

# Chapter 1: Introduction to C# programming language

Nguyen Tien Dong

- 1.1. Introduction to C# programming language
- 1.2. Introduction to .Net Framework
- 1.3. Introduction to .Net Core
- 1.4. Introducing and Installing Visual Studio
- 1.5. How the program works
- 1.6. Rules for writing programs
- 1.7. Basic C# Libraries
- 1.8. Basic I/O commands
- 1.9. Data types
- 1.10. Math operations and expressions

## Section 1:

- 1.1. Introduction to C# programming language
- 1.2. Introduction to .Net Framework
- 1.3. Introduction to .Net Core
- 1.4. Introducing and Installing Visual Studio

## History:

- Developed since 2000 by Anders Hejlsberg at Microsoft.
- C# 1.0 version came out to the public in 2002.
- C# version 10.0, announced in 2022, brings many improvements and new features.

1. **C# is compiled language** (not interpreted language)
2. C# is a powerful, object-oriented programming language developed by Microsoft.
3. C# is the primary programming language for the **.NET platform**.
4. C# supports many programming paradigms, **including object-oriented programming (OOP)**, functional programming, and service-oriented programming.
5. C# provides a clear and understandable structure that makes it easier to write, read, and maintain source code.
6. **C# is also widely used** in game development (with Unity), mobile applications (with Xamarin) and web applications (with ASP.NET).



# 1.2 Introduction to .Net Framework

- **The .NET Framework 1.0**, released in 2002, was the first platform for C#.
- **Developed and run-on Windows platform only**
- It supports ASP.NET Web Forms, WinForms, WCF, Silverlight, WPF, LINQ, ADO.NET Entity Framework, Parallel LINQ, Task Parallel Library, etc
- The .NET Framework 4.8 version was released on April 18, 2019

