THRIVE CONFERENCE

Christian Nagel

https://csharp.christiannagel.com

Pattern Matching with C#

THRIVECONF.COM

#ThriveITConf

3rd - 4th June 2025

Xnet your reliable IT partner

THRIVE CONFERENCE

Zahvaljujemo se našim sponzorjem

Syskit

Pearson VUE

LightningTools

span

Leading Learning Partners Association

Harmonia.si

FALA

SAVA ZAVAROVALNICA

LEMONISH barcaffè ARGETA

Lisjak

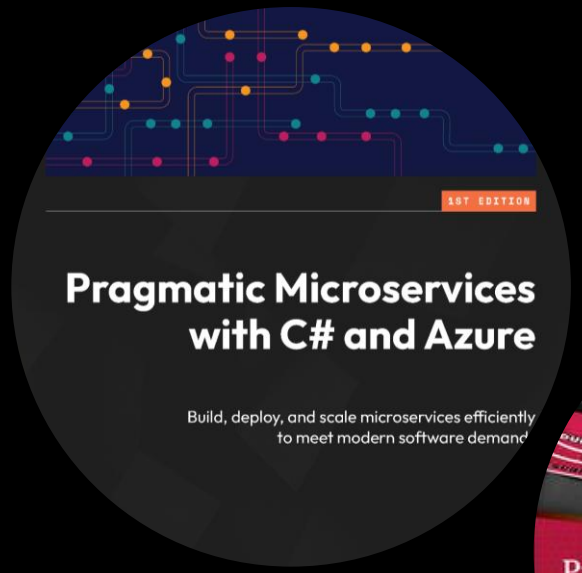
BIO POSESTVO Montis LAURA

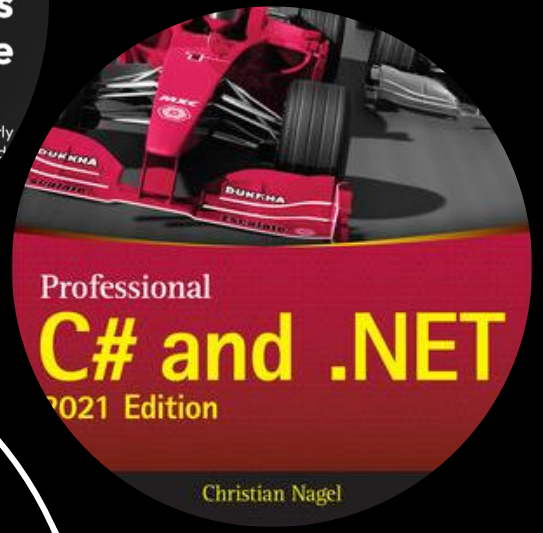
EC-Council Building A Culture Of Security

CISO MAG

Računalniške novice www.racunalniske-novice.com

IVARNOST.SI INFORMACIJSKA VARNOST





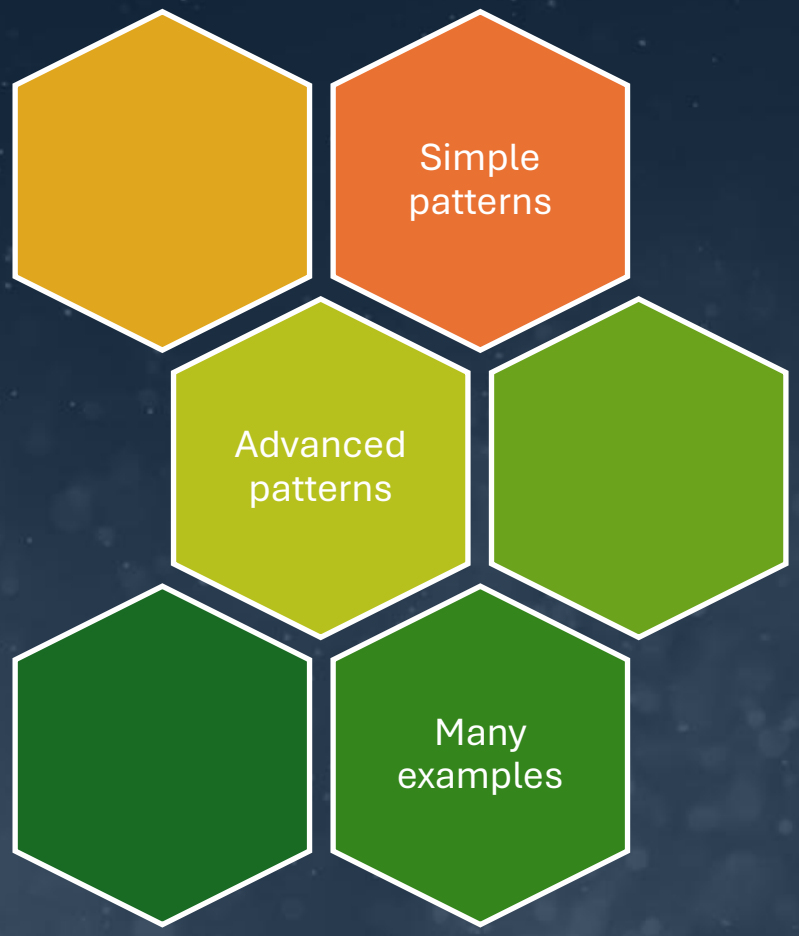
About me...

