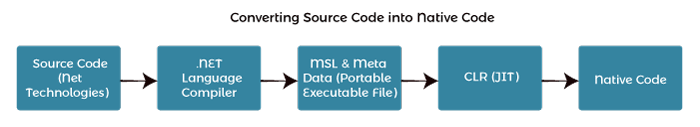
**Common Language Runtime(CLR)**

Major component of .net

Executive engine that handles running applications.

Executive engine: executing the compiling code in a managing environment.

CLR is a run time environment that manages and executes code written in any .Net programming language.



MSIL : Microsoft Intermediate language. Comparable to byte code in java. Platform Independent.

MSIL + Metadata ---> Manifest file.

Metadata : members and types required by CLR to execute the MSIL code.

CLR has JIT compiler which converts the MSIL into Native code(platform dependent).

**Components of CLR**

1. Common Type System

Multiple types of languages can be converted into same Intermediate language using different compiler each type of language. So, this means that when we have the Intermediate language then we can convert into the respective language.

Provides guidlines for declaring,using and managing data types in run time.

Types referred to data type supported by languages

Provides common type system to all the languages.

Supports the object oriented programming concepts. This ensures that object-oriented features work consistently across languages.

Helps in writing language independent code.

CTS categorises types into:

* + Value Types: int i=30; //value allocated by stack
  + Reference types : String i=”Hello”; //memory allocated at heap and address at stack.

1. Common Language Specification

Sub part of CTS

The CLS specifies that only certain types of data types, method signatures, and language features should be used to ensure that code can be consumed by any CLS-compliant language.

Gives language integration to CTS.

Gives the guidelines so that it can be worked with other languages.

Code written in CLS, can be used in other languages.

CLS is a set of rules within CTS that languages must follow to be considered CLS- compliant

1. JIT Compiler

Converts MSIL code into machine level code.

Three Types:

Pre : compiles entire MSIL code into native code before execution

Econo : compiles only those parts of MSIL code required during execution and removes those parts that are not required anymore.

Normal : compiles only those parts of MSIL code required during execution but places them in cache for future use.

1. Garbage Collection

Works as an automatic memory manager

Manage memory by automatically allocating memory according to the requirement.

Allocates heap memory of objects

reclaims the memory allocated to them for future use.

Also ensures the safety of objects by not allowing one object to use the content of another object.

1. Metadata

Data about data

Binary info about the program

Stored along with the MSIL code or in memory along with the CLR portable file ( executable file)

Loaded into memory for proper interpretation of code and related information used.

Helps to implement code in language neutral manner.

1. Assemblies

Combination of both executable code( MISL) and metadata.

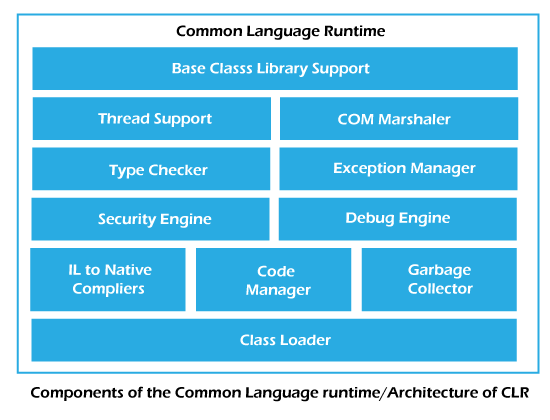
**Functions of CLR**

* It converts the program into native code.
* Handles Exceptions
* Provides type-safety
* Memory management
* Provides security
* Improved performance
* Language independent
* Platform independent
* Garbage collection
* Provides language features such as inheritance, interfaces, and overloading for object-oriented programs.

Managed code : The code that runs the CLR ( source code converted into intermediate code and that intermediate code is used by CLR)

Unmanaged code : the code that runs outside the CLR ( source code converted directly to native code without generating intermediate code)

CLR Structure



1. Base Class Library Support

It is a class library that supports classes for the .NET application.

Plays a crucial role in providing comprehensive set of classes, interfaces and buildings that form a foundation.

