

TECNM campus Nuevo Laredo.

Ing. Sistemas computacionales.

Desarrollo de IOS.

Catedrático

Ing. Humberto Peña Valle M.T.I.

Alumno:

Alejandro Díaz Medrano :19100168 Fecha entrega:02/03/23

Índice.

[1.1 Investigación. 5](#_Toc129269535)

[1.1.1 Macintosh Operating System 5](#_Toc129269536)

[Classic Mac OS. 5](#_Toc129269537)

[OS X 5](#_Toc129269538)

[OS X Server 5](#_Toc129269539)

[Tabla comparativa 5](#_Toc129269540)

[1.1.2.Xcode 7](#_Toc129269541)

[Versiones: 7](#_Toc129269542)

[nib, xib, storyboard 8](#_Toc129269543)

[AppStore (incluir requisitos para publicar Apps) 8](#_Toc129269544)

[Estructura de una App empaquetada 9](#_Toc129269545)

[Modelo-Vista-Controlador 10](#_Toc129269546)

[Simuladores soportados 10](#_Toc129269547)

[Instruments 10](#_Toc129269548)

[Programa de desarrolladores de Apple 11](#_Toc129269549)

[Ventana de un proyecto 11](#_Toc129269550)

[1.1.3.Swift 11](#_Toc129269551)

[Definición completa 11](#_Toc129269552)

[Historia (incluir origen del nombre y logo) 12](#_Toc129269553)

[Paradigmas de programación que soporta 12](#_Toc129269554)

[Tipo de licenciamiento 13](#_Toc129269555)

[Playground 13](#_Toc129269556)

[Sintaxis 14](#_Toc129269557)

[1.1.4.Cocoa, Cocoa Touch y Foundation 15](#_Toc129269558)

[conceptos 15](#_Toc129269559)

[Definición 16](#_Toc129269560)

[Historia (incluir origen del nombre y relación con NeXTSTEP) 16](#_Toc129269561)

[Frameworks que contienen 17](#_Toc129269562)

[1.2.Documentar un resumen del documental Jobs vs Gates 17](#_Toc129269563)

[2.1.Investigar los siguientes elementos de la arquitectura Swift: 19](#_Toc129269564)

[2.1.1.Estructura de un archivo (código fuente) 19](#_Toc129269565)

[2.1.2.Estructura de un archivo IPA 19](#_Toc129269566)

[2.1.3.Espacios de nombres 20](#_Toc129269567)

[2.1.4.Módulos 20](#_Toc129269568)

[2.1.5.Instancias 20](#_Toc129269569)

[2.2.Documentar la explicación, la estructura y sintaxis en Swift de: 20](#_Toc129269570)

[2.2.1.Condicionales 20](#_Toc129269571)

[2.2.2.Ciclos 21](#_Toc129269572)

[2.2.3.Arreglos 21](#_Toc129269573)

[2.2.4.Funciones 21](#_Toc129269574)

[2.2.5.Tuplas 22](#_Toc129269575)

[2.2.6.Enumeraciones 22](#_Toc129269576)

[2.2.7.Diccionarios 22](#_Toc129269577)

[2.2.8.Clases 22](#_Toc129269578)

[2.2.8.1.Propiedades 23](#_Toc129269579)

[2.2.8.2.Métodos 23](#_Toc129269580)

[2.3.Realizar las prácticas de programación sugeridas por el maestro 23](#_Toc129269581)

[Practica 2.3.1 23](#_Toc129269582)

[Practica 2.3.2 24](#_Toc129269583)

[2.3.3 25](#_Toc129269584)

[Practica 2.3.4 26](#_Toc129269585)

[Practica 2.3.5 27](#_Toc129269586)

[Practica 2.3.6 28](#_Toc129269587)

[2.3.7 29](#_Toc129269588)

[Practica 2.3.8 30](#_Toc129269589)

[Practica 2.3.9 31](#_Toc129269590)

[Practica 2.3.10 31](#_Toc129269591)

[Practica 2.3.11 32](#_Toc129269592)

[Practica 2.3.12 33](#_Toc129269593)

[Practica 2.3.13 34](#_Toc129269594)

[Practica 2.3.14 35](#_Toc129269595)

[Practica 2.3.15 35](#_Toc129269596)

[Practica 2.3.16 36](#_Toc129269597)

[Practica 2.3.17 38](#_Toc129269598)

[Practica 2.3.18 39](#_Toc129269599)

[Practica 2.3.19 40](#_Toc129269600)

[Practica 2.3.20 41](#_Toc129269601)

[Practica 2.3.21 42](#_Toc129269602)

[Practica 2.3.22 43](#_Toc129269603)

[Practica 2.3.23 44](#_Toc129269604)

[Practica 2.3.24 45](#_Toc129269605)

[Practica 2.3.25 46](#_Toc129269606)

[Practica 2.3.26 47](#_Toc129269607)

[Practica 2.3.27 48](#_Toc129269608)

[Practica 2.3.28 49](#_Toc129269609)

[Practica 2.3.29 51](#_Toc129269610)

[Practica 2.3.30 54](#_Toc129269611)

[Practica 2.3.31 55](#_Toc129269612)

[Practica 2.3.32 56](#_Toc129269613)

[Practica 2.3.33 57](#_Toc129269614)

[Practica 2.3.34 58](#_Toc129269615)

[Practica 2.3.35 59](#_Toc129269616)

[Practica 2.3.36 61](#_Toc129269617)

[Practica 2.3.37 63](#_Toc129269618)

[Practica 2.3.38 65](#_Toc129269619)

[Practica 2.3.39 66](#_Toc129269620)

[Practica 2.3.40 68](#_Toc129269621)

[Practica 2.3.41 70](#_Toc129269622)

[Practica 2.3.42 73](#_Toc129269623)

[Practica 2.3.43 74](#_Toc129269624)

[Referencias. 76](#_Toc129269625)

# 1.1 Investigación.

## 1.1.1 Macintosh Operating System

Es un sistema operativo (OS) diseñado por Apple Inc. para ser instalado y operado en la serie de computadoras Apple Macintosh. Introducido en 1984, es un sistema operativo basado en la interfaz gráfica de usuario (GUI) que desde entonces se ha lanzado en varias versiones diferentes.

### Classic Mac OS.

Lanzado el 24 de enero de 1984 con el Macintosh original y enterrado el 24 de marzo de 2001 con el lanzamiento del primer Mac OS X. El sistema es un derivado de Unix que mantiene en su interfaz gráfica muchos elementos de las versiones anteriores.

### OS X

Mac OS X está diseñado sólo para los dispositivos y computadoras Macintosh. Es el segundo sistema operativo más utilizado después de Microsoft Windows, y generalmente se ejecuta en los ordenadores portátiles, las computadoras de casa y los ordenadores de sobremesa de Apple. En 2012, Apple cambió el nombre de su línea de software de sistema operativo de OS a OS X y las nuevas versiones recibieron nombres de los animales y los grandes felinos, como OS X Mountain Lion, y OS X Lion, OS X El Capitán, macOS Sierra.

### OS X Server

Mac OS X Server es un sistema operativo para servidores desarrollado por Apple Inc. basado en Unix. Es idéntico a su versión de escritorio, pero incluye además herramientas administrativas gráficas para la gestión de usuarios, redes, y servicios de red como LDAP, Servidor de correo, Servidor Samba, DNS, entre otros. También incorpora en sus versiones más recientes un número adicional de servicios y herramientas para configurarlos, tales como Servidor web, herramientas para crear una Wiki, Servidor iChat, y otros más.

Mac OS X Server 1.0 fue lanzado al mercado el 16 de marzo de 1999, ​es el primer sistema operativo creado por Apple tras la adquisición de NeXT. Es la continuación de Rhapsody. Aunque el Mac OS X Server 1.0 tiene una variante de la interfaz "Platinium" del Mac OS 8, está basado en el sistema operativo de NeXTSTEP en lugar del Mac OS Classic, lo que permitió a los usuarios tener una visión preliminar del sistema operativo Mac OS X.

Versiones de Mac OS X server:

* Mac OS X Server 10.1 (Puma Server)
* Mac OS X Server 10.2 (Jaguar Server)
* Mac OS X Server 10.3 (Panther Server)
* Mac OS X Server 10.4 (Tiger Server)
* Mac OS X Server 10.5 (Leopard Server)
* Mac OS X Server 10.7: (Lion)

### Tabla comparativa

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SO | Creador o Compañía | Fecha de lanzamiento | Costo por licencia | Tipo de núcleo(CPU) | Tipo de Kernel | Arquitectura  soportada | Procesador gráfico | % de usuarios en el mundo | % de usuarios en México | Sistemasde archivos soportado | Firewall |
| macOS Ventura(13) | Apple Inc. | 24 de Octubre del 2022 | no genera ningún costo | Hibrido (XNU) | XNU | X86-64, ARM64 | GPU integrada | 12,22 % | 10,87% | Apple File System (APFS), Mac OS Plus, MS-DOS (FAT) y ExFAT | (no encontrado)  Pero cuenta con un firewall propio del sistema. |
| Microsoft Windows10 | Microsoft | 29 de julio de 2015 | 2199 pesos | |  |  | | --- | --- | |  | Híbrido | | Kernel monolítico. | X64, PAE, PX, SSE2, CMPXCHG16b, LAHF / SAHF y PrefetchW | dxdiag | 72,98% | 42,78% | NTFS, ExATURA, UDF y FAT32 | Microsoft Defender Firewall |
| Linux Ubuntu(19o 20) | Mark Shuttleworth | 17 de octubre de 2019 | NA | Linux | Linux | 32-bit (x86) y 64-bit (x86 64, ARM,IA-32, ppc64, IBM ESA/390 Microsoft Windows,RISC-V | Unity | 1% | (no encontrado) | exFAT y la VPN de código abierto WireGuard | iptables |

# 1.1.2.Xcode

Xcode es un entorno de desarrollo integrado (IDE) para macOS creado por Apple. Se utiliza para desarrollar software para Mac, iPhones, iPads, Apple Watch y Apple TV. Xcode proporciona a los desarrolladores un conjunto de herramientas para diseñar, desarrollar, analizar, crear, probar y desplegar sus aplicaciones. El IDE Xcode incluye un código editor, constructor de interfaces visuales, herramientas de depuración y análisis, y mucho más.

Xcode nació dentro de la desaparecida compañía de tecnología, NeXT que fundó Steve Jobs, luego de que Apple decidiera despedirlo como CEO en los 80. Muchas de las tecnologías desarrolladas en NeXT terminaronnuevamente en Apple cuando Steve Jobs retomo la posición de CEO en los 90.

El origen del nombre "Xcode" proviene de su predecesor, el entorno de desarrollo Project Builder. Project Builder fue lanzado junto con Mac OS X en 2001 y fue el primer IDE de Apple diseñado específicamente para desarrollar aplicaciones para el nuevo sistema operativo basado en Unix.

El nombre en clave de Project Builder era "X", lo que reflejaba el hecho de que era una herramienta de desarrollo para Mac OS X. Cuando Apple actualizó Project Builder y lo lanzó como un IDE independiente, decidieron llamarlo "Xcode" en referencia a su origen en Project Builder y su objetivo de desarrollar aplicaciones para Mac OS X y sus derivados.

En resumen, el nombre "Xcode" es una combinación del nombre en clave de Project Builder ("X") y la palabra "code" (código en inglés), reflejando su objetivo como entorno de desarrollo para Mac OS X y sus derivados.

### Versiones:

* Xcode 10.2.x
* Xcode 10.3
* Xcode 11
* Xcode 11.1
* Xcode 11.2.x
* Xcode 11.3.x
* Xcode 11.4.x
* Xcode 11.5
* Xcode 11.6
* Xcode 11.7
* Xcode 12
* Xcode 12.1
* Xcode 12.2
* Xcode 12.3
* Xcode 12.4
* Xcode 12.5
* Xcode 12.5.1
* Xcode 13
* Xcode 13.1
* Xcode 13.2
* Xcode 13.3
* Xcode 13.4
* Xcode 14.0.x
* Xcode 14.1
* Xcode 14.2

### nib, xib, storyboard

Nib: es un acrónimo de "NeXT Interface Builder" y es un formato de archivo que se utilizaba en la plataforma NeXTSTEP de la que se originó el sistema operativo macOS. Los archivos Nib son archivos binarios que contienen información sobre la interfaz de usuario de una aplicación, como la disposición de los elementos en pantalla y los datos de configuración. Los archivos Nib se crean y editan utilizando la herramienta "Interface Builder" de Xcode.

Xib: es una evolución de los archivos Nib y es un formato de archivo XML utilizado por Xcode para describir la interfaz de usuario de una aplicación. Los archivos Xib se crean y editan utilizando la herramienta "Interface Builder" de Xcode, al igual que los archivos Nib. A diferencia de los archivos Nib, los archivos Xib son archivos de texto legibles por humanos, lo que los hace más fáciles de trabajar con el control de versiones y el trabajo en equipo.

Storyboard: es un formato de archivo introducido por Apple en Xcode 4 como una forma más avanzada de describir la interfaz de usuario de una aplicación. Los archivos Storyboard son archivos XML que contienen información sobre múltiples pantallas de la aplicación, en lugar de una sola pantalla como en los archivos Nib y Xib. Los Storyboards permiten la creación de prototipos de aplicaciones completas y permiten la visualización y el seguimiento de la navegación de la aplicación. Los archivos Storyboard se crean y editan utilizando la herramienta "Interface Builder" de Xcode.

### AppStore (incluir requisitos para publicar Apps)

Un lugar seguro

para encontrar tus

apps favoritas.

Durante más de una década, el App Store ha demostrado ser un lugar seguro y confiable para descubrir y descargar apps. A simple vista, puede parecer una galería de productos, pero es mucho más que eso. Es un lugar innovador que te ofrece experiencias increíbles. Y para que esto sea posible, nos aseguramos de que los casi dos millones de apps disponibles cumplan con las normas de privacidad, seguridad y contenido más estrictas. Así podrás disfrutar cada app con total tranquilidad.

* Crear el ID de la App y el perfil de desarrollo y distribución
  + El perfil de desarrollo se crea para que puedas pre-visualizar tu App antes de publicarse. Para ello, tienes que acceder a la pestaña Development dentro del apartado Provisioning Profiles. La creación de este perfil implica asociar tu iPhone/iPad, a través de su número de identificación (UDID).
  + El perfil de distribución se crea para poder publicar y distribuir tu App en App Store. Se accede en la pestaña Distribution dentro del mismo apartado Provisioning Profiles.
* Prepara el material gráfico
  + Icono (1024px x 1024 px):
  + Capturas promocionales.
  + iPhone: imágenes de 5.5 pulgadas (1242px x 2208 px)
  + iPad: imágenes de 12,9 pulgadas (2048px x 2732 px)
* Cumplimenta los datos relativos a tu App en iTunes Connect
  + Información de la App
  + Precio y Disponibilidad
* Generar el IPA desde la plataforma de creación de la App
  + La IPA de desarrollador que podrás descargar con tu móvil (que recordemos deberá estar asociado a la App como se indica en el paso 1 de esta entrada), para poder previsualizar la App en el dispositivo
  + La IPA de distribución, que es la que tienes que subir a iTunes Connect para poder publicar la App
* Entregar la aplicación
  + Verde: Lista para la venta (ya se habría publicado)
  + Rojo: Rechazada (entonces recibirás una notificación con los pasos a seguir para solucionar la incidencia) o retirada de la venta por parte del desarrollador.
  + Amarillo: Pendiente de revisión.

### Estructura de una App empaquetada

Una aplicación empaquetada para iOS sigue una estructura de directorios específica que se organiza en un archivo comprimido llamado "ipa".

La estructura de un archivo .ipa típicamente contiene los siguientes elementos:

Payload: esta carpeta contiene la aplicación real en sí misma, con la extensión ".app". Dentro de esta carpeta, encontrarás el archivo ejecutable de la aplicación (el archivo .exe equivalente en Windows), así como todos los archivos de recursos necesarios para la aplicación, como imágenes, archivos de sonido y otras librerías.

iTunesMetadata.plist: este archivo contiene información sobre la aplicación, como el nombre, la versión, el autor, el precio y la descripción.

CodeResources: este archivo contiene una lista de todos los archivos en la carpeta Payload y sus respectivas firmas digitales para garantizar que la aplicación no haya sido modificada.

Entitlements.plist: este archivo contiene información sobre los permisos que tiene la aplicación, como acceso a la cámara, micrófono, ubicación, etc.

Provisioning Profile: este archivo contiene información sobre los dispositivos en los que se puede instalar la aplicación, así como la clave privada y el certificado que se utilizarán para firmar la aplicación.

Estos son algunos de los principales componentes que se encuentran en una aplicación empaquetada para iOS. La estructura puede variar según la aplicación, pero en general, sigue esta estructura básica.

### Modelo-Vista-Controlador

MVC (Modelo-Vista-Controlador) es un patrón en el diseño de software comúnmente utilizado para implementar interfaces de usuario, datos y lógica de control. Enfatiza una separación entre la lógica de negocios y su visualización. Esta "separación de preocupaciones" proporciona una mejor división del trabajo y una mejora de mantenimiento

* Modelo: Maneja datos y lógica de negocios.
  + El modelo define qué datos debe contener la aplicación. Si el estado de estos datos cambia, el modelo generalmente notificará a la vista (para que la pantalla pueda cambiar según sea necesario) y, a veces, el controlador (si se necesita una lógica diferente para controlar la vista actualizada).
* Vista: Se encarga del diseño y presentación.
  + La vista define cómo se deben mostrar los datos de la aplicación.
* Controlador: Enruta comandos a los modelos y vistas.
  + El controlador contiene una lógica que actualiza el modelo y/o vista en respuesta a las entradas de los usuarios de la aplicación.

### Simuladores soportados

* Xcode 14.2
  + iOS 12.4-16.2
  + tvOS 12.4-16.1
  + watchOS 7-9.1
* Xcode 14.1
  + OS 12.4-16.1
  + tvOS 12.4-16.1
  + watchOS 7-9.1
* Xcode 13
  + iOS 10.3.1-15
  + tvOS 10.2-15
  + watchOS 3.2-8

### Instruments

Xcode Instruments se describe mejor como una herramienta de prueba y análisis de rendimiento potente y flexible. Es parte del conjunto de herramientas de Xcode, que también incluye herramientas como Create ML y Accessibility Inspector. Los instrumentos se pueden utilizar para:

* Rastree problemas en su código fuente
* Analiza el rendimiento de tu aplicación
* Encuentra problemas de memoria
* Entre otras cosas mas.

Puede acceder a los instrumentos usando Producto ➔ Perfil desde el menú de Xcode para el cual el identificador de acceso directo es ⌘ + I. Esto reconstruirá su producto y abrirá la descripción general de los instrumentos de Xcode.

### Programa de desarrolladores de Apple

Al inscribirse en Apple Developer Program, las personas y las organizaciones reciben todo lo que necesitan para desarrollar apps y poder distribuirlas. Los miembros pueden distribuir apps en App Store para iPhone, iPad, Mac, Apple Watch, Apple TV y iMessage. También pueden distribuir software fuera de Mac App Store, así como ofrecer apps personalizadas a empresas específicas y apps exclusivas a sus empleados a través de Apple Business Manager. Una vez que se verifique la información de tu inscripción y que hayas aceptado el acuerdo de licencia del programa asociado, podrás comprar la membresía. La tarifa anual de Apple Developer Program es de 99 USD y la tarifa anual de Apple Developer Enterprise Program es de 299 USD.

### Ventana de un proyecto

Ventana de un proyecto: Es la ventana principal de Xcode que muestra todos los archivos y carpetas de un proyecto. Desde aquí, se puede acceder a los archivos de código fuente, recursos, bibliotecas, configuraciones y más. También se pueden crear, editar y eliminar archivos y carpetas..

* Panel de navegación: Este panel se encuentra en el lado izquierdo de la ventana del proyecto y muestra una vista jerárquica de los archivos y carpetas en el proyecto. Se puede usar para buscar archivos específicos, agregar nuevos archivos y carpetas, y organizar la estructura del proyecto.
* Panel de utilerías: Este panel se encuentra en el lado derecho de la ventana del proyecto y muestra diferentes opciones y configuraciones para el archivo o elemento seleccionado en el panel de navegación. Aquí se pueden configurar propiedades de la interfaz de usuario, configuraciones de construcción y otros ajustes.
* Panel de depuración: Este panel se utiliza durante la depuración del código de la aplicación y muestra información detallada sobre los errores y excepciones que se producen. También se puede utilizar para inspeccionar variables y objetos durante la ejecución de la aplicación, lo que es útil para encontrar problemas en el código.

## 1.1.3.Swift

### Definición completa

Un poderoso lenguaje al alcance de todos para crear increíbles apps.

Swift es un lenguaje de programación poderoso e intuitivo creado por Apple para desarrollar apps de iOS, Mac, Apple TV y Apple Watch. Está diseñado para brindar a los desarrolladores más libertad que nunca. Y como es fácil de usar y de código abierto, es ideal para que cualquier persona con una idea pueda hacerla realidad.

Swift es un lenguaje rápido y eficiente que proporciona información en tiempo real y puede incorporarse fácilmente al código de Objective-C existente. Así, los desarrolladores no sólo pueden programar de una forma más segura y confiable, sino también ahorrar tiempo y enriquecer la experiencia con las apps.

### Historia (incluir origen del nombre y logo)

Swift es un lenguaje de programación creado por Apple con el compilador LLVM para los sistemas operativos OS X e iOS presentado el 2 de junio del 2014 en el WWDC 2014 junto con su manual. El lenguaje en un principio coexistirá con Objective-C lenguaje que maneja la plataforma de programación oficial de Apple Xcode y permite la edición de código en tiempo real de Cocoa y Cocoa Touch.

El desarrollo de Swift se ha llevado a cabo durante 4 largos años en los que la compañía ha trabajado muy duro. Hay que destacar que gracias a este nuevo lenguaje de programación los de Cupertino también tratarán de limitar que se porten las aplicaciones a otras plataformas móviles, consiguiendo así cierta exclusividad. Como anécdota, debemos decir que durante la presentación de Swift Federeghi trató de llamar la atención de los asistentes señalando que podrían jugar y enredar todo lo que quisieran con su nuevo lenguaje de programación.

Con Swift, Apple pone a disposición de los desarrolladores muchas más funcionalidades y más libertad a la hora de realizar su trabajo gracias a más de 4.000 APIs, entre las que encontramos la integración con Touch ID, integración con Siri, la posibilidad de crear filtros para la aplicación nativa de la cámara o nuevas APIs para iCloud entre otras muchas.

El nombre "Swift" se eligió porque se quería un nombre que fuera fácil de recordar, corto y memorable. Según Chris Lattner, el creador de Swift, el nombre "Swift" representa la velocidad y la facilidad de uso del lenguaje. Además, el nombre se eligió porque era un guiño al poema "A Swiftly Tilting Planet" de Madeleine L'Engle.

El logotipo, se compone de un símbolo que se asemeja a una "S" y una flecha que apunta hacia adelante. La "S" significa Swift, mientras que la flecha representa la velocidad y el progreso. El diseño del logotipo se inspiró en la simplicidad del logotipo de Apple.

El logotipo de Swift se creó en 2014 por el diseñador gráfico de Apple, Rob Janoff, quien también diseñó el logotipo original de Apple en 1977.

### Paradigmas de programación que soporta

El lenguaje de programación Swift es un lenguaje de programación multiparadigma, se dice que un lenguaje es multiparadigma cuando acepta dos o más paradigmas de programación.

Swift acepta varios paradigmas: la orientación a objetos, la orientación a protocolos, y también está orientado a la programación funcional.

Según Apple Swift se basa en lo mejor de C y de Objective-C, pero sin las restricciones de C, es decir, sin las restricciones de ambos lenguajes.

#### POO

La programación orientada a objetos (POO) es un paradigma de programación que se enfoca en modelar el mundo real a través de objetos que interactúan entre sí para realizar tareas. En POO, los objetos se definen por su estado (datos) y su comportamiento (métodos).

#### Programación enfocada a protocolos

la programación orientada a protocolos es un enfoque de programación que se centra en definir comportamientos que los tipos de datos deben tener para cumplir con ciertas funcionalidades. En este enfoque, el énfasis se pone en los protocolos (interfaces) que especifican los métodos y propiedades que un tipo debe implementar para poder ser utilizado en diferentes partes del código.

En Swift, la programación orientada a protocolos es una parte fundamental de su diseño. Swift fomenta la creación de tipos de datos independientes y reutilizables mediante la definición de protocolos. Los protocolos en Swift permiten definir una interfaz común para diferentes tipos de datos, lo que facilita la creación de código genérico y flexible.

### Tipo de licenciamiento

Swift es gratis y de código abierto, y está disponible para desarrolladores, educadores y estudiantes bajo la licencia de código abierto Apache 2.0. Proporcionamos archivos binarios para macOS y Linux que pueden compilar códigos para iOS, macOS, watchOS, tvOS y Linux.

### Playground

Swift Playgrounds es una app revolucionaria para el iPad y la Mac que te enseña a programar de una forma divertida e interactiva con Swift, el lenguaje de programación de Apple que usan los profesionales para crear las apps más populares hoy en día. Y como no requiere conocimientos previos, es ideal para estudiantes que apenas están empezando a programar.

