Problems.Net

SINGULARIS

Анатолий Крыжановский

ООП

```
class Bar { }

void Foo(object a)
{
    Console.WriteLine("object");
}

void Foo(object a, object b)
{
    Console.WriteLine("object, object");
}

void Foo(params object[] args)
{
    Console.WriteLine("params object[]");
}

void Foo<T>(params T[] args)
{
    Console.WriteLine("params T[]");
}
```

```
Foo();
params object[]

Foo(null);
params object[]

Foo(new Bar());
params T[]

Foo(new Bar(), new Bar());
params T[]

Foo(new Bar(), new object());
object, object
```

```
class Foo
{
    public virtual void Quux(int a)
    {
        Console.WriteLine("Foo.Quux(int)");
    }
}

class Bar : Foo
{
    public override void Quux(int a)
    {
        Console.WriteLine("Bar.Quux(int)");
    }

    public void Quux(object a)
    {
        Console.WriteLine("Bar.Quux(object)");
    }
}

class Baz : Bar
{
    public override void Quux(int a)
    {
        Console.WriteLine("Baz.Quux(int)");
    }

    public void Quux<T>(params T[] a)
    {
        Console.WriteLine("Baz.Quux(params T[])");
    }
}
```

```
new Bar().Quux(42);
Bar.Quux(object)

new Baz().Quux(42);
Baz.Quux(params T[])
```

```
class Foo
{
    public Foo()
    {
        Quux();
    }

    public virtual void Quux()
    {
        Console.WriteLine("Foo.Quux()");
    }
}

class Bar : Foo
{
    protected string name;

    public Bar()
    {
        name = "Bar";
    }

    public override void Quux()
    {
        Console.WriteLine("Bar.Quux(), " + name);
    }

    public void Quux(params object[] args)
    {
        Console.WriteLine("Bar.Quux(params object[])");
    }
}
```

```
class Baz : Bar
{
    public Baz()
    {
        name = "Baz";
        Quux();
        ((Foo)this).Quux();
    }
}
```

```
new Baz();
```

Bar.Quux(),

Bar.Quux(params object[])

Bar.Quux(), Baz

```
class Foo
{
    protected class Quux
    {
        public Quux()
        {
            Console.WriteLine("Foo.Quux()");
        }
     }
}

class Bar : Foo
{
    new class Quux
    {
        public Quux()
        {
            Console.WriteLine("Bar.Quux()");
        }
    }
}

class Baz : Bar
{
    public Baz()
    {
        new Quux();
    }
}
```

```
new Baz();
```

```
Foo.Quux()
```

```
class Foo<T>
{
  public static int Bar;
}

void Main()
{
  Foo<int>.Bar++;
  Console.WriteLine(Foo<double>.Bar);
}
```

0

LINQ

```
string GetString(string s)
{
    Console.WriteLine("GetString: " + s);
    return s;
}

IEnumerable<string> GetStringEnumerable()
{
    yield return GetString("Foo");
    yield return GetString("Bar");
}

string[] EnumerableToArray()
{
    var strings = GetStringEnumerable();
    foreach (var s in strings)
        Console.WriteLine("EnumerableToArray: " + s);
            return strings.ToArray();
}

void Main()
{
  EnumerableToArray();
}
```

```
GetString: Foo
EnumerableToArray: Foo
GetString: Bar
EnumerableToArray: Bar
GetString: Foo
GetString: Bar
```

```
IEnumerable<string> Foo()
{
    yield return "Bar";
    Console.WriteLine("Baz");
}

foreach (var str in Foo())
    Console.Write(str);
```

```
BarBaz
```

```
var actions = new List<Action>();
foreach (var i in Enumerable.Range(1, 3))
  actions.Add(() => Console.WriteLine(i));
foreach (var action in actions)
  action();
```

```
.net 4   - 3 3 3
.net 5+ - 1 2 3
```

```
var actions = new List<Action>();
for (int i = 0; i < 3; i++)
  actions.Add(() => Console.WriteLine(i));
foreach (var action in actions)
  action();
```

