

PROYECTO DE USAID

PUENTES PARA EL EMPLEO

Universidad de Oriente

Quelepa

Tecnologías de la Información

Uso de Sintaxis y Estructuras de Control en C# Básico para Xamarin

MANUAL DEL PARTICIPANTE

Nivel 1

2018

Quelepa, El Salvador, Centroamérica



La reproducción de este material ha sido posible gracias al generoso apoyo del pueblo de los Estados Unidos a través de la Agencia de los Estados Unidos para el Desarrollo Internacional (USAID). El contenido es responsabilidad exclusiva del [Nombre del Centro de Formación]; no refleja necesariamente la opinión de USAID o del Gobierno de los Estados Unidos

ÍNDICE

[1.1 Campo de Especialidad 3](#_Toc520280443)

[1.2 Código del Módulo 3](#_Toc520280444)

[1.3 Duración del módulo 3](#_Toc520280445)

[2. DESCRIPCIÓN GENERAL DEL MÓDULO 3](#_Toc520280446)

[2.1 Descripción 3](#_Toc520280447)

[2.2 Competencia del módulo 3](#_Toc520280448)

[Desarrollar operaciones básicas usando sintaxis C# de manera funcional y ordenada. 3](#_Toc520280449)

[2.3 Criterios de evaluación 3](#_Toc520280450)

[2.4 Criterios de promoción 3](#_Toc520280451)

[3.1 Módulo 9: Uso de Sintaxis y Estructura de Control en C# Básico. 3](#_Toc520280452)

[3. DESARROLLO DE UNIDADES DE APRENDIZAJE 3](file:///H:\XAMARIN_USAID\Manuales%20Nivel%20I\Uso%20de%20sintaxis%20y%20estructuras%20de%20contro%20en%20C%23%20Basico%20Xamarin\N1-09%20-%20Uso%20de%20Sintaxis%20y%20Estructuras%20de%20Control%20en%20C%23%20Basico%20Xamarin.docx#_Toc520280453)

[3.1.1 Identificar qué es un programa en informática y manejar correctamente el entorno de desarrollo (IDE) de Visual Studio 2017. 3](#_Toc520280454)

[Actividad Propuesta1 7](#_Toc520280455)

[3.1.2 Manipular las instrucciones básicas en C#. 8](#_Toc520280456)

[Actividad Propuesta1I 11](#_Toc520280457)

[3.1 Módulo 9: C# Básico 12](#_Toc520280458)

[3.1.3 Utilizar colecciones y bucle foreach para controlar elementos en C#. 12](#_Toc520280459)

[Actividad Propuesta 3 14](#_Toc520280460)

[3.1.4 Declaración de funciones que retornan VOID (vacío). 14](#_Toc520280461)

[Firmas de métodos 15](#_Toc520280462)

[Actividad Propuesta 4 17](#_Toc520280463)

[3.1.5 Comprender que es un framework y los elementos que lo componen en C# Xamarin. 17](#_Toc520280464)

[¿Qué es .NET Framework? 18](#_Toc520280465)

[Actividad Propuesta 5 22](#_Toc520280466)

[3. GLOSARIO 23](#_Toc520280467)

[4. PLAN DE EVALUACIONES 24](#_Toc520280468)

[5. JORNALIZACIÓN DEL MÓDULO 25](#_Toc520280469)

[6. FUENTES DE INFORMACIÓN Y MATERIALES DE APOYO 26](#_Toc520280470)

1. INFORMACIÓN DEL MÓDULO

## 1.1 Campo de Especialidad

* Desarrollador de Aplicaciones Web usando C# ASP.net con MVC
* Desarrollador de Aplicaciones Móviles con C# Xamarin

## 1.2 Código del Módulo

SEC

## 1.3 Duración del módulo

4 Semanas





# 2. DESCRIPCIÓN GENERAL DEL MÓDULO

## 2.1 Descripción

Este módulo está orientado para que el estudiante conozca los fundamentos básicos de programacion, depuración, ejecución y el entorno de desarrollo; para el desarrollo de código funcional utilizando el lenguaje de programación C# con lo cual se pretende sentar la base para generar código y la creación de aplicaciones funcionales que resuelvan problemas básicos.

## 2.2 Competencia del módulo

## Desarrollar operaciones básicas usando sintaxis C# de manera funcional y ordenada.

## 2.3 Criterios de evaluación

* Identifica qué es un programa en informática.
* Codifica de manera correcta las instrucciones básicas en C#.
* Codifica instrucciones utilizando el bucle foreach.
* Elabora códigos en C# que implementen funciones que retornan un valor vacío.
* Define qué es un Framework y cuáles son los elementos que lo conforman.

## 2.4 Criterios de promoción

Para dar por aprobado un módulo el participante deberá:

* Haber asistido al 100% de las clases del módulo.
* Desarrollar las actividades que se presentan en el manual.
* Obtener un promedio final mayor o igual a 7.0 en las actividades del módulo.

