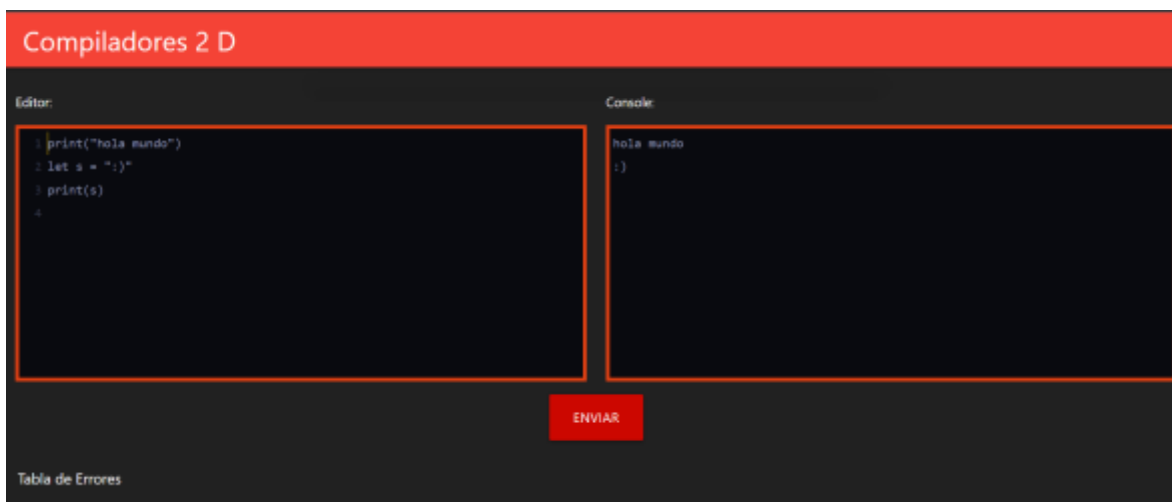


Manual Usuario

Cristian Daniel Gomez Escobar
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

La interfaz con la que se encontrara el usuario



Aquí será donde se podrá ejecutar el programa y se podrá ingresar el código en el programa

COMENTARIOS

Lo que este en un comentario no se tomara en cuenta en el programa

Ej:

```
// Esto es un comentario de una línea
```

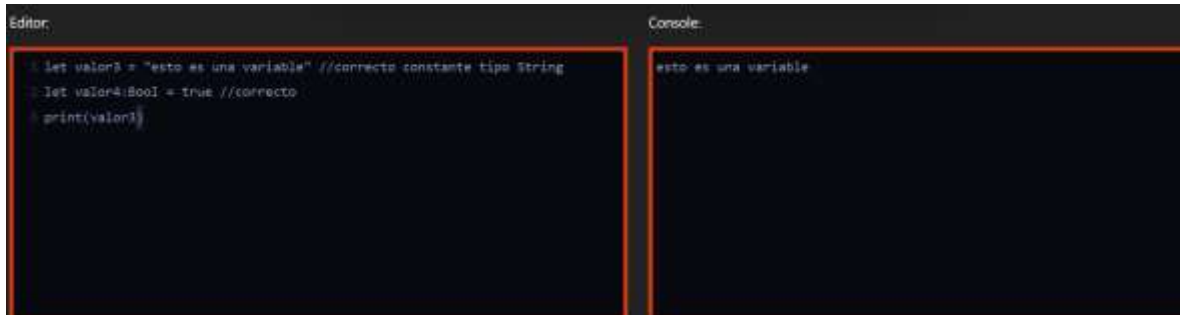
```
/* Esto es un comentario multilínea */
```

Tipos de datos primitivos

Tipo primitivo	Definición	Rango (teorico)	Valor por defecto
Int	Acepta valores numéricos enteros	-2,147,483,648 a +2,147,483,647	nil
Float	Acepta valores numéricos de punto flotante	1.2E-38 a 3.4E+38 (6 dígitos de precisión)	nil
String	Acepta cadenas de caracteres	[0, 65535] caracteres (acotado por conveniencia)	nil
Bool	Acepta valores lógicos de verdadero y falso	true false	nil
Character	Acepta un solo caracter ASCII	[0, 65535] caracteres	nil