.NET Framework 1.0

.NET Framework 1.1

.NET Framework 2.0

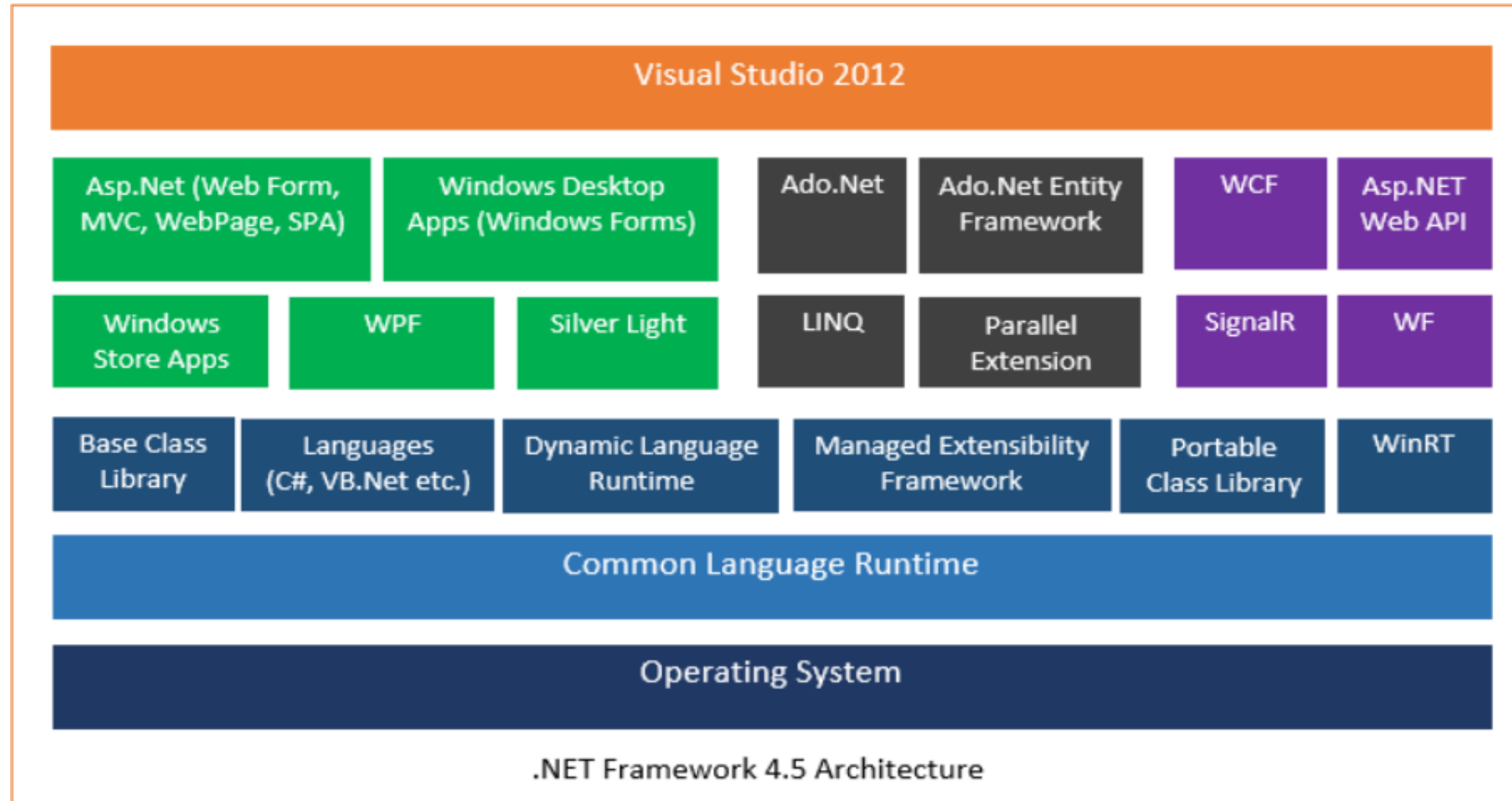
.NET Framework 3.0

.NET Framework 3.5

.NET Framework 4.0

.NET Framework 4.5

# 1.2 Introduction to .Net Framework

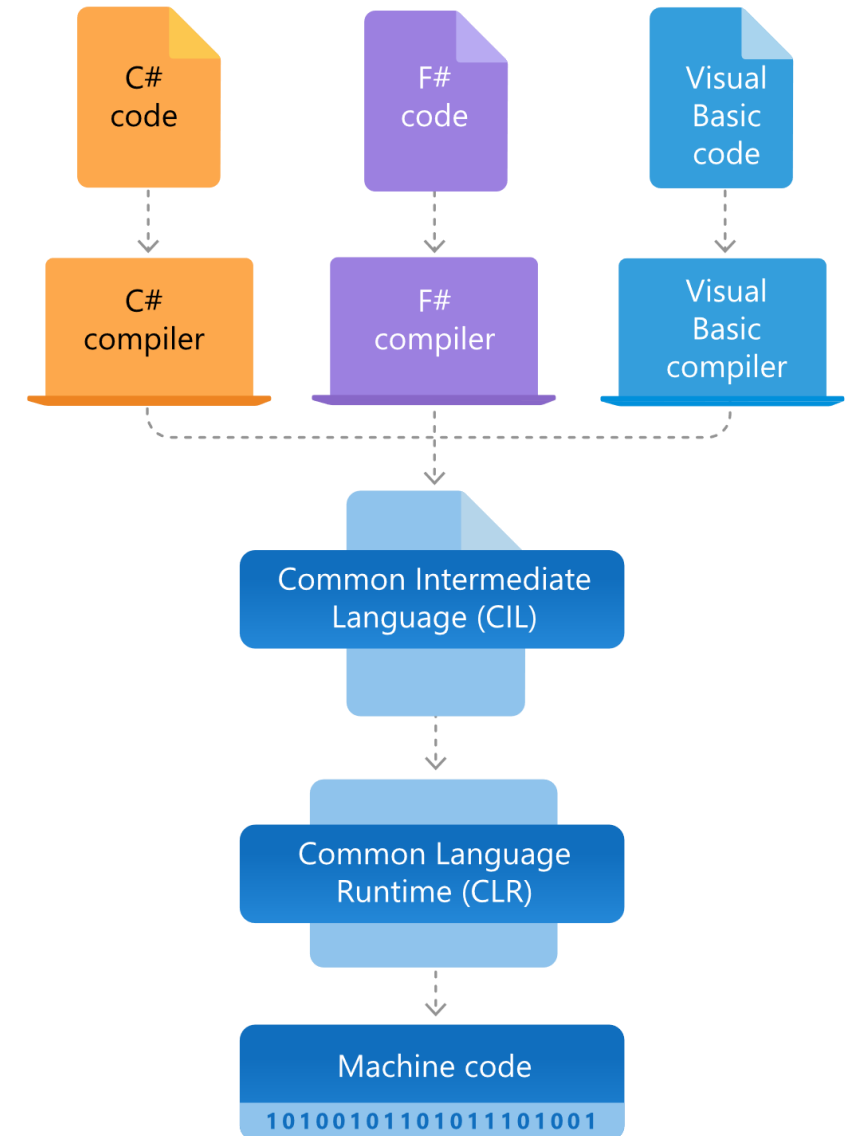


- .NET Framework is a software development framework for **building and running applications on Windows**.
- **.NET Framework is part of the .NET platform**, a collection of technologies for building apps for Linux, macOS, Windows, iOS, Android, and more.