## 3.1 Módulo 9: Uso de Sintaxis y Estructura de Control en C# Básico.

# 3. DESARROLLO DE UNIDADES DE APRENDIZAJE

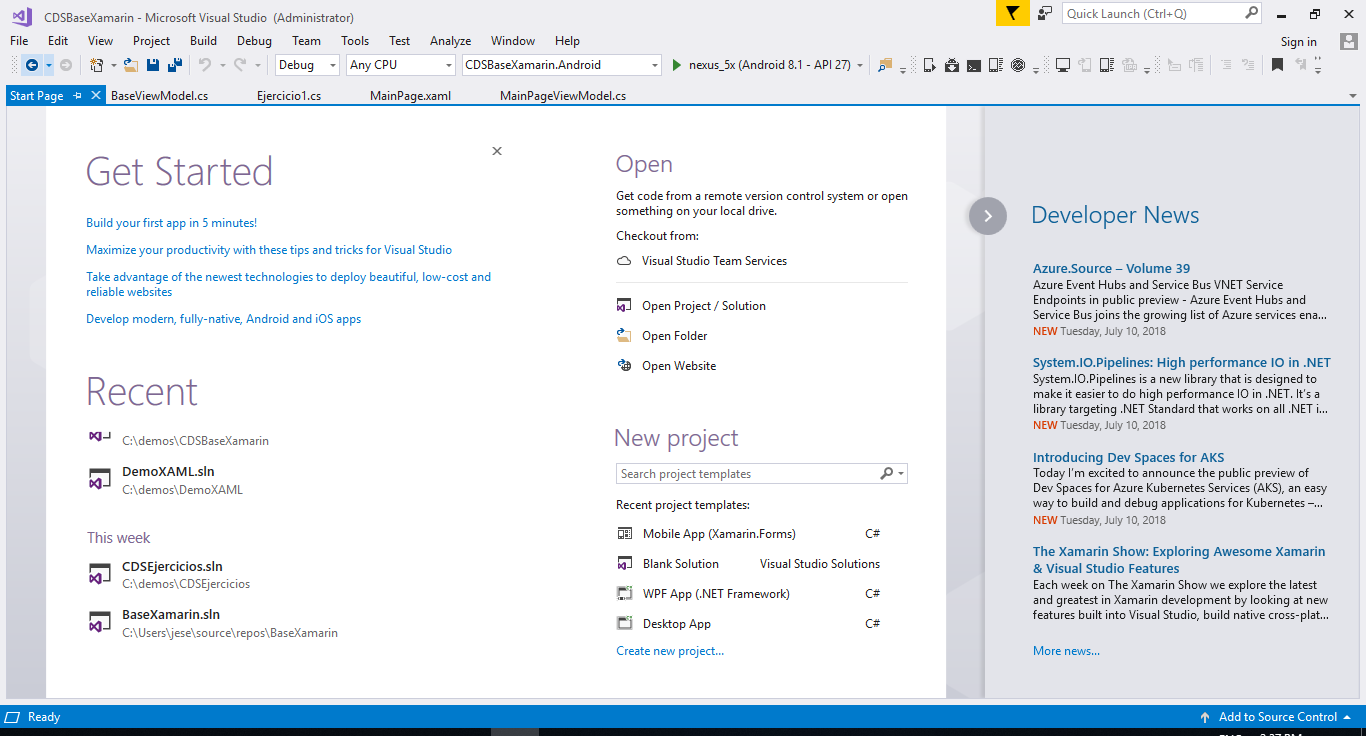
## 3.1.1 Identificar qué es un programa en informática y manejar correctamente el entorno de desarrollo (IDE) de Visual Studio 2017.

**Entorno de Desarrollo Integrado (IDE):**

Un entorno de desarrollo integrado (IDE) es un conjunto de herramientas de desarrollo expuestas a través de una interfaz de usuario común. Algunas de las herramientas se comparten con otros lenguajes de Visual Studio, y otras, como el compilador de C#, son exclusivas de Visual C#.

**Conceptos básicos del IDE**

La siguiente imagen muestra el IDE de Visual Studio 2017 con un proyecto abierto, la ventana Explorador de Soluciones para desplazarse por los archivos de proyecto y la ventana de Team Explorer para navegar por el código fuente y controlar los elementos de trabajo. A continuación, se explican con más detalle las características de la barra de título mencionadas.



**Lista de los Elementos Principales del Entorno de Desarrollo Integrado Visual Studio 2017:**

-Lista de errores (Ventana).

Muestra errores y advertencias de compilación.

- Proyectos y soluciones, Cuadro de diálogo Opciones

Le permite especificar opciones de proyecto y solución.

- Ventana Propiedades

Muestra las propiedades para los proyectos, archivos, controles y campos en las ventanas y los diseñadores del IDE.

- Cuadro de Herramientas

Permite agregar controles a una superficie de diseño.

- Origen de Datos

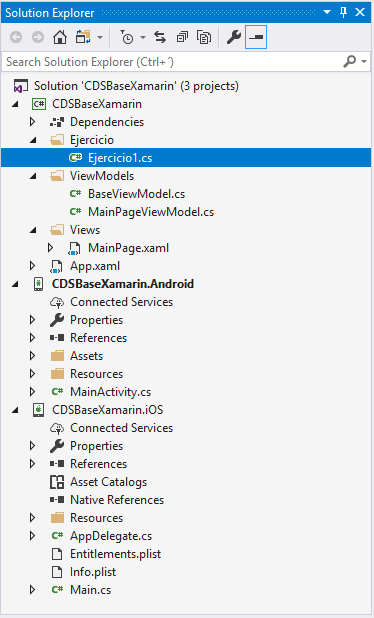
Permite configurar la accesibilidad a las bases de datos.

- Menú Principal

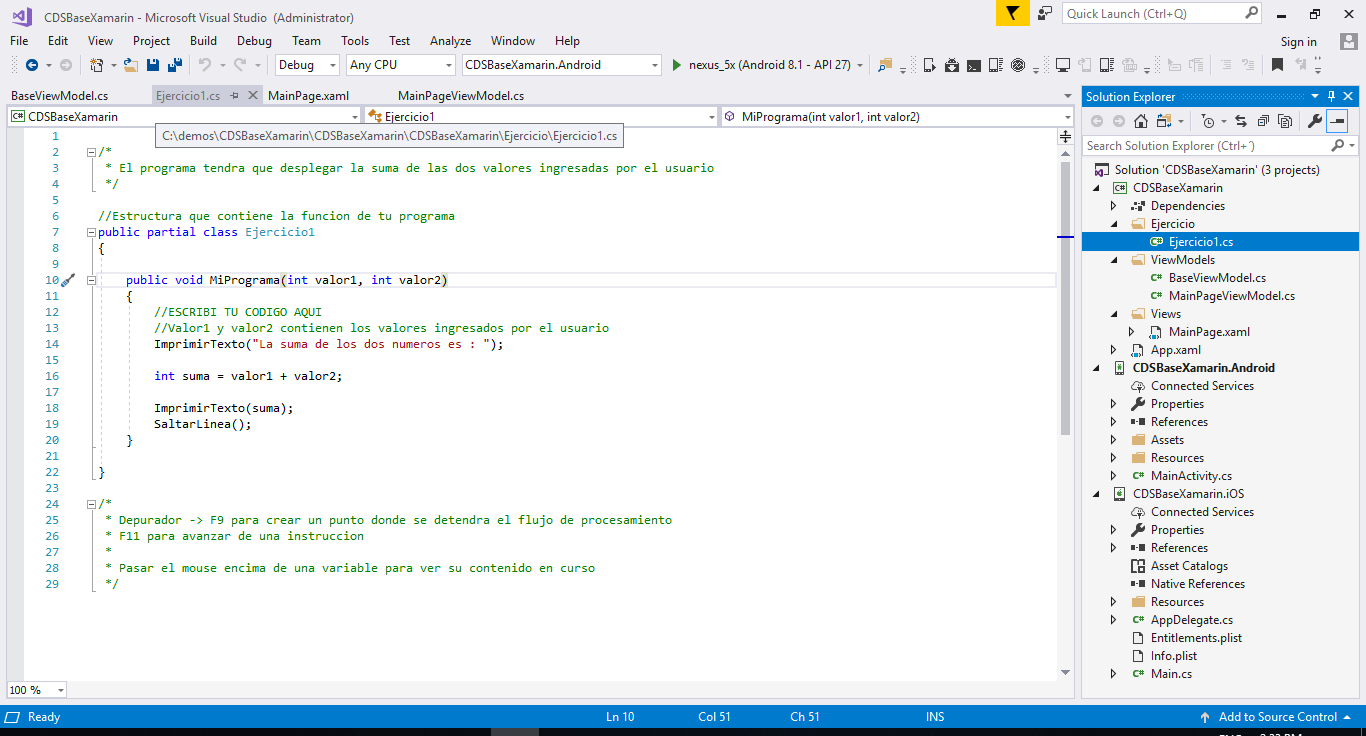
Listado de opciones que permiten desencadenar las distintas acciones permitidas por el IDE donde se pueden crear archivos, ejecutar elementos, entre otros.

