```csharp
using System;

namespace _01_DATA_TYPES
{
    class Program
    {
        // User-defined enum (Value Type)
        enum Color { Red, Blue, Green }

        // User-defined struct (Value Type)
        struct Point
        {
            public int X;
            public int Y;
        }

        // User-defined class (Reference Type)
        class Person
        {
            public string Name;
            public int Age;
        }

        static void Main(string[] args)
        {
            // -------------------------------------
            // Overview of Data Types in C#
            // -------------------------------------
            // C# data types are categorized into:
            // 1. Value Types - store data directly.
            // 2. Reference Types - store reference to data.

            #region 01. Value Type:
            Console.WriteLine("---------- VALUE TYPES ----------");

            // Integer (32-bit signed)
            int number = 26;
            Console.WriteLine($"Integer value: {number}");

            // Nullable Integer (can be null)
            int? age = null;
            Console.WriteLine($"Nullable int (age): {age}");

            // Byte (8-bit unsigned)
            byte smallNumber = 255;
            Console.WriteLine($"Byte value: {smallNumber}");

            // Long (64-bit signed)
            long largeNumber = 984983L;
            Console.WriteLine($"Long value: {largeNumber}");

            // Float (32-bit floating-point)
            float temperature = 5.5f;
            Console.WriteLine($"Float value: {temperature}");

            // Double (64-bit floating-point)
            double pi = 3.14159;
            Console.WriteLine($"Double value: {pi}");

            // Character
            char letter = 'A';
            Console.WriteLine($"Char value: {letter}");

            // Boolean
            bool isActive = true;
            Console.WriteLine($"Boolean value: {isActive}");

            // Enum
            Color favoriteColor = Color.Red;
            Console.WriteLine($"Enum value (Color): {favoriteColor}");

            // Struct
            Point p;
            p.X = 5;
            p.Y = 10;
            Console.WriteLine($"Struct value: X = {p.X}, Y = {p.Y}");
            #endregion

            #region 02. Reference Type:
            Console.WriteLine("\n---------- REFERENCE TYPES ----------");

            // String (immutable)
            string firstName = "Thillai";
```

```
            String lastName = "Tamil";
            Console.WriteLine($"String values: {firstName} {lastName}");

            // Object
            object genericData = 42;
            Console.WriteLine($"Object value: {genericData}");

            // Dynamic
            dynamic dynamicData = "I am dynamic";
            Console.WriteLine($"Dynamic value: {dynamicData}");

            // Array
            int[] numbersArray = { 1, 2, 3, 4 };
            Console.WriteLine("Array values: " + string.Join(", ", numbersArray));

            // Class
            Person person = new Person { Name = "Kumar", Age = 30 };
            Console.WriteLine($"Class (Person): Name = {person.Name}, Age = {person.Age}");

            #endregion

            #region 03. Extras:
            Console.WriteLine("\n---------- TYPE INFO & LIMITS ----------");

            Console.WriteLine($"Integer Min Value: {int.MinValue}");
            Console.WriteLine($"Long Max Value: {long.MaxValue}");
            Console.WriteLine($"Size of int: {sizeof(int)} bytes");
            Console.WriteLine($"Default value of string: {(default(string) == null ? "null" : default(string))}");

            Console.WriteLine("\n---------- PARSING & CONVERSION ----------");

            string input = "123";
            int parsedInt = int.Parse(input);
            Console.WriteLine($"Parsed integer: {parsedInt}");

            bool isValid = double.TryParse(input, out double parsedDouble);
            Console.WriteLine($"Double parsing successful: {isValid}, Result: {parsedDouble}");

            #endregion

            #region 04. Explanation for value and refernece type:
            Console.WriteLine("\n---------- VALUE vs REFERENCE TYPES ----------");

            // Value Type Example
            int a = 10;
            int b = a;
            b = 20;
            Console.WriteLine($"Value Type Example - a: {a}, b: {b}"); // a=10, b=20

            // Reference Type Example
            int[] refArr1 = { 1, 2, 3 };
            int[] refArr2 = refArr1;
            refArr2[0] = 99;
            Console.WriteLine($"Reference Type Example - refArr1[0]: {refArr1[0]}, refArr2[0]: {refArr2[0]}"); // both=99

            Console.WriteLine("\nValue types store data directly. Assigning them copies the value.");
            Console.WriteLine("Reference types store a memory reference. Assigning them shares the same object.");

            Console.WriteLine("\n---------- END OF PROGRAM ----------");
            #endregion
        }
    }
}
```