



C# .NET Core Εισαγωγή

Αθ. Ανδρούτσος



Microsoft .NET (1)

Προγραμματισμός με C#.NET Core

- Το .NET με τον τίτλο *.NET Framework* δημιουργήθηκε από την Microsoft ήδη από το 2002
- Πρόκειται για ένα περιβάλλον εκτέλεσης διαφόρων γλωσσών προγραμματισμού όπως VB.NET C++, C#.NET, κλπ.



Microsoft .NET (2)

Προγραμματισμός με C#.NET Core

- Ο λόγος που η Microsoft δημιούργησε το .NET ήταν για να μπορούν να επικοινωνούν μεταξύ τους προγράμματα που ήταν γραμμένα σε διαφορετικές και ανομοιογενείς γλώσσες προγραμματισμού, που υποστήριζε το .NET



Microsoft .NET (3)

Προγραμματισμός με C#.NET Core

- Η επικοινωνία προγραμμάτων που έχουν γραφεί σε διαφορετικές γλώσσες προγραμματισμού πρέπει να λαμβάνει υπόψη το Type System της κάθε γλώσσας που είναι εν γένει διαφορετικό και να μεταφράζει με ομοιογενή τρόπο μεταξύ των γλωσσών



Microsoft .NET (4)

Προγραμματισμός με C#.NET Core

- Ένα από τα πρώτα συστήματα κατανεμημένου προγραμματισμού που υποστήριζε διαλειτουργικότητα μεταξύ διαφορετικών γλωσσών προγραμματισμού ήταν η CORBA
- Στη συνέχεια, το .NET έκανε το ίδιο πράγμα αλλά εξελίχθηκε περαιτέρω υποστηρίζοντας περισσότερες γλώσσες προγραμματισμού



Microsoft .NET (3)

Προγραμματισμός με C#.NET Core

- Για να επιτευχθεί η δυνατότητα διαλειτουργικότητας χρειάζεται ένα **κοινό σύστημα Τύπων Δεδομένων** στο οποίο να αναφέρονται όλες οι γλώσσες του .NET καθώς και μία **μεταγλώττιση δύο φάσεων** όπου στην 1^η φάση όλες οι γλώσσες θα μεταγλωττίζονται σε ενδιάμεσο κώδικα (Intermediate Language - IL) που στη συνέχεια θα εκτελείται από το περιβάλλον (runtime) .NET



Microsoft .NET (3)

Προγραμματισμός με C#.NET Core

- Πράγματι όλες οι γλώσσες του .NET έχουν ένα Type System με αναφορά στους τύπους του .NET (Common Type System – CTS)
- Επίσης, η μεταγλώττιση είναι δύο φάσεων, όπου πρώτα τα προγράμματα μεταγλωττίζονται σε ενδιάμεσο κώδικα (MSIL) και στη συνέχεια εκτελούνται (διερμηνεύονται) από το CLR (Common Language Runtime) του .NET, που λειτουργεί ως Jitter δηλαδή βελτιστοποιεί τη διαδικασία του interpretation



Common Type System (CTS)

Προγραμματισμός με C#.NET Core

CTS Data Type	VB Keyword	C# Keyword
System.Byte	Byte	byte
System.SByte	SByte	sbyte
System.Int16	Short	Short
System.Int32	Integer	Int
System.Int64	Long	Long
System.UInt16	UShort	Ushort
System.UInt32	UInteger	UInt
System.UInt64	ULong	Ulong
System.Single	Single	Float
System.Double	Double	double
System.Object	Object	object
System.Char	Char	char
System.String	String	string
System.Decimal	Decimal	decimal
System.Boolean	Boolean	bool

- Επίσης, υποστηρίζονται οι τύποι *class*, *interface*, *enum*, *struct*, *delegate* (αντίστοιχο των *callback functions*)



.NET Core

Προγραμματισμός με C#.NET Core

- Το πρόβλημα του .NET Framework ήταν ότι έτρεχε μόνο σε Windows
- Το 2016 η Microsoft δημιούργησε το .NET Core και πλέον σκέτο .NET που τρέχει σε όλα τα Λειτουργικά Συστήματα χρησιμοποιώντας τα CoreCLR και CoreFX (.NET Core Runtime)



.NET Versions (1)

Προγραμματισμός με C#.NET Core

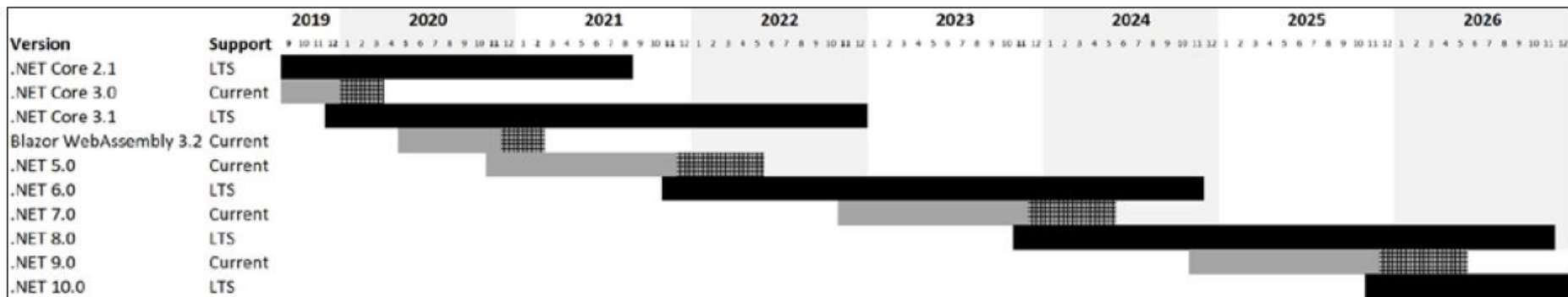
Release	Released	Support Status	Latest
8 (LTS)	11 months ago (14 Nov 2023)	Ends in 2 years (10 Nov 2026)	8.0.10 (08 Oct 2024)
7	1 year and 12 months ago (08 Nov 2022)	Ended 5 months and 3 weeks ago (14 May 2024)	7.0.20 (29 May 2024)
6 (LTS)	2 years and 12 months ago (08 Nov 2021)	Ends in 1 week and 3 days (12 Nov 2024)	6.0.35 (08 Oct 2024)
5	3 years and 11 months ago (10 Nov 2020)	Ended 2 years and 5 months ago (10 May 2022)	5.0.17 (10 May 2022)
Core 3.1 (LTS)	4 years and 11 months ago (03 Dec 2019)	Ended 1 year and 10 months ago (13 Dec 2022)	3.1.32 (13 Dec 2022)
Core 3.0	5 years ago (23 Sep 2019)	Ended 4 years and 8 months ago (03 Mar 2020)	3.0.3 (19 Feb 2020)
Core 2.2	5 years and 11 months ago (04 Dec 2018)	Ended 4 years and 10 months ago (23 Dec 2019)	2.2.8 (19 Nov 2019)
Core 2.1 (LTS)	6 years ago (30 May 2018)	Ended 3 years ago (21 Aug 2021)	2.1.30 (19 Aug 2021)
Core 2.0	7 years ago (14 Aug 2017)	Ended 6 years ago (01 Oct 2018)	2.0.9 (10 Jul 2018)



.NET Versions (2)

