



UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

Preparatoria 20

Fundamentos de la programacion

Doc Hilda Patricia Tamez Villalon

APLICACIÓN DEL LENGUAJE PHP

Equipo 2

Axel Gabriel Aguiar Cavazos

Farid Cavazos Robles

Ian Daniel Sánchez Leal

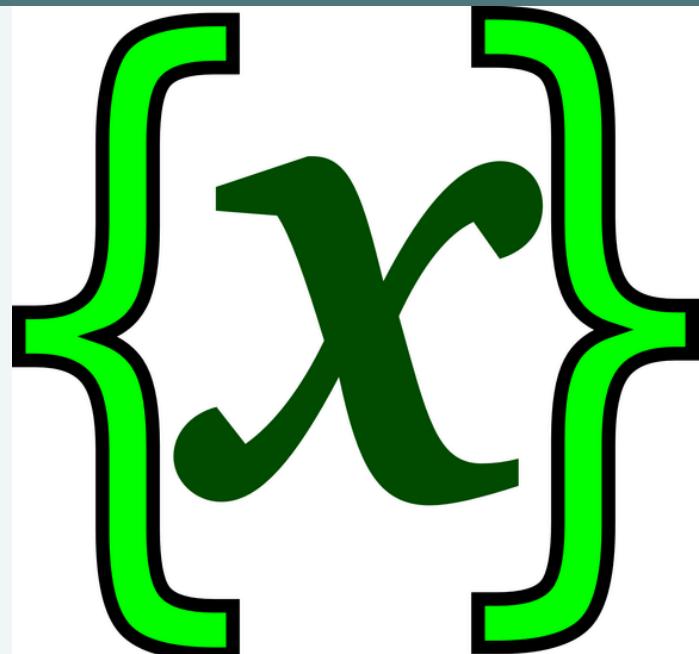
Aldo Gael Alanis Zuñiga

Carlos Alfredo Gutiérrez Cavazos



VARIABLES Y CONSTANTES

Variable: Un elemento que permite almacenar información. En los lenguajes de programación las variables se identifican por su nombre, los nombres de las variables deben empezar con el carácter \$, a continuación tiene que haber una letra o un guion bajo



Constante: Es un identificador (nombre) para un valor simple. Un nombre constante válido comienza con una letra o un carácter de subrayado(sin signo \$ antes de la constante)

```
Archivos Editor Selección Ver Ejecutar Terminal
C:\xampp\htdocs>euroshop>constantes.php
1: //Definiendo una constante y su valor
2: define ("CUOTA", 200000);
3: /* 
4: 1- No se necesita usar $
5: 2- solo se definen con el método define
6: 3- solo se define una única vez
7: 4- es global se puede acceder
8: cualquier parte
9: */
10: 
11: //Definiendo una constante y su valor
12: define ("CUOTA", 200000);
13: $valorCuota = CUOTA;
14: echo "El valor de la cuota: $valorCuota";
```

Constantes

OPERADORES

Si tomamos de las matemáticas que un operador es un símbolo que indica que debe ser llevada a cabo una operación especificada sobre un cierto número de operadores, tendremos que en todo lenguaje de programación encontraremos muchos equivalentes y PHP no es la excepción y por su semejanza de C++ o Java tendremos que a un programador ya inducido en estos lenguajes no se le hará nada complicado.

OPÉRADORES

Operadores Aritméticos

Ejemplo	Nombre	Resultado
$-\$a$	Negación	Opuesto de $\$a$
$\$a + \b	Adición	Suma de $\$a$ y $\$b$
$\$a - \b	Sustracción	Diferencia de $\$a$ y $\$b$
$\$a * \b	Multiplicación	Producto de $\$a$ y $\$b$
$\$a / \b	División	Cociente de $\$a$ y $\$b$
$\$a \% \b	Módulo	Resto de $\$a$ dividido por $\$b$

OPÉRADORES



Operadores de Incremento/decremento

Ejemplo	Nombre	Efecto
<code>++\$a</code>	Pre-incremento	Incrementa \$a en un, y luego retorna \$a.
<code>\$a++</code>	Post-incremento	Retorna \$a, y luego incrementa \$a en uno
<code>--\$a</code>	Pre-decremento	Decrementa \$a en uno, luego retorna \$a
<code>\$a--</code>	Post-decremento	Retorna \$a, luego decrementa \$a en uno

