ASP.NET CORE

Documentation

Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document. When you're ready to add your content, just click here and start typing.]

Dhruvil Dobariya
dhruvildobariya21@gmail.com

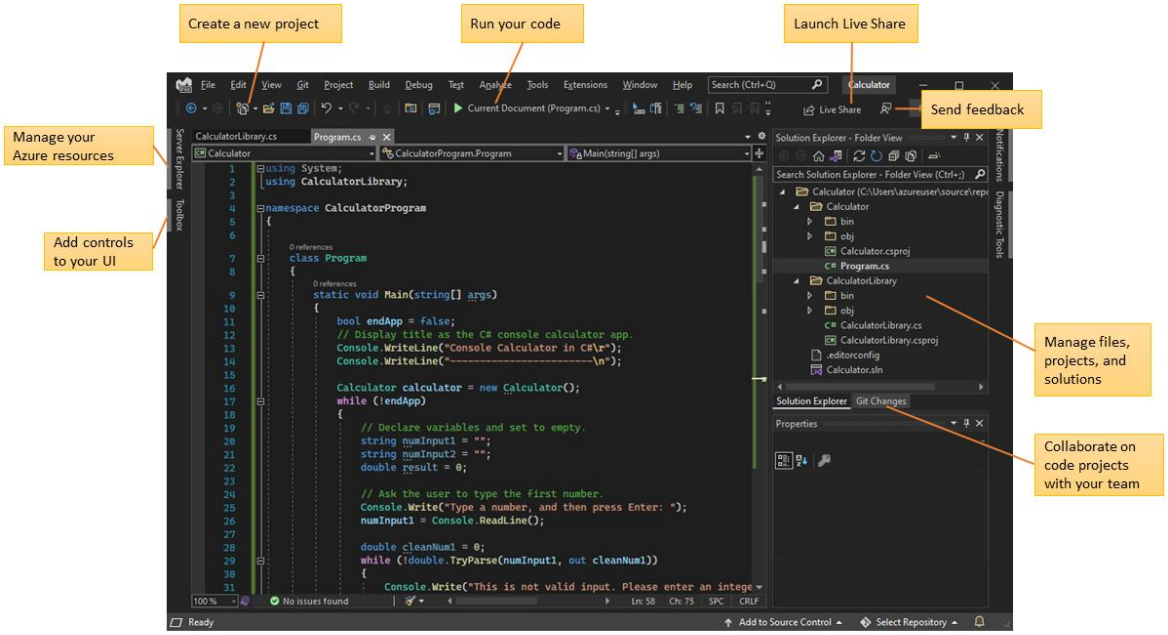
INDEX

VISUAL STUDIO 2019 IDE OVERVIEW

1.1 WHAT IS A VISUAL STUDIO?

- Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps.
- Visual Studio supports 36 different programming languages and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists.

1.2 DIFFERENT TYPES OF WINDOWS



- In Solution Explorer, at upper right, you can view, navigate, and manage your code files. Solution Explorer can help organize your code by grouping the files into solutions and projects.
- The central editor window, where you'll probably spend most of your time, displays file contents. In the editor window, you can edit code or design a user interface such as a window with buttons and text boxes.

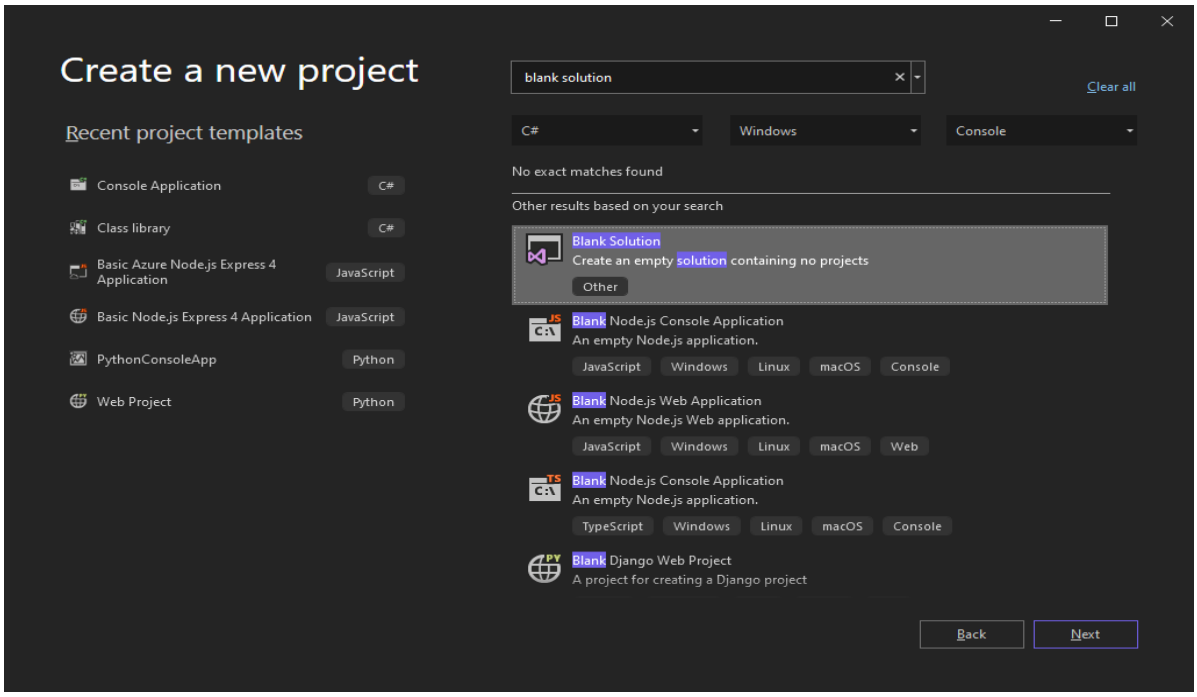
- The central editor window, where you'll probably spend most of your time, displays file contents. In the editor window, you can edit code or design a user interface such as a window with buttons and text boxes.

1.3 Solution and Project

- We are used Solution Explorer to manage project. Using Solution Explorer we should manage folder structure and navigate different files.
- Solution Explorer have a file which is contain solution details and projects details.
- This file have *'.sln'* extension.
- Solution contain one or more projects.
- Project have one file which containe details about project.
- Which have *'.csproj'* extension(if project base on C#).

1.3.1 Create a solution:

- Open Visual Studio, and on the start window, select Create a new project.



- On the Configure your new project page, give the name of Solution, and then select Create.

- Let say our solution name is **'QuickSolution'**.

