LENGUAJE DE PROGRMACION

C#

1. HISTORIA

Let’s start with the first set of slides

HISTORIA

C# (C Sharp) es un lenguaje de programación moderno desarrollado por Microsoft. Fue creado por Anders Hejlsberg y su equipo en la década de 1990 como parte de la plataforma .NET de Microsoft. C# se diseñó específicamente para ser un lenguaje de programación simple, seguro, orientado a objetos y de propósito general.

La historia de C# se remonta al año 2000, cuando Microsoft presentó la plataforma .NET, una infraestructura de desarrollo de software que permitía la creación de aplicaciones para Windows utilizando varios lenguajes de programación. C# fue desarrollado como el lenguaje principal para la plataforma .NET.

El diseño de C# se basó en gran medida en el lenguaje C++ y en otros lenguajes de programación como Java y Delphi. Aunque comparte similitudes sintácticas con C y C++, C# introdujo conceptos de programación orientada a objetos más avanzados y una sintaxis más sencilla y clara.

C# fue lanzado oficialmente junto con la primera versión de la plataforma .NET en febrero de 2002. Desde entonces, ha habido varias versiones y actualizaciones del lenguaje, cada una introduciendo nuevas características y mejoras.

En 2005, Microsoft lanzó C# 2.0, que introdujo características como los genéricos, los delegados anónimos y los métodos parciales. Luego, en 2007, se lanzó C# 3.0, que incluía características importantes como la inferencia de tipos var, las expresiones lambda, las consultas LINQ y los métodos de extensión.

C# 4.0 se lanzó en 2010 y agregó características como los parámetros opcionales, los argumentos con nombre y la covarianza y contravarianza en los tipos genéricos. C# 5.0, lanzado en 2012, introdujo el soporte para programación asincrónica con las palabras clave async y await.

Luego, en 2015, se lanzó C# 6.0, que incluía mejoras en la sintaxis y nuevas características como la interpolación de cadenas, la inicialización automática de propiedades y el uso seguro de la palabra clave null con el operador "?". C# 7.0 (2017), C# 7.1 (2017), C# 7.2 (2017), C# 7.3 (2018) y C# 8.0 (2019) continuaron expandiendo el lenguaje con características como patrones de coincidencia, referencias locales y nulas, tipos de función y mucho más.

En noviembre de 2020, Microsoft lanzó .NET 5.0, que incluía C# 9.0. C# 9.0 presentó características como los registros, inicializadores de propiedades con constructores, funciones estáticas locales, patrones de coincidencia mejorados y mejoras en LINQ.

La historia de C# continúa en constante evolución, con Microsoft lanzando nuevas versiones y actualizaciones periódicas para agregar características y mejoras al lenguaje. C# se ha convertido en uno de los lenguajes de programación más populares en el desarrollo de aplicaciones empresariales y de escritorio en el ecosistema de Microsoft y también se utiliza ampliamente para el desarrollo de aplicaciones web y móviles utilizando frameworks como ASP.NET y Xamarin.

¿QUÉ ES C# ?

Lanzado en 2001 por Microsoft, **inspirado en C++, Java y Pascal para funcionar con la plataforma .NET.**

Anders Hejisberg
Creador de C#

CARACTERÍSTICAS PRINCIPALES

- **Multiplataforma**

- Sintaxis similar a C++ y Java para **atraer a esos desarrolladores.**

- **Multiparadigma:** tiene diferentes estilos para programar. Aunque está **principalmente orientado a objetos**, también **se puede dirigir a eventos y funciones.**

SE UTILIZA PARA:

- Construcción de **videojuegos (Unity).**

unity

- **Aplicaciones de escritorio.**

mono

- Aplicaciones del estándar **web assembly (Blazor).**

WA

- **Aplicaciones móviles nativas** para Android y iOS (Xamarin).

Si deseas dominar C#, este es el curso que estabas buscando:

ed.team/cursos/csharp

EDteam

2. VERSION ACTUAL

La versión estable más reciente de C#.

La versión más reciente de C# es la 11.0 (Aunque hay otra versión todavía en desarrollo). La versión 11 de C# salió en noviembre de 2022 y puede usarse en las versiones más recientes de Visual Studio.