Προγραμματισμός με C#.NET Core

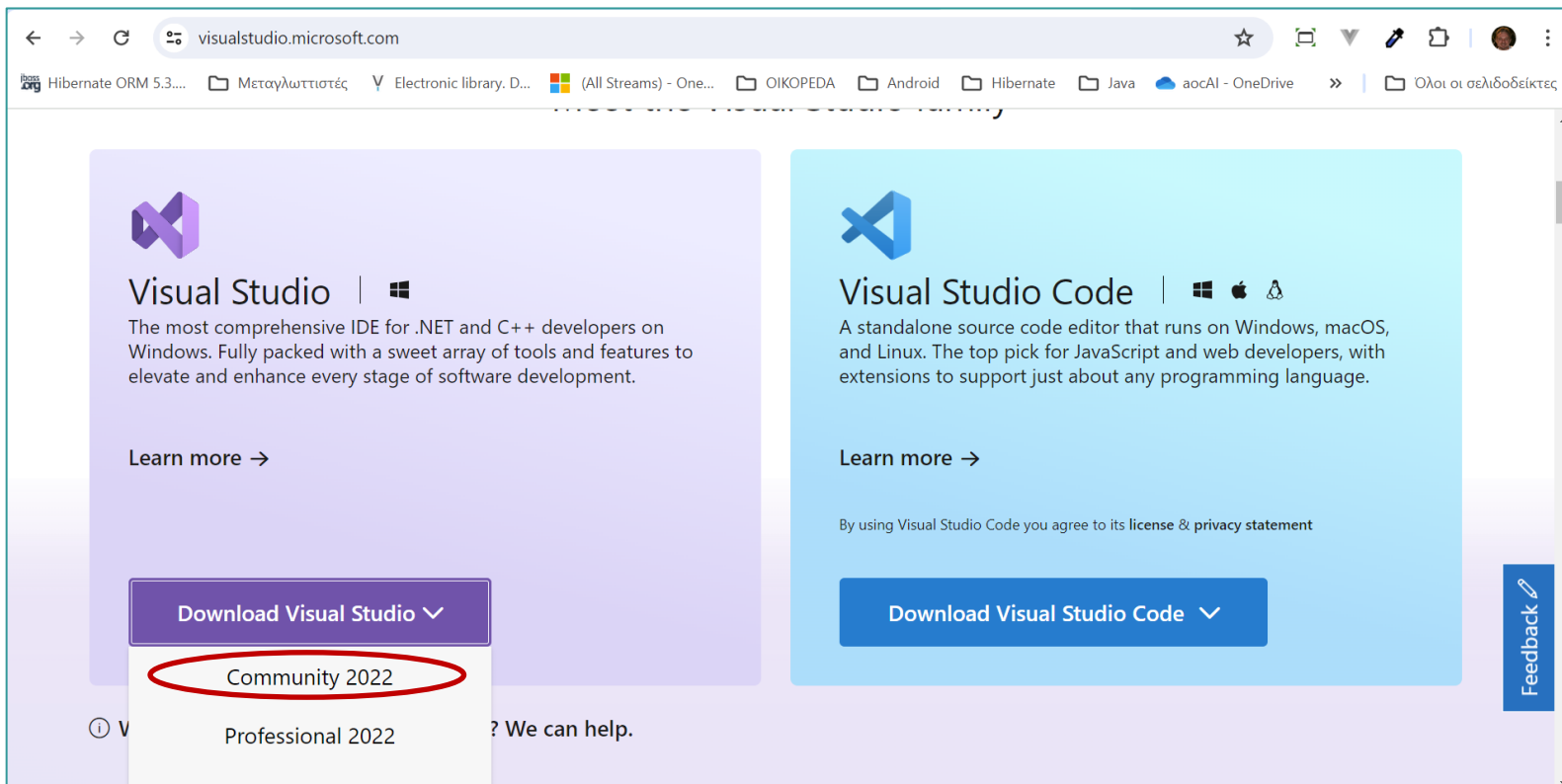
- Οι εκδόσεις LTS (Long Term Support) εκδίδονται κάθε δύο χρόνια και υποστηρίζονται συνήθως για τρία χρόνια
- Οι υπόλοιπες εκδόσεις (Current) υποστηρίζονται για διάστημα περίπου 18 μηνών





Visual Studio Community Edition

Προγραμματισμός με C#.NET Core

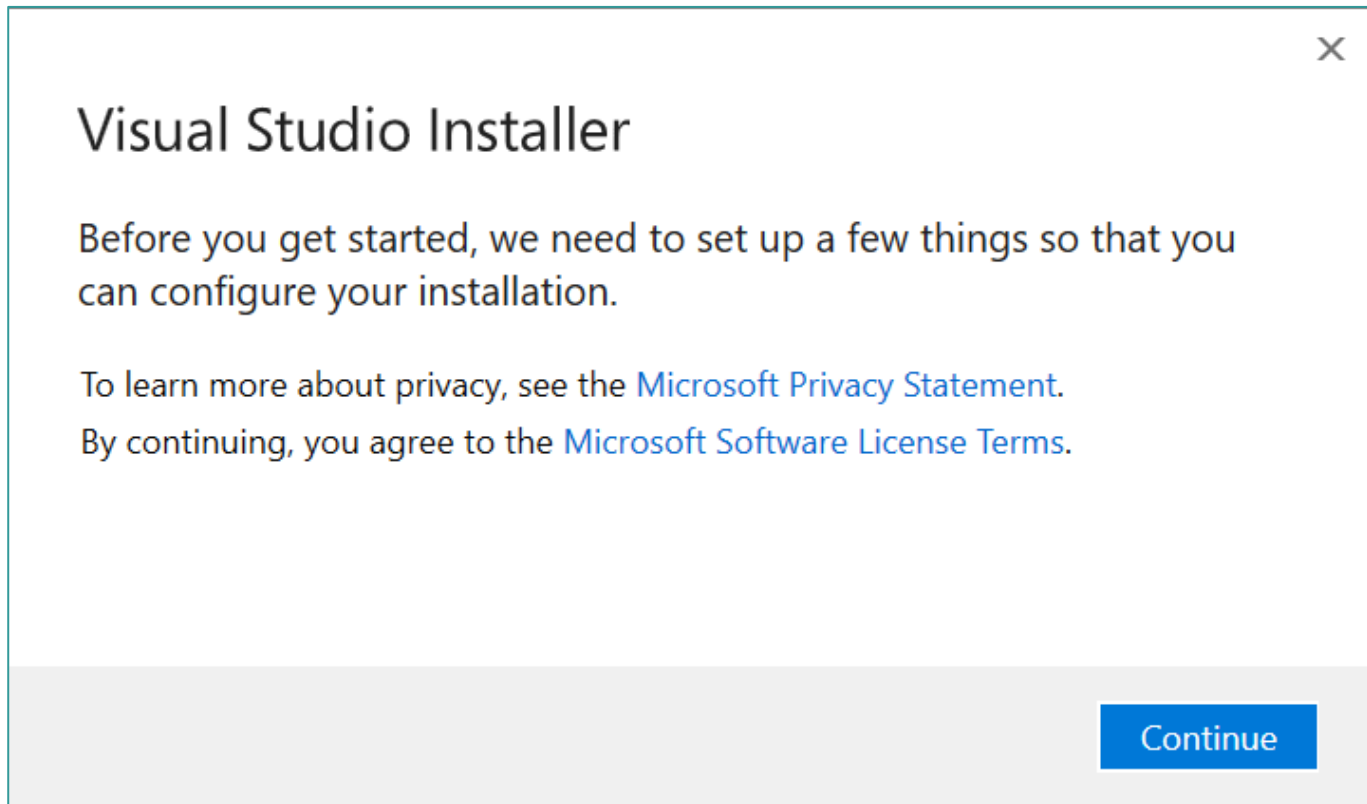


- Το Visual Studio είναι το IDE της Microsoft για το .NET για Windows και Mac. Για Linux, MacOS μπορούμε να χρησιμοποιήσουμε το Visual Studio Code. Κάνουμε download **το community edition** του Visual Studio