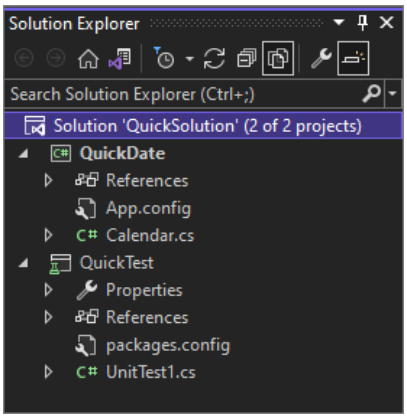
1.3.2 ADD A PROJECT:

- Right-click Solution **'QuickSolution'** in Solution Explorer, and select Add > New Project from the context menu.
- On the Add a new project page, type empty into the search box at the top, and select C# under All languages.
- Select the C# Empty Project (.NET Framework) template, and then select Next.
- On the Configure your new project page, give the name of Project, and then select Create.
- Let say our solution name is **'QuickDate'**.

1.3.3 ADD AN ITEM TO THE PROJECT:

- From the right-click or context menu of the **'QuickDate'** project in Solution Explorer, select Add > New Item.
- Expand Visual C# Items, and then select Code. In the middle pane, select the Class item template. Under Name, give the name of file, and then select Add.
- Let say our file name is **'Calander.cs'**.

1.3.4 ADD A SECOND PROJECT:

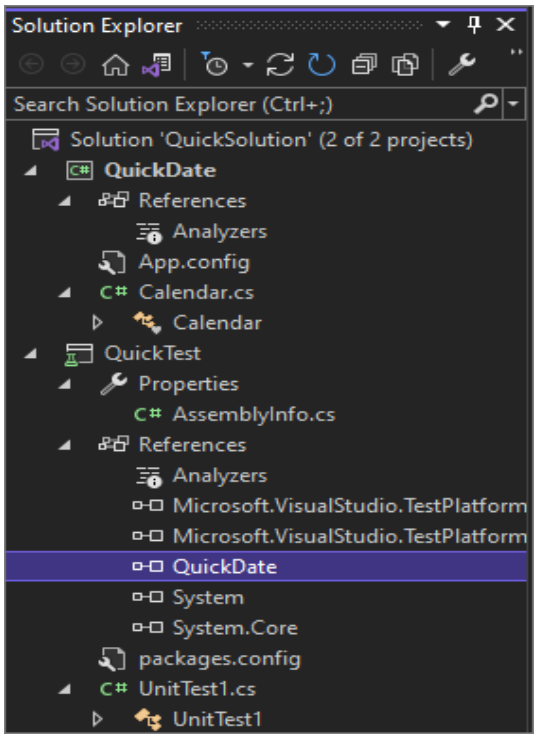


- From the right-click or context menu of Solution **'QuickSolution'** in Solution Explorer, select Add > New Project.
- In the Add a new project dialog box, type unit test into the search box at the top, and then select C# under All languages.
- Select the C# Unit Test Project (.NET Framework) project template, and then select Next.
- On the Configure your new project page, give the name of project, and then select Create.
- Let say our second project name is **'QuickTest'.**
- Add one file inside the second project.
- Let say our file name is **'UnitTest1.cs'**.

1.3.5 ADD A PROJECT REFERENCE:

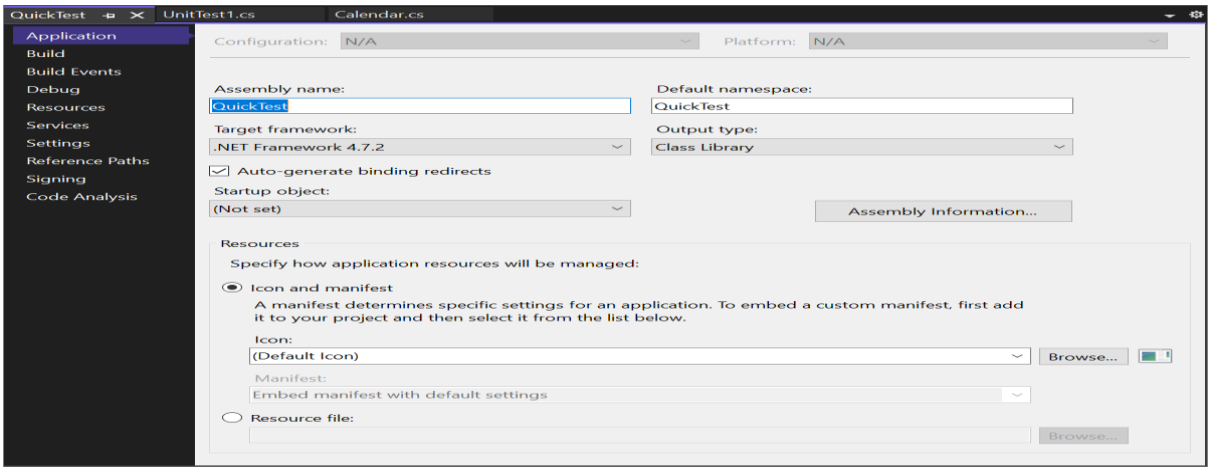
- In Solution Explorer, right-click the References node of the **'QuickTest'** project, and select Add Reference from the context menu.
- In the Reference Manager dialog box, under Projects, select the checkbox next to **'QuickDate',** and then select OK.

- Now a reference to the **'QuickDate'** project appears under the **'QuickTest'** project in Solution Explorer.



1.3.6 PROJECT PROPERTIES:

- In Solution Explorer, right-click the **'QuickTest'** project and select Properties, or select the project and press Alt+Enter.



1.4 CODE EDITOR FEATURES

- Some popular features in Visual Studio that improve your productivity when developing software include

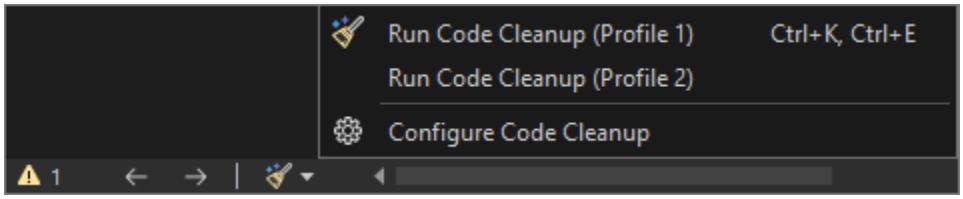
1.4.1 Squiggles and Quick Actions:

- Squiggles are wavy underlines that alert you to errors or potential problems in your code as you type.
- These visual clues help you fix problems immediately, without waiting to discover errors during build or runtime.

