# Intérprete de un lenguaje para SWIFT (Examen septiembre 2020)

Swift es un lenguaje de programación de propósito general que se puede usar en dispositivos móviles, de escritorio, servidores, etc. En su diseño se ha tenido en cuenta la seguridad, el rendimiento y la programación interactiva combinando lo mejor de los nuevos avances. Se sitúa en el entorno de Apple, pero recibe contribuciones de la comunidad de código abierto ("open source").

Se puede acceder a un completo manual del lenguaje de programación Swift en el siguiente enlace:

https://swift.org/documentation

Aunque los fundamentos que aparecen en este ejercicio se pueden encontrar en:

https://docs.swift.org/swift-book/GuidedTour/GuidedTour.html

https://docs.swift.org/swift-book/LanguageGuide/TheBasics.html

https://docs.swift.org/swift-book/LanguageGuide/BasicOperators.html

https://docs.swift.org/swift-book/LanguageGuide/ControlFlow.html

También hay disponible un compilador online de Swift, donde podrá probar el resultado de cualquier fichero escrito en el lenguaje de programación Swift: <a href="http://online.swiftplayground.run">http://online.swiftplayground.run</a>

Se desea **implementar mediante Java, JFlex y Cup un intérprete del lenguaje de programación Swift** al que llamaremos **SwiftPL** pero limitando y variando ligeramente sus funciones a un núcleo básico.

#### Aspectos léxicos

El cuerpo de un programa en SwiftPL se compone de una o más <u>sentencias</u>, cada una de ellas escrita sin partirse en diferentes líneas. Cada línea puede acabar <u>opcionalmente</u> con un punto y coma ';' . En una misma línea podría existir más de una sentencia y se separarían mediante puntos y coma. El uso de espacios o tabuladores no tiene efecto (el sangrado de las líneas tampoco). No se permiten sentencias vacías terminadas en punto y coma.

Ejemplos de sentencias válidas en SwiftPL	Ejemplos de sentencias no válidas en SwiftPL
2 + 3	2 +
2 + 3;	2 + 3 4 +5
2 + 3; 4 + 5	
2 + 3; 4 + 5;	

NOTA: en Swift algunas sentencias podrían ocupar más de una línea, pero en este subconjunto del lenguaje no tendremos en cuenta esa opción.

Para mostrar cualquier resultado por pantalla existe el comando <u>print</u> que sirve para imprimir. Por ejemplo, la sentencia para imprimir por pantalla el texto "Hola Mundo" sería <u>print("Hello World")</u>. Esta sentencia incluye el salto de línea después de escribir lo que recibe como parámetro.

Cada sentencia que se ejecuta devuelve un valor. Ese valor NO se imprime automáticamente por pantalla y debe ser asignado a una variable o a una constante si se quiere utilizar posteriormente. Por ejemplo, las sentencias anteriores (2+3; 4+5) se ejecutarían, pero no tendría ninguna salida por pantalla. Tampoco tendrían ningún efecto posterior porque el resultado no es almacenado.

Si se encuentra una sentencia que contenga un <u>error</u>, se generará como salida la palabra error y se detendrá la ejecución.

Los <u>comentarios</u> son parecidos a los de C. Se usa la doble barra (//) para hacer un comentario en línea. También tiene comentarios de bloque, usando /\* para iniciar el comentario y \*/ para terminarlo. La diferencia es que se usa anidamiento de bloques de comentarios, para facilitar la acción de comentar bloques de código que ya pudiesen contener comentarios.

C C .CDI	C 1: 1
Sentencias en SwiftPL	Salida por pantalla
1+2;	4
print(1 + 3);	6
1 + 4	
print(1 + 5)	
1+2; print(1 + 3);	4
1 + 4; print(1 + 5)	6
1+2; print(1 + 3);	4
	6
1 + 4; print(1 + 5);	
1+2; print(1 + 3);	4
;	error (sobra un ;)
1 + 4; print(1 + 5);	, ,
1 + 4; print(1 + 5); 1 + 2	4
print (2+2)	5
print (3+2);	error (sobra un ;)
print (4+2;);	==== (===============================
print (5+2);	
print (1+2); // comentario en linea	3
/* print (1+3); bloque comentado */	5
print (1+4); // comentario en linea	
print (1+2); // comentario en linea	3
print (1+3); // comentario en linea /* sique	4
/* print (1+4); bloque	9
comentado */	9
Comentado */	
/*	
,	
print (1+5);	
/* print (1+6); bloque	
comentado */	
<pre>print (1+7); // comentario en linea */</pre>	
<pre>print (1+8); // comentario en linea</pre>	
<pre>print (1+2); // comentario en linea</pre>	3
/*	error
print (1+5);	(porque en la línea 5
/* print (1+6); bloque	se acaba el
comentado */	comentario y continua
<pre>print (1+7); // comentario en linea */ sigue</pre>	con texto no legal)
<pre>print (1+8); // comentario en linea</pre>	

Los <u>nombres para las variables y constantes</u> utilizan caracteres alfanuméricos y el guión bajo ('\_'). El nombre debe empezar por una letra. Este lenguaje es sensible a mayúsculas y minúsculas, por lo que si dos nombres usan diferentes letras (aunque sean igual en mayúscula y minúscula), serán diferentes. Para definir constantes se usa la palabra reservada let y para las variables, la palabra reservada var.

Para la <u>asignación</u> se usa el signo '=' para hacer que el valor de la sentencia a la derecha del signo sea asignado en la variable o constante a la izquierda del signo. Tiene un comportamiento especial cuando se usa en la inicialización de variables y constantes (ver más adelante).