#### Prototipos

Los prototipos en Playground Swift son una forma rápida y eficiente de probar y experimentar con el código de Swift sin tener que crear un proyecto completo. Los playgrounds son una herramienta interactiva que permite escribir, probar y depurar código en tiempo real.

En el Playground Swift, puedes escribir código Swift en tiempo real y ver los resultados de manera inmediata. Además, puedes utilizar diferentes herramientas y recursos que te permiten explorar las características del lenguaje de programación, como la documentación integrada, las sugerencias de autocompletado y la depuración interactiva.

#### Prototipos de apps y juegos

Los prototipos de aplicaciones y juegos en Swift son una forma eficiente de probar y experimentar con ideas de diseño y mecánicas de juego antes de comenzar a desarrollar una aplicación o juego completo. Los prototipos pueden ayudarte a validar tus ideas y a descubrir problemas antes de invertir tiempo y recursos en el desarrollo completo de la aplicación o juego.

### Sintaxis

* Comentarios: Los comentarios en Swift se utilizan para documentar el código y explicar su funcionamiento. Los comentarios de una sola línea comienzan con doble barra (//) y los comentarios de varias líneas se encierran entre /\* y \*/.
* Uso de punto y coma: En Swift, el uso de punto y coma (;) es opcional al final de cada instrucción.
* Nomenclatura de identificadores: Los identificadores en Swift se utilizan para nombrar variables, constantes, funciones y tipos de datos. Los identificadores deben comenzar con una letra o un guión bajo, no pueden contener espacios y deben ser únicos.
* Literales: En Swift, los literales se utilizan para representar valores fijos en el código. Por ejemplo, un literal de cadena representa una cadena de caracteres y un literal numérico representa un número.
* Tipos de datos: Swift tiene varios tipos de datos, incluyendo tipos numéricos (como Int y Float), tipos de cadena (String), tipos booleanos (Bool) y tipos de colección (como Array y Dictionary). Además, Swift admite rangos de valores utilizando la sintaxis de rango (por ejemplo, 1...5 representa el rango de valores del 1 al 5).
* Alias de tipos: Los alias de tipos se utilizan para dar nombres alternativos a los tipos de datos existentes en Swift. Por ejemplo, typealias MyInt = Int permite usar MyInt en lugar de Int en todo el código.
* Inferencia de tipos: Swift utiliza la inferencia de tipos para determinar el tipo de datos de una variable o constante en función del valor asignado. Por ejemplo, si se asigna el valor "Hola" a una variable, Swift inferirá que el tipo de dato es String.
* Declaración de variables y constantes: En Swift, las variables se declaran utilizando la palabra clave var y las constantes utilizando la palabra clave let. Por ejemplo, var miVariable = 5 declara una variable llamada miVariable con un valor inicial de 5.
* Nomenclatura de variables y constantes: Los nombres de variables y constantes en Swift deben comenzar con una letra o un guión bajo y pueden contener caracteres alfanuméricos y guiones bajos. Además, Swift admite caracteres Unicode especiales para nombrar variables y constantes.
* Tipo de dato Optional: En Swift, el tipo de dato Optional se utiliza para representar valores que pueden ser nulos o vacíos. La sintaxis para declarar una variable Optional es agregar un signo de interrogación (?) después del tipo de dato (por ejemplo, var miVariable: Int?).
* Secuencias de escape en Swift: Las secuencias de escape se utilizan en Swift para representar caracteres especiales dentro de una cadena de caracteres. Por ejemplo, \n representa un salto de línea y \t representa una tabulación.
* Operadores:
  + Aritméticos: Los operadores aritméticos en Swift incluyen suma (+), resta (-), multiplicación (\*), división (/) y módulo (%).
  + Relacionales: Los operadores relacionales se utilizan para comparar dos valores y devolver un valor booleano. Los operadores relacionales en Swift incluyen igualdad (==), desigualdad (!=), mayor que (>), menor que (<), mayor o igual que (>=) y menor o igual que (<=).
  + Lógicos: Los operadores lógicos se utilizan para combinar dos valores booleanos y devolver un valor booleano. Los operadores lógicos en Swift incluyen AND lógico (&&), OR lógico (||) y NOT lógico (!).
  + Bit a bit (bitwise): Los operadores bit a bit se utilizan para manipular los bits individuales de un valor. Los operadores bit a bit en Swift incluyen AND bit a bit (&), OR bit a bit (|), XOR bit a bit (^) y NOT bit a bit (~).
  + Asignación: El operador de asignación (=) se utiliza para asignar un valor a una variable o constante.
  + Rango: El operador de rango se utiliza para crear un rango de valores. La sintaxis para el operador de rango es (inicio...fin) para crear un rango cerrado o (inicio..<fin) para crear un rango semi-abierto.
  + Condicional ternaria: El operador condicional ternario (condición ? verdadero : falso) se utiliza para tomar una decisión y asignar un valor a una variable o constante en función de esa decisión.
  + Decisiones: Los operadores de decisión en Swift incluyen el operador if, que se utiliza para tomar una decisión basada en una condición, y el operador switch, que se utiliza para tomar una decisión basada en múltiples condiciones.
  + Ciclos en Swift: Los ciclos en Swift incluyen el ciclo for-in, que se utiliza para iterar sobre una secuencia de valores, el ciclo while, que se utiliza para ejecutar una serie de instrucciones mientras se cumple una condición, y el ciclo repeat-while, que se utiliza para ejecutar una serie de instrucciones al menos una vez y luego repetir mientras se cumple una condición.

## 1.1.4.Cocoa, Cocoa Touch y Foundation

### conceptos

Framework:

Un framework es un conjunto de herramientas y bibliotecas de software que se utilizan para desarrollar aplicaciones o software específicos. Un framework proporciona una estructura de trabajo y una serie de bibliotecas de software que se utilizan para realizar tareas específicas, lo que facilita y acelera el proceso de desarrollo de software. Los frameworks también pueden proporcionar patrones de diseño, arquitecturas y convenciones de codificación que se utilizan para crear aplicaciones de manera más estructurada y organizada.

API:

Una API (Application Programming Interface) es un conjunto de definiciones, protocolos y herramientas que se utilizan para desarrollar aplicaciones de software. Una API proporciona un conjunto de funciones, métodos y procedimientos que se utilizan para interactuar con un software o servicio específico, lo que permite a los desarrolladores crear aplicaciones que utilicen las funciones y características de ese software o servicio. Las API también pueden proporcionar documentación, ejemplos de código y herramientas de desarrollo que ayudan a los desarrolladores a utilizar las funciones de una API de manera efectiva y eficiente. En resumen, una API es una interfaz de programación que permite a los desarrolladores interactuar con un software o servicio específico de manera programática.

.

### Definición

Cocoa, Cocoa Touch y Foundation son frameworks de desarrollo de software de Apple que se utilizan para crear aplicaciones para Mac, iOS, watchOS y tvOS.

Cocoa: es un framework de desarrollo de software para Mac OS X. Se basa en el lenguaje de programación Objective-C y proporciona una serie de clases y herramientas para el desarrollo de aplicaciones de escritorio para Mac.

Cocoa Touch: es un framework de desarrollo de software para iOS. Se basa en el lenguaje de programación Swift y proporciona una serie de clases y herramientas para el desarrollo de aplicaciones móviles para iPhone, iPad y iPod Touch.

Foundation: es un framework de desarrollo de software para Mac OS X e iOS. Proporciona una serie de clases y herramientas para el desarrollo de aplicaciones que necesitan acceso a servicios de bajo nivel, como el sistema de archivos, la red y los servicios de localización.

Algunos conceptos importantes dentro de cada uno de estos frameworks son:

En Cocoa, algunos conceptos importantes incluyen las clases NSView, NSWindow, NSButton, NSTextField y NSImage, que se utilizan para crear interfaces de usuario y manejar eventos de usuario.

En Cocoa Touch, algunos conceptos importantes incluyen las clases UIView, UIButton, UILabel, UIImageView y UIImage, que se utilizan para crear interfaces de usuario y manejar eventos de usuario en dispositivos móviles.

En Foundation, algunos conceptos importantes incluyen las clases NSString, NSData, NSURL, NSDate y NSFileManager, que se utilizan para trabajar con datos y servicios de bajo nivel en aplicaciones de Mac OS X e iOS. Además, Foundation proporciona una serie de colecciones de datos, como NSArray y NSDictionary, que se utilizan para almacenar y manipular datos de manera eficiente.

### Historia (incluir origen del nombre y relación con NeXTSTEP)

Cocoa es un framework de desarrollo de software de Apple que se utiliza para desarrollar aplicaciones para Mac OS X. El nombre "Cocoa" se refiere a la tradición de utilizar nombres de alimentos en la nomenclatura de Apple. Cocoa es el sucesor de NeXTSTEP, un sistema operativo desarrollado por NeXT Computer, Inc., una empresa fundada por Steve Jobs después de su salida de Apple en 1985.

NeXTSTEP fue un sistema operativo innovador que introdujo muchas características que son ahora comunes en los sistemas operativos modernos, como el uso de un entorno gráfico de usuario y una interfaz de programación de aplicaciones (API) basada en objetos. La API de NeXTSTEP se llamaba "Cocoa" y estaba basada en el lenguaje de programación Objective-C.

En 1997, Apple adquirió NeXT Computer, Inc. y con ella adquirió NeXTSTEP y Cocoa. Apple comenzó a utilizar la tecnología de NeXTSTEP y Cocoa en su sistema operativo Mac OS X, que se lanzó en 2001. Desde entonces, Cocoa se ha convertido en el principal framework de desarrollo de software de Apple para Mac OS X y iOS.

### Frameworks que contienen

* Core Audio: Es un framework que proporciona servicios de audio de bajo nivel para aplicaciones en la plataforma de Apple. Este framework se utiliza para grabar, reproducir y manipular archivos de audio en macOS e iOS. También permite la creación de software de efectos de sonido y sintetizadores de audio.
* Core Data: Es un framework que se utiliza para manejar datos estructurados en aplicaciones. Este framework proporciona una forma de almacenar y recuperar datos de manera eficiente en macOS e iOS. Core Data es utilizado en aplicaciones que necesitan manejar grandes cantidades de datos y que requieren un alto rendimiento.
* Core Image: Es un framework que proporciona herramientas para la manipulación de imágenes en macOS e iOS. Core Image se utiliza para procesar imágenes en tiempo real y aplicar efectos visuales en tiempo real.
* Core Animation: Es un framework que proporciona herramientas para crear animaciones en macOS e iOS. Core Animation se utiliza para crear animaciones fluidas y complejas en aplicaciones.
* Bonjour: Es un framework que se utiliza para descubrir servicios en redes locales. Bonjour se utiliza para que dispositivos en una misma red se puedan descubrir mutuamente y puedan comunicarse entre sí.
* Core Location: Es un framework que proporciona información sobre la ubicación de un dispositivo en tiempo real. Core Location se utiliza en aplicaciones que necesitan conocer la ubicación del dispositivo, como aplicaciones de mapas y de navegación.
* UIKit: Es un framework que proporciona herramientas para la creación de interfaces de usuario en iOS. UIKit se utiliza para crear la mayoría de las interfaces de usuario en aplicaciones de iOS.

# 1.2.Documentar un resumen del documental Jobs vs Gates

Introducción:

El documental de mentes brillantes se centra en dos perspectivas y personalidades que producen una relación primeramente amigable (de negocios) y posteriormente de rivalidades donde ambas personas representan dos corporaciones que disputan en el mismo territorio (tecnología computacional) y a su vez se centran en cuadrantes diferente uno enfocado en el desarrollo general de su producto (alianza con otros empresas) colocando así una normalización o estándar en el funcionamiento interno (software) en la mayor cantidad de dispositivos, y el segundo donde se centra en el crecimiento de su compañía restringiendo y mejorando únicamente sus productos para trazar la innovación y mantenerse en el frente del mercado de la tecnología.

El joven Bill se encontraba estudiando en la Universidad de Harvard y durante el año de 1974 con el surgimiento de una computadora llamada Alter 8800 (sin terminal, teclado, ni otros componentes que se relación a las computadoras el día de hoy), Bill y su amigo Paul Allen diseñan un lenguaje de programación para la computadora, lenguaje el cual logran vender a los creadores de la computadora Alter, sustentando con el dinero de la venta una pequeña empresa que deciden llamar “Microsoft”

Steve Jobs se reúne con su amigo Steve Woznia un apasionado por la electrónica y las computadoras quien se le ocurre una la idea de construir una computadora con un microprocesador, pero añadiendo un teclado y un monitor Jobs al ser un hombre visionario hace que apueste a empezar a comercializarla iniciando la empresa Apple. Bajo esta cuestión la Apple1 fue un fracaso comercial (vendiendo únicamente en el mercado de aficionados a las computadoras) lo que provoco que el desarrollo de la Apple2 se desarrollara más en el diseño y formulando un concepto innovador en los computadores.

Bill analiza el hardware de la Apple2 encontrando la manera de mejorar su funcionamiento en software desarrollando una tarjeta que permita el uso de diversos programas orientados a las oficinas (procesadores de texto, hojas de cálculo, entre otros) Bill Gates ve una posibilidad de cambio dándole una habilidad a la Apple para la inducción en el ambiente de los negocios, Gates contacta a Jobs para darle a conocer cómo puede su computadora personal convertirse en una máquina de negocios, en este momento hay una victoria tanto para Apple y para Microsoft.

Para 1980 Microsoft se vuelve el proveedor de softcards para los usuarios de Apple convirtiendo a este ingreso en mayor de todos.

Con los pasos anteriores Bill se fija un objetivo aun mayor contactando así a IBM y les vende un sistema operativo que aún no poseía, teniendo conocimiento de algunas compañías que se dedicaban a realizar sistemas operativos compra uno por el valor de $50.000 dólares y lo renombra Microsoft DOS, adquiriendo todos los derechos de este, realiza una negociación súper beneficiosa para él ya que le daba el software a IBM pero que podía venderlo a terceros (bajo el contrato de no exclusiva).

Mientras tanto Jobs contacta a Xerox una compañía de programación orientada a objetos y cosas de la red, Jobs propone a esta compañía que invierta en Apple el valor de $1.000.000 en condición de que los dejaran ingresar a conocer sus instalaciones Xerox.

Jobs toma la idea de una Interfax grafica de un usuario de Xerox que consistía en manejo de las ventanas, ya que esto cambiaría la apariencia, y crea un equipo para incorporarla en una nueva computadora llamada Macintosh, este proyecto demanda toda su atención y delega a Jhon un ex ejecutivo de Pepsi para que sea el director de Apple.

Steve Jobs invita a Gates para que cree el software para la Macintosh. Y Bill aprovechando la confianza dada por Jobs no pierde la oportunidad y comienza el desarrollo de una interfaz gráfica propia la cual tenía por nombre clave “Interfaz Gráfica” culminando en lo que se conoció como “Windows”.

Jobs realiza su lanzamiento de la Macintosh dándole un gran éxito, pero era lenta y sus ventas cayeron a la mitad. Para ese tiempo la junta de Apple le solicita la renuncia a Jobs para mantenerlo fuera de la compañía esto debido a que representaba más perdidas por sus enemistades y pensamientos liberales. Mientras que esto sucedía el Sistema Operativo “Microsoft DOS” se convierte en el estándar para las computadoras.

Al lograr la estandarización del sistema operativo se le otorgaba a cualquier consumidor que comprase una computadora este tendría hojas de cálculo, procesadores de texto, y demás programas de Microsoft, tanto así que los competidores acusan a Gates de monopolio.

Apple sigue en declive para 1996 ya estaba cerca de la bancarrota, nuevamente la junta solicita a Jobs su regreso después de 12 años de ausencia, para lo cual Jobs contacta a Gates para que inyecte capital a su compañía esto fue aceptado por Gates debido a dos razones la principal Jobs le comento “Requieres de enemigos en el mercado y si Apple se hunde terminaras sin oponentes”, la segunda es que vio la posibilidad de salir de sus inconvenientes legales que pasaba por el momento.

En el 2001 Jobs incursiona en productos nuevos como en el IPad, aun así, no contento con esto Jobs crea el IPhone para el 2007 su gran producto. Pero debido a la relación de enemistad de Apple y Bill este percibe que su imagen iba en decrecimiento y con el fin de mejorar su imagen se dedica a la filantropía, crea la Fundación Bill y Melinda Gates siendo la organización privada más grande del mundo. Para Jobs su salud en el 2011 lo hace renunciar debido a un cáncer de páncreas que padecía, y muere el 05 de octubre de ese mismo año y para el siguiente año Apple se convierte en la compañía más valiosa del mundo.

# 2.1.Investigar los siguientes elementos de la arquitectura Swift:

## 2.1.1.Estructura de un archivo (código fuente)

Un archivo de código fuente de Swift tiene una estructura básica que incluye las siguientes partes:

Directivas del preprocesador: Son las líneas que comienzan con el símbolo "#". Estas líneas se utilizan para importar otros archivos de código fuente, habilitar o deshabilitar características específicas de compilación, o para definir constantes que se utilizarán en el código.

Declaraciones de importación: Estas líneas se utilizan para importar módulos o frameworks externos que se utilizarán en el código.

Definiciones de tipos: Las definiciones de tipos incluyen las declaraciones de clases, estructuras y enumeraciones que se utilizan en el código.

Funciones: Las funciones son bloques de código que se utilizan para realizar una tarea específica. En Swift, las funciones se definen utilizando la palabra clave "func" seguida del nombre de la función y sus parámetros.

Variables y constantes: Las variables y constantes son elementos importantes en Swift y se utilizan para almacenar valores. En Swift, las variables se definen utilizando la palabra clave "var", mientras que las constantes se definen utilizando la palabra clave "let".

Comentarios: Los comentarios son secciones de texto que se utilizan para documentar el código. En Swift, los comentarios de una línea comienzan con "//" y los comentarios de varias líneas se colocan entre "/" y "/".

## 2.1.2.Estructura de un archivo IPA

Un archivo IPA (iOS App Store Package) es un formato de archivo utilizado para distribuir aplicaciones iOS. El archivo IPA contiene todo lo necesario para instalar y ejecutar la aplicación en un dispositivo iOS, incluyendo el código binario, recursos, bibliotecas y marcos de trabajo.

La estructura de un archivo IPA es la siguiente:

Carpeta "Payload": Contiene la aplicación empaquetada en sí misma. El archivo dentro de esta carpeta suele tener una extensión .app.

Archivo "Info.plist": Es un archivo de configuración que contiene información sobre la aplicación, como su nombre, versión, icono y permisos requeridos.

Carpeta "Symbols": Contiene los símbolos de depuración de la aplicación para su uso con el depurador de Xcode.

Archivo "iTunesArtwork": Es el icono de la aplicación en formato JPEG o PNG utilizado en la App Store.

Archivo "iTunesMetadata.plist": Contiene información de metadatos utilizada por la App Store, como el nombre del desarrollador, la categoría de la aplicación y la descripción de la aplicación.

La estructura de un archivo IPA puede variar dependiendo de cómo se haya empaquetado la aplicación. En algunos casos, también puede contener otros archivos y carpetas necesarios para que la aplicación se ejecute correctamente.

Para crear un archivo IPA, primero se debe compilar el código fuente de la aplicación utilizando Xcode o alguna otra herramienta de compilación. A continuación, se debe empaquetar la aplicación en un archivo ZIP y cambiar la extensión a .ipa.

## 2.1.3.Espacios de nombres

Los espacios de nombres (namespaces) son una forma de evitar conflictos de nombres entre diferentes componentes de un programa. En Swift, los espacios de nombres se implementan mediante el uso de módulos.

## 2.1.4.Módulos

Un módulo en Swift es una unidad de código que se puede importar y utilizar en otros programas. Un módulo puede contener clases, estructuras, enumeraciones, funciones y otros tipos de datos. Cada módulo tiene un nombre único que lo identifica.

Cuando se importa un módulo en un programa, se crea un espacio de nombres que contiene todos los tipos de datos y funciones definidos en ese módulo. Esto significa que no hay conflictos de nombres entre diferentes módulos, incluso si contienen tipos de datos o funciones con el mismo nombre.

## 2.1.5.Instancias

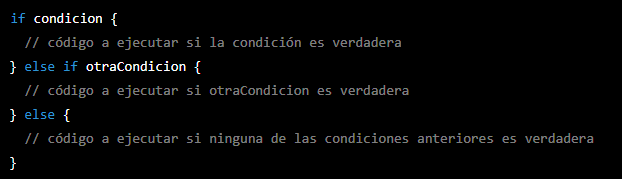
Las instancias en Swift son una forma de crear un objeto a partir de una clase o estructura. Una instancia es una copia única de la clase o estructura que se puede utilizar para acceder a sus propiedades y métodos. Las instancias se crean mediante la palabra clave "new" o mediante la inicialización de una clase o estructura utilizando su constructor.

Cada instancia tiene su propio conjunto de propiedades y métodos que se pueden acceder y modificar de forma independiente de otras instancias de la misma clase o estructura.

# 2.2.Documentar la explicación, la estructura y sintaxis en Swift de:

## 2.2.1.Condicionales

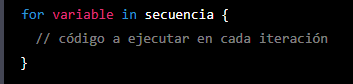
Los condicionales en Swift son estructuras de control que permiten que el flujo de ejecución de un programa se desvíe en diferentes direcciones en función de una condición booleana. En Swift, se utilizan los siguientes operadores de comparación: == (igual a), != (diferente a), < (menor que), > (mayor que), <= (menor o igual que), >= (mayor o igual que).



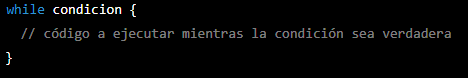
## 2.2.2.Ciclos

Los ciclos en Swift son estructuras de control que permiten que una sección de código se repita varias veces. En Swift, existen tres tipos de ciclos: for-in, while y repeat-while.

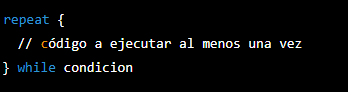
Sintaxis de un ciclo for:



La sintaxis de un ciclo while



La sintaxis de un ciclo repeat-while



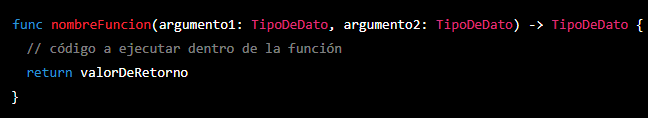
## 2.2.3.Arreglos

un arreglo es una estructura de datos que permite almacenar múltiples valores del mismo tipo en una única variable. Los arreglos en Swift son tipos por valor, lo que significa que cuando se asignan a una variable, se crea una copia independiente de los datos.



## 2.2.4.Funciones

Una función en Swift es un bloque de código que se puede llamar desde otra parte del programa. Las funciones pueden recibir argumentos y devolver un valor opcional.



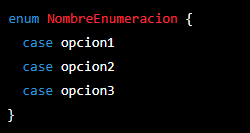
## 2.2.5.Tuplas

Una tupla en Swift es una colección de valores que se pueden agrupar y pasar como un único valor. Las tuplas se utilizan a menudo para devolver varios valores de una función.



## 2.2.6.Enumeraciones

Una enumeración en Swift es un tipo de dato que define un conjunto de valores relacionados. Las enumeraciones se utilizan a menudo para representar un conjunto de opciones o estados.



## 2.2.7.Diccionarios

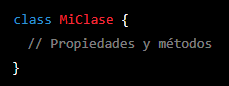
Los diccionarios en Swift son una estructura de datos que permiten almacenar y acceder a valores asociados a una clave única. La sintaxis para definir un diccionario en Swift es la siguiente:



donde TipoClave es el tipo de dato de la clave y TipoValor es el tipo de dato del valor.

## 2.2.8.Clases

Las clases en Swift son una forma de definir objetos que agrupan propiedades y métodos relacionados. La sintaxis para definir una clase en Swift es la siguiente:

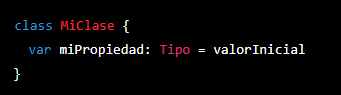


Para crear una instancia de una clase se usa la siguiente sintaxis:



### 2.2.8.1.Propiedades

La manera de definir las propiedades dentro de las clases, usan la siguiente sintaxis

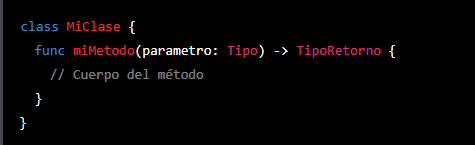


Para acceder a las propiedades se emplea la sintaxis



### 2.2.8.2.Métodos

Para definir los métodos dentro de las clases se emplea la siguiente sintaxis.