**Explorador de Soluciones.**

Muestra la estructura de un proyecto utilizando Xamarin cross-plafform.

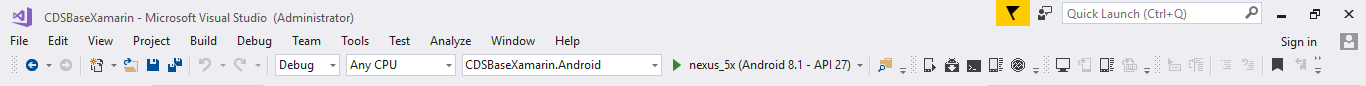


Area de Trabajo.



Menu Principal.

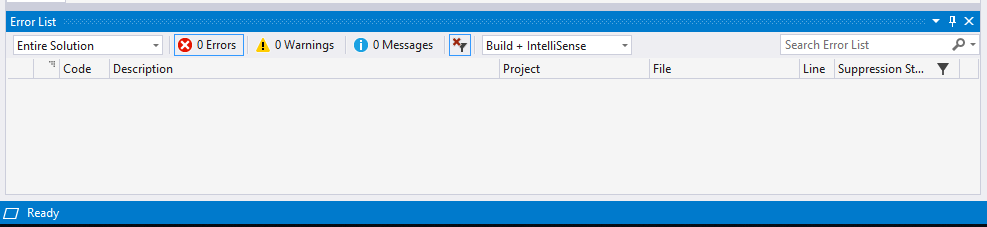
Permite observar el comportamiento del programa en tiempo de ejecución y detectar problemas.



Ventana de Propiedades.

**Ventana de errores.**

Muestra errores y advertencias de compilación.

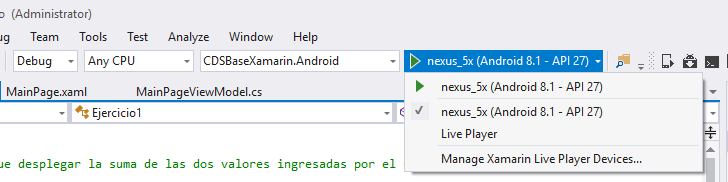


**Depurador.**

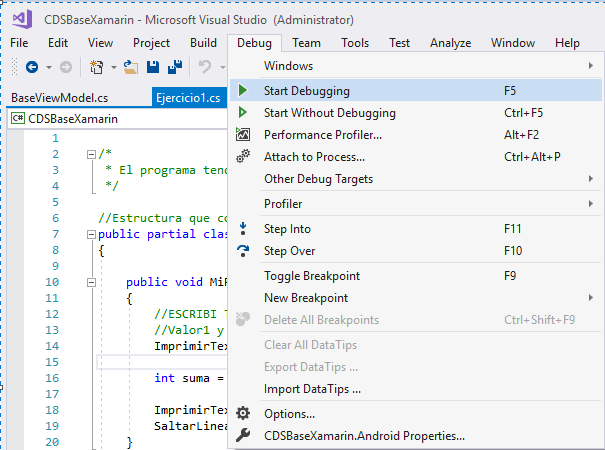
El depurador de Visual Studio permite observar el comportamiento del programa en tiempo de ejecución y detectar problemas. El depurador trabaja con todos los lenguajes de programación de Visual Studio y sus bibliotecas asociadas. Con el depurador se puede interrumpir la ejecución del programa para examinar el código, evaluar y modificar variables, ver registros, muestras las instrucciones creadas a partir del código fuente y observar el espacio que ocupa la aplicación en memoria.

Para iniciar la depuración podemos hacerlo de las siguientes formas:

1. Desde el menú principal. Haciendo clic en el icono play 

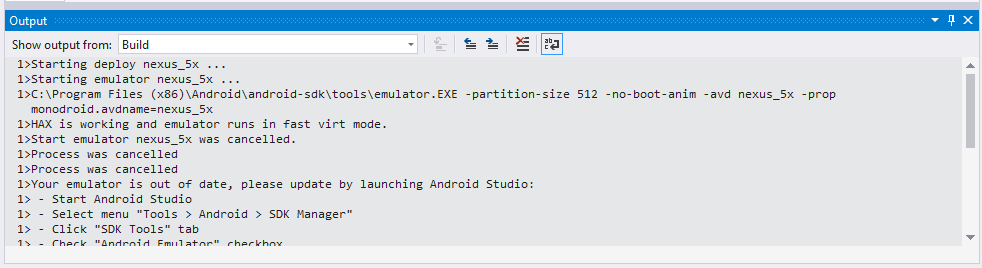


1. Desde el menú principal, pestaña Debug, en la opción Start Debugging.



**Ventana de Salida:**

La ventana de salida puede mostrar mensajes de estado de diversas características del entorno de desarrollo integrado (IDE). Para abrir la ventana Salida, en la barra de menú, elija View/Output.



**Ejercicio:**

Identificar los diferentes componentes del IDE de Visual Studio.

**Programa Informático:** Secuencia de instrucciones, escritas para realizar una tarea específica en una computadora.

**Lenguajes de Programación:** lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar.

**Ejemplos de programas:**

Los ejemplos que se mostraran acontinuacion están realizados utilizando el lenguaje de programación C# en XAMARIN.

**¿Qué es Xamarin?**

Xamarin es un conjunto de bibliotecas que contienen versiones .NET de las APIS nativas Mac, iOS y Android.

**¿Qué es una *App*?**

Es una aplicación de software que se instala en un dispositivo móvil o Tablet para ayudar a su usuario en su labor correcta, ya sea de carácter profesional o de ocio y entretenimiento.