Sentencias en SwiftPL	Salida por pantalla
var a = 1;	1
print (a)	
let a = 1	3
<pre>print (a+2);</pre>	
<pre>var a = 1; let b = a+1;</pre>	3
<pre>print (a+b)</pre>	
<pre>var a = 1; print(a);</pre>	1
let $b = A+1;$	error (variable A no declarada)
<pre>print (a+b)</pre>	
<pre>var a = 2; print(a);</pre>	2
let 1_mas_a = 1 + a;	error (var. no empieza por letra)
<pre>print (a+b)</pre>	
var a = 2	9
let $b = a+1;$	
a=a*3	
<pre>print (a+b)</pre>	
a = 1; print(a)	error
b = a+1;	<pre>/* por a = 1; y b = a+1; porque hay que</pre>
<pre>print (a+b)</pre>	usar let o var*/
let b = a+1;	error
var a = 1;	/* por let b = a+1; porque a está sin
<pre>print (a+b)</pre>	definir */

Existen múltiples tipos de datos, pero usaremos únicamente los tipos numéricos entero y real (Int y Double) y el tipo booleano (Bool). Es posible definir varias variables o constantes del mismo tipo en una misma sentencia sin tener que inicializarlas. En el caso de las constantes sólo se podrán inicializar una única vez (en la declaración de la variable o después), mientras que las variables podrán cambiar su valor tantas veces como se desee. En el caso de declararlas sin inicializarlas (variables o constantes), es preciso indicar de forma explícita el tipo de datos que usará, para lo que se indica usando el signo de los dos puntos (':') seguido del nombre del tipo.

En la declaración de variables o constantes cuando se inicialicen con un valor, no es necesario definir el tipo de dato, porque se deducirá del valor asignado. En ese caso, en la declaración con inicialización (sin explicitar el tipo) debe existir, al menos, un espacio antes y después del signo '=' porque en caso contrario se trataría de un error. No se permite usar variables que únicamente hayan sido declaradas

En caso de inicializar el valor, también se podrá hacer explícito usando el signo de los dos puntos seguido del tipo después del nombre de la variable y antes de la asignación.

Sentencias en SwiftPL	Salida por pantalla
<pre>var implicitoInt = 1</pre>	1
let implicitoDouble = 2.0	2.0
let explicitoDouble:Double = 3	3.0
<pre>print(implicitoInt)</pre>	
<pre>print(implicitoDouble)</pre>	
print(explicitoDouble)	
let implicitoDouble = 2.0	2.0
let explicitoDouble: Double = 3	1.5
<pre>print(implicitoDouble)</pre>	
<pre>print(explicitoDouble / implicitoDouble)</pre>	
var a= 1;	error (hay que dejar, al
let b =2;	menos, un espacio antes y
<pre>print (a+b)</pre>	después del igual)
var a = 2	8
let b : Int;	
b=a*3	
<pre>print (a+b)</pre>	
var a,b: Int	5
a=1	
b=a*3	
a=a*2	
<pre>print (a+b)</pre>	
var a,b: Int	error (no se puede
let b : Int;	redeclarar una variable
a=4	o constante)
b=a*3	
<pre>print (a+b)</pre>	
var a = 2	error (por b=a*3 porque no
let b = a+1;	se puede cambiar una
b=a*3	constante)
<pre>print (a+b)</pre>	
var x, y, z: Double	error (x sin inicializar)
print (x)	

Consideraremos los operadores aritméticos (para números) de suma, resta, multiplicación, división ( + , - , \* , / ). No se pueden operar variables de distinto tipo, para ello habría que hacer un casting explícito (que no se pide en este ejercicio).

Consideraremos los operadores relacionales (>, >=, <, <=, ==, !=, !, && y ||) para operaciones lógicas cuyas palabras reservadas para el valor de "verdadero" y "falso" son, respectivamente, true y false.

Se aplican los mismos criterios de prioridad y asociatividad que en C o Java, permitiendo el uso de paréntesis.

Sentencias en SwiftPL	Salida por pantalla
var c = 1.0	9.0
var d = 2.3	11.3
var e = 3.4 + 5.6	29.7
var f = c * (d + e)	
var g = e * d + e	
print(e)	
print(f)	
print(g)	
let implicitoInt = 1	2.0
let implicitoDouble = 2.0	error (no se puede
<pre>print(implicitoDouble)</pre>	operar una variable
<pre>print(implicitoDouble / implicitoInt )</pre>	Int con una Double)
let x = 3 * -4	error (el operador *
let $y = 3 - *4$	no es unario)
print(x)	, ,
let a = false	false
<pre>print(a)</pre>	true
print(!a)	
var a = true	true
<pre>print(a)</pre>	true
print(a     false)	
let a = true	true
<pre>print(a)</pre>	false
var b = 3 < 2	
print(a && b)	
let a = true	true
print(a)	error (comparación
var b = 3 < 2 < 1	ilegal)
print(a && b)	
var x, y, z: Double	true
x = 2; y = 4; z = 8	
var a: Bool ;	
a = (z / y) == x	
print (a)	

## **Arrays**

Se pueden crear arrays fácilmente usando los signos de los corchetes. Los elementos se separan por comas y después del último elemento puede quedar una coma (aunque no es necesario). Todos los datos tienen que ser el mismo tipo y, como la declaración de variables unarias, no precisa de una declaración explícita, aunque también lo admite. El acceso a un elemento del array se hace indicando la posición del elemento en el array entre corchetes, siendo 0 el número de la primera posición. En esta versión del lenguaje Swift, restringiremos la definición de arrays a aquellos que se definen en la propia declaración (es fácil crearlos desde cero o añadir o quitar elementos, pero no se tendrán en cuenta esas características en este ejercicio).

