IV Swift Full Stack Bootcamp 2024
SWIFT 5.9
Ejercicios Nivel Intermedio / Repaso
Enunciados

Ejercicio 1:
¿Qué distintos tipos de datos existen en Swift y cuál es su función? Crea dos datos prototipo de cada tipo de dato en Swift para ejemplificar su función principal.

Ejercicio 2:
Crea 4 datos distintos de distintos tipos de datos numéricos y realiza 10 operaciones aritméticas con ellos: suma, resta, multiplicación, división y resto de división. Combina varias operaciones en una sola.

Ejercicio 3:
Realiza un algoritmo que sea capaz de discernir si un número es divisible por otro, en distintos tipos de datos numéricos, con o sin precisión decimal.

Ejercicio 4:
Crea dos tipos distintos de bucles que sean capaces de mostrar todos los números enteros entre 1 y 20. Usando **for** y **while**.

Ejercicio 5:
Crea un *array* de números enteros y calcula su resultado sumando los valores impares y multiplicando por sus valores pares.

Ejercicio 6:
Crea un diccionario que contenga el nombre y la edad de 10 personas. Luego, obtén el dato de cada uno de ellos y busca la persona que tiene más edad y la que menos de todo el diccionario.

Ejercicio 7:
Crea una enumeración con los días de la semana y crea un flujo de tipo *switch* que de un mensaje que defina cada día de la semana o si estamos en fin de semana.

Ejercicio 8:
Crea una clase que abstraiga el concepto de **Personas** y recoja sus datos de nombre, edad y dirección. Crea 5 personas distintas.

Ejercicio 9:
Crea un *struct* de un **Rectángulo**, que tendrá las propiedades de **ancho** y **alto** del mismo. Crea varios rectángulos y construye una función que te permita saber cuál es el más grande y el más pequeño según su área, que obtenemos multiplicando ancho por alto.

Ejercicio 10:
Crea una clase llamada **Libro** con las propiedades **título, autor** y **año.** Crea 5 libros. Ahora con esta clase haz lo siguiente:

- Crea un método información que muestre en una sola cadena el título, autor y año de publicación del mismo.
- Crea una función que reciba dos libros y diga si son del mismo autor.
- Crea una función que reciba dos libros y diga si son del mismo año.

Ejercicio 11:
Crea una clase hija de **Libro** que será **Libro Físico** y **Libro Electrónico.** En el primero registra el número de páginas del libro y en el segundo la **URL** donde puede comprarse en formato electrónico. Crea 4 subclases, 2 de libro físico y 2 de libro electrónico.

Ejercicio 12:
Crea una estructura llamada **Punto** con las propiedades **x** e **y.** Agrega un método a la estructura llamado **mover** que reciba dos parámetros (**x** e **y**) y actualice las propiedades de **x** e **y** con los nuevos valores. Luego, crea un objeto de esta estructura e inicializa las propiedades con valores y llama al método **mover** para cambiar la posición del punto.

Ejercicio 13:
Crea una clase llamada **CuentaBancaria** con propiedades como **titular** y **saldo.** Agrega métodos a la clase llamados **depositar** y **retirar** que reciban un parámetro **cantidad** y actualicen el **saldo** de la cuenta en consecuencia. Luego, crea un objeto de esta clase e inicializa las propiedades con **valores** y llama a los métodos **depositar** y **retirar** para cambiar el saldo de la cuenta. Comprueba y prueba que no pueda retirarse más dinero del que hay en la cuenta.

Ejercicio 14
Agrega un método a la clase **Persona** del ejercicio 8 llamado **saludar** que muestre un mensaje de saludo personalizado en la consola, utilizando el nombre de la persona. Luego, crea un objeto de la clase **Persona** e inicializa las propiedades con valores y llama al método **saludar.**

Ejercicio 15
Agrega un método a la clase **Libro** del ejercicio 10 llamado **prestar** que reciba un parámetro con el tipo **Persona** del ejercicio 8 y muestre un mensaje en la consola indicando que el libro ha sido prestado a la persona con el nombre de dicha persona. Luego, crea un objeto de la clase **Libro** e inicializa las propiedades con valores y llama al método prestar para prestar el libro a una **Persona.**

Ejercicio 16
Agrega un método a la estructura **Punto** del ejercicio 12 llamado **distancia** que reciba un parámetro **otroPunto** de tipo **Punto** y calcule y devuelva la distancia entre el punto actual y el punto recibido como parámetro. Luego, crea dos objetos de la estructura **Punto** e inicializa las propiedades con valores y llama al método distancia para calcular y mostrar la distancia entre los dos puntos en la consola.

Ejercicio 17
Agrega un método a la clase **CuentaBancaria** del ejercicio 13 llamado **info** que muestre el titular de la cuenta y el saldo actual en la consola**.** Luego**,** crea un objeto de la clase **CuentaBancaria** e inicializa las propiedades con valores y llama al método **info** para mostrar la información de la cuenta en la consola**.** Después**,** llama al método depositar para agregar una cantidad a la cuenta y llama nuevamente al método **info** para mostrar el saldo actualizado en la consola**.**

Ejercicio 18
Agrega un método a la clase **Persona** llamado **birthdate** que aumente la edad de la persona en 1**.** Luego**,** crea un objeto de la clase **Persona** e inicializa las propiedades con valores y llama al método **birthdate.**

Ejercicio 19
Crea una propiedad calculada para calcular el área del rectángulo del ejercicio 9**.** Luego crea una función separada de este *struct* llamada **calcularAreaRectangulos** que tome como parámetros un *array* de **Rectángulo** y calcule el área total de todos los rectángulos**.** Luego**,** crea un *array* de objetos **Rectángulos** e inicializa las propiedades con valores y llama a la función **calcularAreaRectangulos** para calcular y mostrar el área total en la consola**.** Realiza la implementación mediante programación estructurada con bucles **for** pero también por programación funcional**.**