COSTANTES

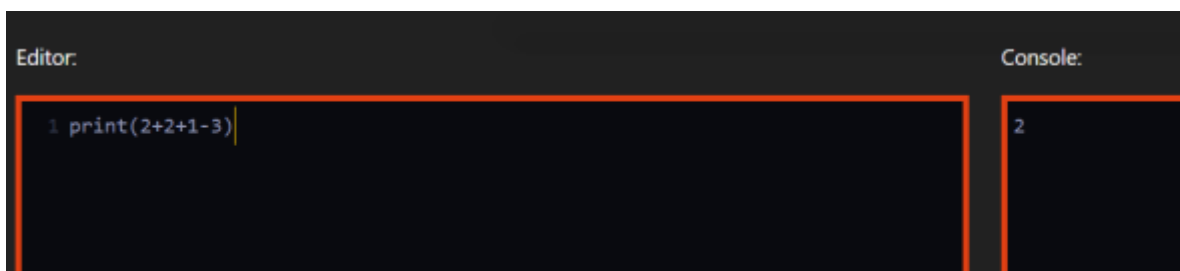
Estas serán declaradas anteponiendo la palabra reservada “let”, esto indica que el valor de dicha constante no podrá cambiar a lo largo de la ejecución, por lo tanto, no será válida cualquier asignación que se haga sobre una constante y deberá ser notificada como un error.



The screenshot shows a code editor on the left and a console on the right. The editor contains three lines of JavaScript code: `let valor3 = "esto es una variable" //correcto constante tipo String`, `let valor4:Bool = true //correcto`, and `print(valor3)`. The console displays the output of the first line: `esto es una variable`.

OPERADORES ARITMETICOS

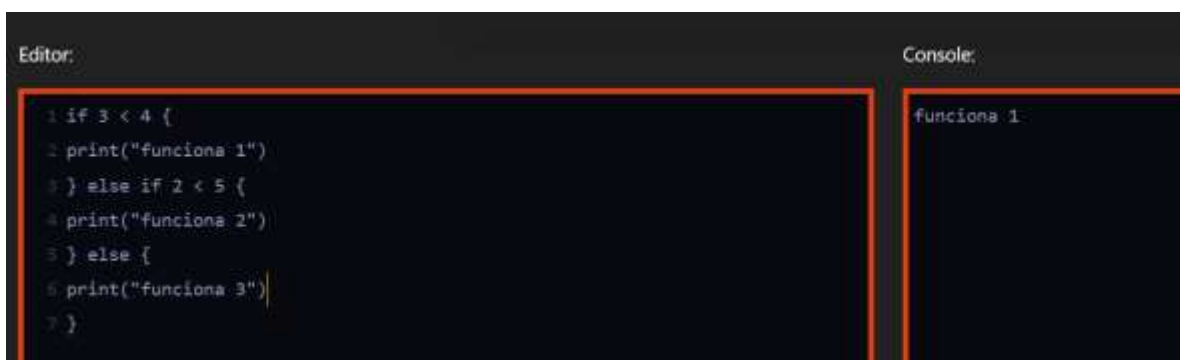
Estos tomaran valores numéricos de expresiones o retronaran un valor numérico



The screenshot shows a code editor on the left and a console on the right. The editor contains the line `print(2+2+1-3)`. The console displays the result of the expression: `2`.

