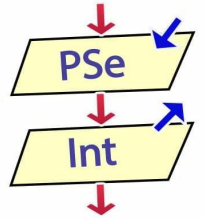
Agencia de Aprendizaje a lo largo de la vida

CODO A CODO INICIAL
Clase 8

Algoritmia 8

BA

Estructuras de control IV
Operadores lógicos



Les damos la bienvenida

Vamos a comenzar a grabar la clase

Clase 07

Algoritmia 7 - Estructuras condicionales III

- Uso de la estructura Segun... en lugar de condicionales múltiples.

Clase 08

Algoritmia 8 - Estructuras condicionales IV

- Operadores lógicos: AND (y), OR (o), NOT (no).
- Tablas de verdad.
- Uso de condicionales junto a operadores lógicos.
- Precedencia de operadores.

Clase 09

Algoritmia 9 - Estructuras Repetitivas I

- Estructuras repetitivas.
- Uso de "Repetir...Hasta Que"
- Condicionales en las estructuras repetitivas.

Operadores lógicos

En programación, los **operadores lógicos** “AND”, “OR” y “NOT” son herramientas para evaluar y combinar condiciones. Permiten construir expresiones lógicas a partir de valores booleanos.

“**AND**” devuelve verdadero solo si ambas condiciones son verdaderas, “**OR**” es verdadero si al menos una de las condiciones es verdadera, y “**NOT**” invierte el valor de una condición.

Estos operadores proporcionan la capacidad de crear algoritmos flexibles y adaptativos en la programación.

Operadores lógicos en PSeINT

Antes de empezar, necesitamos conocer una particularidad de **PSeInt**. El editor de código de PSeInt modifica la imágen en pantalla de dos operadores lógicos. Estos operadores se escriben en PSeInt como **&** (AND), **|** (OR) y **!** (NOT). Pero, al escribirlos, PSeInt transforma automáticamente el **&** y el **|** como se ve en la siguiente tabla:

Operador	**AND** (Y)	**OR** (O)	**NOT** (NO)
Cómo se escribe:	**&**	**|**	**!**
Cómo se ve en pantalla:	∧	∨	**!**

Tablas de verdad

Las **tablas de verdad** de un **operador lógico** en programación son representaciones sistemáticas de **todas las combinaciones posibles** de valores de entrada y sus correspondientes resultados de salida.

Al analizar las tablas de verdad, los programadores pueden comprender y prever el comportamiento de las expresiones lógicas en distintas situaciones, facilitando la toma de decisiones y el diseño de algoritmos con lógica booleana.

En un momento analizaremos la tabla de verdad de cada operador lógico de PSeInt.

Operador NOT (!) | Tabla de verdad

La tabla de verdad de este operador es es sencilla, ya que únicamente invierte el valor lógico.
En la columna de la izquierda, tenemos las entradas. En la de la derecha, las salidas, es decir, el resultado lógico de aplicar el operador a la entrada:

Entrada	!Entrada
Verdadero	Falso
Falso	Verdadero

Operador NOT (!)

