Data Types in C#

In C#, data types are categorized into two main types: Value Types and Reference Types. Each serves a different purpose and is used based on the needs of the program. Here's an overview of the different data types in C#:

Value Types

Value types hold data directly. These include simple types, enum types, and struct types.

Simple Types

- Integral Types:

- sbyte (8-bit signed integer)
- byte (8-bit unsigned integer)
- short (16-bit signed integer)
- ushort (16-bit unsigned integer)
- int (32-bit signed integer)
- uint (32-bit unsigned integer)
- long (64-bit signed integer)
- ulong (64-bit unsigned integer)
- char (16-bit Unicode character)

- Floating Point Types:

- float (single-precision floating point)
- double (double-precision floating point)

- Decimal Type:

- decimal (128-bit precise decimal values with 28-29 significant digits)

- Boolean Type:

- bool (represents a boolean value, true or false)

Enum Types

enum: A distinct type with a set of named constants called the enumerator list.

Struct Types

struct: A value type that can encapsulate data and related functionality.

Reference Types

Reference types store references to their data (objects), which are stored on the heap. These include classes, arrays, delegates, and interfaces.

Class Types

class: A blueprint for creating objects, supporting inheritance, polymorphism, encapsulation, and more.

Array Types

array: A collection of items stored at contiguous memory locations.

Delegate Types

delegate: A type that represents references to methods with a specific parameter list and return type.

Interface Types

interface: A contract that defines a set of methods and properties that the implementing class must provide.

Nullable Types

Nullable: A special version of value types that can represent normal value or a null reference, e.g., int?.

Object Type

object: The ultimate base class for all data types in C#. Every type directly or indirectly derives from object.

String Type

string: Represents a sequence of Unicode characters. The string type is an alias for System.String.

Dynamic Type

dynamic: A type that bypasses compile-time type checking. It resolves type at run-time.

Example Usage

```
using System;

namespace DataTypeExample
{
```

```
  class Program
  {
    static void Main(string[] args)
    {
      // Value types
      int integer = 10;
      float floatingPoint = 3.14f;
      char character = 'A';
      bool boolean = true;

      // Reference types
      string text = "Hello, World!";
      int[] array = { 1, 2, 3, 4, 5 };

      // Nullable type
      int? nullableInt = null;

      // Dynamic type
      dynamic dynamicVar = 1;
      dynamicVar = "Now I'm a string";

      // Display the values
      Console.WriteLine("Integer: " + integer);
      Console.WriteLine("Floating Point: " + floatingPoint);
      Console.WriteLine("Character: " + character);
      Console.WriteLine("Boolean: " + boolean);
      Console.WriteLine("Text: " + text);
      Console.WriteLine("Array: " + string.Join(", ", array));
      Console.WriteLine("Nullable Int: " + nullableInt);
      Console.WriteLine("Dynamic Var: " + dynamicVar);
    }
  }
}
```