Visual Studio 2022

Προγραμματισμός με C#.NET Core

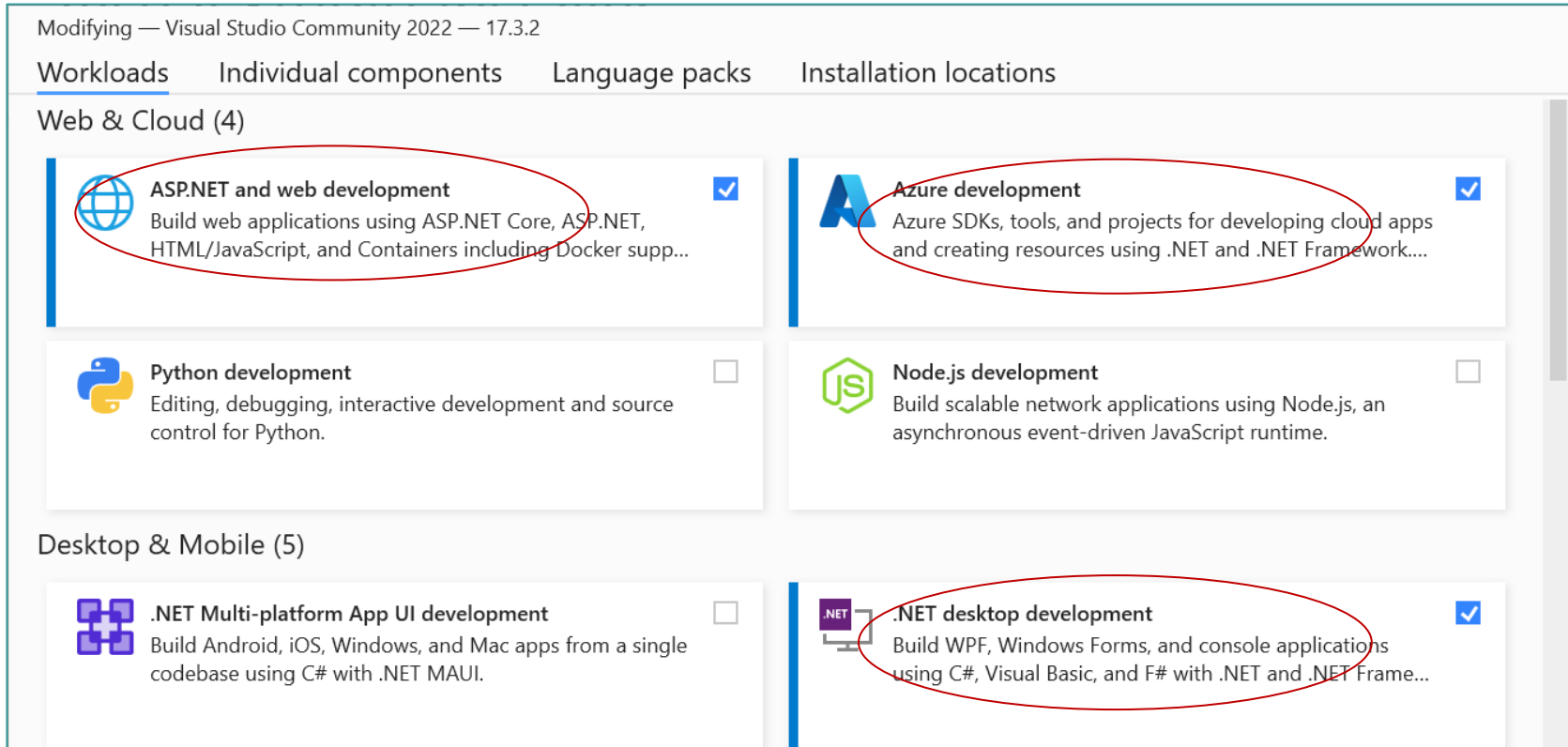


- Ξεκινάμε την διαδικασία της εγκατάστασης



Workloads (1)

Προγραμματισμός με C#.NET Core



- Η εγκατάσταση έχει τη λογική των workloads, ανεξάρτητων βιβλιοθηκών που μπορούμε να εγκαταστήσουμε ανάλογα με τις ανάγκες. Για το παρόν course επιλέγουμε τα workloads που είναι σε κόκκινο κύκλο



Workloads (2)

Προγραμματισμός με C#.NET Core

Modifying — Visual Studio Community 2022 — 17.3.2

Workloads

Individual components

Language packs

Installation locations



Desktop development with C++

Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.



Mobile development with C++

Build cross-platform applications for iOS, Android or Windows using C++.



Universal Windows Platform development

Create applications for the Universal Windows Platform with C#, VB, or optionally C++.



Gaming (2)



Game development with Unity

Create 2D and 3D games with Unity, a powerful cross-platform development environment.



Game development with C++

Use the full power of C++ to build professional games powered by DirectX, Unreal, or Cocos2d.



- Επιλέγουμε και την C++ για να δούμε και ένα project με C++



Workloads (3)

Προγραμματισμός με C#.NET Core

Other Toolsets (5)



Data storage and processing

Connect, develop, and test data solutions with SQL Server, Azure Data Lake, or Hadoop.



Data science and analytical applications

Languages and tooling for creating data science applications, including Python and F#.



Visual Studio extension development

Create add-ons and extensions for Visual Studio, including new commands, code analyzers and tool windows.



Office/SharePoint development

Create Office and SharePoint add-ins, SharePoint solutions, and VSTO add-ins using C#, VB, and JavaScript.



Linux and embedded development with C++

Create and debug applications running in a Linux environment or on an embedded device.



- Data Storage and processing για ανάπτυξη εφαρμογών με SQL Server



Modify Installation

Προγραμματισμός με C#.NET Core

- Οποιαδήποτε στιγμή μετά την εγκατάσταση μπορούμε να προσθέσουμε workloads
- Στο Visual Studio πάμε στο **Tools > Get Tools and Features...** και ανοίγει ο Visual Studio Installer
- Ή, ανοίγουμε τον **Visual Studio Installer** και επιλέγουμε workloads ή components που θέλουμε να εγκατασταθούν. Επιλέγουμε **Modify**.



Visual Studio Code (Linux)

Προγραμματισμός με C#.NET Core

dotnet.microsoft.com/en-us/download/dotnet/8.0

Download .NET 8.0

Not what you're looking for? Visit the [downloads](#) page for more options.

8.0.10 **Security patch**

[Release notes](#) Latest release date October 8, 2024

Build apps - SDK 8.0.403

OS	Installers	Binaries
Linux	Package manager instructions	Arm32 Arm32 Alpine Arm64 Arm64 Alpine x64 x64 Alpine
macOS	Arm64 x64	Arm64 x64
Windows	x64 x86 Arm64	x64 x86 Arm64

Run apps - Runtime 8.0.10

ASP.NET Core Runtime 8.0.10

The ASP.NET Core Runtime enables you to run existing web/server applications. **On Windows, we recommend installing the Hosting Bundle, which includes the .NET Runtime and IIS support.**

IIS runtime support (ASP.NET Core Module v2)
18.0.24262.10

- Σε περιβάλλον Linux εγκαταστήστε το VSC. Εγκαταστήστε το .NET SDK για το version 8.0 (τα runtimes περιλαμβάνονται στα SDKs) από <https://dotnet.microsoft.com/en-us/download/dotnet/8.0>
- Εγκαταστήστε το C# extension (View / Extensions) C# for Visual Studio (powered by OmniSharp)



Έλεγχος εκδόσεων

Προγραμματισμός με C#.NET Core

C:\> Γραμμή εντολών

```
Microsoft Windows [Version 10.0.19045.5011]  
(c) Microsoft Corporation. Με επιφύλαξη κάθε νόμιμου δικαιώματος.
```

```
C:\Users\agana>dotnet --list-sdks  
5.0.403 [C:\Program Files\dotnet\sdk]  
8.0.403 [C:\Program Files\dotnet\sdk]
```

```
C:\Users\agana>_
```