**Architecture of Net framework:** The two core components of the .NET Framework integral to any application or service development are:

- **Common Language Runtime (CLR)**
- **NET Framework Class Library (FCL)**

(Source: [Link](#))





- ◆ The two core components of the .NET Framework integral to any application or service development are:



- .NET applications are written in the C#, F#, or Visual Basic programming language. Code is compiled into a language-agnostic Common Intermediate Language (CIL). Compiled code is stored in assemblies—files with a .dll or .exe file extension.
- When an app runs, the CLR takes the assembly and uses a just-in-time compiler (JIT) to turn it into machine code that can execute on the specific architecture of the computer it is running on.

## Common Language Runtime(CLR)

- ◆ A common runtime for all .NET languages
  - Common type system
  - Common metadata
  - Intermediate Language (IL) to native code compilers
  - Memory allocation and garbage collection
  - Code execution and security
- ◆ **Over 20 languages supported today**
  - C#, VB, Jscript, Visual C++ from Microsoft
  - Perl, Python, Smalltalk, Cobol, Haskell, Mercury, Eiffel, Oberon, Oz, Pascal, APL, CAML, Scheme, etc.

## Common Language Runtime(CLR)

- ◆ A common runtime for all .NET languages
  - Common type system
  - Common metadata
  - Intermediate Language (IL) to native code compilers
  - Memory allocation and garbage collection
  - Code execution and security
- ◆ Over 20 languages supported today
  - C#, VB, Jscript, Visual C++ from Microsoft
  - Perl, Python, Smalltalk, Cobol, Haskell, Mercury, Eiffel, Oberon, Oz, Pascal, APL, CAML, Scheme, etc.

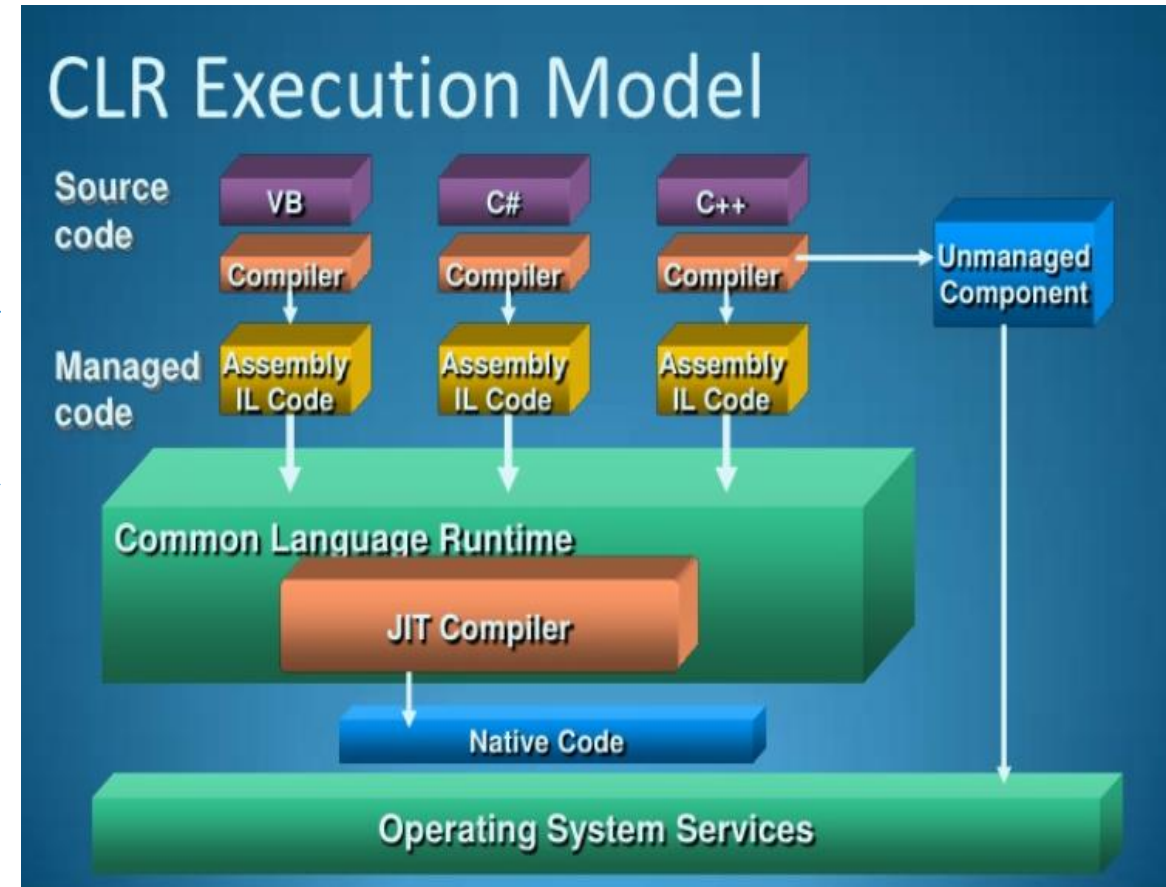
## Common Language Runtime(CLR)