Para acceder a los métodos dentro de una clase se usa la siguiente sintaxis

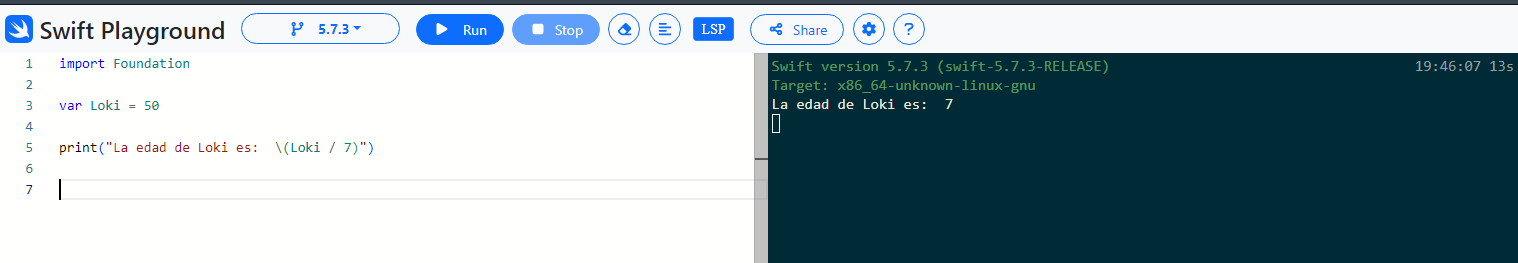


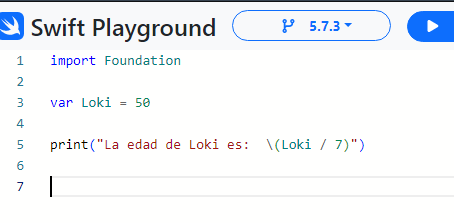
# 2.3.Realizar las prácticas de programación sugeridas por el maestro

### Practica 2.3.1

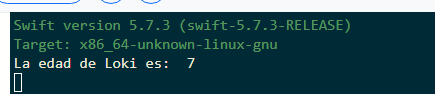
Se desea obtener el calculo de la edad de Loki en años humanos, a través de la edad en años perros.

Definición de una variable “Loki” y un método de impresión “print()” con el cálculo de la variable entre 7 siendo esta la consideración definida al principio de la práctica.



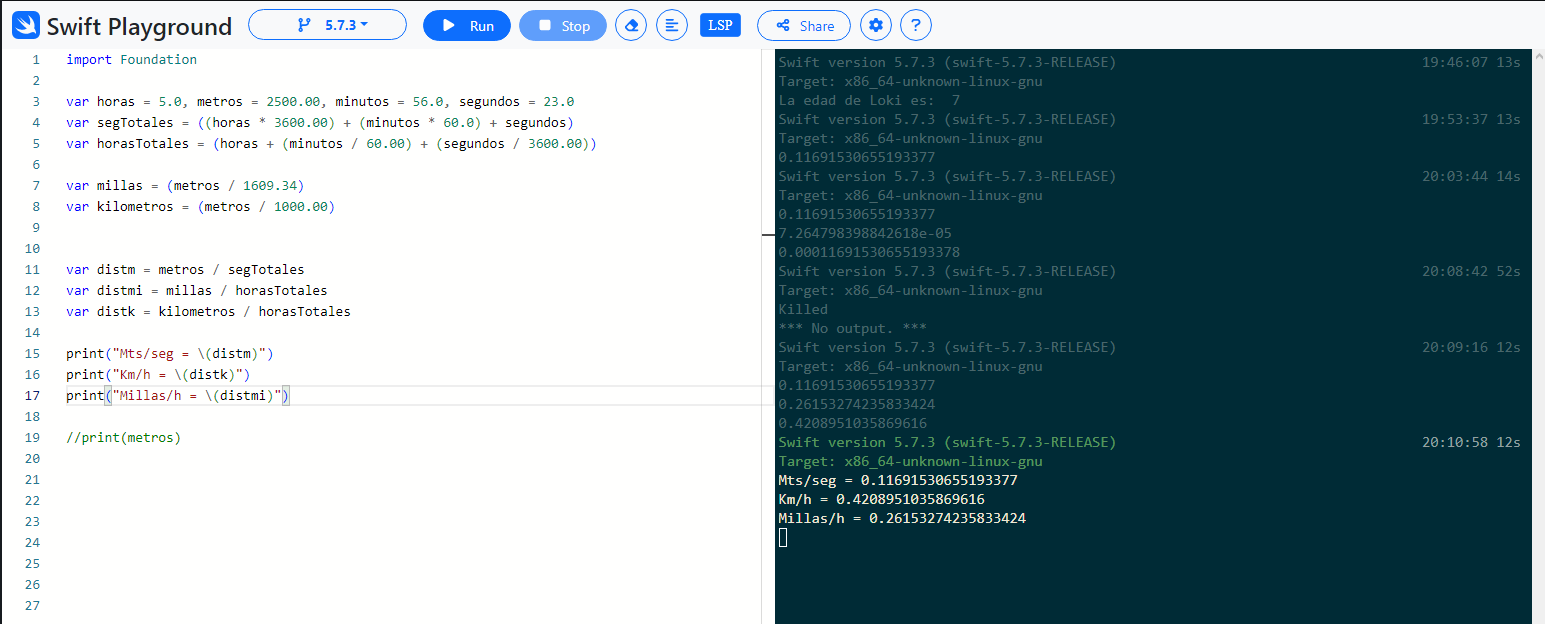


Como se puede percibir la variable es inicializada con el valor de 50 y su método de impresión es el resultado de dividir la variable entre 7.

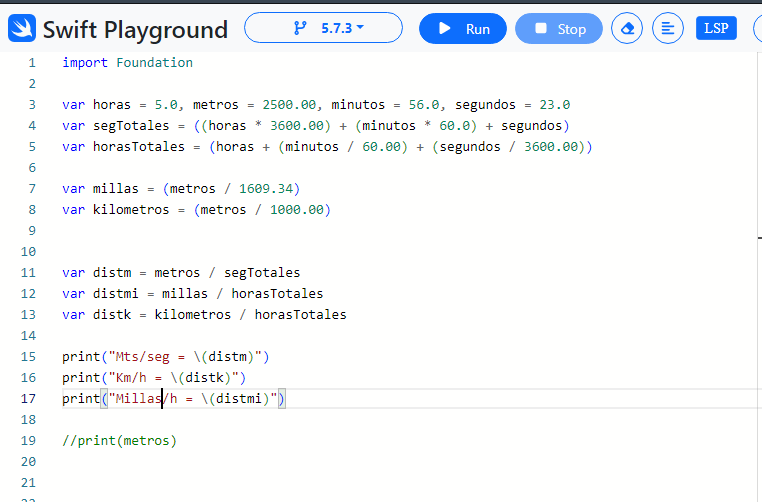


### Practica 2.3.2

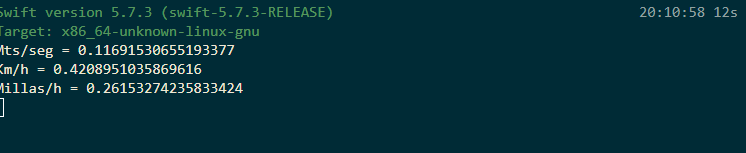
Se requiere almacenar 3 valores para obtener un cálculo de distancia recorrida en cierto tiempo en esta práctica se tomarán cálculos más específicos con datos de tipo real.



Primeramente, definimos las variables brindado uniformidad a los datos otorgando la conversión a segundos y horas de las diversas medidas de tiempo, posteriormente definimos una variable que sume todos estos tiempos para introducirlos en el cálculo de la distancia donde se toman los datos correspondiendo a su debida división y brindado un formato para su resultado esperado.

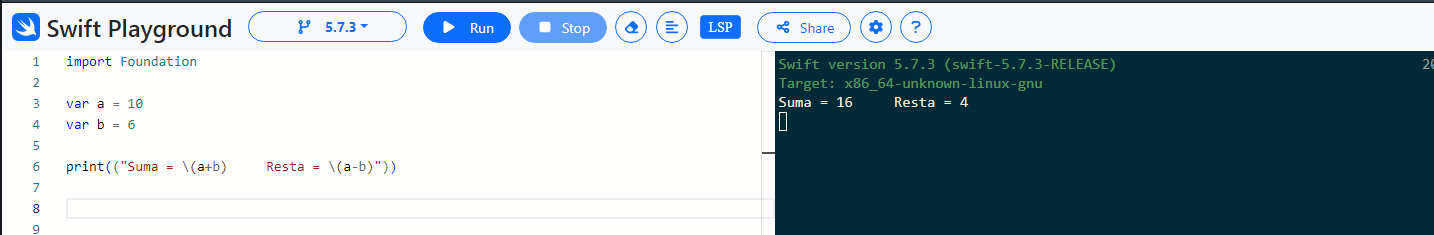


El código empleado para esta práctica define varias variables donde se define la información correspondiente para cada caso y su debida implementación.

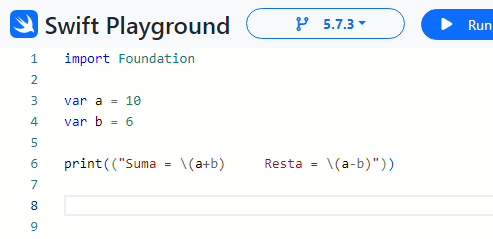


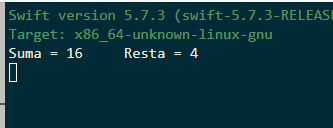
### 2.3.3

Se requiere almacenar 2 variables e imprimir el valor de la suma y diferencia de ambos valores en sus respectivas operaciones.



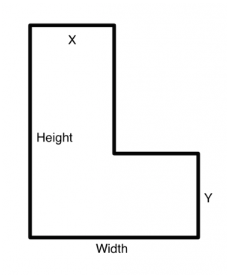
El codigo solamente almacena los valores en las variables a y b dejando a el método de impresión la operación correspondiente.



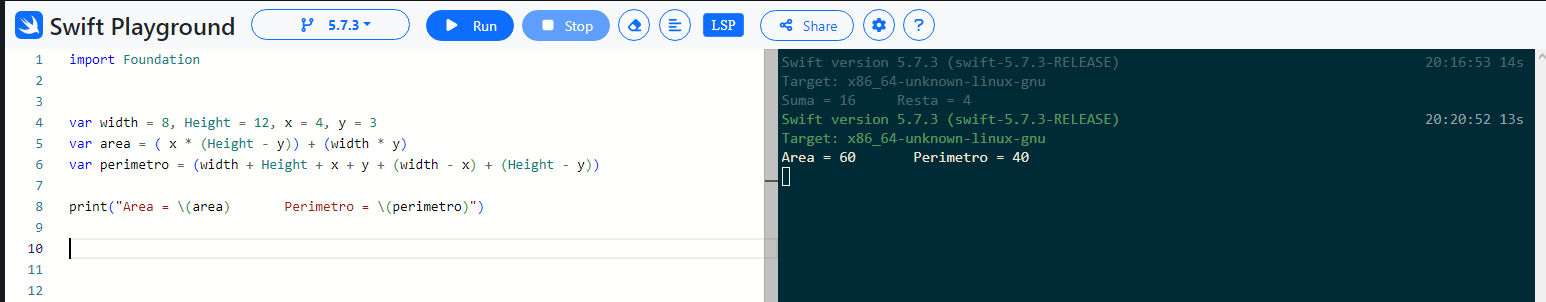


### Practica 2.3.4

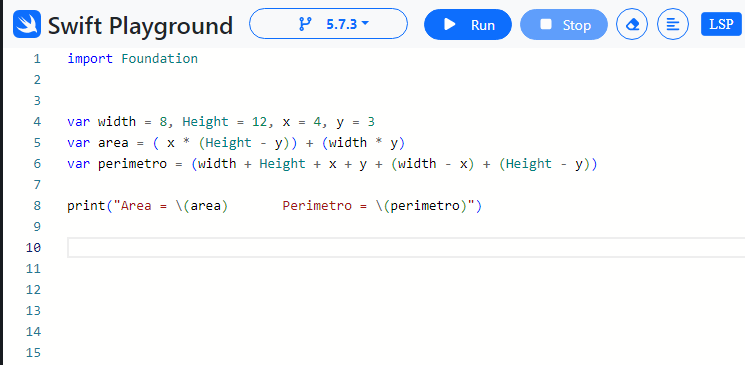
Definiendo 4 valores deberá calcularse el perímetro y área de la siguiente figura.

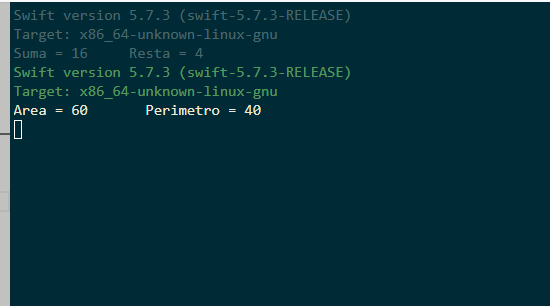
en base a esta figura realizar los cálculos pertinentes.

Primeramente se tienen que definir los valores de cada variable para este caso se emplearan los mismo nombres en las variables, el calculo definido para esta practica es una relación de altura y anchura con su correspondiente medida de manera vertical.



Únicamente con la definición de las variables en 4 variables y el resultado almacenado en otras dos variables es posible imprimir el valor del área y perímetro en la pantalla de resultado.

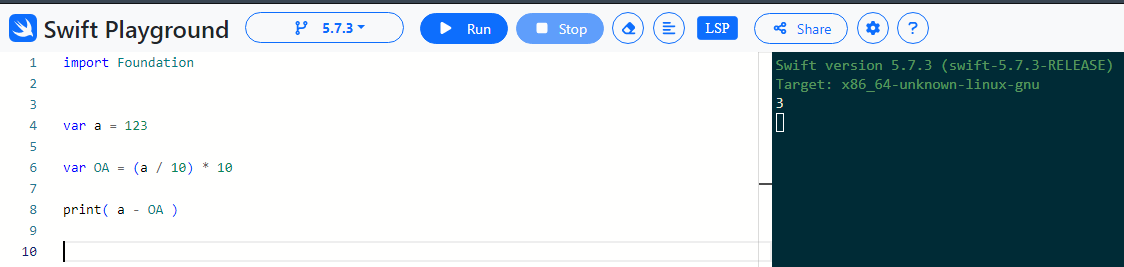




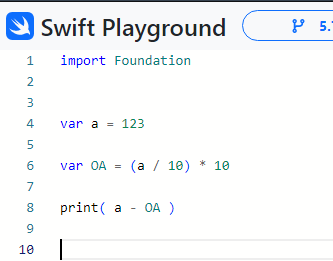
En este caso debemos de considerar la forma irregular del área por lo que no bastara la base regular de la formula base \* altura.

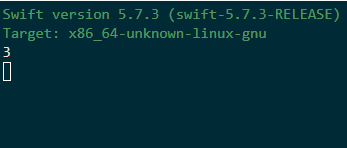
## Practica 2.3.5

Definiendo un valor de 3 dígitos debemos de obtener el ultimo digito de este conjunto.



En esta práctica únicamente se definieron dos variables una donde se almacena el conjunto de dígitos y otro que procede a realizar la operación donde se obtiene un valor similar, pero sin el ultimo digito, al momento de imprimir en consola se resta el valor obtenido al valor original obteniendo así el ultimo digito del conjunto de números.

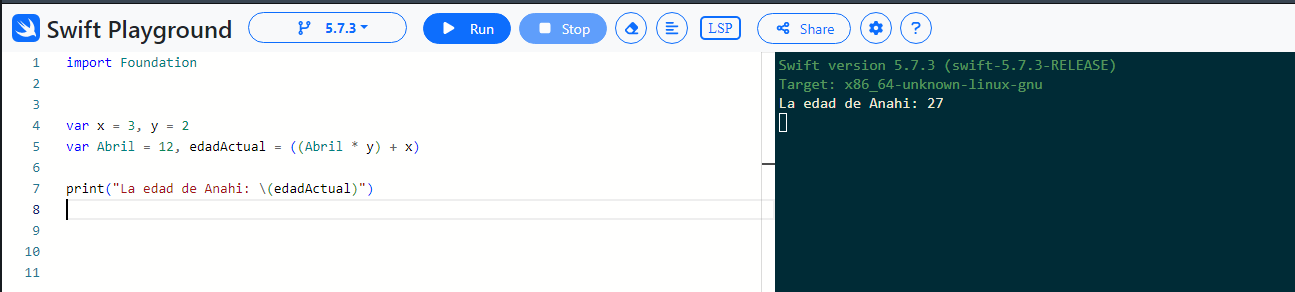




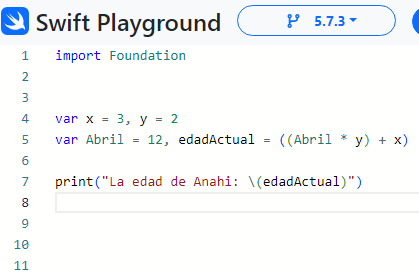
### Practica 2.3.6

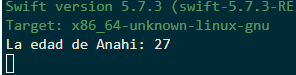
Se requiere calcular la edad de Anahí a partir de la edad de su hermana Abril

Para esta práctica se requiere almacenar la edad de Abril y los años que quieren calcular para obtener el dato a partir de una secuencia de operaciones e imprimir el resultado.



Almacenamos la edad de actual de abril, los años que transcurrirán y los años de diferencia para almacenar el resultado de una operación en una variable e imprimirla.

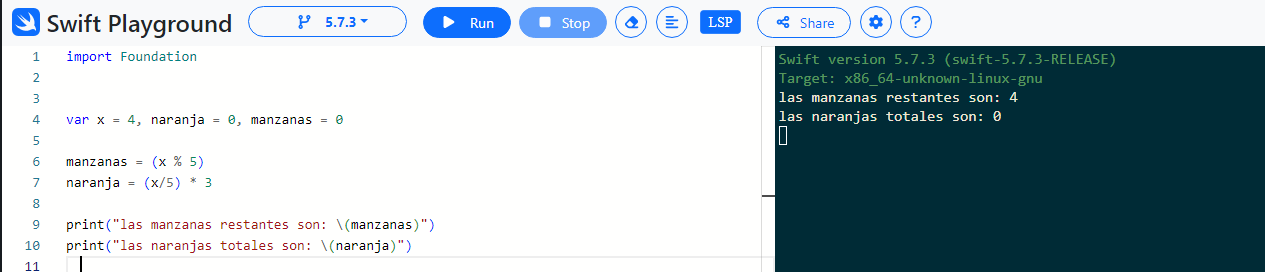




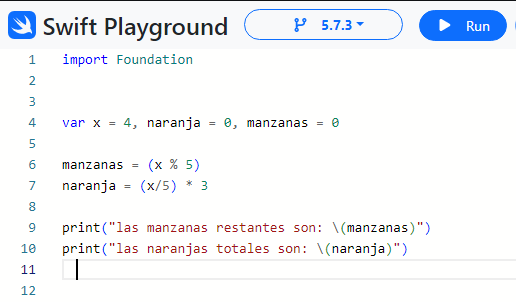
Para tener un resultado favorable es necesario definir una lógica en el sentido del cálculo a partir de lo que se desea obtener.

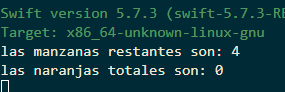
### 2.3.7

Se debe de calcular el resultado de un intercambio de 3 naranjas por 5 manzanas donde se definirán las manzanas que se tienen para calcular el número máximo de naranjas que se puede obtener y el sobrante en manzanas.



Para esta practica se definen tres variables (naranjas, manzanas, x) donde las operaciones de las manzanas restantes seria el residuo modular de x/5 para obtener las manzanas restantes y las naranjas únicamente serán el cociente de x/5 \* 3 que es equivalente al numero de naranjas que se obtendrán

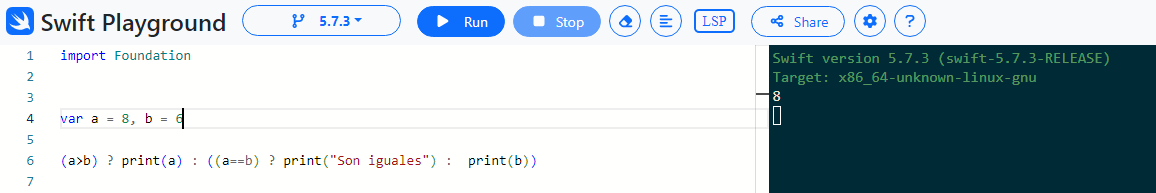




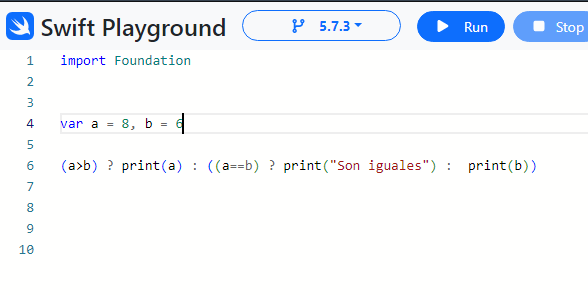
### Practica 2.3.8

Definiendo dos números se deberá de definir el número mayor o si son iguales

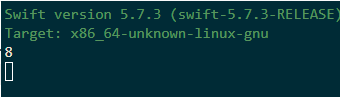
Para la práctica se definieron 2 variables (a, b) y mediante un operador ternario se define cual es mayor o si son iguales.



El Codigo consiste de la definición de dos variables y dos condiciones ternarias anidadas.



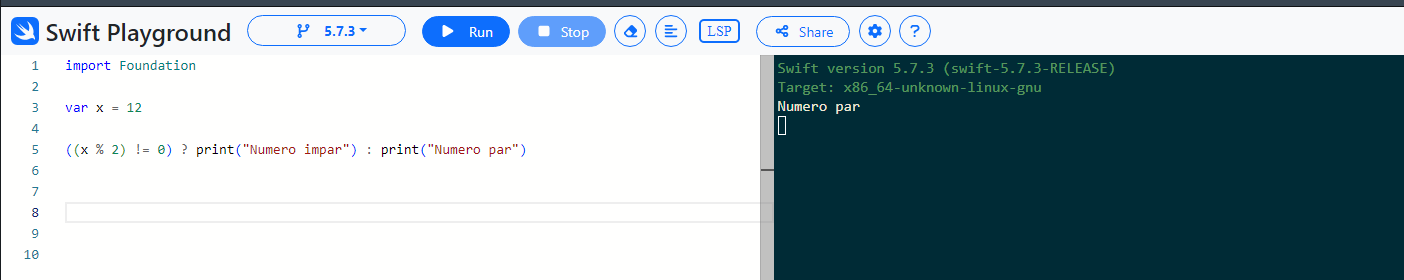
Se deberán considerar los casos y sus respectivos resultados.



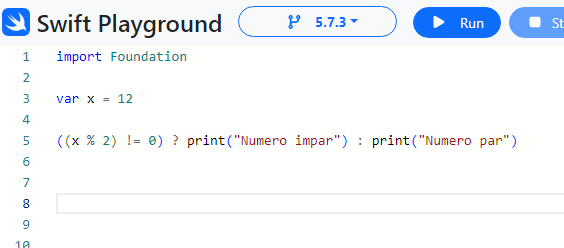
### Practica 2.3.9

Se requiere conocer si X es un numero par o impar

Se deberá de definir una variable (X) con el valor a analizar y mediante una condición ternaria se en la cual se evaluara mediante una división modular se definirá si es par o impar.



El código consiste en la definición de la variable y una condición ternaria, esto es debido a que no existe ambigüedad en las respuestas de esta práctica.

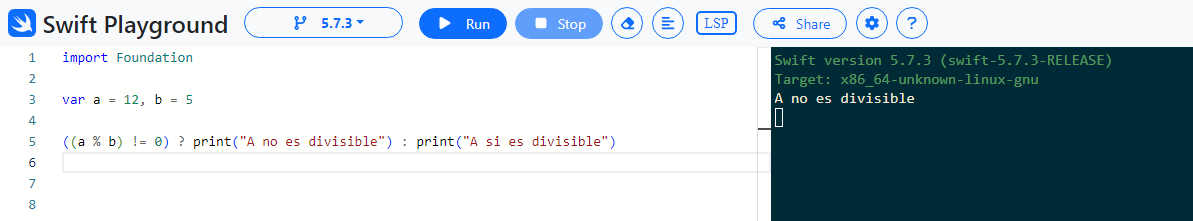




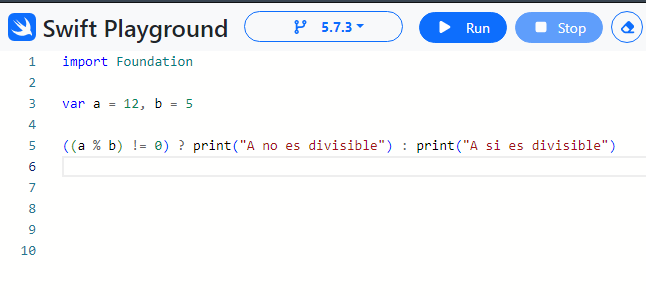
### Practica 2.3.10

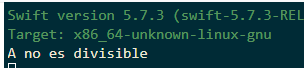
Se requiere conocer si el valor de “a” es divisible entre el valor de “b”.

Para esta practica se deberá de considerar dos resultas si es “Divisible” o si “no es divisible”, por lo que la aplicación de una condición ternaria reduce el código a aplicar.



El código consta de 2 líneas funcionales la primera que define las dos variables y la condición ternaria que efectúa la impresión del resultado dependiendo de los dos casos.

