Minutes of Session: Day 2 – 19th July

Trainer: Parth Shukla

Participants: Wipro NGA

Session Time: 10:00 AM – 7:00 PM

Topics Covered

1. C# Fundamentals and Language Syntax

- **Variables**
    - Declaration, initialization, scope, and naming conventions.
- **Operators**
    - Arithmetic: `+`, `-`, `*`, `/`, `%`
    - Relational: `==`, `!=`, `<`, `>`, `<=`, `>=`
    - Logical: `&&`, `||`, `!`
    - Assignment: `=`, `+=`, `-=`, etc.

Note: What all different version of c# are there, latest version how major functionality is introduced in each versions:

C# 1.0	2002	Basic OOP(classes, interface, inheritance) Properties, indexers, delegates, events
C# 2.0	2005	Generics(collection), Anonymous types, Nullable types, Partial classes
C# 3.0	2007	LINQ, Lambda expressions, Extension methods,Automatics properties, Object and Collection Initializers
C# 4.0	2010	Dynamic typing, names and optional arguments, COM interoperability

C# 5.0	2012	Async and await, caller info attributes
C# 6.0	2015	Interpolated strings($"..."), null-conditional operator(? .), Auto-property initializes
C# 7.0 - 7.3	2017 - 2018	Tuples, pattern matching, Local functions, ref local and return, Out variables, Digit separators
C# 8.0 (NET Core)	2019	Async Streams, ranges and indices(x..y), Switch expressions, Default interface methods, nullable reference types
C# 9.0 (.NET 5)	2020	Records, Init-only setters, top level programs, target-types new expressions, improved pattern matching
C# 10.0	2021	Global using directives , File scoped namespaces, record struct, constant interpolated strings
C # 11.0	2022	Raw strings literals , list patterns, required members, Generics attributes and UTF-8 string literals
C# 12.0	2023	Collection expressions, primary constructors for non-records, Alias any type, interceptors

.NET Platform release

.NET framework 1.0	2002	BCL, ASP.NET, ADO.NET, CLR, Winform
.NET framework 2.0	2005	
	2006	
…	…	…
.NET framework 4.7/4.8	2017-2019	Win 10 compatibility

