A frequency array in C# refers to a data structure used to store the frequency or count of elements in a given collection, typically an array or any enumerable data source. This structure maintains a record of how many times each element occurs within the collection.

**//frequency array example:**

int[] numbers = { 1, 2, 2, 3, 3, 3, 4, 4, 4, 4 };

var frequencyArray = numbers.GroupBy(n => n).ToDictionary(g => g.Key, g => g.Count());

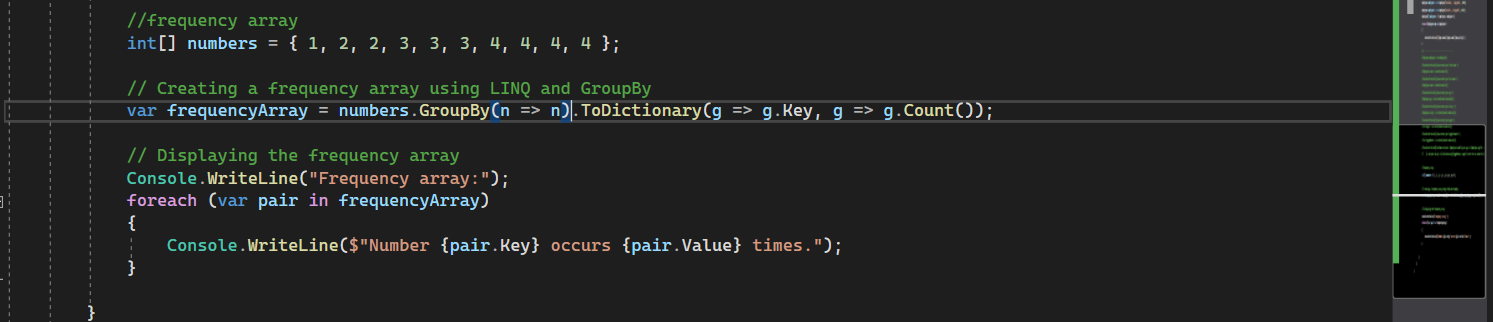
Console.WriteLine("Frequency array:");

foreach (var pair in frequencyArray)

