



# What's new in C# 8, 9 and 10?

Filip Ekberg

# Hello,

I'm Filip Ekberg

@fekberg

 **smorgasbord** free @ [filipekberg.se](mailto:filipekberg.se)

 PLURALSIGHT

 **fekberg**



Using Statements for Static Members  
Implicitly typed local variables  
Extension methods  
Getter & Setter separate accessibility  
Null-conditional operators  
Exception Filters  
Anonymous methods  
Named and optional parameters  
Static classes  
Lambdas  
Auto-properties  
String interpolation  
Nullable types  
Dynamic binding  
Digit Separators  
private protected  
stackalloc with Span  
Default Expressions  
digit separators  
non-trailing named arguments  
out improvements  
Expression trees  
Dictionary Initializers  
Tuples and Deconstruction  
Ref Assemblies  
Async & Await  
Infer Tuple Names  
Query expressions  
ref returns  
Await inside Catch & Finally blocks  
nameof operator  
Partial types  
Improved generic constraints  
Generic co- and contravariance  
Pattern Matching  
Object and collection initializers  
conditional ref operator  
Tuple equality  
Binary Literals  
Async Main  
Expression-bodied members  
Auto-Property Initializers  
Anonymous types  
ref readonly & in parameter



13 contributors



98 lines (79 sloc) 14.6 KB

Raw

Blame

History



## Language Feature Status

This document reflects the status, and planned work in progress, for the compiler team. It is a live document and will be updated as work progresses, features are added / removed, and as work on feature progresses. This is not an exhaustive list of our features but rather the ones which have active development efforts behind them.

### C# Next

Feature	Branch	State	Developers	Reviewer	LDM Champ
Caller expression attribute	caller-expression	Prototype	alrz	jcouv	jcouv
Target-typed new	target-typed-new	Prototype	alrz	jcouv	jcouv
Generic attributes	generic-attributes	In Progress	AviAvni	agocke	mattwar
Default in deconstruction	decon-default	Implemented	jcouv	gafter	jcouv
Relax ordering of <code>ref</code> and <code>partial</code> modifiers	ref-partial	In Progress	alrz	gafter	jcouv
Parameter null-checking	param-nullchecking	In Progress	fayrose	agocke	jaredpar



# Highlights from C# 7.0 => C# 7.3

# Tuples before C# 7.0

C# Program.cs

```
3
4 var tuple =
5     new Tuple<string, string, int>("Filip", "Ekberg", 34);
6
7 tuple.
```

- Equals
- GetHashCode
- GetType
- Item1
- Item2
- Item3
- ToString
- ToValueTuple<>

`bool Tuple<string, string, int>.Equals(object? obj)` ×

Returns a value that indicates whether the current `Tuple<T1, T2, T3>` object is equal to a specified object.

# Tuples in C# 7.0

Program.cs

```
3  
4 var tuple = (first: "Filip", last: "Ekberg", age: 34);  
5
```

```
6 tuple.
```

age

CompareTo

Equals

first

GetHashCode

GetType

last

ToString

ToTuple<>

(field) int (string first, string last, int age).age ×

Gets the value of the current ( T1, T2, T3 ) instance's third element.

# Deconstruct an object

```
public void Deconstruct(out string name, out int age)
{
    name = Name;
    age = Age;
}
```

```
var (name, age) = person;
```

```
Console.WriteLine($"{name}, is {age} years old");
```



# Tuples & Deconstruction

```
(int x, int y) coordinates = (10, 20);
```

```
coordinates.x;  
coordinates.y;
```

```
var coordinate = new Coordinate(10, 20);
```

```
var (x, y) = coordinate;
```

```
public void Deconstruct(out int x, out int y)  
{  
    x = X;  
    y = Y;  
}
```

# C# 8 and Beyond

# C# 8.0

Readonly members

Default interface implementations

**Pattern matching** enhancements

Using declarations

Static local functions

**Nullable reference types**

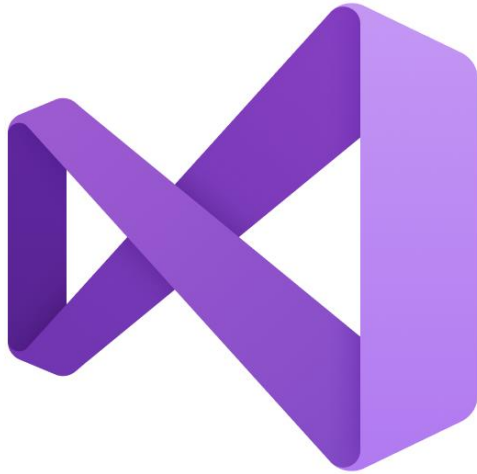
**Asynchronous streams**

Asynchronous disposable

Ranges and Indices

Null-coalescing assignment


# How do you use C# 8, 9 and 10?



- Visual Studio 2019 or later
- .NET Core 3.0 for C# 8
- .NET 5.0 for C# 9
- .NET 6.0 for C# 10

# Add a new project

## Recent project templates

 ASP.NET Core Web Application C#

 WPF App (.NET Framework) C#

 ASP.NET Web Application (.NET Framework) C#

WPF



[Clear](#)

All Languages

All Platforms

All Project Types



WPF App (.NET Framework)

Windows Presentation Foundation client application

C#

Windows

Desktop



WPF App (.NET Core)

Windows Presentation Foundation client application

C#

Windows

Desktop



WPF Custom Control Library (.NET Core)

Windows Presentation Foundation custom control library

C#

Windows

Desktop

Library



WPF User Control Library (.NET Core)

Windows Presentation Foundation user control library

C#

Windows

Desktop

Library



WPF App (.NET Framework)

Windows Presentation Foundation client application

Visual Basic

Windows

Desktop



WPF Library (.NET Core)


Windows Presentation Foundation client application

[Next](#)

# Add a new project

## Recent project templates

 ASP.NET Core Web Application C#

 WPF App (.NET Framework) C#

 ASP.NET Web Application (.NET Framework) C#

Forms



[Clear](#)

All Languages

All Platforms

All Project Types



Windows Forms App (.NET Core)

A project for creating an application with a Windows Forms (WinForms) user interface

C#

Windows

Desktop



Windows Forms App (.NET Framework)

A project for creating an application with a Windows Forms (WinForms) user interface

C#

Windows

Desktop



Windows Forms Control Library (.NET Framework)

A project for creating controls to use in Windows Forms (WinForms) applications

C#

Windows

Desktop

Library



Windows Forms App (.NET Framework)

A project for creating an application with a Windows Forms (WinForms) user interface

Visual Basic

Windows

Desktop



Windows Forms Control Library (.NET Framework)

A project for creating controls to use in Windows Forms (WinForms) applications

Visual Basic

Windows

Desktop

Library



ASP.NET Web Application (.NET Framework)

Project templates for creating ASP.NET applications. You can create ASP.NET Web

[Next](#)



# Default Interface Implementations

```
public interface IProductRepository
{
    Task<Product> FindByAsync() => Task.FromResult(default);
}
```

# Shared Libraries

```
public interface IProductRepository
{
    Task<Product> FindByAsync() => Task.FromResult(default);
    Product FindBy(string articleId);
    int GetStockFor(string articleId);
    IEnumerable<Product> GetAll();
    void DecreaseStockBy(string articleId, int amount);
    void IncreaseStockBy(string articleId, int amount);
}
```

(awaitable) Task<Product> IProductRepository.FindByAsync()

Usage:

Product x = await FindByAsync();


Target runtime doesn't support default interface implementation.

[Show potential fixes \(Ctrl+.\)](#)

# Shared Libraries

```
public interface IProductRepository
{
    Task<Product> FindByAsync() => Task.FromResult(default);

    Product FindBy(string articleId);
    int GetStockFor(string articleId);
    IEnumerable<Product> GetAll();
    void DecreaseStockBy(string articleId, int amount);
    void IncreaseStockBy(string articleId, int amount);
}
```

 (awaitable) Task<Product> IProductRepository.FindByAsync()

Usage:

Product x = await FindByAsync();

Target runtime doesn't support default interface implementation.

[Show potential fixes \(Ctrl+.\)](#)

# Shared Libraries

```
public interface IProductRepository  
{  
    Task<Product> FindByAsync() => Task.FromResult(default);  
}
```

⚙️ (awaitable) Task<Product> IProductRepository.FindByAsync()

Usage:

Product x = await FindByAsync();

Target runtime doesn't support default interface implementation.

---

Show potential fixes (Ctrl+.)

# Nullable reference types

# Server Error in '/' Application.

*Object reference not set to an instance of an object.*

**Description:** An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error that originated in the code.

**Exception Details:** System.NullReferenceException: Object reference not set to an instance of an object.

**Source Error:**

```
Line 431:  
Line 432:  
Line 433:  
Line 434:  
Line 435: }
```

New Comparision

Unhandled exception has occurred in your application. If you click Continue, the application will ignore this error and attempt to continue. If you click Quit, the application will close immediately.

Object reference not set to an instance of an object.

Details

Continue

Quit

See the end of this message for details on invoking just-in-time (JIT) debugging instead of this dialog box.