### ◆ Using the .NET Framework:

- The code of a program is compiled into CIL (formerly called MSIL) and stored in a file called assembly
- This assembly is then compiled by the CLR to the native code at run-time

### ◆ In traditional Windows applications:

- Codes were directly compiled into the executable native code of the operating system





## Advantages of CLR

- ◆ Interoperation between managed code and unmanaged code (COM, DLLs)
- ◆ Managed code environment
- ◆ Improved memory handling
- ◆ JIT(**Just-In-Time**) Compiler allows code to run in a protected environment as managed code
- ◆ JIT allows the IL code to be hardware independent
- ◆ CLR also allows for enforcement of code access security
- ◆ Verification of type safety
- ◆ Access to Metadata (enhanced Type Information)

- **.NET Core 1.0** (2016) is the first open-source, **cross-platform** version of .NET.  
    .NET Core 3.1 (2019) is the last LTS (long term support) version of .NET Core.
- **.NET 5.0 (2020)** combines all previous .NET versions (**including .NET Framework, .NET Core, Xamarin**) into a single platform. .NET 6.0 (2021) is the first LTS version of .NET unifying. .NET 7.0 (2022) is the latest version to date.
- **Some famous web applications written in .NET:** Stack Overflow, Microsoft, GoDaddy.

## .NET 5 (.NET) = .NET Core vNext

- Released on November 10, 2020 (Visual Studio 2019 and C# 9.0)



# What is the .NET Standard?

- ◆ .NET Standard is a specification that can be used across all .NET implementations. It is used for developing library projects only. This means if we are creating a **library** in .NET Standard we can use those in .NET Framework and .NET Core
- ◆ To create uniformity means to allow usage in all the .NET implementations. .NET Standard has support for Mono platform, Xamarin, Universal Windows Platform, and Unity

## Comparisons Table

.NET Core	.NET Framework	.NET Standard
For New Application Development.	For Maintenance of Existing Applications only.	For Developing Library Projects only.
Cross-Platform	Windows Only	Cross-Platform
High Performance	Average Performance	-
Open Source <a href="https://github.com/dotnet/core-sdk">https://github.com/dotnet/core-sdk</a>	Private	Open Source <a href="https://github.com/dotnet/standard">https://github.com/dotnet/standard</a>
CoreCLR and CoreFX	CLR and BCL	-
Visual Studio / Visual Studio Code	Visual Studio	Visual Studio / Visual Studio Code
Free	Free	Free



# Benefits of using .NET

- ◆ **Open Source:** Open source and community-oriented on GitHub.
- ◆ **Cross-Platform:** .NET Core can run on Windows, Linux, and macOS
- ◆ **Command-line tools:** Create, build, and run projects from the command line
- ◆ **Modular:** Ships as NuGet packages
- ◆ **Host Agnostic:**
  - .NET Core on the server side is not dependent on IIS and, with two lightweight servers: Kestrel and WebListener
  - It can be self-hosted as a Console application and can be also gelled with mature servers such as IIS, Apache, and others through a reverse proxy option

# Benefits of using .NET

- ◆ Support for leveraging platform-specific capabilities, such as **Windows Forms** and **WPF(Windows Presentation Foundation)** on Windows and the native bindings to each native platform from **Xamarin**
- ◆ High performance
- ◆ Side-by-side installation
- ◆ Small project files (SDK-style)
- ◆ Visual Studio, Visual Studio for Mac, and Visual Studio Code integration

# .NET components

- ◆ **Language compilers:** These turn source code written with languages such as C#, F#, and Visual Basic into intermediate language (IL) code stored in assemblies. NET language compilers for C# and Visual Basic, also known as **Roslyn**
- ◆ **Common Language Runtime (CoreCLR):** This runtime loads assemblies, compiles the IL code stored in them into native code instructions for computer's CPU, and executes the code within an environment that manages resources such as threads and memory
- ◆ **Base Class Libraries (BCLs)** of assemblies in NuGet packages (CoreFX): These are prebuilt assemblies of types packaged and distributed using NuGet for performing common tasks when building applications

