import Foundation

//Lenguaje de programación SWIFT

//Esto es un comentario

//Puedes poner en practica a lo largo del archivo lo que quieras, para eso son lo playgrounds. Haz y deshaz tantas veces como quieras

/\*

Comentar más de

una línea utilizamos: /\* \*/

En este archivo debes recordar que se pueden utilizar en repetidas ocaciones, incluso

segmentos de código

\*/

//Variables y constantes

let MiApellido : String = "Minor"

//Esto es una declaración de una constante, un valor que no va a cambiar durante la ejecución del programa

let miNombre : String = "Andrés"

//Esto es una declaración de una variable, un valor que va a cambiar, esperamos que cambie en nuestro programa, podría decirse también que son espacios en memoria que pueden tomar más de un valor durante la ejecución de un programa.

//Entiendase programa como una pequeña aplicación en nuestro celular o en una computadora

var miEdad : Int = 23

var miDireccion : String = "nombreDeCalle"

var miNumeroDeCasa : Int = 19

var miNumeroInterior : Int = 4

// Como ven las constantes se declara y se definen de la misma forma

let primeraCarrera : String = "Informática"

let miNacionalidad : String = "Mexicana"

/\*

Toda variable se compone de tres elementos. Sean constantes o mutables, que cambian su valor. Estos elementos son:

Modificador de acceso:

Es una palabra reservada que me indica cómo se va a reservar el espacio en memoria principal durante la ejecución de un programa.

Identificador:

El nombre del valor que voy a guardar, el nombre de ese espacio, cómo reconozco ese espacio en memoria. Un identificador es único e impermutable, por lo menos durante la ejecución de mi programa. Podemos tener muchas variables dentro de un programa y un identificador hace justo como su nombre lo dice, darle identidad a una variable, de esa forma al agregar o modificar esos espacios en memoria el procesador sabrá qué espacios en memoria corresponden a cuál variable

Tipo de dato:

El tipo de dato es el distintivo que determina cómo se va a guardar y cómo se va a acceder a los valores de diferentes variables, qué operaciones podemos realizar o se puede realizar con esos tipos de datos y cómo realizarlas

Operador de asignación:

Es el símbolo que indica que se va a asignar un valor

Valor literal:

El valor que quieres que contenga esa variable. El valor, literalmente, de algo.

\*/

//Modificador | identificador | : | Tipo de dato | Operador de asignación | y valor literal

var minutos : Int = 60

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* OPERACIONES ARITMÉTICAS \*\*\*\*\*\*\*\*\*\*\*+\*\*\*\*\*\*\*\*\*/

//Para realizar operaciones aritmeticas utilizamos operadores aritmeticos: + - \* / %

//Se les conoce como operadores binarios pues necesitan dos

//Suma

4 + 4

//Resta

4 - 4

//Division

4 / 2

//Multiplicación

4 \* 4

//Módulos. Calculan el residuo de una división: <Dividendo> % <Divisor>

5 % 2

4 % 2

81 % 4

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Entonces, para esto utilizamos los operadores

Si queremos realizar operaciones entre variables o entre constantes

lo hacemos de la siguiente forma

\*/

//El tipo de dato es Int, que significa que vamos a trabajar con números enteros

var operando1 : Int = 5

//Los valores pueden repetirse, siempre y cuando esten almacenados bajo un identificador diferente no podemos utilizar dos identificadores iguales, porque como se menciona arriba, deben ser únicos

var operando2 : Int = 5

var operando3 : Int = -10

operando1 + operando2

operando3 / operando2

//Notese que puedes hacer operaciones con la misma variable

(operando1 + operando1 \* operando1) / operando1

//Para realizar operaciones diferentes podemos utilizar () para darle jerarquias a las operaciones

operando3 + 5 \* ( 4 - (1006 / (3 + (4 + 8 \* 5) + 1)))

//La jerarquia de las operaciones es igual que en Álgebra y aritmetica y todas las metemáticas tal vez

// pero siempre va a respetar el orden, la secuencia en la que las escribimos

1 + 2 - 1 \* 4 / 5

16 / 4 \* 5

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Tipos de Datos \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

var entero : Int = 20

var numeroDecimal : Double = 5.5

var otroNumeroDecimal : Float = 5.5//Es igual que el double, ocupa más espacio en memoria principal

var caracter : Character = "\u{00AE}"

var cadena : String = "Locuaz e impertinente"

var muchasCadenas : String = """

Recuerda poner

tres dobles comillas,

dar un salto de línea

y comenzar a escribir.

...

\(entero)

\(numeroDecimal)

\(cadena)

...

...

Al acabar cierra el texto dando un salto de línea final

con tres dobles comillas en la última línea

"""

//print es una funcion del lenguaje que "imprime" un valor dado en el dispositivo de salida standard, en esta caso la consola del monitor.

print(muchasCadenas)

var cadenaCOnDoblesComillas = "\"Quiero que se impriman las dobles comillas\""

//Siempre y cuando tenga cadenas como tipos de dato(String), puedo utilizar el operador + para concatenar

var cadena1 :String = "Hola"

var espacio : String = " "

var cadena2 : String = "Mundo"

var sentencia : String = cadena1 + espacio + cadena2

//Tipo de dato booleano, es una bandera que indica si una condición es falsa o verdadera

var booleano : Bool = true //false es el otro valor.