11

FEATURES

C# Versión 11.0

- Mejoras en el String.
- Strings UTF-8.
- Patrones de listas.
- Tipos de archivos locales.
- Valores "*struct*" predeterminados.
- Advertencia 7

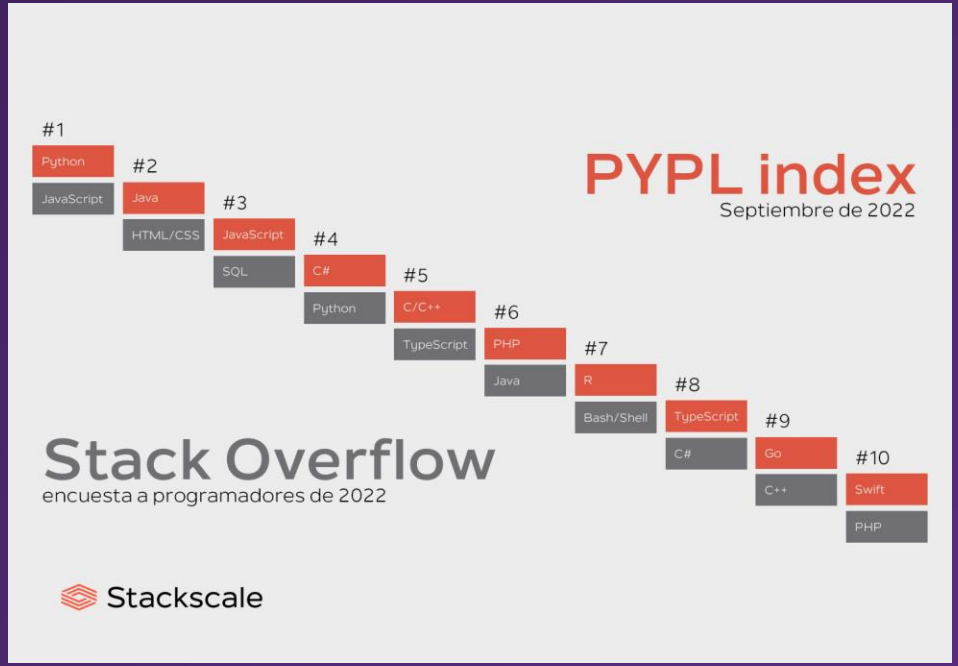
Hay más cambios respecto a las versiones anteriores, pero estos son los más importantes.

Ubicación en Ranking y Utilidad del lenguaje.

Hay rankings que nos permiten averiguar qué posición en el mercado ocupan ciertos lenguajes de programación, y C# no es la excepción.

Ubicación en el Ranking

C# es de los lenguajes más demandados en el mercado actual, puesto que, según el ranking PYPL, C# ocupa el puesto #4 en la lista de los lenguajes más demandados en septiembre de 2022, con aproximadamente 18.000 trabajos y un salario medio anual de $97.000; superando a lenguajes como C, C++ y PHP (Aunque los puestos de este último pueden variar según las fuentes).



Utilidad del lenguaje

Ventajas:

Es mucho más rápida y segura que lenguajes como C, permite el desarrollo de aplicaciones para Windows, complementos y apps para móviles; además de ser usado por motores de desarrollo como Unity y ofrecer una gran variedad de tipos de datos.

Desventajas:

La curva de aprendizaje en C# puede ser un poco más complicada que en otros lenguajes al iniciar con la programación orientada a objetos. También, algunos aseguran que lenguajes como C++ tienen un mejor rendimiento que C# a pesar de ser considerados como "inferiores". Además de necesitar implementaciones extra para poder ser ejecutado en sistemas operativos no-Microsoft como Linux por ejemplo.

Usos Comunes:

C# se usa bastante en el desarrollo de juegos, puesto que es el lenguaje que usa el motor de creación de juegos más popular: Unity. También se usa bastante para desarrollar aplicaciones de escritorio para Windows, además de que también se puede emplear en el desarrollo de aplicaciones web mediante ASP.NET.

CONCEPTOS DE CLASES Y OBJETOS

CLASES

C# es un lenguaje de programación orientado a objetos, por lo tanto, los programas y aplicaciones realizados con este lenguaje están compuestos por clases, a partir de las cuales se crean objetos, que colaboran entre sí para resolver los problemas o necesidades de las aplicaciones.

Una clase es un modelo o plantilla que define las características y comportamientos de un objeto. En C#, se define una clase utilizando la palabra clave "class".

```
class MiClase
{
    // Propiedades
    public int MiPropiedad { get; set; }

    // Métodos
    public void MiMetodo()
    {
        // Código del método
    }
}
```

OUR PROCESS IS EASY

PROPIEDADES

Las propiedades son miembros de una clase que permiten acceder y manipular los datos de un objeto. Pueden tener un acceso público, protegido o privado, y se definen utilizando los modificadores de acceso "public", "protected" o "private".

OBJETO

un objeto es una instancia de una clase. Una clase define la estructura y el comportamiento de un de objeto, mientras que un objeto es una entidad concreta creada en tiempo de ejecución basada en esa clase. puedes crear múltiples objetos de esa clase usando la palabra clave "new".

METODO

Los métodos son bloques de código que encapsulan una funcionalidad específica. Pueden recibir parámetros y devolver valores. Puedes definir métodos dentro de una clase y luego llamar a esos métodos en objetos de esa clase.