Ejemplos de XAMARIN App (aplicación movil):

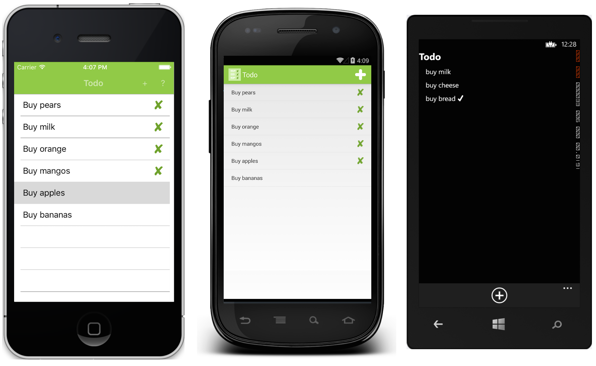


Ilustración 1App de categoría Comunicación.

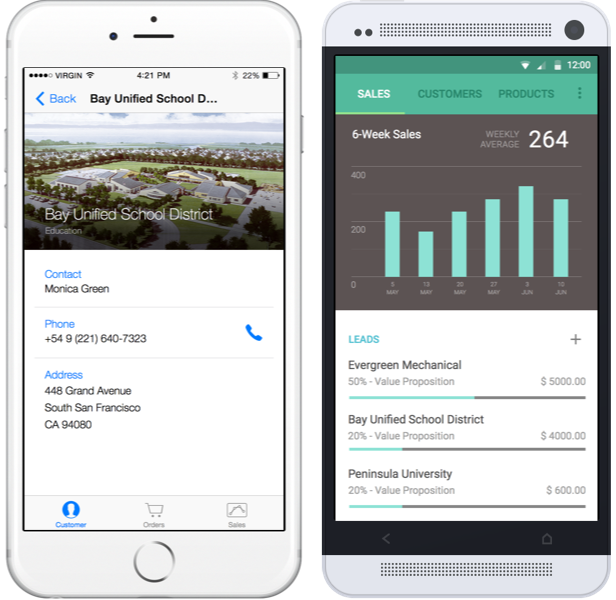


Ilustración 2 App de la categoría laboral/profesional.

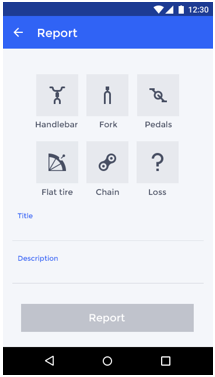


Ilustración 3 App de la categoría diversión/deportes.

**Ejercicio:**

Listar almenos 5 ejemplos de aplicaciones móviles.

**Practica:**

Revisar y analizar el código generado para el ejercicio de HolaXamarin.

### Actividad Propuesta1

ACTIVIDAD #1

Prueba Escrita.

**OBJETIVO:**

Identificar los elementos basicos del entorno de desarrollo en C# y conocer la funcion del depurador.

**INTRUCCIONES:**

1. Lea detenidamente cada una de las las preguntas planteadas en la prueba escrita.
2. Conteste correctamente cada una de las preguntas.
3. La respuesta debe ser escrita con tinta azul o negra.
4. Resuelva de manera puntual la prueva escrita Actividad #1 del manual uso de sintaxis y estructura de control en C# básico.



**TIEMPO ESTIMADO:**

40 minutos



**PARTICIPANTES:**

Individual



**MATERIALES**

LAPICERO TINTA AZUL

LAPICERO TINTA NEGRA

PRUEBA ESCRITA

## 3.1.2 Manipular las instrucciones básicas en C#.

**Sintaxis:** conjunto de reglas que deben seguirse al escribir el código fuente de los programas para considerarse como correctos dependiendo del lenguaje de programación a utilizar.

**Operadores:** En C#, un *operador* es un elemento de programa que se aplica a uno o varios *operandos* en una expresión o instrucción.

Los operadores que toman dos operandos, como los operadores aritméticos (+, -, \*, /) se conocen como operadores *binarios*.

**Declaración de Variables de Tipos de Datos Primitivos.**

C# es un lenguaje fuertemente tipado. Todas las variables y constantes tienen un tipo, al igual que todas las expresiones que se evalúan como un valor. Cada una de las firmas de método especifica un tipo para cada parámetro de entrada y para el valor devuelto. La biblioteca de clases .NET Framework define un conjunto de tipos numéricos integrados, así como tipos más complejos que representan una amplia variedad de construcciones lógicas, como el sistema de archivos, conexiones de red, colecciones y matrices de objetos, y fechas. Los programas de C# típicos usan tipos de la biblioteca de clases, así como tipos definidos por el usuario que modelan los conceptos que son específicos del dominio del problema del programa.

**Variable:** son espacios reservados en la memoria que, como su nombre indica, pueden cambiar de contenido a lo largo de la ejecución de un programa. Una variable corresponde a un área reservada en la memoria principal del ordenador.

**Tipos Enteros:**

Permite almacenar valores numéricos de tipo entero (no decimales) en variables.

Sintaxis:



**Tipos String:**

Permite almacenar cadenas de caracteres (texto) en variables.

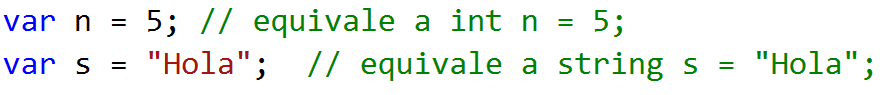
Sintaxis:



**Definición automática de tipos de variables en C#.**

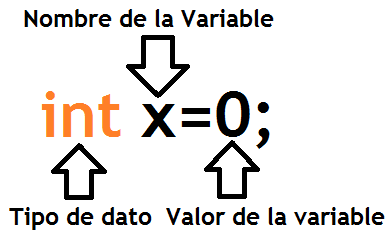
En C# es posible **dejar al compilador la tarea de determinar el tipo de una variable** que hayamos inicializado de manera explícita.

El tipo se deduce a partir de la expresión de inicialización. De este modo nos evitamos tener que declarar el tipo correcto, sobre todo en casos en los que no lo tenemos claro de antemano.

Así, por ejemplo, en el siguiente ejemplo el compilador infiere que la variable **n** es de tipo **int** y que **s** es de tipo **string** por el tipo de las constantes que se intenta asignar a cada una:

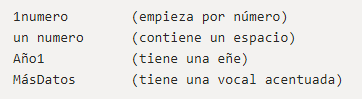
**Declarando variables en C#.**

Para poder declarar una variable en C# se debe seguir una sencilla sintaxis, indicando primero el tipo de datos que se almacenara, luego asignar el nombre de la variable teniendo en cuenta no usar palabras reservadas.



Los nombres de variables pueden estar formados por letras, números o el símbolo de subrayado (\_) y deben comenzar por letra o subrayado. No deben tener espacios intermedios. También hay que recordar que las vocales acentuadas y la eñe son problemáticas, porque no son letras "estándar" en todos los idiomas, así que no se pueden utilizar como parte de un identificador en la mayoría de lenguajes de programación

Ejemplo de nombres no validos:



**Ejemplo:** Declarar una variable que pueda almanenar un color.

string Color = ‘rojo’;

**Vista desde C#**

****

**Ejercicio:**

Declarar variables para almacenar la edad, nombre, apellidos y año de nacimiento.