Exception Text

System.NullReferenceException: Object reference not set to an instance of an object  
at System.Configuration.SettingsBase.SetPropertyValueByName(String propertyName, Object value)  
at System.Configuration.SettingsBase.set\_Item(String propertyName, Object value)  
at System.Configuration.ApplicationSettingsBase.set\_Item(String propertyName, Object value)  
at DBCompare.Common.WriteRecentList(List`1 Keys, String \_KeyName)  
at DBCompare.Common.WriteRecentList(String \_KeyName, ComboBox cb)

Microsoft Development Environment

An unhandled exception of type 'System.NullReferenceException' occurred in Unknown Module.

Additional information: Object reference not set to an instance of an object.

Break

Continue

Ignore

Help

```
edPosts");
```

```
st<string> fruits = null;  
ruits ?? new List<string>()).Add("apple");  
nsole.WriteLine(fruits.Count);
```

Form1

Unhandled exception has occurred in your application. If you click Continue, the application will ignore this error and attempt to continue. If you click Quit, the application will close immediately.

Object reference not set to an instance of an object.

Details

Continue

Quit

Exception Unhandled

System.NullReferenceException: 'Object reference not set to an instance of an object.'

fruits was null.

View Details

Copy Details

Start Live Share session...

Exception Settings



# What is the problem with nullability?

- **Null-checks all over the place** makes the code harder to read
- **What does null mean?** Object wasn't found? Error?
- **Encourages mutability**
- Issues with mutability in multi-threading

```
if(person != null &&  
    person.Address != null &&  
    person.Address.Street != null)  
{ }
```

```
if(person?.Address?.Street != null)  
{ }
```

#nullable enable

```
string name = null;  
name.Split(' ');
```



[ ] (local variable) `string` name

'name' may be null here.

CS8602: Dereference of a possibly null reference.

Show potential fixes (Alt+Enter or Ctrl+.)

```
string name = null;  
name.Split(' ');
```



[ ] (local variable) `string` name

'name' may be null here.

CS8602: Dereference of a possibly null reference.

Show potential fixes (Alt+Enter or Ctrl+.)

Treat warnings as errors

☐ None

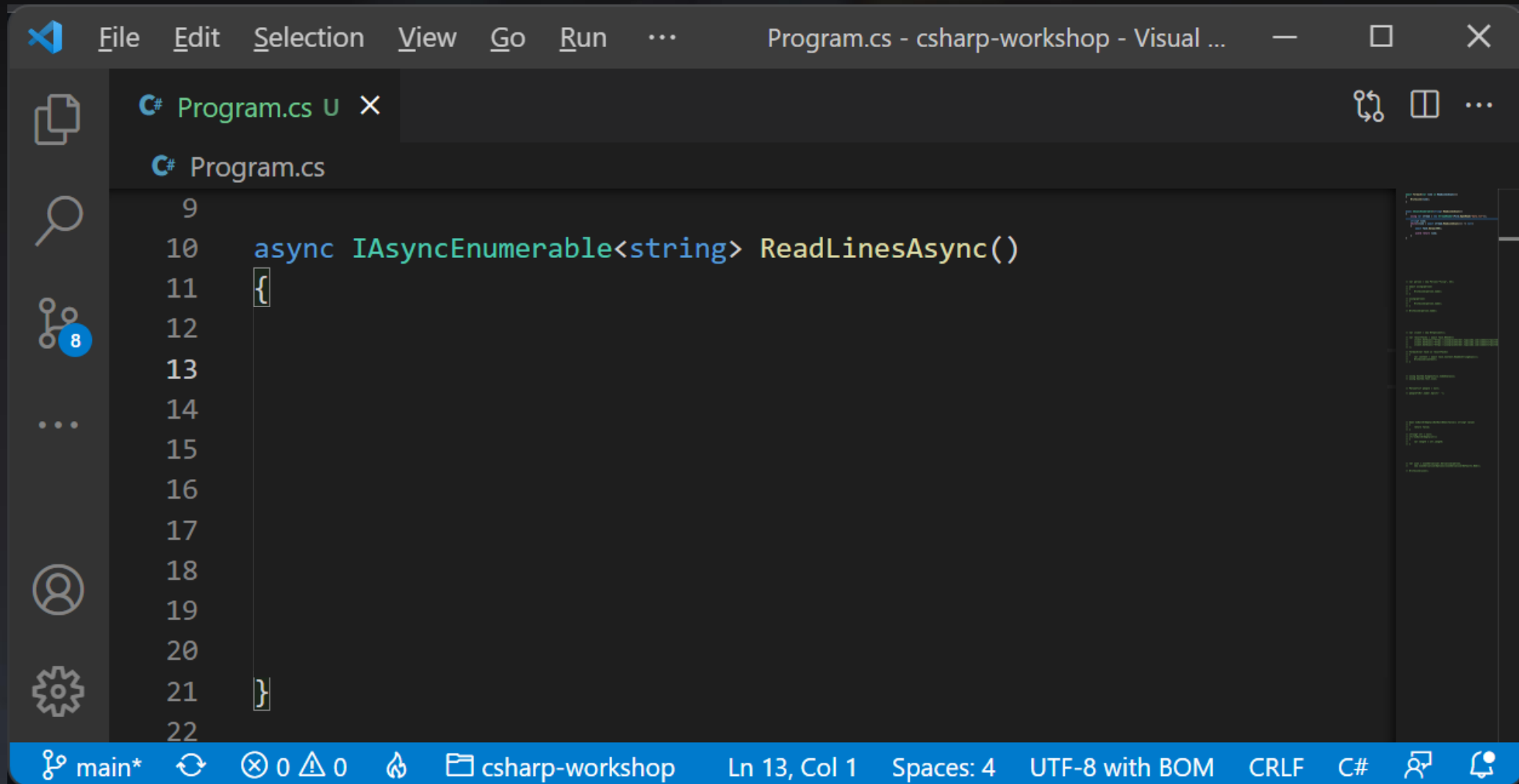
☒ All

Error List		
Entire Solution ▾ 8 Errors 0 Warnings 0 of 26 Messages Build + IntelliSense ▾		
	Code	Description ^
✖	CS8618	Non-nullable property 'Genre' is uninitialized. Consider declaring the property as nullable.
✖	CS8618	Non-nullable property 'ImdbID' is uninitialized. Consider declaring the property as nullable.
✖	CS8618	Non-nullable property 'ImdbID' is uninitialized. Consider declaring the property as nullable.
✖	CS8618	Non-nullable property 'Poster' is uninitialized. Consider declaring the property as nullable.
✖	CS8618	Non-nullable property 'Released' is uninitialized. Consider declaring the property as nullable.
✖	CS8618	Non-nullable property 'Runtime' is uninitialized. Consider declaring the property as nullable.
✖	CS8618	Non-nullable property 'Title' is uninitialized. Consider declaring the property as nullable.
✖	CS8618	Non-nullable property 'Year' is uninitialized. Consider declaring the property as nullable.



