A **delegate** is a type that holds a reference to **methods with a specific signature**. Think of it like a pointer to a method, or a variable that can point to different methods at runtime.

using System;

public class Program

{

// Step 1: Declare a delegate

public delegate void GreetDelegate(string name);

// Step 2: Create methods to match the delegate signature

public static void SayHello(string name)

{

Console.WriteLine("Hello, " + name);

}

public static void SayGoodbye(string name)

{

Console.WriteLine("Goodbye, " + name);

}

public static void Main()

{

// Step 3: Create delegate instance and point to methods

GreetDelegate greet = SayHello;

greet("Aditi");

// Reassigning to another method

greet = SayGoodbye;

greet("Aditi");

}

}

**Multicast Delegates**

You can point to **multiple methods** using +=:

GreetDelegate greetAll = SayHello;

greetAll += SayGoodbye;

greetAll("Team");

**Anonymous Methods and Lambda with Delegates**

GreetDelegate greet = delegate(string name)

{

Console.WriteLine("Hi, " + name);

};

greet("Anjali");

// With lambda

GreetDelegate greet2 = (name) => Console.WriteLine("Hey, " + name);

greet2("Swapnil");

**Built-in Delegates**

C# provides **generic delegates**:

| **Type** | **Description** |
| --- | --- |
| Action | Points to a method that returns void |
| Func | Points to a method that returns a value |
| Predicate | Returns a bool (typically used for conditions) |

Action<string> print = (msg) => Console.WriteLine(msg);

Func<int, int, int> add = (a, b) => a + b;

Predicate<int> isEven = x => x % 2 == 0;

print("Welcome!");

Console.WriteLine(add(3, 4)); // 7

Console.WriteLine(isEven(10)); // True

## When to Use Delegates?

* Callback methods (e.g., button click in WinForms)
* Events & event handlers
* LINQ, sorting, filtering
* Decoupling logic