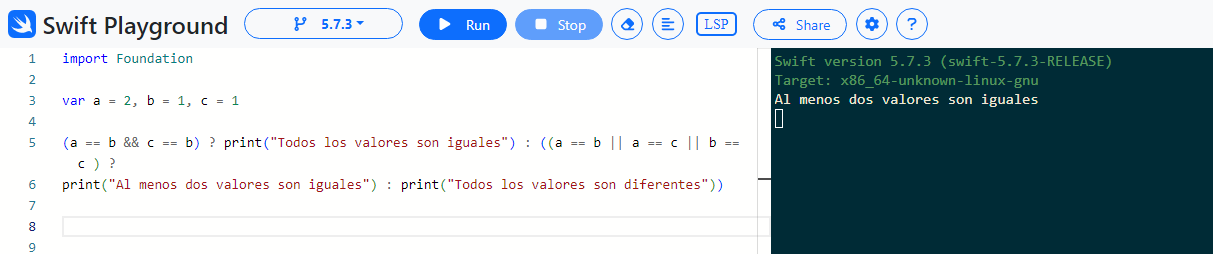




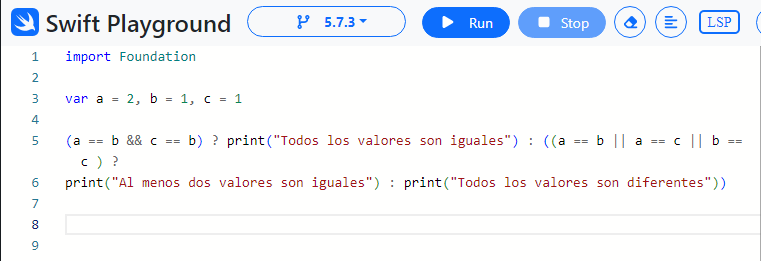
### Practica 2.3.11

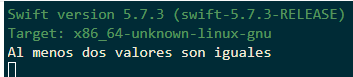
Se requiere definir si al menos dos valores o todas las variables son iguales, en caso contrario dictar que todas las variables son diferentes.

Se almacenarán tres variables y mediante 2 condiciones ternarias anidadas se brindará un resultado.



Consta de 3 variables donde se almacenan los valores iniciales y mediante dos condiciones ternarias se puede concluir las tres condiciones la primera definirá si las tres variables son iguales, la segunda evaluara si al menos dos son iguales en caso contrario se efectuará el else de la segunda condición.



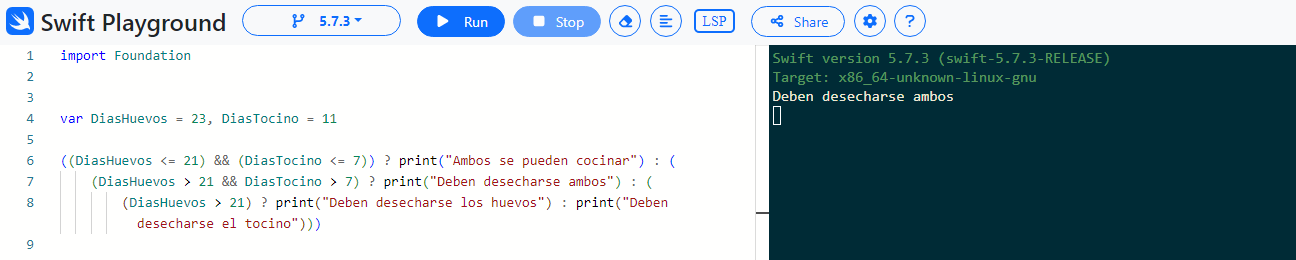


Para esta practica se deberá de considerar como aplicar el razonamiento y dirección de cada caso en los if y else que se aplican mediante la condición ternaria.

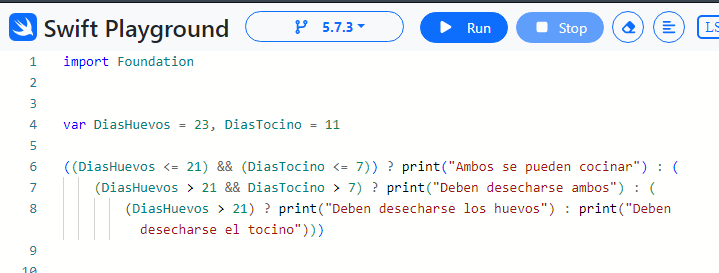
### Practica 2.3.12

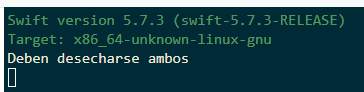
Se requiere definir si dos elementos pueden o no cocinarse o en caso de que uno no se pueda consumir definir cual

Para esta practica se definirán dos variables y 3 condiciones ternarias anidadas.



Definiremos el valor de los días en ambas variables y en base a la informacion brindada definiremos mediante tres condiciones ternarias si es viable cocinar ambos, tirar un elemento o tirar ambos, por lo que en caso de no cumplir la primera condición (ambos son validos) se descartara de inmediato y continuara con la condición de si un elemento es inservible.

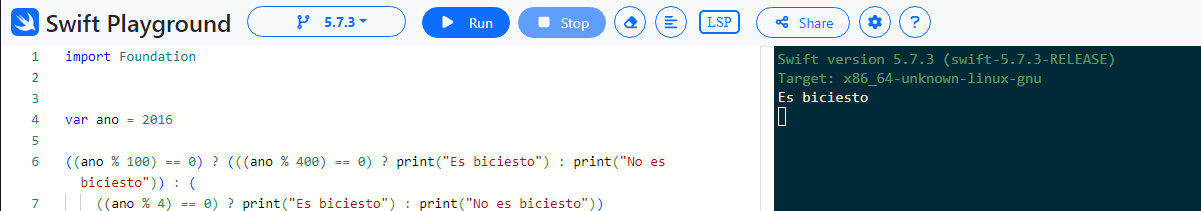




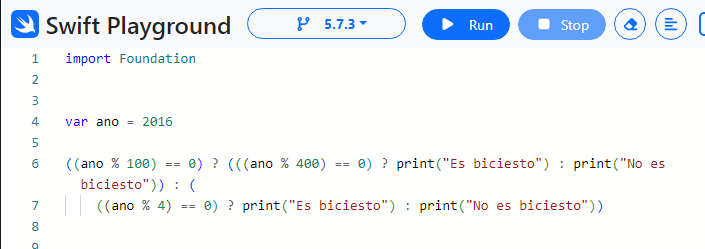
### Practica 2.3.13

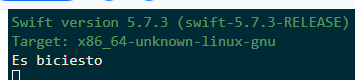
Dado un año se deberá definir si este es bisiesto o no.

Para esta practica deberemos de definir una variable (año) y tres condiciones ternarias anidadas.



Lo primero es definir el valor del año a analizar y consecutivamente se deberá de aplicar dos condiciones ternarias anidadas para definir si es o no bisiesto en base a el cumplimiento de una condición si es múltiplo de 400 en caso de no cumplir se deberá de analizar si es simplemente múltiplo de 4.

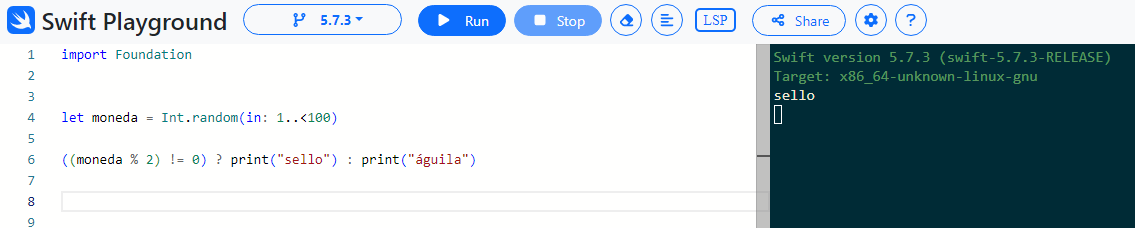




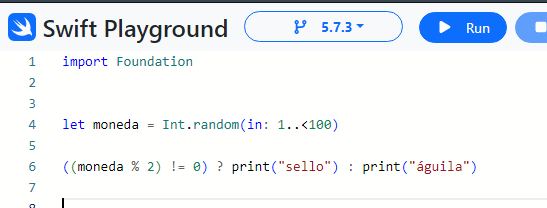
### Practica 2.3.14

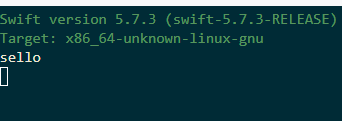
Se requiere emplear random para obtener un numero aleatorio y definir en base a si es par o impar su igualdad en agila y sello.

Con solo una variable que almacené el valor random y una condición ternaria que efectué cada condición dada (par = águila e impar = sello)



Únicamente se aplicara random para obtener un valor aleatorio y almacenarlo en la variable para posteriormente introducirla en la condición ternaria en base a la división modular de la variable antes definida.

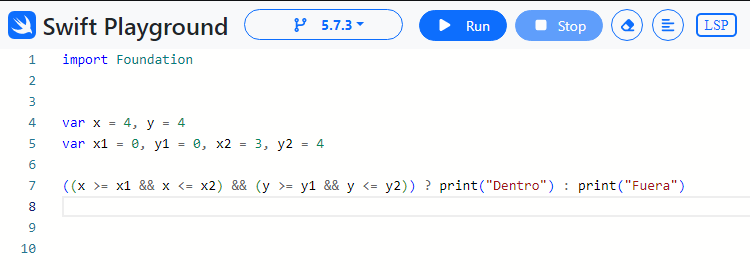


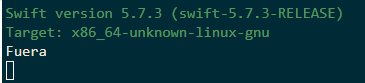


### Practica 2.3.15

En base a dos variables definir si el punto generado (en base a un mapa cartesiano) se encuentra dentro o fuera de un área definida.

Para esta práctica se definirán 6 variables 4 para los puntos de referencia, y 2 para el punto cardinal a evaluar, esto se someterá a una condición ternaria compuesta por los 4 puntos valorados.



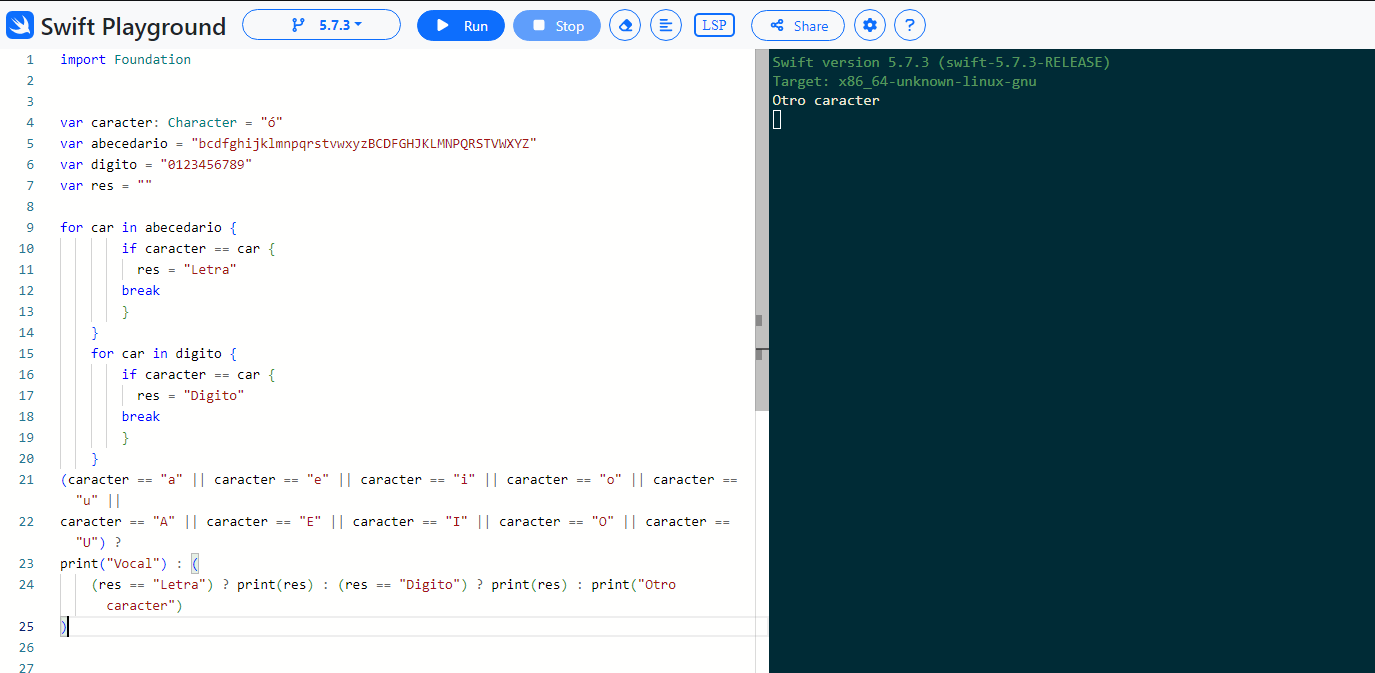


La elección de encontrarse dentro solo se compone de dos partes válidas y cada una a su vez se divide en dos condiciones bajo la relación de un AND para encontrar si esta entre los márgenes de cada punto.

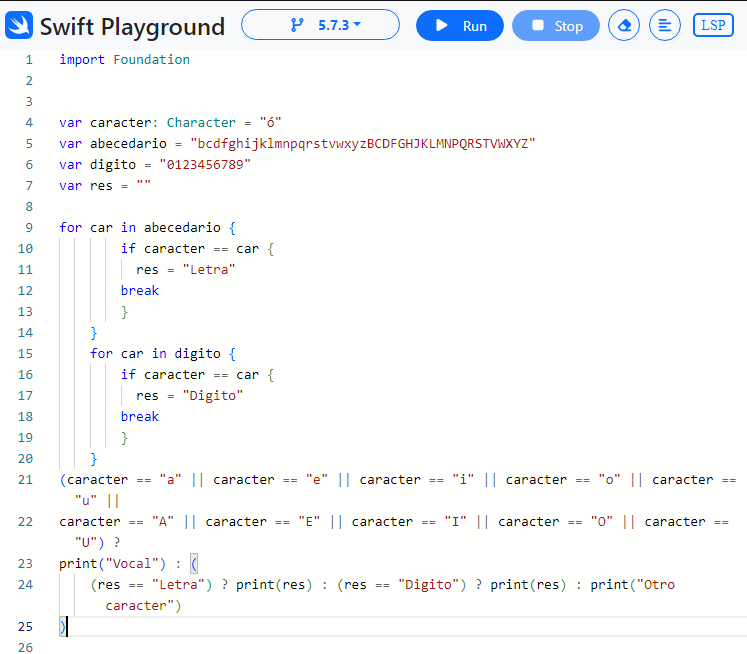
### Practica 2.3.16

Se requiere analizar y verificar si un carácter es un digito, vocal, letra u otro.

Para esta práctica se abordó un punto de vista básico donde solo se definen cadenas, con los carates válidos y se recorren para encontrar coincidencias con el carácter dado para posteriormente someterlo a tres condiciones ternarias anidadas que categorizan el resultado en 4 estados (vocales, letras, dígitos y otros)



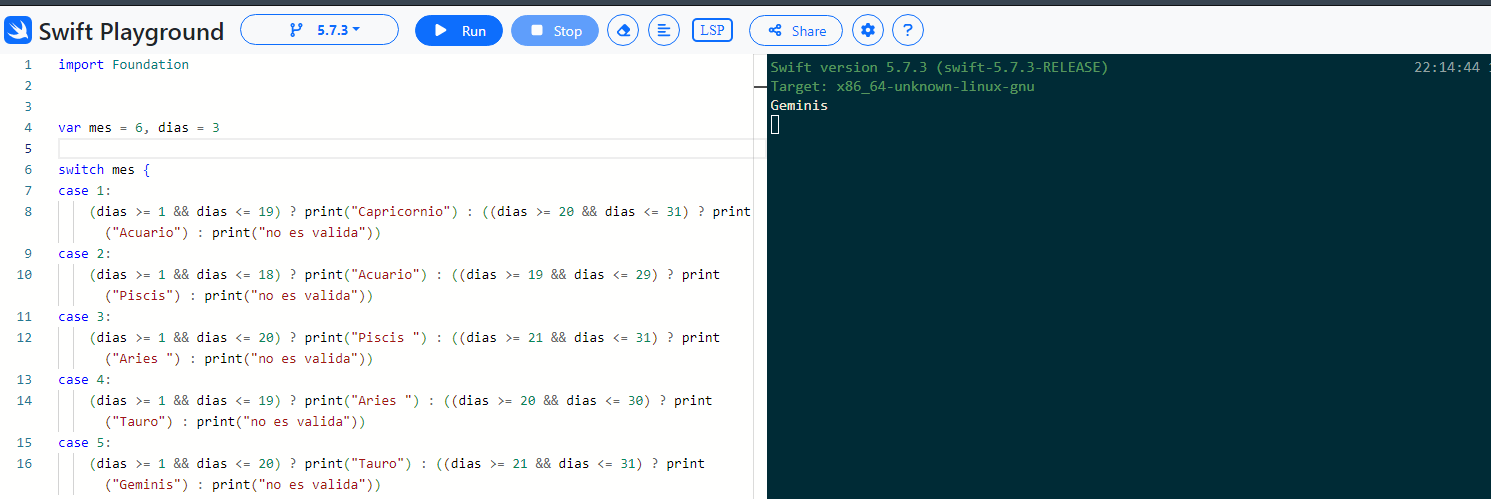
Afrontando el carácter como un pivote en las listas se define si se encuentra categorizado en digito o letra posteriormente se somete a una condición ternaria que imprime en caso de ser vocal, en caso de no serlo se somete a la segunda condición que efectúa si el resultado es letra en caso de no serlo se somete a la última condición que especifica si es digito de no serlo se categorizara como otro.



### Practica 2.3.17

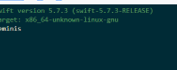
En base a un mes y día brindados deberá de definirse el signo zodiacal de la fecha.

Para esta práctica se efectuará el uso de un switch con sus diversos casos y que contienen una condición ternaria dentro de cada caso.



Consta de 12 casos uno por cada mes y dentro de cada caso se encuentra un rango de fechas definido por una condición ternaria, dependiendo del valor del mes es el caso que se seleccionara y en base a la fecha, esta brindara un resultado.





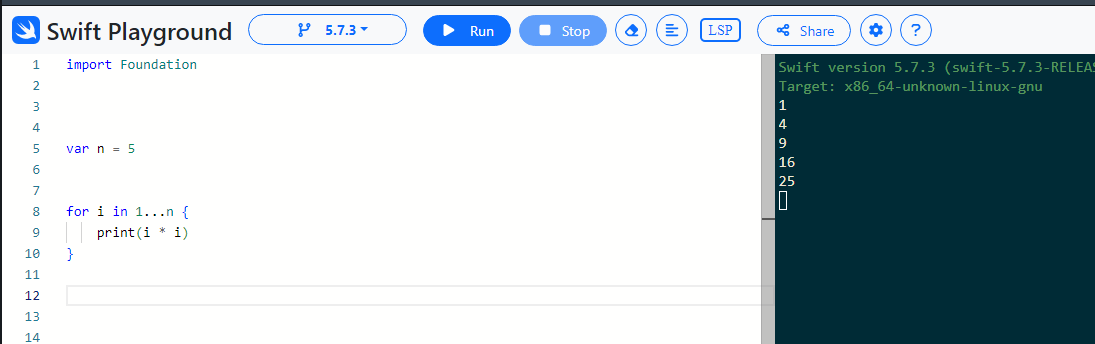
(imagen del editor visual studio code, debido a la mejor apreciación del codigo)

Se deberán de mantener en cuenta el rango valido de cada signo zodiacal.

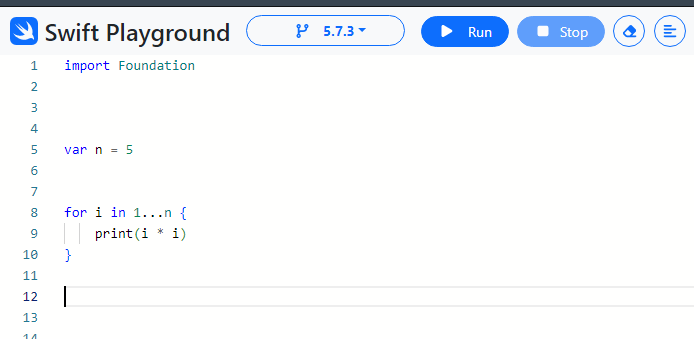
### Practica 2.3.18

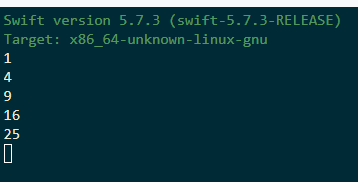
Se requiere imprimir la lista de los cuadrados de cada número hasta llegar al número brindado.

Para esta práctica se empleó un ciclo for el cual imprime el cuadrado de cada número hasta llegar al número brindado.



Almacenamos el valor dentro de una variable la cual será sometida a un ciclo for que efectuará el total de los pasos desde el primer número hasta que llegue al número definido en la variable, por cada vuelta dentro del ciclo se imprimirá el cuadrado del número en curso.

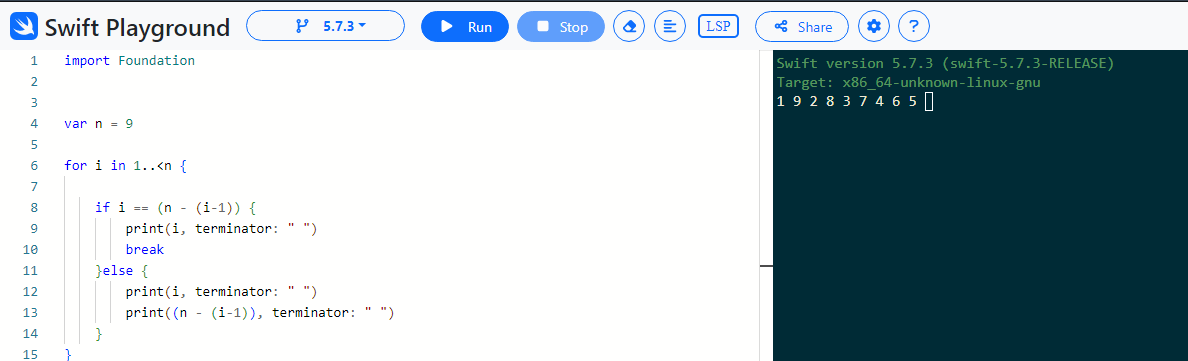




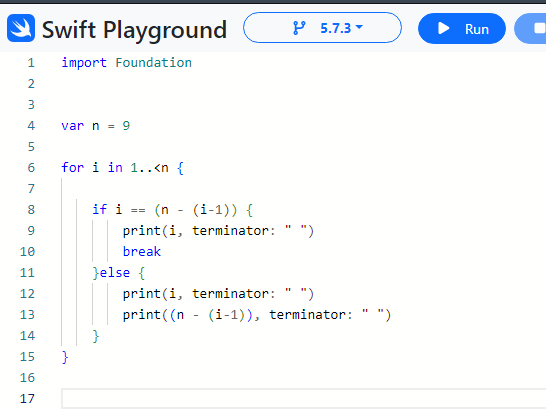
### Practica 2.3.19

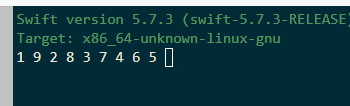
Se requiere imprimir una cadena que cumpla con las siguientes características deberá de contener el siguiente patrón 1, N, 2, N-1,…. Imprimiendo todo en un renglón.

Para esta practica se empleara un ciclo for que contiene una condición if-else que efectuara la impresión de la cadena con las características requeridas.



Para esta practica se empleara el ciclo que se repite N veces y tomara la condición de imprimir de manera interna con una condición base que define si en caso de que el iterador “i” sea igual al valor de “ (n – (i – 1))” que es valor medio del ciclo este imprimirá únicamente el valor medio en caso de no cumplirse este imprimirá el valor del iterador y el valor de “(n-(i-1))” con la terminación de cadena de un espacio en blanco.

