

01_DATA_TYPES\01_DATA_TYPES\Program.cs

```
1  using System;
2
3  namespace _01_DATA_TYPES
4  {
5      class Program
6      {
7          // User-defined enum (Value Type)
8          enum Color { Red, Blue, Green }
9
10         // User-defined struct (Value Type)
11         struct Point
12         {
13             public int X;
14             public int Y;
15         }
16
17         // User-defined class (Reference Type)
18         class Person
19         {
20             public string Name;
21             public int Age;
22         }
23
24         static void Main(string[] args)
25         {
26             // -----
27             // Overview of Data Types in C#
28             // -----
29             // C# data types are categorized into:
30             // 1. Value Types - store data directly.
31             // 2. Reference Types - store reference to data.
32
33             #region 01. Value Type:
34             Console.WriteLine("----- VALUE TYPES -----");
35
36             // Integer (32-bit signed)
37             int number = 26;
38             Console.WriteLine($"Integer value: {number}");
39
40             // Nullable Integer (can be null)
41             int? age = null;
42             Console.WriteLine($"Nullable int (age): {age}");
43
44             // Byte (8-bit unsigned)
45             byte smallNumber = 255;
46             Console.WriteLine($"Byte value: {smallNumber}");
47
48             // Long (64-bit signed)
49             long largeNumber = 984983L;
50             Console.WriteLine($"Long value: {largeNumber}");
51         }
52     }
53 }
```

```
52 // Float (32-bit floating-point)
53 float temperature = 5.5f;
54 Console.WriteLine($"Float value: {temperature}");
55
56 // Double (64-bit floating-point)
57 double pi = 3.14159;
58 Console.WriteLine($"Double value: {pi}");
59
60 // Character
61 char letter = 'A';
62 Console.WriteLine($"Char value: {letter}");
63
64 // Boolean
65 bool isActive = true;
66 Console.WriteLine($"Boolean value: {isActive}");
67
68 // Enum
69 Color favoriteColor = Color.Red;
70 Console.WriteLine($"Enum value (Color): {favoriteColor}");
71
72 // Struct
73 Point p;
74 p.X = 5;
75 p.Y = 10;
76 Console.WriteLine($"Struct value: X = {p.X}, Y = {p.Y}");
77 #endregion
78
79 #region 02. Reference Type:
80 Console.WriteLine("\n----- REFERENCE TYPES -----");
81
82 // String (immutable)
83 string firstName = "Thillai";
84 String lastName = "Tamil";
85 Console.WriteLine($"String values: {firstName} {lastName}");
86
87 // Object
88 object genericData = 42;
89 Console.WriteLine($"Object value: {genericData}");
90
91 // Dynamic
92 dynamic dynamicData = "I am dynamic";
93 Console.WriteLine($"Dynamic value: {dynamicData}");
94
95 // Array
96 int[] numbersArray = { 1, 2, 3, 4 };
97 Console.WriteLine("Array values: " + string.Join(", ", numbersArray));
98
99 // Class
100 Person person = new Person { Name = "Kumar", Age = 30 };
101 Console.WriteLine($"Class (Person): Name = {person.Name}, Age =
{person.Age}");
102
103 #endregion
104
```

```
105     #region 03. Extras:
106     Console.WriteLine("\n----- TYPE INFO & LIMITS -----");
107
108     Console.WriteLine($"Integer Min Value: {int.MinValue}");
109     Console.WriteLine($"Long Max Value: {long.MaxValue}");
110     Console.WriteLine($"Size of int: {sizeof(int)} bytes");
111     Console.WriteLine($"Default value of string: {(default(string) == null ?
"null" : default(string))}");
112
113     Console.WriteLine("\n----- PARSING & CONVERSION -----");
114
115     string input = "123";
116     int parsedInt = int.Parse(input);
117     Console.WriteLine($"Parsed integer: {parsedInt}");
118
119     bool isValid = double.TryParse(input, out double parsedDouble);
120     Console.WriteLine($"Double parsing successful: {isValid}, Result:
{parsedDouble}");
121
122     #endregion
123
124     #region 04. Explanation for value and reference type:
125     Console.WriteLine("\n----- VALUE vs REFERENCE TYPES -----");
126
127     // Value Type Example
128     int a = 10;
129     int b = a;
130     b = 20;
131     Console.WriteLine($"Value Type Example - a: {a}, b: {b}"); // a=10, b=20
132
133     // Reference Type Example
134     int[] refArr1 = { 1, 2, 3 };
135     int[] refArr2 = refArr1;
136     refArr2[0] = 99;
137     Console.WriteLine($"Reference Type Example - refArr1[0]: {refArr1[0]},
refArr2[0]: {refArr2[0]}"); // both=99
138
139     Console.WriteLine("\nValue types store data directly. Assigning them copies
the value.");
140     Console.WriteLine("Reference types store a memory reference. Assigning them
shares the same object.");
141
142     Console.WriteLine("\n----- END OF PROGRAM -----");
143     #endregion
144 }
145 }
146 }
147
```