Struct y Clases

En un ordenador hay 2 tipos de memoria para almacenar información cuando estamos programando, uno de esos espacios se denomina 'stack';
cuando declaramos una variable, un tipo primitivo se almacena siempre en ella

La memoria stack es de acceso rápido mientras que la heap accede de forma más lenta.

Otra diferencia fundamental entre ambos tipos es que lo que almacenamos en 'stack' es temporal: variables dentro de un método, locales...
En cambio las variables globales y métodos se almacenan en el 'heap'

En cambio si nos creásemos una instancia de una clase
(objeto Coche por ejemplo), internamente lo que ocurre es diferente.
Cuando el compilador lee la instrucción:
'Coche coche 1 = new Coche();'

internamente se reserva un espacio en la memoria del ordenador,
pero en la 'heap', y se crea a su vez una referencia a ese objeto creado
en la 'stack'.

Es decir, se crea una especie de vínculo/referencia:
-Memoria stack: *coche1*
-Memoria heap: *Objeto Coche*

Esto es importante a ala hora de explicar/trabajar con los 'struct', ya que los struct se almacenan en la memoria 'stack'
como si fuese un tipo primitivo o una referencia

Las Clases se almacenan en la memoria 'heap-referencia'
y las struct en la 'stack-valor'.

Ver ejemplo ejercicio empleado

Cuando trabajamos con **estructuras**, los cambios se ejecutan en la copia propia y esos datos se mantendrán inalterables.

Está es la diferencia fundamental entre una clase y una estructura:
-Clases - memoria Heap
-struct - memoria Stack

Ejemplo Struct – 'Customer' :

```
public struct Customer
{
    public int ID;
    public string Name;

    public Customer(int customerID, string customerName)
    {
        ID = customerID;
        Name = customerName;
    }
}

class TestCustomer
{
    static void Main()
    {
        Customer c1 = new Customer();  //using the default constructor

        System.Console.WriteLine("Struct values before initialization:");
        System.Console.WriteLine("ID = {0}, Name = {1}", c1.ID, c1.Name);
        System.Console.WriteLine();

        c1.ID = 100;
        c1.Name = "Robert";

        System.Console.WriteLine("Struct values after initialization:");
        System.Console.WriteLine("ID = {0}, Name = {1}", c1.ID, c1.Name);
    }
}
```

Trabajar con estructuras es muy similar a trabajar con clases, salvo que internamente funcionan totalmente opuestas.
Y además las estructuras no permiten 'constructores vacíos'