It abstracts platform specific details, helps the developers to write code without being concerned about underlying operating system.

includes classes for exception handling, enabling developers to handle errors and unexpected situations in a consistent manner across different applications.

Defines wide range of data types and collection classes that are fundemental to data manipulation and storage.

Support File I/O

Supports Multithreading

1. COM Marshaler

provides communication between the COM objects and the application.

COM : common object model. Binary interface standard for software applications. It provides a way for software components to communicate and interact with software components regardless of the language they are written in.

1. Code manager

Managing the code during execution.

Allocate memory to objects.

1. Type Checker

Checks the type of the variable declared.

int a;

Double b;

String c;

When we use different types, it checks the type of variable.

1. Debug Engine

Find and remove bugs from the code.

1. Class Loader

Load all classes at runtime.

Source code will be compiled and converted into Intermediate Language( very secure)

We can send the intermediate language code to anyone, then they can execute the code but can’t steal our code.

For example, If we send the intermediate language to client and that is loaded, where there will be CLR in client machine, then the CLR links that with class loader and that class loader links with all the libraries and classes required for the execution of the class.

The main task of class loader is to load data in the runtime.

Source code ----> compiled to Intermediate language by compiler ----> Using CLR ‘s class loader all the libraries and class ( data) loaded ----> Using the JIT compiler the data is converted to machine code ( native code) ----> project runs on system.

1. Exception Manager

Exception : logically the program is correct, but due to the input the program results in inconsistencies.

an event or occurrence that disrupts the normal flow of a program's instructions.

Exception Manager handles the exception of both managed(intermediate code) and unmanaged code(native or assembly code).

1. Thread Support

Multiprogramming : Multiple programs running parallel

Multithreading : a process having multiple sub parts(threads) and are executing parallely

Thread support helps in performing the multi threading

1. Security Checker

Some memory of the database should not be accessed.

The data where access is not granted shouldnt be available to everyone.

This task is done by security checker.

1. Garbage Collector

Garbage in a program : the variables , objects which are unused anymore

Collects the information about all the objects which are unused and removes from memory.

Objects created are stored in heap, garbage collector periodically checks the heap for unused objects and removes them from the memory when they are unused.

1. IL to Native compilers

converting Intermediate Language (IL) code, also known as Common Intermediate Language (CIL), to native machine code is performed by the Just-In-Time (JIT) compiler in the context of the Common Language Runtime (CLR) in the .NET framework.

**Difference between Obj and Bin Folder in visual studio?**

"bin" folder is generated during the build process and typically changes when you build the project, reflecting the latest version of the compiled binaries ready for execution or deployment.

"obj" folder contains temporary and intermediate artifacts needed during the build

**Pdb file**

used by debugger

debugging information file associated with compiled code.

contain debug information that helps tools like Visual Studio to map the compiled machine code back to the original source code.

Stores debug symbols include info about source file names,line numbers,variable names and type information. This info enables to provide more meaningful debugging experience.

It is generated along with the .exe file.

**Build rebuild and clean**

**Build:**

Compile the source code and generate executable files.

When you initiate a build, the build system compiles only the source files that are modified or are out of date since the last build.

Aims to minimize the time during compilation.

A build might include several steps, such as compilation, linking, and copying resources. The result is the creation of the executable or library file in the specified output directory.

If you have already built your project, subsequent builds are often faster because only the modified or out-of-date files are recompiled.

**Rebuild:**

When you initiate a rebuild,it cleans the existing build( deletes all the compiled files) and again the performs the full build of entire project.

Use the rebuild option when you want to start fresh and ensure that every part of your project is up-to-date. It can be useful in situations where there might be inconsistencies or when you want to make sure that everything is compiled from scratch.

**Clean:**

removing all the build artifacts and intermediary files, leaving only the source code. It deletes the contents of the output directories (such as "bin" and "obj" folders) without actually compiling anything.

**Build** compiles only modified or out-of-date files, resulting in a faster process.

**Rebuild** deletes all existing build output and performs a full compilation of the entire project.

**Clean** removes all build artifacts and intermediary files, leaving only the source code.

Server explorer

Located in view menu

Tool that allows developers to interact with various data sources within the IDE.