IF

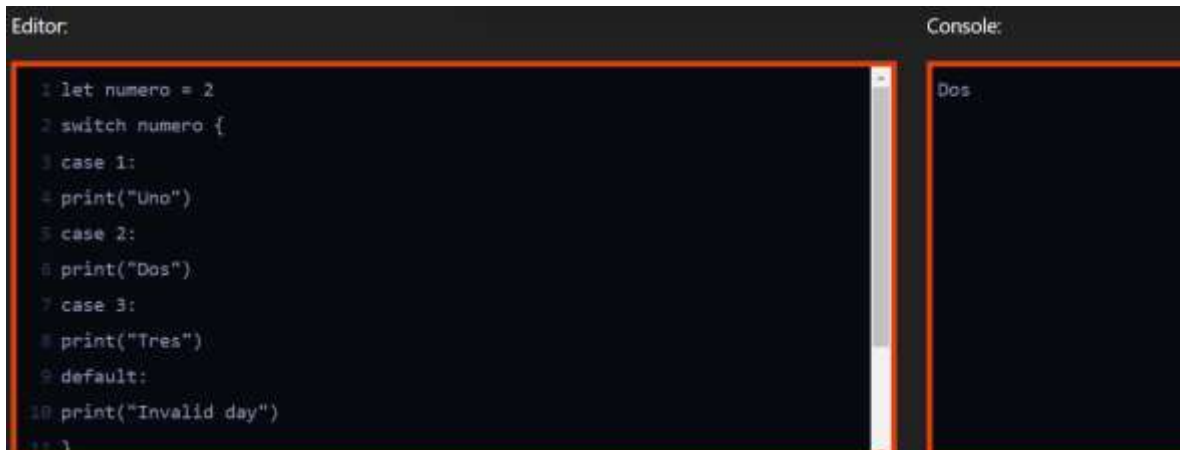
Este Ejecuta un bloque de sentencias si una condición especificada es evaluada como verdadera. Si la condición es evaluada como falsa, otro bloque de sentencias puede ser ejecutado.



The screenshot shows a code editor on the left and a console on the right. The editor contains an if-else statement: `if 3 < 4 { print("funciona 1") } else if 2 < 5 { print("funciona 2") } else { print("funciona 3") }`. The console displays the output of the first condition: `funciona 1`.

SENTENCIA SWITCH

Este evalúa una expresión, comparando el valor de esa expresión con un case, y ejecuta declaraciones asociadas a ese case, así como las declaraciones en los case que siguen.



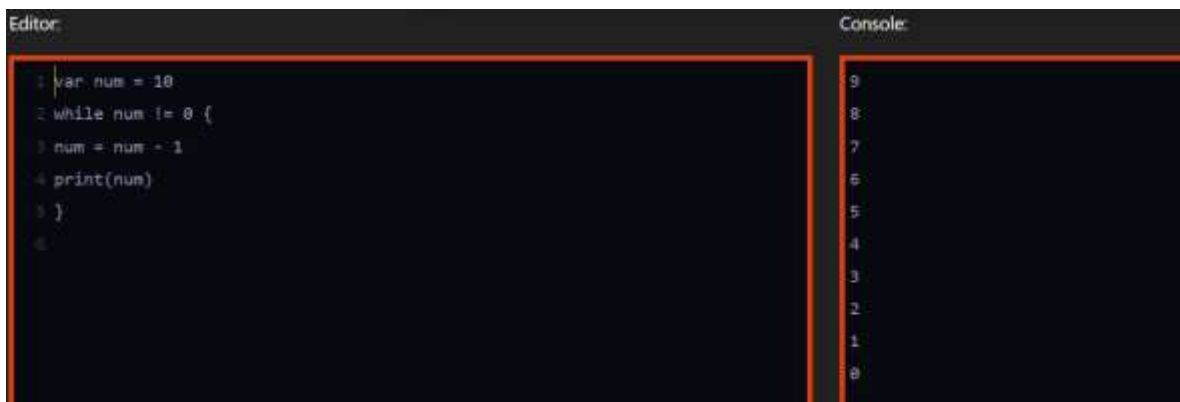
The screenshot shows a code editor on the left and a console on the right. The code in the editor is a JavaScript switch statement that checks the value of 'numero' and prints the corresponding string. The console shows the output 'Dos'.

```
Editor:
1 let numero = 2
2 switch numero {
3   case 1:
4     print("Uno")
5   case 2:
6     print("Dos")
7   case 3:
8     print("Tres")
9   default:
10    print("Invalid day")
11 }
```

```
Console:
Dos
```

WHILE

Este crea un bucle que ejecuta un bloque de sentencias especificadas mientras cierta condición se evalúe como verdadera (true). Dicha condición es evaluada antes de ejecutar el bloque de sentencias y al final de cada iteración



The screenshot shows a code editor on the left and a console on the right. The code in the editor is a JavaScript while loop that prints numbers from 10 down to 0. The console shows the output of the loop.

```
Editor:
1 var num = 10
2 while num != 0 {
3   num = num - 1
4   print(num)
5 }
6
```

```
Console:
9
8
7
6
5
4
3
2
1
0
```

FOR

Este bucle for se comportará como un for moderno, que recorrerá alguna estructura compuesta, en este caso sobre: cadenas, arreglos unidimensionales y rangos. La variable que recorre los valores se comportará como una constante, por lo tanto, no se podrán modificar su valor en el bloque de sentencias, su valor únicamente cambiará con respecto a las iteraciones.