# Why C# is selected as develop application?

- ◆ C# was developed by Anders Hejlsberg and his team during the development of .NET
- ◆ C# is a modern, object-oriented, and type-safe programming language. C# enables developers to build many types of secure and robust applications that run in the .NET ecosystem. C# has its roots in the C family of languages and will be immediately familiar to C, C++, Java, and JavaScript programmers
- ◆ 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

# Why C# is selected as develop application?

- ◆ The following reasons make C# a widely used professional language
  - 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

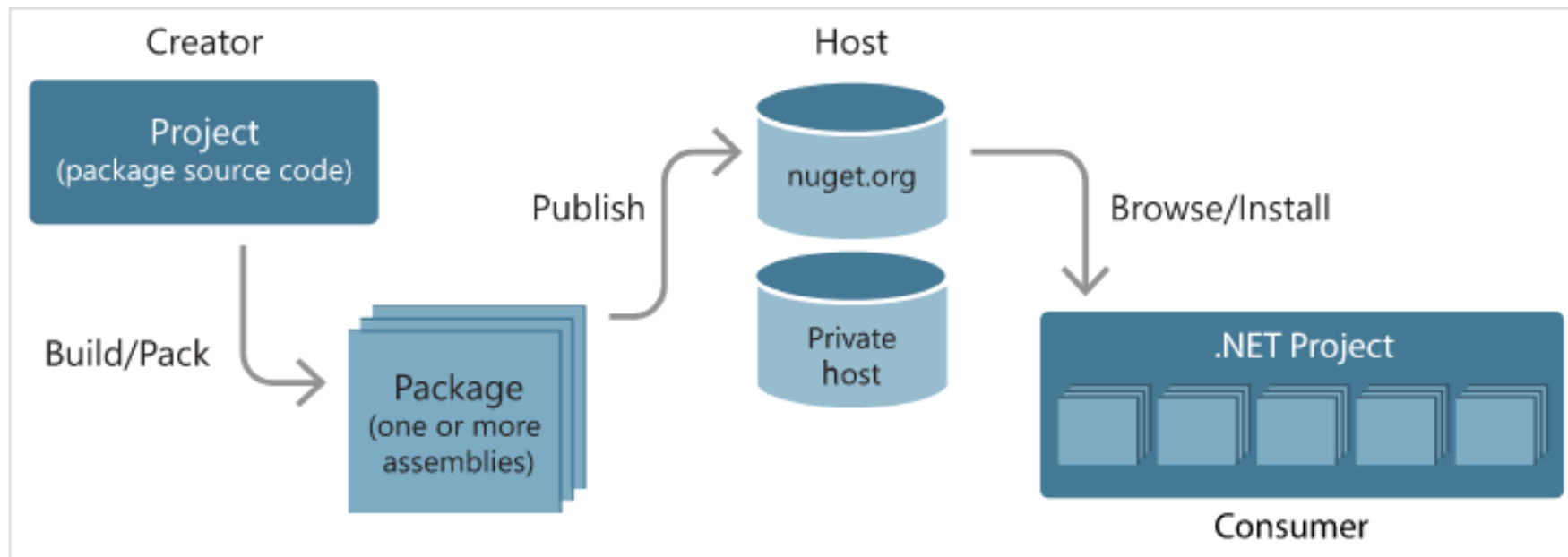
- ◆ More C# features :

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/>



# Introduction to Nuget packages

- ◆ .NET is split into a set of packages, distributed using a Microsoft supported package management technology named NuGet. Each of these packages represents a single assembly of the same name
  - For example, the System.Collections package contains the System.Collections.dll assembly



- Visual Studio is an integrated development environment (IDE) from Microsoft that supports many programming languages, including C#.
- Visual Studio provides many features such as debugger, source code editor with auto-completion, source code version control, etc.
- Visual Studio Installation Instructions: Download and install from Microsoft's homepage, select the necessary components for C# application development.
- Visual Studio provides templates for creating C# applications quickly.
- Visual Studio Code is a lighter version of Visual Studio, works on many operating systems and also supports C# programming.

Installing: [Link](#)

## Meet the Visual Studio family



### Visual Studio |

The most comprehensive IDE for .NET and C++ developers on Windows. Fully packed with a sweet array of tools and features to elevate and enhance every stage of software development.

[Learn more →](#)

[Download Visual Studio](#) ▾



### Visual Studio for Mac |

A comprehensive IDE for .NET developers that's native to macOS. Includes top-notch support for web, cloud, mobile, and game development.

