Desc.

C# cambia en la sintaxis al programar y nos ofrece otras funcionalidades y paradigma al programar, aunque como python también es un lenguaje fuertemente tipado

Tipos de datos

La mayoría de datos son compartidos entre muchos lenguajes aunque siempre existen excepciones.

Números

- **sbyte**: Entero con signo de 8 bits.
  - Rango: `-128 a 127`.
- **byte**: Entero sin signo de 8 bits.
  - Rango: `0 a 255`.
- **short**: Entero con signo de 16 bits.
  - Rango: `-32,768 a 32,767`.
- **ushort**: Entero sin signo de 16 bits.
  - Rango: `0 a 65,535`.
- **int**: Entero con signo de 32 bits.
  - Rango: `-2,147,483,648 a 2,147,483,647`.
- **uint**: Entero sin signo de 32 bits.
  - Rango: `0 a 4,294,967,295`.
- **long**: Entero con signo de 64 bits.
  - Rango: `-9,223,372,036,854,775,808 a 9,223,372,036,854,775,807.`
- **ulong**: Entero sin signo de 64 bits.
  - Rango: `0 a 18,446,744,073,709,551,615.`

Punto flotante

- **float**: Número de punto flotante de precisión simple de 32 bits.
- Rango: `7 dígitos de precisión.`

- **double**: Número de punto flotante de doble precisión de 64 bits.
- Rango: `15-16 dígitos de precisión.`
- **decimal**: Número decimal de alta precisión de 128 bits.
    - Rango: `28 dígitos de precisión.`

Otros númericos

- `char`: Representa un solo carácter Unicode de 16 bits..
- `bool`: Representa un valor booleano, que puede ser `true` o `false` | `0` o `1`.

Cadenas | Strings

- **string:** cadenas de caracteres definidas por comillas simples, dobles o invertidas
- **char**: Representa un único carácter.
    - Se utiliza para almacenar un único carácter, como letras, dígitos, signos de puntuación, etc. Se representa con comillas simples (`''`).

Decisiones | Booleanos

- **bool:** solo pueden tener dos valores (true | false)

Conjuntos | Listas, Arrays etc..

- **Listas:**
    - importar la clase `using System.Collections.Generic;`
    - Listas de un solo tipo
    - Definida por objetos con `new`
    - Acceso a través de índices

```
// Lista números
List<int> numeros = new List<int>();
// Agregar elementos a la lista
numeros.Add(10);
// Acceder a elementos de la lista por índice
Console.WriteLine("Elemento en el índice 1: " + numeros[1]);
foreach (int numero in numeros)
{
        Console.WriteLine(numero);
}
```

- **Tuplas:**

- Se definen por paréntesis **( )** y separando elementos por **,**
- Pueden contener elementos de diferentes tipos
- Se accede por **.** o **desestructuración**
- Colección de datos que no cambian
- Inmutables

```
// Tupla con dos elementos
var persona = ("Juan", 30);

// Acceder a los elementos de la tupla utilizando la notación de punto
Console.WriteLine("Nombre: " + persona.Item1);
Console.WriteLine("Edad: " + persona.Item2);

// Desestructurar la tupla para acceder a sus elementos
(string nombre, int edad) = persona;
Console.WriteLine("Nombre: " + nombre);
Console.WriteLine("Edad: " + edad);

// Cambiar valores de la tupla ( No debemos porque por teoría son inmutables )
persona.Item1 = "Pedro";
persona.Item2 = 35;

Console.WriteLine("Nombre: " + persona.Item1);
Console.WriteLine("Edad: " + persona.Item2);
```

- **Conjuntos:**
  - Uso de la clase `using System.Collections.Generic;`
  - Colección **NO** ordenada
  - Sin elementos duplicados
  - Pueden contener elementos de diferentes tipos
  - **NO** se puede acceder con índice
  - Búsqueda eficiente
  - Útiles para conjuntos que nunca cambian

```
using System.Collections.Generic;

//Conjunto de números enteros
HashSet<int> numeros = new HashSet<int>();

// Agregar elementos al conjunto
numeros.Add(10);
```

```
// Eliminar un elemento del conjunto
numeros.Remove(20);

// Intentar agregar un elemento duplicado
numeros.Add(10); // Este elemento no se agregará

// Mostrar el tamaño del conjunto (elementos únicos)
Console.WriteLine("Número de elementos en el conjunto: " + numeros.Count);

// Iterar sobre el conjunto e imprimir sus elementos
Console.WriteLine("Elementos del conjunto:");
foreach (int numero in numeros)
{
        Console.WriteLine(numero);
}
```