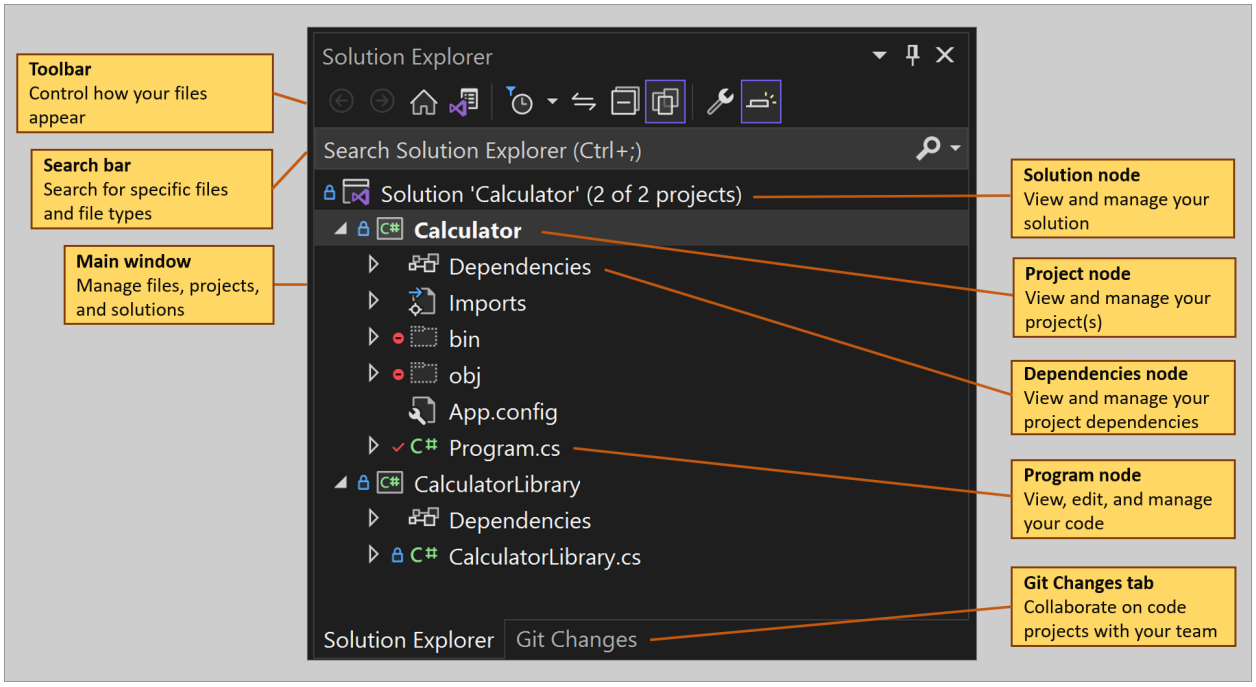
Transition to .NET Core and Modern .NET

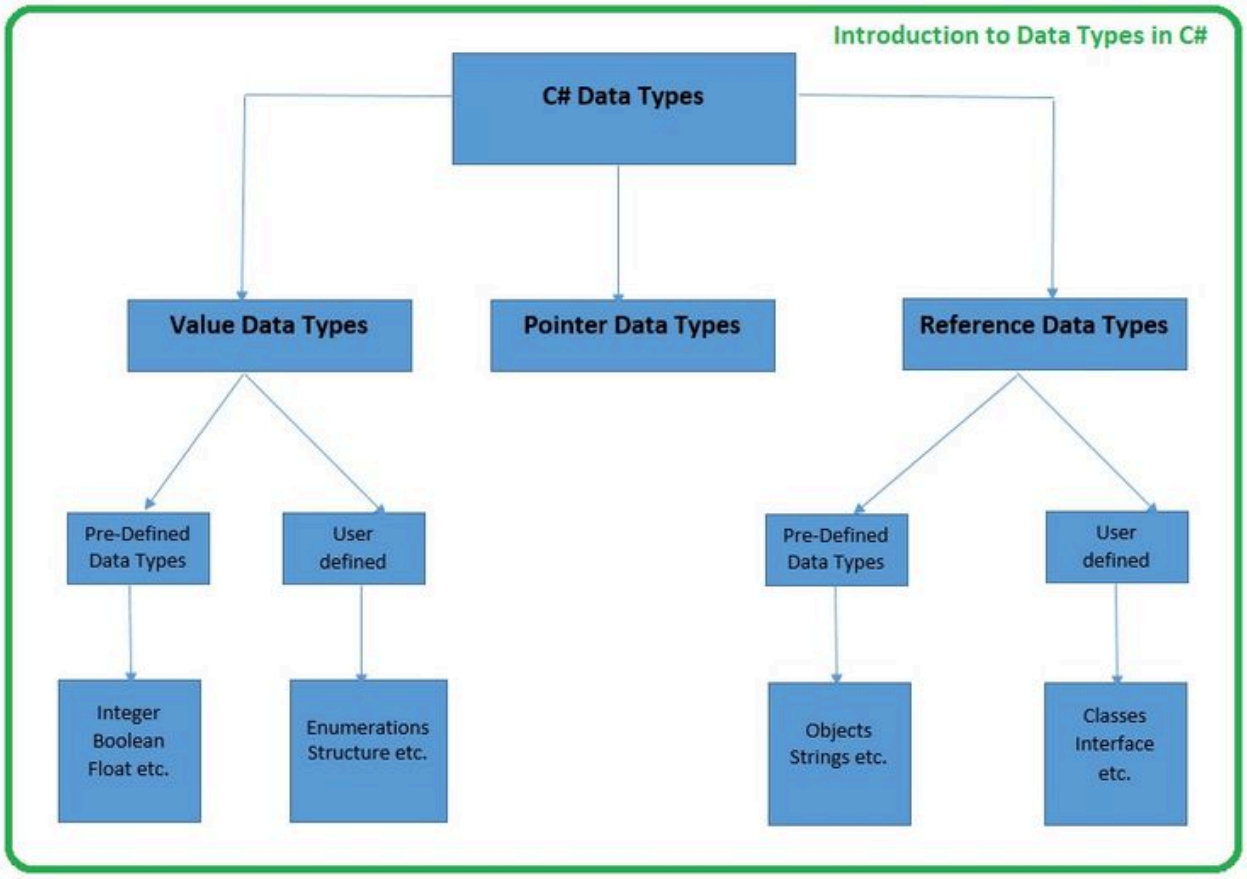
.NET Core 1.0	2016	Cross platform support, Modular CLR, CLI tools, ASP.NET Core
.NET Core 2.0	2017	.NET standard 2.0 support, Razor pages
.NET Core 3.0	2019	C# 8.0 features gPRC support