[Learn more →](#)

[Read more about activating your license](#)

[Download Visual Studio  
for Mac](#)



### Visual Studio Code |

A standalone source code editor that runs on Windows, macOS, and Linux. The top pick for JavaScript and web developers, with extensions to support just about any programming language.

[Learn more →](#)

By using Visual Studio Code you agree to its [license](#) & [privacy statement](#)

[Download Visual  
Studio Code](#) ▾

# Summary Section 1

- ◆ Concepts were introduced:
  - Overview about .NET Core, .NET 5(.NET) and .NET Framework
  - Overview .NET Framework and .NET 5(.NET) Architecture
  - Overview of Visual Studio.NET
  - Explain about Cross-platform application with .NET
  - Why .NET Core and C# Language is selected as develop application?
  - Explain and demo using “dotnet CLI” to create C# Console App
  - Overview NuGet package
  - Create and Run cross-platform Console application with C#

Section 2:

1.5. How the C# program works

1.6. Rules for writing programs

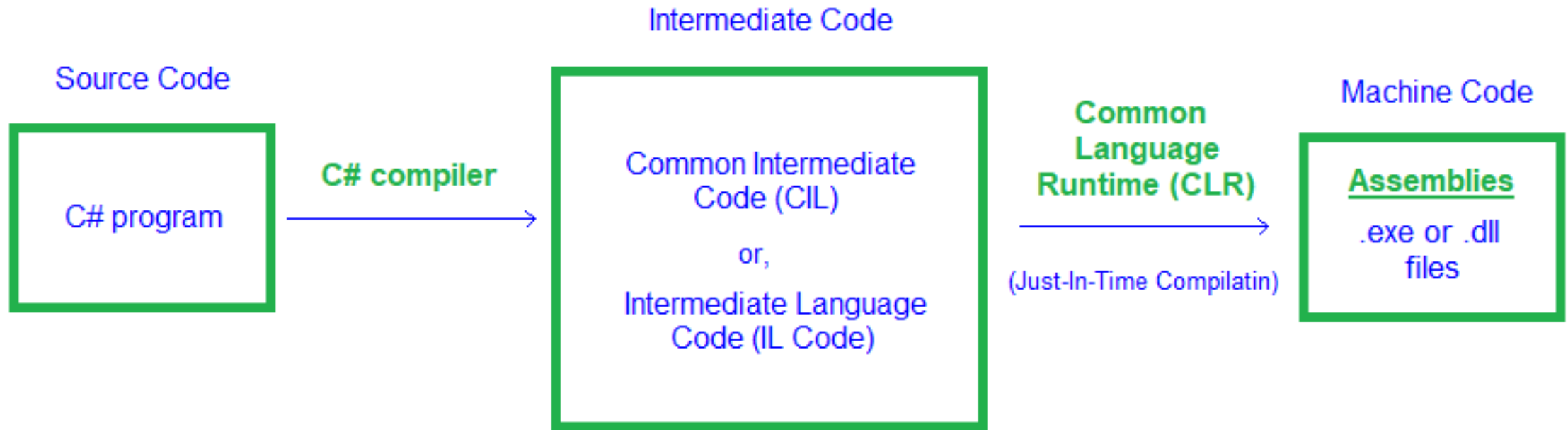
1.7. Basic C# Libraries



# 1.5 How the C# program works

1. The code is written in the C# programming language.
2. The C# source code is compiled into Common Intermediate Language (CIL) by the C# compiler.
3. CLR (Common Language Runtime) at run time compiles CIL to machine code.
4. The machine code is then executed on the operating system.
5. This process ensures that C# is cross-platform.