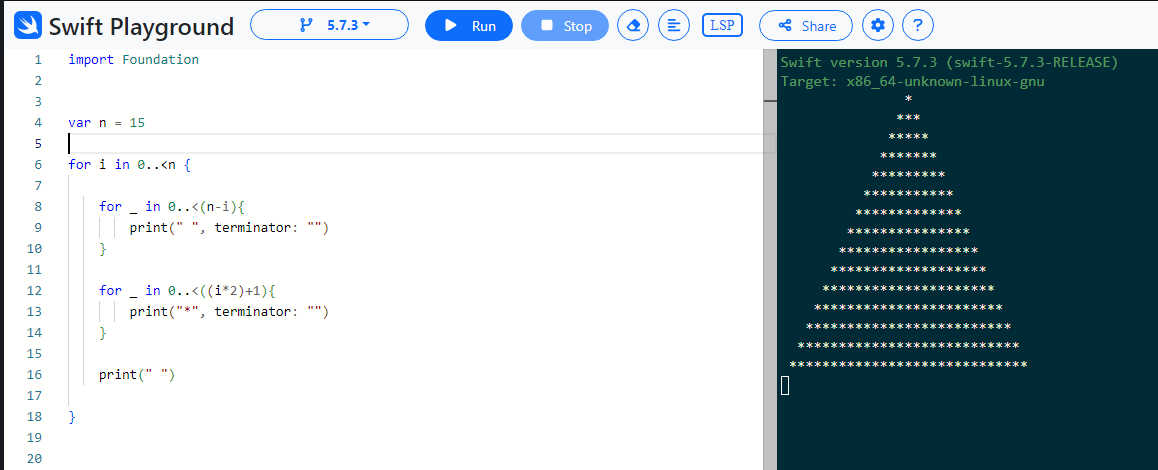




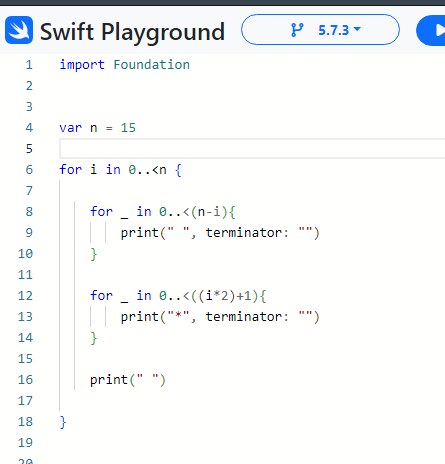
### Practica 2.3.20

Se requiere imprimir la figura de una pirámide realizada con los caracteres (\*).

Esta practica efectura el uso de tres ciclos for el primero define la altura y este contien 2 mas que definen el ancho y el salto de línea.



El valor de n es un valor brindado por el usuario que define la altura de la figura, el primer ciclo definirá las iteraciones realizadas para brindar esta característica, el segundo for que se encuentra dentro del primer ciclo define los espacios en blanco, necesarios para dar forma y el tercer ciclo (también definido dentro del primer ciclo) define los caracteres (\*) que se verán en pantalla, ya que definen la figura total, la ultima instrucción imprime el salto de línea.

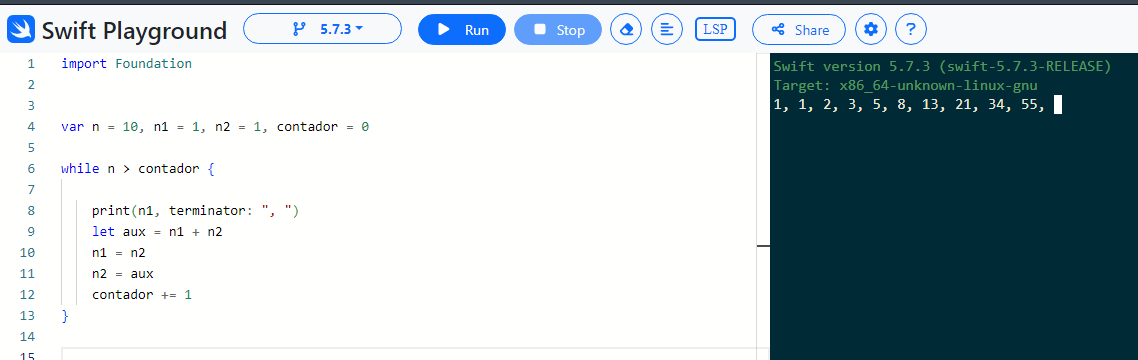




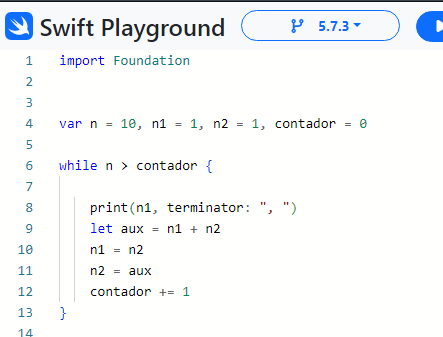
### Practica 2.3.21

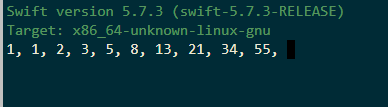
Se requiere imprimir el valor de la serie Fibonacci de un numero N

Se requiere brindar un valor como meta, e inicializar los valores de n1 y n2 que siempre serán igual a uno posteriormente se dictara el patrón del número actual más el número anterior.



Empleando el uso de 4 variables n el número brindado (meta), n1 y n2 que son los números brindados y un contador que es igual a 0, empleando el uso de un ciclo while se compara que el valor de n sea mayor al valor del contador dentro de este se imprime el valor de n1 con la terminación de línea en “,”, empleamos una constante como auxiliar que tomara el valor siguiente mientras que n1 obtiene el valor anterior (actual en la impresión) y n2 obtiene el valor siguiente el contador se incrementa y continua este ciclo.

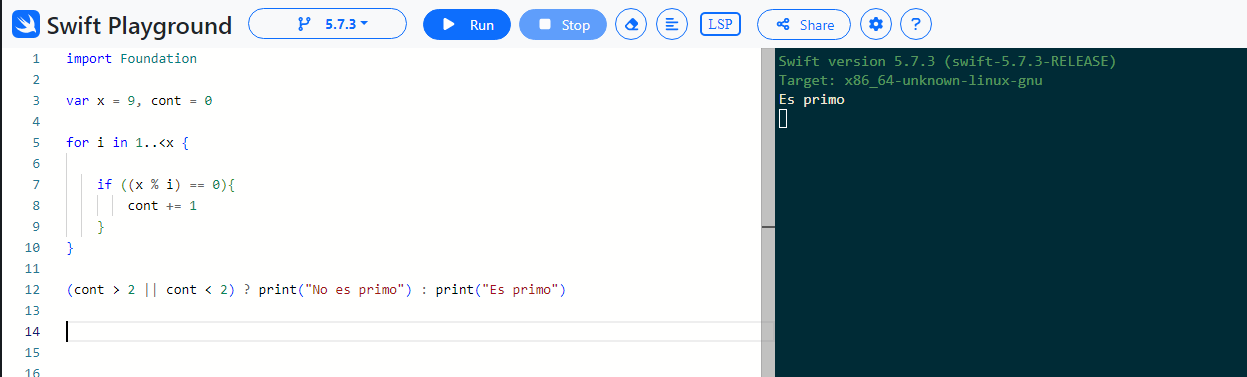




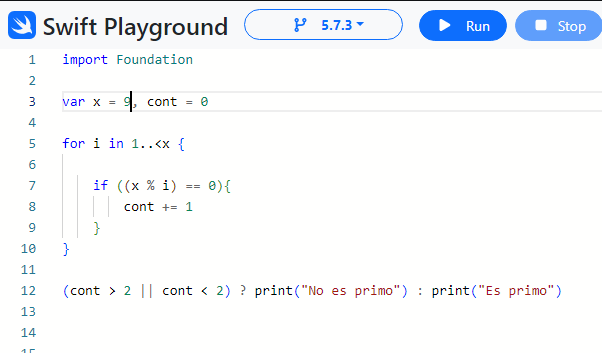
### Practica 2.3.22

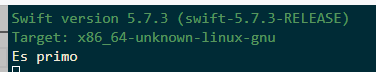
Imprimir si un valor brindado es primo.

Empleando dos variables, un ciclo for, un condicional if y una condición ternaria se puede obtener el resultado de si un número es primo o no.



Primeramente almacenamos el valor que se brinda en la variable x e inicializamos un contador en 0 estas dos variables se someterán a un ciclo for que se encargara de realizar una división modular en base a cada iteración realizada por lo que en caso de ser exacta esta incrementara el contador en +1, finalmente al teminar el ciclo se someterá a una condición ternaria que verificara si el contador es mayor a 2 o menor a 2 (es diferente de 2) este no es primo en caso contrario el número es primo.



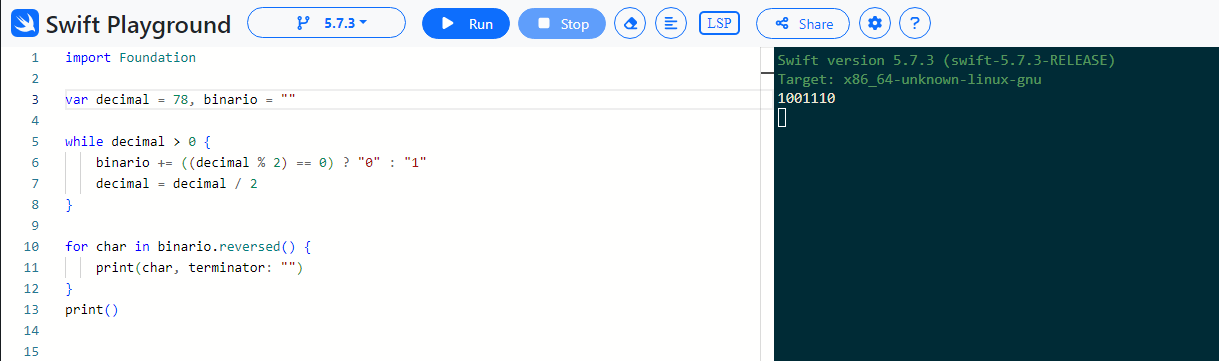


Se deberá de considerar que los números primos solo son divisibles entre 1 y si mismos.

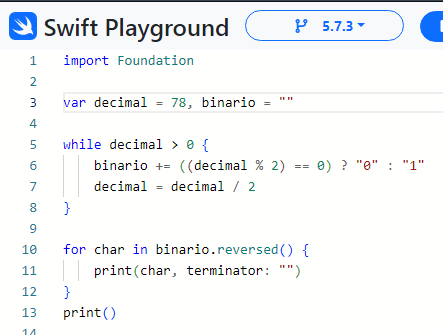
### Practica 2.3.23

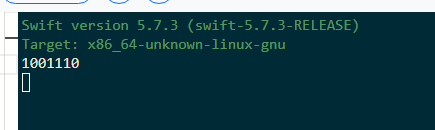
Se requiere realizar la conversión de un numero decimal a binario empleando el método escalera.

Para esta práctica se deberá de realizar divisiones modulares y realizar un reordenamiento para generar el formato correcto.



El Codigo consiste en 2 variables, la primera el valor en decimal, la segunda es una cadena vacía, después de definir las variables entrara en un ciclo while que definirá la condición de que siempre que el valor decimal sea mayor a 0 realizara una concatenación a la cadena bajo la condición ternaria de una división modular de la variable decimal entre 2 si es exacta se concatena un 0 en caso contrario se concatena un 1 y se divide el valor de decimal entre 2, seguidamente se somete a un ciclo for donde se recorre la cadena de manera inversa y se imprime el carácter con una terminación de impresión vacía.



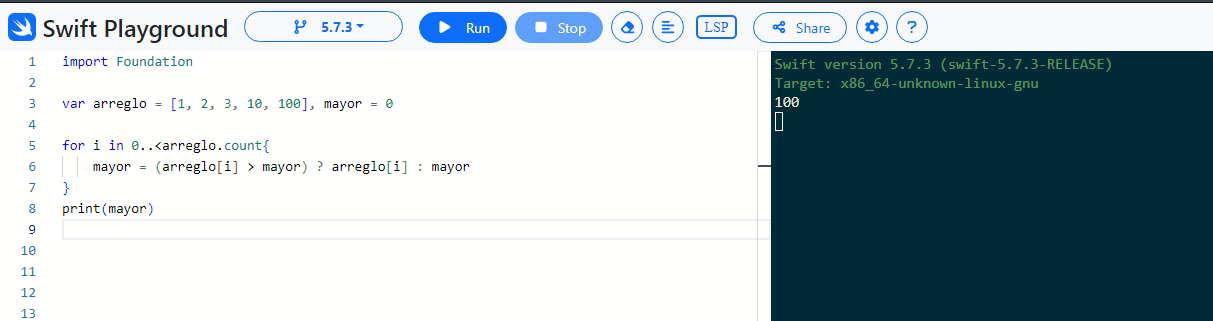


Se deberá de considerar el método de división en escalera el cual consiste en que si una división de un numero entre 2 no es exacta se coloca un 1 y si lo es se coloca un 0 finalmente se junta todo empezando por la ultima división y terminando con la primera (solamente residuos).

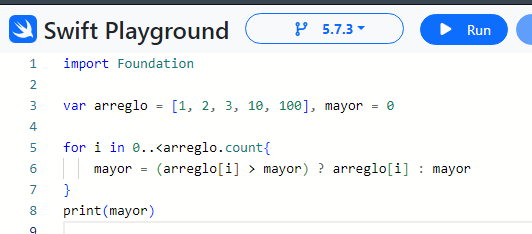
### Practica 2.3.24

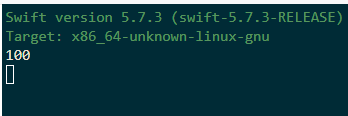
Se requiere imprimir el número mayor dentro de un arreglo.

Para esta práctica se empleará el uso de los arreglos (arrays) y un for para su recorrido, de esta manera verificaremos internamente si el número mayor es el almacenado o el que está actualmente en la posición del arreglo.



Empleando un arreglo, un for y una condición ternaria se puede realizar la actividad, principalmente se define el arreglo, posteriormente se induce al ciclo for donde internamente se somete a la condición ternaria que valida si el valor actual del arreglo es mayor al número almacenado como mayor de ser el caso correcto se almacenara el nuevo valor, en caso contrario se almacenara el mismo valor.

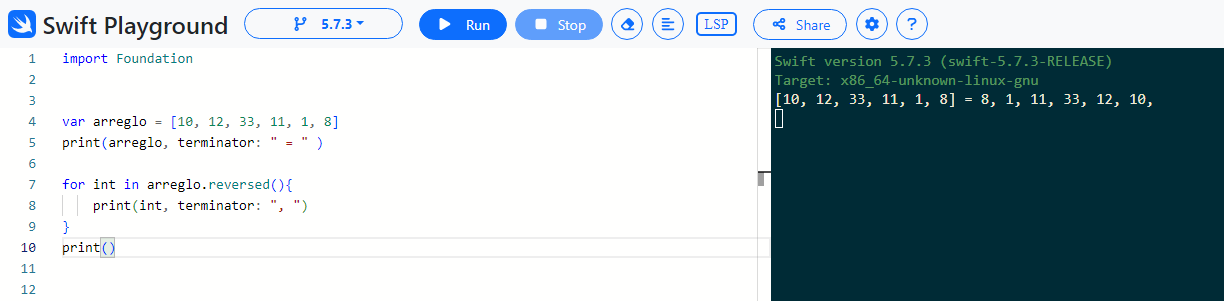




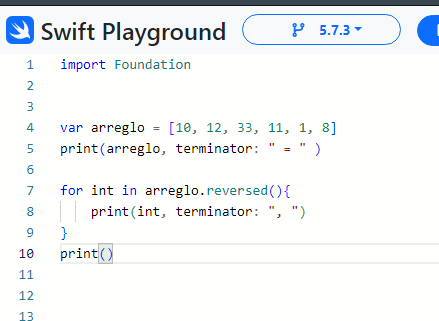
### Practica 2.3.25

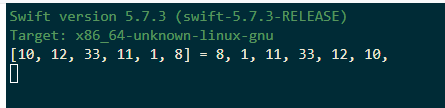
Se requiere imprimir un vector de manera inversa al que se encuentra almacenada.

Para esta práctica se empleara un ciclo for en el cual se imprimirán los valores correspondientes.



Principalmente se define el arreglo seguido de esto se somete a un ciclo for donde su tope será inverso mediante el método reversed() el cual invierte el ciclo comenzando por el final, procede a iniciar el ciclo y únicamente se imprime el valor con una terminación de “,”.

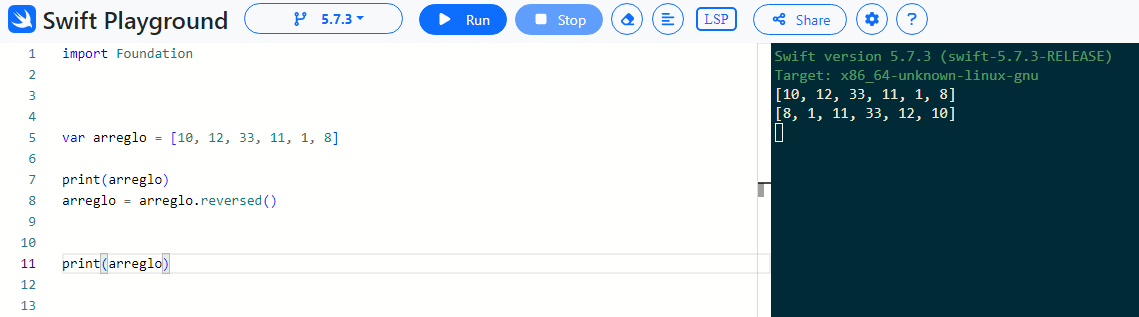




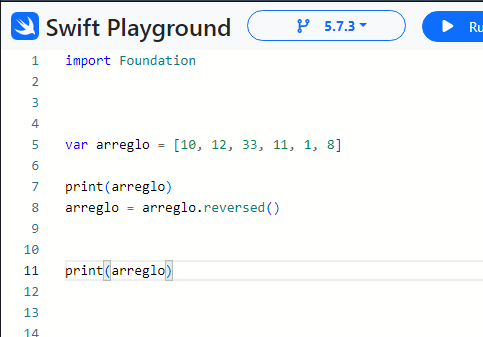
### Practica 2.3.26

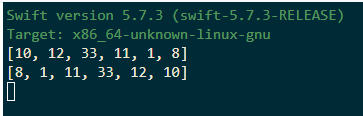
Se requiere de invertir un arreglo sin crear otro.

Se empleara el uso del método reversed().



Se define el arreglo, seguido de esto imprime el arreglo original, se almacena el valor del arreglo a el arreglo con el método de reversed(), se imprime el arreglo alterado.

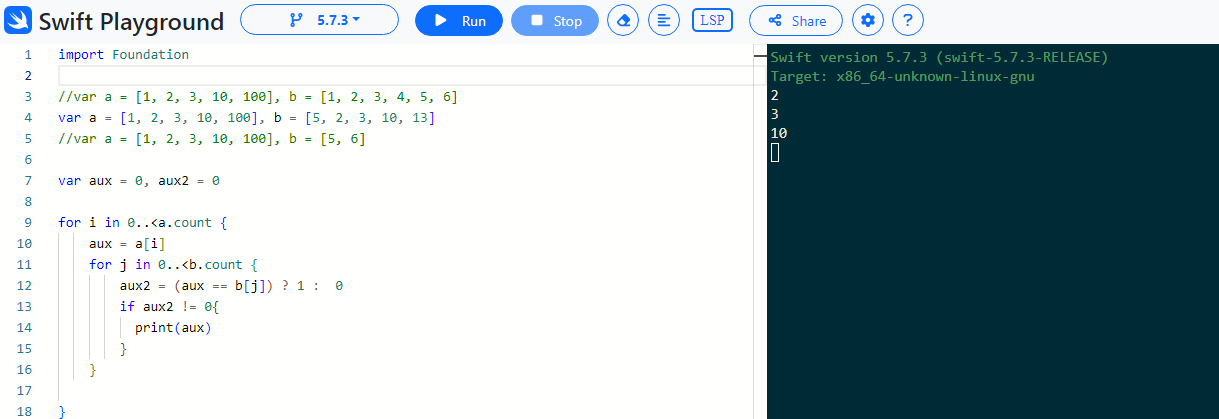




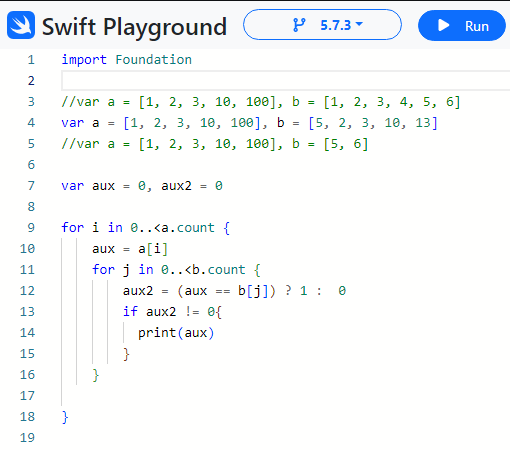
### Practica 2.3.27

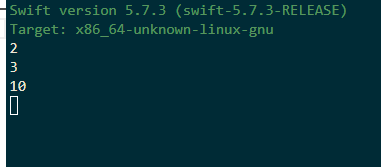
Se requiere imprimir los valores en la intersección de los arreglos A y B.

Para esta práctica se requiere imprimir los valores que se encuentran en la intersección de A y B por lo cual se emplearan 2 ciclos for uno que recorre el arreglo A y otro que recorre el arreglo B.



Principalmente se definen los 2 arreglos posteriormente se definirán 2 variables como auxiliares, seguido de esto se someten a un ciclo de recorrido del arreglo tomando como aux el valor externo, después se induce al segundo ciclo se aplica una condición ternaria en la cual si el valor actual de arreglo B es igual al valor aux este almacenara en la aux2 el valor 1 o 0, se induce a una condición en la que se verifica el valor de aux2 sea diferente de 0 en caso de serlo se imprime el valor de la aux

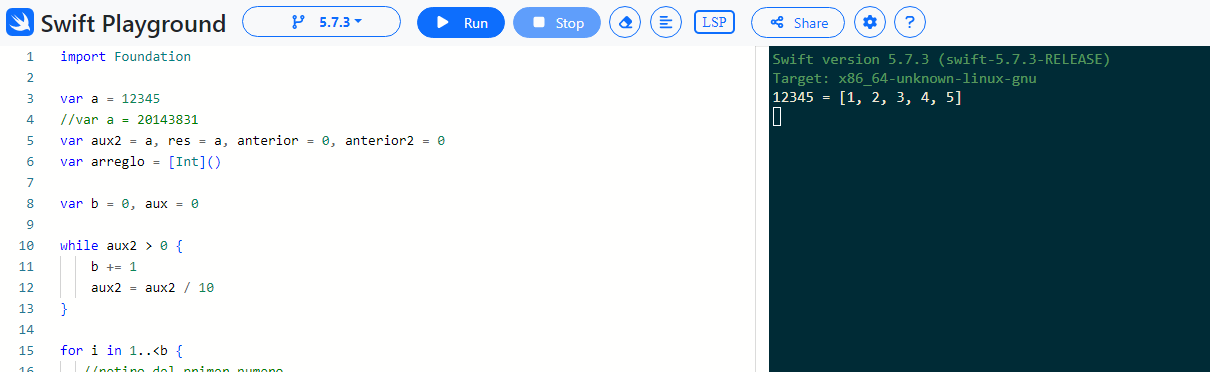




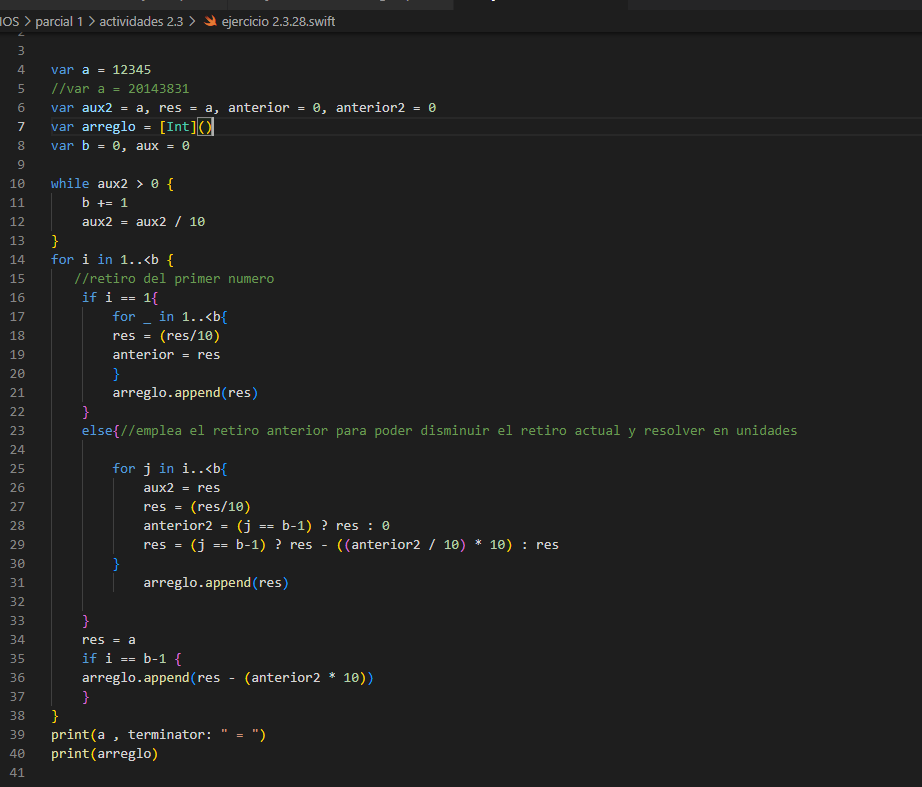
### Practica 2.3.28

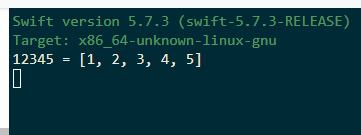
Se requiere separar cada digito en un arreglo son convertir a cadena y de manera aritmética.