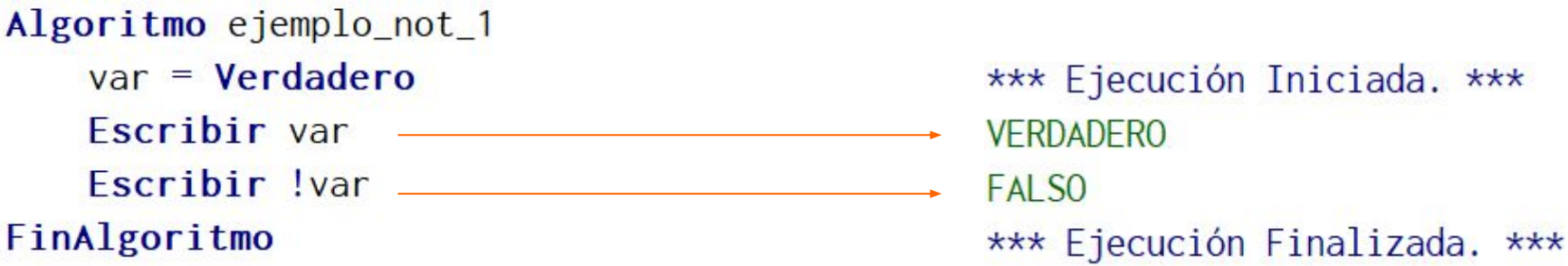
El operador **NOT** (NO, o negación) **invierte el estado lógico** de aquello a lo que se lo aplica. Si una expresión es Verdadero, será es Falso; si es Falso, será Verdadero. Veamos un ejemplo:

```
Algoritmo operadorNot

    var = Verdadero
    Escribir !var

FinAlgoritmo
```

```
*** Ejecución Iniciada. ***
FALSO
*** Ejecución Finalizada. ***
```

Operador NOT (!) | Ejemplo

En este ejemplo, vemos cómo **NOT** invierte el estado lógico en la salida de la variable por consola. La primera salida (*Verdadero*) corresponde al valor almacenado en la variable. La segunda (*Falso*) corresponde a la negación de ese valor mediante el operador NOT para su salida por consola.



Operador NOT (!) | Ejemplo

En el algoritmo de la derecha usamos el operador NOT para evaluar si la clave es distinta de 7.

La expresión **Si !clave == 7** es lo mismo que decir "*Si clave NO es igual a 7*", lo que también podría expresarse como **clave != 7**.

Siendo que el valor de la variable *clave* es 5, la salida por pantalla mostrará el mensaje **Acceso denegado**.

```
Algoritmo operadorNotCondicional

    clave = 5
    Si !clave == 7 Entonces
        Escribir "Acceso denegado"
    SiNo
        Escribir "Acceso concedido"
    FinSi

FinAlgoritmo
```

Operador NOT (!) | Escribir

La negación de un valor en **Escribir** no cambia el valor almacenado en la variable. Si es necesario cambiarlo, es necesario **reasignarle** a la variable su propio valor negado, guardarle lo inverso de lo que tenía antes;

```
Algoritmo ejemplo_not_2
    var = Verdadero
    Escribir var
    Escribir !var
    Escribir var // Sigue siendo Verdadero: no cambió

    var = !var // Le reasigno su propio valor negado
    Escribir var // Y ahora sí pasó a ser Falso
FinAlgoritmo
```

```
*** Ejecución Iniciada. ***
VERDADERO
FALSO
VERDADERO
FALSO
*** Ejecución Finalizada. ***
```

Operador AND (&) | Tabla de verdad

El operador lógico **AND** es utilizado para combinar dos condiciones y produce un resultado verdadero (*true*) **solo si ambas condiciones son verdaderas**.

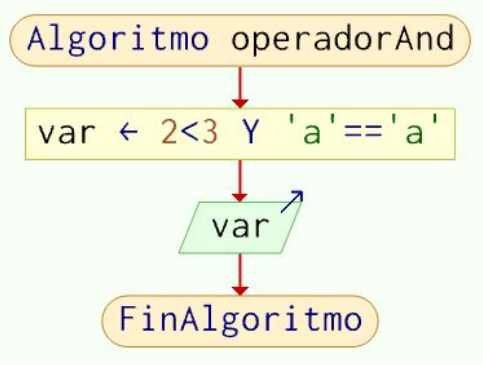
Entradas		Salida
A	**B**	**A & B**
Verdadero	**Verdadero**	**Verdadero**
Verdadero	**Falso**	**Falso**
Falso	**Verdadero**	**Falso**
Falso	**Falso**	**Falso**