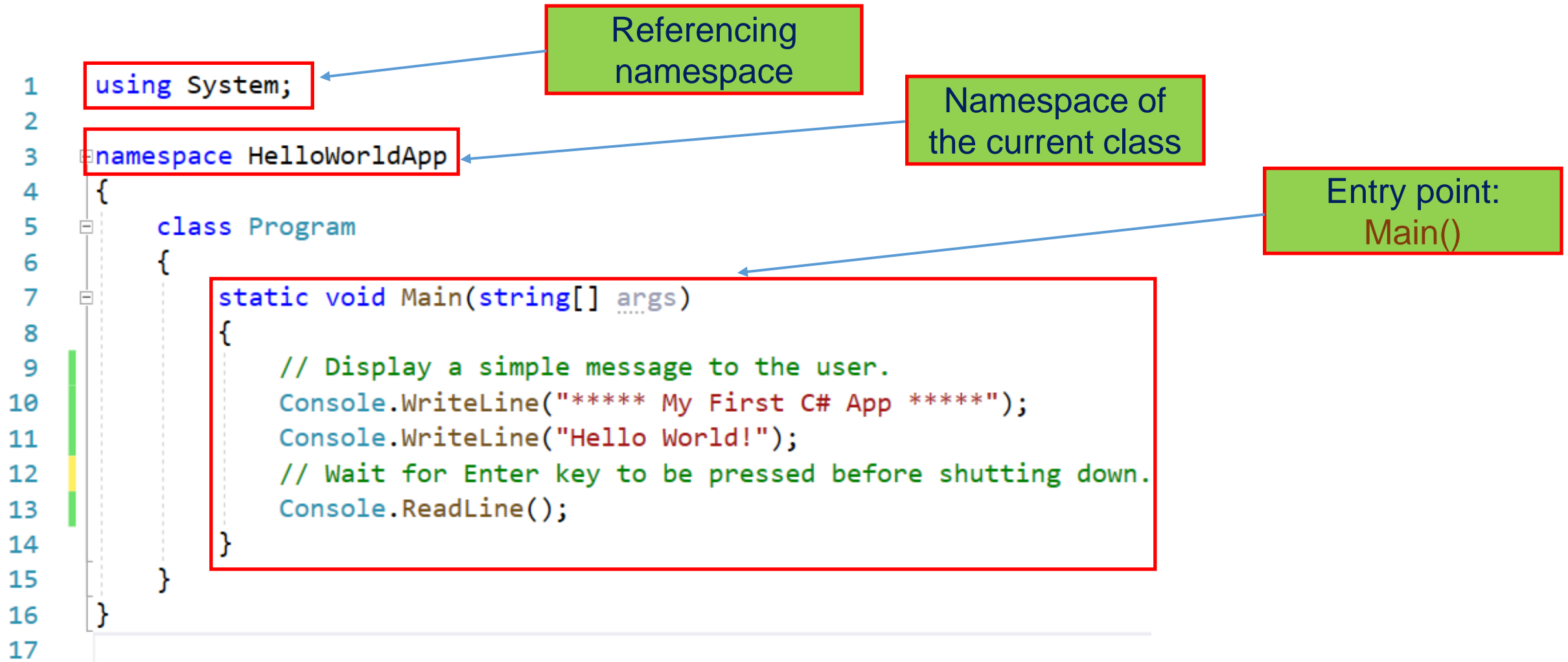
# 1.5 How the C# program works



How C# Code Gets Compiled and Executed

1. Use correct C# syntax and keywords.
2. Classes and methods must be clearly named and properly describe their function.
3. Variables need to be initialized before being used.
4. Use only necessary libraries and namespaces.
5. Always follow the rules of object-oriented programming in C#.

## Structure of a C# program



### **Note the following points:**

- 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.

A C# program consists of the following parts :

- Namespace declaration
- A class
- Class methods
- Class attributes
- A Main method
- Statements and Expressions
- Comments



## Namespaces in C#:

- ◆ Namespaces are used to organize the classes. It helps to control the scope of methods and classes in larger .Net programming projects
- ◆ The biggest advantage of using namespace is that the class names which are declared in one namespace will not clash with the same class names declared in another namespace. It is also referred as named group of classes having common features
- ◆ The members of a namespace can be namespaces, interfaces, structures, and delegates.

## Namespaces in C#:

- ◆ To define a namespace in C#, we will use the namespace keyword followed by the name of the namespace and curly braces containing the body of the namespace as follows:

```
namespace name_of_namespace
{
    // Namespace (Nested Namespaces)
    // Classes
    // Interfaces
    // Structures
    // Delegates
}
```

```
namespace MyNamespace
{
    // MyClass is the class in the namespace MyNamespace
    class MyClass
    {
        // class code
    }
}
```

## Variations on the Main() Method

- ◆ By default, Visual Studio will generate a Main() method that has a void return value and an array of string types as the single input parameter
- ◆ To construct application's entry point using any of the following signatures:

```
static int Main(string[] args){  
    // Must return a value before exiting!  
    return 0;  
}  
// No return type, no parameters.  
static void Main(){  
}  
// int return type, no parameters.  
static int Main(){  
    // Must return a value before exiting!  
    return 0;  
}
```