- If you hover over a squiggle, you see more information about the error.
- A lightbulb might also appear in the left margin showing Quick Actions you can take to fix the error.

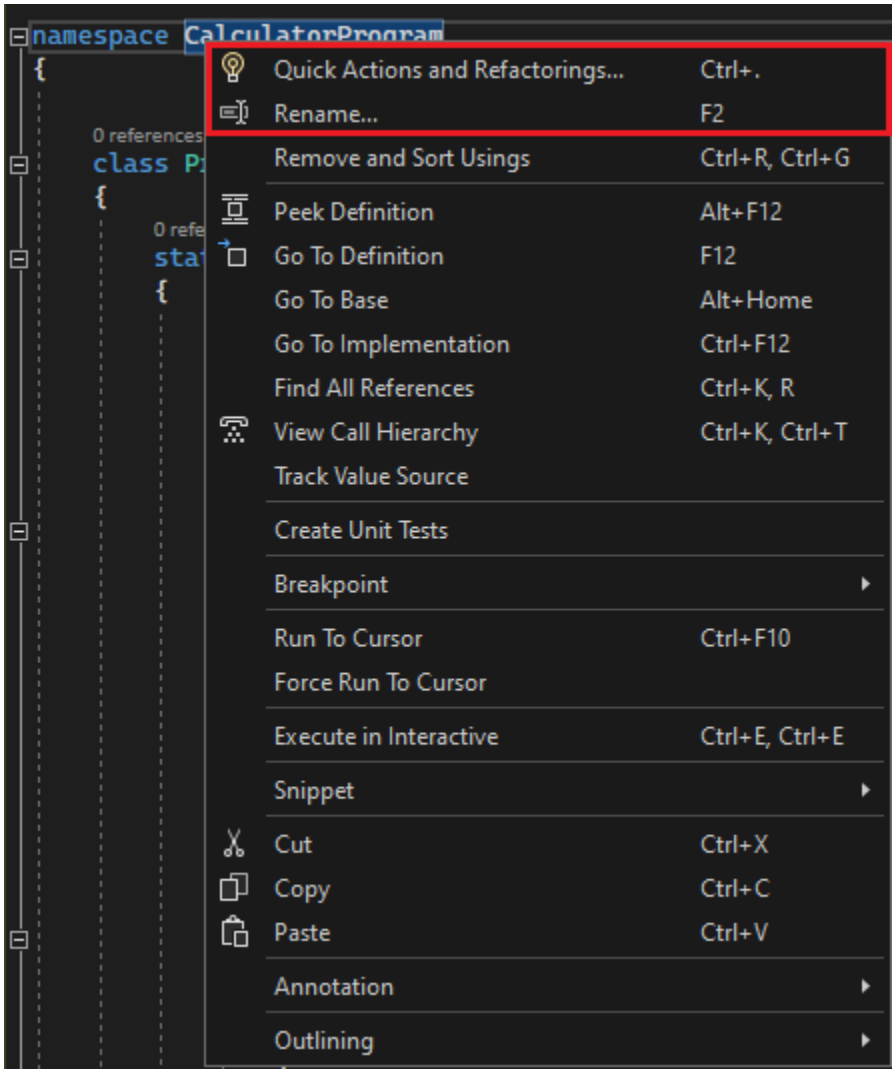
1.4.2 Code Cleanup:

- With the click of a button, you can format your code and apply any code fixes suggested by your code style settings, .editorconfig conventions, and Roslyn analyzers.
- Code Cleanup, currently available for C# code only, helps you resolve issues in your code before it goes to code review.



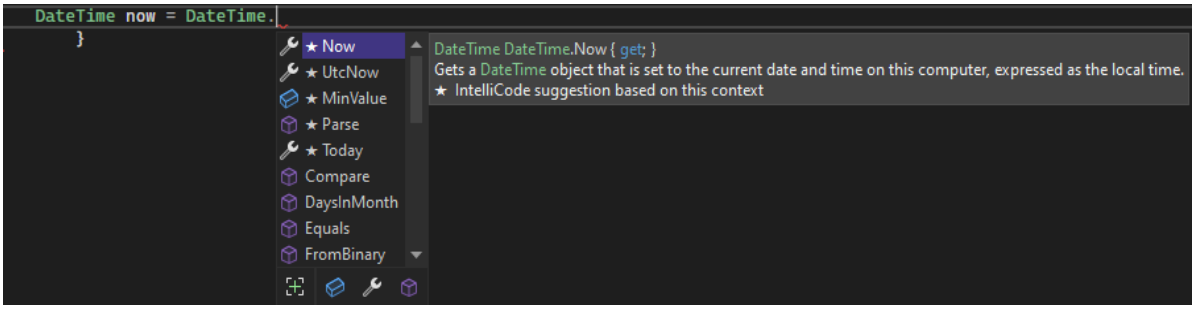
1.4.3 Refactoring:

- Refactoring includes operations such as intelligent renaming of variables, extracting one or more lines of code into a new method, and changing the order of method parameters.



1.4.4 IntelliSense:

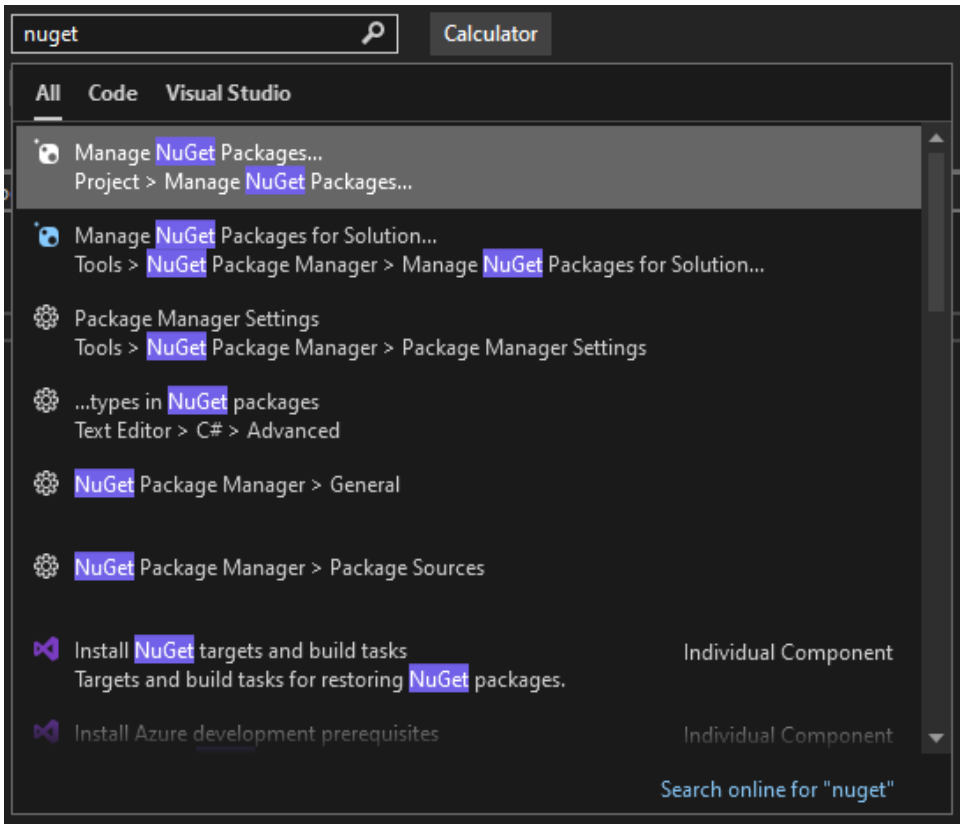
- IntelliSense is a set of features that display information about your code directly in the edor and, in some cases, write small bits of code for you.
- It's like having basic documentation inline in the editor, so you don't have to look up type