**Estructuras de control en C#**

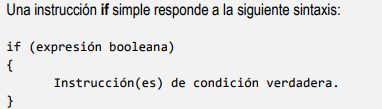
Las estructuras de control nos permiten controlar el flujo del programa: tomar decisiones, realizar acciones repetitivas etc, dependiendo de unas condiciones que nosotros mismos establezcamos.

**Estructuras condicionales**

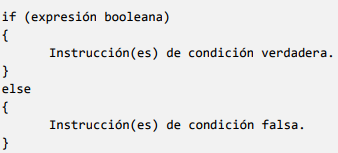
Las estructuras condicionales se utilizan para indicarle al computador que debe evaluar una condición y, a partir del resultado, ejecutar el bloque de instrucciones correspondiente.

**Sentencia if**

La instrucción **if** realiza (selecciona) una acción si una condición es verdadera, o ignora la acción si la condición es falsa.



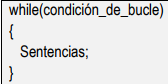
La instrucción **if…else** realiza una acción si una condición es verdadera o realiza una acción distinta si la condición es falsa.



**Estructuras Repetitivas en C#**

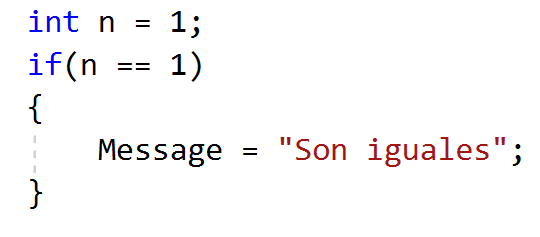
Las estructuras repetitivas se utilizan cuando se quiere que un conjunto de instrucciones se ejecuten un cierto número finito de veces, por ejemplo, escribir algo en pantalla cierta cantidad de veces, mover un objeto de un punto a otra cierta cantidad de pasos, o hacer una operación matemática cierta cantidad de veces.

**Estructura básica de un bucle "while":** Si queremos hacer que una sección de nuestro programa se repita mientras se cumpla una cierta condición, usaremos la orden "while". Esta orden tiene dos formatos distintos, según comprobemos la condición al principio o al final del bloque repetitivo.

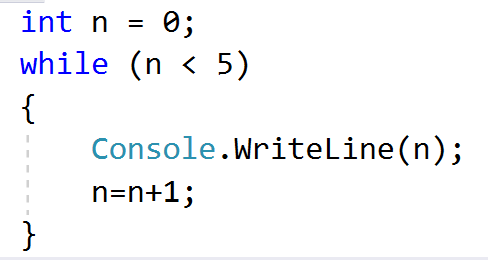


**Ejemplos:**

**Sentencia if**



**Sentencia while**



### Actividad Propuesta1I

ACTIVIDAD #2

Identificación de Sintaxis del Lenguaje C#

**OBJETIVO:**

Identificar la sintaxis de las instrucciones básicas de C#, para utilizar estructuras de control que permitan solucionar situaciones.

**INTRUCCIONES:**

1. Lea detenidamente cada una enuncia de los ejercicios planteados
2. Descargue el archivo con el nombre act2\_estructuras\_de\_control\_xamarin del repositorio del CDS.
3. Modique el archivo para que permita determinar si una persona es mayor de edad en base a su edad.



**TIEMPO ESTIMADO:**

50 minutos



**PARTICIPANTES:**

Individual



**MATERIALES**

Computadora

Internet

Repositorios guia

## 3.1 Módulo 9: C# Básico

## 3.1.3 Utilizar colecciones y bucle foreach para controlar elementos en C#.

Si hablamos de colecciones tendríamos que referirnos al Namespace System.Collections. Generic, en este espacio de nombre están en cerradas todas las clases Generic, trabajaremos con estas colecciones que son las mas comunes:

* 1. List
  2. Queue
  3. Stack
  4. Dictionary
  5. SortedList

**FOREACH.**

El foreach nos servirá para recorrer un listados de tipo de datos (int, double, string, etc.) o un listados de listado de objetos (entiéndase como objeto una clase creada previamente con sus respectivas variables y propiedades).

Al recorrer mediante el foreach, no necesitamos validar cuantos elementos componen nuestro arreglo, la instrucción funcionará hasta que lea el último elemento del arreglo o hasta que nosotros mediante una validación interrumpamos el flujo del ciclo.

foreach (‘Tipo’  Nombre in Listado)

{

                // Instrucciones.

}

**‘Tipo’**:

Se refiere al tipo de dato que va ligado al listado de objetos que queremos recorrer. Ejemplo:

Si el listado es de string, el valor en ‘**Tipo**’ será string.

Si el listado es de int, el valor en ‘**Tipo**’ será int.

Si el listado es de una clase Persona, el valor en ‘**Tipo**’ será ‘Persona’.

**‘Nombre’:**

Se refiere a cómo vamos a identificar el valor que obtendremos del listado.

**‘Listado’:**

Se refiere al listado de tipo de datos u de objetos que vamos a recorrer.

**¿Cómo romper el clico dentro del foreach?**

Para romper el ciclo dentro de la instrucción podemos usar la instrucción

·      **break**;  // Solo romperá el ciclo y seguirá con las demás instrucciones fuera del foreach que lo genero.

·      **throw**; // Para lanzar una excepción que se irá hasta el catch que hayas programado. Las demás instrucciones que hayas programado en el try ya no se ejecutarán, solo las que estén en catch.

·      **return**; // Esta instrucción romperá todo el ciclo impidiendo que continúen las demás instrucciones y devolviendo el control a la función principal o finalizando el programa. Si tienes más instrucciones fuera del código foreach estas ya no se ejecutarán.

Un "foreach" es al igual que un for, una manera iterativa de recorrer un array o cadena de elementos de un cierto tipo (int, string, object).

Lo bueno del foreach, es que ya no hay que preocuparse del tamaño del array, ya que no se sale de rango (out of bounds). Hay que tener en cuenta que no se puede cambiar el tamaño del array dentro del foreach.

Es una manera muy simple y rápida de construir un for. En el ejemplo siguiente se verá una cadena de tipo int que esta inicializada con los elementos 0,1,2,3,5,8,13.

Por ultimo hace un foreach para recorrer cada elemento de la cadena con el puntero "i".

"numero" es del tipo int porque cadena aloja objetos del tipo "int", de no seguir con esta regla se provoca un error en tiempo de compilación.