# Asynchronous Streams

# Streaming data

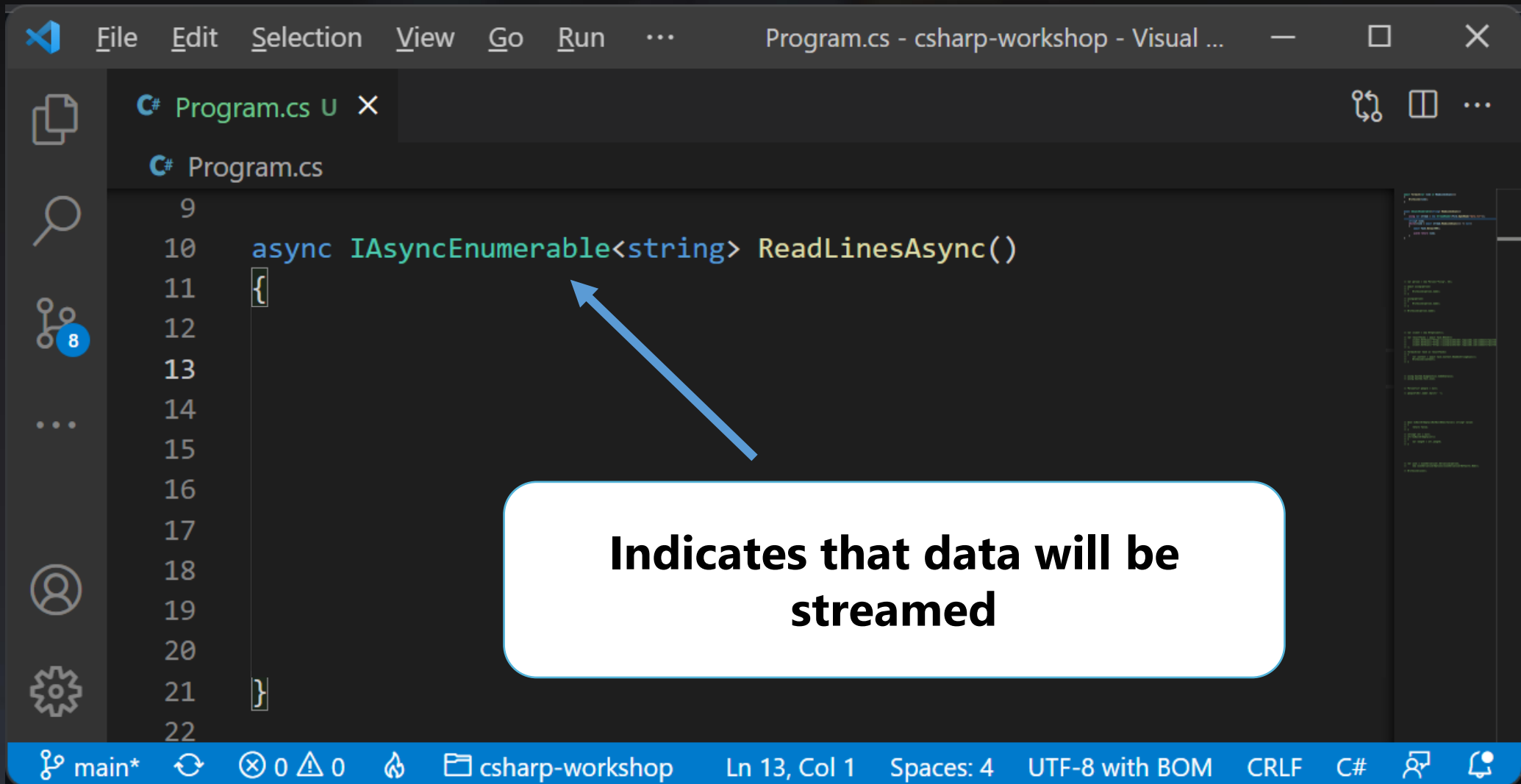


The image shows a screenshot of the Visual Studio Code editor interface. The title bar at the top reads "Program.cs - csharp-workshop - Visual ...". The menu bar includes "File", "Edit", "Selection", "View", "Go", "Run", and a dropdown menu. The left sidebar contains icons for Explorer, Search, Source Control (with a badge showing 8 changes), Run and Debug, and Settings. The main editor area displays a C# file named "Program.cs" with the following code:

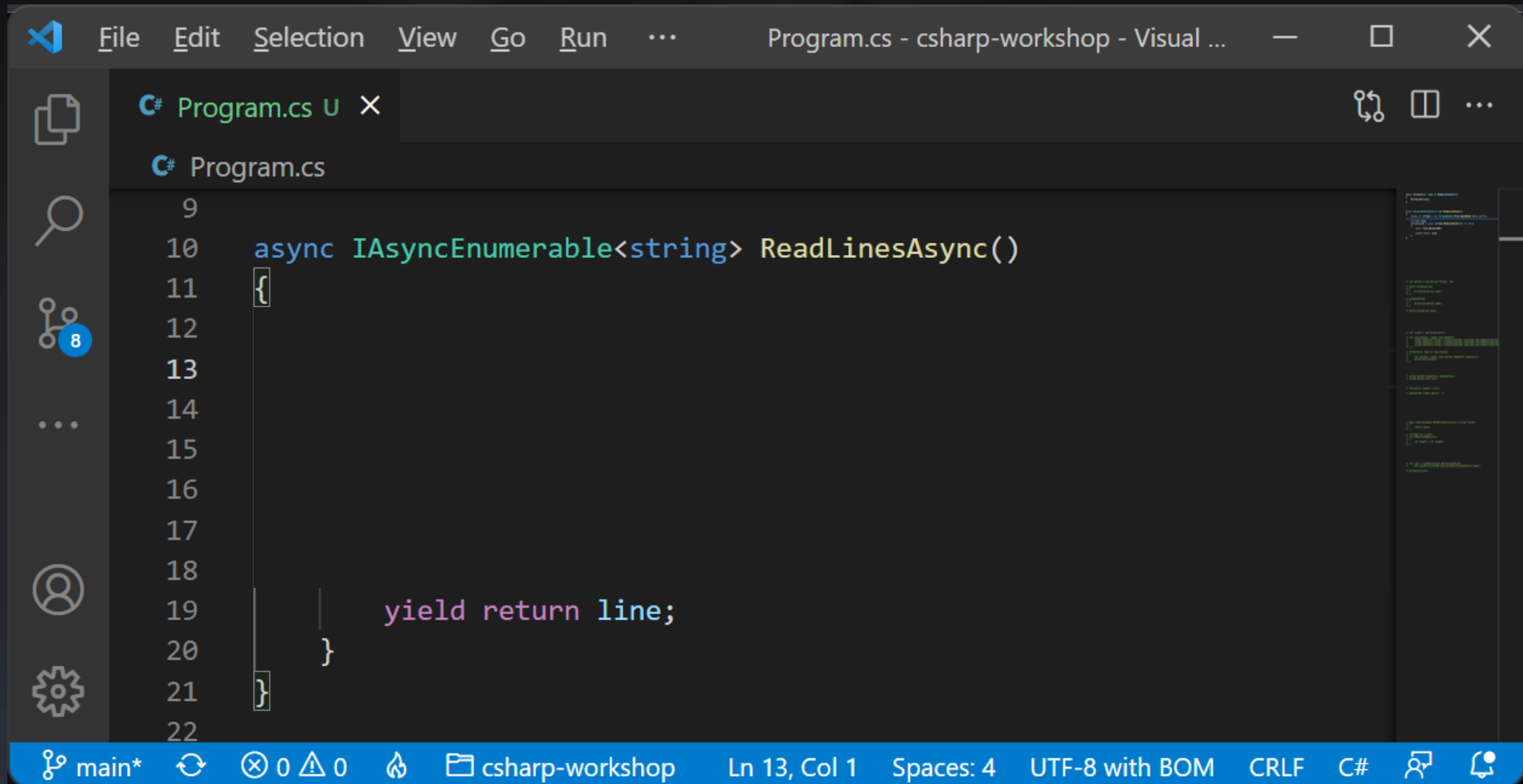
```
9
10  async IEnumerable<string> ReadLinesAsync()
11  {
12
13
14
15
16
17
18
19
20
21  }
22
```

The status bar at the bottom shows "main\*", a refresh icon, "0 0" errors/warnings, a flame icon, the folder "csharp-workshop", "Ln 13, Col 1", "Spaces: 4", "UTF-8 with BOM", "CRLF", "C#", and icons for Run and Debug.

# Streaming data



# Streaming data

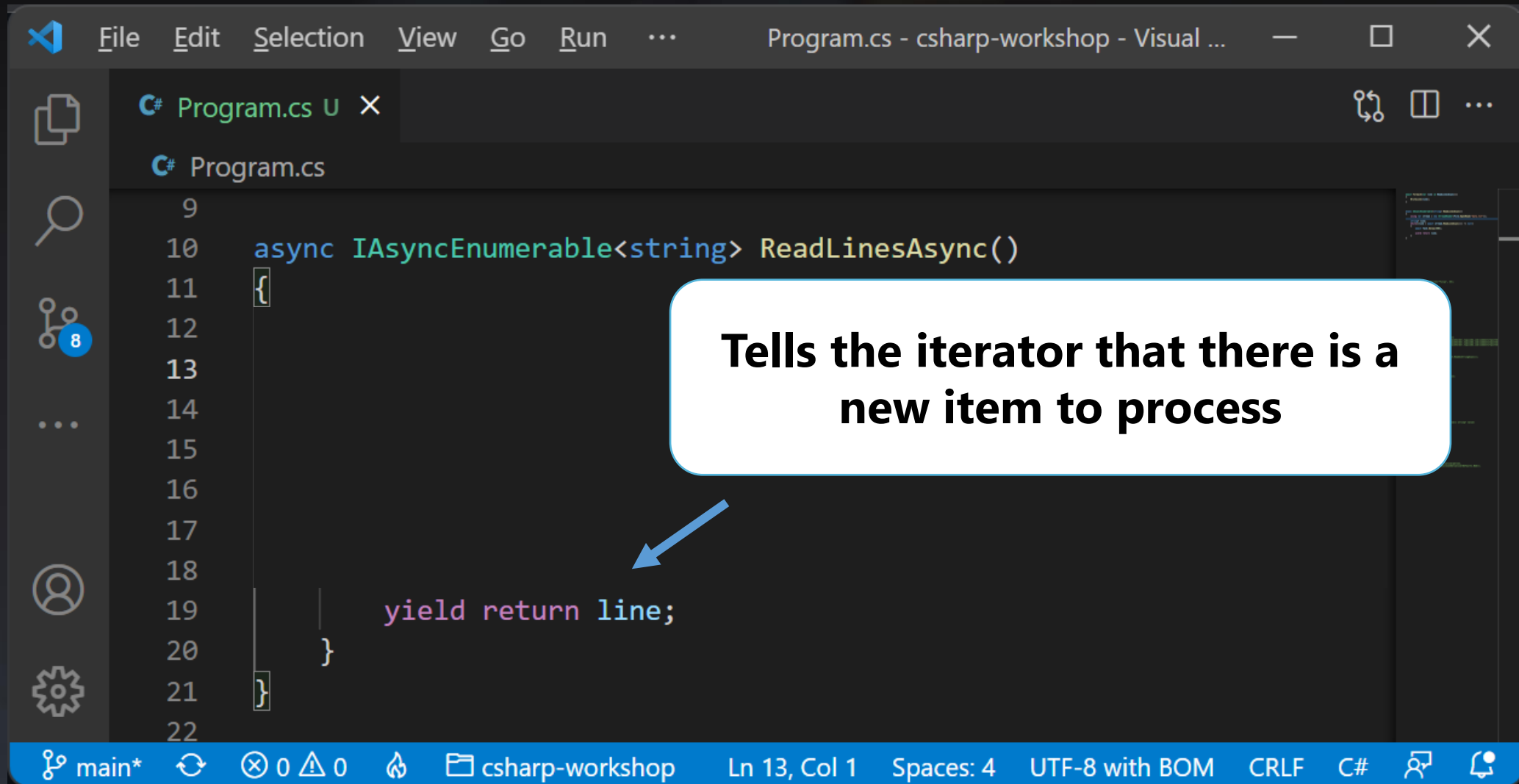


The screenshot shows the Visual Studio Code editor interface. The title bar at the top reads "Program.cs - csharp-workshop - Visual ...". The menu bar includes "File", "Edit", "Selection", "View", "Go", "Run", and a dropdown menu. The left sidebar contains icons for Explorer, Search, Source Control (with a blue badge showing '8'), Run and Debug, and Settings. The main editor area displays a C# file named "Program.cs" with the following code:

```
9
10  async IEnumerable<string> ReadLinesAsync()
11  {
12
13
14
15
16
17
18
19      yield return line;
20  }
21
22
```

The status bar at the bottom shows "main\*", a refresh icon, "0 0" (errors/warnings), a flame icon, "csharp-workshop", "Ln 13, Col 1", "Spaces: 4", "UTF-8 with BOM", "CRLF", "C#", and icons for Run and Debug.