OPERADORES

Operadores de comparación

Ejemplo	Nombre	Resultado
<code>\$a == \$b</code>	Igual	TRUE si \$a es igual a \$b después de la manipulación de tipos
<code>\$a === \$b</code>	Idéntico	TRUE si \$a es igual a \$b, y son del mismo tipo
<code>\$a != \$b</code>	Diferente	TRUE si \$a no es igual a \$b después de la manipulación de tipos
<code>\$a !== \$b</code>	No idéntico	TRUE si \$a no es igual a \$b después de la manipulación de tipos
<code>\$a < \$b</code>	Menor que	TRUE si \$a es estrictamente menor que \$b
<code>\$a > \$b</code>	Mayor que	TRUE si \$a es estrictamente mayor que \$b
<code>\$a <= \$b</code>	Menor o igual que	TRUE si \$a es menor o igual que \$b
<code>\$a >= \$b</code>	Mayor o igual que	TRUE si \$a es mayor o igual que \$b

OPÉRADORES

Operadores Lógicos

Ejemplo	Nombre	Resultado
$\$a \text{ and } \b	And (y)	TRUE si tanto $\$a$ como $\$b$ son TRUE
$\$a \text{ or } \b	Or (o inclusivo)	TRUE si cualquiera de $\$a$ o $\$b$ es TRUE
$\$a \text{ xor } \b	Xor (o exclusivo)	TRUE si $\$a$ o $\$b$ es TRUE, pero no ambos
$! \$a$	Not (no)	TRUE si $\$a$ no es TRUE
$\$a \&& \b	And (y)	TRUE si tanto $\$a$ como $\$b$ son TRUE
$\$a \text{ II } \b	Or (o inclusivo)	TRUE si cualquiera de $\$a$ o $\$b$ es TRUE