- dotnet --list-sdks



dotnet --list-runtimes

Προγραμματισμός με C#.NET Core

```
Γραμμή εντολών
(c) Microsoft Corporation. Με επιφύλαξη κάθε νόμιμου δικαιώματος.

C:\Users\asana>dotnet --list-sdks
5.0.403 [C:\Program Files\dotnet\sdk]
8.0.403 [C:\Program Files\dotnet\sdk]

C:\Users\asana>dotnet --list-runtimes
Microsoft.AspNetCore.All 2.1.30 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.All]
Microsoft.AspNetCore.App 2.1.30 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
Microsoft.AspNetCore.App 3.1.21 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
Microsoft.AspNetCore.App 3.1.32 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
Microsoft.AspNetCore.App 5.0.12 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
Microsoft.AspNetCore.App 5.0.17 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
Microsoft.AspNetCore.App 6.0.35 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
Microsoft.AspNetCore.App 8.0.10 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
Microsoft.NETCore.App 2.1.30 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
Microsoft.NETCore.App 3.1.21 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
Microsoft.NETCore.App 3.1.32 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
Microsoft.NETCore.App 5.0.12 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
Microsoft.NETCore.App 5.0.17 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
Microsoft.NETCore.App 6.0.35 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
Microsoft.NETCore.App 8.0.10 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
Microsoft.WindowsDesktop.App 3.1.21 [C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App]
Microsoft.WindowsDesktop.App 3.1.32 [C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App]
Microsoft.WindowsDesktop.App 5.0.12 [C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App]
Microsoft.WindowsDesktop.App 5.0.17 [C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App]
Microsoft.WindowsDesktop.App 6.0.35 [C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App]
Microsoft.WindowsDesktop.App 8.0.10 [C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App]

C:\Users\asana>
```

- dotnet --list-runtimes



Το πρώτο πρόγραμμα στην C# Hello World

Προγραμματισμός με C#.NET Core

- Το όνομα του αρχείου δεν είναι τεχνικά απαραίτητα το ίδιο με το όνομα της κλάσης (class) αλλά κατά σύμβαση είναι, και συνίσταται να είναι! Η κλάση που περιέχει την Main είναι η Program και το αρχείο είναι το Program.cs

```
1      using System;
2
3      namespace HelloWorld
4      {
5          /// <summary>
6          /// Το πρώτο πρόγραμμα Hello World, Hello Coding Factory!
7          /// </summary>
8          internal class Program
9          {
10             static void Main(string[] args)
11             {
12                 Console.Write("Hello, Coding Factory!");
13                 Console.WriteLine(); // shortcut -- cw και δύο φορές tab
14             }
15         }
16     }
```



Προγράμματα στην C#

Προγραμματισμός με C#.NET Core

- Κάθε πρόγραμμα στην C# το γράφουμε μέσα σε κλάσεις
- Κάθε πρόγραμμα πρέπει να περιέχει τουλάχιστον μία κλάση
- Τα namespaces στην C# είναι παρόμοια (αλλά όχι ίδια) με τα packages στην Java. Έχουν τον ρόλο της οργάνωσης των κλάσεων αλλά τεχνικά δεν αντιστοιχούν σε φακέλους, είναι λογικοί containers
- Καλό είναι να τα αντιστοιχούμε σε φακέλους όπως στην Java για να υπάρχει συνάφεια και λογική συνέπεια στο Project μας αλλά δεν είναι απαραίτητο



Κλάσεις στην C#

Προγραμματισμός με C#.NET Core

- Κάθε κλάση έχει ένα όνομα και αποτελείται από μέλη που δηλώνονται ανάμεσα στα άγκιστρα { και }
- Τα ονόματα (κλάσεων, μεταβλητών, κλπ.) ονομάζονται αναγνωριστικά (identifiers)
- Κατά σύμβαση τα αναγνωριστικά των κλάσεων ξεκινούν με κεφαλαίο γράμμα, Pascal Case



Αναγνωριστικά (Identifiers)

Προγραμματισμός με C#.NET Core

- Τα αναγνωριστικά (identifiers) είναι όλα τα ονόματα που δίνει ο προγραμματιστής σε κλάσεις, μεθόδους, μεταβλητές, κλπ.
- Ξεκινούν με γράμμα, _ (όχι με ψηφίο), και οι επόμενοι χαρακτήρες μπορούν να περιλαμβάνουν γράμματα ή αριθμητικά ψηφία
- Δεσμευμένες λέξεις (όπως η class, κλπ) απαγορεύεται να χρησιμοποιούνται ως αναγνωριστικά
- Πεζά δεν είναι ίδια με κεφαλαία (case sensitive)



Ονοματοδοσία αναγνωριστικών

Προγραμματισμός με C#.NET Core

- Δηλώνουν τη σημασία του αντικειμένου που προσδιορίζουν
- Στις **κλάσεις** και γενικά στους τύπους (interfaces, enums) κατά σύμβαση το πρώτο γράμμα είναι κεφαλαίο και το πρώτο γράμμα κάθε λέξης που περιέχεται επίσης κεφαλαίο (Pascal Case)
- Στις **μεταβλητές** κατά σύμβαση το πρώτο γράμμα είναι μικρό και το πρώτο γράμμα κάθε λέξης που περιέχεται κεφαλαίο (Camel Case)
- Στις **μεθόδους** κατά σύμβαση το πρώτο γράμμα είναι κεφαλαίο και το πρώτο γράμμα κάθε επόμενης λέξης ξεκινά με κεφαλαίο (Pascal Case)



Κλάσεις και μέθοδοι

Προγραμματισμός με C#.NET Core

```
1  using System;
2
3  namespace HelloWorld
4  {
5      /// <summary>
6      /// Το πρώτο πρόγραμμα Hello World, Hello Coding Factory!
7      /// </summary>
8      internal class Program
9      {
10         static void Main(string[] args)
11         {
12             Console.Write("Hello, Coding Factory!");
13             Console.WriteLine(); // shortcut -- cw και δύο φορές tab
14         }
15     }
16 }
```

- Οι μέθοδοι επιτελούν λειτουργίες και τυπικά ομαδοποιούν εντολές
 - Η κλάση Program περιέχει μόνο την μέθοδο `Main()`
 - Οι μέθοδοι αποτελούνται από την επικεφαλίδα και το σώμα της μεθόδου



Η μέθοδος Main (1)

Προγραμματισμός με C#.NET Core

- Main → Ειδική μέθοδος - από όπου ξεκινά να εκτελείται ένα πρόγραμμα C#
- Η επικεφαλίδα της Main είναι η:
`static void Main(string args[])` ή
`static int Main(string args[])`
- Ο χαρακτηρισμός ορατότητας (access modifier) στις μεθόδους των κλάσεων της C# by default είναι **private**
- Αυτό σημαίνει πως μπορεί να κληθεί μόνο από το runtime ως entry point



Η μέθοδος Main (3)

Προγραμματισμός με C#.NET Core

- **void** – Αν χαρακτηρίσουμε την main ως void, τότε δεν υπάρχει επιστρεφόμενο αποτέλεσμα
- **int** – Ο χαρακτηρισμός της main ως int είναι έγκυρος και προέρχεται από τη C++ (όπου η main επιστρέφει int) και ο σκοπός είναι να επιστρέφει το πρόγραμμά μας στο λειτουργικό σύστημα μία τιμή (0 τελείωσε καλώς, <> 0 τελείωσε με συνθήκη λάθους). Επομένως σε αυτή την περίπτωση θα πρέπει να έχουμε μία εντολή return



int Main

Προγραμματισμός με C#.NET Core

```
1  using System;
2
3  namespace HelloWorld
4  {
5      /// <summary>
6      /// Το πρώτο πρόγραμμα Hello World, Hello Coding Factory!
7      /// </summary>
8      internal class Program
9      {
10         static int Main(string[] args)
11         {
12             Console.Write("Hello, Coding Factory!");
13             Console.WriteLine(); // shortcut -- cw και δύο φορές tab
14             return 0;
15         }
16     }
17 }
```

- Παράδειγμα χρήσης της main όταν επιστρέφει int



Τυπικές Παράμετροι

Προγραμματισμός με C#.NET Core

- Ακολουθία (πιθανώς κενή) από τύπους-αναγνωριστικά που διαχωρίζονται με κόμμα
- Από όπου λαμβάνει η μέθοδος δεδομένα



Το σώμα της μεθόδου

Προγραμματισμός με C#.NET Core

- Μετά την επικεφαλίδα ακολουθεί το σώμα που περιλαμβάνει τις εντολές
- Κάθε εντολή τελειώνει με ένα ελληνικό ερωτηματικό
- Το σώμα αρχίζει με το σύμβολο { που σημαίνει Αρχή/Begin και τελειώνει με το } που σημαίνει Τέλος/End
- Στην C# το αρχικό { ξεκινάει by convention στην επόμενη γραμμή όπως στην C++ (Allman Style) και όχι όπως στην Java που ξεκινάει στην ίδια γραμμή (K&R Style)