/\*\*\*\* Operadores aritmeticos y el operador de signación

También podemos realizar lo siguiente:

Tomar el valor de una variable y sumarle otro valor.

Sólo lo podemos hacer en el caso de que sean del mismo tipo

\*/

entero += 5

entero -= 2

entero \*= 10

entero /= 4

cadena1 += "\nMundo"// Notese que se utiliza \n para realizar un salto de línea y las cadenas sólo se pueden concatenar, no las podemos restar, ni multplicar jeje

/\*

Esto es un casteo de tipos para que podamos hacer operaciones entre diferentes valores con

con diferentes tipos de datos \*/

/\*\* SINTAXIS:

modificador + identificador + : + tipoDeDato + operador asignacion + variable operando + tipoDeDato al que lo quiero convertir seguido de la variable operando que quiero convertir entre parentesis\*/

var suma : Double = numeroDecimal + Double(entero)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Funciones

Las funciones son segmentos de un programa, tareas específicas que se definen para ser empleadas a lo largo de un programa tantas veces como queramos, y permiten dividir un programa en ese conjunto de tareas que realiza para trabajarlas por separado, se trata de la descomposición de un problema a resolver para estructurarlo de mejor manera y sea también más entendible, para ti y para las computadoras.

Hay tres tipos de funciones:

Funciones sin parámetros

Funciones con parámetros

Funciones con parámetros que devuelven un valor( print() es un ejemplo!!!)

\*/

//Funciones sin parámetros

var operandoDouble1 : Double = 1.0

var operandoDouble2 : Double = 2.0

func sumar(){

let suma : Double = operandoDouble1 + operandoDouble2

print(suma)

}

func multiplicar(){

let multiplicar : Double = operandoDouble1 \* operandoDouble2

print(multiplicar)

}

sumar()

multiplicar()

//Funciones con parámetros

func restar(\_ operando1 : Int,\_ operando2 : Int){

let resta : Int = operando1 - operando2

print(resta)

}

func multplicar(\_ operando1 : Int,\_ operando2 : Int){

let multiplicacion : Int = operando1 \* operando2

print(multiplicacion)

}

//Funciones con parámetros que devuelven un valor, notese al finalizar la declaración de parámetros cómo indicamos qué valor se va a devolver con: -> TipoDeDato

func funcionSumar(\_ operando1 : Int,\_ operando2 : Int)->Int{

let suma : Int = operando1 + operando2

return suma

}

func funcionRestar(\_ operando1 : Int,\_ operando2 : Int)->Int{

let resta : Int = operando1 - operando2

return resta

}

func funcionMultplicar(\_ operando1 : Int,\_ operando2 : Int)->Int{

let multiplicacion : Int = operando1 \* operando2

return multiplicacion

}

func funcionDividir(\_ operando1 : Int,\_ operando2 : Int)->Int{

let division : Int = operando1 / operando2

return division

}

sumar()

funcionSumar(3, 5)

restar(6, 7)

funcionDividir(40, 5)

multplicar(50, 4)

//Nuevamente hacemos un casteo de tipos para que la expresion sea como Int

funcionSumar(Int(operando1),Int(operando2))

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Estructuras de control

Las estructuras de control nos permiten como su nombre lo dice controlar el flujo de nuestro programa

bifurcar este flujo, crear flujos alternos dentro de nuestro programa

De seleccion

if-else

switch-case

Iterativas, ciclos, bucles

for

for each

while

do-while

\*/

//Dale play aqui para que recuerdes qué valor tiene el booleano

booleano

entero = 30

//para tomar estas deciciones vamos a comenzar a ver que existen otros operadores, se llaman operadores relacionales o de comparación: en este caso ==

if(booleano == true){

print("Verdadero")

}else{

print("Falso")

}

/\*Trduccion de la sentencia anterior:

"Si el valor de booleano es igual a "true" entonces

ejecuta la funcion print para la cadena "Verdadero"

De lo contrario

ejecuta la funcion print para la cadena "Falso"

"

\*/

if(entero >= 10){

booleano = false

}

print(entero, booleano)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Operadores relacionales ó de comparación

!= Diferente de

== Igual que

<

>

>=

<=

Operadores Lógicos

&& AND Debe cumplir ambas condiciones

|| OR Puede cumplir por lo menos una de las dos condiciones

! NOT, un valor opuesto o complemento en valores booleanos

NAND

NOR

XOR

NXOR

\*/

if(operando1 == operando2){

print("Son igualitos")

}

if(operando1 != operando2){

print("No son iguales")

}

if(operando1 <= operando2){

print("\(operando2) es menor que \(operando1)")

}

if(operando1 >= operando2){

print("\(operando1) es mayor que \(operando2)")

}

var edadActual : Double = 23.00

var laEdadAlaQuemeCaso : Double = 30.00

if( edadActual < laEdadAlaQuemeCaso){

print("\nAún no cumples \(Int(laEdadAlaQuemeCaso))\nNo te casas suertudo\n")

}else{

print("TeCasas!!!")

}

if !booleano{

print("La negación de \(booleano) es \(!booleano)\n\n")

}