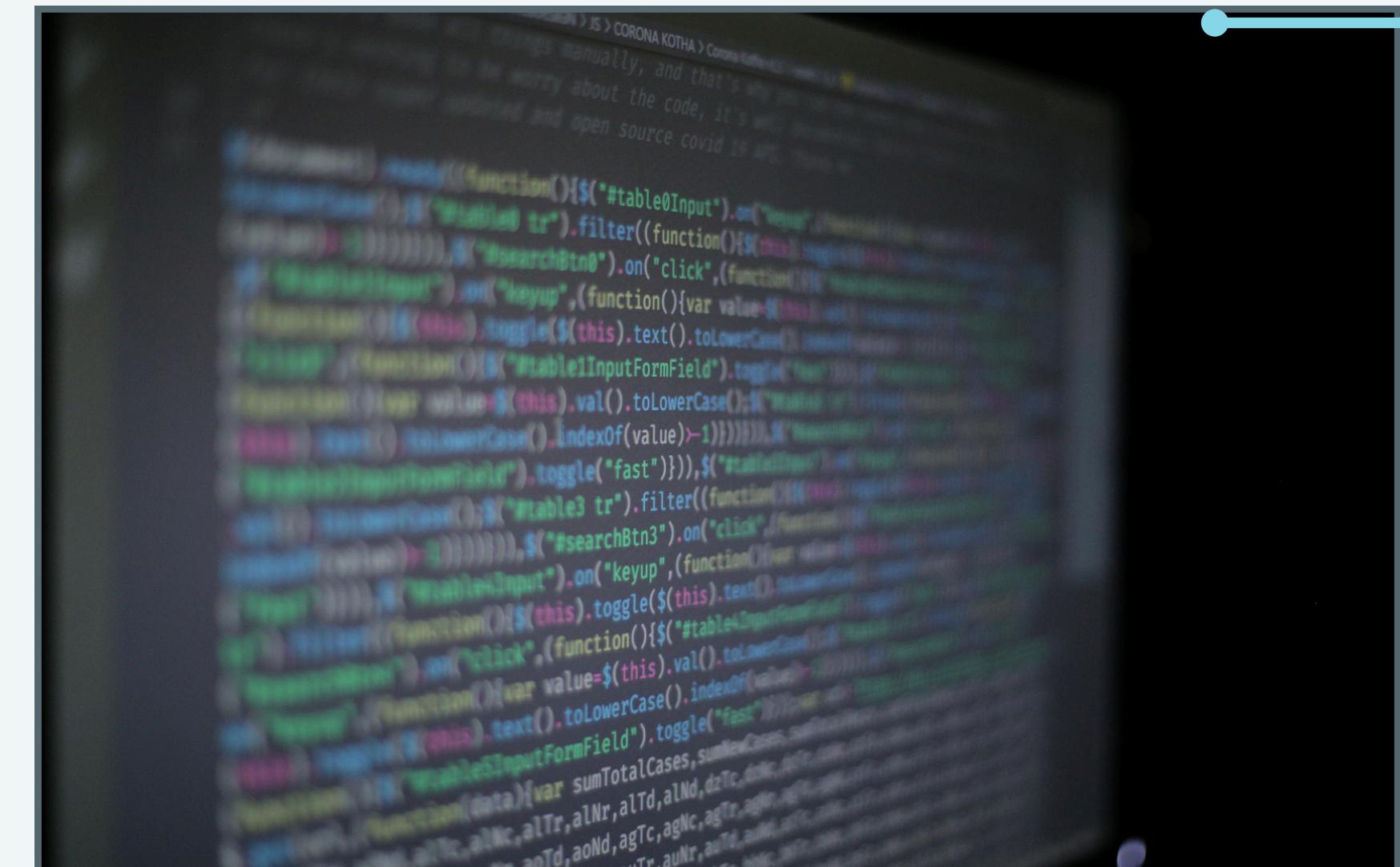
ESTRUCTURAS CONDICIONALES

- Estructuras condicional IF

Las condiciones if, else y elseif permiten condicionar la ejecución de un bloque de sentencias al cumplimiento de una condición

La sintaxis

```
if(expresion_1){  
    bloque_de_sentencias_1  
} elseif(expresion_2){  
    bloque_de_sentencias_2  
}else{  
    bloque_de_sentencias_3  
}
```