Operador AND (&) | Ejemplo

El **operador AND** es una **conjunción lógica**. Trabaja con dos operandos. Sólo si ambos operandos son verdaderos, dará como resultado Verdadero. Veamos un ejemplo:

```
Algoritmo operadorNot

    var = 2 < 3 ∧ "a" == "a"
    Escribir var

FinAlgoritmo
```

```
*** Ejecución Iniciada. ***
VERDADERO
*** Ejecución Finalizada. ***
```



Operador AND (&) | Ejemplo

En el siguiente ejemplo se evalúa si un alumno aprobó una materia universitaria. Para ello, debe tener una nota mayor o igual a 4 en los dos parciales.

```
Algoritmo operadorAndCondicional
     primerParcial = 7
     segundoParcial = 10
     // AND: sólo si las dos condiciones se cumplen, dirá “Materia aprobada”
     Si primerParcial >= 4 ∧ segundoParcial >= 4 Entonces
          Escribir "Materia aprobada"
     SiNo
          Escribir "Materia desaprobada"
     FinSi
FinAlgoritmo
```

Operador AND (&) junto con OR (|)

Veamos un ejemplo que combina operadores aritméticos con un operador lógico OR, para investigar qué resultado se obtiene:

```
Algoritmo ejemplo_and_1
    var1 = 2 * 3 > 4 - 1 // Verdadero
    var2 = 2 < 1 // Falso
    var3 = var1 ∧ var2 // Verdadero Y Falso == Falso

    Escribir var3
    var3 = Falso ∧ var2 // Falso Y Falso == Falso
    Escribir var3
FinAlgoritmo
```

```
*** Ejecución Iniciada. ***
FALSO
FALSO
*** Ejecución Finalizada. ***
```

Operador OR (|) | Tabla de verdad

El operador lógico OR se utiliza para combinar dos condiciones y produce un resultado verdadero (*true*) **si al menos una de las condiciones es verdadera**.

Entradas		Salida
A	**B**	**A | B**
Verdadero	**Verdadero**	**Verdadero**
Verdadero	**Falso**	**Verdadero**
Falso	**Verdadero**	**Verdadero**
Falso	**Falso**	**Falso**

Operador OR (|) | Ejemplo

El **operador OR** es una **disyunción lógica**. Trabaja con dos operandos. Sólo si ambos operandos son falsos, dará como resultado Falso; si no, Verdadero. Veamos un ejemplo:

```
Algoritmo operadorOr

    var = 2 < 3 V "a" == "a"
    Escribir var

FinAlgoritmo
```

```
*** Ejecución Iniciada. ***
VERDADERO
*** Ejecución Finalizada. ***
```

Operador OR (|) | Ejemplo

El siguiente código solo admite en un curso a los mayores de 18 años o aquellos menores que ya hayan finalizado el secundario:

```
Algoritmo operadorOrCondicional
     edad = 17 // el aspirante es menor de edad
     secundarioTerminado = Verdadero // pero ya terminó el secundario
     // OR: con que una sola condición se cumpla, toda la expresión resulta Verdadera
     Si edad >= 17 V secundarioTerminado == Verdadero Entonces
          Escribir "Cumplís los requisitos para cursar"
     SiNo
          Escribir "No cumplís los requisitos para cursar"
     FinSi
FinAlgoritmo
```

Operador OR (|) y operadores aritméticos

Veremos un ejemplo que combina operadores aritméticos con un operador lógico OR, para investigar qué resulta.

```
Algoritmo ejemplo_or_1
    var1 = 2 * 3 > 4 - 1 // Verdadero
    var2 = 2 < 1 // Falso
    var3 = var1 v var2 // Verdadero

    Escribir var3
    var3 = Falso v var2
    Escribir var3
FinAlgoritmo
```

```
*** Ejecución Iniciada. ***
VERDADERO
FALSO
*** Ejecución Finalizada. ***
```

Operadores lógicos | Resumen

- En un operador lógico **AND**, la expresión sólo dará Verdadero si la primera condición **Y** la segunda se cumplen.