information elsewhere.

1.4.5 Visual Studio search:

- Visual Studio menus, options, and properties can seem overwhelming at times.
- Visual Studio search, or *'Ctrl+Q'*, is a great way to rapidly find IDE features and code in one place.

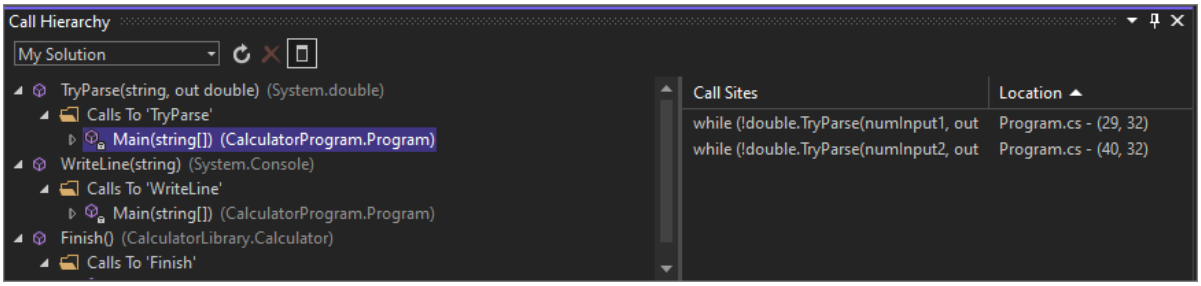


1.4.6 Live Share:

- Collaboratively edit and debug with others in real time, regardless of your app type or programming language.
- You can instantly and securely share your project. You can also share debugging sessions, terminal instances, localhost web apps, voice calls, and more.

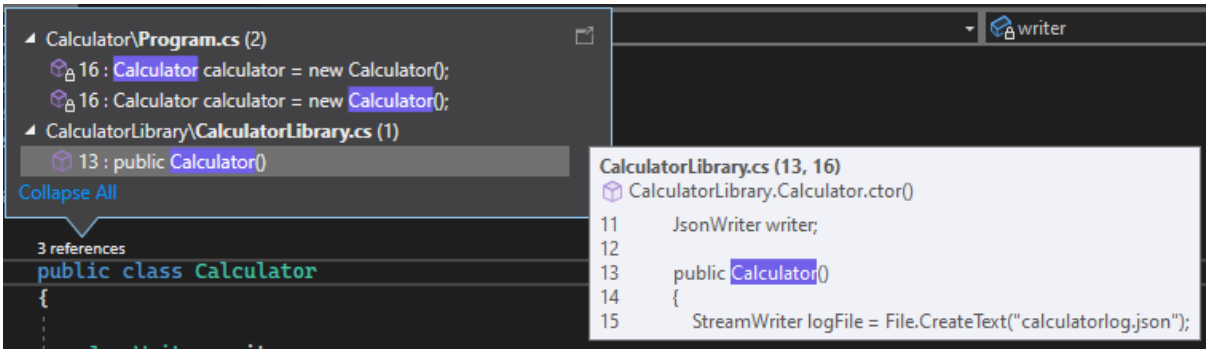
1.4.7 Call Hierarchy:

- The Call Hierarchy window shows the methods that call a selected method.
- This information can be useful when you're thinking about changing or removing the method, or when you're trying to track down a bug.



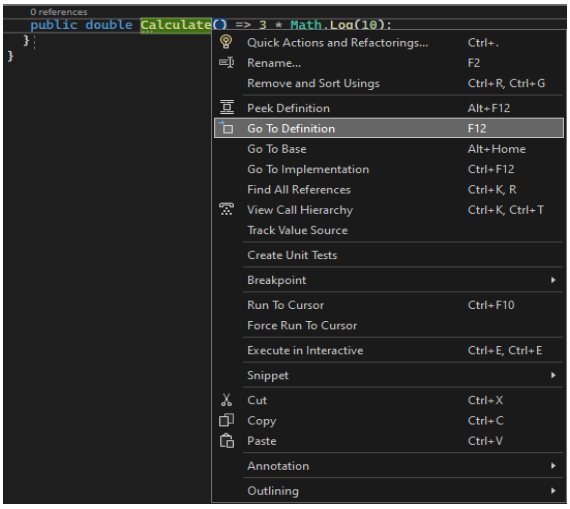
1.4.8 CodeLens:

- CodeLens helps you find code references, code changes, linked bugs, work items, code reviews, and unit tests, without leaving the editor.