Sentencias válidas en SwiftPL	Salida por pantalla
var numbers = [3,9,27]	3
<pre>print (numbers[0])</pre>	
<pre>var numbers = [3,9,27]</pre>	[3, 9, 27]
<pre>print (numbers)</pre>	
var numbers = [3,9,27]	[3, 9, 18]

1 10	
numbers[2] = 18	
<pre>print (numbers)</pre>	
<pre>var numbers = [3,9,27]</pre>	[3, 9, 27]
<pre>print (numbers)</pre>	error (fuera de rango)
numbers[3] = 18	
<pre>print (numbers)</pre>	
<pre>var numbers = [3,9,81]</pre>	[9, 9, 81]
<pre>numbers[0] = numbers[2] / numbers[1]</pre>	
<pre>print (numbers)</pre>	
<pre>var numbers: [Double] = [3,9,81]</pre>	[3.0, 9.0, 81.0]
<pre>print (numbers)</pre>	
var n = [3,9,27]	true
let ordenado = $(n[0]< n[1]) &&(n[1]< n[2])$	
<pre>print (ordenado)</pre>	
<pre>var reales: [Double] = [3,9,81]</pre>	error (no se pueden
<pre>var enteros = [5,7,11]</pre>	operar variables de
<pre>print (reales[0] * enteros[0])</pre>	distinto tipo)

#### Sentencias de control: FOR

De entre las sentencias de control disponibles se pide implementar la del <u>bucle</u> for, que tiene la siguiente sintaxis

```
for name in sequence { statement1 }
```

donde sequence es una secuencia de elementos (un array o un rango de números) y name es el nombre de la variable que se usará para recorrer los valores de la secuencia. Esa variable será declarada de forma implícita según el tipo de la secuencia y no existirá fuera del ámbito del bucle for. Si no se necesita una variable que vaya tomando valores, se puede usar el signo de guión bajo ('\_').

Un rango se puede definir como cerrado usando la siguiente sintaxis: valor\_menor...valor\_mayor donde se pasa por todos los números enteros entre valor\_menor y valor mayor incluyendo ambos extremos.

Sentencias en SwiftPL	Salida por pantalla
<pre>var enteros = [5,7,11]</pre>	5
for numero in enteros	7
{	11
print (numero)	
}	
var enteros = [5,7,11]	5
for numero in enteros	7
{	11
print (numero)	error (la variable
}	numero no está fuera
print (numero)	del ámbito del for)
<pre>var enteros = [5,7,11]</pre>	5
for index in 02 {	7
<pre>print(enteros[index])</pre>	11
}	
let base, exponente: Int	16
base = 2	
exponente = 4	
var resultado = 1	

```
for _ in 1...exponente {
    resultado = resultado * base
}
print(resultado)

var a = [2,4,6]
    var b = [2,3]
    for i in a {
        for j in b{
            print(i+j)
        }
    }
}
```

### **Implementación**

Se pide implementar un intérprete que lea un programa escrito en **SwiftPL** de un fichero de texto con extensión .SwiftPL y que dé lugar a la realización de las distintas acciones que se describen mediante ejemplos en el ANEXO y se dirigen a la salida estándar, por ejemplo:

> java SwiftPL a16.SwiftPL

Como resultado de este ejercicio se entregará un fichero comprimido SwiftPL.zip que contenga al menos los ficheros **SwiftPL.jflex**, **SwiftPL.cup** y **SwiftPL.java** más todos aquellos ficheros .java que sean necesarios para la compilación del intérprete mediante la secuencia de instrucciones:

- > cup SwiftPL.cup
- > jflex SwiftPL.jflex
- > javac \*.java

# **ANEXO**

Construir un intérprete que permita ejecutar los siguientes tipos de sentencias: Es necesario detectar los errores y advertirlos, pero no se exigen los detalles que se dan en el ejemplo.

Entrada	Descripción	Resultado de la ejecución	EJEMPLO
1+2;		4	0_base_01.SwiftPL
print(1 + 3);		6	_ <del>_</del>
1 + 4			
print(1 + 5)			
1+2; print(1 + 3);		4	0_base_02.SwiftPL
1 + 4; print(1 + 5)		6	0 hama 02 Garift DI
1+2; print(1 + 3);		4	0_base_03.SwiftPL
1 + 4 · print(1 + 5) ·		0	
1 + 4; print(1 + 5); 1+2; print(1 + 3);		4	0 base 04.SwiftPL
;		error (sobra un ;)	
1 + 4; print(1 + 5);		·	
1 + 2		4	0_base_05.SwiftPL
print (2+2)		5	
<pre>print (3+2); print (4+2;);</pre>		error (sobra un ;)	
print (5+2);		uii , )	
print (1+2); // comentario en linea		3	0 base 06.SwiftPL
/* print (1+3); bloque comentado */		5	
print (1+4); // comentario en linea			
print (1+2); // comentario en linea		3	0_base_07.SwiftPL
print (1+3); // comentario en linea /* sigue		4	
<pre>/* print (1+4); bloque comentado */</pre>		9	
Comencado "/			
/*			
print (1+5);			
/* print (1+6); bloque	Fichero con		
comentado */	elementos		
print (1+7); // comentario en linea */	básicos, comentarios,		
print (1+8); // comentario en linea print (1+2); // comentario en linea	punto y coma,	3	0 base 08.SwiftPL
/*	espacios,	error (porque	~_~~~~~~~~
print (1+5);	sangrado, nombres de	en la línea 5	
/* print (1+6); bloque	variables,	se acaba el	
comentado */	asignaciones,	comentario y	
print (1+7); // comentario en linea */ sigue	errores, etc.	continua con texto no	
print (1+8); // comentario en linea		texto no legal)	
var a = 1;		1	0 base 09.SwiftPL
print (a)			
let a = 1		3	0_base_10.SwiftPL
print (a+2);			
<pre>var a = 1; let b = a+1;</pre>		3	0_base_11.SwiftPL
print (a+b)		1	0 hago 12 confern
<pre>var a = 1; print(a); let b = A+1;</pre>		error (variable A no	0_base_12.SwiftPL
print (a+b)		declarada)	
var a = 2; print(a);		2	0 base 13.SwiftPL
let 1_mas_a = 1 + a;		error (var. no	_ <b>_</b>
print (a+b)		empieza por letra)	
var a = 2		9	0_base_14.SwiftPL
let b = a+1;			
a=a*3			
<pre>print (a+b) a = 1; print(a)</pre>		error	0 base 15.SwiftPL
b = a+1;		/* por a = 1; y b	~~~~_
print (a+b)		= a+1; porque hay	
		que usar let o	
		var*/	
let b = a+1;		error	0_base_16.SwiftPL
var a = 1;		/* por let b =	
print (a+b)		a+1; porque a está sin definir */	
	l	sin derinit ^/	

