Variables & Data Types in C#

Week 2: Programming Fundamentals

Press Space for next slide →

What Are Variables?

- **Named containers** that store data in memory
- Each variable has a:
  - Name (identifier)
  - Type (data kind)
  - Value (actual data)
- Variables allow us to:
  - Store user input
  - Track program state
  - Perform calculations
  - Manipulate data

Variable Declaration Syntax

```
dataType variableName = initialValue;

// Examples
int age = 25;
string name = "John";
double price = 19.99;
bool isAvailable = true;
```

Multiple Ways to Declare

```
// Without initialization
int count;
count = 10;

// Multiple variables
int x = 5, y = 10, z = 15;

// Constants
const double PI = 3.14159;
```

Memory Visualization

age	**25**
name	**"John"**
price	**19.99**
isAvailable	**true**

Integer Types

- Used for whole numbers
- Different sizes for different ranges
- Most common: `int`

```
// Common integer types
byte smallNumber = 255;
short mediumNumber = 32767;
int regularNumber = 2147483647;
long bigNumber = 9223372036854775807L;
```

```
// Integer literals
int decimal = 42;         // Decimal (base 10)
int hex = 0x2A;           // Hexadecimal (base 16)
int binary = 0b101010;    // Binary (base 2)
```

Integer Types Overview

Type	Size	Range
`byte`	1 byte	0 to 255
`short`	2 bytes	-32K to 32K
`int`	4 bytes	-2.1B to 2.1B
`long`	8 bytes	-9.2E+18 to 9.2E+18

Don't forget the `L` suffix for `long` literals!

Floating-Point Types

- Used for decimal numbers
- Different precision levels
- Common types: `float`, `double`, `decimal`

```
// Float - less precision, requires 'f' suffix
float temperature = 98.6f;

// Double - default for decimal literals
double pi = 3.14159265359;

// Decimal - high precision for financial calculations
decimal accountBalance = 1250.75m;
```

Floating-Point Comparison

Type	Size	Precision	Suffix
`float`	4 bytes	~7 digits	`f`
`double`	8 bytes	~15-16 digits	none
`decimal`	16 bytes	28-29 digits	`m`

🏛 Use `decimal` for money calculations!

Other Common Types

Boolean Type

- Stores `true` or `false` values
- Used for logical conditions

```
bool isStudent = true;
bool hasCompletedCourse = false;

// Commonly used in conditions
if (isStudent && !hasCompletedCourse) {
    Console.WriteLine("Keep studying!");
}
```

Character Type

- Stores a single Unicode character
- Uses single quotes

```
char grade = 'A';
char symbol = '#';
char unicodeChar = '\u00A9'; // © copyright symbol
```

String Type

- Stores text (sequence of characters)
- Uses double quotes
- Reference type (not a primitive)

Type Conversion

Implicit Conversion (Safe)

- Smaller type to larger type
- No data loss occurs

```
int num = 100;
double biggerNum = num;  // int → double
```

Explicit Conversion (Cast)

- Larger type to smaller type
- Potential data loss
- Requires cast operator

```
double price = 19.99;
int dollars = (int)price;  // 19, loses .99
```

Using Convert Class

```
// String to number
string input = "42";
int number = Convert.ToInt32(input);
double amount = Convert.ToDouble("123.45");

// Between types
int intValue = Convert.ToInt32(3.14159);  // 3
bool boolValue = Convert.ToBoolean(1);    // true
```

Parse Methods

```
int age = int.Parse("25");
double weight = double.Parse("68.5");
```

TryParse (Safer Approach)

```
string input = "abc";
int result;
bool success = int.TryParse(input, out result);
// success = false, result = 0
```

Variable Naming Rules & Conventions

Rules

- Must start with letter or underscore
- Can contain letters, digits, and underscores
- Cannot use C# keywords (unless prefixed with @)
- Case-sensitive

Invalid Names

```
int 1stPlace = 1;      // Can't start with number
int student-count = 0; // Can't use hyphen
int class = 101;        // Can't use keyword
```

Conventions

- **camelCase** for local variables

```
int studentAge = 21;
string fullName = "John Smith";
```

- **PascalCase** for classes, methods

```
class StudentRecord { }
void CalculateGrade() { }
```

- Descriptive names

```
// Good
int numberOfStudents = 25;

// Bad
int n = 25;
```

Best Practices

- Initialize variables when declaring them

```
int count = 0; // Good
int count;     // Less ideal
```

- Use appropriate types for your data

```
decimal price = 19.99m; // Good for money
double price = 19.99;   // Less ideal for money
```

- Use constants for fixed values

```
const double PI = 3.14159;
const int MAX_STUDENTS = 30;
```

- Keep variable scope as small as possible
- Use meaningful names that explain purpose

Common Pitfalls

Using Uninitialized Variables

```
int x;
Console.WriteLine(x); // Compile error
```

Integer Division Truncation

```
int a = 5, b = 2;
double result = a / b;  // 2.0, not 2.5!

// Fixed version:
double result = (double)a / b;  // 2.5
```

Forgetting Type Suffixes

```
float x = 1.5;     // Error: 1.5 is double
float y = 1.5f;    // Correct: 1.5f is float

decimal price = 10.99;     // Error
decimal price = 10.99m;    // Correct
```

Type Conversion Errors

```
string input = "abc";
int number = int.Parse(input);  // Throws exception!

// Better approach:
if (int.TryParse(input, out int number)) {
    // Use number safely
} else {
    // Handle invalid input
}
```

Floating-Point Comparison

```
// Potentially problematic due to precision
if (balance == 100.0) { ... }

// Better approach:
if (Math.Abs(balance - 100.0) < 0.0001) { ... }
```

Let’s See It In Action

Live Coding Demo

Student Coding Task

Create a program that represents a student profile:

1. Declare variables for:
   - Student name (string)
   - Age (integer)
   - GPA (floating-point)
   - Enrollment status (boolean)
   - Student ID (string)
2. Initialize with sample data
3. Display formatted output
4. Convert GPA to int and back

Expected Output

```
Student Profile:
Name: Maria Garcia
Age: 22
GPA: 3.85
Full-time student: True
Student ID: S12345

GPA as integer: 3
GPA back to decimal: 3.00
```

Debugging Common Issues

Wrong Data Types

```
// Error: Type mismatch
int studentAge = "twenty";
```

Missing Type Suffixes

```
// Error: Can't implicitly convert
float score = 95.5;   // Missing 'f'
```

Naming Errors

```
// Error: Illegal space
int high score = 100;
```

Type Conversion Failures

```
// Runtime Error:
string input = "abc";
int value = Convert.ToInt32(input);
```

Uninitialized Variables

```
// Compile Error:
decimal price;
Console.WriteLine(price);
```

Boolean Conversion Errors

```
// Error: Can't directly convert
bool passed = "yes";
```

Questions?

Next Week: Operators & Expressions

END