

C# 9 Становится лучше

Лободанов Юрий

Sharepoint разработчик, Conteq

ylobodanov@ya.ru

Target-typed new expression

```
static void Main(string[] args)
{
    SomeClass oldExample = new SomeClass();
    var exampleOfVar = new SomeClass();
    SomeClass newExample = new();
}
```

Target-typed new expression

```
static SomeClass Method(SomeClass item)
{
    return new SomeClass();
    return new();
}

static void Main(string[] args)
{
    Method(new SomeClass())
    Method(new());
}
```

Target-typed new expression

```
var someListOld = new List<SomeClassWithLoooongName>()  
{  
    new SomeClassWithLoooongName(),  
    new SomeClassWithLoooongName(),  
    new SomeClassWithLoooongName()  
};  
List<SomeClassWithLoooongName> someListNew = new()  
{  
    new(), new(), new()  
};
```

Target-typed conditional expressions

```
static void Main(string[] args)
{
    var oldExpr = bool ? 0 : (int?)null;
    var oldExpr2 = bool ? 0 : new int?();
    int? newExpr = bool ? 0 : null;
    Person person = student ?? teacher;
}
```

Target-typed – полезно?

- Небольшое уменьшение объема кода
- Читаемость кода станет лучше
- Использовать тернарный оператор стало удобнее
- Грядет второй сезон войны за «правильный» код

Covariant return types

```
Public class Product
{
    public virtual Product Order(string name) => new Product();
}
public class Book : Product
{
    public override Product Order(string name) => new Book();
}
static void Main(string[] args)
{
    var book = (Book)new Book().Order("BookName");
}
```

Covariant return types

```
Public class Product
{
    public Product Order(string name) => new Product();
}
public class Book : Product
{
    public override Book Order(string name) => new Book();
}
static void Main(string[] args)
{
    var book = new Book().Order("BookName");
}
```


Pattern matching improvements

```
if (!(figure is Square)) { ... } //Old
```

```
if (figure is not Square) { ... } //New
```

```
var square = new Square();
```

```
if (square is object && square.SomeProp && square.Area > 1 ) { ... }
```

```
if (square is {SomeProp: true, Area: > 1 }) { ... }
```

```
if (square is {Area: > 1 and (<10 or 25) }) { ... }
```

Pattern matching improvements

```
public decimal CalculateToll(object vehicle) =>
    vehicle switch
    {
        DelTruck t => 10.00m,
        Car c => 2.00m,
        { } => throw new ArgumentException("unknown type", nameof(vehicle)),
        null => throw new ArgumentNullException(nameof(vehicle))
    };
```

Pattern matching improvements

```
public decimal CalculateToll(object vehicle) =>
    vehicle switch
    {
        DelTruck => 10.00m,
        Car => 2.00m,
        { } => throw new ArgumentException("unknown type", nameof(vehicle)),
        null => throw new ArgumentNullException(nameof(vehicle))
    };
```

Pattern matching improvements

```
static float GetCarTax(Car car) =>
    car switch
    {
        Car c when c.Cost < 1000f => 100f,
        Car c when c.Cost < 10000f => 150f,
        Car c when c.Cost < 100000f => 1000f,
        Car c when c.Cost >= 100000f => c.Cost / 150f,
        _ => throw new ArgumentException("", nameof(car))
    };
```

Pattern matching improvements

```
static float GetCarTax(Car car) =>
car.Cost switch
{
    < 1000f => 100f,
    < 10000f => 150f,
    < 100000f => 1000f,
    >= 100000f => car.Cost/150f,
    _ => throw new ArgumentException(" u dumb. Try again", nameof(car))
};
```

Pattern matching improvements

```
static string GetSomeStringByInt(int value) =>
    value switch
    {
        1 or 2 => "Some string 1",
        > 2 and < 5 => "Some string 2",
        (>100 or 99) and (<110 and not 109) => "Some string 3",
        > 1000 and not 1001 => "Some string 4",
        _ => "Some string 5"
    };

```


Pattern matching improvements

```
static float GetCarTax(object car) =>
    car switch
    {
        Truck t=> t.Cost switch
        {
            <= 100000f => 300f,
            > 100000f => 1300f,
            _ => throw new ArgumentException("TruckCostErr", nameof(t))
        },
        ...Car...
        _ => throw new ArgumentException("I do not know this one", nameof(car))
    };

```

Extension GetEnumerator

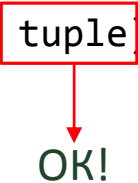
```
static void Main()
{
    var tuple = (new SClass(), new SClass(), new SClass(), new SClass(), new SClass());
    foreach (var item in tuple)
    {
    }
}
```



Error CS1579 foreach statement cannot operate on variables of type '...' because '...' does not contain a public instance or extension definition for 'GetEnumerator'

Extension GetEnumerator

```
public static IEnumerator<T> GetEnumerator<T>(this ValueTuple<T, T, T, T, T> source)
{
    yield return source.Item1;
    yield return source.Item2;
    yield return source.Item3;
    yield return source.Item4;
    yield return source.Item5;
}
...
var tuple = (new SClass(), new SClass(), new SClass(), new SClass(), new SClass());
foreach (var country in tuple)
{
}
OK!
```



Lambda Discard Parameters

```
button1.Click += (s, e) => ShowDialog(); //Old
```

```
button1.Click += (_, _) => ShowDialog(); //New
```

Static anonymous functions

```
static void SomeMethod(Func<int, int> func)
{
    Console.WriteLine(func(10));
}
static void Main(string[] args)
{
    int localY = 10;
    SomeMethod(num => num * localY); // Выведет 100
    SomeMethod(static num => num * localY); // Ошибка компиляции
    const int SomeConst = 20;
    SomeMethod(static num => num * SomeConst); // Выведет 200
}
```

Что еще?

- Attributes on local functions
- Top-level statements
- Function pointers
- Extending Partial
- ...

Спасибо за внимание!
