Making Decisions in C#

By
Narasimha Rao T
Microsoft.Net FSD Trainer
Professional Development Trainer
tnrao.trainer@gmail.com

Introduction to Control Structures

Control structures determine the flow of execution in a program. They allow your program to make **decisions**, **repeat tasks**, and **handle different conditions dynamically**.

Types of control structures in C#:

- **Conditional Statements**: if, else if, else, switch
- **Looping Constructs**: for, while, do-while, foreach
- **Jump Statements**: break, continue, return

Note: These tools are essential to building logic in real-world programs — from handling user input to processing data conditionally.

if, else if, else

Syntax:

```
if (condition) {
    // code block if condition is true
} else if (anotherCondition) {
    // code block if anotherCondition is true
} else {
    // code block if none of the above conditions are true
}
```

Example:

```
int num = 10;

if (num > 0) {
    Console.WriteLine("Positive");
} else if (num == 0) {
    Console.WriteLine("Zero");
} else {
    Console.WriteLine("Negative");
}
```

switch-case

Syntax:

```
switch (variable) {
    case value1:
        // code
        break;
    case value2:
        // code
        break;
    default:
        // code
        break;
}
```

Example:

```
int day = 3;

switch (day) {
    case 1:
        Console.WriteLine("Monday");
        break;
    case 2:
        Console.WriteLine("Tuesday");
        break;
    case 3:
        Console.WriteLine("Wednesday");
        break;
    default:
        Console.WriteLine("Another day");
        break;
}
```

Introduction to Loops

- Loops allow your program to repeat a block of code multiple times based on a condition.
- They are a core part of automating repetitive tasks in C#.
- Types of loops in C#:
    - **for** – when the number of iterations is known
    - **while** – when looping is based on a condition
    - **do-while** – similar to while, but guarantees at least one execution
    - **foreach** – best for iterating over collections or arrays

- **Loops help minimize code duplication and handle dynamic input sizes.**

Loops

for Loop

```csharp
for (int i = 0; i < 5; i++) {
    Console.WriteLine(i);
}
```

while Loop

```
int i = 0;
while (i < 5) {
    Console.WriteLine(i);
    i++;
}
```

do-while Loop

Executes at least once.

```
int i = 0;
do {
    Console.WriteLine(i);
    i++;
} while (i < 5);
```

foreach Loop

Used for collections.

```
int[] numbers = { 1, 2, 3 };

foreach (int n in numbers) {
    Console.WriteLine(n);
}
```

break and continue

`break` : Exits the loop

`continue` : Skips to the next iteration

```
for (int i = 0; i < 10; i++) {
    if (i == 5) break;
    if (i % 2 == 0) continue;
    Console.WriteLine(i); // prints odd numbers until 5
}
```

Q & A

Narasimha Rao T
tnrao.trainer@gmail.com