3 3 3

```
var list = new List<string>
    {
        "Foo", "Bar", "Baz"
    };
var query = list.Where(c => c.StartsWith("B"));
list.Remove("Bar");
Console.WriteLine(query.Count());
```

```
1
```

```
var list = new List<string> { "Foo", "Bar", "Baz" };
var startLetter = "F";
var query = list.Where(c => c.StartsWith(startLetter));
startLetter = "B";
query = query.Where(c => c.StartsWith(startLetter));
Console.WriteLine(query.Count());
```

```
2
```

```
int Inc(int x)
{
  Console.WriteLine("Inc: " + x);
  return x + 1;
}

void Main()
{
  var numbers = Enumerable.Range(0, 10);
  var query =
    (from number in numbers
     let number2 = Inc(number)
     where number2 % 2 == 0
     select number2).Take(2);
  foreach (var number in query)
    Console.WriteLine("Number: " + number);
}
```

```
Inc: 0
Inc: 1
Number: 2
Inc: 2
Inc: 3
Number: 4
```

Math

```
Console.WriteLine( "| Number | Round | Floor | Ceiling | Truncate | Format |");
foreach (var x in new[] { -2.9, -0.5, 0.3, 1.5, 2.5, 2.9 })
{
    Console.WriteLine(string.Format(
        CultureInfo.InvariantCulture,
        "| {0,6} | {1,5} | {2,5} | {3,7} | {4,8} | {0,6:N0} |",
        x, Math.Round(x), Math.Floor(x), Math.Ceiling(x), Math.Truncate(x)));
}
```

```
| Number | Round | Floor | Ceiling | Truncate | Format |
|   -2.9 |    -3 |    -3 |      -2 |       -2 |     -3 |
|   -0.5 |     0 |    -1 |       0 |        0 |     -1 |
|    0.3 |     0 |     0 |       1 |        0 |      0 |
|    1.5 |     2 |     1 |       2 |        1 |      2 |
|    2.5 |     2 |     2 |       3 |        2 |      3 |
|    2.9 |     3 |     2 |       3 |        2 |      3 |
```

```
Console.WriteLine(
    "0.1 + 0.2 {0} 0.3",
    0.1 + 0.2 == 0.3 ? "==" : "!=");
```

```
0.1 + 0.2 != 0.3
```

```
var zero = 0;
try
{
    Console.WriteLine(42 / 0.0);
    Console.WriteLine(42.0 / 0);
    Console.WriteLine(42 / zero);
}
catch (DivideByZeroException)
{
    Console.WriteLine("DivideByZeroException");
}
```

```
Infinity
Infinity
DivideByZeroException
```

```
var maxInt32 = Int32.MaxValue;
checked
{
    Console.Write("Checked Int32 increased max: ");
    try
    {
         Console.WriteLine(maxInt32 + 42);
    }
    catch
    {
        Console.WriteLine("OverflowException");
    }
}
```

```
Checked Int32 increased max: OverflowException
```

```
var maxDouble = Double.MaxValue; checked
{
    Console.Write("Checked Double increased max: ");
    try
    {
        Console.WriteLine(maxDouble + 42);
    }
    catch
    {
        Console.WriteLine("OverflowException");
    }
}
```

```
Checked Double increased max: 1,79769313486232E+308
```

```
var maxDecimal = Decimal.MaxValue;
checked
{
    Console.Write("Checked Decimal increased max: ");
    try
    {
        Console.WriteLine(maxDecimal + 42);
    }
    catch
    {
        Console.WriteLine("OverflowException");
    }
}
```

```
Checked Decimal increased max: OverflowException
```

```
var maxInt32 = Int32.MaxValue;
unchecked
{
    Console.Write("Unchecked Int32 increased max: ");
    try
    {
        Console.WriteLine(maxInt32 + 42);
    }
    catch
    {
        Console.WriteLine("OverflowException");
    }
}
```

```
Unchecked Int32 increased max:-2147483607
```

```
var maxDouble = Double.MaxValue;
unchecked
{
    Console.Write("Unchecked Double increased max: ");
    try
    {
        Console.WriteLine(maxDouble + 42);
    }
    catch
    {
        Console.WriteLine("OverflowException");
    }
}
```

```
Unchecked Double increased max: 1,79769313486232E+308
```

```
var maxDecimal = Decimal.MaxValue;
unchecked
{
    Console.Write("Unchecked Decimal increased max: ");
    try
    {
        Console.WriteLine(maxDecimal + 42);
    }
    catch
    {
        Console.WriteLine("OverflowException");
    }
}
```

```
Unchecked Decimal increased max: OverflowException
```

```
int a = 0;
int Foo()
{
    a = a + 42;
    return 1;
}

void Main()
{
    a += Foo();
    Console.WriteLine(a);
}
```

```
1
```

```
byte foo = 1;
dynamic bar = foo;
Console.WriteLine(bar.GetType());
bar += foo;
Console.WriteLine(bar.GetType());
```

```
System.Byte
System.Int32
```