Convinent interface to explore and manipulate data related sources like the database server.

Key features and functions of server explorer:

1. Database connections

Allows you to create and manage various connections to database like SQL Server, MySQL, Oracle and more.

Can view tables, stored procedures, views and other database objects.

1. Data Connections

Other services like file systems and web services.

1. Web services

Integrate web services directly in the project

1. Azure Resources

Manage azure resources like virtual machines, database and storage accounts.

1. Querying and editing

Writing SQL queries and editing them from within the database.

Team explorer

provides integration with various version control and team collaboration services.

provides a centralized interface for accessing various team-related features within Visual Studio.

helps teams organize and track their development tasks.

Solution explorer

allows developers to navigate, manage, and organize the structure of their projects

provides a hierarchical view of the files, projects, and components within a solution.

shows references to external libraries and assemblies that are used by your projects. This includes references to .NET framework libraries, third-party libraries, and other project dependencies.

properties of each project, file, or folder are accessible through Solution Explorer.

Right-clicking on items in Solution Explorer opens a context menu with various actions and options. This menu allows you to perform tasks such as adding new items, managing references, renaming files, and more.

Solution Explorer provides options for grouping and sorting items based on different criteria.

can create, delete, rename, and organize files and folders directly from Solution Explorer.

**Object browser**

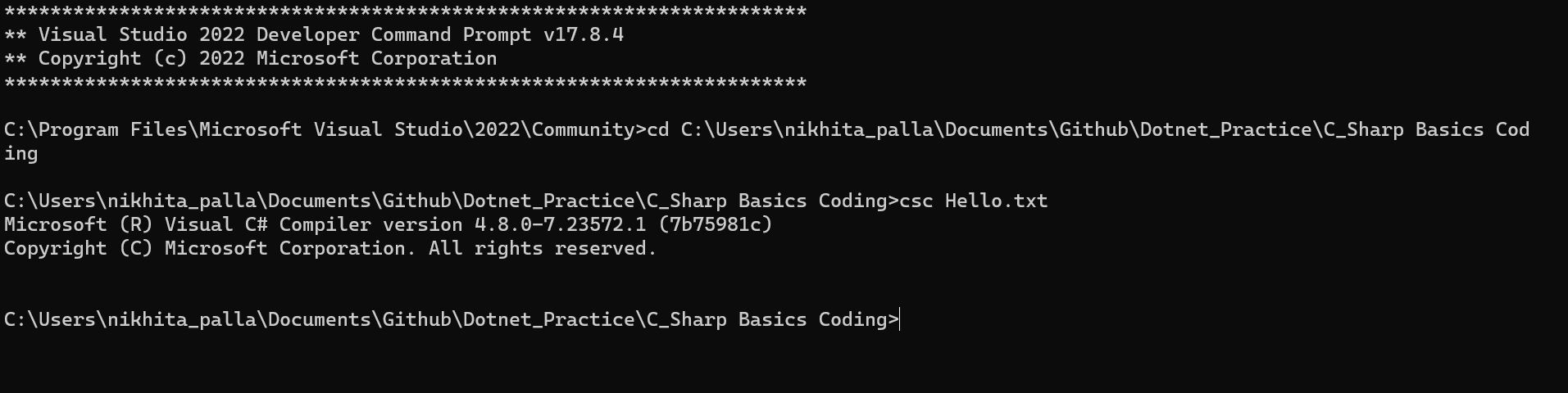
If you know what class (or interface or enum or, even, member) you want, you can search for it in Object Browser using the search box at the top of Object Browser's window. Once you find what you're looking for, just click on the Add to References icon at the top of Object Browser to add a reference to the relevant library to whatever project you have selected in Solution Explorer.

explore and navigate through the types, members, and namespaces available in assemblies, including those in the .NET Framework and other referenced libraries.