# Streaming data



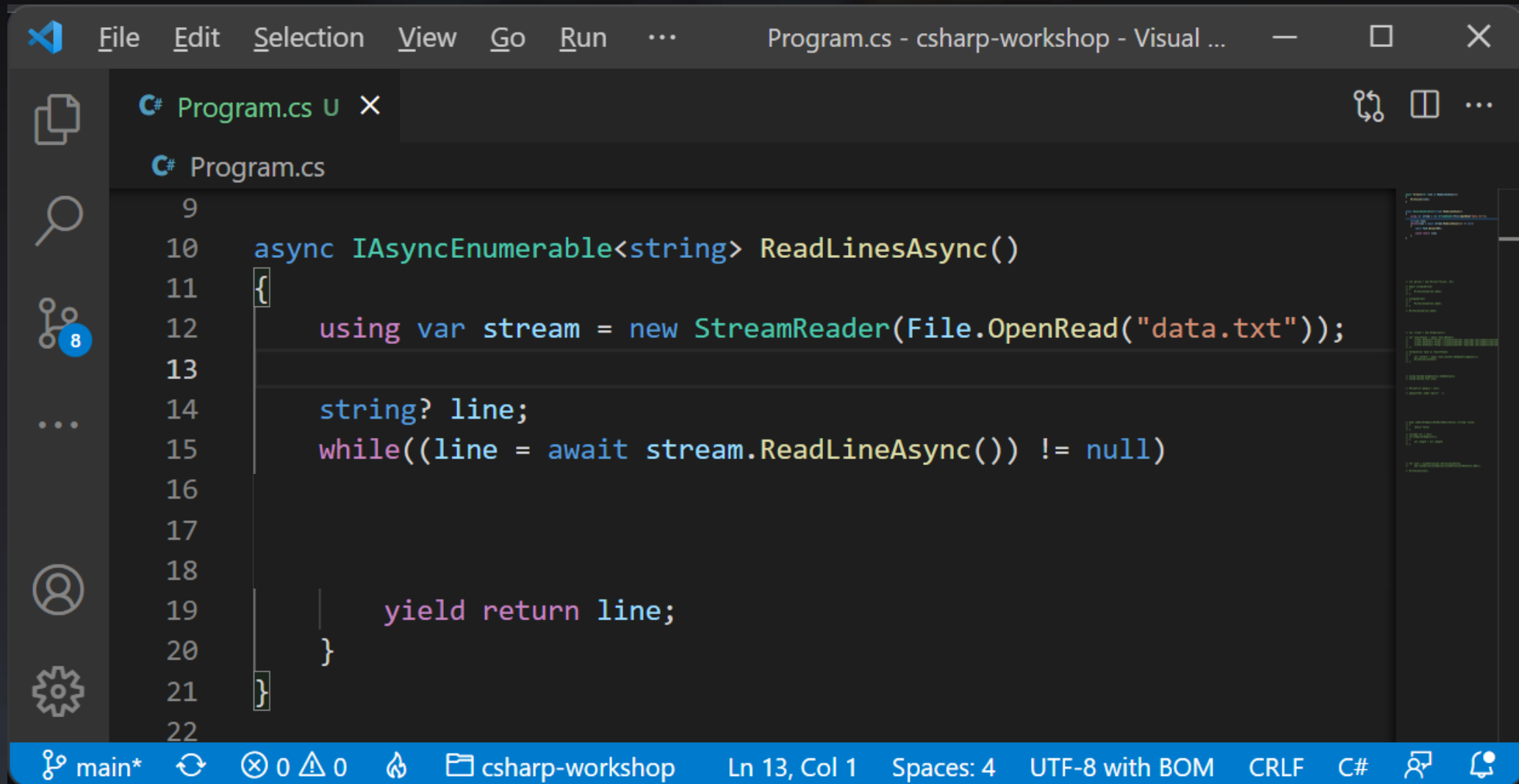
The screenshot shows the Visual Studio Code editor with a C# file named `Program.cs`. The code defines an asynchronous enumerable `ReadLinesAsync()` using `yield return` to stream data. A callout box highlights the `yield return line;` statement, explaining its function.

```
9
10 async IEnumerable<string> ReadLinesAsync()
11 {
12
13
14
15
16
17
18
19     yield return line;
20 }
21
22
```

**Tells the iterator that there is a new item to process**

main\* 0 0 csharp-workshop Ln 13, Col 1 Spaces: 4 UTF-8 with BOM CRLF

# Streaming data



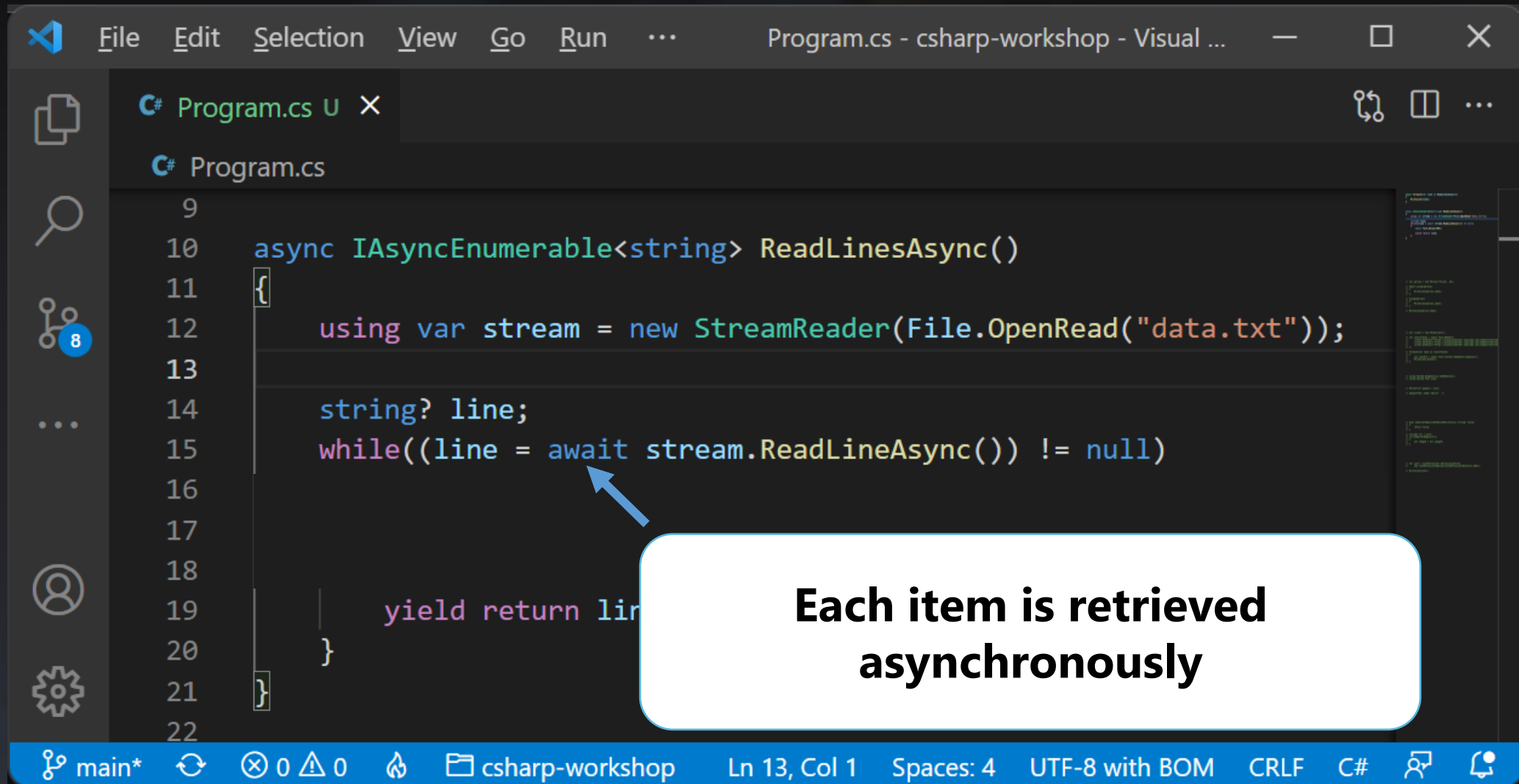
The image shows a screenshot of the Visual Studio Code editor interface. The title bar at the top reads "Program.cs - csharp-workshop - Visual ...". The menu bar includes "File", "Edit", "Selection", "View", "Go", "Run", and a dropdown menu. The left sidebar contains icons for Explorer, Search, Source Control, and Settings. The main editor area displays a C# file named "Program.cs" with the following code:

```
9
10 async IEnumerable<string> ReadLinesAsync()
11 {
12     using var stream = new StreamReader(File.OpenRead("data.txt"));
13
14     string? line;
15     while((line = await stream.ReadLineAsync()) != null)
16
17
18
19         yield return line;
20     }
21 }
22
```

The status bar at the bottom shows "main\*", a refresh icon, "0 0" errors/warnings, a flame icon, the folder "csharp-workshop", "Ln 13, Col 1", "Spaces: 4", "UTF-8 with BOM", "CRLF", "C#", and icons for a comment and a bell.