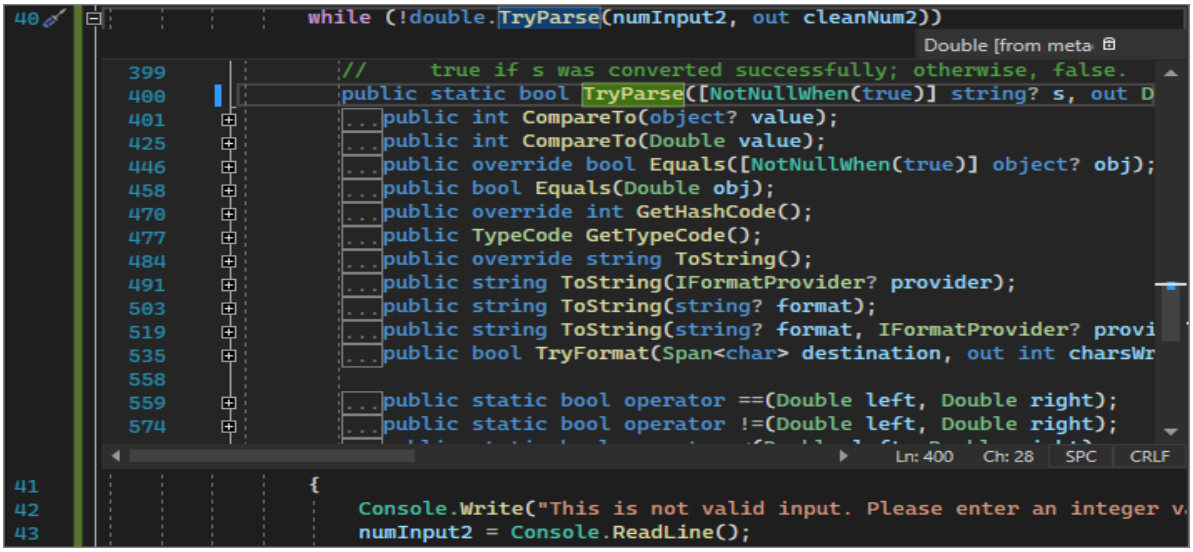
1.4.9 Go To Definition:

- The Go To Definition feature takes you directly to the location of a function or type definition.



1.4.10 Peek Definition:

- The Peek Definition window shows a method or type definition without opening a separate file.



1.5 Popular keyboard shortcuts for Visual Studio:

1.5.1 Build:

Commands	Keyboard shortcuts	Command ID
Build solution	Ctrl+Shift+B	Build.BuildSolution
Cancel	Ctrl+Break	Build.Cancel
Compile	Ctrl+F7	Build.Compile
Run code analysis on solution	Alt+F11	Build.RunCodeAnalysisonSolution

1.5.2 Debug:

Commands	Keyboard shortcuts [Special contexts]	Command ID
Break at function	Ctrl+B	Debug.BreakatFunction
Break all	Ctrl+Alt+Break	Debug.BreakAll

Delete all breakpoints	Ctrl+Shift+F9	Debug.DeleteAllBreakpoints
Exceptions	Ctrl+Alt+E	Debug.Exceptions
Quick watch	Ctrl+Alt+Q or Shift+F9	Debug.QuickWatch
Restart	Ctrl+Shift+F5	Debug.Restart
Run to cursor	Ctrl+F10	Debug.RunToCursor
Set next statement	Ctrl+Shift+F10	Debug.SetNextStatement
Start	F5	Debug.Start
Start without debugging	Ctrl+F5	Debug.StartWithoutDebugging
Step into	F11	Debug.StepInto
Step out	Shift+F11	Debug.StepOut
Step over	F10	Debug.StepOver
Stop debugging	Shift+F5	Debug.StopDebugging
Toggle breakpoint	F9	Debug.ToggleBreakpoint

1.5.3 EDIT:

Commands	**Keyboard shortcuts [Special contexts]**	**Command ID**
Break line	Enter [Text Editor, Report Designer, Windows Forms Designer] or Shift+Enter [Text Editor]	Edit.BreakLine
Collapse to definitions	Ctrl+M, Ctrl+O [Text Editor]	Edit.CollapseToDefinitions

Comment selection	Ctrl+K, Ctrl+C [Text Editor]	Edit.CommentSelection
Complete word	Alt+Right Arrow [Text Editor, Workflow Designer] or Ctrl+Spacebar [Text Editor, Workflow Designer] or Ctrl+K, W [Workflow Designer] or Ctrl+K, Ctrl+W [Workflow Designer]	Edit.CompleteWord
Copy	Ctrl+C or Ctrl+Insert	Edit.Copy
Cut	Ctrl+X or Shift+Delete	Edit.Cut
Delete	Delete [Team Explorer] or Shift+Delete [Sequence Diagram, UML Activity Diagram, Layer Diagram] or Ctrl+Delete [Class Diagram]	Edit.Delete
Find	Ctrl+F	Edit.Find
Find all references	Shift+F12	Edit.FindAllReferences
Find in files	Ctrl+Shift+F	Edit.FindinFiles
Find next	F3	Edit.FindNext
Find next selected	Ctrl+F3	Edit.FindNextSelected
Format document	Ctrl+K, Ctrl+D [Text Editor]	Edit.FormatDocument

Format selection	Ctrl+K, Ctrl+F [Text Editor]	Edit.FormatSelection
Go to	Ctrl+G	Edit.GoTo
Go to declaration	Ctrl+F12	Edit.GoToDeclaration
Go to definition	F12	Edit.GoToDefinition
Go to find combo	Ctrl+D	Edit.GoToFindCombo
Go to next location	F8	Edit.GoToNextLocation
Insert snippet	Ctrl+K, Ctrl+X	Edit.InsertSnippet
Insert tab	Tab [Report Designer, Windows Forms Designer, Text Editor]	Edit.InsertTab
Line cut	Ctrl+L [Text Editor]	Edit.LineCut
Line down extend column	Shift+Alt+Down Arrow [Text Editor]	Edit.LineDownExtendColumn
Line open above	Ctrl+Enter [Text Editor]	Edit.LineOpenAbove
List members	Ctrl+J [Text Editor, Workflow Designer] or Ctrl+K, Ctrl+L [Workflow Designer] or Ctrl+K, L [Workflow Designer]	Edit.ListMembers
Navigate to	Ctrl+,	Edit.NavigateTo
Open file	Ctrl+Shift+G	Edit.OpenFile
Overtype mode	Insert [Text Editor]	Edit.OvertypeMode
Parameter info	Ctrl+Shift+Spacebar [Text Editor, Workflow Designer] or Ctrl+K, Ctrl+P [Workflow Designer] or Ctrl+K, P [Workflow Designer]	Edit.ParameterInfo