Εντολές (1)

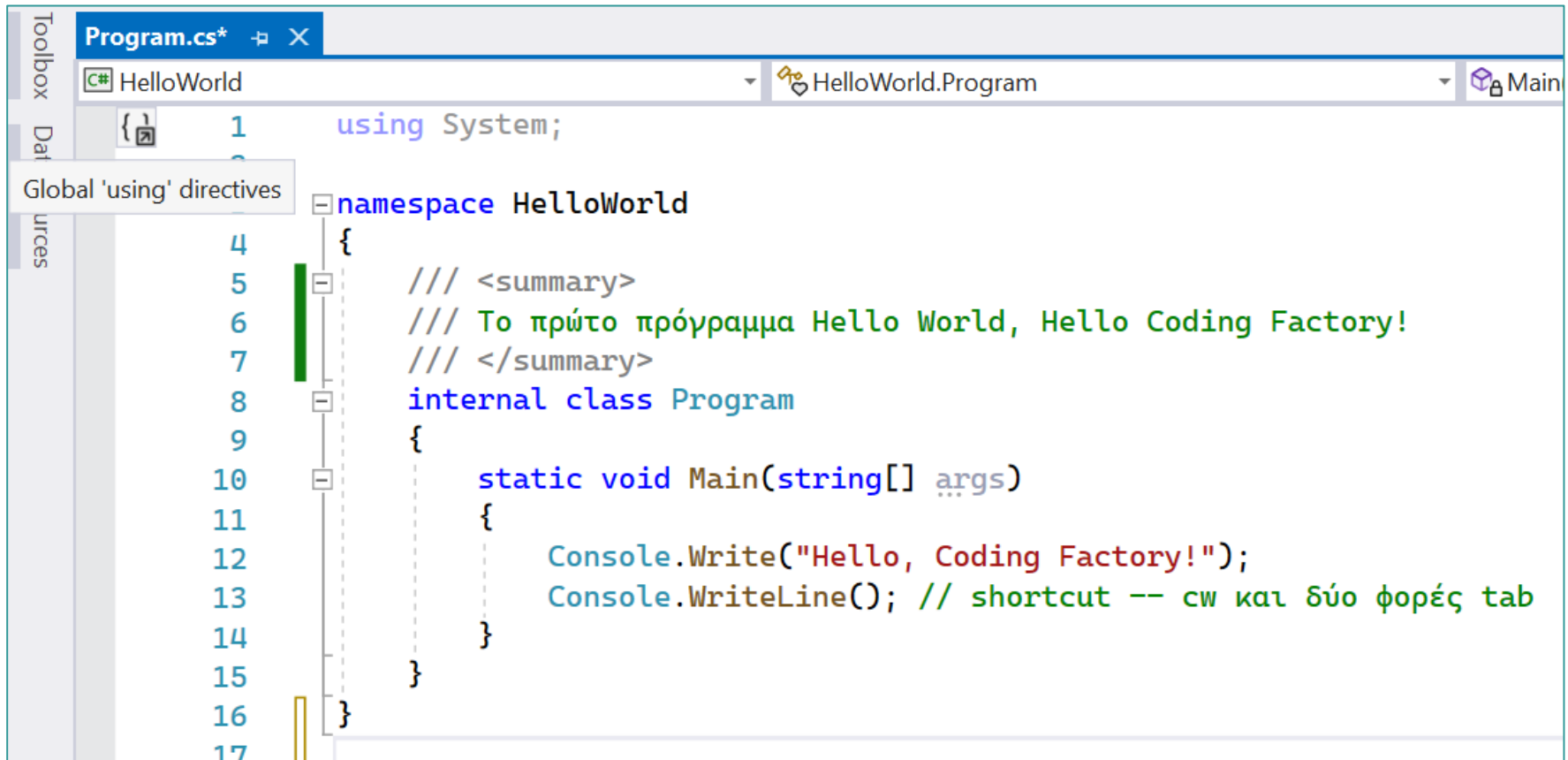
Προγραμματισμός με C#.NET Core

- Στο παράδειγμα μας υπάρχει μία μόνο εντολή, που είναι μία κλήση της μεθόδου **WriteLine** που ανήκει στην κλάση **Console** του namespace **System**. Οπότε το πλήρες όνομα είναι **System.Console.WriteLine()**
- Αν κάνουμε ***using System;*** μπορούμε να αναφερθούμε ως **Console.WriteLine()**
- Στο .NET 6.0 και στο .NET 8.0 του Visual Studio το *using System* γίνεται αυτόματα μέσω των *Global using Directives* (αυτό δεν σημαίνει πως γίνεται αυτόματα σε όλους τους editors)



Global using Directives

Προγραμματισμός με C#.NET Core



```
1  using System;
2
3  namespace HelloWorld
4  {
5      /// <summary>
6      /// Το πρώτο πρόγραμμα Hello World, Hello Coding Factory!
7      /// </summary>
8      internal class Program
9      {
10         static void Main(string[] args)
11         {
12             Console.WriteLine("Hello, Coding Factory!");
13             Console.WriteLine(); // shortcut -- cw και δύο φορές tab
14         }
15     }
16 }
17
```



Console.WriteLine

Προγραμματισμός με C#.NET Core

```
1 using System;
2
3 namespace HelloWorld
4 {
5     /// <summary>
6     /// Το πρώτο πρόγραμμα Hello World, Hello Coding Factory!
7     /// </summary>
8     internal class Program
9     {
10         static void Main(string[] args)
11         {
12             Console.Write("Hello, Coding Factory!");
13             Console.WriteLine(); // shortcut -- cw και δύο φορές tab
14         }
15     }
16 }
17
```

- Το **πραγματικό όρισμα** είναι η συμβολοσειρά Hello, Coding Factory!
- Η μέθοδος WriteLine εμφανίζει στο stdout (κονσόλα) ότι βρίσκεται μέσα στα διπλά εισαγωγικά ακριβώς όπως έχει πληκτρολογηθεί και αφήνει μία αλλαγή γραμμής



Εντολές (2)

Προγραμματισμός με C#.NET Core

- Η Main μας θα μπορούσε να γραφεί με διαφορετικό τρόπο και να περιέχει αντί για μία, δύο εντολές:
 - `System.Console.Write("Hello World buy THANOS!")`
 - `System.Console.WriteLine();`
- Πρώτα γράφει το αλφαριθμητικό με την `Write` και μετά γράφει μία αλλαγή γραμμής



Διαδικασία Μεταγλώττισης

Προγραμματισμός με C#.NET Core

- Η μεταγλώττιση γίνεται σε MS-DOS με την εντολή:
 - `csc όνομα-αρχείου.cs`
- Για παράδειγμα το πρόγραμμα του παραδείγματος το αποθηκεύουμε σε αρχείο με όνομα **HelloWorld.cs** και στη συνέχεια δίνουμε την εντολή:
 - `csc HelloWorld.cs`
- Παράγεται το αρχείο **HelloWorld.exe**
- **Προσοχή στα using.** Σε περιβάλλοντα άλλα πλην του Visual Studio, δεν υπονοούνται από τα global using directives



Μεταγλώττιση

Προγραμματισμός με C#.NET Core

- Σε περιβάλλον MS-DOS (View/Terminal) δίνουμε: **csc Program.cs**

```
Developer PowerShell
+ Developer PowerShell | [Icons]
Directory: C:\Users\A8ana\source\repos2022com\CodingFactoryTestbed221\HelloWorld

Mode                LastWriteTime         Length Name
----                -
d-----         24/8/2022    7:33 μμ             bin
d-----         31/8/2022    2:04 μμ             obj
d-----         31/8/2022    2:03 μμ          Properties
-a-----         24/8/2022    7:33 μμ      249 HelloWorld.csproj
-a-----          4/9/2022    3:50 μμ      408 Program.cs

PS C:\Users\A8ana\source\repos2022com\CodingFactoryTestbed221\HelloWorld>
```

```
PS C:\Users\A8ana\source\repos2022com\CodingFactoryTestbed221\HelloWorld> csc .\Program.cs
```