- **Diccionarios:**
  - Se crea mediante **clave-valor**
  - Se definen con llaves **{ }**
  - claves únicas e induplicables
  - Se accede mediante claves
  - Representa datos estructurados

```
using System;
using System.Collections.Generic;

class Program
{
    static void Main(string[] args)
    {
        // Diccionario: personaje
        Dictionary<string, dynamic> personaje = new Dictionary<string, dynamic>()
        {
            { "nombre", "Aragorn" },
            { "clase", "Guerrero" },
            { "nivel", 50 },
            { "puntos de vida", 1000 },
            { "puntos de magia", 200 },
            { "arma", "Espada Andúril" },
            { "armadura", "Coraza de Mithril" },
            { "habilidades", new List<string> { "Corte Mortal", "Grito de Guerra",
"Defensa Impenetrable" } }
        };
```

```
        // Acceder a los valores
        Console.WriteLine("Nombre del personaje: " + personaje["nombre"]);
        Console.WriteLine("Clase del personaje: " + personaje["clase"]);
        Console.WriteLine("Nivel del personaje: " + personaje["nivel"]);
        Console.WriteLine("Puntos de vida del personaje: " + personaje["puntos de
vida"]);
        Console.WriteLine("Puntos de magia del personaje: " + personaje["puntos de
magia"]);
        Console.WriteLine("Arma del personaje: " + personaje["arma"]);
        Console.WriteLine("Armadura del personaje: " + personaje["armadura"]);
        Console.Write("Habilidades del personaje: ");
        foreach (string habilidad in personaje["habilidades"])
        {
            Console.Write(habilidad + ", ");
        }
    }
}
```

Lanzamiento de dados

Crearemos un programa que simule en bucle una tirada de dados en la que el usuario decidirá si quiere una tirada más, si su respuesta es sí, recorreremos el bucle infinitamente, en caso de NO, saldremos del programa.

Procedimiento

- Hacer uso de una constante en el programa
- Mostrar el resultado en terminal
- Hacer uso de la función `ReadLine()` y `WriteLine()`

Ejercicio

```
# Titulo programa

# Bucle

# Mostrado en pantalla
```

Simulador de rol

Crearemos un juego de rol en el que al inicio deberemos indicar la información de nuestro personaje utilizando variables asignadas por el usuario y constantes estáticas.

Además de esto deberemos simular una batalla en el que el personaje recibirá 20 pts de daño y mostrar sus estadísticas de nuevo.

Información

- Nombre
- Salud
- Nivel
- Experiencia
- Salud máxima --> Constante

Ejercicio

```
// Definir estadisticas

// Mostrar datos

// Resultado batalla

// Mostrar datos
```

Resultado

Ejercicio 1

```
using System;

class Program
{
    static void Main()
    {
        // Generar un objeto de tipo Random para simular el lanzamiento del dado
        Random random = new Random();

        // Definir una variable para almacenar el resultado del lanzamiento del
dado
        int resultadoDado;
```

```
        // Mensaje de bienvenida
        Console.WriteLine("¡Bienvenido al Simulador de Lanzamiento de Dado!");

        // Bucle para permitir múltiples lanzamientos del dado
        while (true)
        {
            // Pedir al usuario que presione una tecla para lanzar el dado
            Console.WriteLine("\nPresiona cualquier tecla para lanzar el dado...");
            Console.ReadKey(true);

            // Generar un número aleatorio entre 1 y 6 para simular el lanzamiento
del dado
            resultadoDado = random.Next(1, 7);

            // Mostrar el resultado del lanzamiento del dado
            Console.WriteLine("¡Has obtenido un " + resultadoDado + "!\n");

            // Preguntar al usuario si desea lanzar el dado nuevamente
            Console.Write("¿Quieres lanzar el dado nuevamente? (s/n): ");
            char respuesta = Console.ReadKey(true).KeyChar;

            // Salir del bucle si el usuario no quiere lanzar el dado nuevamente
            if (respuesta != 's' && respuesta != 'S')
            {
                Console.WriteLine("\n\n¡Gracias por jugar!");
                break;
            }
        }
    }
}
```

Ejercicio 2

```
using System;

class Program
{
    static void Main()
    {
        // Solicitar al usuario que ingrese la información del personaje
        Console.WriteLine("¡Bienvenido al juego de rol!");
        Console.Write("Por favor, ingresa el nombre de tu personaje: ");
        string nombre = Console.ReadLine();
```

```
        Console.Write("Ingresa la salud inicial de tu personaje: ");
        int salud = Convert.ToInt32(Console.ReadLine());

        Console.Write("Ingresa el nivel de tu personaje: ");
        int nivel = Convert.ToInt32(Console.ReadLine());

        Console.Write("Ingresa la experiencia de tu personaje: ");
        int experiencia = Convert.ToInt32(Console.ReadLine());

        // Constante para la salud máxima
        const int SaludMaxima = 100;

        // Mostrar la información del personaje
        Console.WriteLine("\nInformación del personaje:");
        Console.WriteLine("Nombre: " + nombre);
        Console.WriteLine("Salud: " + salud + "/" + SaludMaxima);
        Console.WriteLine("Nivel: " + nivel);
        Console.WriteLine("Experiencia: " + experiencia);

        // Simular una batalla y actualizar la salud del personaje
        int danioRecibido = 20;
        salud -= danioRecibido;

        // Mostrar las estadísticas actualizadas del personaje después de la
batalla
        Console.WriteLine("\nDespués de la batalla:");
        Console.WriteLine("Salud: " + salud + "/" + SaludMaxima);
    }
}
```