# Streaming data

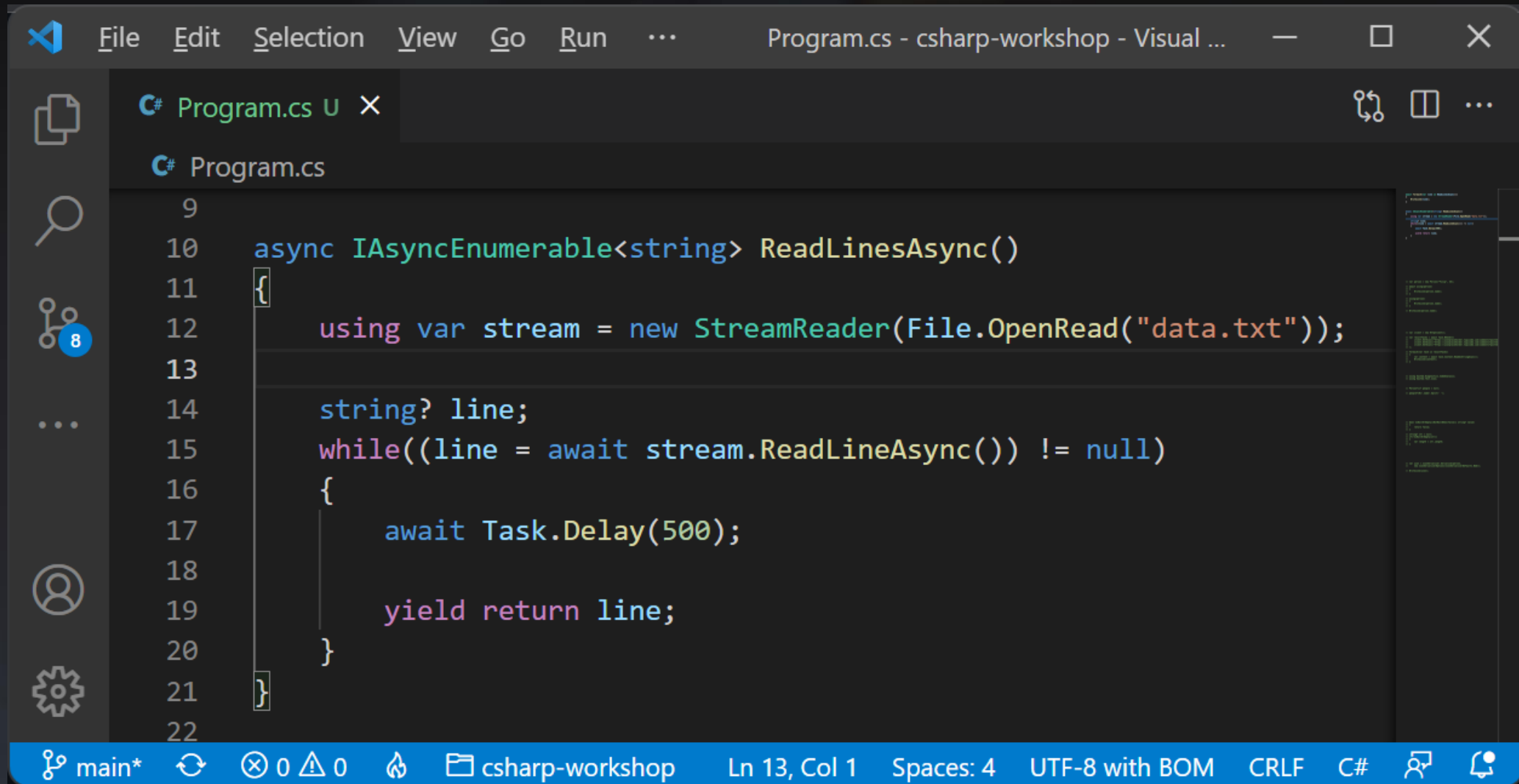


```
9
10 async IEnumerable<string> ReadLinesAsync()
11 {
12     using var stream = new StreamReader(File.OpenRead("data.txt"));
13
14     string? line;
15     while((line = await stream.ReadLineAsync()) != null)
16
17
18
19     yield return line;
20 }
21 }
22
```

Each item is retrieved asynchronously

main\* 0 0 csharp-workshop Ln 13, Col 1 Spaces: 4 UTF-8 with BOM CRLF

# Streaming data



The screenshot shows the Visual Studio Code editor interface. The title bar at the top reads "Program.cs - csharp-workshop - Visual ...". The menu bar includes "File", "Edit", "Selection", "View", "Go", "Run", and a dropdown menu. The left sidebar contains icons for Explorer, Search, Source Control, and Settings. The Explorer view shows a file named "Program.cs". The main editor area displays the following C# code:

```
9
10 async IEnumerable<string> ReadLinesAsync()
11 {
12     using var stream = new StreamReader(File.OpenRead("data.txt"));
13
14     string? line;
15     while((line = await stream.ReadLineAsync()) != null)
16     {
17         await Task.Delay(500);
18
19         yield return line;
20     }
21 }
22
```

The status bar at the bottom shows "main\*", a refresh icon, "0 0", a flame icon, "csharp-workshop", "Ln 13, Col 1", "Spaces: 4", "UTF-8 with BOM", "CRLF", "C#", and icons for a comment and a bell.

# Consuming the IAsyncEnumerable

```
await foreach(var element in GetElements())
```

# C# 9.0

## **Records**

## **Init only setters**

## **Top-level statements**

## **Pattern matching enhancements**

## **Target-typed new expressions**

Native sized integers

Function pointers

Suppress emitting localsinit flag

static anonymous functions

Target-typed conditional expressions

Covariant return types

Extension GetEnumerator support for  
foreach loops

Lambda discard parameters

Attributes on local functions

Module initializers

New features for partial methods

# Pattern matching



# Matching a type & "attribute"

C# 1.0 – 6.0

```
if(fruit.GetType() == typeof(Apple) && fruit.Color == Color.Green)
{
    var apple = fruit as Apple;

    var food = MakeApplePieFrom(apple);
}
```

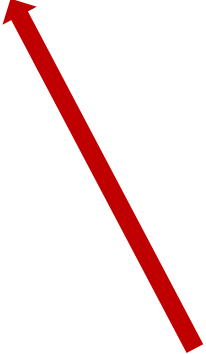
C# 7.0

```
switch(fruit)
{
    case Apple apple when apple.Color == Color.Green:
        var food = MakeApplePieFrom(apple);
        break;

    default:
        break;
}
```

# Pattern matching: **Switch expression**

```
var result = input switch  
{  
  
};
```



**The object you want to create patterns for**



# Pattern matching: **Switch expression**

```
var result = input switch  
{  
};
```



**Fill the switch body with expressions**

# Pattern matching: **Switch expression**

```
var whatFruit = fruit switch  
{  
    Apple => "This is an apple",  
    _ => "This is not an apple"  
};
```



**This is what is returned when  
the pattern is a match**

# Patterns in C#

- **Type** pattern
- **Positional** pattern
- **Property** pattern
- **Tuple** pattern
- **Relational** pattern
- Conjunctive "**and**" pattern
- Disjunctive "**or**" pattern
- **Parenthesized** pattern
- Negated "**not**" pattern
- **Recursive** patterns