Paste	Ctrl+V or Shift+Insert	Edit.Paste
Peek definition	Alt+F12 [Text Editor]	Edit.PeekDefinition
Redo	Ctrl+Y or Shift+Alt+Backspace or Ctrl+Shift+Z	Edit.Redo
Replace	Ctrl+H	Edit.Replace
Select all	Ctrl+A	Edit.SelectAll
Select current word	Ctrl+W [Text Editor]	Edit.SelectCurrentWord
Selection cancel	Esc [Text Editor, Report Designer, Settings Designer, Windows Forms Designer, Managed Resources Editor]	Edit.SelectionCancel
Surround with	Ctrl+K, Ctrl+S (available only in Visual Studio 2019 and earlier)	Edit.SurroundWith
Tab left	Shift+Tab [Text Editor, Report Designer, Windows Forms Editor]	Edit.TabLeft
Toggle all outlining	Ctrl+M, Ctrl+L [Text Editor]	Edit.ToggleAllOutlining
Toggle bookmark	Ctrl+K, Ctrl+K [Text Editor]	Edit.ToggleBookmark
Toggle completion mode	Ctrl+Alt+Space [Text Editor]	Edit.ToggleCompletionMode
Toggle outlining expansion	Ctrl+M, Ctrl+M [Text Editor]	Edit.ToggleOutliningExpansion
Uncomment selection	Ctrl+K, Ctrl+U [Text Editor]	Edit.UncommentSelection
Undo	Ctrl+Z or Alt+Backspace	Edit.Undo
Word delete to end	Ctrl+Delete [Text Editor]	Edit.WordDeleteToEnd

Word delete to start	Ctrl+Backspace [Text Editor]	Edit.WordDeleteToStart

1.5.4 File: popular shortcuts:

Commands	Keyboard shortcuts [Special contexts]	Command ID
Exit	Alt+F4	File.Exit
New file	Ctrl+N	File.NewFile
New project	Ctrl+Shift+N	File.NewProject
New web site	Shift+Alt+N	File.NewWebSite
Open file	Ctrl+O	File.OpenFile
Open project	Ctrl+Shift+O	File.OpenProject
Open web site	Shift+Alt+O	File.OpenWebSite
Rename	F2 [Team Explorer]	File.Rename
Save all	Ctrl+Shift+S	File.SaveAll
Save selected items	Ctrl+S	File.SaveSelectedItems
View in browser	Ctrl+Shift+W	File.ViewinBrowser

1.5.5 Project:

Commands	Keyboard shortcuts [Special contexts]	Command ID
Add existing item	Shift+Alt+A	Project.AddExistingItem
Add new item	Ctrl+Shift+A	Project.AddNewItem

1.5.6 Refactor:

Command	Keyboard shortcut [Special contexts]	Command ID
Extract method	Ctrl+R, Ctrl+M	Refactor.ExtractMethod

1.5.7 TOOLS:

Command	Keyboard shortcut [Special contexts]	Command ID
Attach to process	Ctrl+Alt+P	Tools.AttachtoProcess

1.5.8 VIEW:

Commands	Keyboard shortcuts [Special contexts]	Command ID
Class view	Ctrl+Shift+C	View.ClassView
Edit label	F2	View.EditLabel
Error list	Ctrl+\, Ctrl+E or Ctrl+\, E	View.ErrorList
Navigate backward	Ctrl+-	View.NavigateBackward
Navigate forward	Ctrl+Shift+-	View.NavigateForward
Object browser	Ctrl+Alt+J	View.ObjectBrowser
Output	Ctrl+Alt+O	View.Output
Properties window	F4	View.PropertiesWindow
Refresh	F5 [Team Explorer]	View.Refresh
Server explorer	Ctrl+Alt+S	View.ServerExplorer
Show smart tag	Ctrl+. or Shift+Alt+F10 [HTML Editor Design View]	View.ShowSmartTag
Solution explorer	Ctrl+Alt+L	View.SolutionExplorer
TFS Team Explorer	Ctrl+\, Ctrl+M	View.TfsTeamExplorer
Toolbox	Ctrl+Alt+X	View.Toolbox
View code	Enter [Class Diagram] or F7 [Settings Designer]	View.ViewCode

View designer	Shift+F7 [HTML Editor Source View]	View.ViewDesigner

1.5.9 WINDOW:

Commands	**Keyboard shortcuts [Special contexts]**	**Command ID**
Activate document window	Esc	Window.ActivateDocumentWindow
Close document window	Ctrl+F4	Window.CloseDocumentWindow
Next document window	Ctrl+F6	Window.NextDocumentWindow
Next document window nav	Ctrl+Tab	Window.NextDocumentWindowNav
Next split pane	F6	Window.NextSplitPane

PROJECT TYPES

2.1 WINDOWS DEVELOPMENT

- There are three main application framework available in .Net Framework for developing windows application.
  - Windows Forms
  - Windows Presentation Foundation (WPF)
  - Universal Windows Platform (UWP)

2.1.1 WINDOWS FORMS:

- The first version of Windows Forms was released in 2002 at the same time as .NET framework 1.0.
- The code was written in an event-driven manner.
- Here application contain multiple windows, called as a forms.
- Business logic of application spread across the many event handlers in multiple forms.
- So it's difficult to manage large application.
- So for the avoid this issue we are used MVP (model-view-presenter) design pattern architecture.
- But, all of this makes Windows Forms not very suitable for creating new applications.

2.1.2 WINDOWS PRESENTATION FOUNDATION (WPF):

- Windows Presentation Foundation (WPF) was released as a part of .NET framework 3.5 in 2007.
- It is saved as an XML file using a special syntax named XAML (Extensible Application Markup Language).
- Unlike the Windows forms, this XML file is much easier to understand and edit manually.
- Also, the synchronization between the designer and the XML file is bidirectional.
- Any changes made directly to the XML file are immediately visible in the designer.
- This allows for greater flexibility when editing the layout.
- But, the code is still event driven.
- But, here the data property and event handler can be bound to control using XAML markup.
- So taking advantage of this and introduce the MVVM (model-view-viewmodel) design pattern.

- Even today, WPF is the most versatile and flexible framework for creating Windows desktop applications and as such the recommended choice for most new Windows desktop applications.

2.1.3 Universal Windows Platform (UWP):

- The origin of Universal Windows Platform (UWP) can be traced back to the release of Windows 8 in 2012.
- This is used to accompanying framework for development of touch-first applications, called Metro applications.
- With the release of Windows 10 in 2015, the framework got its final name and eventually supported development of applications like…
    - Windows desktop
    - Windows Mobile
    - Windows IoT Core
    - Windows Mixed Reality
    - Xbox Ones
- It is very similar to WPF.
- User interfaces are save as a XAML files.
- It used MVVM design pattern.
- UWP applications can call some Win32 APIs when their code is written in C++/CX.
- Windows API and WinAPI, Win32 is the main set of Microsoft Windows APIs used for developing 32-bit applications.
- These APIs are responsible for functions in the following categories:
    - Administration
    - Management – Install
    - Configure
    - Service applications or systems
- UWP applications are your only choice if you want to target any non-desktop Windows devices.
- You might also prefer them over WPF for Windows desktop applications if you want to target other Windows devices with the same application or want to publish your application in Microsoft Store as long as you don't need any Win32 APIs not available to you in UWP applications.