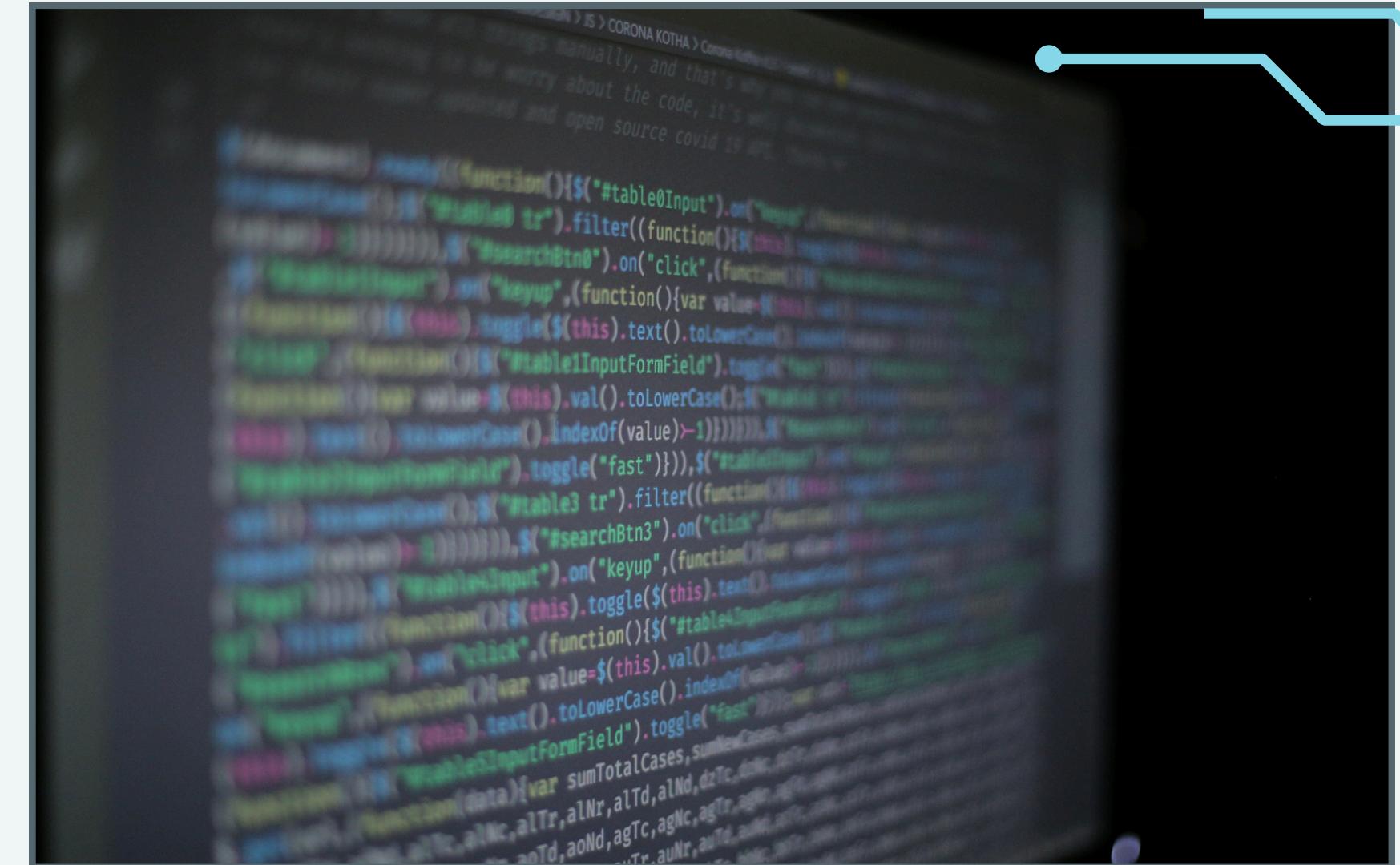
Farid Cavazos Robles

ESTRUCTURAS CONDICIONALES

- Estructura condicional switch
La sentencia switch es equivalente a una construcción if...elseif...en las que las expresiones son comparaciones de igualdad de la misma expresión

La sintaxis

```
switch(expresion_1)
{
    case valor_1:
        bloque_de_sentencias_1;
        break;
    case valor_2:
        bloque_de_sentencias_2;
        break;
    ...
    case valor_n:
        bloque_de_sentencias_n;
        break;
}
```



ESTRUCTURAS CONDICIONALES

- Estructuras condicional IF

Las condiciones if, else y elseif permiten condicionar la ejecucion de un bloque de sentencias al cumplimiento de una condicion

La sintaxis

```
if(expresion)
{
    bloque_de_sentencias
}
```



BUCLLES

- La expresión while evalua al principio si el resultado es TRUE y ejecuta el código, y FALSE el bucle se termina. La sintaxis del bucle while es:

while (expresión)