# C# 10.0

## **Record structs**

Improvements of structure types

## **global using directives**

**File-scoped namespace** declaration

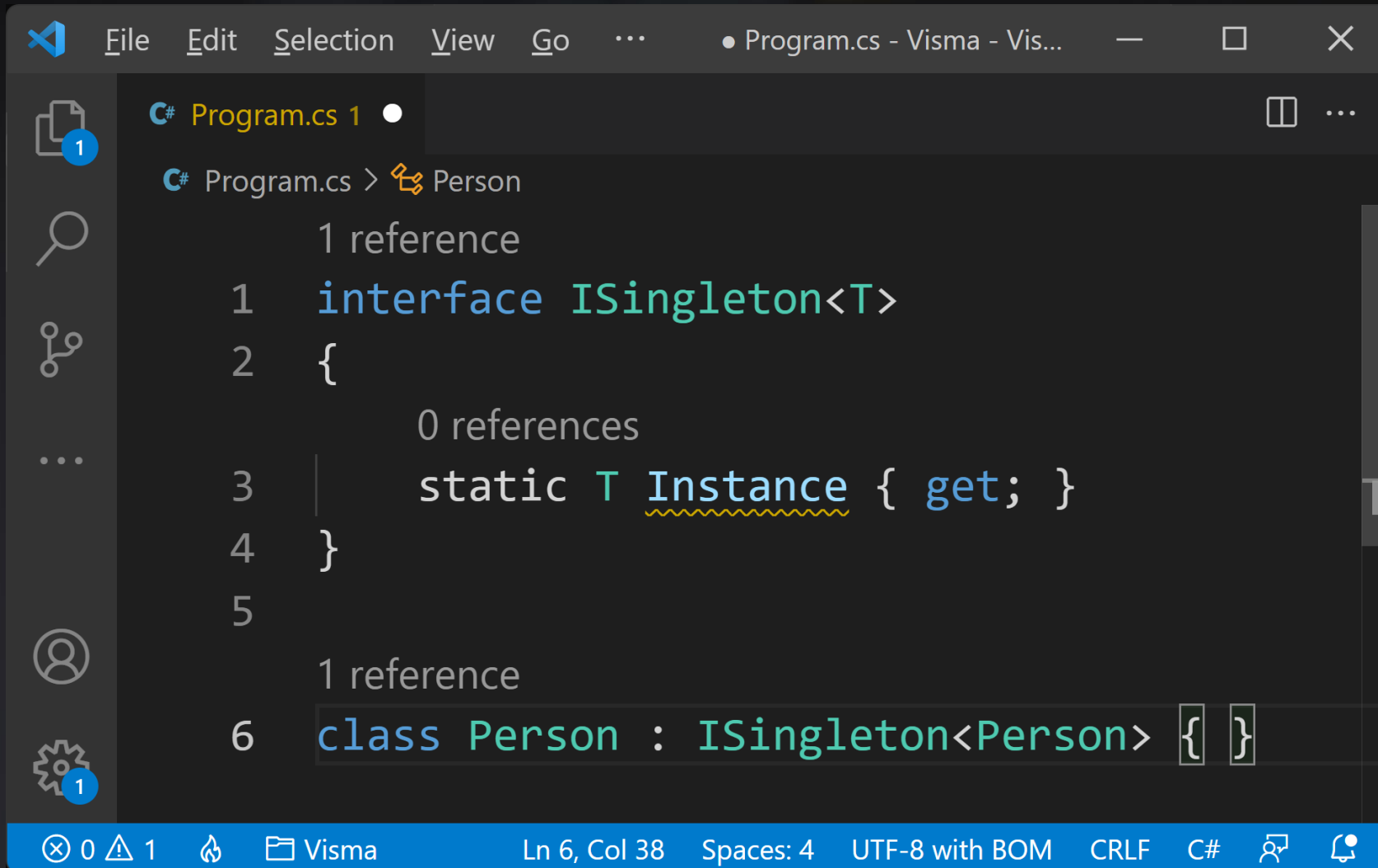
Extended **property patterns**

Improvements on lambda expressions

Allow **const interpolated strings**

Allow both assignment and declaration in the same deconstruction

# Static abstract members in interfaces

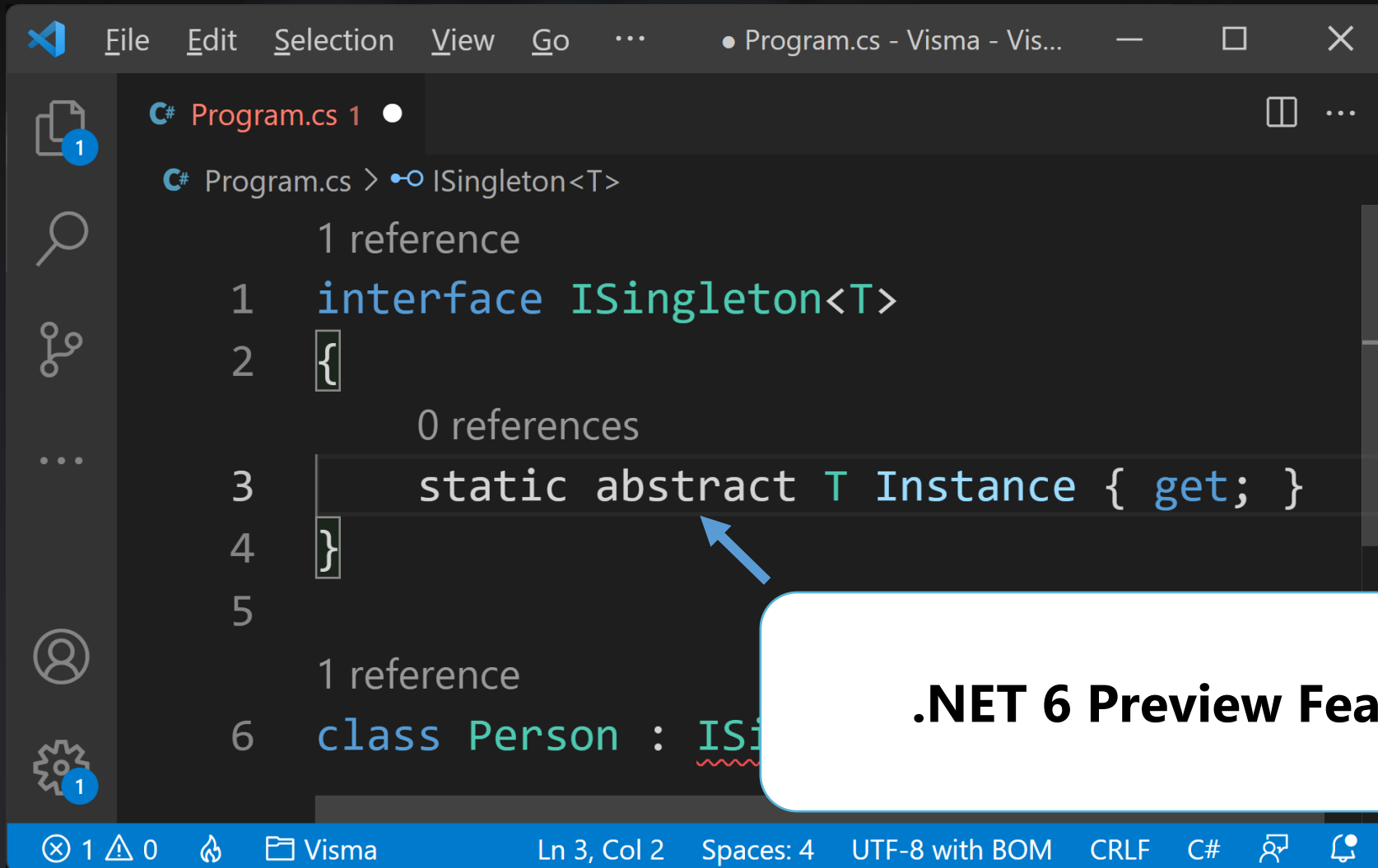


The screenshot shows the Visual Studio IDE with a C# file named Program.cs. The code defines an interface `ISingleton<T>` with a static abstract property `Instance` and a class `Person` that implements this interface. The `Instance` property is underlined with a yellow squiggly line, indicating it is abstract. The `Person` class is highlighted with a blue selection box.

```
File Edit Selection View Go ... • Program.cs - Visma - Vis...
C# Program.cs 1
C# Program.cs > Person
    1 reference
1  interface ISingleton<T>
2  {
    0 references
3      static T Instance { get; }
4  }
5
    1 reference
6  class Person : ISingleton<Person> { }
```

