Switch Expressions and Pattern Matching in C#

Switch expressions in C# are a **concise and readable** way to handle multiple conditions. They work seamlessly with **pattern matching**, allowing us to write expressive and maintainable code.

---

1. Constant Pattern Matching

Matching specific values, similar to traditional `case` statements:

```
string GetCategory(int number) => number switch
{
    1 => "One",
    2 => "Two",
    3 => "Three",
    _ => "Other" // Default case
};

Console.WriteLine(GetCategory(2)); // Output: Two
```

---

2. Relational Pattern Matching (>=, <=)

Using **ranges** instead of listing out every possible value:

```
string GetGrade(int score) => score switch
{
    >= 90 => "A",
    >= 80 => "B",
    >= 70 => "C",
    >= 60 => "D",
    _ => "F"
};

Console.WriteLine(GetGrade(85)); // Output: B
```

---

3. Type Pattern Matching

Handling different **types dynamically**:

```
string DescribeObject(object obj) => obj switch
{
    int n => $"It's an integer: {n}",
    string s => $"It's a string: {s}",
    bool b => $"It's a boolean: {b}",
    _ => "Unknown type"
};

Console.WriteLine(DescribeObject(42));        // Output: It's an
integer: 42
Console.WriteLine(DescribeObject("Hello"));   // Output: It's a
string: Hello
Console.WriteLine(DescribeObject(true));      // Output: It's a
boolean: True
```

---

4. Positional Pattern Matching (with Tuples)

Matching multiple values at once using **tuples**:

```
string WeatherAdvice(string weather, bool isWeekend) => (weather,
isWeekend) switch
{
    ("Sunny", true) => "Go to the beach!",
    ("Sunny", false) => "Enjoy a walk after work.",
    ("Rainy", _) => "Stay inside and read a book.",
    _ => "Just another day."
};

Console.WriteLine(WeatherAdvice("Sunny", true)); // Output: Go to the
beach!
```

---

5. Property Pattern Matching (Matching Object Properties)

Extracting and matching **specific properties** inside an object:

```
class Person
{
    public string Name { get; set; }
    public int Age { get; set; }
}

string GetDiscount(Person person) => person switch
{
    { Age: < 12 } => "Child discount",
    { Age: >= 65 } => "Senior discount",
    _ => "Regular price"
};

Console.WriteLine(GetDiscount(new Person { Name = "Alice", Age = 10
})); // Output: Child discount
Console.WriteLine(GetDiscount(new Person { Name = "Bob", Age = 70 }));
// Output: Senior discount
```

Summary

✅ **Constant patterns** – Match exact values.
✅ **Relational patterns** – Use comparisons like `>=`, `<=`.
✅ **Type patterns** – Match and handle different types.
✅ **Tuple patterns** – Match multiple values at once.
✅ **Property patterns** – Match object properties.

Switch expressions + pattern matching = **cleaner, more readable, and powerful C# code!** 🚀