		Blazor introduction

Modern .NET(Unified platform)

.NET 5	2020	Unified platform (No more .NET Core branding) Single SDK and base class library C# 9.0
.NET 6	2021	Minimal API Hot reloads MAUI preview C# 10 was introduced
.NET 7	2022	.NET MAUI(official release) Native AOT compilation Enhance native -Cloud Support C# 11.0
NET 8	2023	Improved container support Blazor Full stack Interceptor preview(C# 12.0) Native AOT compilation
.NET 9	2024	.NET aspires for microservices Enhanced WebAssembly(WASM) Performance focused features





- **Type Conversion**

- **Implicit Conversion**: Done automatically when no data loss.
- **Explicit Conversion**: Requires cast or conversion method.
  Examples: `Convert.ToInt32()`, `int.Parse()`, `.ToString()`

2. Control Structures

- **Conditional Statements**
  - `if`, `else if`, `else`
  - `switch-case` for menu-driven or value-based branching
- **Loops**
  - `for`, `while`, `do-while`
  - Use case examples and visual representation through flowcharts

3. Arrays in C#

Definition:
An array is a data structure in C# that holds a fixed number of elements of the same type in a contiguous memory location.

Types of Arrays:

Type	Definition	Example
Single-Dimensional	Linear array with one index	`int[] nums = new int[5];`
Multi-Dimensional	Matrix-like array	`int[,] matrix = new int[2,3];`

Jagged Array	Array of arrays (each row can have different length)	`int[][] jagged = new int[3][];`

Comparison Table:

Feature	Single-Dimensional	Multi-Dimensional	Jagged Array
Syntax	`int[] arr = new int[5];`	`int[,] arr = new int[2,3];`	`int[][] arr = new int[3][];`
Memory Layout	Linear	Rectangular	Irregular
Accessing Element	`arr[i]`	`arr[i,j]`	`arr[i][j]`
Flexibility in Row Size	Not Applicable	All rows must be equal	Rows can be of different sizes
Use Case	Simple lists	Tables or Matrices	Sparse or unequal rows

---

Hands-on Activities

- Variable declaration and value manipulation examples
- Logical operator-based examples such as eligibility calculators

- Flowchart-based control flow visualization
- Single-dimensional and jagged array implementations
- **Mini Project**: *Matrix Calculator*
    - Menu-driven calculator for matrix operations: addition, subtraction, multiplication using arrays and loops

---

Best Practices Discussed

- Always initialize variables to avoid runtime exceptions
- Use `var` judiciously; prefer explicit types in public APIs
- Use `switch-case` when multiple conditions depend on a single value
- Avoid infinite loops by controlling loop conditions properly
- Use `foreach` loop for better readability in array iteration
- Avoid hardcoding array sizes unless fixed; use constants or dynamic collection types like `List<T>` when needed

---

Interview and Viva Questions

Variables and Type Conversion

1. What is the difference between value type and reference type in C#?
2. How is type casting handled in C#? Give examples.
3. What is boxing and unboxing in C#?

Control Structures

4. Explain the difference between `if`, `else if`, and `switch`.
5. When would you use a `do-while` loop instead of `while`?
6. What is the purpose of `break`, `continue`, and `goto` in loops?

Arrays

7. What is the difference between a jagged array and a multi-dimensional array?
8. How do you initialize a jagged array?
9. What are the advantages of using a jagged array over a 2D array?
10. What happens if you try to access an index outside the bounds of an array?
11. Explain the default values in C# arrays.

General

12. What is the role of the `var` keyword? When should it be avoided?
13. How does C# ensure type safety in variable declarations?
14. What is the scope of a variable declared inside a loop or condition?

Key Takeaways

- Solid understanding of C# language fundamentals and syntax
- Ability to use control structures efficiently
- Clear understanding of arrays and their different types in C#
- Hands-on experience with logical problem solving using code
- Preparedness for basic technical interviews related to C#