Value Type

```
struct Foo
{
    int value;
    public override string ToString()
    {
        if (value == 2)
            return "Baz";
        return (value++ == 0) ?
                "Foo" :
                "Bar";
    }
}
```

```
void Main()
{
    var foo = new Foo();
    Console.WriteLine(foo);
    Console.WriteLine(foo);
    object bar = foo;
    object qux = foo;
    object baz = bar;
    Console.WriteLine(baz);
    Console.WriteLine(bar);
    Console.WriteLine(baz);
    Console.WriteLine(qux);
}
```

```
Foo
Foo
Foo
Bar
Baz
Foo
```

```
public struct Foo
{
    public int Value;
    public void Change(int newValue)
    {
        Value = newValue;
    }
}

public class Bar
{
    public Foo Foo { get; set; }
}

void Main()
{
    var bar = new Bar { Foo = new Foo() };
    bar.Foo.Change(5);

    Console.WriteLine(bar.Foo.Value);
}
```

0

```
var x = new
    {
        Items = new List<int> { 1, 2, 3 }.GetEnumerator()
    };
while (x.Items.MoveNext())
    Console.WriteLine(x.Items.Current);
```

```
public struct Foo
{
    public byte Byte1;
    public int Int1;
}

Console.WriteLine(Marshal.SizeOf(typeof(Foo)));
```

```
8
```

```
public struct Bar
{
  public byte Byte1;
  public byte Byte2;
  public byte Byte3;
  public byte Byte4;
  public int Int1;
}

Console.WriteLine(Marshal.SizeOf(typeof(Bar)));
```

```
8
```

Strings

```
Console.WriteLine(1 + 2 + "A");
Console.WriteLine(1 + "A" + 2);
Console.WriteLine("A" + 1 + 2);
```

```
3A
1A2
A12
```

```
Console.WriteLine(1 + 2 + 'A');
Console.WriteLine(1 + 'A' + 2);
Console.WriteLine('A' + 1 + 2);
```

```
68
68
68
```

```
try
{
    Console.WriteLine(((string)null + null + null) == null);
}
catch (Exception e)
{
    Console.WriteLine(e.GetType());
}
```

```
False
```

```
Console.WriteLine( string.Compare(a.ToUpper(), b.ToUpper()));

Console.WriteLine( string.Compare(a, b,
    StringComparison.OrdinalIgnoreCase));
```

```
var x = "AB";
var y = new StringBuilder().Append('A').Append('B').ToString();
var z = string.Intern(y);
Console.WriteLine(x == y);
Console.WriteLine(x == z);
Console.WriteLine((object)x == (object)y);
Console.WriteLine((object)x == (object)z);
```

```
True
True
False
True
```

Threads

```
[ThreadStatic]
static readonly int Foo = 42;

void Main()
{
    var thread = new Thread(() => Console.WriteLine(Foo));
    thread.Start();
    thread.Join();
}
```

```
0
```