- Training
- Coaching
- Consulting
- Development

- Microsoft MVP
- www.cninnovation.com
- blogs.cninnovation.com

- Connect via LinkedIn / Bluesky

Topics



C# is not just an object-oriented programming language

Component

Declarative

Functional

Why pattern matching?

Readable code

Simplified control flow

Reduced type checks

Code sample – C# patterns

- Type and declaration
- Constant
- Relational
- Var
- Property
- Positional
- Discard

Type and declaration pattern

```
if (value is int intValue1)
    Console.WriteLine($"1. Type pattern: {intValue1} is an int");
```

Type pattern with generic types

```
object genericObj = new List<int>[1, 2, 3];
if (genericObj is List<int> intList)
    Console.WriteLine($"2. Generic type pattern: List<int> with count {intList.Count}");
else if (genericObj is List<string> stringList)
    Console.WriteLine($"2. Generic type pattern: List<string> with count {stringList.Count}");
```

Constant pattern

```
if (value is 42)
    Console.WriteLine("3. Constant pattern: Value is 42");
```

Relational pattern

```
if (value is int intValue2 && intValue2 > 40)
    Console.WriteLine("4. Relational pattern: Value is greater than 40");
```

Var pattern

```
if (value is var v)
    Console.WriteLine($"5. var pattern: {v}");
```

Property pattern

```
if (person is { FirstName: "Clark"})
    Console.WriteLine($"6. Property pattern: {person} is a Clark");
if (person is { Address: { City: "Smallville"} })
    Console.WriteLine($"6. Recursive property pattern: {person} lives in Smallville");
if (person is { Address.City: "Gotham City" })
    Console.WriteLine($"6. Dot-separated recursive property pattern: {person} lives in Gotham City");
```

Positional pattern

```
if (point is (3, 4))
    Console.WriteLine("7. Positional pattern: Point is (3, 4)");
```

Discard pattern

```
var tuple = (42, "hello", true);
if (tuple is (42, _, _))
    Console.WriteLine("8. Discard pattern: First item is 42, others are ignored");
```

Sample application

Traffic light

List patterns

```
int[] numbers = [1, 2, 3, 4];
if (numbers is [1, 2, .. var rest])
    Console.WriteLine($"9. List pattern: Starts with 1, 2. Rest: {string.Join(",", rest)}");
```

List patterns with Spans

```
Span<int> spanNumbers = stackalloc[] { 10, 20, 30, 40, 50 };
if (spanNumbers is [10, .. var middle, 50])
  Console.WriteLine($"10. List pattern with Span: Starts with 10, ends with 50. " +
    $"Middle: {string.Join(",", [.. middle ])}");
```

Logical Patterns

```
int test = 15;
if (test is > 10 and < 20)
    Console.WriteLine($"10. Logical pattern: {test} is between 10 and 20");
if (test is < 10 or > 20)
    Console.WriteLine($"10. Logical pattern: {test} is less than 10 or greater than 20");
if (test is not 0)
    Console.WriteLine($"10. Logical pattern: {test} is not zero");
```