0 1 Visma Ln 6, Col 38 Spaces: 4 UTF-8 with BOM CRLF C#

# Static abstract members in interfaces

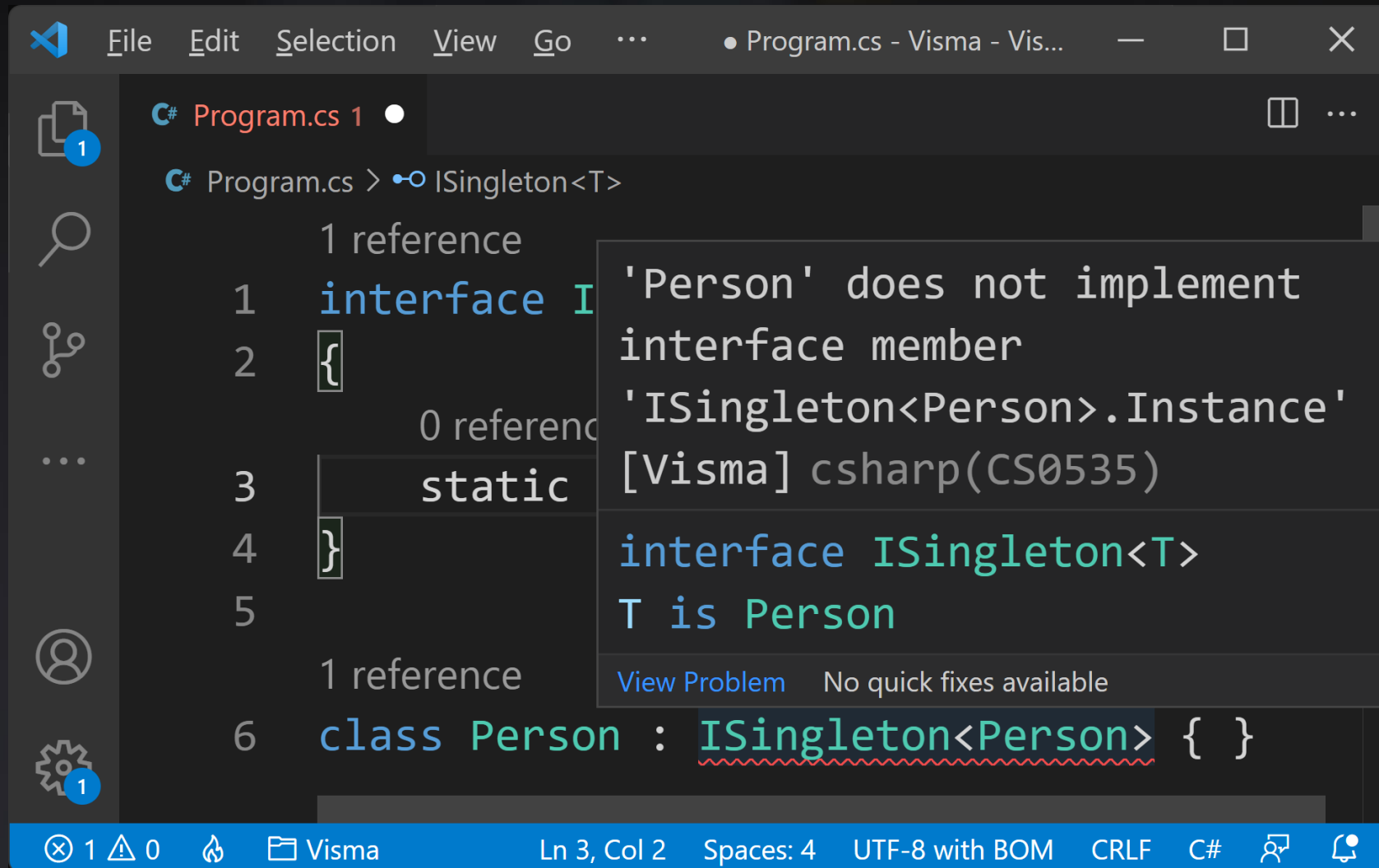


The screenshot shows the Visual Studio IDE with a C# file named Program.cs. The code defines an interface `ISingleton<T>` with a static abstract property `Instance` of type `T`. Below the interface, a class `Person` is shown, which implements the `ISingleton<T>` interface. A blue arrow points to the `static abstract` keywords in the `Instance` property declaration. The status bar at the bottom indicates the file is named 'Visma' and is using UTF-8 encoding with BOM.

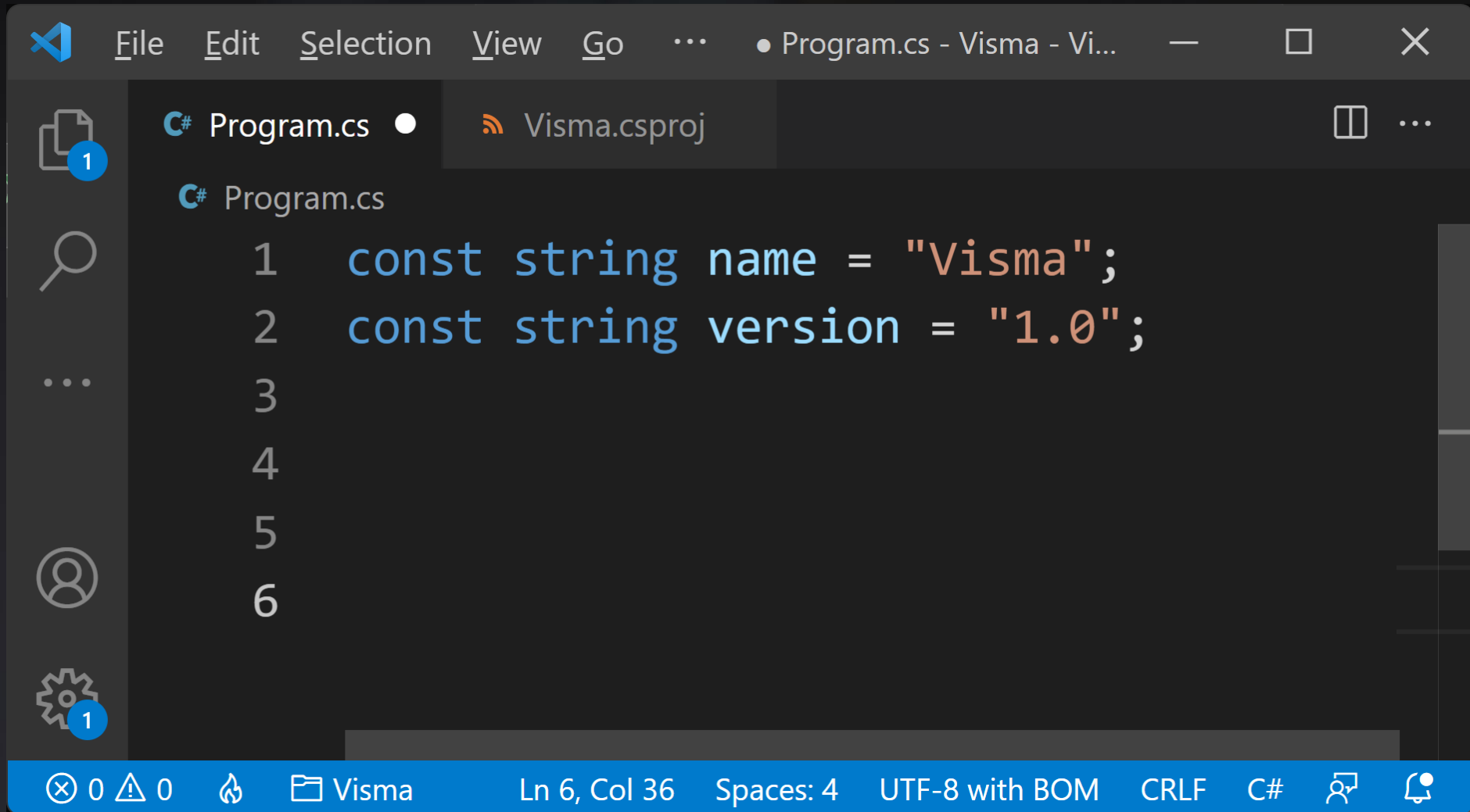
```
File Edit Selection View Go ... • Program.cs - Visma - Vis...
C# Program.cs 1
C# Program.cs > ISingleton<T>
1 reference
1 interface ISingleton<T>
2 {
0 references
3     static abstract T Instance { get; }
4 }
5
1 reference
6 class Person : ISingleton<T>
```

**.NET 6 Preview Feature**

# Static abstract members in interfaces



# Constant Interpolated Strings



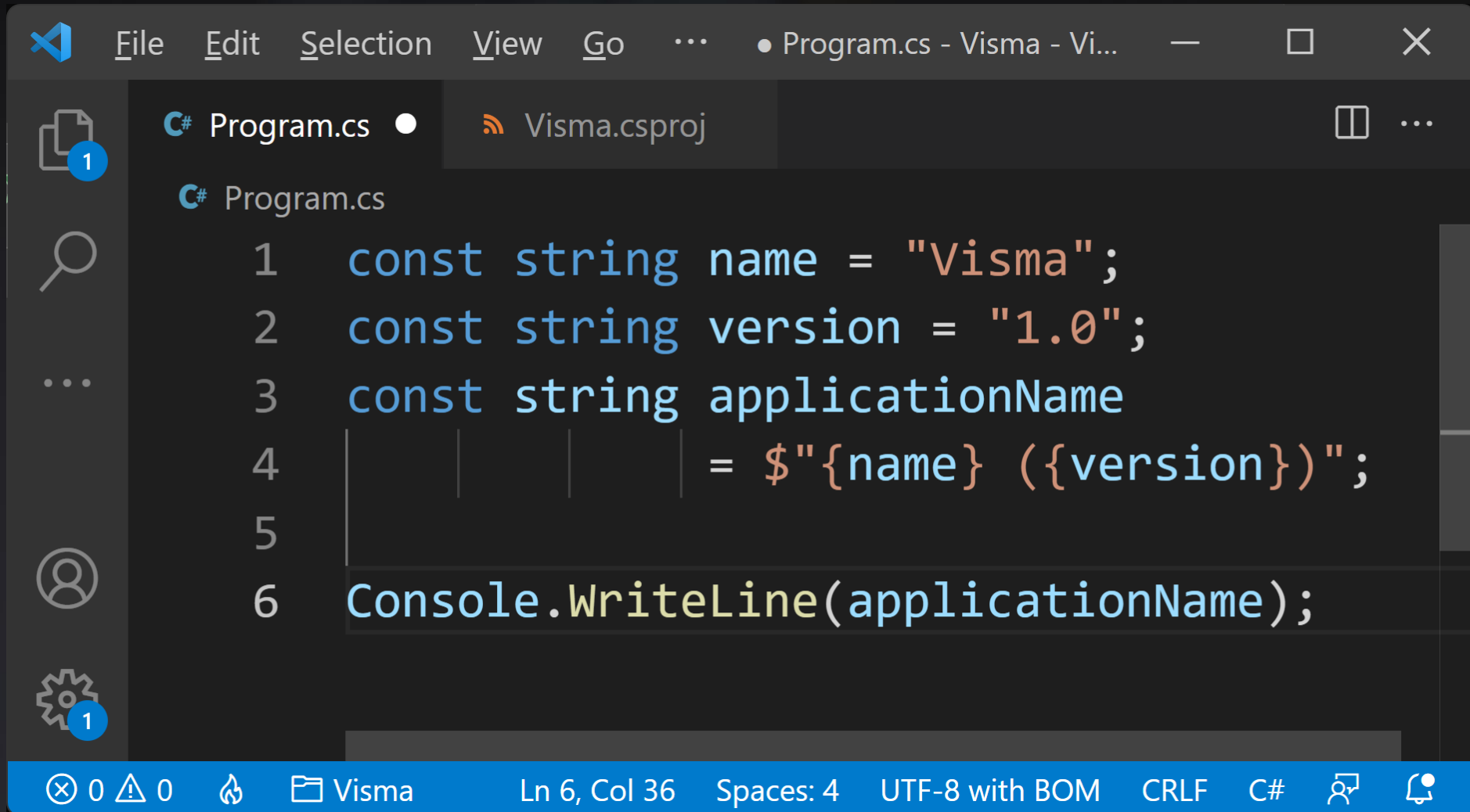
The image shows a screenshot of the Visual Studio Code editor interface. The title bar at the top reads "Program.cs - Visma - Vi...". The menu bar includes "File", "Edit", "Selection", "View", "Go", and a dropdown menu. The left sidebar contains icons for Explorer (with a "1" badge), Search, Source Control, and Settings (with a "1" badge). The main editor area displays a C# file named "Program.cs" with the following code:

```
1  const string name = "Visma";  
2  const string version = "1.0";  
3  
4  
5  
6
```