let implicitoDouble = 2.0 let explicitoDouble:Double = 3 print(implicitoInt) print(implicitoDouble) print(explicitoDouble) let implicitoDouble = 2.0 let explicitoDouble = 2.0 let explicitoDouble = 2.0 let explicitoDouble = 2.0 let explicitoDouble = 3 print(implicitoDouble) print(explicitoDouble) var a= 1; let b = 2; print (a+b) var a = 2 let b: Int; b=a*3 print (a+b) var a,b: Int a=1 b=a*3 print (a+b) var a,b: Int let b: Int; a=4 b=a*3 print (a+b) var a = 2 let b: Int; a=4 b=a*3 print (a+b) var a = 2 let b: Int; a=4 b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b)	Entrada	Descripción	Resultado de la ejecución	EJEMPLO
let explicitoDouble:Double = 3 print(implicitoInt) print(implicitoDouble) print(explicitoDouble) let implicitoDouble = 2.0 let explicitoDouble   Double = 3 print(implicitoDouble) print(explicitoDouble) print(explicitoDouble) print(explicitoDouble) print(explicitoDouble) print(explicitoDouble / implicitoDouble) var a= 1; let b = 2; print (a+b) var a = 2 let b: Int; b=a*3 print (a+b) var a,b: Int a=1 b=a*3 a=a*2 print (a+b) var a,b: Int let b: Int; b=a*3 print (a+b) var a,b: Int let b: Int; b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b)	var implicitoInt = 1	_	1	1 var 01.SwiftPL
print(implicitoInt) print(print(implicitoDouble) print(print(implicitoDouble) print(print(implicitoDouble) let implicitoDouble = 2.0 let explicitoDouble: Double = 3 print(implicitoDouble) print(print(print)	<pre>let implicitoDouble = 2.0</pre>		2.0	
print(implicitoDouble) print(explicitoDouble) let implicitoDouble = 2.0 let explicitoDouble: Double = 3 print(implicitoDouble) print(explicitoDouble) print(explicitoDouble = 3 print(explicitoDouble) print(e	<pre>let explicitoDouble:Double = 3</pre>		3.0	
print(explicitoDouble) let implicitoDouble = 2.0 let explicitoDouble = Double = 3 print(implicitoDouble) print(explicitoDouble) print(exp	<pre>print(implicitoInt)</pre>			
let implicitoDouble = 2.0 let explicitoDouble: Double = 3 print(implicitoDouble) print(explicitoDouble) / implicitoDouble) var a = 1; let b = 2; print (a+b) var a = 2 let b : Int; b=a*3 a=a*2 print (a+b) var a,b: Int let b : Int; b=a*3 b=a*3 print (a+b) var a = 2 let b : Int; b=a*3 a=a*2 print (a+b) var a,b: Int let b : Int; a=4 b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b)	<pre>print(implicitoDouble)</pre>			
let explicitoDouble: Double = 3 print(implicitoDouble) print(explicitoDouble / implicitoDouble) var a= 1; let b = 2; print (a+b) var a = 2 let b : Int; b=a*3 print (a+b) var a,b: Int let b : Int; let b : Int; b=a*3 print (a+b) var a,b: Int let b : Int; a=4 b=a*3 print (a+b) var a = 2 let b : Int; b=a*3 print (a+b) var a,b: Int let b : Int; a=4 b=a*3 print (a+b) var a = 2 let b : Int; b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b)	<pre>print(explicitoDouble)</pre>			
print(implicitoDouble) print(explicitoDouble / implicitoDouble) var a= 1; let b = 2; print (a+b) var a = 2 let b : Int; b=a*3 print (a+b) var a,b: Int let b : Int; b=a*3 print (a+b) var a,b: Int let b : Int; let b : Int; b=a*3 print (a+b) var a,b: Int let b : Int;	let implicitoDouble = 2.0		2.0	1 var 02.SwiftPL
print(explicitoDouble / implicitoDouble) var a = 1; let b = 2; print (a+b) var a = 2 let b : Int; b=a*3 print (a+b) var a,b: Int let b : Int; b=a*3 print (a+b) var a,b: Int let b : Int; b=a*3 print (a+b) var a,b: Int let b : Int; b=a*3 print (a+b) var a,b: Int let b : Int; b=a*3 print (a+b) var a,b: Int let b : Int; b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b)	let explicitoDouble: Double = 3		1.5	
<pre>var a= 1; let b = 2; print (a+b) var a = 2 let b : Int; b=a*3 print (a+b) var a,b: Int let b : Int; b=a*3 print (a+b) var a,b: Int let b : Int; b=a*3 print (a+b) var a = 2 let b : Int; b=a*3 print (a+b) var a = 2 let b : Int; b=a*3 print (a+b) var a = 2 let b : Int; b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b)</pre> <pre> var a = 1; b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b)</pre> <pre> error (hay que dejar, al menos, un espacio antes y después del igual)  1_var_03.SwiftPL  let var_04.SwiftPL  let var_abelian let var_</pre>	print(implicitoDouble)			
let b = 2; print (a+b)  var a = 2 let b : Int; b=a*3 print (a+b)  var a,b: Int a=1 b=a*3 a=a*2 print (a+b)  var a,b: Int let b : Int; b=a*3 print (a+b)  var a = 2 let b = a+1; b=a*3 print (a+b)  var a = 2 let b = a+1; b=a*3 print (a+b)   menos, un espacio antes y después del igual)  8  1_var_04.SwiftPL  beclaración de variables y tipos de datos  tipos de datos  a	<pre>print(explicitoDouble / implicitoDouble)</pre>			
print (a+b)  var a = 2 let b: Int; b=a*3 print (a+b)  var a,b: Int a=1 b=a*3 a=a*2 print (a+b)  var a,b: Int let b: Int; b=a*3 print (a+b)  var a = 2 let b: Int; b=a*3 print (a+b)  var a = 2 let b: Int; b=a*3 print (a+b)  var a = 2 let b: Int; b=a*3 print (a+b)  var a = 2 let b: a+1; b=a*3 print (a+b)	var a= 1;		error (hay que dejar, al	1 var 03.SwiftPL
<pre>var a = 2 let b : Int; b=a*3 print (a+b) var a,b: Int a=1 b=a*3 a=a*2 print (a+b) var a,b: Int let b : Int; a=4 b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b)</pre> <pre> Declaración de variables y tipos de datos  1_var_04.SwiftPL  beclaración de variables y tipos de datos  1_var_05.SwiftPL  error (no se puede redeclarar una variable o constante)  error (por b=a*3 porque no se puede cambiar una constante)  1_var_07.SwiftPL  1_var_07.SwiftPL  arror (por b=a*3 porque no se puede cambiar una constante)  1_var_07.SwiftPL  arror (por b=a*3 porque no se puede cambiar una constante)</pre>	let b =2;		menos, un espacio antes y	
let b: Int; b=a*3 print (a+b)  var a,b: Int a=1 b=a*3 a=a*2 print (a+b)  var a,b: Int let b: Int; a=4 b=a*3 print (a+b)  var a = 2 let b = a+1; b=a*3 print (a+b)  Declaración de variables y tipos de datos  5  1_var_05.SwiftPL  error (no se puede redeclarar una variable o constante)  error (por b=a*3 porque no se puede cambiar una constante)  1_var_07.SwiftPL	print (a+b)		después del igual)	
b=a*3 print (a+b)  var a,b: Int a=1 b=a*3 a=a*2 print (a+b)  var a,b: Int let b: Int; a=4 b=a*3 print (a+b)  var a = 2 let b = a+1; b=a*3 print (a+b)  Declaración de variables y tipos de datos   1_var_05.SwiftPL  a_1_var_05.SwiftPL  error (no se puede redeclarar una variable o constante)  a_1_var_06.SwiftPL  a_1_var_06.Swift	var a = 2		8	1 var 04.SwiftPL
print (a+b)  var a,b: Int a=1 b=a*3 a=a*2 print (a+b)  var a,b: Int let b: Int; a=4 b=a*3 print (a+b)  var a = 2 let b = a+1; b=a*3 print (a+b)  var a,b: Int error (no se puede redeclarar una variable o constante)  error (por b=a*3 porque no se puede cambiar una constante)  error (por b=a*3 porque no se puede cambiar una constante)	let b : Int;			
<pre>var a,b: Int a=1 b=a*3 a=a*2 print (a+b) var a,b: Int let b : Int; a=4 b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b)</pre> tipos de datos  1_var_05.SwiftPL  a_1_var_05.SwiftPL  a_1_var_05.SwiftPL  a_1_var_06.SwiftPL  a_1_var_06.	b=a*3	Declaración de		
a=1 b=a*3 a=a*2 print (a+b) var a,b: Int let b: Int; a=4 b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b)  var a = 2 let b = a+1; b=a*3 print (a+b)	print (a+b)			
b=a*3 a=a*2 print (a+b)  var a,b: Int let b: Int; a=4 b=a*3 print (a+b)  var a = 2 let b = a+1; b=a*3 print (a+b)  print (a+b)  a=4 b=a*3 print (a+b)  a=5 b=a*3 print (a+b)  a=6 constante)  error (no se puede redeclarar una variable o constante)  a=6 constante)  error (por b=a*3 porque no se puede cambiar una constante)	var a,b: Int	tipos de datos	5	1_var_05.SwiftPL
a=a*2 print (a+b)  var a,b: Int let b: Int; a=4 b=a*3 print (a+b)  var a = 2 let b = a+1; b=a*3 print (a+b)  var a = 2 let b = a+1; b=a*3 print (a+b)	a=1			
print (a+b)  var a,b: Int let b: Int; a=4 b=a*3 print (a+b)  var a = 2 let b = a+1; b=a*3 print (a+b)  var (a+b)  var a = 2 let b = a+1; b=a*3 print (a+b)	b=a*3			
<pre>var a,b: Int let b : Int; a=4 b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b)</pre> <pre>error (no se puede     redeclarar una variable o     constante)</pre> <pre>let b = a+1; b=a*3 print (a+b)</pre> <pre>error (por b=a*3 porque     no se puede cambiar una     constante)</pre>	a=a*2			
<pre>let b : Int; a=4 b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b)</pre> <pre>error (por b=a*3 porque no se puede cambiar una constante)</pre> <pre>let b = a+1;</pre> print (a+b)	print (a+b)			
a=4 b=a*3 print (a+b) var a = 2 let b = a+1; b=a*3 print (a+b)  constante)  error (por b=a*3 porque no se puede cambiar una constante)  print (a+b)	var a,b: Int		error (no se puede	1_var_06.SwiftPL
b=a*3 print (a+b)  var a = 2 let b = a+1; b=a*3 print (a+b)  error (por b=a*3 porque no se puede cambiar una constante)	let b : Int;		redeclarar una variable o	
print (a+b)  var a = 2  let b = a+1; b=a*3 print (a+b)  error (por b=a*3 porque no se puede cambiar una constante)  constante)	a=4		constante)	
<pre>var a = 2 let b = a+1; b=a*3 print (a+b)</pre> error (por b=a*3 porque no se puede cambiar una constante)  1_var_07.SwiftPL no se puede cambiar una constante)	b=a*3			
<pre>let b = a+1; b=a*3 print (a+b)</pre> no se puede cambiar una constante)	print (a+b)			
b=a*3 print (a+b) constante)	var a = 2		error (por b=a*3 porque	1_var_07.SwiftPL
print (a+b)	let $b = a+1;$		no se puede cambiar una	
	b=a*3		constante)	
	print (a+b)			
var x, y, z: Double error (x sin inicializar)   1_var_08.SwiftPL	var x, y, z: Double		error (x sin inicializar)	1_var_08.SwiftPL
print (x)	print (x)		,	