2.2 Class Library

- Class library is a type of project in .Net which is help use to manage application in different types of modules.

- It will also used to share code across the multiple projects.
- There are three types of class libraries that you can use:
  - Platform-specific class libraries
  - Portable class library
  - .Net Standard class library

2.2.1 Platform-specific class libraries:

- It have access to all the APIs in a given platform (for example, .NET Framework on Windows, Xamarin iOS), but can only be used by apps and libraries that target that platform.
- Platform-specific libraries are bound to a single .NET platform and can therefore take significant dependencies on a known execution environment.
- Platform-specific libraries have been the primary class library type for the .NET Framework.

2.2.2 Portable class libraries:

- It have access to a subset of APIs, and can be used by apps and libraries that target multiple platforms.

2.2.3 .NET Standard class libraries:

- It is a combination of the platform-specific and portable library concept into a single model that provides the best of both.
- .NET Standard exposes a set of library contracts.
- .NET implementations must support each contract fully or not at all.
- Each implementation supports a set of .NET Standard contracts.
- .NET Standard class library is supported on the platforms that support its contract dependencies.
- These libraries do expose many more APIs than Portable Class Libraries.
- The following implementations support .NET Standard libraries:
  - .NET Core
  - .NET Framework
  - Mono
  - Universal Windows Platform (UWP)

2.3 Mobile Development

- In .Net we have two mobile development frameworks:
  - Xamarin

- .Net MAUI

2.3.1 Xamarin:

- Xamarin extends the .NET developer platform with tools and libraries specifically for building apps for Android, iOS, tvOS, watchOS, macOS, and Windows.
- Xamarin apps are native apps! Whether you're designing a uniform UI across platforms.

2.3.2 .Net MAUI:

- .NET Multi-platform App UI (.NET MAUI) is a framework for building modern, multi-platform, natively compiled iOS, Android, macOS, and Windows apps using C# and XAML in a single codebase.
- This is support in .Net 6 and above.
- We can also use .Net Blazor Native, Which is help use to develop hybrid applications with C# instead of JavaScript.
- Also you can share youre Blazor web components directly in .NET MAUI apps while having access to native device capabilities and packaging.

2.4 Web Development

- ASP.NET offers many frameworks for creating web applications:
  - Web Forms
  - ASP.NET MVC
  - ASP.NET Web Pages
- Many other new framework launched .Net like Blazor, .Net MAUI, Microservices.

2.4.1 Web Forms:

- With ASP.NET Web Forms, we can build dynamic websites using a familiar drag-and-drop, event-driven model.
- This is only available in .Net Framework.

2.4.2 ASP.Net MVC:

- This is used to create powerful web application based on MVC design pattern with full control and using agile methodology.
- It is available in both .Net Framework and .Net Core.

2.4.3 ASP.Net Web Pages:

- ASP.NET Web Pages and the Razor syntax provide a fast, approachable, and lightweight way to combine server code with HTML to create dynamic web content.

- Connect to databases, add video, link to social networking sites, and include many more features that help you create beautiful sites that conform to the latest web standards.
- It is also available in both .Net Framework and .Net Core.

2.4.4 ASP.NET WEB API:

- It is used to create HTTP services that reach a broad range of clients, including browsers and mobile devices.
- It is available in both .Net Framework and .Net Core.

2.4.5 BLAZOR WEBASSEMBLY:

- It is used to develop single page client side web application with the help of Html, Css and C#.
- Because it's real .NET running on WebAssembly, you can re-use code and libraries from server-side parts of your application.
- It is only available in .Net Core.

2.4.6 BLAZOR SERVER:

- It is used to develop real time web application with the SignalR concept.
- It is only available in .Net Core.

INTRODUCTION TO C#

3.1 WHAT IS A C#?

- C# is a general purpose object oriented programing language.
- It developed by Microsoft.
- C# was developed by Anders Hejlsberg and his team during the development of .Net
- Framework.
- C# is designed for Common Language Infrastructure (CLI), which consists of the executable code and runtime environment that allows use of various high-level languages on different computer platforms and architectures.

3.1.1 FEATURES OF C#:

- It is a modern, general-purpose programming language
- It is object oriented.
- It is component oriented.
- It is easy to learn.
- It is a structured language.
- It produces efficient programs.
- It can be compiled on a variety of computer platforms.
- It is a part of .Net Technology.

3.2 "HELLO WORLD" PROGRAM

- C# Contain following part in a program:
  - Namespace declaration
  - A class
  - Class methods
  - Class attributes or properties
  - A Main Method
  - Statements and Expressions
  - Comments
- C# is case sensitive.
- All statements and expression must end with a semicolon (;).
- The program execution starts at the Main method.
- Unlike Java, program file name could be different from the class name.

```
using System;

namespace HelloWorldAPP
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

```
Output : Hello World
```

3.3 NAMESPACE:

- Namespace is a collection of classes.
- It helps to archive level of separation of classes.
- Namespace have a following type as a member
    - Namespace (Nested)
    - Classes
    - Interfaces
    - Structures
    - Delegates
    - Enum
- It is not **mandatory to declare in program**, but they do play an important role to manage large project.
- Namespace also solve a naming conflict problem, we can put two same class name inside different namespace.
- We have following types of namespace declaration formats
    - Block level namespace
    - File Scoped namespace
    - Nested namespce

3.3.1 BLOCK LEVEL NAMESPACE:

- In this namespace have scope inside two parentheses.

```
namespace Namespace1
{
```

```
    class Program
    {
        //Code
    }
}
```

3.3.2 File scoped level namespace:

- It have file level scope.
- It support C# 10 or above.

```
namespace Namespace;
class Program
{
    //Code
}
```

3.3.3 Nested namespace:

- Namespace inside namespace is called nested namespace.

```
namespace OuterNamespace
{
    class Program
    {
        //Code
    }
    namespace InnerNamespace
    {
        class InnerClass
        {
            //Code
        }
    }

}
```

3.4 The "using" Keyword

- If we want to import some namespace inside the program, then we use **"using"** keyword.
- For Ex : We want to print something, so We are use "Console.Writeline();"
- Here is the "Console" is the class which is contain "System" namespace.

- So we need to import “System” namespace inside the Program.
- So using “using” keyword we can import “System” namespace inside the Program.