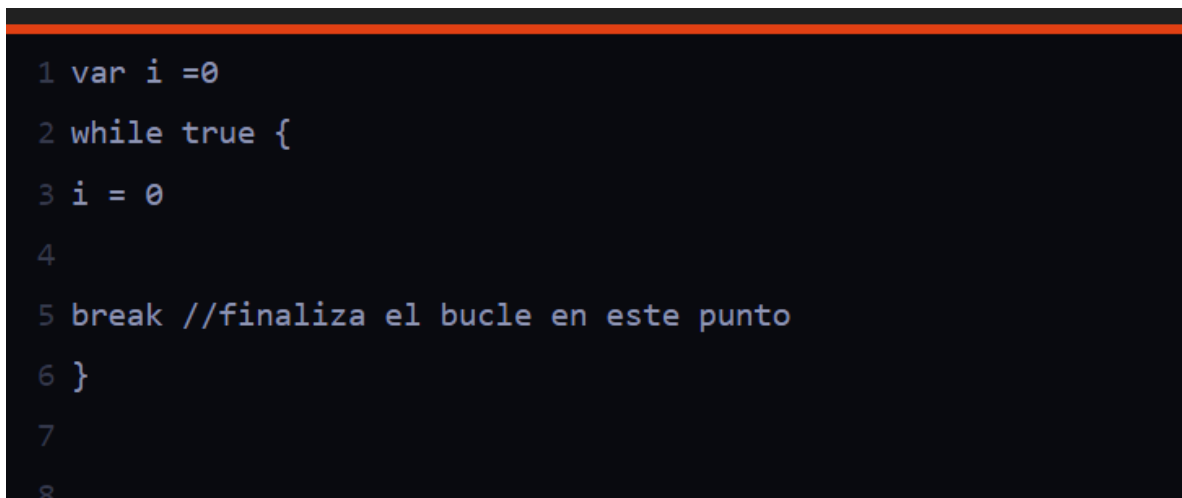
```
1 for i in 1..5 {
2   print(i)
3 }
4
5 for c in "hola" {
6   print(c)
7 }
8
```



BREAK

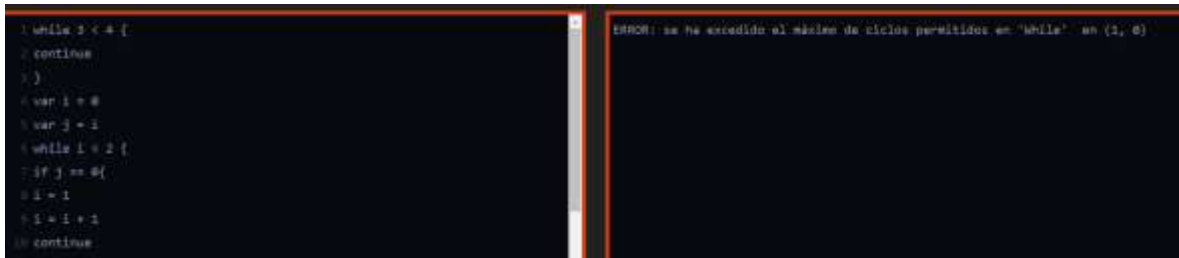
Esta sentencia termina el bucle actual ó sentencia switch y transfiere el control del programa a la siguiente sentencia a la sentencia de terminación de estos elementos.

```
1 var i = 0
2 while true {
3   i = 0
4
5   break //finaliza el bucle en este punto
6 }
7
8
```



CONTINUE

Esta sentencia termina la ejecución de las sentencias de la iteración actual (en un bucle) y continúa la ejecución con la próxima iteración

