

.NET FSD Bootcamp

10th July, 2025



Title: C# Programming
Sub-Title : Data Types and Variables
Presented by: Narasimha Rao T

C# Programming : Data Types & Variables

By

Narasimha Rao T

Microsoft.Net FSD Trainer

Professional Development Trainer

Day-2 Index

1. Value vs reference types
2. Primitive types (int, float, char, bool, string)
3. var and type inference
4. Constants (const, readonly)
5. Type conversion: implicit, explicit, Convert, Parse, TryParse
6. Nullable types, default values, boxing/unboxing

1. Value vs Reference Types

Value Types

- Stored on the stack
- Hold actual data
- Examples: `int` , `float` , `bool` , `char` , `struct`

Reference Types

- Stored on the heap
- Hold reference (address) to the actual object
- Examples: `string` , arrays, `class` , `object` , delegates

Key Differences

Feature	Value Type	Reference Type
Memory	Stack	Heap
Copied on assign	Yes (by value)	No (reference copied)
Nullable	Use <code>?</code>	Already nullable

Example:

```
int a = 5;
int b = a; // value copied
b = 10;
// a is still 5

string s1 = "hello";
string s2 = s1; // reference copied
s2 = "world";
// s1 may still point to "hello", but strings are immutable
```

2. Primitive Types

Integral & Floating Point:

```
int age = 25;  
float temp = 36.6f;  
double pi = 3.14159;  
long population = 7000000000;
```

Character & Boolean:

```
char grade = 'A';  
bool isLoggedIn = true;
```


String:

```
string result = String.Empty;  
string name = "Narasimha";  
string email = "tnrao.trainer@gmail.com";  
string city = "Hyderabad";
```

3. `var` and Type Inference

- Let the **compiler infer the type** based on the value assigned.

```
var message = "Hello";    // string
var score = 89;           // int
```

Must be initialized immediately

Useful with LINQ or anonymous types

4. Constants

const (Compile-time constant)

```
const double Pi = 3.14;
```

readonly (Run-time constant, used with fields)

```
readonly DateTime createdAt = DateTime.Now;
```

Modifier	Settable?	When
const	No	Compile-time
readonly	Only in ctor	Runtime

5. Type Conversion

Implicit Conversion

```
int a = 100;  
long b = a; // safe
```

Explicit Conversion (Casting)

```
double pi = 3.14;  
int approx = (int)pi; // 3
```

Convert Class

```
string numStr = "123";  
int num = Convert.ToInt32(numStr);
```

Parse & TryParse

```
int n = int.Parse("456");  
bool isValid = int.TryParse("abc", out int result); // false
```

`Parse` throws exception if format is invalid; `TryParse` is safer.

6. Nullable Types, Default Values, Boxing/Unboxing

Nullable Types

```
int? x = null;  
if (x.HasValue) Console.WriteLine(x.Value);
```

Default Values

```
int i = default;      // 0  
bool b = default;     // false  
string s = default;   // null
```

Boxing / Unboxing

```
object box = 42;           // boxing
int unbox = (int)box;      // unboxing
```

7. CodeAlong: "Quiz Score Tracker"

Goal:

- Input name and score
- Display summary

Code:

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Enter student name: ");
        string name = Console.ReadLine();

        Console.Write("Enter quiz score: ");
        bool valid = int.TryParse(Console.ReadLine(), out int score);

        if (valid)
        {
            Console.WriteLine($"{name}'s score: {score}");
        }
        else
        {
            Console.WriteLine("Invalid score input.");
        }
    }
}
```

8. Breakout Task: Guess the Output (Types & Conversions)

Activity Instructions:

- Show 4–5 code snippets
- Ask students to **predict output**
- Example:

```
double d = 5.7;  
int x = (int)d;  
Console.WriteLine(x);    // ?
```

Answer: 5

```
var val = "123";  
int num = Convert.ToInt32(val);  
Console.WriteLine(num); // ?
```

9. Whiteboard: Variable Declaration Rules Showdown

Rules to Debate & Write:

- Can variables start with digits?
- Can you use `var` without assignment?
- Are C# variable names case-sensitive?
- Difference between `int x = 5;` and `var x = 5;`

Example Rules:

- No keywords as variable names (`int int = 5;`)
- Variables are case-sensitive (`total` \neq `Total`)
- `var` must be assigned at declaration
- `const` must be initialized

10. Group Error-Hunt: Type Bugs

Buggy Code Example:

```
string s = null;  
int i = (int)s; //
```

Fix:

```
int i = Convert.ToInt32(s); // throws exception if s is null
```

Summary

- Value vs reference types
- Primitive data types
- Type inference with `var`
- Constant declarations
- Type conversions and parsing
- Nullable types and boxing
- Hands-on practice and debugging

Q & A