```
using System;

namespace HelloWorldAPP
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

3.5 CLASS

- Class is a user define type which is describe how certain object look.
- In C# we are use “class” key word to define any class.
- Class have a property and behavior.
- Class contain following members
  - methods
  - variables
- Structure of class:

```
[Access Specifier] class [ClassName]
{
    //Code
}
```

3.5.1 NAMING CONVENTIONS OF CLASS:

- It should begin with an alphabet.
- It should being Pascal casing.
- There may be more than one alphabet, but without any spaces between them.
- Digits may be used but only after alphabet.
- No special symbol can be used except the underscore (_) and currency ($) symbol. When multiple words are needed, an underscore should separate them.
- No keywords or command can be used as a variable name.

3.5.2 Access Specifiers:

- In a C# we have a following assess specifiers

Caller's location	public	protected internal	protected	internal	private protected	private
Within the class	✔	✔	✔	✔	✔	✔
Derived class (same assembly)	✔	✔	✔	✔	✔	✘
Non-derived class (same assembly)	✔	✔	✘	✔	✘	✘
Derived class (different assembly)	✔	✔	✔	✘	✘	✘
Non-derived class (different assembly)	✔	✘	✘	✘	✘	✘

3.5.3 Nested Classes:

- Class inside the class called nested class.

```
namespace HelloWorldAPP
{
    class Program
    {
        class InnerClass
        {
            //Code
        }

        //Code
    }
}
```

3.6 Methods

- Method is a block code which is contain a series of statement.
- Method declare inside class, struct or interface.

```
[Access Specifier] [return datatype] [Method Name]([datatype of args] args)
{
    //Code
    return [return member]

}
```

3.7 VARIABLES

- It is used to represent storage locations.

Type	Example
Integral types	sbyte, byte, short, ushort, int, uint, long, ulong, and char
Floating point types	float and double
Decimal types	decimal
Boolean types	true or false values, as assigned
Nullable types	Nullable data types

```
[datatype] [name of variable]
```

- For Ex:

```
class A
{
    public static int x;
    int y;

    void F(int[] v, int a, ref int b, out int c)
    {
        int i = 1;
        c = a + b++;
    }
}
```

3.8 DATA TYPE

- It specify type of data.
- In C# we have following types of datatypes
    - Value Data Type

- Reference Data Type
- Pointer Data Type

3.8.1 Value Data Type:

- It directly store value in a memory.
- It accepts both signed and unsigned literals.
- It belongs to **System.ValueType** namespace.
- Signed and Unsigned Integer Data Type…

Alias	Type Name	Type	Size(bits)	Range	Default Value
sbyte	System.Sbyte	signed integer	8	-128 to 127	0
short	System.Int16	signed integer	16	-32768 to 32767	0
Int	System.Int32	signed integer	32	-2^{31} to $2^{31}-1$	0
long	System.Int64	signed integer	64	-2^{63} to $2^{63}-1$	0L
byte	System.byte	unsigned integer	8	0 to 255	0
ushort	System.UInt16	unsigned integer	16	0 to 65535	0
uint	System.UInt32	unsigned integer	32	0 to 2^{32}	0
ulong	System.UInt64	unsigned integer	64	0 to 2^{63}	0

- Floating Point Type…

Alias	Type name	Size(bits)	Range (aprox)	Default Value
float	System.Single	32	$\pm1.5 \times 10^{-45}$ to $\pm3.4 \times 10^{38}$	0.0F
double	System.Double	64	$\pm5.0 \times 10^{-324}$ to $\pm1.7 \times 10^{308}$	0.0D

- Character Type…

Alias	Type name	Size In(Bits)	Range	Default value
char	System.Char	16	U +0000 to U +ffff	'\0'

- Boolean Type…

Alias	Type name	Values
bool	System.Boolean	True / False

3.8.2 Reference Data Type:

- The reference data type will contain memory address of data because reference don't store value of data directly.
- The built in reference type are…
  - String
  - Object
- String…
- It represents a sequence of Unicode characters and its type name is **System.String**. So, string and String are equivalent.

```
string s1 = "Dhruvil"; // creating through string keyword
String s1 = "Dobariya"; // creating through String class
```

- Object…
- In C#, all types, predefined and user-defined, reference types and value types, inherit directly or indirectly from Object.
- So basically it is the base class for all the data types in C#.
- Before assigning values, it needs type conversion.
- When a variable of a value type is converted to object, it's called boxing. When a variable of type object is converted to a value type, it's called unboxing.
- Its type name is **System.Object**.

3.8.3 Pointer Datatype:

- The Pointer Data Types will contain a memory address of the variable value.
- To get the pointer details we have a two symbols
  - ampersand (&)
  - asterisk (*).
- Ampersand (&): It is Known as Address Operator. It is used to determine the address of a variable.

- Asterisk (*): It also known as Indirection Operator. It is used to access the value of an address.
- Syntax…

```
type* identifier;
```

- Example:

```
int* p1, p;   // Valid syntax
int *p1, *p;   // Invalid
```

- Program:

```
// Error: Unsafe code requires the `unsafe'
// command line option to be specified
// For its solution:
// Go to your project properties page and
// check under Build the checkbox Allow
// unsafe code.
using System;
namespace Pointerprogram
{
    class GFG {
        static void Main()
        {
            unsafe
            {

                // declare variable
                int n = 10;

                // store variable n address
                // location in pointer variable p
                int* p = &n;
                Console.WriteLine("Value :{0}", n);
                Console.WriteLine("Address :{0}", (int)p);
            }
        }
    }
}
```

UNDERSTANDING C# PROGRAM

4.1 PROGRAM FLOW

- Let us understand the flow of hello world program.

```
using System;

namespace HelloWorldAPP
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

- .Net compiler platform also known as Roslyn.
- It is open source compiler and code analysis API for C# and VB.
- Compiler enter in program from **Main** method.
- Compiler follow code execution process.
- We can use different way to compile program.
    - Using Visual Studio
    - Using Command Line
        - For Ex : **csc Program.cs** and enter (if unsafe code in program then use **csc /unsafe Program.cs**)
        - Then **Program** enter
- Program compile using CLR.

4.2 UNDERSTANDING SYNTAX

- First of all we import some namespace using “using” keyword.
- Then we define namespace of program
- Then we should define class of program which contain **Main** method.
- Then inside class we should define **Main** method.
- **Main** method must static.
- Under the **Main** method we should write program.
- **Main** method return nothing and take array of string as an argument.

4.3 Common Language Runtime (CLR)

- It is a runtime environment.
- It manages life cycle and execution of .Net.
- It also provide easy service for deployment.
- Developer should write program in any language like C#, VB or F#, It convert into intermediate language.
-