Entrada	Descripción	Resultado de la ejecución	EJEMPLO
var c = 1.0	•	9.0	2 oper 01.SwiftPL
var d = 2.3		11.3	
var e = 3.4 + 5.6		29.7	
var f = c * (d + e)			
var g = e * d + e			
<pre>print(e)</pre>			
<pre>print(f)</pre>			
<pre>print(g)</pre>			
<pre>let implicitoInt = 1</pre>		2.0	2 oper 02.SwiftPL
<pre>let implicitoDouble = 2.0</pre>		error (no se puede operar	
<pre>print(implicitoDouble)</pre>		una variable Int con una	
<pre>print(implicitoDouble / implicitoInt )</pre>		Double)	
let x = 3 * -4		error (el operador * no es	2 oper 03.SwiftPL
let $y = 3 - *4$		unario)	
<pre>print(x)</pre>			
let a = false		false	2 oper 04.SwiftPL
<pre>print(a)</pre>	Operaciones aritméticas y	true	
<pre>print(!a)</pre>	booleanas		
var a = true	Dooreanas	true	2 oper 05.SwiftPL
<pre>print(a)</pre>		true	
<pre>print(a     false)</pre>			
let a = true		true	2_oper_06.SwiftPL
<pre>print(a)</pre>		false	
var b = 3 < 2			
print(a && b)			
let a = true		true	2 oper 07.SwiftPL
<pre>print(a)</pre>		error (comparación ilegal)	
var b = 3 < 2 < 1		, , ,	
<pre>print(a &amp;&amp; b)</pre>			
var x, y, z: Double		true	2_oper_08.SwiftPL
x = 2; y = 4; z = 8			
var a: Bool ;			
a = (z / y) == x			
print (a)			