**Difference between trace and debug**

while both debugging and tracing involve logging information for diagnostic purposes, debugging is primarily focused on finding and fixing issues during development, and the information is usually more detailed. Tracing, on the other hand, is often used in production or testing environments to monitor the application's behavior, and the information is typically less detailed than debugging information

**Running a C# code**

****

To open a assembly

Ildasm Hello.exe

A screenshot of a computer

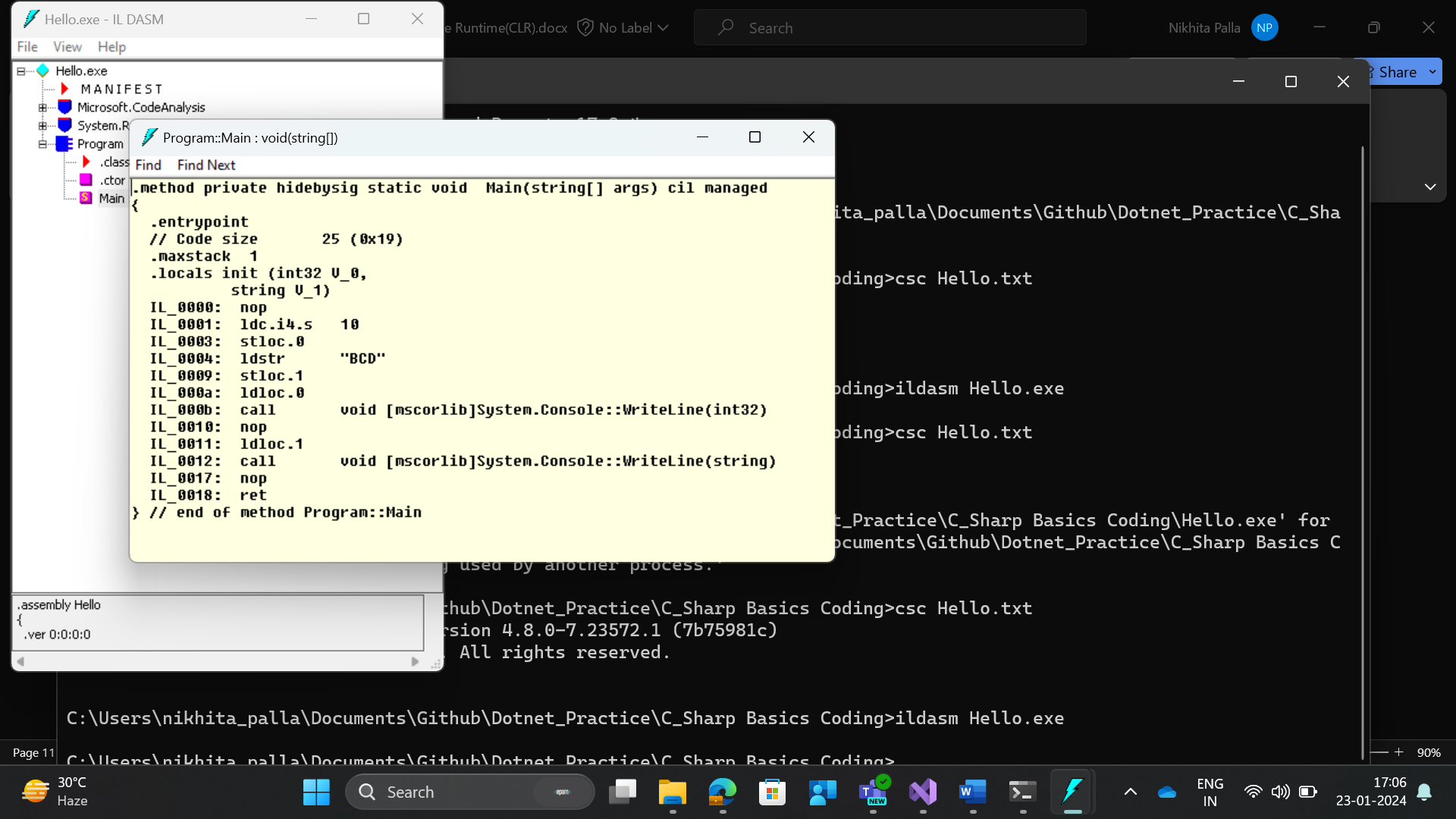