{

bloque_de_sentencias

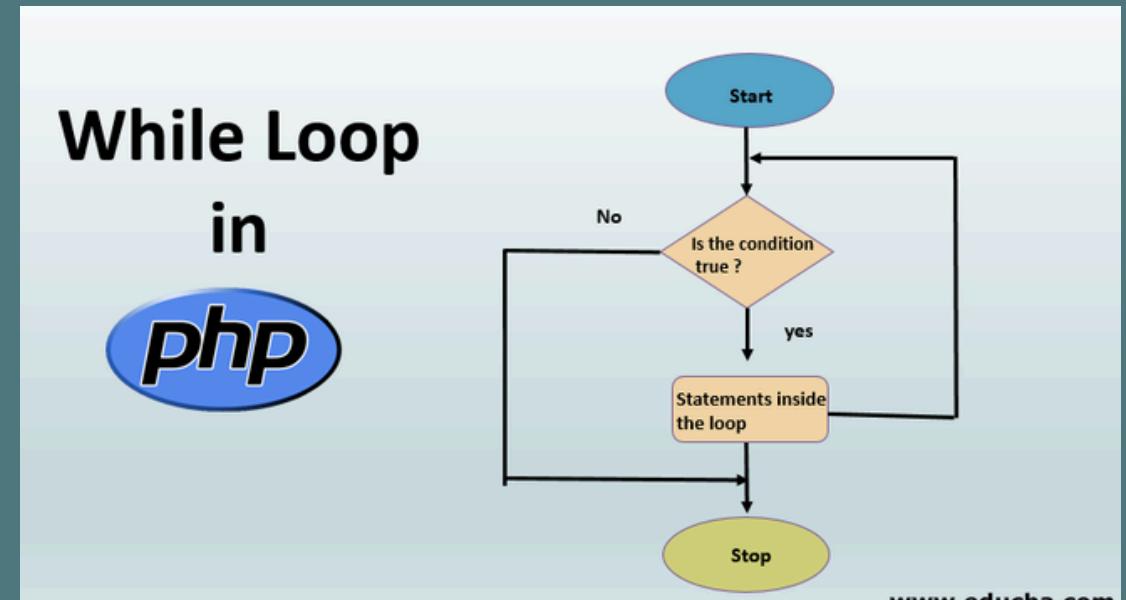
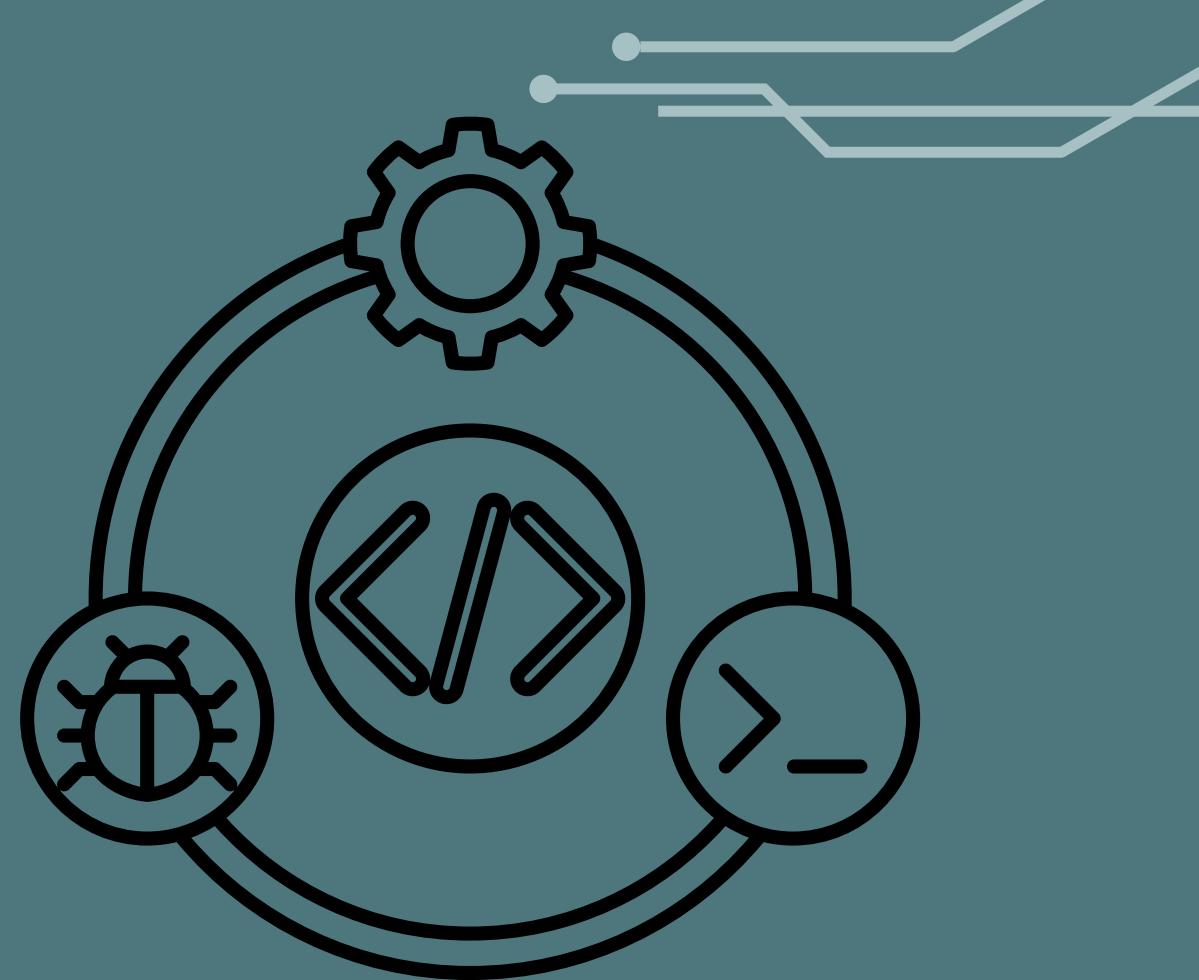
}

