
  int estado = 0;

  if (estado == 0 )
  {
    printf("Está apagado!\n");
  }

  if (estado == 1)
  {
    printf("Está encendido\n");
  }


  return 0;
}
```

```
Está apagado!
```

mnievas@frc.utn.edu.ar

Consultas
Martes
Edificio Salcedo 10:00 - 12:00