Description automatically generated

Manifest or Metadata

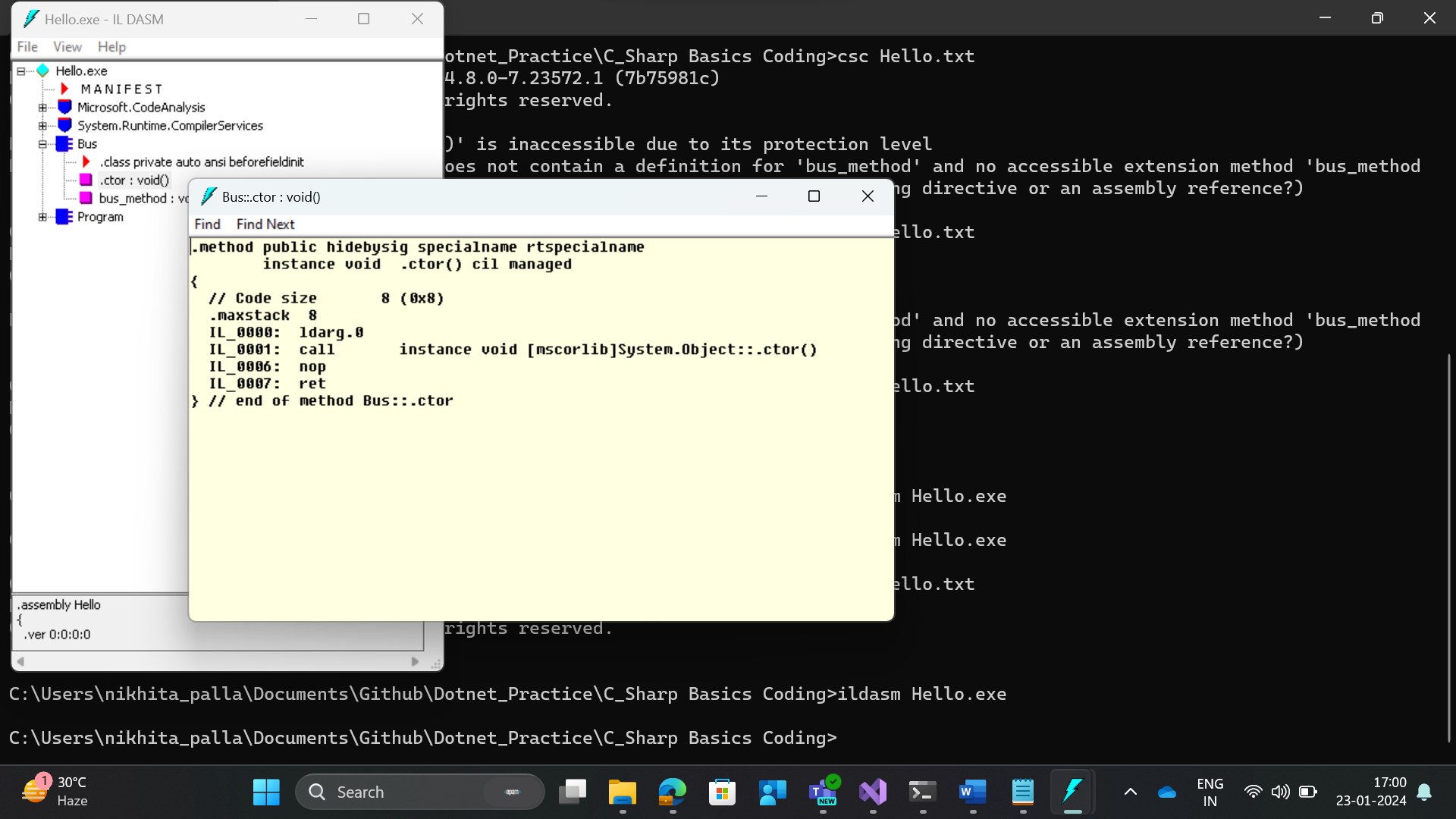
A screenshot of a computer

Description automatically generated

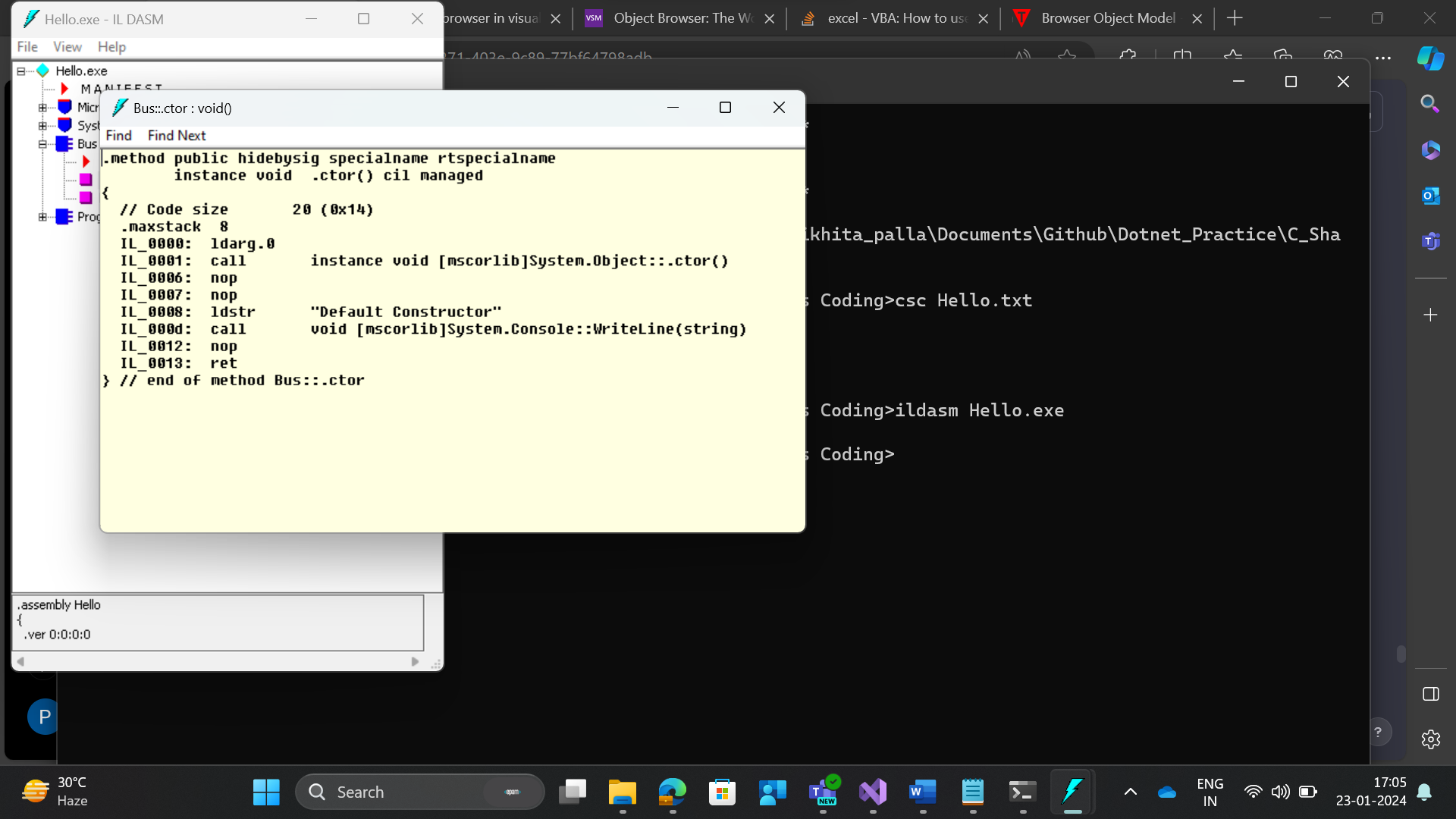
IL -> main



IL -> .ctor -🡪 constructor



When we create a default constructor within our code:



Comparing VB and c# code MSIL

**Compiling VB code**

**Imports System**

**Module Module1**

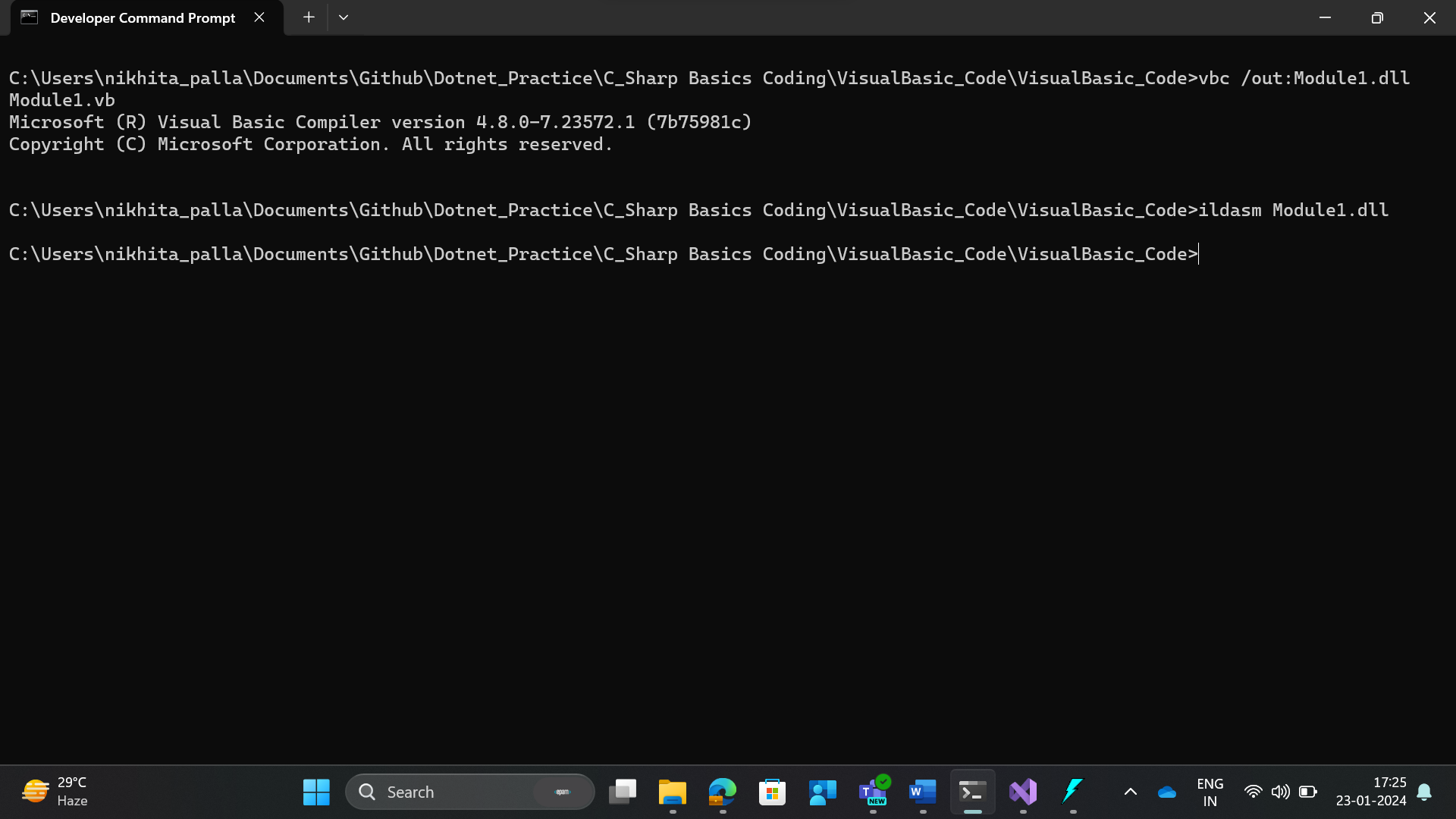
**Sub Main()**

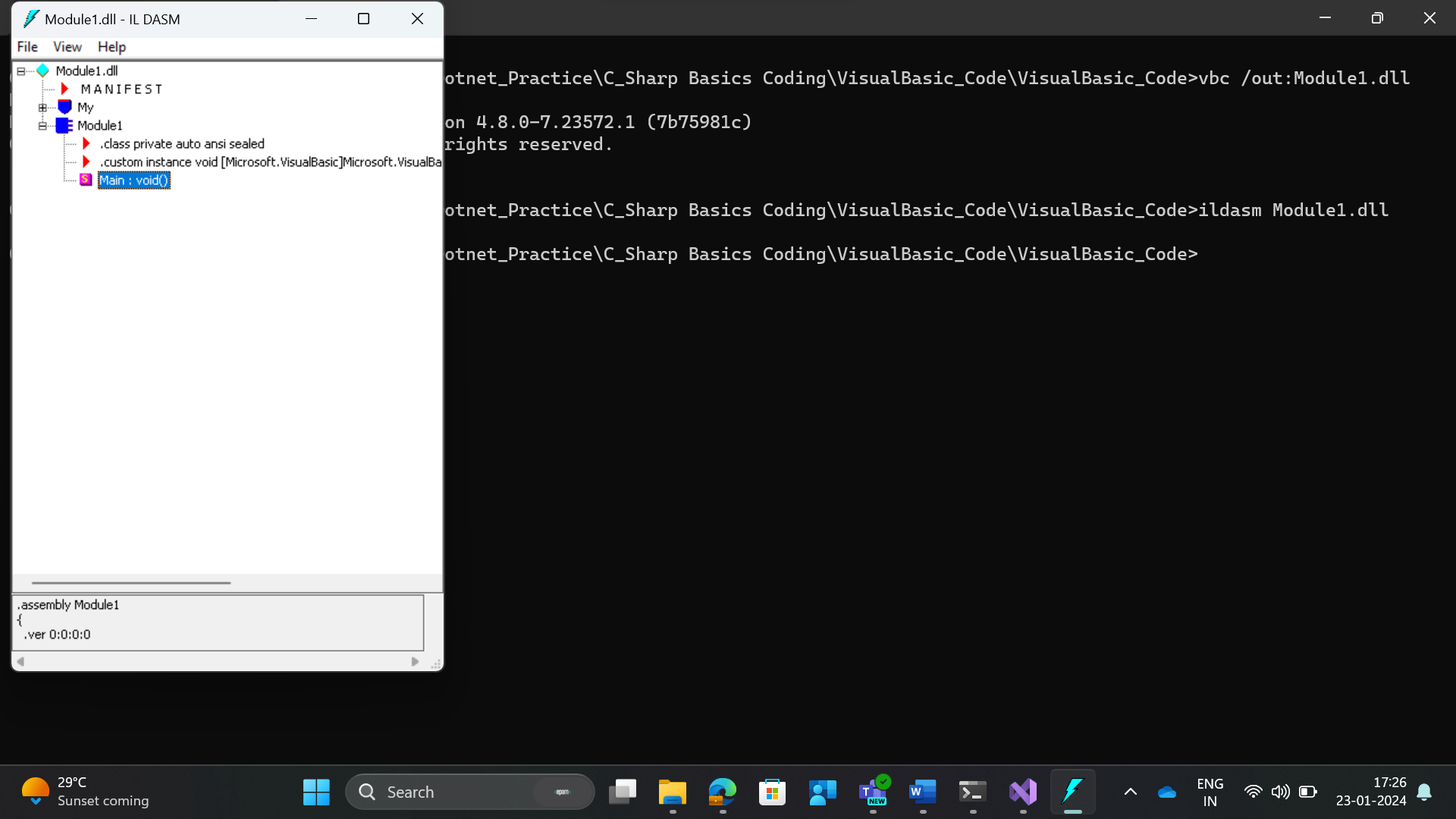
**Dim a As Integer = 10**

**Console.WriteLine(a)**

**End Sub**

**End Module**





A screenshot of a computer

Description automatically generated

**Compiling C# code**

using System;

class Program

{

static void Main(String[] args)

{

int a=10;

Console.WriteLine(a);

}

}

A screen shot of a computer program

Description automatically generated