Con base a un número dado este se separará de manera aritmética mediante operaciones para brindar una solución en la que se reducirán los valores de derecha a izquierda dejando el valor anterior para eliminar el valor consecutivo en el valor actual que toma 1 digito más a la derecha.



Principalmente se definen las variables, posteriormente se define un arreglo vacío, se contabilizan los dígitos como caracteres (de uno en uno) mediante un ciclo while continua con la inducción a un ciclo for en el que se toma como primer valor el recorrido de la disminución del valor entre 10 al contener el primer valor este entrará en otro ciclo for el cual reducirá todo hasta el primer digito del número a la izquierda, dejando el valor unitario y almacenándolo en el arreglo, consecutivamente entrara a un tercer ciclo el cual realizara la misma operación pero tomara en cuenta el recorrido de cada variable externa para posicionar un digito más en a la derecha y de esta manera recorrer el numero de una manera dinámica para obtener el valor de cada digito.

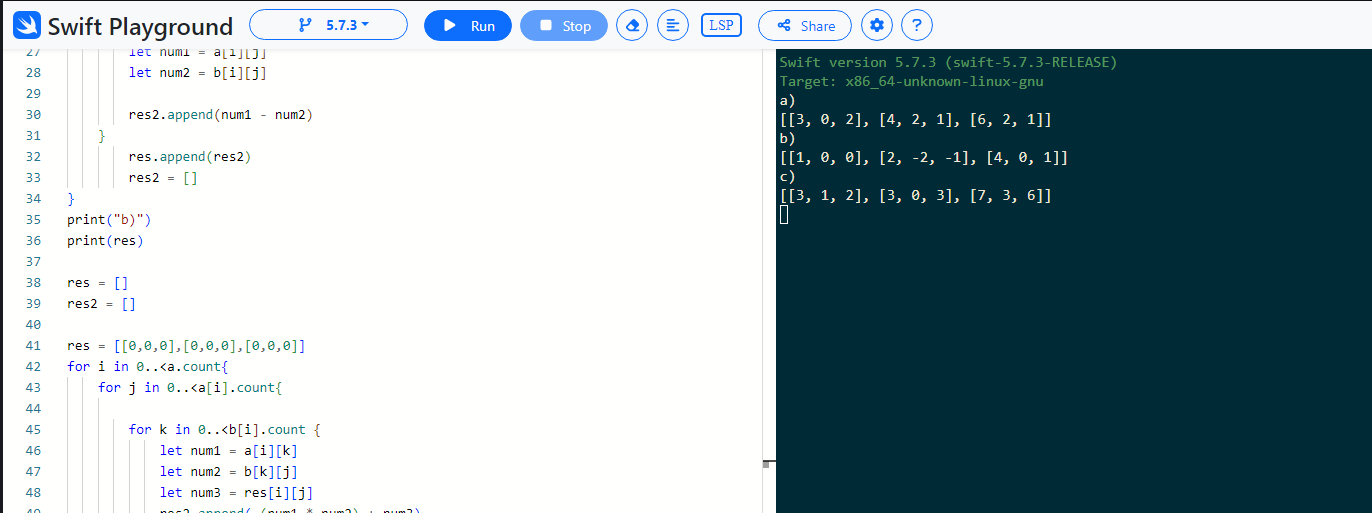




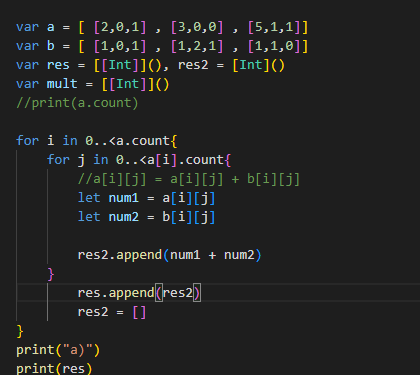
### Practica 2.3.29

Dadas 2 matrices calcular la suma de ambas, resta y multiplicación.

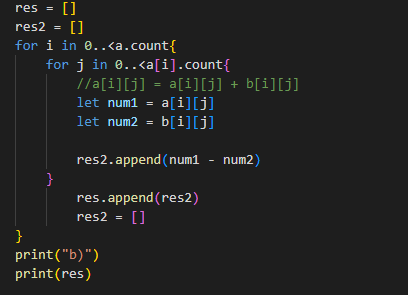
La suma de las matrices se realizará por fila y columna (posiciones dentro de la matriz) de una en una debido que la suma es el valor de la posición actual más el valor de la posición igual en la matriz contraria.



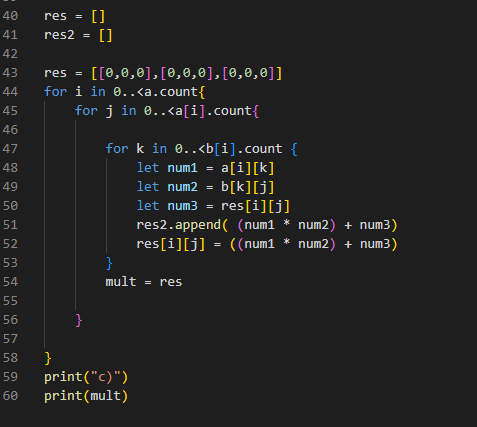
Correspondiente a la suma solo se recorre la matriz mediante dos ciclos for anidados que recorren las posiciones de manera simultanea y suman las casillas en la misma posisicon.

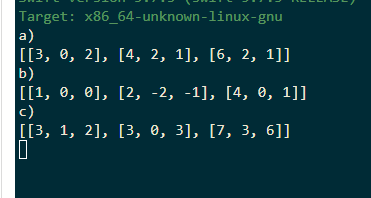


Correspondiente a la resta es similar a la suma únicamente se restan de manera simultanea la posición actual de cada casilla (es la misma en ambas matrices).



Para la referente a la multiplicación se tomaran3 ciclos for anidados donde se recorren columna en el primero, fila en el segundo y en el tercero se recorrerá la columna del segundo arreglo con esto se podrán multiplicar los valores en las mimas casillas y sumar los valore restantes en la fila.



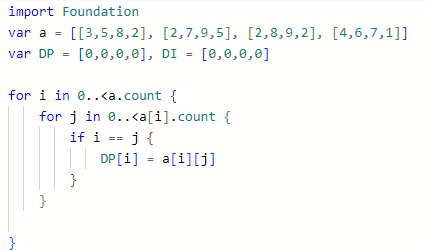


(la notación de cada inciso)

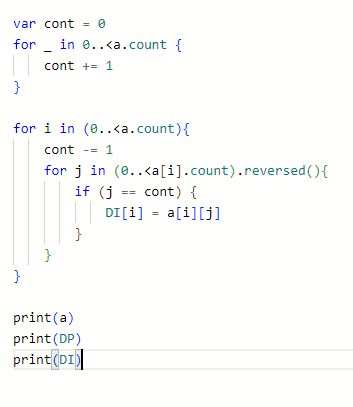
### Practica 2.3.30

Dados una matriz cuadrada A imprimir los valores en la diagonal y diagonal inversa.

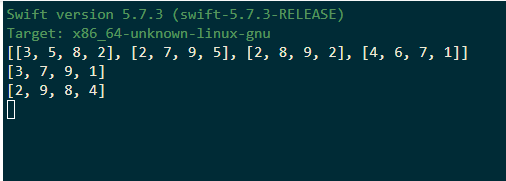
Se hace uso de dos ciclos for para el recorrido de la matriz en cada caso para el inverso y el regular, en el caso regular solo se enfoca en los valores de fila y columna iguales, en el caso del inverso únicamente se recorre la fila regular pero la columna es inversa mediante el método reversed().



Aquí se imprime la diagonal regular donde solo se recorre la matriz mediante dos ciclos for anidados y una condición selecciona la casilla donde la fila y columna sean iguales.



En la inversa se hace uso de un contador para delimitar el máximo de la columna donde se delimita para conocer el valor de la fila y columna igual en este caso es similar al anterior pero las referencias son el cont (valor máximo de columna) se encuentra al final y la fila se encuentra al final inicia la impresión se mantiene este patrón reduciendo el cont cada vez que se cicle una columna.

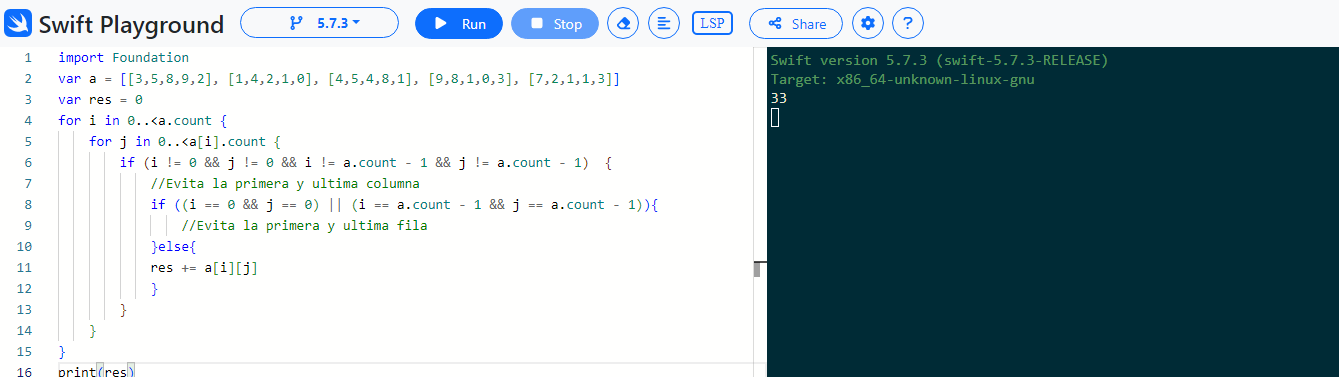


Deberá de pre visualizarse la matriz de una manera inversa a la regular y no se puede aplicar el mismo método en los dos debido a la posición anormal de cada casilla ya que son inversos ambos métodos.

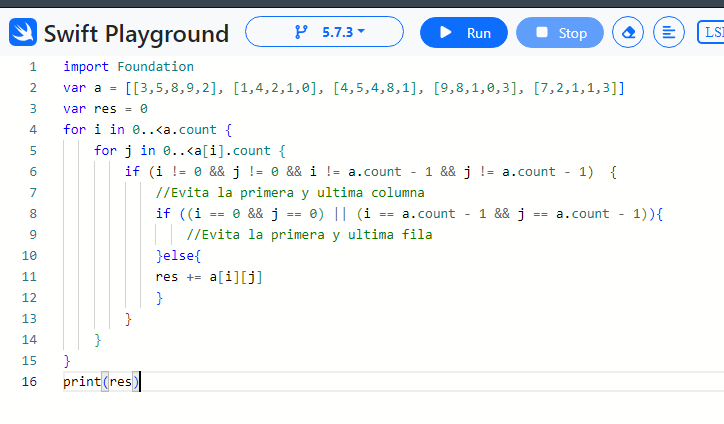
### Practica 2.3.31

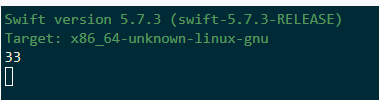
Dada una matriz cuadrada suma los valores que no se encuentran en el periférico de la figura.

Para esta práctica se emplearán 2 ciclos for anidados y 2 condicionales los cuales especificarán los valores válidos y aceptables.



El primer ciclo recorrerá las columnas, el segundo recorrerá los valores en las filas, y los condicionales definirán que no se encuentre un valor en la columna 0 o final, ni tampoco en la fila 0 o final, motivo por el que se tomaran únicamente los valores que cumplan con estos requisitos debido a que se encuentran relacionados con un AND.

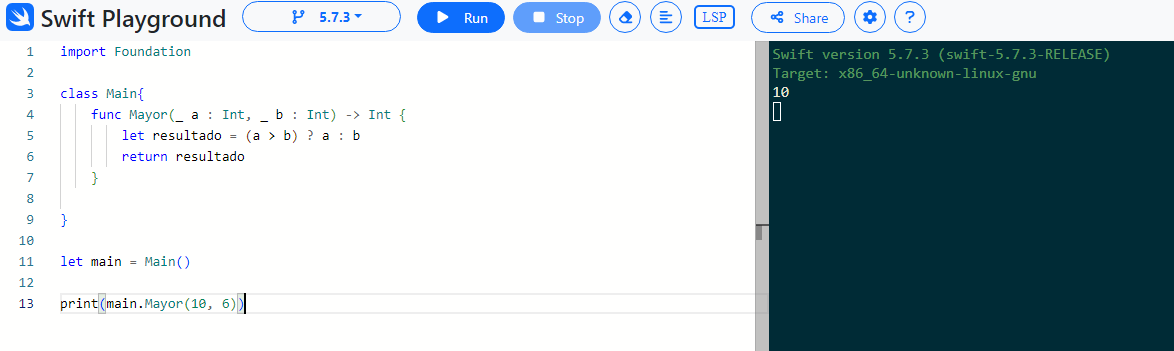




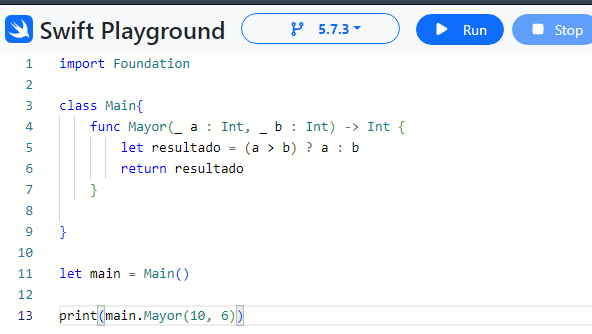
### Practica 2.3.32

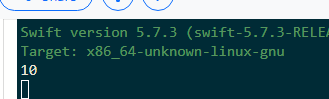
Escriba un método que reciba 2 valores y retorne la suma.

Para este código se optó por emplear una clase para mantener dentro el método el cual realiza la lógica mediante una condición ternaria.



Dentro de la función se reciben 2 parámetros a los que decidimos ignorar la etiqueta y este método retornara un entero, únicamente asigna a una constante el valor asignado (retornado) de una condición ternaria la cual se encarga de identificar el valor mayor de los dos parametros.

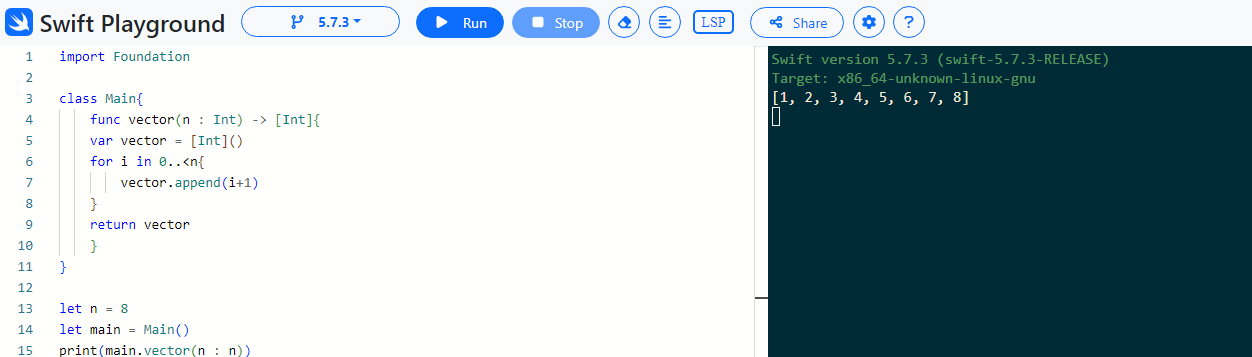


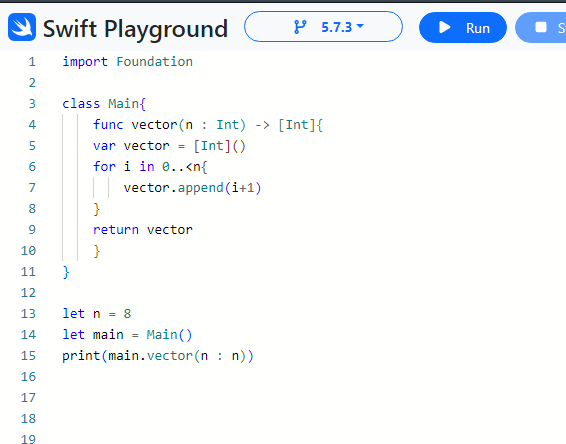


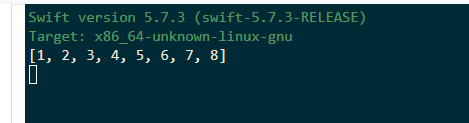
### Practica 2.3.33

Define un método que reciba un valor entero y devuelva un arreglo con los valores de 1 a N.

El método se prepara para recibir un entero en este caso no se ignora la etiqueta posteriormente se define que retornara un arreglo de enteros se procede a definir un vector y generar un for que recorra desde 1 hasta N número y por cada iteración esta se agregara al arreglo.



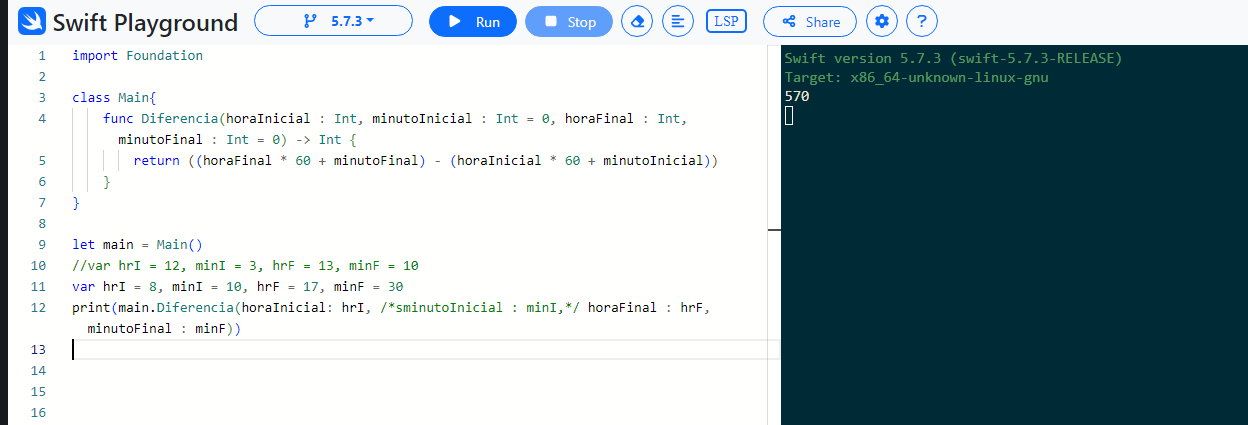




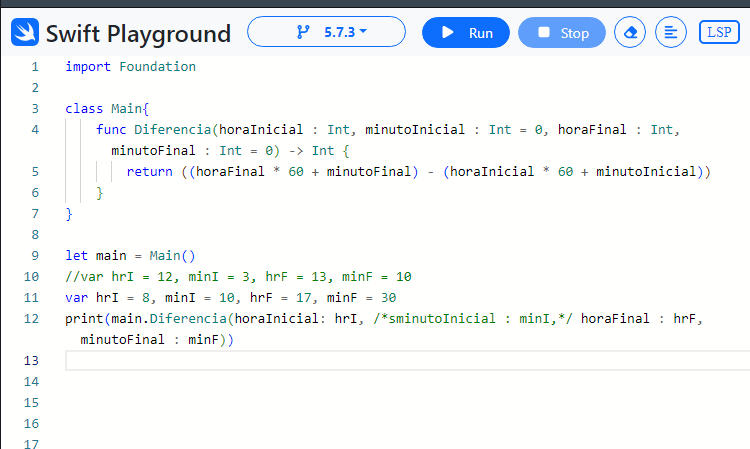
### Practica 2.3.34

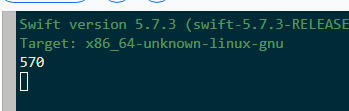
Programar una función que calcule la diferencia entre dos horas en minutos.

Para este programa solo se empleará la función preparada para recibir los parámetros en enteros pero con la condición de que si se omiten los minutos estos tendrán un valor de 0



Aquí la función si requiere de las etiquetas debido a que por medio de estas se podrá omitir el valor de los minutos sin necesidad de alterar la funcionalidad del método, únicamente se retorna el resultado de la operación y se imprime.





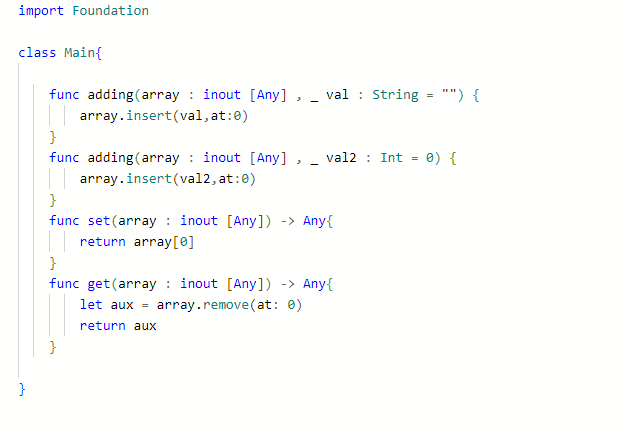
### Practica 2.3.35

Se requiere formular las funciones de una pila.

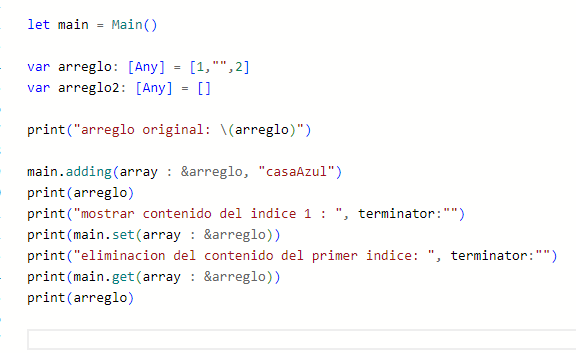
Para este método se emplearan funciones especificas de los arreglos.



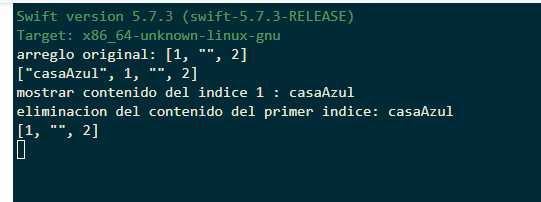
Dividiendo el código en dos segmentos



Los primeros dos métodos realizan la misma acción agregar un elemento al inicio de la lista (arreglo) pero con un enfoque de parámetros diferentes (overwrite) el tercer método muestra lo que se encuentra al principio de la lista, y el tercer método elimina el primer valor de la lista.



Su aplicación es mediante una constante de la clase main y con los parámetros específicos en este caso se ignoran las etiquetas para permitir el polimorfismo.

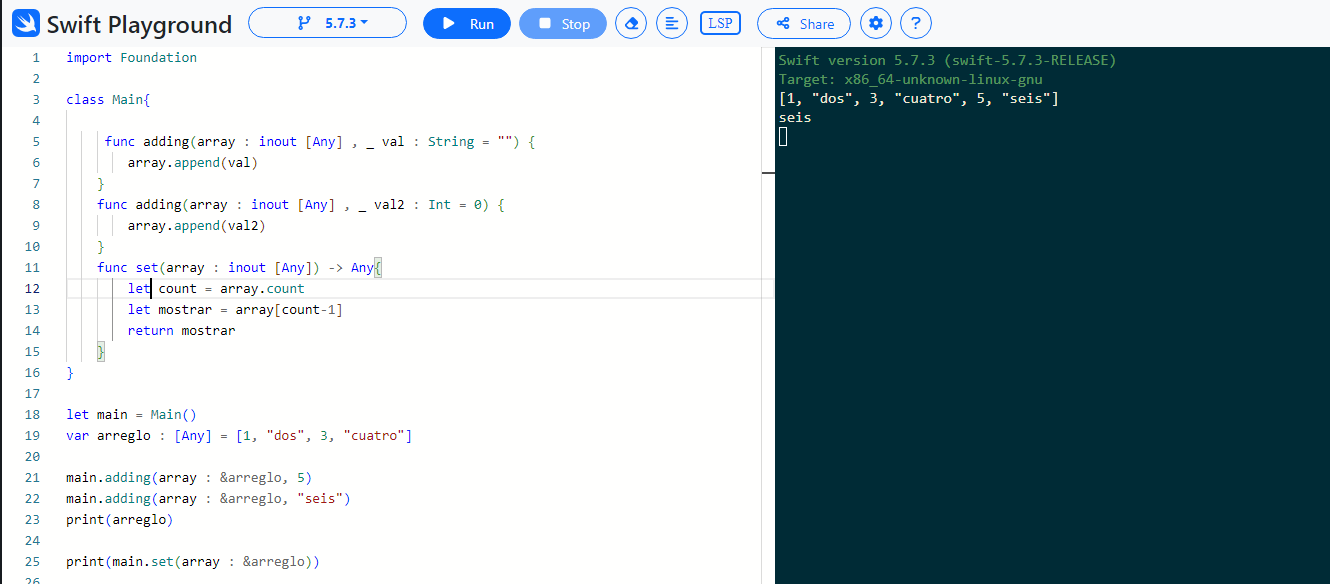


Se deberá de tener en cuenta el funcionamiento de una pila con sus funcionamientos.