The status bar at the bottom shows "0" errors, "0" warnings, a folder icon, "Visma", "Ln 6, Col 36", "Spaces: 4", "UTF-8 with BOM", "CRLF", "C#", and icons for Run and Debug.



# Constant Interpolated Strings




The screenshot shows the Visual Studio IDE with a C# file named Program.cs. The code defines three constant strings and uses an interpolated string to combine them. The status bar at the bottom indicates the current position is at line 6, column 36, with 4 spaces, UTF-8 encoding with BOM, and CRLF line endings.

```
File Edit Selection View Go ... • Program.cs - Visma - Vi...
C# Program.cs • Visma.csproj
C# Program.cs
1  const string name = "Visma";
2  const string version = "1.0";
3  const string applicationName
4  = $"{name} ({version})";
5
6  Console.WriteLine(applicationName);
ⓧ 0 ⚠ 0 🔥 📁 Visma Ln 6, Col 36 Spaces: 4 UTF-8 with BOM CRLF C# 🔍 🔔
```

# Inferred Delegate Type (Lambda)

```
Func<string, int> parse  
    = (string number) => int.Parse(number);
```



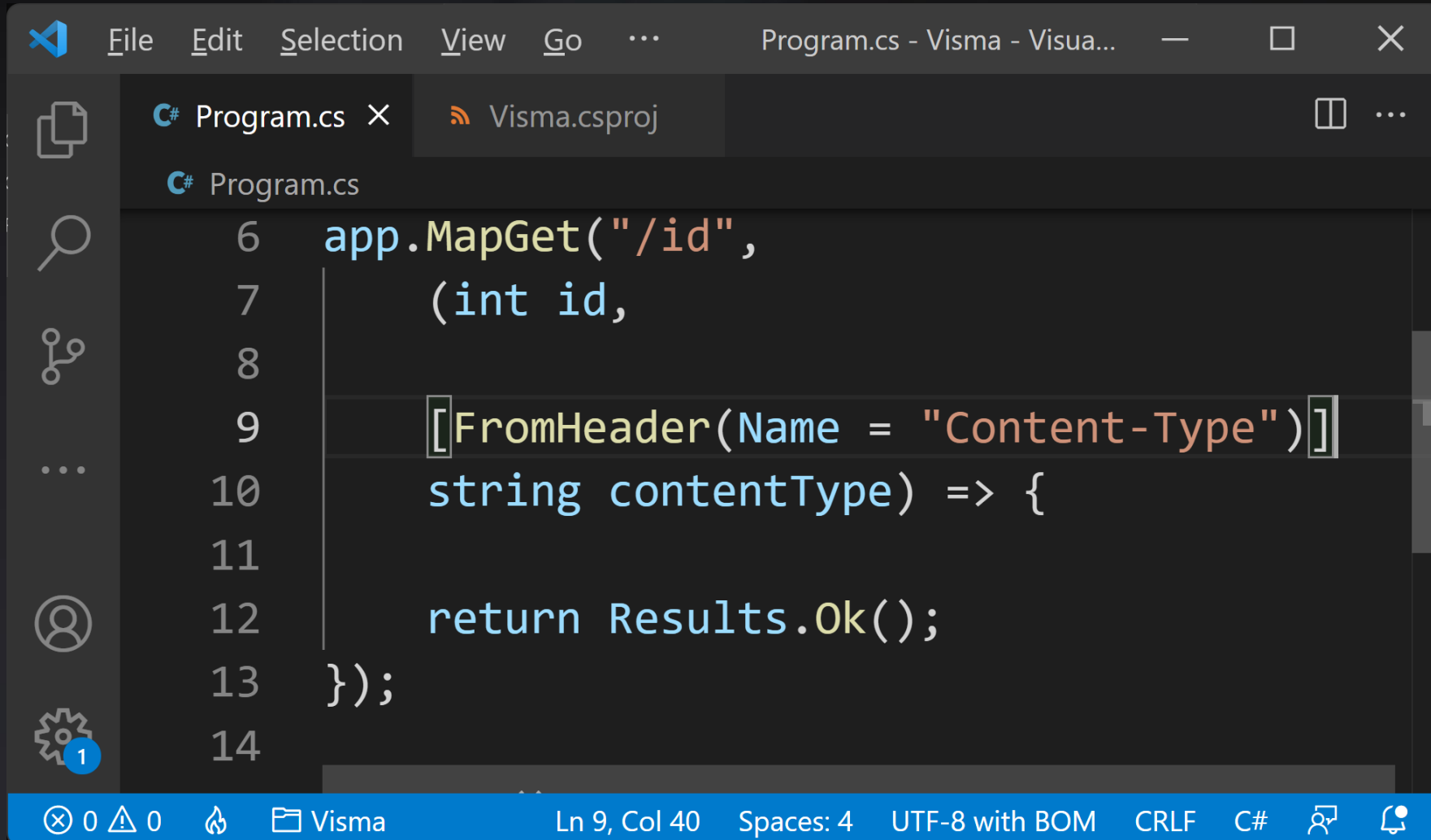
**C# 9.0 requires you to explicitly  
specify delegate type**

# Inferred Delegate Type (Lambda)

```
Func<string, int> parse  
    = (string number) => int.Parse(number);
```

```
var parse  
    = (string number) => int.Parse(number);
```

# Attributes in Lambdas



The screenshot shows the Visual Studio IDE with a C# file named Program.cs. The code defines a lambda function for the MapGet method. The lambda function has two parameters: an integer id and a string contentType. The contentType parameter is annotated with the FromHeader attribute, specifying the name 'Content-Type'. The lambda function returns Results.Ok().

```
6 app.MapGet("/id",  
7     (int id,  
8  
9     [[FromHeader(Name = "Content-Type")]]  
10    string contentType) => {  
11  
12    return Results.Ok();  
13 });  
14
```

The status bar at the bottom indicates the current position is Line 9, Column 40, with 4 spaces, UTF-8 encoding with BOM, and CRLF line endings.

**What's next?**

# C# 11.0

## More patterns

**List** pattern

**Slice** pattern

**Span** pattern

**nameof**(parameter)

**Cache delegates** for static method group

**Required** properties

```
required string Name { get; init; }
```

**Raw string** literals

**Static members** in interfaces

**UTF-8 String** Literals

```
Span<byte> span = "Filip Ekberg";
```

**Generic attributes**

```
[ThisIsAGenericAttribute<string>()]
```

**Parameter null-checking** (!!) has been **withdrawn**  
due to community feedback



Filip Ekberg

Pluralsight Author

2067 Followers

Filip is an enthusiastic developer that strives to learn something new every day. With over a decade of experience in .NET, Filip actively spreads his knowledge and ideas around the globe, be it speaking at conferences or online. Filip has worked in a range of different technologies such as WPF,...

Show more...

- fekberg.com
- Twitter
- Facebook
- LinkedIn

CONTENT AUTHORED

20

All time

TOPICS AUTHORED

C#

TOTAL RATINGS

4,020

AVG CONTENT RATING

4.4

Content authored

- C# 10 Advanced Language Features

NEW!

Course · Intermediate · 6 hr 43m · Mar 31, 2022 · ★★★★★ (17)
- Implement Microsoft Azure Event Hub

NEW!

Lab · Beginner · 1 hr 15m · Feb 1, 2022 · ★★★★★ (131)
- Globalization in C# Applications: Best Practices

Course · Intermediate · 2 hr 35m · May 6, 2021 · ★★★★★ (27)
- Building Multithreaded C# Applications with the Task Parallel Library

Course · Intermediate · 1 hr 45m · Dec 22, 2020 · ★★★★★ (86)
- Applying Asynchronous Programming in C#

Course · Intermediate · 3 hr 19m · Dec 11, 2020 · ★★★★★ (241)

# Thanks!

I'm Filip Ekberg

@fekberg

 **smorgasbord** free @ filipekberg.se

 PLURALSIGHT

 **fekberg**

