A **is** operátort C#-ban **típusellenőrzésre** használjuk. Segít megállapítani, hogy egy objektum egy adott típusba tartozik-e.

**1️⃣ Alap használat: Típusellenőrzés**

A **is** operátor azt vizsgálja, hogy egy objektum **egy adott típusba tartozik-e**.  
Ha igen, akkor **true**, ha nem, akkor **false**.

🔹 **Példa:**

object obj = "Hello";

if (obj is string)

{

Console.WriteLine("Ez egy string!");

}

✅ **Mivel az obj egy "Hello" string, az is string igaz lesz.**

🔹 **További példák:**

object obj1 = 42;

object obj2 = "Text";

Console.WriteLine(obj1 is int); // true (mert 42 egy int)

Console.WriteLine(obj2 is string); // true (mert "Text" egy string)

Console.WriteLine(obj1 is string); // false (mert 42 nem string)

**2️⃣ is operátor és öröklődés**

Az **is** operátor az **öröklődő osztályoknál** is működik.

🔹 **Példa öröklődéssel:**

class Animal { }

class Dog : Animal { }

Animal myPet = new Dog();

Console.WriteLine(myPet is Dog); // true (mert myPet egy Dog példány)

Console.WriteLine(myPet is Animal); // true (mert Dog örökli az Animal osztályt)

Console.WriteLine(myPet is string); // false (mert nem string)

✅ **Ha egy objektum egy leszármazott osztály példánya, az is igaz lesz az ősosztályra is.**

**3️⃣ is és as operátor kombinálása**

Ha az is **igaz**, akkor gyakran szükséges az átalakítás (casting).  
Erre használhatod az **as** operátort.

🔹 **Példa:**

object obj = "Hello";

if (obj is string str)

{

Console.WriteLine($"A string hossza: {str.Length}");

}

🔍 **Mit csinál ez a kód?**

* Ha az obj **string**, akkor a str változóban megkapja az értékét.
* Ha **nem string**, akkor a str változó **nem jön létre**.

Ez **hatékonyabb**, mint egy külön típusellenőrzés és átalakítás:

if (obj is string)

{

string str = (string)obj; // Régi módszer (kettő helyett egy lépés)

Console.WriteLine(str.Length);

}

**4️⃣ is not – Az ellenkezője**

C# 9.0-tól lehetőség van az **is not** használatára.

🔹 **Példa:**

object obj = 123;

if (obj is not string)

{

Console.WriteLine("Nem egy string!");

}

Ez egy **olvasmányosabb módja** a !(obj is string) írásának.

**5️⃣ is használata absztrakt osztályokkal és interfészekkel**

Az is hasznos, ha egy objektum egy **interfészt** vagy **absztrakt osztályt** valósít meg.

🔹 **Példa interfésszel:**

interface IMovable

{

void Move();

}

class Car : IMovable

{

public void Move() => Console.WriteLine("Autó mozgásban!");

}

IMovable obj = new Car();

if (obj is IMovable)

{

Console.WriteLine("Ez egy mozgatható objektum!");

}

✅ **Mivel a Car implementálja az IMovable interfészt, az is IMovable igaz.**

**Összegzés**

| **Kifejezés** | **Mit csinál?** | **Eredmény** |
| --- | --- | --- |
| obj is string | Ellenőrzi, hogy az obj egy string-e | true vagy false |
| obj is int | Ellenőrzi, hogy az obj egy int-e | true vagy false |
| obj is Animal | Ellenőrzi, hogy az obj egy Animal-példány (vagy annak leszármazottja) | true vagy false |
| obj is IMovable | Ellenőrzi, hogy az obj megvalósítja-e az IMovable interfészt | true vagy false |
| obj is not string | Ellenőrzi, hogy az obj **nem** string | true vagy false |
| if (obj is string str) | Ha obj egy string, akkor str változóként használhatjuk | Hatékonyabb, mint külön ellenőrzés és castolás |

Az **is operátor nagyon hasznos**, különösen **öröklődés, interfészek és típusbiztonság** esetén! 🚀