- La principal diferencia entre el bucle while es el do...while es que se ejecuta al menos una vez y el while depende de que la condición sea verdadera. La sintaxis del bucle do while es:

do{

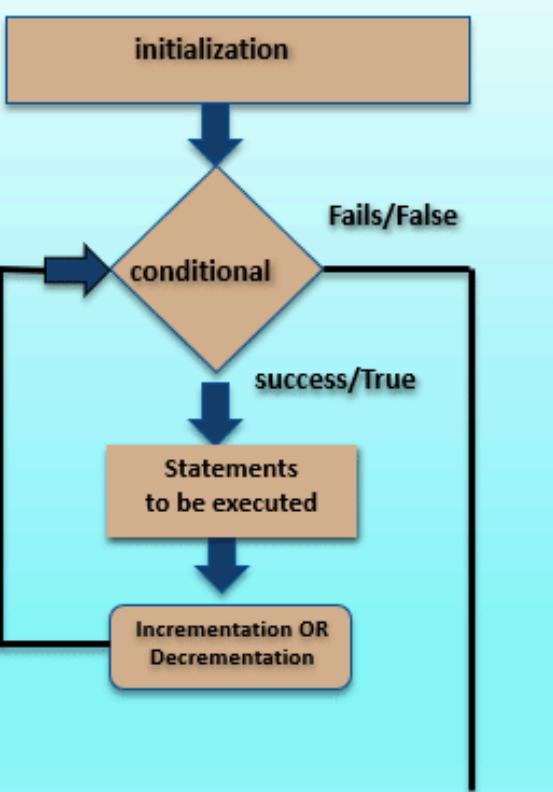
bloque_de_sentencias

} while (expresión)



BUCLÉS

For Loop in

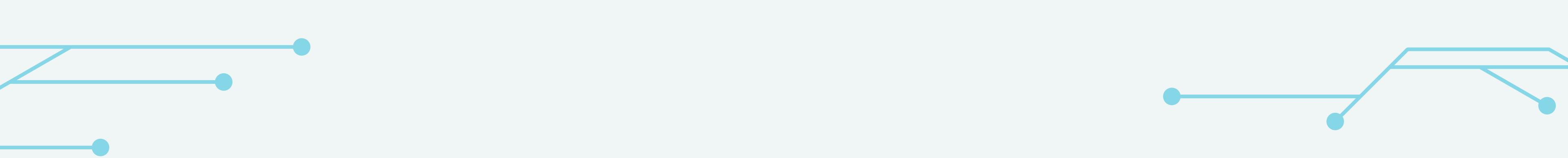


- En el ciclo For la expresión final se evalua al principio de cada iteración, si el resultado es TRUE se ejecuta el bloque de sentencias; si el resultado es FALSE el bucle se termina. Su sintaxis es:

```
for (expresión_inicial; expresión_final; expresión_paso)
{
    bloque_de_sentencias
}
```

- El ciclo for each ejecuta el bloque de sentencias tantas veces como elementos contenga la matriz \$matriz y, en cada iteración, la variable \$valor toma uno de los valores de la matriz. La sintaxis es:

```
Foreach ($matriz as $valor)
{
    bloque_de_sentencias
}
```



**MUCHAS
GRACIAS**