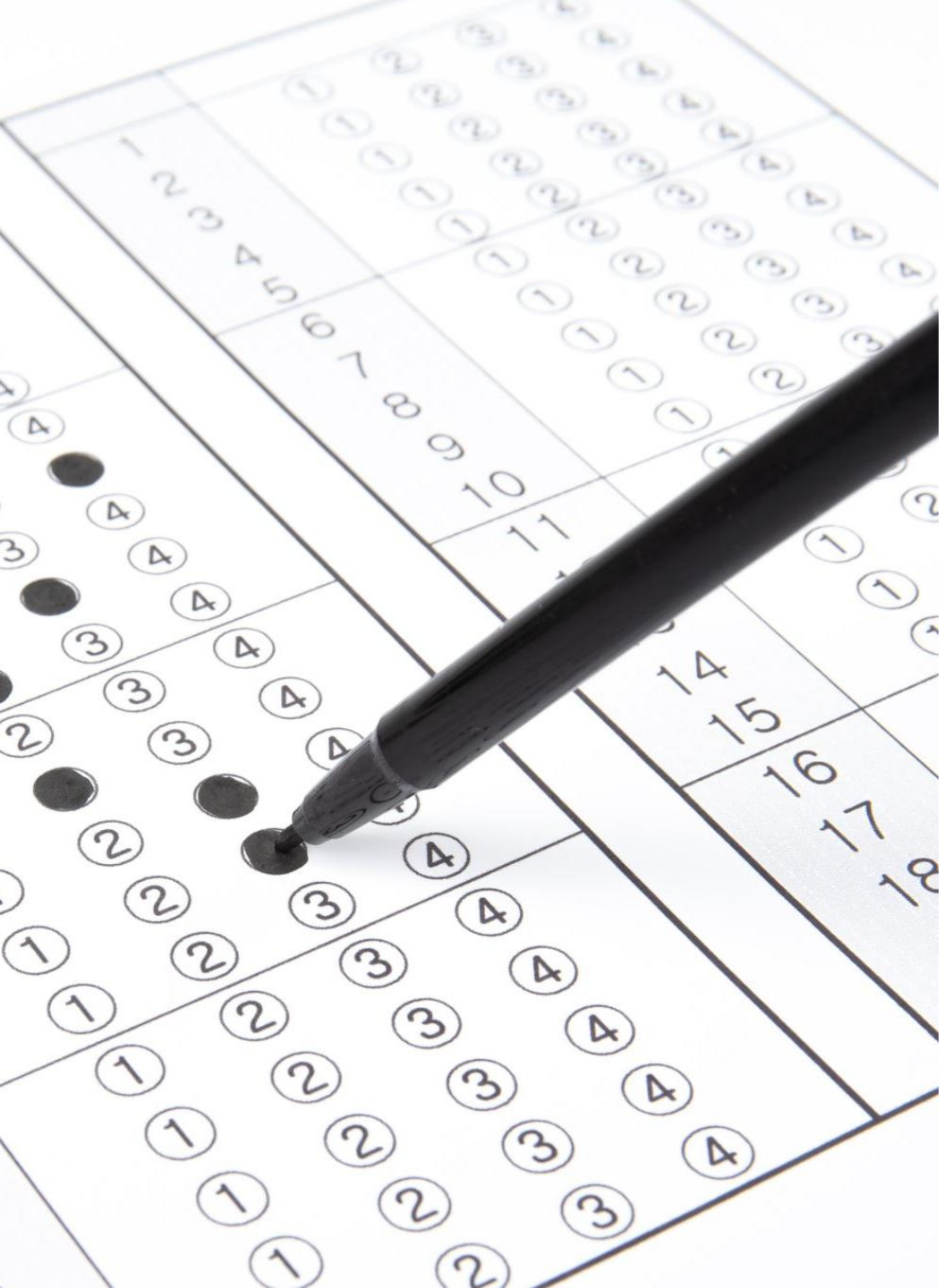
Parenthesized Pattern

```
if (parenthesizedTest is (< 0 or > 10) and not 100)
    Console.WriteLine($"11. Parenthesized pattern: {parenthesizedTest} is less than 0 or greater than 10,
and not 100");
```

Benchmark List Recursive (.NET 9 - .NET 10)

Method – Value 300, .NET 9	Mean
|| == (no pattern)	1.3541 ns
is or	0.6286 ns
switch or	0.2376 ns

Method – Value 300, .NET 10	Mean
|| == (no pattern)	0.4124 ns
is or	0.0007 ns
switch or	0.0189 ns



Sample

Calculate numbers

Benchmark List Recursive (.NET 9)

