**C# Data Types - Complete Reference Guide**

**Table of Contents**

1. [Numeric Types](#numeric-types)
2. [String Types](#string-types)
3. [Array Types](#array-types)
4. [Custom Types](#custom-types)
5. [Nullable Types](#nullable-types)
6. [Dynamic Types](#dynamic-types)
7. [Best Practices](#best-practices)

**Numeric Types**

**Integer Types**

**System.Int32 (int)**

The most commonly used integer type in C#.

int userAge = 25;

int maxValue = int.MaxValue; *// 2,147,483,647*

int minValue = int.MinValue; *// -2,147,483,648*

*Common Operations:*

* Addition: int sum = userAge + 5;
* Multiplication: int product = userAge \* 2;
* Division: int division = userAge / 3;
* Modulus: int remainder = userAge % 3;

**System.Int64 (long)**

Used for very large numbers.

long populationCount = 8\_000\_000\_000;

long fileSize = 1024L \* 1024L \* 1024L;

**Smaller Integer Types**

byte colorComponent = 255; *// 0 to 255*

short temperature = -40; *// -32,768 to 32,767*

**Floating-Point Types**

**System.Double**

double pi = 3.14159265359;

double scientificNotation = 6.022e23;

**System.Decimal**

decimal accountBalance = 1234.56m;

decimal interestRate = 0.0725m;

**String Types**

**String Creation and Manipulation**

string firstName = "John";

string lastName = "Doe";

string fullName = $"{firstName} {lastName}";

**String Operations**

* Concatenation: string.Concat(str1, str2)
* Interpolation: $"Hello, {name}!"
* Verbatim strings: @"C:\Program Files\App"

**Array Types**

**Single-Dimensional Arrays**

int[] numbers = new int[5];

int[] initialized = { 1, 2, 3, 4, 5 };

**Multi-Dimensional Arrays**

int[,] matrix = new int[3, 3];

matrix[0, 0] = 1;

**Jagged Arrays**

int[][] jagged = new int[3][];

jagged[0] = new int[] { 1, 2, 3 };

**Custom Types**

**Classes**

public class Person

{

public string Name { get; set; }

public int Age { get; set; }

public Person(string name, int age)

{

Name = name;

Age = age;

}

}

**Structs**

public struct Point

{

public double X { get; set; }

public double Y { get; set; }

}

**Nullable Types**

int? nullableInt = null;

bool? nullableBool = null;

DateTime? nullableDate = null;

**Dynamic Types**

dynamic dynamicVar = 100;

dynamicVar = "Now I'm a string";

**Best Practices**

**Numeric Types**

1. Use int for most whole numbers
2. Use decimal for financial calculations
3. Use double for scientific calculations

**String Handling**

1. Use string interpolation for readability
2. Use StringBuilder for multiple concatenations
3. Use verbatim strings for file paths

**Arrays**

1. Check bounds before accessing elements
2. Consider using List<T> for dynamic collections
3. Use jagged arrays for better performance

**Custom Types**

1. Use classes for reference types
2. Use structs for small value types
3. Implement appropriate interfaces

**Nullable Types**

1. Always check HasValue before accessing Value
2. Use null coalescing operator (??) for defaults
3. Handle null cases explicitly

**Dynamic Types**

1. Use only when necessary
2. Handle potential runtime errors
3. Consider performance implications