Ejercicio 20
Crea una función llamada **prestarLibro** para que tome como parámetros un objeto de la clase **Libro** de los ejercicios 10 y 15 y una persona de los ejercicios 14**,** 15 y 18 y muestre un mensaje en la consola indicando que el libro ha sido prestado a la persona con el nombre recibido como parámetro**.** Luego**,** crea un objeto de la clase **Libro** e inicializa las propiedades con valores y llama a la función **prestarLibro** para prestar el libro a una persona**.**

Ejercicio 21
Crea una función llamada **filtrarPuntos** que tome como parámetros un *array* de objetos **Punto** de los ejercicios 12 y 16 y una distancia límite**.** Devuelve un *array c*on los puntos que se encuentran dentro de la distancia límite**.** Luego**,** crea un *array* de objetos **Punto** e inicializa las propiedades con valores y llama a la función **filtrarPuntos** para obtener un *array* con los puntos dentro de la distancia **límite.** Realiza esta función mediante programación estructurada y con programación funcional**.**

Ejercicio 22
Agrega un método a la clase **CuentaBancaria** de los ejercicios 13 y 17 llamado **transferir** que reciba dos parámetros (**otraCuenta** y **cantidad**) y transfiera la cantidad indicada de la cuenta actual a la cuenta indicada en el parámetro **otraCuenta.** Controla que haya saldo suficiente para hacer esta operación**.** Luego**,** crea dos objetos de la clase **CuentaBancaria** e inicializa las propiedades con valores y llama al método **transferir** para transferir una cantidad de una cuenta a otra**.**

Ejercicio 23:
Crea un *array c*on 50 números enteros aleatorios. A partir del mismo, crea una cadena de respuesta que cuando encuentre que alguno de ellos es un número primo lo incluya. El resultado ha de ser una cadena similar a: "*Los números primos aleatorios de esta lista son x, x, x, x, x*".

Ejercicio 24:
Realiza una función que analice un texto y devuelva el número de palabras del mismo, su longitud completa, y qué longitud tiene cada palabra que aparezca en dicho texto en un listado.

Ejercicio 25:
Crea una fábrica de chocolate con clases. La clase principal hará chocolatinas de n onzas por tableta (n es aleatorio). Las tabletas pueden ser de chocolate negro, con leche o blanco. En cada tableta además podría venir un billete dorado para visitar la fábrica, pero en una probabilidad muy baja.

Ejercicio 26:
Vamos a crear un colegio donde crearemos la estructura de datos necesarios usando *structs* en vez de clases.
- S*truct* para los **Profesores**, con nombre y edad.
- S*truct* para las **Asignaturas**, con nombre de asignatura y cursos donde se imparte.
- S*truct* para los **Alumnos**, con nombre, edad y curso.
- S*truct* para los **Cursos** con el tutor (profesor) y los alumnos.
- S*truct* del **Colegio** con los distintos cursos del mismo.

Queremos poder extraer listados y/o conteo de las asignaturas que imparte cada profesor, cuántos alumnos tiene cada uno y cuántos alumnos distintos hay en el colegio que al menos estén en una asignatura.

Para probar que funciona crea una serie de datos de prueba.

Ejercicio 27:
Vamos a crear la estructura de un videojuego.

- **Personaje** tiene un valor de vida y un nombre. Es capaz de morir cuando se queda sin vida y de saludar diciendo su nombre.
- **Héroe** hereda de **Personaje.** Tiene valor de fuerza de ataque y es capaz de atacar a un **Enemigo** que recibe como parámetro, reduciendo su vida en un ataque igual a un número entre 0 y la mitad de su fuerza.
- **Mago** hereda de **Héroe.** Tiene un valor de magia que suma la mitad de su valor al daño ocasionado en un ataque y reduce en 1/4 el daño infligido cuando es atacado por un **Enemigo.**
- **Guerrera** hereda de **Héroe.** Tiene una espada que en su inicialización tendrá un valor de ataque que será un número aleatorio de 0 a la mitad de su fuerza. Dicha espada inflige un daño extra a los enemigos igual al total de dicha fuerza.
- **Enemigo** hereda de **Personaje.** Tiene valor de fuerza de ataque y puede tener tres tipos de armas, la cual se elegirá en su inicialización. Puede atacar a un héroe, recibido como parámetro. El arma **Hacha** hará un daño de 10 sobre el daño del ataque, el arma **Sable** lo hará de 5 y el arma **Cadena** lo hará de 2. También puede ser que el **Enemigo** se cree sin arma y entonces no aumente su fuerza en el ataque.

Crear todos los inicializadores, métodos y propiedades necesarios, así como la capacidad de atacar y morir de cada personaje susceptible de ello, además del resto de comportamientos indicados. Usa enumeraciones donde sea más práctico para representar datos.

Ejercicio 28:
Enviada a una función un *array* de números enteros, devuelve un diccionario en cuya clave esté cada número único en dicho array y en el valor las veces que aparece consecutivamente dentro del *array* enviado.

Ejercicio 29:
Enviada una frase a una función, descomponga esta en sus distintas palabras. Una vez hecho, devuelva un diccionario con las palabras como clave y en el valor, el número de veces que aparece cada palabra en la frase.

Ejercicio 30:
Crea una función que convierta números enteros en su versión en números romanos, partiendo de sus equivalencias:
1 -> I, V -> 5, X -> 10, L -> 50, C -> 100, D -> 500, M -> 1000

Recuerda que los números antes a la mitad, como el 4, se representan con una resta: IV (5V - 1I); CM (1000M - 100C).