Εκτέλεση Program.exe

Προγραμματισμός με C#.NET Core

Mode	LastWriteTime	Length	Name
----	-----	-----	----
d-----	24/8/2022 7:33 μμ		bin
d-----	31/8/2022 2:04 μμ		obj
d-----	31/8/2022 2:03 μμ		Properties
-a----	24/8/2022 7:33 μμ	249	HelloWorld.csproj
-a----	4/9/2022 3:50 μμ	408	Program.cs
-a----	4/9/2022 4:31 μμ	3584	Program.exe

```
PS C:\Users\A8ana\source\repos2022com\CodingFactoryTestbed221\HelloWorld> .\Program
Hello, Coding Factory!
PS C:\Users\A8ana\source\repos2022com\CodingFactoryTestbed221\HelloWorld>
```

- Εκτέλεση με το όνομα του εκτελέσιμου:
Program



Εκτέλεση του προγράμματος

Προγραμματισμός με C#.NET Core

- HelloWorld
 - Τα Όνομα.exe αρχεία είναι εκτελέσιμα και εκτελούνται με την εντολή Όνομα ή ./Όνομα
 - Το αναμενόμενο αποτέλεσμα είναι Hello, Coding Factory!

```
PS C:\Users\A8ana\source\repos2022com\CodingFactoryTestbed221\HelloWorld> .\Program
Hello, Coding Factory!
PS C:\Users\A8ana\source\repos2022com\CodingFactoryTestbed221\HelloWorld>
```



Visual Studio (1)

Προγραμματισμός με C#.NET Core

- Το πρόγραμμα Visual Studio της Microsoft το οποίο εγκαταστήσαμε χρησιμοποιείται για τη συγγραφή προγραμμάτων C#
- Εναλλακτικά χρησιμοποιούμε το Visual Studio Code



Visual Studio (2)

Προγραμματισμός με C#.NET Core

- Το VS παρέχει ένα παραθυρικό περιβάλλον που ενσωματώνει όλα τα στάδια στη διαδικασία συγγραφής / μεταγλώττισης / αποσφαλμάτωσης και εκτέλεσης ενός προγράμματος.
- Τέτοια περιβάλλοντα είναι γνωστά ως Integrated Development Environments (IDEs)



Create a new project

Προγραμματισμός με C#.NET Core

Get started



Clone a repository

Get code from an online repository like GitHub or Azure DevOps



Open a project or solution

Open a local Visual Studio project or .sln file



Open a local folder

Navigate and edit code within any folder



Create a new project

Choose a project template with code scaffolding to get started

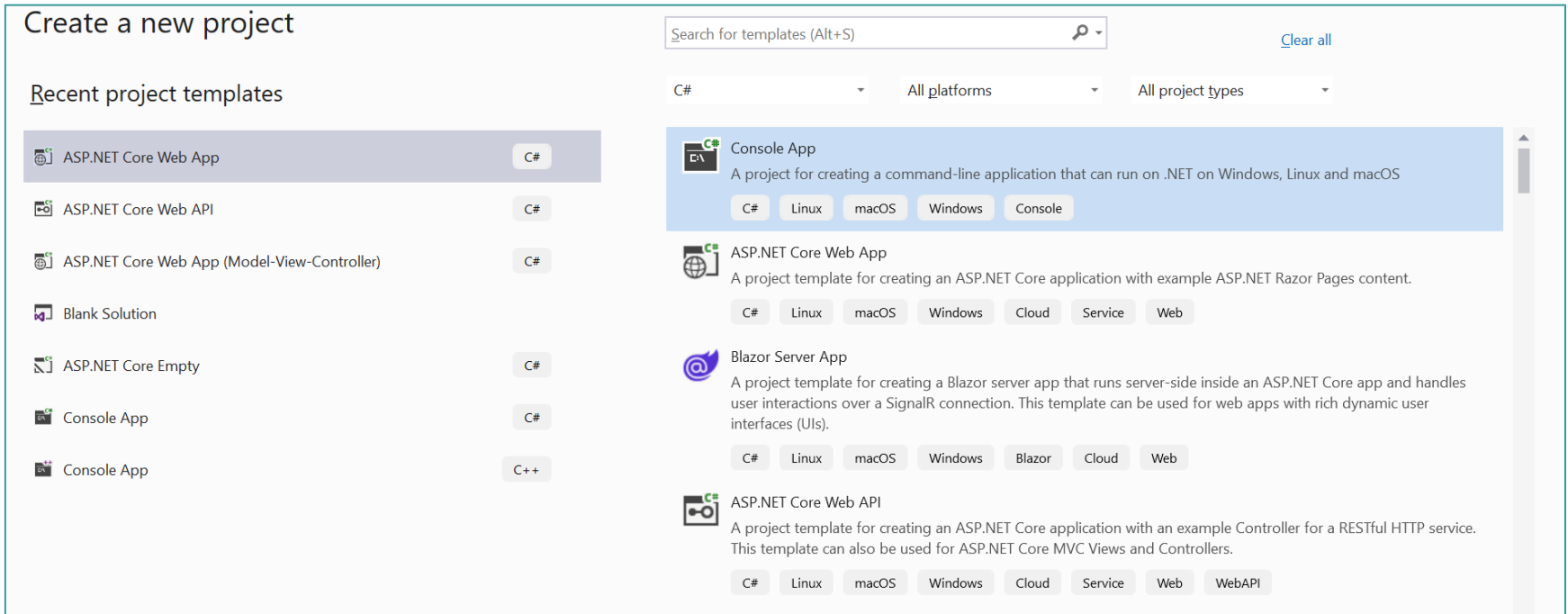
[Continue without code](#) →

- Επιλέγουμε
Create a new
project



Νέο C# Project στο VS

Προγραμματισμός με C#.NET Core



- Αφού ανοίξουμε το VS πρέπει να δημιουργήσουμε ένα νέο Project μέσα στο οποίο θα δημιουργήσουμε κλάσεις της C#
- Δημιουργία project (**Console App**):
 - File -> New -> Project (Επιλέγουμε Console App)



Project & Solution Name

Προγραμματισμός με C#.NET Core

Configure your new project

Console App

C#

Linux

macOS

Windows

Console

Project name

HelloWorld

Location

C:\Users\A8ana\source\repos

...

Solution name ⓘ

CodingFactory6Test

☐

Place solution and project in the same directory

Project will be created in "C:\Users\A8ana\source\repos\CodingFactory6Test\HelloWorld\"



No top-level statements

Προγραμματισμός με C#.NET Core

Additional information

Console App

C#

Linux

macOS

Windows

Console

Framework ⓘ

.NET 8.0 (Long Term Support)

☐ Enable container support ⓘ

Container OS ⓘ

Linux

Container build type ⓘ

Dockerfile

☒ Do not use top-level statements ⓘ

☐ Enable native AOT publish ⓘ

- Απενεργοποιούμε τα Top-Level Statements, τα οποία επιτρέπουν να γράφουμε κατευθείαν το σώμα των μεθόδων χωρίς τις επικεφαλίδες κλάσεων και μεθόδων
- Τα top level statements μπορούν να χρησιμοποιηθούν για testing σκοπούς και όχι σε μεγάλα προγράμματα



Additional information

Console App

C#

Linux

macOS

Windows

Console

Framework ⓘ

.NET 8.0 (Long Term Support)

☐ Enable container support ⓘ

Container OS ⓘ

Linux

Container build type ⓘ

Dockerfile

☒ Do not use top-level statements ⓘ

☐ Enable native AOT publish ⓘ

- Ο κώδικας που παράγεται προς εκτέλεση είναι ενδιάμεσος κώδικας (.exe ή .dll)
- Υπάρχει η δυνατότητα για Ahead-Of-Time compilation ώστε να προ-μεταγλωττιστεί ο κώδικας σε native machine code, χωρίς την ανάγκη JIT που τρέχει χωρίς CLR κατευθείαν στην μηχανή



Solution vs Project

Προγραμματισμός με C#.NET Core

- Το Solution αντιστοιχεί σε φάκελο και μπορεί να περιλαμβάνει πολλά projects (assemblies)
- Το Project είναι η βασική δομική μονάδα που επίσης αντιστοιχεί σε φάκελο
- Σε κάθε Project μπορούμε να έχουμε μία Main



Hello World!