Un Ejemplo Aplicado del Foreach es:

var Numeros = new List<int> {1,2,3,4,5}; foreach (int numero in Numeros)

{

}

Ahora si quisieramos aplicarlo para imprimir un listado de numero en especifico seria:

foreach (int numero in Numeros)

{

ImprimirNumero(numero);

}

La salida seria una lista de números del 0 al 4.

Tambien se puede usar para listat elementos de una colección.

var Personas = new List<Persona>();

foreach (Persona persona in Personas)

{

ImprimirTexto(persona.Nombre);

}

Esto retornaria los nombres de las personas contenidas en la colección Personas:

Juan

Pedro

Andres

Rocío

Ángel

René

Aronette

Karla

### Actividad Propuesta 3

ACTIVIDAD #3

Uso de colecciones y Foreach

**OBJETIVO:**

Utilizar ciclo foreach para la resolucion de problemas

**INTRUCCIONES:**

1. Descargue el ejercicio con nombre: acti3\_foreach\_xamarin del repositorio del CDS.
2. Ejecute el codigo y tome nota de la salida.
3. Modifique el archivo para que la salida sean las cabezeras del servidor.
4. Modifique el archivo para que la salida sea los numeros del 1 al 10

**TIEMPO ESTIMADO:**

50 minutos

**PARTICIPANTES:**

Individual

**MATERIALES**

Computadora

Visual Studio

Repositorio de ejercicios



## Resultado de imagen para oficina vectorResultado de imagen para participantes vector3.1.4 Declaración de funciones que retornan VOID (vacío).

Un método es un bloque de código que contiene una serie de instrucciones. Un programa hace que las sentencias se ejecuten llamando al método y especificando cualquier argumento de método requerido. En C#, cada instrucción ejecutada se realiza en el contexto de un método.

Normalmente, cualquier aplicación que desarrollemos utilizando el Microsoft.NET Framework y Visual C#, tendrán varios métodos, cada uno con un propósito especifico. Algunos métodos son fundamentales para la operación de una aplicación. Por ejemplo, todas las aplicaciones de escritorio de Visual C#, deben tener un método llamada **Main** que define el punto de entrada de la aplicación. Cuando el usuario ejecuta una aplicación Visual C#, el CLR ejecuta el método **Main** de esa aplicación.

# Firmas de métodos

Un método está compuesto por dos elementos:

1. La especificación del método.
2. El cuerpo del método.

La especificación del método define el nombre del método, los parámetros que el método puede tomar, el tipo de valor devuelto del método y los modificadores de acceso del método.

La combinación del nombre del método y su lista de parámetros se conoce como **la firma del método.**

La definición del valor devuelto de un método **no** se considera como parte de la firma. Cada método en una clase debe tener una firma única.

**Nombrando un método**

un nombre de método tiene las mismas restricciones sintácticas que el nombre de una variable. El nombre del método debe comenzar con una letra o un guion bajo y solo puede contener letras, guiones y caracteres numéricos. Visual C# es sensible a mayúsculas y minúsculas, por lo que una clase puede contener dos métodos que tengan el mismo nombre y se diferencien solo en el uso de mayúsculas y minúsculas de una o más letras, aunque esto no es una buena práctica de codificación.

Dos lineamientos recomendados por Microsoft como una buena práctica al elegir el nombre del método son:

Utilizar verbos o frases de verbos para nombrar un método. Esto ayuda a que otros desarrolladores puedan entender la estructura de nuestro código.

Utilizar la nomenclatura *Pascal Casing.* No se recomienda iniciar los nombres de los métodos con un guion o una letra minúscula.

Un "foreach" es al igual que un for, una manera iterativa de recorrer un array o cadena de elementos de un cierto tipo (int, string, object).

Lo bueno del foreach, es que ya no hay que preocuparse del tamaño del array, ya que no se sale de rango (out of bounds). Hay que tener en cuenta que no se puede cambiar el tamaño del array dentro del foreach.

Es una manera muy simple y rápida de construir un for. En el ejemplo siguiente se verá una cadena de tipo int que esta inicializada con los elementos 0,1,2,3,5,8,13.

Por ultimo hace un foreach para recorrer cada elemento de la cadena con el puntero "i".

"numero" es del tipo int porque cadena aloja objetos del tipo "int", de no seguir con esta regla se provoca un error en tiempo de compilación.

**Implementando un cuerpo de Método**

el cuerpo de un método es un bloque de código que se implementa mediante el uso de cualquiera de las sentencias de programación de Visual C#. El cuerpo está encerrado entre llaves.

Podemos definir variables dentro del cuerpo de un método, en cuyo caso solo existirán mientras se esté ejecutando el método. Cuando el método finalice, las variables declaradas ya no serán accesibles.

El siguiente ejemplo muestra el código del método **ShowMessage** que contiene una variable llamada **Status**. La variable **Status** solo está disponible dentro del bloque de código **ShowMessage.** Si tratamos de utilizar la variable **Status** fuera del alcance del método, el compilador mostrará un mensaje de error indicando que ‘**Status’** no existe en el contexto actual.

void ShowMessage()

{

var Status = Service.Status

}

**Especificando parámetros**

Los parametros son variables locales que se crean cuando el método se ejecuta y les son asignados los valores que se especifican cuando el método es invocado. Los parámetros deben ser especificados entre paréntesis después del nombre del método. Cada parámetro está separado por una coma. Si un método no requiere algún parámetro, se especifica una lista de parámetros vacía (solo se incluyen los paréntesis después del nombre del método). El método **ShowMessage** descrito anteriormente es un ejemplo de un método que no requiere parámetros.

Para cada parámetro, se debe especificar el tipo y el nombre. Por convención, los parámetros utilizan la nomenclatura *Camel Casing.*

El siguiente ejemplo muestra cómo definir un parámetro de tipo **int**.

void SaveItems(int saveItems)

{

saveItems = Service.ItemsCount;

}

### Actividad Propuesta 4

ACTIVIDAD #4

Uso de metodos VOID

**OBJETIVO:**

Identificar y declarar metodos vacios sin retorno de manera correcta

**INTRUCCIONES:**

1. Descargue el ejercicio con nombre: acti4\_void\_xamarin del repositorio del CDS.
2. Ejecute el codigo y tome nota de la salida.
3. Modifique el archivo y agregue un nuevo metodo sin retorno.
4. Modifique el archivo y cree un metodo void cuyo nombre sea vacio\_xamarin

**TIEMPO ESTIMADO:**

50 minutos

**PARTICIPANTES:**

Individual

**MATERIALES**

Computadora

Visual Studio

Repositorio de ejercicios



### Resultado de imagen para participantes vectorResultado de imagen para oficina vector3.1.5 Comprender que es un framework y los elementos que lo componen en C# Xamarin.