Entrada	Descripción	Resultado de la ejecución	EJEMPLO
var numbers = [3,9,27]		3	3_vect_01.SwiftPL
<pre>print (numbers[0])</pre>			
var numbers = [3,9,27]		[3, 9, 27]	3_vect_02.SwiftPL
<pre>print (numbers)</pre>			
var numbers = [3,9,27]		[3, 9, 18]	3_vect_03.SwiftPL
numbers[2] = 18			
print (numbers)			
var numbers = [3,9,27]		[3, 9, 27]	3_vect_04.SwiftPL
print (numbers)		error (fuera de rango)	
numbers[3] = 18	- 6: 1.4		
print (numbers)	Definición y		
var numbers = [3,9,81]	uso de arrays	[9, 9, 81]	3_vect_05.SwiftPL
<pre>numbers[0] = numbers[2] / numbers[1]</pre>			
print (numbers)			
<pre>var numbers: [Double] = [3,9,81]</pre>		[3.0, 9.0, 81.0]	3_vect_06.SwiftPL
print (numbers)			
var n = [3,9,27]		true	3_vect_07.SwiftPL
let ordenado =			
(n[0] <n[1])&&(n[1]<n[2])< td=""><td></td><td></td><td></td></n[1])&&(n[1]<n[2])<>			
print (ordenado)			
<pre>var reales: [Double] = [3,9,81]</pre>		error (no se pueden operar	3_vect_08.SwiftPL
var enteros = [5,7,11]		variables de distinto tipo)	
<pre>print (reales[0] * enteros[0])</pre>			