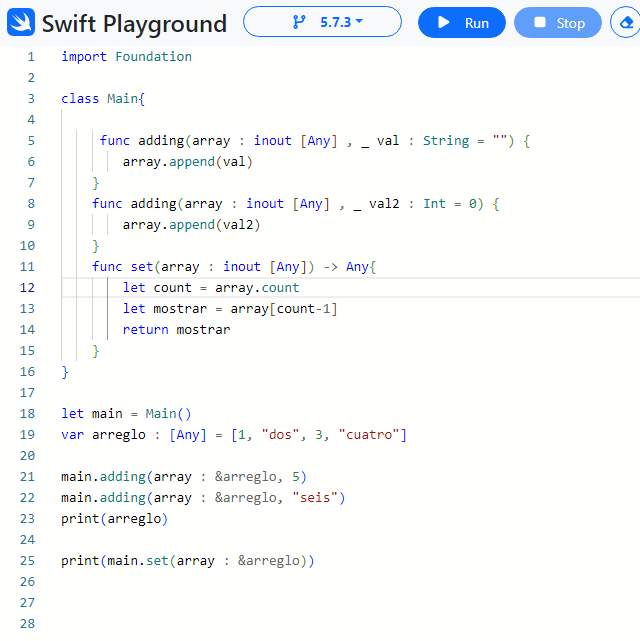
### Practica 2.3.36

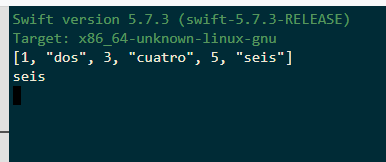
Generar funciones que apliquen al funcionamiento de una cola.

Para esta práctica se emplearán funciones de los arreglos para efectuar los cambios en estos mismos bajo los parámetros de una cola.



En este caso se generaron 3 funciones las 2 primeras efectúan lo mismo, pero con parámetros diferentes para aplicar el polimorfismo por lo que se evita el uso de las etiquetas únicamente agregamos el valor al final del arreglo, para el método set en este caso solo se recibe el arreglo se genera el número de elementos en el arreglo, se vacía el último elemento en una constante y se retorna (se encuentra en Any) debido a que puede regresar cualquier tipo de dato.

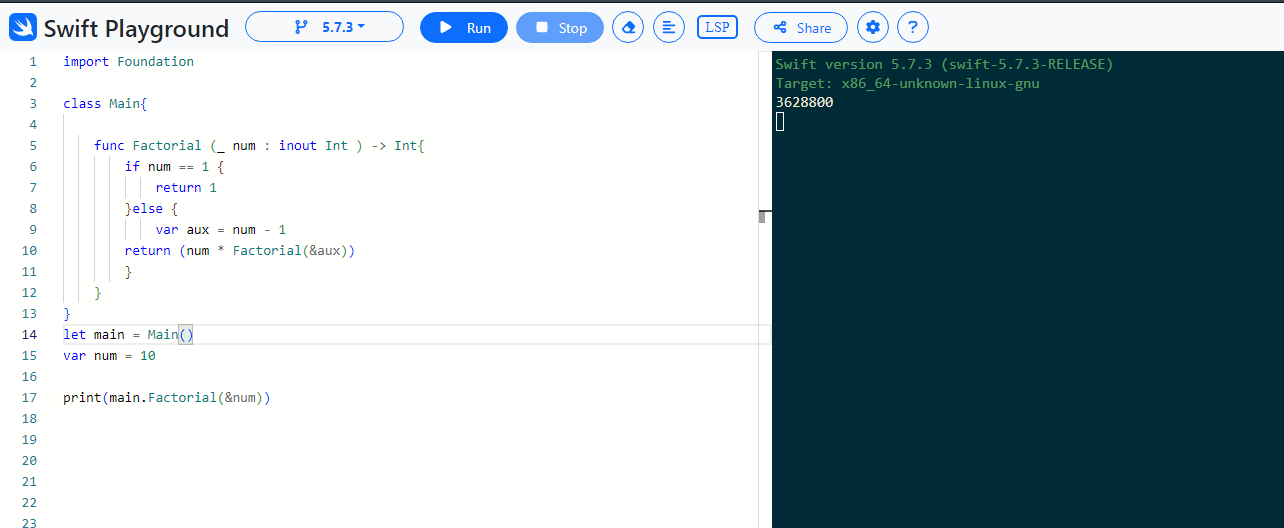




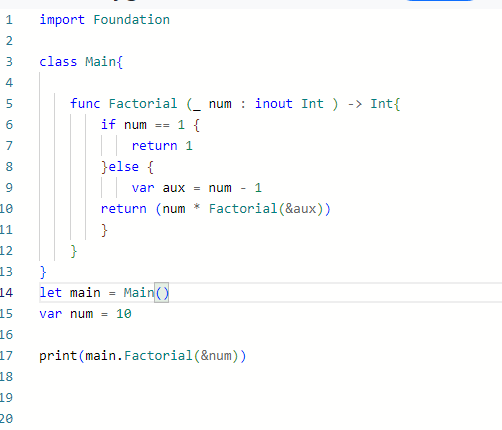
### Practica 2.3.37

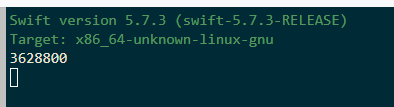
Calcula la factorial de un numero usando recursividad.

Para esta practica solamente se empleo un método recursivo y un if-else para saber cuando detener la recursividad.



Lo principal es que el método retorna un entero y recibe un entero pero a este le omitimos la etiqueta, la recursividad se genera al llamar al mismo meto desde si miso generando una llamada que contiene alguna característica diferente en este caso se manda la variable por referencia y se efectúan cambios para disminuir el numero original y factorizar el mismo.

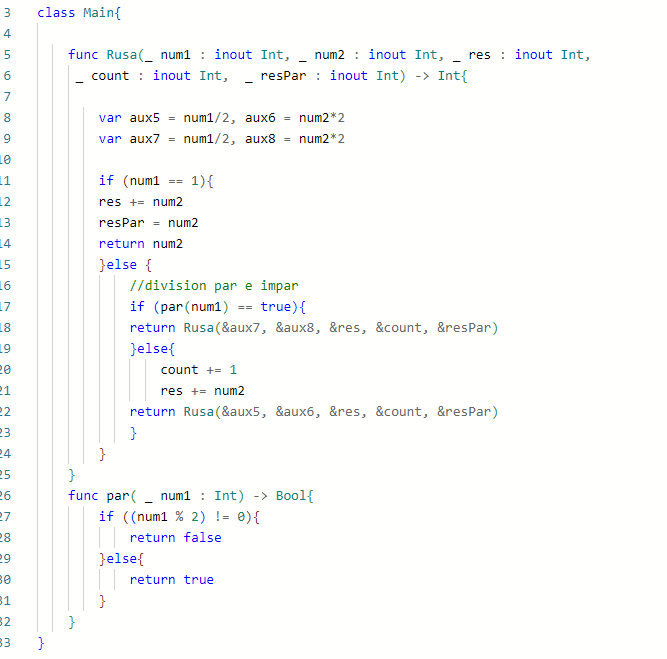




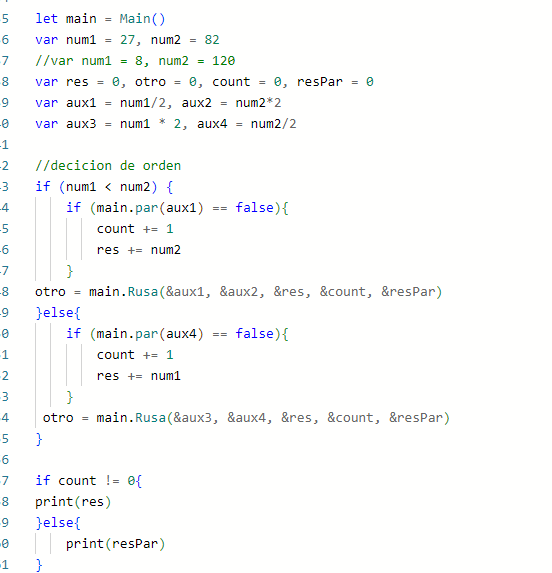
### Practica 2.3.38

Calcula la multiplicación de dos números por el método de la multiplicación rusa.

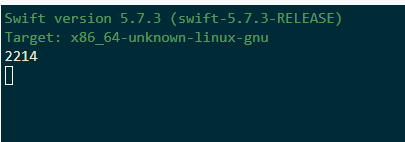
Para este método se pasaran los parámetros por referencia para poder realizar modificaciones directas con dicho acceso se puede generar una recursividad funcional.



Dividiendo el código en dos secciones podemos demostrar que esta se compone de dos funciones, en la primera se obtiene el calculo de las divisiones y multiplicaciones y en el segundo se obtienen los valores de si un numero es par o impar debido a las reglas del método ruso.



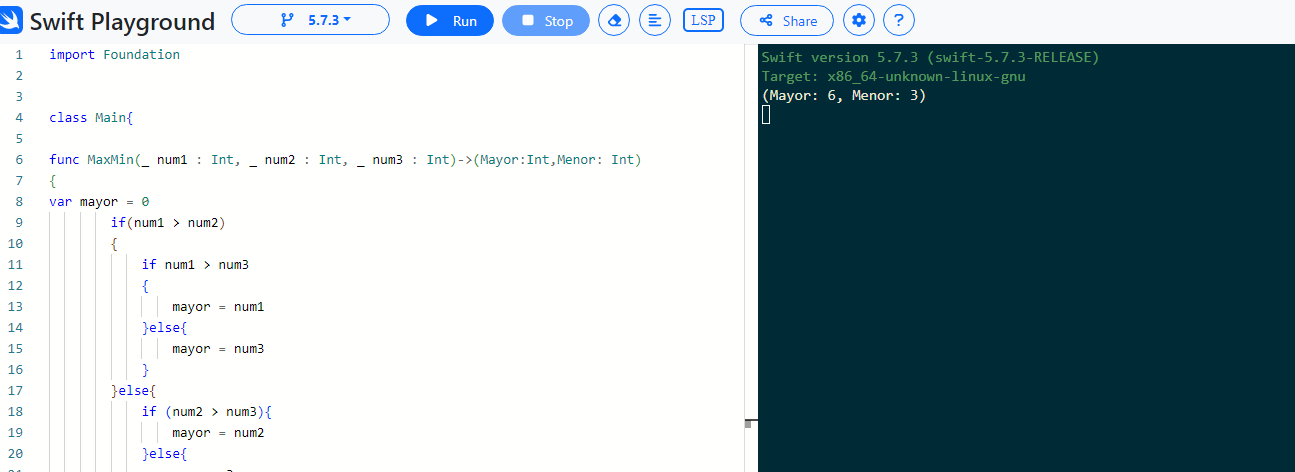
En el segundo bloque se define el orden debido a que el número mayor se multiplica y el menor se divide aquí también se envía al método de par o impar. Finalmente, si es impar algún cálculo de la división se suman los resultados de divisiones impares, en caso contrario solo se toma el ultimo valor de la multiplicación.



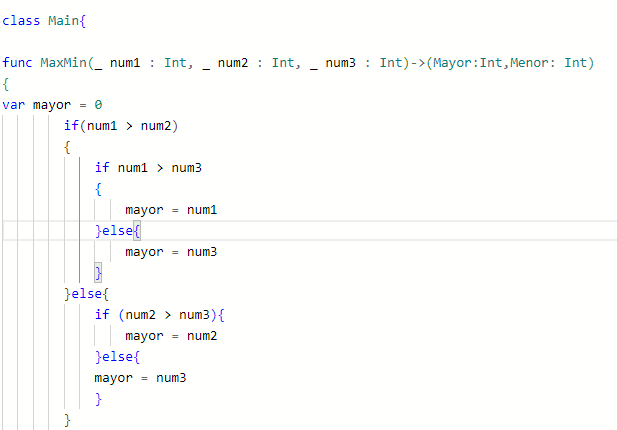
### Practica 2.3.39

Escribe una función que devuelva el numero mayor y menor de 3 numero.

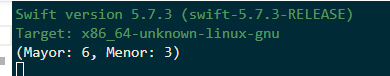
Para esta practica se emplearan if anidados.



Primeramente se recibirán 3 parámetros con las etiquetas omitidas, y se retornaran 2 valores enteros dentro del mismo se definirán 2 variables una mayo y otra menor en la que entraran en condicionales para conocer cual valor es el mayor y el menor, serán almacenadas de forma correspondiente, se retornaran los valore en un solo return.



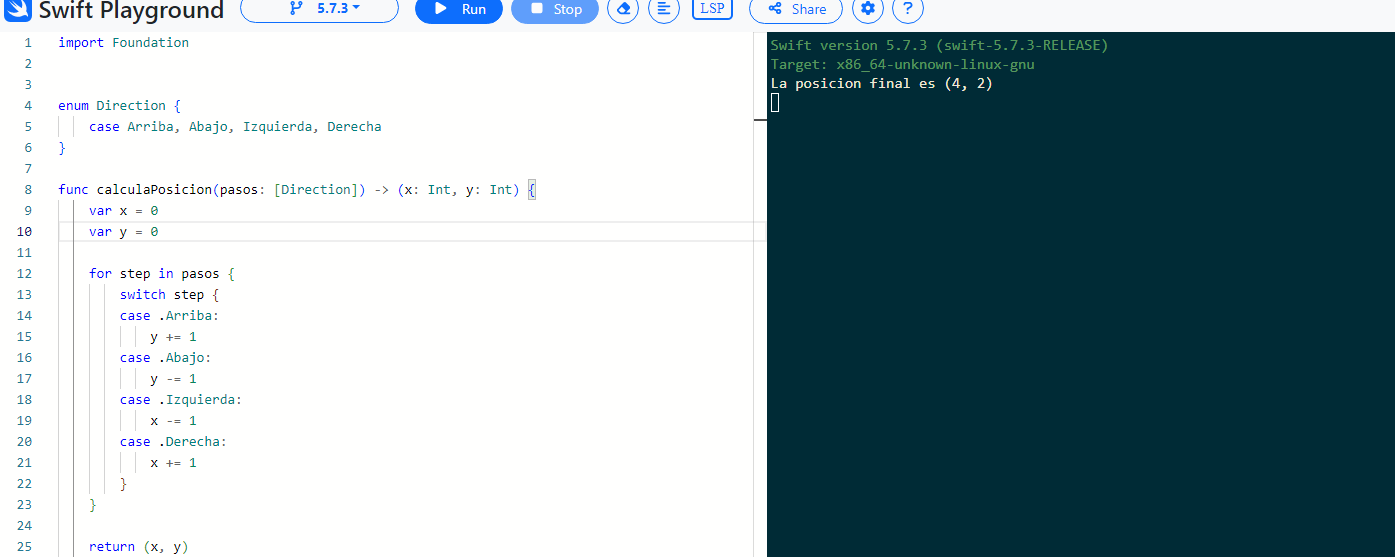




### Practica 2.3.40

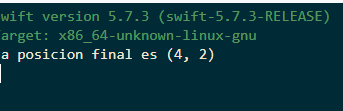
Generando las condiciones de desplazamiento en un plano cartesiano y haciendo uso de las enum calcula la posición final en base a los pasos brindados.

Para esta práctica se definirá la enum de Direcction donde se almacenarán los pasos, una función que calculara los pasos según sean recibidos por medio de un arreglo que descifra los casos del enum



La función se define con un parámetro de arreglo y una salida de dos enteros, dentro del mismo se definen las variables x, y las cuales se modificarán según sea el caso en cada opción brindada por el arreglo de los pasos.

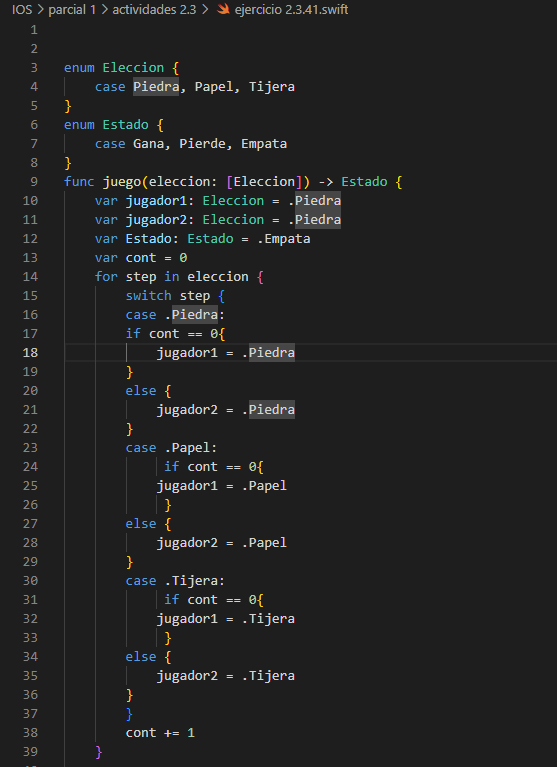




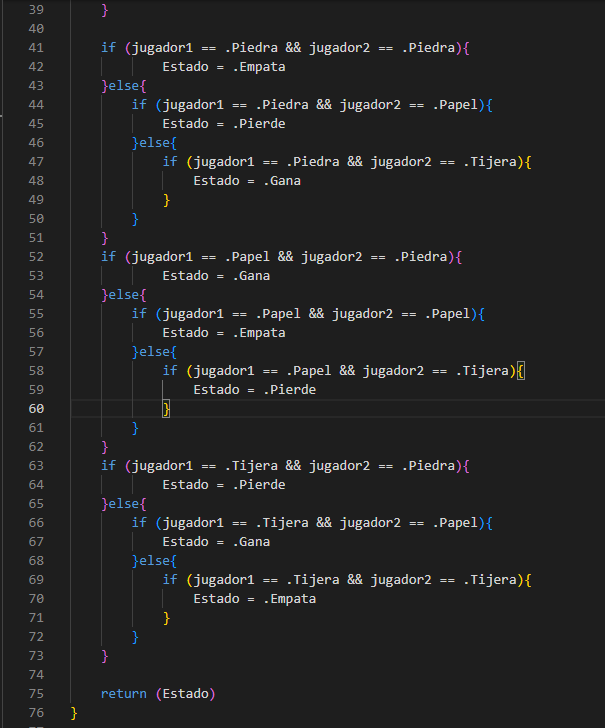
### Practica 2.3.41

Sin utilizar valoraciones numericas y definiendo dos enum genera un juego de piedra papel o tijeras con estados de gana pierde empate con respecto al primer jugador.

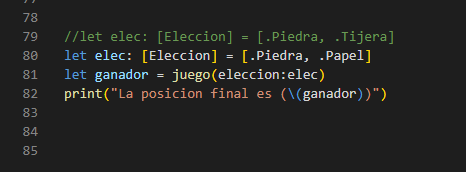
Se tendrán que generar dos enum pero en la funcion solo se obtendrán resultados en base a los if un case que se encuentran dentro de un for y un switch

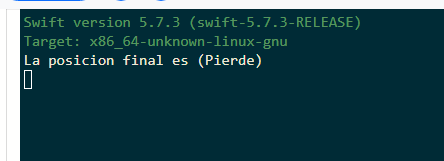


En el primer punto solo se brinda un sentido a las ambas partes de los jugadores conforme a sus elecciones anteriores.



En el segundo fragmento de la misma función se puede persivir que mediante los if se generan casos donde se puede modificar el estado y finalmente una vez establecido el estado este se retorna .





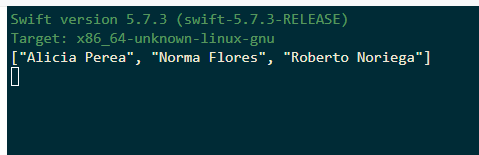
### Practica 2.3.42

Brindando un diccionario genera los nombres completos por nombre y apellido.

Para esta practica se emplearan el diccionario un ciclo for y una variable a la que se le concatenara el orden correcto de los valores



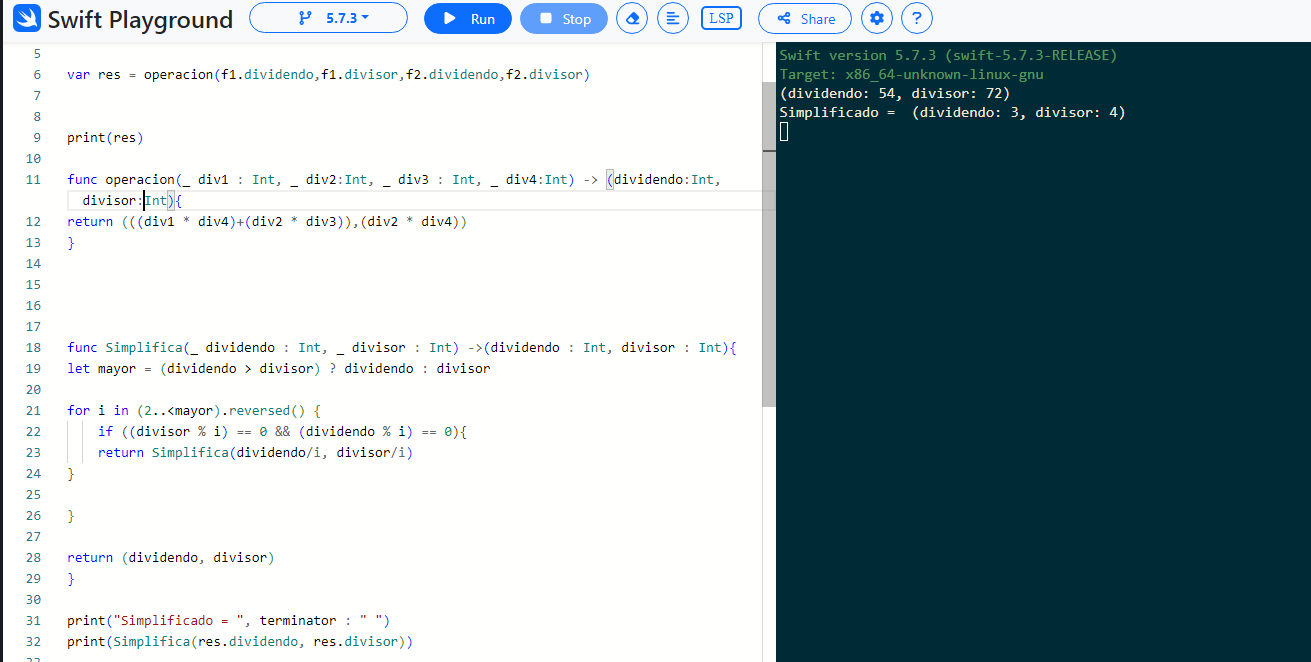
Principalmente se induce un ciclo for para empezar a recorrer el arreglo del diccionario posteriormente se ordenan y concatenan a una variable temporal del ciclo for, y se agregan al final del arreglo.



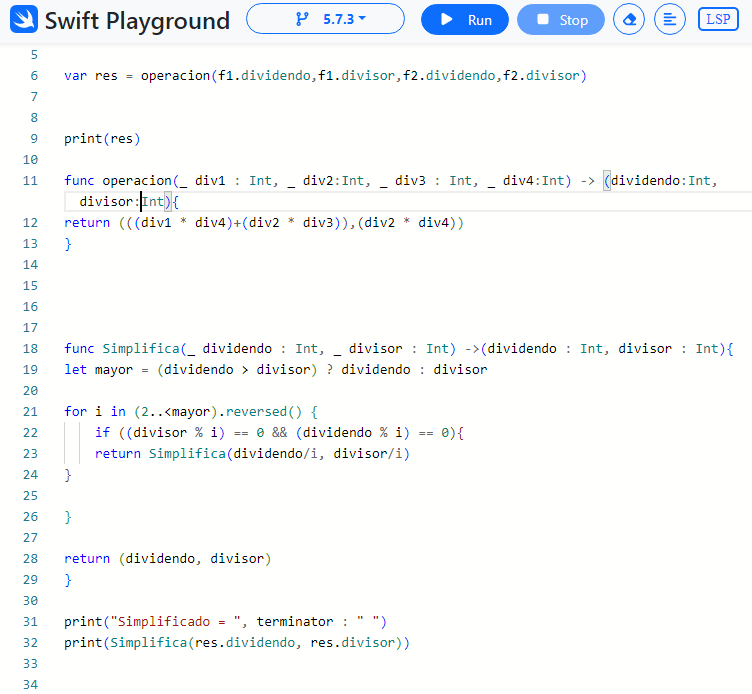
### Practica 2.3.43

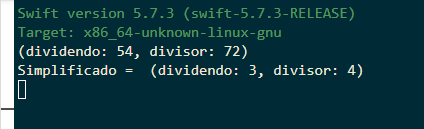
Teniendo dos tuplas cada una representa una fracción escribe una suma de fracción.

Principalmente se toman las dos tuplas se envían a la función de operación donde se realizará la suma de estas mismas y finalmente se pasarán por otra función para simplificar de ser posible.