ENCAPSULAMIENTO

en C# es un concepto clave de la programación orientada a objetos que permite controlar el acceso a los miembros de una clase, como propiedades y métodos, para proteger los datos internos y ocultar su implementación.

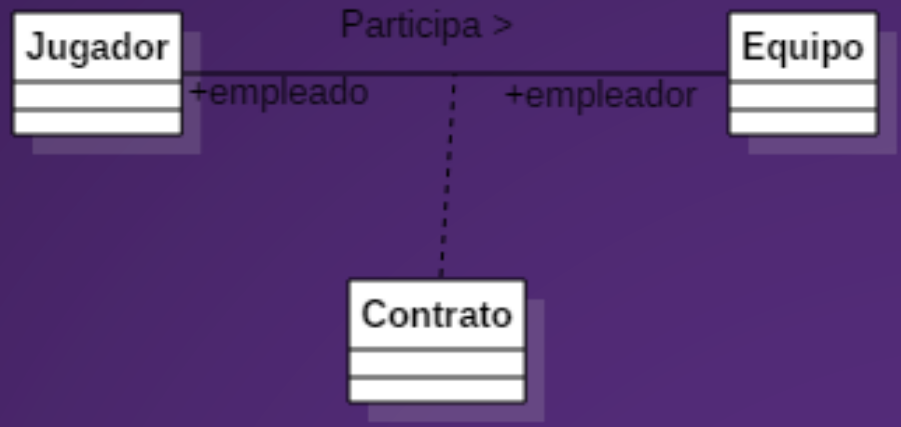
- public: El miembro es accesible desde cualquier parte del programa.
- private: El miembro es accesible solo desde dentro de la propia clase.
- protected: El miembro es accesible desde la propia clase y desde las clases derivadas (heredadas).
- internal: El miembro es accesible dentro del ensamblado actual.

```
class MiClase
{
    private int miVariablePrivada; // Variable privada

    public int MiPropiedadPublica { get; set; } // Propiedad pública

    public void MetodoPublico()
    {
        // Código del método público
        miVariablePrivada = 10; // Acceso a la variable privada
        MiPropiedadPublica = 20; // Acceso a la propiedad pública
        MetodoPrivado(); // Llamada al método privado
    }

    private void MetodoPrivado()
    {
        // Código del método privado
        // Solo puede ser llamado desde dentro de la clase
    }
}
```

Conceptos de Contenedores / Asociación, Agregación Composición

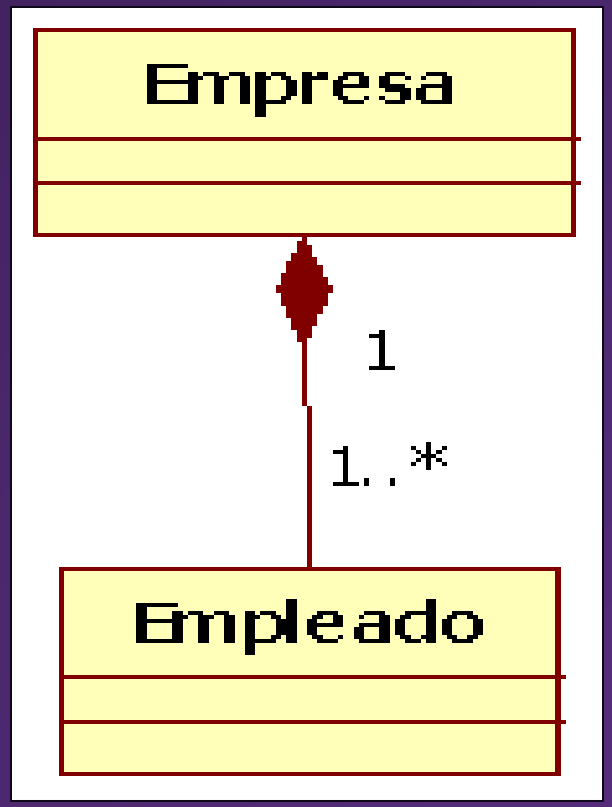
Asociación

La asociación describe una relación entre dos objetos en la que cada uno de ellos puede existir de forma independiente del otro. Esta relación puede ser de uno a uno, uno a muchos o muchos a muchos. Los objetos asociados pueden interactuar entre sí a través de métodos y propiedades, pero no dependen uno del otro para existir. Por ejemplo, en un sistema de gestión de una biblioteca, puede haber una asociación entre las clases "Libro" y "Autor". Cada libro tiene un autor asociado, pero tanto el libro como el autor pueden existir por separado.



