Especialización en Back End III

Operadores

En todo lenguaje de programación necesitamos de operadores para poder resolver problemas de programación. Repasemos qué es un operador, sus tipos y funcionalidades.

¿Qué es un operador?

Un operador es un elemento del programa que se aplica a uno o varios operandos en una expresión o instrucción. Los operadores, junto con los operandos, forman una expresión que es una fórmula que define el cálculo de un valor.

¿Qué tipos de operadores existen?

Existen diferentes tipos de operadores. Go posee los siguientes operadores:

- Aritméticos
- Relacionales
- Lógicos
- De asignación
- De dirección

Veamos cada uno de ellos en detalle.

Operadores aritméticos

Estos operadores nos van a permitir llevar a cabo operaciones básicas sobre las variables y constantes que declaremos. Veamos algunos de estos operadores en la siguiente tabla. La columna “Ejemplo” está calculada suponiendo que X:=10 e Y:=5.

Operador	Descripción	Ejemplo
+	Suma los operadores de los extremos.	X + Y resulta 15
–	Sustrae el operando de la derecha al operando de la izquierda.	X – Y resulta 5
*	Multiplica los operadores de los extremos.	X * Y resulta 50
/	División del número de la izquierda (numerador) entre el número de la derecha (denominador).	X / Y resulta 2
%	Operador modular o de residuo de división entera.	X / Y resulta 0
++	Operador de incremento. Incrementa en 1 el operador de la izquierda. No permite el decremento prefijo (predecremento).	X++ resulta 11
--	Operador de decremento. Decrementa en 1 el operador de la izquierda, es decir, no permite el decremento prefijo.	X-- resulta 9

Operadores relacionales

Estos operadores son aquellos que nos retornan un valor de verdad. Veamos algunos ejemplos en la siguiente tabla. La columna “Ejemplo” está calculada suponiendo que X:=1 e Y:=2.

Operador	Descripción	Ejemplo
==	Retorna verdadero cuando ambos operandos son iguales. Devuelve falso en cualquier otro caso.	X == Y retorna falso
!=	Retorna verdadero solamente cuando los operandos son distintos.	X != Y retorna verdadero
>	Retorna verdadero cuando el operando de la izquierda es mayor que el de la derecha.	X > Y retorna falso
<	Retorna verdadero cuando el operando de la izquierda es menor que el de la derecha.	X < Y retorna verdadero
>=	Retorna verdadero cuando el operador de la izquierda es mayor o igual al de la derecha.	X >= Y retorna falso
<=	Retorna verdadero cuando el operador de la izquierda es menor o igual a el de la derecha	X <= Y retorna verdadero

Operadores lógicos

Operador	Descripción	Ejemplo
&&	Operador lógico de conjunción (*AND*). Compara dos valores *bool* o expresiones relacionales.	A && B resulta falso X > 0 && Y < 6 resulta verdadero
||	Operador lógico de disyunción (*OR*). Compara dos valores *bool* o expresiones relacionales.	A || B resulta verdadero X < 0 || Y > 6 resulta falso
!	Operador de negación. Invierte (niega) el valor *bool* del operando.	!A resulta falso !B resulta verdadero

Operadores de asignación

Estos operadores nos permiten modificar el valor de nuestras variables durante la ejecución del programa.

Operador	Descripción	Ejemplo
=	Operador de asignación simple.	X = Y + Z asigna a X la suma de Y y Z (ni Y ni Z se modifican)
+=	Operador de suma y asignación.	X += Y asigna a X el valor de X + Y
-=	Operador de resta y asignación.	X -= Y asigna a X el valor de X – Y
*=	Operador de multiplicación y asignación.	X *= Y asigna a X el valor de X * Y
/=	Operador de división y asignación.	X /= Y asigna a X el valor de X /Y
%=	Operador de módulo y asignación.	X %= Y asigna a X el valor de X % Y

Operadores de dirección

Operador	Descripción	Ejemplo
&	Regresa la dirección en memoria del operando.	&X regresa la dirección en memoria de X
*	Apuntador a una variable.	*P apunta a una variable

Precedencia de operadores

La precedencia de un operador indica que tan "estrechamente" se unen dos expresiones juntas. Por ejemplo, en la expresión **1 + 5 * 3**, la respuesta es **16** y no **18** porque el operador de multiplicación ("*****") tiene una precedencia mayor que el operador de adición ("**+**"). Los paréntesis pueden ser usados para forzar la precedencia, si es necesario. Por ejemplo: **(1 + 5) * 3**, se evalúa como **18**.

¿Qué aprendimos?

Para finalizar, podemos destacar el importante uso de los operadores en la programación. Aprendimos qué es un operador y sus distintos tipos en Go.