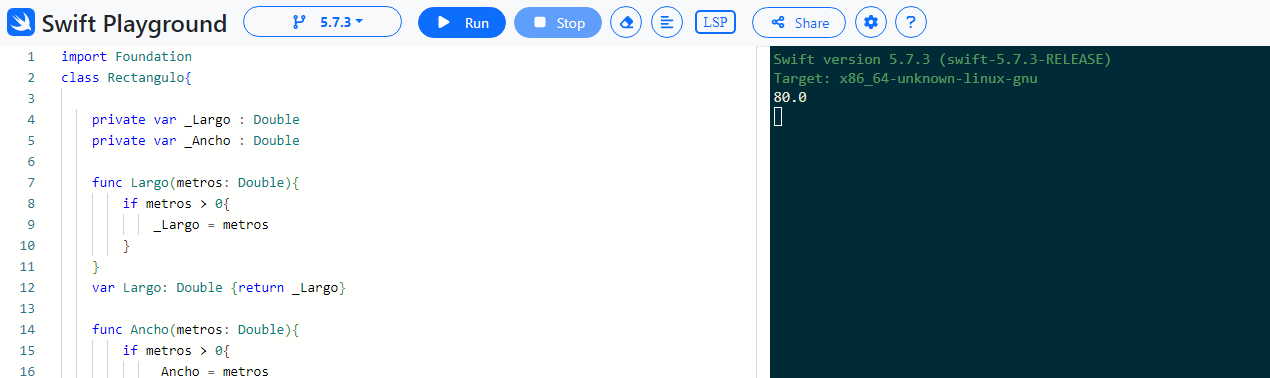
Generalmente se están recibiendo tuplas de manera directa cuando recibimos o retornamos múltiples parámetros en una función por eso mismo el enviar y recibir tuplas es algo que no requiere un cambio en Swift. Con la primera función únicamente se resuelve la operación, mientras que la segunda se aplica únicamente si el resultado contiene alguna simplificación mediante un método de división modular entre divisor y dividendo.



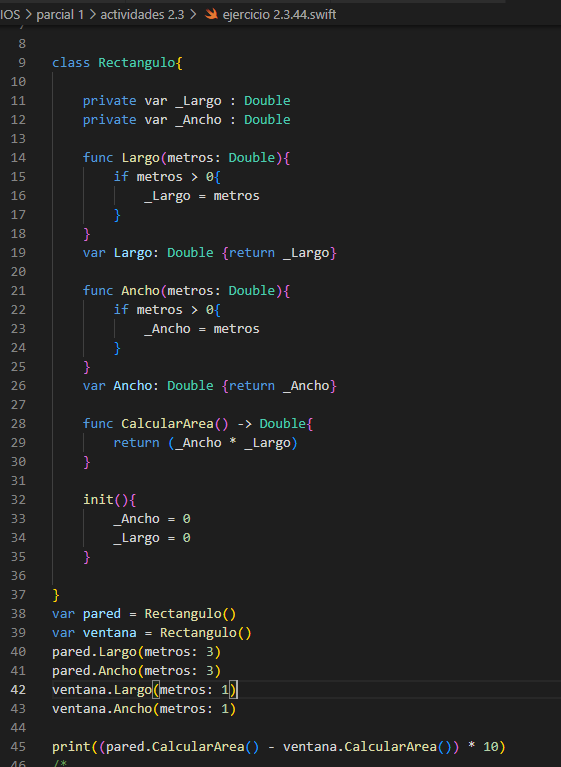


### Practica 2.3.44

Se requiere crear una clase llamada rectángulo que permita la simulación de una pared y una ventana y permita simular el tiempo estimado que tardaría en pintarse.



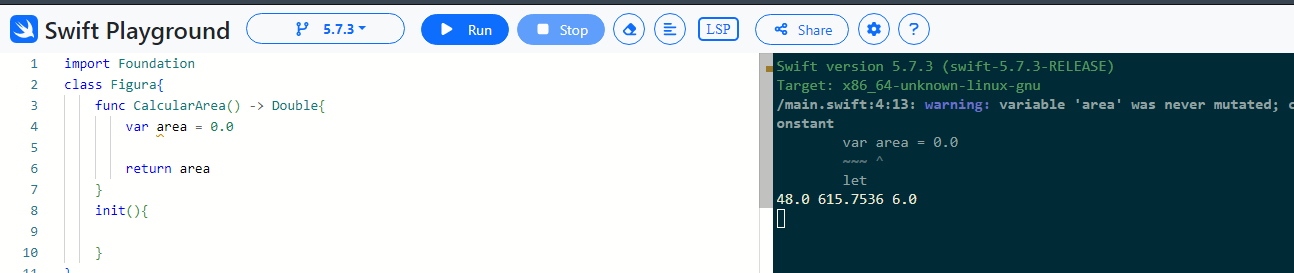
Para la realización de esta practica se hace uso de una clase denominada rectángulo que contiene las propiedades de ancho y largo y permite calcular el área que ocuparía dicho rectángulo.



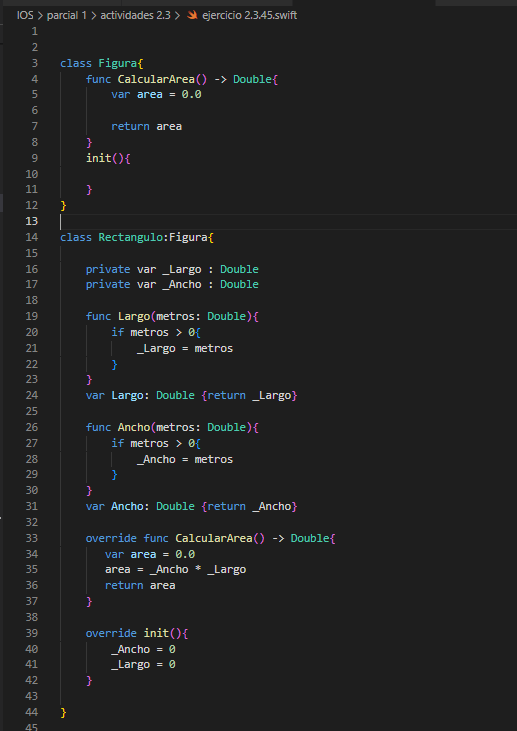
El método de empleo únicamente es definir dos objetos introducir sus respectivas medidas y posteriormente realizar el cálculo, debido a que la ventana se encuentra dentro de la pared es necesario restar el área de la ventana y en este caso se multiplica \* 10 debido a que es el tiempo que se tardarían en pintar un m^2.

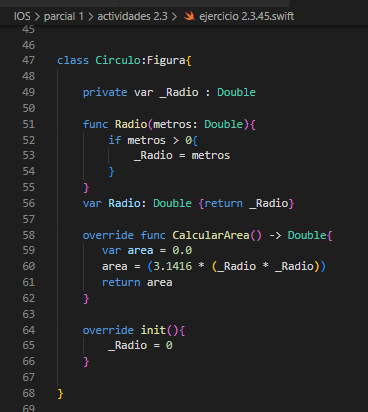
### Practica 2.3.45

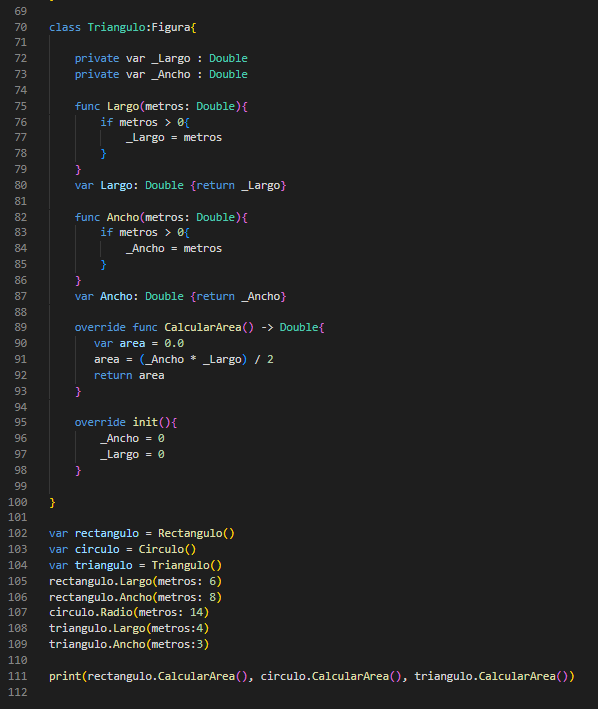
Para esta práctica se requiere de generar una clase de la cual heredaran 3 clases más y permitirá la adaptación de la información y el método a la nueva clase heredada.



En este caso tomamos una clase como base un meto el cual se adaptará a la clase para permitir un mejor funcionamiento, de esta clase padre se heredan tres clases las cuales sobrescribirán el método para adaptarlo a sus necesidades.



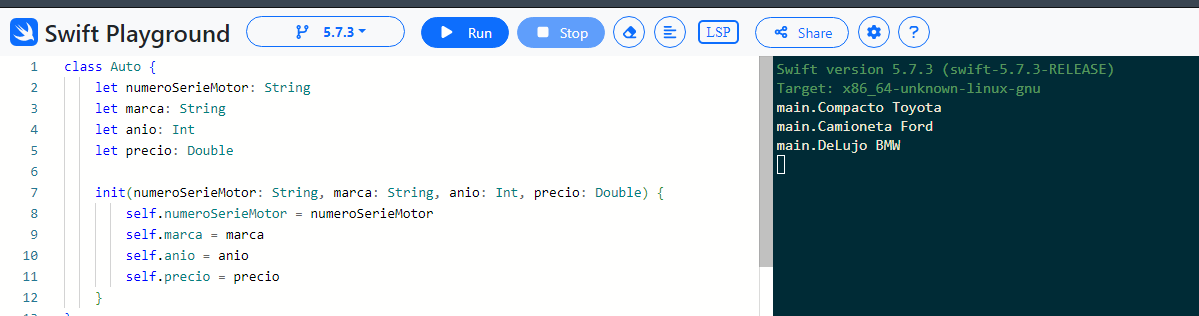


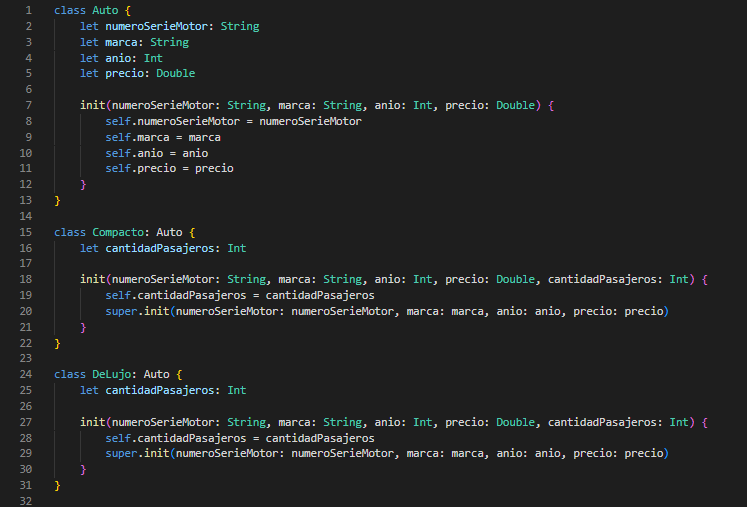


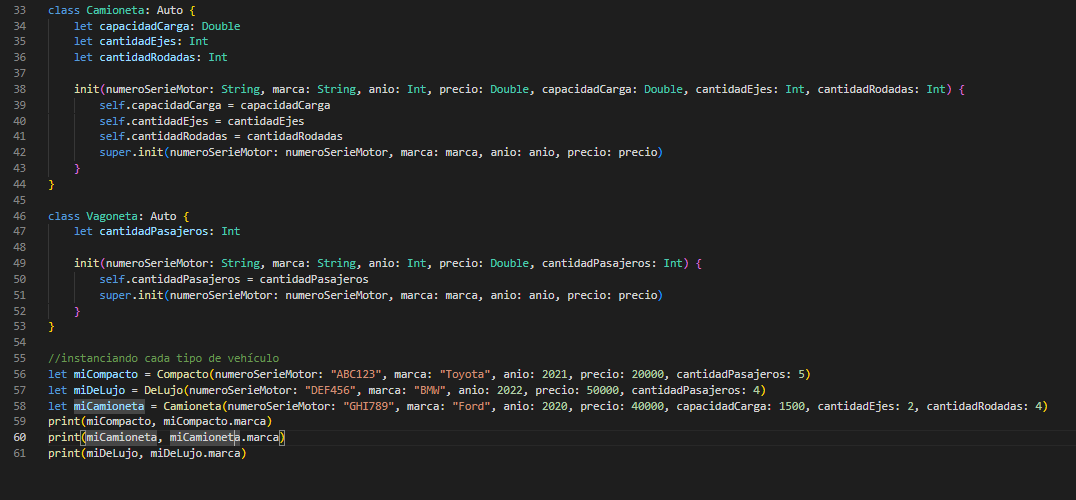
### Practica 2.3.46

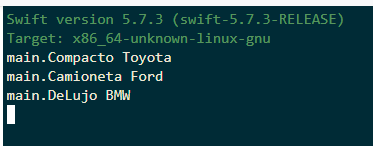
Se requiere generar una clase que apoye en la categorización de vehículos.

Principalmente se genera una clase padre llama “Auto” de la cual derivaran los 4 tipos diferentes de autos que vamos a categorizar, esta clase contendrá las propiedades bases que comparten todos los vehículos en este caso como no se pide el uso de la actualización de información únicamente se adaptó para recibir la información pública y de forma inmutable.







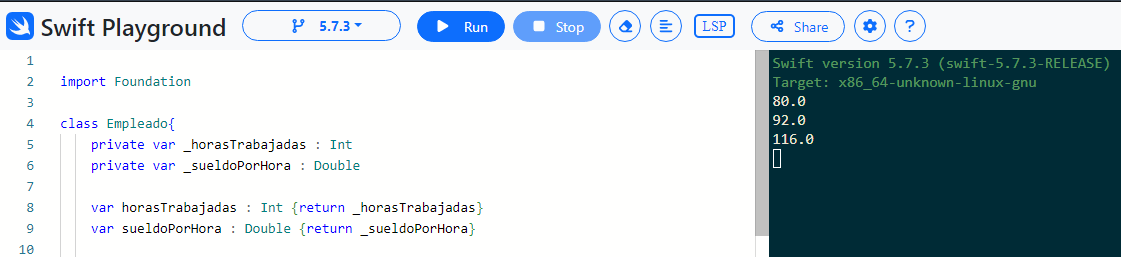


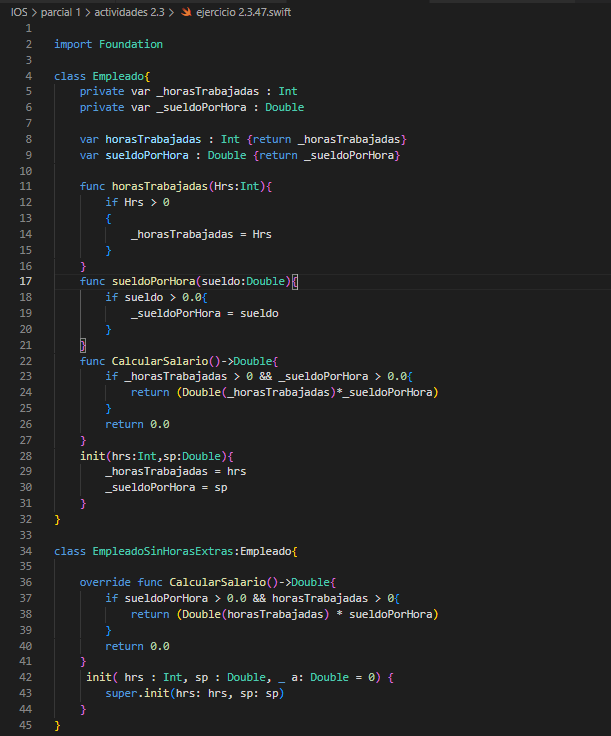
De esta manera se permite contemplar que cada parámetro de información requerida para este caso.

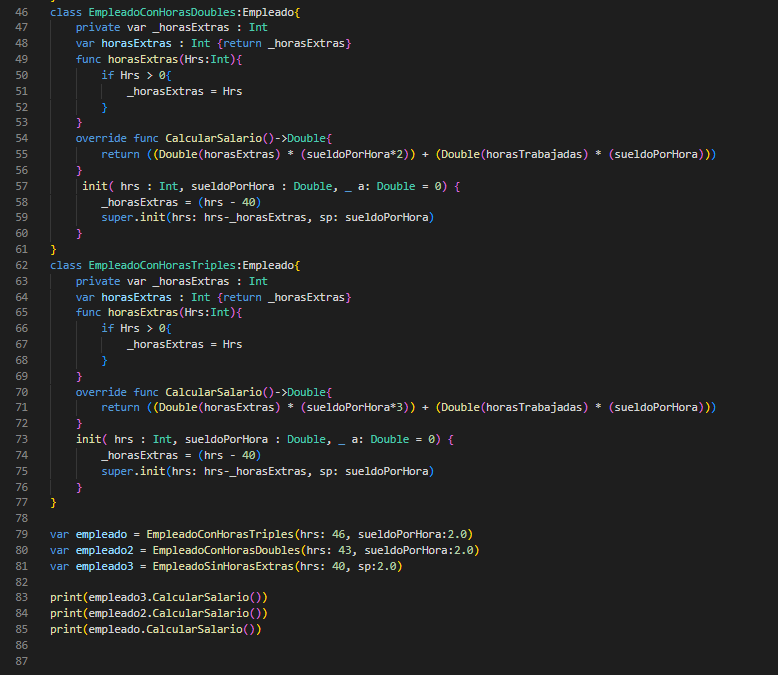
### Practica 2.3.47

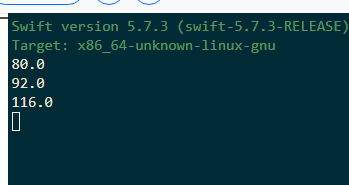
En esta práctica se requiere categorizar a los empleados en 3 tipos (los que no trabajan horas extras, los que se les paga doble las horas extras y a los que se les pagan triple las horas extra).

Dicha clase padre contendrá la idea básica y general de un empleado promedio son horas, y los otros dos contendrán una propiedad en la que se almacenaran las horas extras realizadas, con esto la clase contendrá un método para retornar el total a pagar por sus servicios.



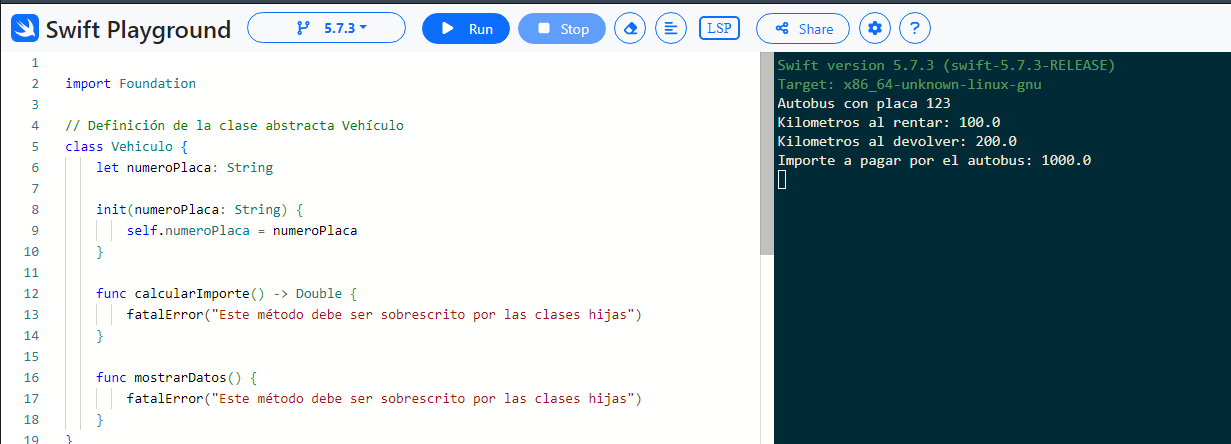




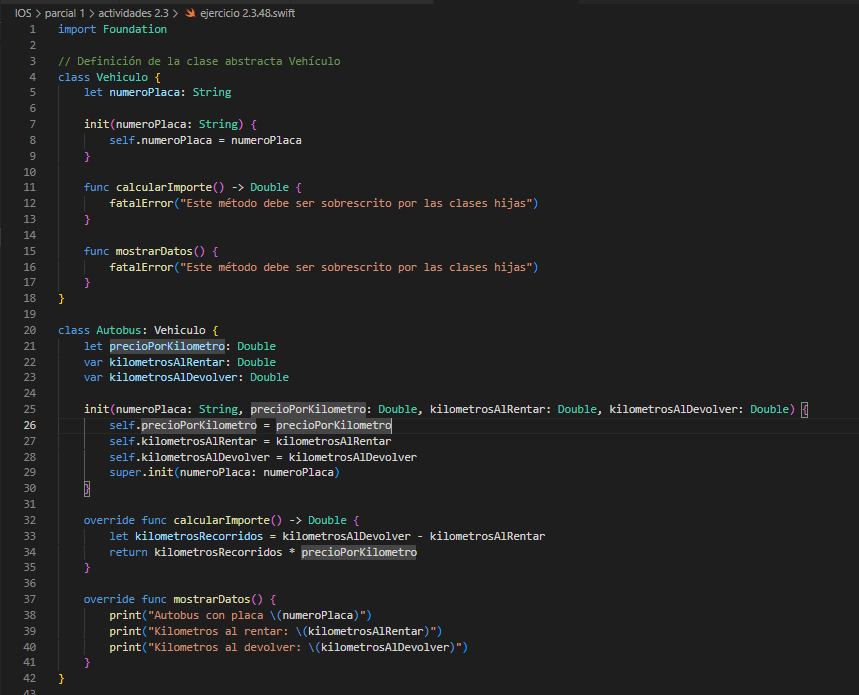


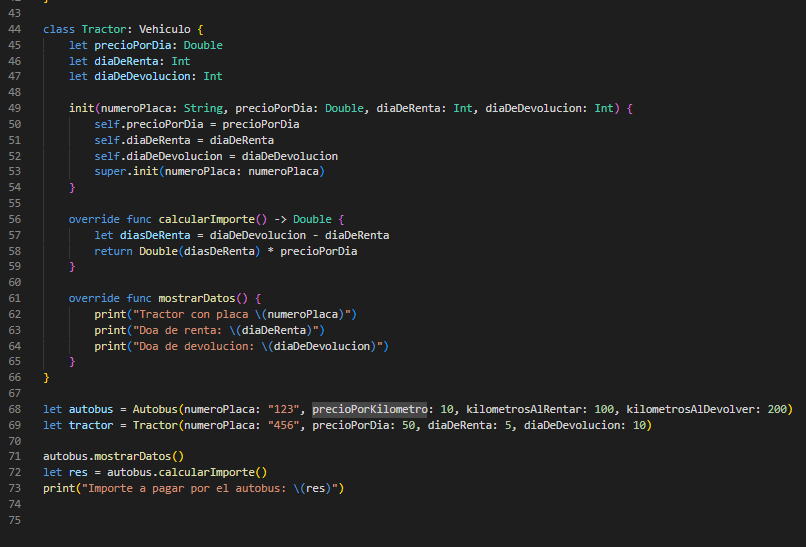
### Practica 2.3.48

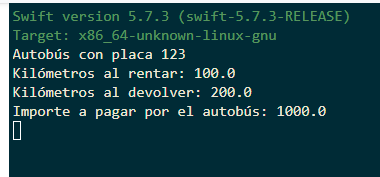
Para esta práctica se requiere categorizar dos tipos de vehículos, pero en este caso se tendrá que definir una clase abstracta para su implementación.



En la clase se puede observar que se manejan dos errores que se disparan en caso de que el método sea invocado a partir de la clase abstracta.







# Referencias.

* <https://www.ecured.cu/Mac_OS>
* <https://www.applesfera.com/os-x/macos-sistema-operativo-mas-longevo-apple-encima-mac-os-classic>
* <https://www.cleverfiles.com/howto/es/what-is-mac-os-x.html>
* <https://www.arimetrics.com/glosario-digital/xcode>
* <http://dspace.ups.edu.ec/handle/123456789/4346>
* <https://developer.apple.com/es/support/xcode/>
* <https://www.apple.com/mx/app-store/>
* <https://developer.mozilla.org/es/docs/Glossary/MVC>
* <https://www.swiftbeta.com/arquitectura-model-view-controller-en-swift/>
* <https://developer.apple.com/es/support/xcode/>
* <https://developer.apple.com/es/support/enrollment/>
* <https://www.apple.com/mx/swift/>
* <https://www.ecured.cu/Swift>
* <https://www.apple.com/mx/swift/#:~:text=Swift%20es%20gratis%20y%20de,%2C%20watchOS%2C%20tvOS%20y%20Linux>.
* <https://code.tutsplus.com/es/tutorials/swift-desde-cero-una-introduccion-a-funciones--cms-22879>
* <https://developer.apple.com/library/archive/documentation/MacOSX/Conceptual/OSX_Technology_Overview/CocoaApplicationLayer/CocoaApplicationLayer.html>
* <https://developer.apple.com/library/archive/documentation/ToolsLanguages/Conceptual/DevPortalGuide/ManagingYourApplicationsandInAppPurchases/ManagingYourApplicationsandInAppPurchases.html>