Method	Mean	Ratio	Allocated
AccumulateFor	27.52 ns	1.0	-
AccumulateForeach	27.57 ns	1.0	-
AccumulateGeneric	27.36 ns	0.99	-
AccumulateRecursive	1,625.50 ns	59.07	22376 B
AccumulateRecursiveSpan	200.66 ns	7.29	-

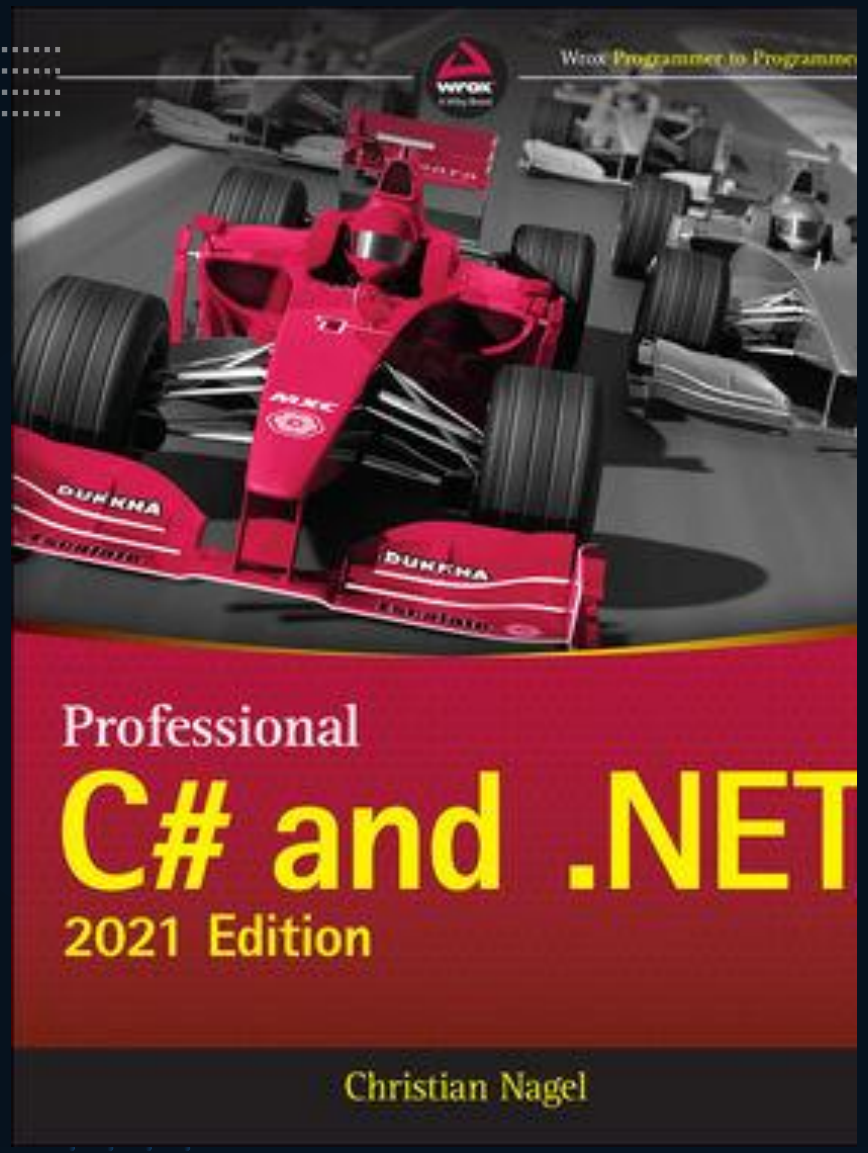
Sample application

Calculate balance from CSV file

Summary Pattern Matching

- Code readability
- Safer and more efficient data handling
- Powerful control flow
- Easy deconstruction of data types
- Improves performance

Questions?



More information

- https://github.com/cnilearn/thrive2025
- https://csharp.christiannagel.com
- https://www.cninnovation.com

THRIVE CONFERENCE

Thank you

We greatly value your input. Please take a moment to complete our survey, accessible either through the Run.events app or by the link below:

https://bit.ly/Thrive25Survey

Syskit · Pearson VUE · LightningTools · span · Leading Learning Partners Association · Harmonia.si · FALA · SAVA ZAVAROVALNICA

barcaffè · ARGETA · Lisjak · Montis · EC-Council Building A Culture Of Security · CISO MAG · Računalniške novice · IVARNOST.SI