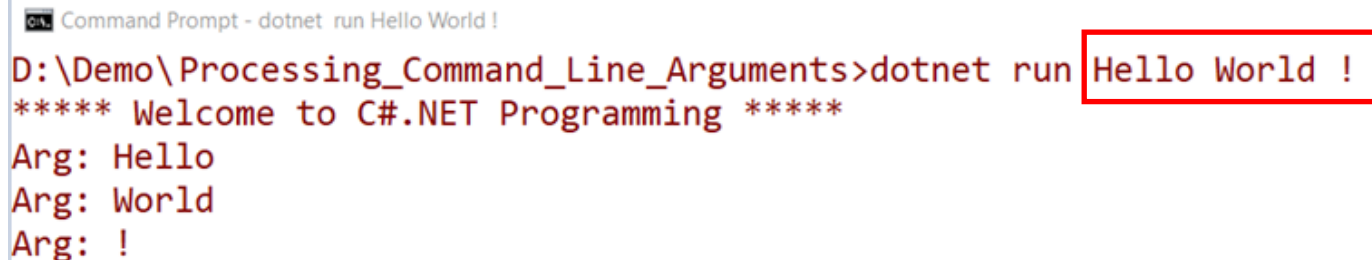
The Main() method can be **asynchronous**

```
static Task Main()  
static Task<int> Main()  
static Task Main(string[])  
static Task<int> Main(string[])
```

## Processing Command-Line Arguments:

- ◆ Create C# Console App as the following and run it by **dotnet** CLI

```
1  using System;
2  namespace Processing_Command_Line_Arguments {
3      class Program{
4          static void Main(string[] args){
5              string msg = "***** Welcome to C#.NET Programming *****";
6              Console.WriteLine("{0}",msg);
7              // Process any incoming args.
8              for (int i = 0; i < args.Length; i++){
9                  Console.WriteLine($"Arg: {args[i]}");
10             }
11             Console.ReadLine();
12         }
13     }
14 }
```



```
Command Prompt - dotnet run Hello World !
D:\Demo\Processing_Command_Line_Arguments>dotnet run Hello World !
***** Welcome to C#.NET Programming *****
Arg: Hello
Arg: World
Arg: !
```

1. `System` - Chứa các class cơ bản và cốt lõi của .NET Framework.
2. `System.Collections.Generic` - Chứa các định nghĩa cho các collections thông dụng như `List`, `Stack`, `Queue`, và `Dictionary`.
3. `System.IO` - Chứa các class cho thao tác với tệp và dòng dữ liệu.
4. `System.Text` - Chứa các class giúp làm việc với các chuỗi và định dạng văn bản.
5. `System.Threading` - Chứa các class để làm việc với đa luồng và đồng bộ hóa.

### System Namespace

1. Contains basic types like Boolean, Double, String, Int32 and DateTime.
2. Contains basic classes like Math, GC (garbage collector), and Random.
3. Contains classes and interfaces related to exception handling, such as Exception and IDisposable.
4. Provide objects like Console to manipulate the system.
5. Provides basic delegates and events.



### System.Collections.Generic Namespace

1. Contains classes like `List<T>`, `LinkedList<T>`, and `Stack<T>` for storing collection data.
2. Provides `Dictionary<TKey, TValue>`, `HashSet<T>` and other data structures based on hashtable.
3. Supports generic programming, allowing the same code to be used with many different data types.
4. Provide `Queue<T>` and `Stack<T>` classes to perform queue and stack operations.
5. These classes allow easy and safe traversing of collections.

## System.IO Namespace

1. Provides classes for manipulating files and data streams (I/O).
2. Includes classes like `FileStream`, `StreamReader`, and `StreamWriter` for reading and writing files.
3. Includes `Directory` and `DirectoryInfo` for directory creation, migration, and browsing.
4. Contains classes and methods for handling files and data streams safely and efficiently.
5. Support handling I/O errors through exception classes like `FileNotFoundException`.

### System.Text Namespace

1. Contains `StringBuilder` - a powerful class for string processing and compilation.
2. Contains encoding classes such as `ASCII`, `Unicode`, `UTF7`, and `UTF8`.
3. Includes tools that make string manipulation and formatting easy.
4. Provides classes and methods for comparing and sorting strings.
5. Support for internationalization through classes like `StringInfo` and `NormalizationForm`.

### System.Threading Namespace

1. Provides classes and methods for multithreading.
2. Contains classes like Thread, ThreadPool, and Timer to manage and control threads.
3. Contains classes like Monitor, Mutex, and Semaphore to synchronize threads.
4. Includes tools to handle race conditions and deadlock situations.
5. Supports asynchronous programming with classes like Task and Task<TResult>.

# Summary Section 2

- ◆ Section 2 were introduced:
  - Rules to code C# Program
  - Structure of C# program, overview of namespaces and main method
  - Overview list of 5 common namespace use in C#

Section 3:

1.8. Basic I/O commands

1.9. Data types

1.10. Math operations and expressions





CMC UNIVERSITY

THANK YOU