<pre>var enteros = [5,7,11] for numero in enteros {     print (numero) } var enteros = [5,7,11] for numero in enteros {     print (numero) }  var enteros = [5,7,11] for numero  var enteros = [5,7,11] for index in 02 {     print(enteros[index]) } let base, exponente: Int     base = 2 exponente = 4 var resultado = 1 for _ in 1exponente {     resultado = resultado * base } print(resultado) var a = [2,4,6] var b = [2,3]</pre> 4_for_01.SwiftPL  5 7 11 error (la variable numero no está fuera del ámbito del for)  5 7 11 let or 11 error (la variable numero no está fuera del ámbito del for)  5 7 11 16 4_for_03.SwiftPL  6 4_for_04.SwiftPL  7 11 16 4_for_03.SwiftPL  7 11 16 4_for_03.SwiftPL  7 11 16 4_for_03.SwiftPL  7 11 16 4_for_03.SwiftPL  7 11 16 16 4_for_05.SwiftPL  7 11 17 18 19 19 10 10 10 11 11 11 11 11 11 11 11 11 11		1	1	
for numero in enteros {     print (numero) } var enteros = [5,7,11] for numero in enteros {     print (numero) } print (numero) var enteros = [5,7,11] for index in 02 {     print(enteros[index]) } let base, exponente: Int base = 2 exponente = 4 var resultado = 1 for _ in 1exponente {     resultado = resultado * base } print(resultado) var a = [2,4,6] var b = [2,3] for i in a {     for j in b{	Entrada	Descripción	Resultado de la ejecución	EJEMPLO
<pre>{     print (numero) } var enteros = [5,7,11] for numero in enteros {     print (numero) } print (numero) var enteros = [5,7,11] for index in 02 {     print(enteros[index]) } let base, exponente: Int base = 2 exponente = 4 var resultado = 1 for _ in 1exponente {     resultado = resultado * base } print(resultado) var a = [2,4,6] var b = [2,3] for i in a {     for j in b{</pre>			5	4_for_01.SwiftPL
<pre>print (numero) } var enteros = [5,7,11] for numero in enteros {     print (numero) } print (numero) var enteros = [5,7,11] for index in 02 {     print(enteros[index]) } let base, exponente: Int base = 2 exponente = 4 var resultado = 1 for _ in 1exponente {     resultado = resultado * base } print(resultado) var a = [2,4,6] var b = [2,3] for i in a {     for j in b{</pre> // A_for_02.SwiftPL  // Ill error (la variable numero no está fuera del ámbito del for)  // Just de operaciones con bucle for  // Ill error (la variable numero no está fuera del ámbito del for)  // Ill // A_for_03.SwiftPL  // Ill //	for numero in enteros		7	
<pre> yar enteros = [5,7,11] for numero in enteros {     print (numero) } var enteros = [5,7,11] for index in 02 {     print(enteros[index]) } let base, exponente: Int base = 2 exponente = 4 var resultado = 1 for _ in 1exponente {     resultado = resultado * base } print(resultado) var a = [2,4,6] var b = [2,3] for i in a {     for j in b{</pre>	[{		11	
for numero in enteros {     print (numero) }  print (numero) var enteros = [5,7,11] for index in 02 {     print(enteros[index]) } let base, exponente: Int base = 2 exponente = 4 var resultado = 1 for _ in 1exponente {     resultado = resultado * base } print(resultado) var a = [2,4,6] var b = [2,3] for i in a {     for j in b{	print (numero)			
for numero in enteros {     print (numero) }  print (numero) var enteros = [5,7,11] for index in 02 {     print(enteros[index]) } let base, exponente: Int base = 2 exponente = 4 var resultado = 1 for _ in 1exponente {     resultado = resultado * base } print(resultado) var a = [2,4,6] var b = [2,3] for i in a {     for j in b{	}			
for numero in enteros {     print (numero) }  print (numero) var enteros = [5,7,11] for index in 02 {     print(enteros[index]) }  let base, exponente: Int base = 2 exponente = 4 var resultado = 1 for _ in 1exponente {     resultado = resultado * base }  print(resultado) var a = [2,4,6] var b = [2,3] for i in a {     for j in b{   7 11 error (la variable numero no está fuera del ámbito del for)  5 7 11 error (la variable numero no está fuera del ámbito del for)  5 7 11 error (la variable numero no está fuera del ámbito del for)  5 7 11 error (la variable numero no está fuera del ámbito del for)  5 7 11 error (la variable numero no está fuera del ámbito del for)  5 7 11 6 7 11 error (la variable numero no está fuera del ámbito del for)  5 7 11 6 7 7 11 6 7 7 11 7 7 11 6 7 7 11 7 7 11 6 7 7 11 7 7 11 6 7 7 11 7 7 11 6 7 7 11 6 7 7 11 6 7 7 11 6 7 7 11 6 7 7 11 6 7 7 11 6 7 7 11 6 7 7 11 6 7 7 11 6 7 7 11 7 7 11 6 7 7 11 6 7 7 11 6 7 7 11 6 7 11 6 7 7 11 6 7 7 11 6 7 11 6 7 7 11 6 7 7 11 6 7 11 6 7 7 11 7 11 6 7 7 11 6 7 7 11 6 7 11 6 7 7 11 6 7 11 6 7 7 11 6 7 7 11 7 11 6 7 7 11 6 7 7 11 6 7 7 11 6 7 11 6 7 7 11 11 6 7 7 11 6 7 11 6 7 11 6 7 11 6 7 7 11 11 6 7 7 11 6 7 7 11 6 7 7 11 6 7 7 11 6 7 7 11 6 7 7 11 11 6 7 7 11 11 6 7 7 11 11 6 7 7 11 11 6 7 7 7 11 11 6 7 7 11 11 6 7 7 11 11 6 7 7 11 11 6 7 7 11 11 6 7 7 11 11 6 7 7 11 11 6 7 7 11 11 6 7 7 11 11 6 7 7 11 11 6 7 7 11 11 6 7 7 11 11 6 7 7 11 11 11 11 11 11 11 11 11 11 11 11	var enteros = [5,7,11]		5	4 for 02.SwiftPL
<pre>print (numero) print (numero) print (numero) var enteros = [5,7,11] for index in 02 {     print(enteros[index]) } let base, exponente: Int base = 2 exponente = 4 var resultado = 1 for _ in 1exponente {     resultado = resultado * base } print(resultado) var a = [2,4,6] var b = [2,3] for i in a {     for j in b{</pre> Uso de operaciones con bucle for  11 16 4_for_03.SwiftPL  4_for_03.SwiftPL  6 7 11 16 4_for_04.SwiftPL  4_for_05.SwiftPL  4_for_05.SwiftPL  6 7  4_for_05.SwiftPL  6 7  4_for_05.SwiftPL  6 7  4_for_05.SwiftPL  6 7  6 7  6 7  7  11 11 11 11 12 16 16 17  4_for_05.SwiftPL  6 7  4_for_05.SwiftPL  7  7 11 11 11 12 12 13 14 15 16 16 17  4_for_05.SwiftPL  6 7  7 11 11 11 11 11 11 11 11 11 11 11 1	for numero in enteros		7	
