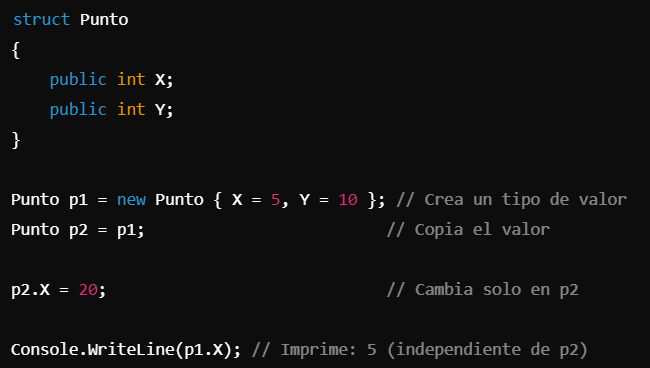
La diferencia entre **tipos de valor** (almacenados en el **stack**) y **tipos de referencia** (almacenados en el **heap**) es fundamental para comprender cómo funciona C# y por qué los struct tienen un comportamiento distinto de las clases. Aquí tienes una explicación clara con ejemplos:

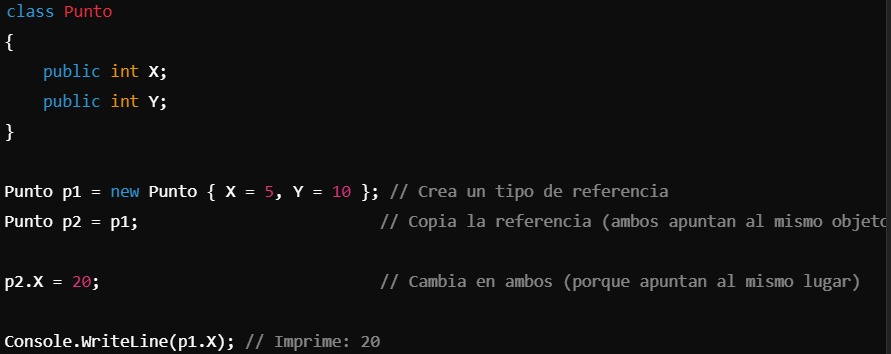
**Tipos de Valor (Stack)**

1. **Qué son:**
   * Los struct y tipos primitivos (como int, double, bool) son **tipos de valor**.
   * Esto significa que almacenan directamente los **datos**.
2. **Dónde se almacenan:**
   * Se guardan en el **stack** (pila de ejecución), que es una memoria muy rápida pero limitada en tamaño.
   * El **stack** gestiona la memoria de forma automática: cuando una variable sale de su ámbito, la memoria se libera.
3. **Cómo funcionan:**
   * Al copiar un tipo de valor, se copia **todo el valor**, creando una copia completamente independiente.
   * No hay relación entre las dos variables después de la copia.
4. **Ejemplo:**

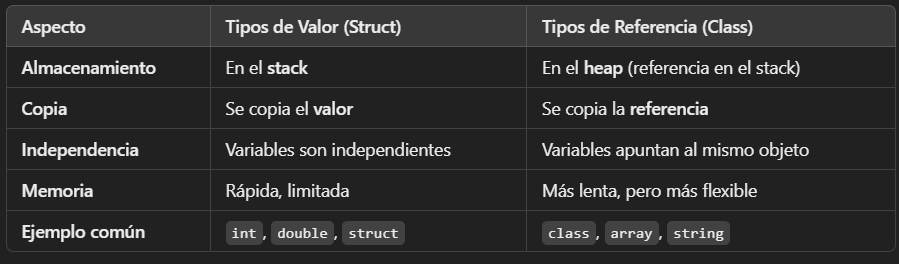


**Tipos de Referencia (Heap)**

1. **Qué son:**
   * Las class y otros objetos como string, arrays o listas son **tipos de referencia**.
   * Esto significa que almacenan una **referencia** (dirección de memoria) a los datos reales que están en el **heap**.
2. **Dónde se almacenan:**
   * La referencia (puntero) se almacena en el **stack**, pero los datos reales están en el **heap**.
   * El **heap** es más grande que el stack y se utiliza para almacenar objetos que pueden vivir más tiempo. Sin embargo, la gestión de memoria es más lenta porque requiere un **garbage collector**.
3. **Cómo funcionan:**
   * Al copiar un tipo de referencia, se copia solo la **referencia**, no los datos.
   * Ambos apuntan al mismo objeto en el heap, por lo que un cambio en una variable afecta a la otra.
4. **Ejemplo:**

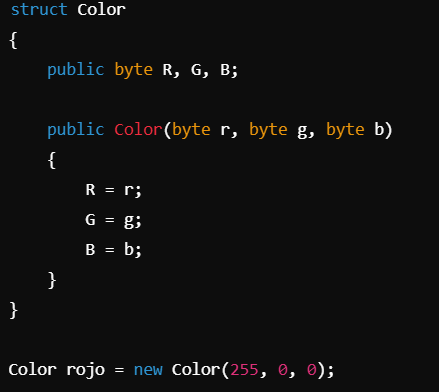


Diferencias Clave:



**Por qué usar struct:**

* Son útiles para datos pequeños e inmutables (como coordenadas, colores, etc.).
* Ejemplo:



* + Si los datos son más complejos o necesitan comportamientos avanzados (herencia, polimorfismo, etc.), deberías usar class.