Προγραμματισμός με C#.NET Core

```
Program.cs
C# HelloWorld HelloWorld.Program Main(string[] args)
1 using System;
2
3 namespace HelloWorld
4 {
5     0 references
6     internal class Program
7     {
8         0 references
9         static void Main(string[] args)
10        {
11            Console.WriteLine("Hello, Coding Factory!");
12        }
13    }
14 }
```

- Το πρώτο πρόγραμμα. Αποτελείται από namespace (ίδιο με το όνομα του Project), την Program class και την Main method



Using System

Προγραμματισμός με C#.NET Core

The screenshot shows the Visual Studio IDE with a C# project named 'HelloWorld'. On the left, the 'Solution Explorer' displays a file named 'GlobalUsings.g.cs' which contains the directive 'using System;'. The 'Code Explorer' on the right shows the code in 'Program.cs', which includes the 'using System;' directive at line 1, followed by a namespace declaration 'namespace HelloWorld' at line 3, an internal class 'Program' at line 5, and a static method 'Main' at line 7. The 'Main' method contains a single line of code: 'Console.WriteLine("Hello, Coding Factory!");' at line 9. The code is color-coded according to C# syntax rules.

```
1 using System;
2
3 namespace HelloWorld
4 {
5     0 references
6     internal class Program
7     {
8         0 references
9         static void Main(string[] args)
10        {
11            Console.WriteLine("Hello, Coding Factory!");
12        }
13    }
14 }
```

- Με την `using` εισάγουμε namespaces. Στο .NET 8.0 υπάρχει το `GlobalUsings.g.cs` και έτσι παρόλο που μέχρι και την έκδοση .NET 5.0 έπρεπε να κάνουμε `using System`, εδώ γίνεται έμμεσα. Το `System` είναι ένα namespace του .NET που μας δίνει την κλάση `Console` για να κάνουμε I/O



Namespaces (1)

Προγραμματισμός με C#.NET Core

```
Program.cs
C# HelloWorld HelloWorld.Program Main(string[] args)
1 using System;
2
3 namespace HelloWorld
4 {
5     0 references
6     internal class Program
7     {
8         0 references
9         static void Main(string[] args)
10        {
11            Console.WriteLine("Hello, Coding Factory!");
12        }
13    }
14 }
```

- Τα namespaces είναι αντίστοιχα με τα packages της Java μόνο που δεν αντιστοιχούν κατ' ανάγκη σε φακέλους αλλά είναι λογικοί containers που περιέχουν κλάσεις



Namespaces (2)

Προγραμματισμός με C#.NET Core

- Για παράδειγμα μπορούμε να έχουμε ένα namespace *HelloWorld.Model* που να αντιστοιχεί σε ένα υποφάκελο του Project μας με όνομα Model και να περιέχει διάφορες κλάσεις του Domain Model



Using

- Με την *using* όπως αναφέραμε εισάγουμε namespaces ώστε να μπορούμε να μην γράφουμε το FQN (Fully Qualified Name) της κλάσης ή της μεθόδου
- Με *using static* εισάγουμε static κλάσεις (utility κλάσεις) στο Project μας, π.χ. *using static System.Math;*



Access Modifiers - class

Προγραμματισμός με C#.NET Core

```
Program.cs  + X
C# HelloWorld  HelloWorld.Program  Main(string[] args)
1  using System;
2
3  namespace HelloWorld
4  {
5      0 references
6      internal class Program
7      {
8          0 references
9          static void Main(string[] args)
10         {
11             Console.WriteLine("Hello, Coding Factory!");
12         }
13     }
14 }
```

- Ο χαρακτηρισμός *internal* που είναι default για τις κλάσεις είναι χαρακτηρισμός εμβέλειας που δίνει πρόσβαση σε επίπεδο Assembly (Project). Ο χαρακτηρισμό των outer κλάσεων μπορεί να είναι *public* ή *internal*



Main method (1)

Προγραμματισμός με C#.NET Core

```
Program.cs
C# HelloWorld HelloWorld.Program Main(string[] args)
1 using System;
2
3 namespace HelloWorld
4 {
5     0 references
6     internal class Program
7     {
8         0 references
9         static void Main(string[] args)
10        {
11            Console.WriteLine("Hello, Coding Factory!");
12        }
13    }
14 }
```

- Η Main μέθοδος είναι όπως και στην Java, το entry point, από όπου ξεκινά και εκτελείται ένα πρόγραμμα C#
- By default, τα class members στη C# είναι private (όπως και στη C++, ενώ στην Java είναι private-package).



Main method (2)

```
Program.cs [X]
C# HelloWorld HelloWorld.Program Main(string[] args)
1 using System;
2
3 namespace HelloWorld
4 {
5     0 references
6     internal class Program
7     {
8         0 references
9         static void Main(string[] args)
10        {
11            Console.WriteLine("Hello, Coding Factory!");
12        }
13    }
14 }
```

- Παρατηρούμε επίσης, ότι το όνομα της Main ξεκινά με κεφαλαίο και κάθε επόμενη λέξη με κεφαλαίο, όπως κατά σύμβαση όλα τα ονόματα μεθόδων στη C# (Pascal Case)
- Το ίδιο ισχύει και για τα ονόματα των κλάσεων



Main method (3)

Προγραμματισμός με C#.NET Core

```
Program.cs
C# HelloWorld HelloWorld.Program Main(string[] args)
1 using System;
2
3 namespace HelloWorld
4 {
5     0 references
6     internal class Program
7     {
8         0 references
9         static void Main(string[] args)
10        {
11            Console.WriteLine("Hello, Coding Factory!");
12        }
13    }
14 }
```

- Η επικεφαλίδα της Main είναι `static void Main(string[] args)` όπως και στην Java
- Εναλλακτικά η Main μπορεί να επιστρέφει `int` όπως και στη C και να επιστρέφει μία τιμή προς το ΛΣ (0 αν όλα πάνε καλά, και $\neq 0$ σε περίπτωση λάθους)



Σώμα της Main

Προγραμματισμός με C#.NET Core

```
Program.cs [X]
C# HelloWorld HelloWorld.Program Main(string[] args)
1 using System;
2
3 namespace HelloWorld
4 {
5     0 references
6     internal class Program
7     {
8         0 references
9         static void Main(string[] args)
10        {
11            Console.WriteLine("Hello, Coding Factory!");
12        }
13    }
14 }
```

- Μέσα στο σώμα της Main που προσδιορίζεται από τα {} γράφουμε εντολές
- Για output η C# μας δίνει την System.Console.WriteLine() (αντίστοιχη της System.out.println() στην Java)



WriteLine

Προγραμματισμός με C#.NET Core

```
Program.cs [X]
C# HelloWorld HelloWorld.Program Main(string[] args)
1 using System;
2
3 namespace HelloWorld
4 {
5     0 references
6     internal class Program
7     {
8         0 references
9         static void Main(string[] args)
10        {
11            Console.WriteLine("Hello, Coding Factory!");
12        }
13    }
14 }
```

- Η WriteLine() εκτυπώνει στο std output (κονσόλα) το αλφαριθμητικό μέσα στα double quotes και γράφει και ένα χαρακτήρα αλλαγής γραμμής