<pre>print (numero) var enteros = [5,7,11] for index in 02 {     print(enteros[index]) } let base, exponente: Int base = 2 exponente = 4 var resultado = 1 for _ in 1exponente {     resultado = resultado * base } print(resultado) var a = [2,4,6] var b = [2,3] for i in a {     for j in b{</pre> numero no está fuera del ámbito del for)  5 7 11 16 16 4_for_03.SwiftPL  6 7  16 4_for_04.SwiftPL  6 7  4_for_05.SwiftPL  6 7  4_for_05.SwiftPL  6 7  4_for_05.SwiftPL  6 7  6 7  7  7  7  7  8 4  4  4  4  4  4  4  6  7  7  7  7  7  7  8  4  5  6  6  6  7  7  7  7  7  7  7  7  7  8  4  4  4  4  4  4  4  6  7  6  7  7  7  7  7  7  7  8  4  4  6  6  7  6  7  7  7  7  7  7  7  7  7	{		11	
<pre>print (numero) var enteros = [5,7,11] for index in 02 {     print(enteros[index]) } let base, exponente: Int base = 2 exponente = 4 var resultado = 1 for _ in 1exponente {     resultado = resultado * base } print(resultado) var a = [2,4,6] var b = [2,3] for i in a {     for j in b{</pre> Uso de operaciones con bucle for  11  16  4_for_03.SwiftPL  4_for_04.SwiftPL  5  4  4_for_05.SwiftPL  5  6  6  7  4  4_for_05.SwiftPL  5  6  7  11  7  12  4_for_04.SwiftPL  5  6  7  4  4_for_05.SwiftPL  6  7  4  4_for_05.SwiftPL  6  7  6  6  7  7  11  7  12  7  12  7  13  7  14  4  4  4  4  4  4  4  4  4  4  4  4	print (numero)		error (la variable	
<pre>var enteros = [5,7,11] for index in 02 {     print(enteros[index]) } let base, exponente: Int base = 2 exponente = 4 var resultado = 1 for _ in 1exponente {     resultado = resultado * base } print(resultado) var a = [2,4,6] var b = [2,3] for i in a {     for j in b{</pre> Uso de operaciones con bucle for  11  16  4_for_03.SwiftPL  4_for_04.SwiftPL  6  7  11  4_for_05.SwiftPL  5  6  6  7  12  4_for_05.SwiftPL  5  6  6  7  12  4_for_05.SwiftPL  6  7  13  4_for_05.SwiftPL  6  7  14  4_for_05.SwiftPL  6  7  15  6  6  7  16  7  17  17  18  4_for_05.SwiftPL  7  19  4_for_05.SwiftPL  7  10  4_for_05.SwiftPL  7  11  10  11  10  11  10  11  10  11  10  11  10  11  10  11  10  11  10  11  10  11  11  11  12  14  15  16  7  17  11  10  10  11  11  10  11  11	<b> </b> }		numero no está fuera	
for index in 02 {     print(enteros[index]) }  let base, exponente: Int base = 2 exponente = 4 var resultado = 1 for _ in 1exponente {     resultado = resultado * base } print(resultado) var a = [2,4,6] var b = [2,3] for i in a {     for j in b{  Uso de operaciones con bucle for  11  16  4_for_04.SwiftPL  4_for_04.SwiftPL  4_for_05.SwiftPL  5_6 6_7	print (numero)		del ámbito del for)	
<pre>for index in 02 {     print(enteros[index]) } let base, exponente: Int base = 2 exponente = 4 var resultado = 1 for _ in 1exponente {     resultado = resultado * base } print(resultado) var a = [2,4,6] var b = [2,3] for i in a {     for j in b{</pre> Uso de operaciones con bucle for  16  4_for_04.SwiftPL  4_for_05.SwiftPL  5_6 6_7  4_for_05.SwiftPL	var enteros = [5,7,11]		5	4 for 03.SwiftPL
print(enteros[index])  let base, exponente: Int  base = 2 exponente = 4 var resultado = 1 for _ in 1exponente {     resultado = resultado * base }  print(resultado) var a = [2,4,6] var b = [2,3] for i in a {     for j in b{  operaciones con bucle for  16  4_for_04.SwiftPL  4_for_05.SwiftPL  4_for_05.SwiftPL	for index in 02 {	,	7	
<pre>con bucle for let base, exponente: Int base = 2 exponente = 4 var resultado = 1 for _ in 1exponente {     resultado = resultado * base } print(resultado) var a = [2,4,6] var b = [2,3] for i in a {     for j in b{</pre> <pre> con bucle for  4_for_04.SwiftPL  4_for_05.SwiftPL  4_for_05.SwiftPL  6_ 7</pre>	<pre>print(enteros[index])</pre>		11	
<pre>let base, exponente: Int base = 2 exponente = 4 var resultado = 1 for _ in 1exponente {     resultado = resultado * base } print(resultado) var a = [2,4,6] var b = [2,3] for i in a {     for j in b{</pre> 16 4_for_04.SwiftPL  4_for_04.SwiftPL  4_for_05.SwiftPL  5 6 7	}			
<pre>base = 2 exponente = 4 var resultado = 1 for _ in 1exponente {     resultado = resultado * base } print(resultado) var a = [2,4,6] var b = [2,3] for i in a {     for j in b{</pre> 4 4 4_for_05.SwiftPL 5 6 7	let base, exponente: Int		16	4 for 04.SwiftPL
<pre>var resultado = 1 for _ in 1exponente {     resultado = resultado * base } print(resultado) var a = [2,4,6] var b = [2,3] for i in a {     for j in b{</pre>	base = 2	101		
<pre>for _ in 1exponente {     resultado = resultado * base } print(resultado) var a = [2,4,6] var b = [2,3] for i in a {     for j in b{</pre> 4	exponente = 4			
resultado = resultado * base } print(resultado) var a = [2,4,6] var b = [2,3] for i in a {     for j in b{	var resultado = 1			
<pre> } print(resultado)  var a = [2,4,6] var b = [2,3] for i in a {     for j in b{</pre>	for _ in 1exponente {			
<pre>var a = [2,4,6] var b = [2,3] for i in a {     for j in b{</pre>	resultado = resultado * base			
<pre>var a = [2,4,6] var b = [2,3] for i in a {     for j in b{</pre>	<b> </b> }			
<pre>var b = [2,3] for i in a {     for j in b{</pre> <pre>5 7</pre>	<pre>print(resultado)</pre>			
<pre>var b = [2,3] for i in a {     for j in b{</pre> <pre>5 7</pre>	var a = [2,4,6]		4	4 for 05.SwiftPL
for j in b{	var b = [2,3]		5	
	for i in a {		6	
<pre>print(i+j)</pre>	for j in b{		7	
9	<pre>print(i+j)</pre>		8	
h l	}		9	
[/	[}			