## ¿Qué es .NET Framework?

.NET Framework es un entorno de ejecución administrado para Windows que proporciona diversos servicios a las aplicaciones en ejecución. Consta de dos componentes principales: Common Language Runtime (CLR), que es el motor de ejecución que controla las aplicaciones en ejecución, y la biblioteca de clases de .NET Framework, que proporciona una biblioteca de código probado y reutilizable al que pueden llamar los desarrolladores desde sus propias aplicaciones. Los servicios que ofrece .NET Framework a las aplicaciones en ejecución son los siguientes:

* Administración de la memoria. En muchos lenguajes de programación, los programadores son responsables de asignar y liberar memoria y de administrar la vida útil de los objetos. En las aplicaciones de .NET Framework, CLR proporciona estos servicios en nombre de la aplicación.
* Sistema de tipos comunes. En los lenguajes de programación tradicionales, el compilador define los tipos básicos, lo que complica la interoperabilidad entre lenguajes. En .NET Framework, los tipos básicos los define el sistema de tipos de .NET Framework y son comunes a todos los lenguajes que tienen como destino .NET Framework.
* Biblioteca de clases extensa. En lugar de tener que escribir cantidades extensas de código para controlar operaciones comunes de programación de bajo nivel, los programadores usan una biblioteca de tipos accesible en todo momento y sus miembros desde la biblioteca de clases de .NET Framework.
* Marcos y tecnologías de desarrollo. .NET Framework incluye bibliotecas para determinadas áreas de desarrollo de aplicaciones, como ASP.NET para aplicaciones web, ADO.NET para el acceso a los datos, Windows Communication Foundation para las aplicaciones orientadas a servicios y Windows Presentation Foundation para las aplicaciones de escritorio de Windows, la plataforma Xamarin para el desarrollo de aplicaciones moviles.



Información general sobre la plataforma Xamarin.

Poco después de que Microsoft anunciara el .NET Framework en junio del año 2000, la empresa **Ximian** fundada por Miguel de Icaza y Nat Friedman, inició un proyecto de código abierto llamado **Mono** para crear una implementación alternativa del compilador de C# y el .NET Framework que pudiera correr sobre Linux.

Una década después, en 2011, los fundadores de Ximian, que fue adquirida por Novell, fundaron **Xamarin.** Xamarin aun contribuye a la versión de código abierto de Monoj jy lo ha adaptado para formar la base de las soluciones móviles multiplataforma.

En mayo 28 de 2014, Xamarin introdujo Xamarin.Forms que permite a los desarrolladores escribir código de interfaz de usuario que puede ser compilado para dispositivos iOS, Android y Windows.

En marza de 2016, Microsoft adquirió Xamarin.Forms conel fin de proporcionar una opción de desarrollo móvil multiplatafomra a la extensa comunidad de desarrolladores Microsoft. Xamarin.Forms se encuentra ahora disponible de forma gratuita a todos los usuarios de Visual Studio.

**Un solo lenguaje para todas las plataformas.**

Durante los 3 primeros años de su existencia, Xamarin se enfocó principalente en tecnologías de compliacion y en 3 conjuntos básicos de bibliotecas .NET:

* Xamarin.Mac, que es una evolución del proyecto MonoMac.
* Xamarin.iOS, que es una evolución de MonoTouch.
* Xamarin.Android, que es una evolución de Mono para Android también conocido como MonoDroid.

Colectivamente, esas bibliotecas son conocidas como **la plataforma Xamarin.** Las bibliotecas contienen versiones .NET de las APIS nativas Mac, iOS y Android. Los programadores que utilicen esas bibliotecas pueden escribir aplicaciones en C# para acceder a als APIs nativas de esas 3 plataformas, pero, además como un bono adicional, pueden acceder a a la biblioteca de clases del .NET Framwork.

**Entiendo la plataforma Xamarin.Forms**

la plataforma Xamarin consiste en diversos elementos que nos permiten desarrollar aplicaciones para iOS, Android y Windows.

**El lenguaje C#**. Nos permite utilizar un sintaxis familiar y características sofisticadas como *Generic, Linq y la biblioteca Parallel Task Library*.

**Mono .NET Framework.** Proporciona una implementación multiplataforma de las extensas características del microsfot .NET Framework. Además, debido a que el lenguje base es C# con el .NET Framework, los proyectos pueden ser estructuraas para compartir código ente proyectos iOS, Android Windows.

**Compilador.** Genera un aaplicacon nativa dependieno de al plataforma. El compilador también realiza muchas optimizaciones para el despliegue móvil, tal como la no inclusión de código no utilizado.

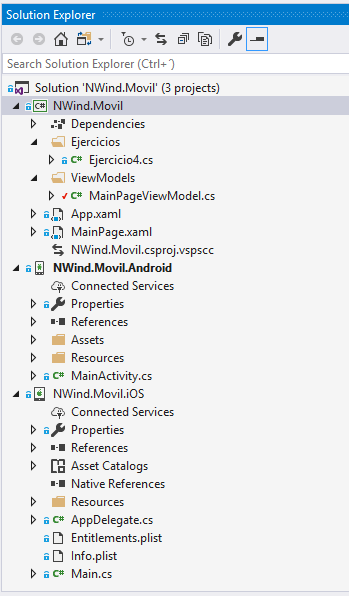
**IDE.** El IDE Xamarin Studio y Xamarin para Visual Studio permite crear, compilar y desplegar proyectos Xamarin.

Aunque Xamarin permite escribir aplicaciones en C# y compartir el mismo código a través de multiples plataformas, la implementación actual en cada sistema es diferente.

**Anatomía de una aplicación Xamarin**

Visual Studio es un IDE muy completo de Microsoft. Incluye un diseñador visual completamente integrado, un editor de texto completo con herramientas de refactorización, un explorador de ensamblados, integración de código fuente y mucho más. Este artículo se centra en el uso de algunas características básicas deVisual Studio con el complemento de Xamarin.

Visual Studio organiza el código en soluciones y proyectos. Una solución es un contenedor que puede incluir uno o varios proyectos. Un proyecto puede ser una aplicación, una biblioteca auxiliar o una aplicación de prueba, entre otros. La aplicación de Phoneword consta de una solución que contiene cuatro proyectos, como se muestra en la siguiente captura de pantalla.



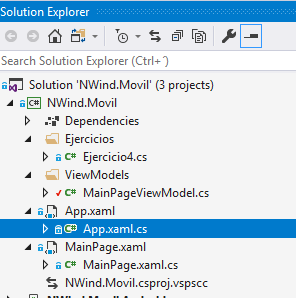
**Los proyectos son:**

**NWind.Movil**: este proyecto es el de la biblioteca de .NET Standard que incluye todo el código compartido y la interfaz de usuario compartida.