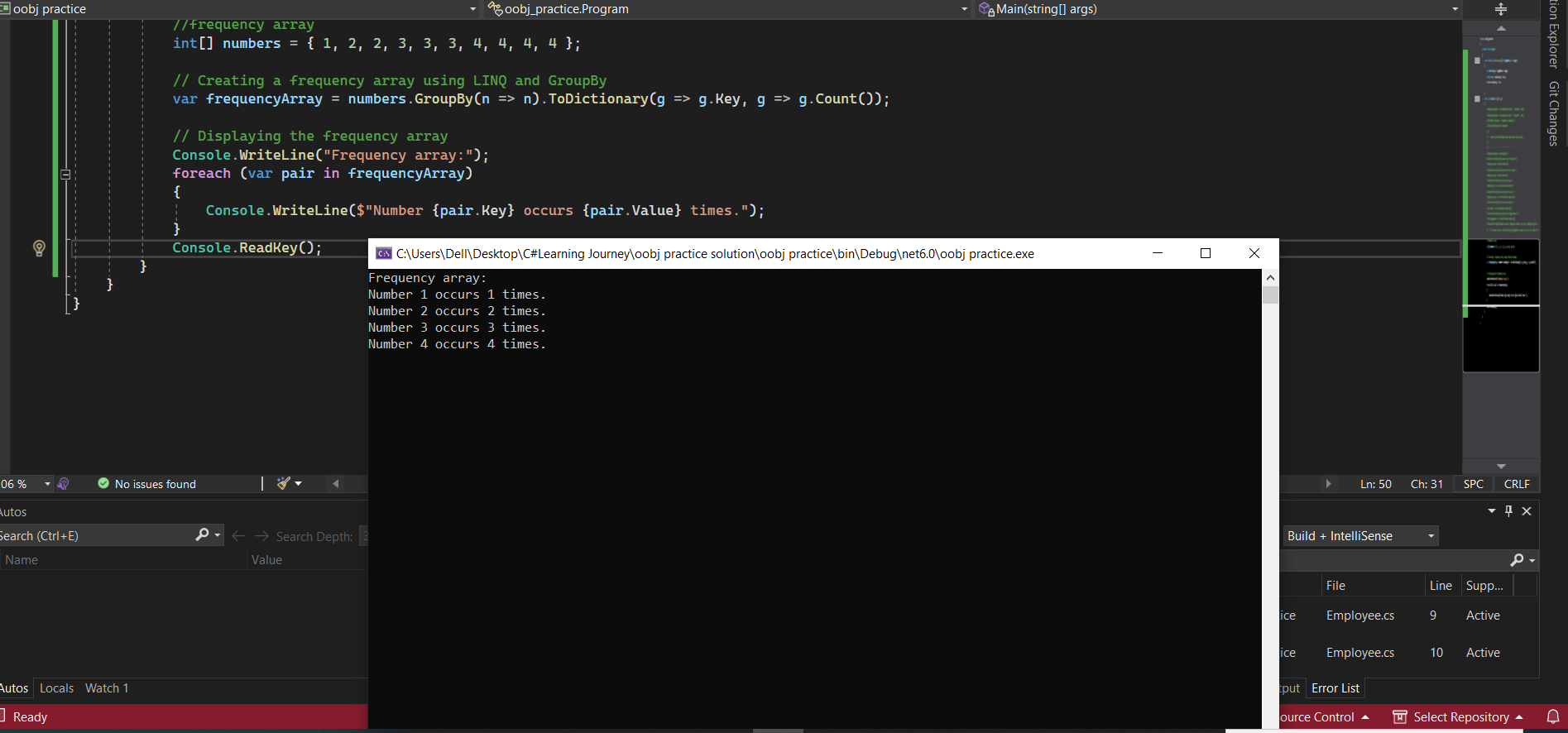
{

Console.WriteLine($"Number {pair.Key} occurs {pair.Value} times.");

}

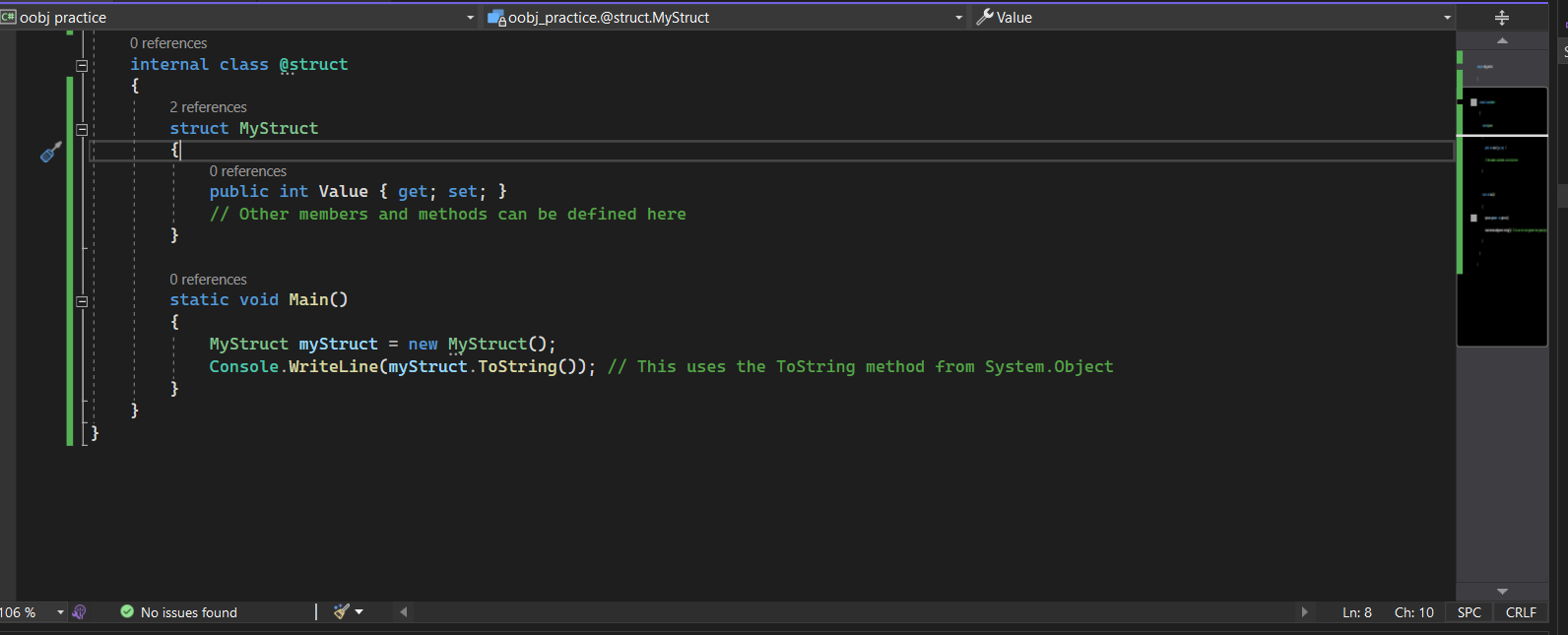


**Output:**



**how struct not support inheritance but struct inherit from object in C#:**

a struct is a **value type** that represents a lightweight structure, while **a class is a reference type** that supports features like inheritance. **Structs don’t support inheritance because they cannot inherit from other structs or classes**, nor can they serve as a base for inheritance. all types in C#, including structs, implicitly inherit from the System. Object class, which is the root of the .NET type hierarchy. This inheritance is known as implicit or direct inheritance from System. Object. This means that even though structs themselves cannot inherit from other structs or classes, they inherit some behavior and methods from the System. Object class. For example, structs inherit methods like **ToString()**, **Equals()**, **GetHashCode()**, and other members of System. Object. But this inheritance is limited to the methods defined in System.Object; it doesn't allow a struct to participate in inheritance hierarchies like classes do.



In this example, the **MyStruct** struct doesn't explicitly inherit from any other type, but it implicitly inherits the **ToString()** method (and other **System.Object methods**) due to its relationship with **System.Object**.