- En un operador lógico **OR**, basta con que la primera condición **O** la segunda se cumpla para que toda la expresión resulte Verdadera.

- Un operador lógico **NOT** sólo dará Verdadero si se aplica a un operando que era Falso, y Falso si se aplica a uno que era Verdadero.

Jerarquía de operadores

PSeInt posee **operadores aritméticos**, **operadores relacionales** y **operadores lógicos**. Necesitamos conocer en qué orden se evalúan las operaciones que combinan diferentes tipos de operadores.

Siempre se resuelven primero los operadores aritméticos, luego los operadores relacionales y finalmente los operadores lógicos. Podemos usar **paréntesis** para alterar este orden jerárquico, forzando a PSeInt a resolver primero lo que está entre ellos.

Jerarquía de operadores

El orden jerárquico general es el siguiente:

1. Operadores aritméticos.
2. Operadores relacionales.
3. Operadores lógicos.

En la columna de la derecha podemos ver en detalle la prioridad de cada operador particular dentro de su clase.

Cuando la prioridad es la misma, se resuelve la operación de izquierda a derecha.

1- Operadores aritméticos	( )
	^ %
	* /
	+ -
2- Operadores relacionales	< <= == != >= >
3- Operadores lógicos	!
	&
	|

Desafíos

Desafío 1: Autenticación de Usuario

Escribe un programa que solicite al usuario ingresar un **nombre de usuario** y una **contraseña**.

Utiliza operadores lógicos para verificar que el nombre de usuario sea "*admin*" y la contraseña sea "*secreta*". Si ambos son correctos, muestra un mensaje de autenticación exitosa; de lo contrario, indica que la autenticación ha fallado.

Desafío 2: Verificación de Rango de Números

Desarrolla un programa que solicite al usuario ingresar un número.

Utiliza operadores lógicos para verificar si el número es impar y está en el rango de 10 a 50 (inclusive).

Muestra un mensaje indicando si el número cumple con lo solicitado o no.

Desafío 3: Validación de Números Positivos

Crea un programa que solicite al usuario ingresar dos números.

Utiliza operadores lógicos para verificar si ambos números son positivos.

Muestra un mensaje indicando si ambos números son positivos o si al menos uno de ellos no cumple con esta condición.

Desafío 4: Sistema de Descuento

Diseña un programa para una tienda que ofrezca descuentos a los clientes.

Solicita al usuario ingresar el total de la compra y si es miembro del programa de lealtad (“sí” o “no”).

Aplica un descuento del 10% si la compra es mayor a $100 y el cliente es miembro del programa de lealtad.

Si la compra es mayor a $200, aplica un descuento adicional del 5%.

Muestra el total final con los descuentos aplicados.

Desafío 5: Calculadora de Índice de Masa Corporal

Diseña un programa que solicite al usuario ingresar su peso (en kilogramos) y su altura (en metros). Calcula el Índice de Masa Corporal (IMC) utilizando la fórmula

$$IMC = \frac{peso(Kg)}{altura^2(m)}$$

Luego, clasifica la condición física del usuario utilizando operadores lógicos:

- IMC menor que 18.5: Bajo peso.
- IMC mayor que 18.5 y menor que 25: Peso normal.
- IMC mayor que 25 y menor 30: Sobrepeso.
- IMC mayor o igual que 30: Obesidad.

Muestra el IMC calculado y la clasificación de la condición física del usuario.

Material extra

Artículos de interés

Material extra:

- Operadores lógicos en PSeInt | CoderFree

Videos:

- Operadores en programación | Fernando Masino
- Operadores lógicos en PSeInt | Carlos Cimino

No te olvides de dar el presente

Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.
- Realizar el Ejercicio de Repaso.

Todo en el Aula Virtual.

Muchas gracias por tu atención.

Nos vemos pronto