Write - WriteLine

Προγραμματισμός με C#.NET Core

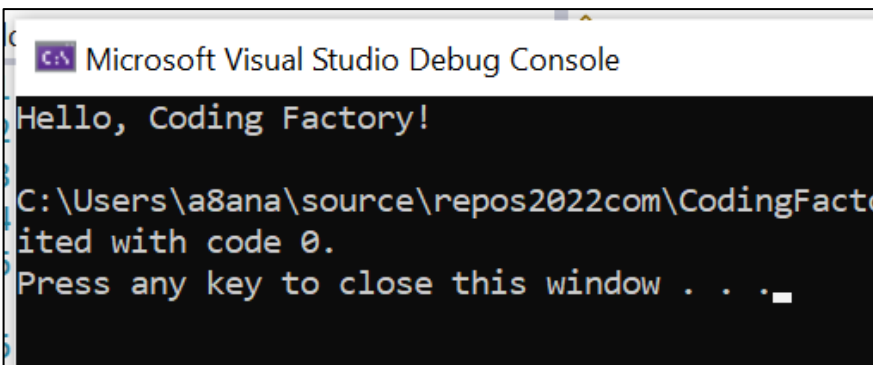
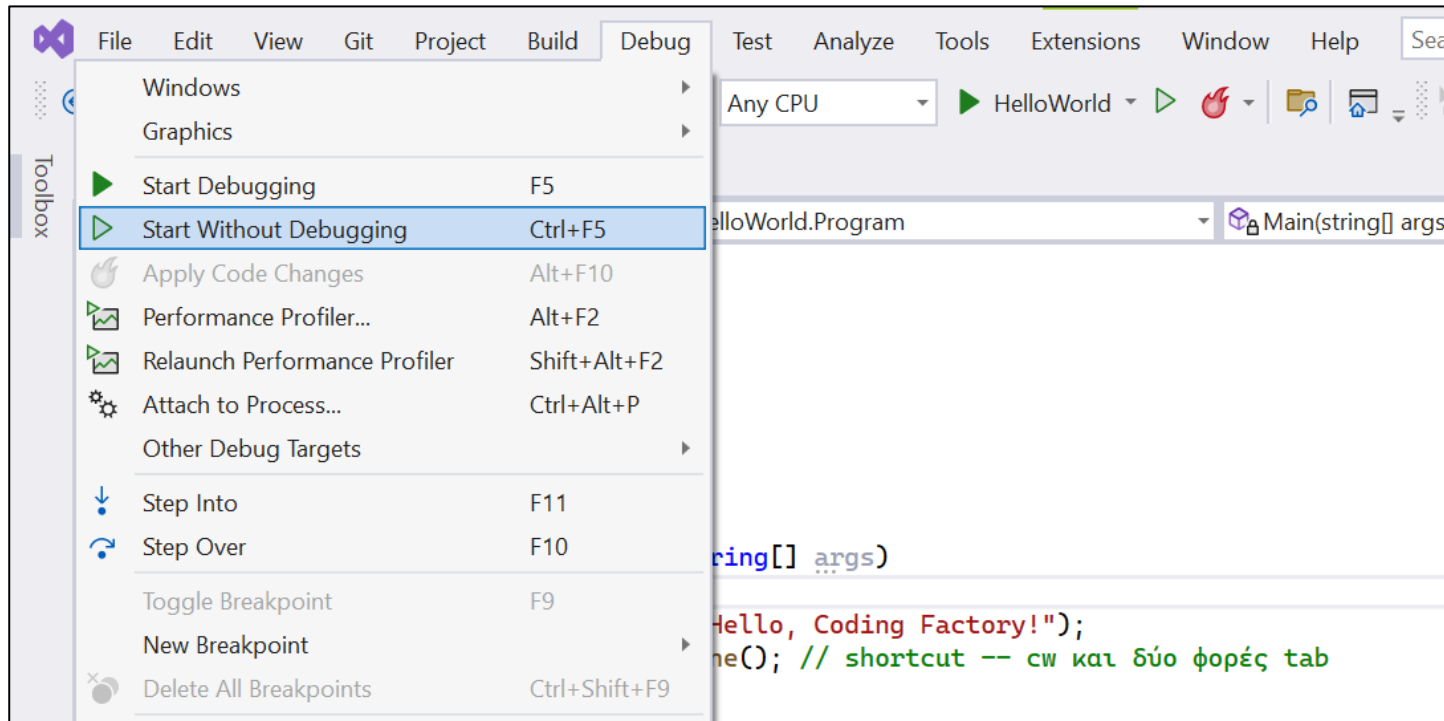
```
1  using System;
2
3  namespace HelloWorld
4  {
5      0 references
6      internal class Program
7      {
8          0 references
9          static void Main(string[] args)
10         {
11             Console.Write("Hello, Coding Factory!");
12             Console.WriteLine(); // shortcut -- cw και δύο φορές tab
13         }
14     }
15 }
```

- Το ίδιο όπως πριν με δύο εντολές. Το shortcut για την WriteLine() είναι cw και tab



Run

Προγραμματισμός με C#.NET Core



- Ctrl+F5 (Start without debugging)
- F5 (Start with debugging)





Visual Studio Code

Προγραμματισμός με C#.NET Core




C# Snippets

Jorge Serrano |  514,912 installs |  (7) | Free

C# Snippets for Visual Studio Code

[Install](#)

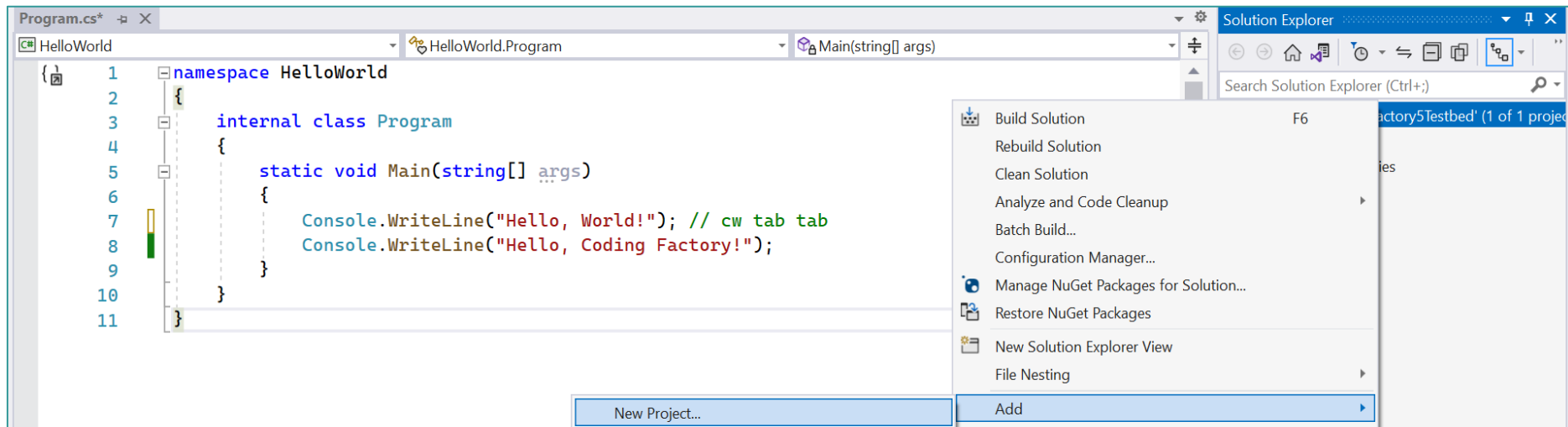
[Trouble Installing?](#) 

- Στο VSC υπάρχει το C# Snippets για shortcuts



Add new Project

Προγραμματισμός με C#.NET Core



- Για να προσθέσουμε νέο Project κάνουμε δεξί κλικ πάνω στο Solution και προσθέτουμε νέο Project



Solution Properties

Προγραμματισμός με C#.NET Core

- Όταν έχουμε περισσότερα από ένα projects ορίζουμε να εκτελείται με το current selection

Solution 'CodingFactory5Testbed' Property Pages

Configuration: N/A Platform: N/A Configuration Manager...

Common Properties

- Startup Project
- Project Dependencies
- Code Analysis Settings
- Debug Source Files
- Configuration Properties

☒ Current selection

☐ Single startup project

☐ Multiple startup projects:

Project	Action
HelloWorld	None

OK Ακυρο Εφαρμογή