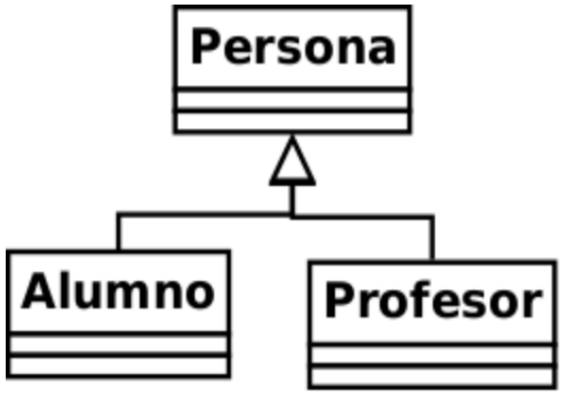
Agregación

La agregación describe una relación entre dos objetos, pero en este caso, uno de los objetos es una "parte" o "componente" del otro objeto. La relación de agregación implica que un objeto puede contener o incluir a otro objeto, pero ambos pueden existir de forma independiente. La agregación es una relación de "todo-parte". Por ejemplo, en un sistema de gestión de una universidad, puede haber una clase "Universidad" que agregue objetos de la clase "Estudiante". Los estudiantes pueden existir por separado y pueden ser parte de la universidad, pero no dependen de la universidad para existir.



Composición

La composición es una forma en la que los objetos están estrechamente vinculados y uno depende del otro para existir. En una relación de composición, el objeto "todo" es responsable de la creación y destrucción de los objetos "parte". Si el objeto "todo" se destruye, también se destruyen los objetos "parte". Por ejemplo, en un sistema de gestión de un automóvil, puede haber una clase "Automóvil" que se compone de objetos de las clases "Motor", "Ruedas" y "Carrocería". Si se destruye el objeto automóvil, también se destruirán los objetos motor, ruedas y carrocería.




```
public class Persona{
....
}
public class Alumnos extends Persona{
....
}
public class Profesor extends Persona{
....
}
```

Herencia

La herencia es un mecanismo que permite definir una clase nueva basada en una clase existente. La nueva clase hereda todas las propiedades y métodos de la clase base llamada superclase y puede agregar nuevos atributos y métodos o modificar los existentes. La herencia permite la creación de una jerarquía de clases, donde las clases más específicas llamadas subclases heredan características de las clases más generales.

Por ejemplo, podemos tener una clase base llamada "Animal" con atributos y métodos genéricos como "nombre" y "hacerSonido()". Luego, podemos crear subclases como "Perro" y "Gato" que hereden de la clase "Animal". Estas subclases pueden agregar atributos específicos como "raza" y sobrescribir métodos existentes o agregar nuevos métodos.

La herencia permite la reutilización de código, evita la duplicación y permite modelar de manera más precisa las relaciones entre los objetos del dominio.

Polimorfismo

El polimorfismo es la capacidad de objetos de diferentes clases de responder al mismo mensaje o método de una manera específica para cada uno de ellos. En otras palabras, el polimorfismo permite tratar objetos de diferentes clases de forma uniforme a través de una interfaz común.

El polimorfismo se basa en la herencia y se logra mediante la sobrescritura de métodos en las subclases. Cuando se invoca un método en un objeto, el comportamiento del método puede variar según el tipo real del objeto en tiempo de ejecución.

El polimorfismo permite tratar objetos de diferentes clases como si fueran del mismo tipo, lo que facilita la creación de código más flexible y extensible.

GUI: Desktop, CLI , Web y móvil

Aplicaciones de escritorio:

se utiliza comúnmente para desarrollar aplicaciones de escritorio en Windows utilizando el framework Windows Presentation Foundation(WPF) o Windows Forms. Estos frameworks permiten crear interfaces gráficas de usuario(GUI) interactivas y ricas en funcionalidades para aplicaciones de escritorio.

Aplicaciones de línea de comandos(CLI)

también se puede utilizar para desarrollar aplicaciones de línea de comandos que se ejecutan en una interfaz de consola. Estas aplicaciones suelen ser útiles para tareas automatizadas, scripts y utilidades de línea de comandos.

Aplicaciones web:

es ampliamente utilizado en el desarrollo de aplicaciones web mediante el framework ASP.NET. ASP.NET permite crear aplicaciones web dinámicas y escalables, utilizando el patrón de diseño Modelo-Vista-Controlador(MVC) o el modelo de páginas Razor. Además, C# puede ser utilizado en conjunto con tecnologías web como HTML, CSS y JavaScript para crear aplicaciones web completas.

Aplicaciones móviles

también se puede utilizar para desarrollar aplicaciones móviles para plataformas como Android e iOS. Esto es posible gracias a frameworks como Xamarin, que permite crear aplicaciones móviles nativas utilizando C#. Con Xamarin, se puede compartir gran parte del código entre las versiones de Android e iOS, lo que agiliza el proceso de desarrollo.

TEAM PRESENTATION

Juan Camilo Uribe

1152326

Faiber Galvis Romero

1152310

Juan David Mendoza

1152295

Julián Hernández

1152302