**NWind.Movil.Android**: este proyecto incluye el código específico de Android y es el punto de entrada de la aplicación Android.

**NWind.Movil.iOS:** este proyecto incluye el código específico de iOS y es el punto de entrada de la aplicación de iOS.

En la siguiente captura de pantalla se muestra el contenido del proyecto de la biblioteca de .NET Standard en Visual Studio:



El proyecto tiene un nodo Dependencias que contiene los nodos NuGet y SDK. El nodo NuGet contiene el paquete NuGet de Xamarin.Forms que se ha agregado al proyecto, y el nodo SDK contiene el metapaquete NETStandard.Library que hace referencia al conjunto completo de paquetes NuGet que definen .NET Standard.

El proyecto también consta de varios archivos:

**App.xaml**: el marcado XAML para la clase App, que define un diccionario de recursos para la aplicación.

**App.xaml.cs**: el código subyacente para la clase App, que es el responsable de crear instancias de la primera página que se mostrarán mediante la aplicación en cada plataforma, y para controlar los eventos del ciclo de vida de la aplicación.

**IDialer.cs**: la interfaz IDialer, que especifica que el método Dial debe proporcionarse mediante cualquier clase de implementación.

**MainPage.xaml**: el marcado XAML para la clase MainPage, que define la interfaz de usuario para la página que se muestra cuando se inicia la aplicación.

**MainPage.xaml.cs**: el código subyacente para la clase MainPage, que contiene la lógica de negocios que se ejecuta cuando el usuario interactúa con la página.

**Ejercicio4.cs**: la lógica de negocios responsable de convertir números con letras en un número de teléfono, que se invoca desde MainPage.xaml.cs.

ACTIVIDAD #5

DEFINICION DE FRAMEWORK Y SUS ELEMENTOS

**OBJETIVO:** Conocer la aplicación y elementos de una aplicación Xamarin.Forms

**INTRUCCIONES:**

1. Descargue el ejercicio con nombre acti5\_framework\_xamarin del repositorio del CDS
2. Abra su prioyecto en el explorador de soluciones y observe los componentes que forman su proyecto.
3. Indique los componentes primarios que conforman su proyecto.
4. Tome captura de pantalla de cada componente señalado.
5. Describa brevemente como es la relacion entre estos componentes.



**TIEMPO ESTIMADO:**

50 minutos



**PARTICIPANTES:**

Individual



**MATERIALES**

Computadora

Visual Studio

### Actividad Propuesta 5

# 3. GLOSARIO

**SINTAXIS:**

Conjunto de reglas que deben seguirse al escribir el código fuente de los programas para considerarse como correctos dependiendo del lenguaje de programación a utilizar.

**OPERADOR:**

Un operador en C# es un símbolo formado por uno o más caracteres que permite realizar una determinada operación entre uno o más datos y produce un resultado.

**VARIABLES:**

Son espacios reservados en la memoria que, como su nombre indica, pueden cambiar de contenido a lo largo de la ejecución de un programa. Una variable corresponde a un área reservada en la memoria principal del ordenador.

**FUNCIÓN:**

Conjunto de instrucciones que retornan un valor, la cual es llamada a través de un identificador.

**IDE:**

Un entorno de desarrollo integrado​​ o entorno de desarrollo interactivo, en inglés Integrated Development Environment (IDE), es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software.

**MÉTODO:**

Un método es un bloque de código que contiene una serie de instrucciones. Los programas hacen que las instrucciones se ejecuten mediante una llamada al método y la especificación de los argumentos de método necesarios.

**PARAMETRO:**

es una variable utilizada para recibir valores de entrada en una rutina, subrutina o método. Dichos valores, que serán enviados desde la rutina invocante, son llamados argumentos.

# 4. PLAN DE EVALUACIONES

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item | Actividad y  Tema evaluado | Tipo de evaluación | Semana | Ponderación |
| 1 | Actividad 1  Identificar que es un programa en informática y manejar correctamente el entorno de desarrollo de C# Xamarin. | Prueba Escrita | 1 | 20% |
| 2 | Actividad 2  Manipular las instrucciones básicas en C# Xamarin. | Archivo de Ejecucion | 2 | 20% |
| 3 | Actividad 3  Utilizar colecciones y bucle foreach para controlar elementos en C# Xamarin. | Archivo de Ejecucion | 3 | 20% |
| 4 | Actividad 4  Declarar de manera acertada los métodos de retorno vacío (VOID) en C# Xamarin. | Archivo de Ejecucion | 4 | 20% |
| 5 | Actividad 5  Comprender que es un framework y los elementos que lo componen en C# Xamarin. | Archivo de Ejecucion | 5 | 20% |
| **TOTAL** | | | | **100%** |

# 5. JORNALIZACIÓN DEL MÓDULO

|  |  |  |  |
| --- | --- | --- | --- |
| SEMANA | SUB COMPETENCIA | ESCENARIO DE APRENDIZAJE | DURACIÓN EN HORAS |
| 1 | Identificar que es un programa en informática y manejar correctamente el entorno de desarrollo de C# Xamarin |  | 2 horas |
| 2 | Manipular las instrucciones básicas en C# Xamarin. |  | 8 horas |
| 3 | Utilizar colecciones y bucle foreach para controlar elementos en C# Xamarin. |  | 8 horas |
| 4 | Declarar de manera acertada los métodos de retorno vacío (VOID) en C# Xamarin. |  | 6 horas |
| 5 | Comprender que es un framework y los elementos que lo componen en C# Xamarin. |  | 8 horas |
| **TOTAL DE HORAS DEL MÓDULO** | |  | **32 HORAS** |

# 6. FUENTES DE INFORMACIÓN Y MATERIALES DE APOYO

Landa Cosio, Nicolas Arrioja. (2010). *C# Guia Total del Programador.* Buenos Aires: RedUSERS.

Ferguson, Jeff. (2003). *La Biblia de C#.* Madrid: Anaya Multimedia.

González Seco, José Antonio. *El lenguaje de programación C#.*

*Introducción a Visual C# y Visual Basic* - https://msdn.microsoft.com/es-es/library/dd492171.aspx - Consultado el 20 – 02 – 2018

*Documentacion C#* - https://msdn.microsoft.com/es-es/ - Consultado el 21 – 02 – 2018

*Programacion en C#* - http://www.nachocabanes.com/csharp/curso2015/index.php - Consultado el 21 – 02 – 2018

*Orden de Operaciones* - http://ponce.inter.edu/cremc/operacion.html - Consultado el 21 - 02 – 2018