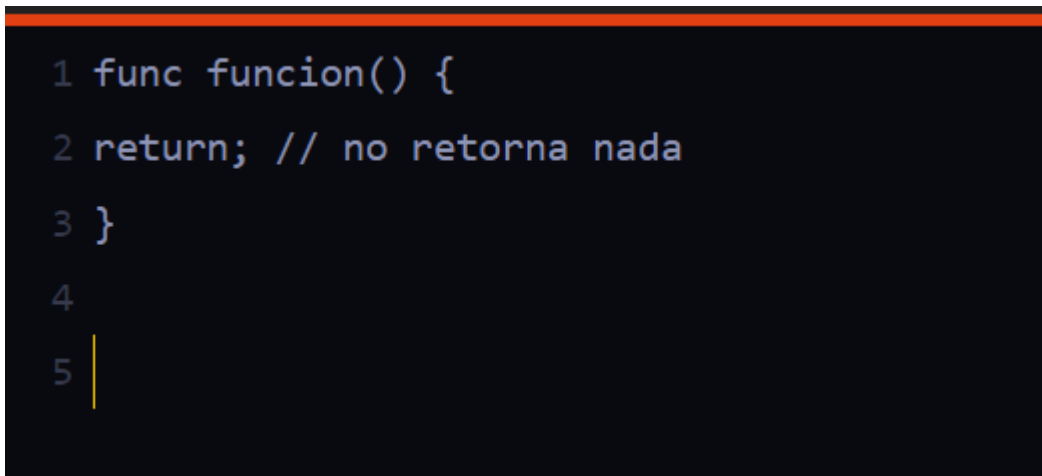


```
1 while i < 4 {  
2   continue  
3 }  
4 var i = 0  
5 var j = 1  
6 while i < 2 {  
7   if j == 0 {  
8     i = 1  
9     i = i + 1  
10    continue  
11  }
```

ERROR: se ha excedido el máximo de ciclos permitidos en 'while' en (1, 0)

RETURN

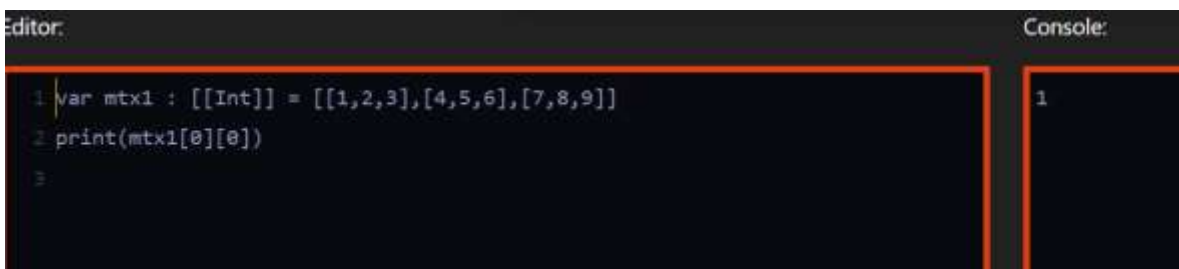
Sentencia que finaliza la ejecución de la función actual, puede o no especificar un valor para ser devuelto a quien llama a la función.



```
1 func funcion() {  
2   return; // no retorna nada  
3 }  
4  
5
```

MATRICES

permiten almacenar solamente datos de tipo primitivo, la diferencia principal entre el vector y la matriz es que esta última organiza sus elementos en n dimensiones y la manipulación de datos es con la notación [] además que su tamaño



```
Editor: Console:  
1 var mtx1 : [[Int]] = [[1,2,3],[4,5,6],[7,8,9]]  
2 print(mtx1[0][0])  
3
```

1

STRUCTS

acceso a atributos de los structs por medio del operador '.', el cual nos permite acceder a los atributos ya sea para asignarles un valor ó acceder al valor.

```
1 struct Verdura{
2 let nombre: String
3 var precio: Int
4 }
5 var brocoli = Verdura(nombre:"brocoli", precio: 5)
6
7 print(brocoli.nombre)
```

brocoli

FUNCIONES

```
1 func fun () -> String {
2 return "valor";
3 }
4 print(fun())
```

valor