

C# .NET

Laborator 1

Sa ne cunoastem



Setup statie de lucru

- Sistem de operare: Windows 10 (recomandat)
- Source Control Management: GIT <https://git-scm.com/downloads>
- Cont GitHub <https://github.com/>
- Mediu Integrat de Dezvoltare (IDE): Visual Studio 2019 Community Edition - vezi https://my.visualstudio.com/Downloads?q=visual%20studio%202019&wt.mc_id=o~msft~vscom~older-downloads
 - Cerinte de sistem:
 - <https://docs.microsoft.com/en-us/visualstudio/releases/2019/system-requirements>
- Vor urma și alte tool-uri (SQL Server, VS2022, tool-uri și pachete, etc), pe masura ce avansam prin materialul de curs

Setup statie de lucru

- [Ghid instalare GIT](#)
- [Ghid instalare Visual Studio Community 2019](#)
- [Ghid instalare Visual Studio Code](#)

In timp ce instalam

- Vom povesti câte puțin despre:
 - Ce înseamnă sa fii programator.
 - Cum funcționează un program, cum se ajunge de la cod sursa la cod executabil și de ce au fost inventate limbajele de programare managed
 - Paradigme de dezvoltare software (cum ne gandim si structuram codul)
 - Limbajul C# si platforma .NET

Obiectivele cursului

- Sa invatati programare
 - Gandire analitica
 - Algoritmica
 - OOP
 - Clean code
 - Asp .NET Web Api
 - EF Core
- Sa invatati C#
- Sa puteti participa cu success la un interviu si sa fiti angajati
 - CV
 - Simulare de interviu*
- Sa invatati sa invatati

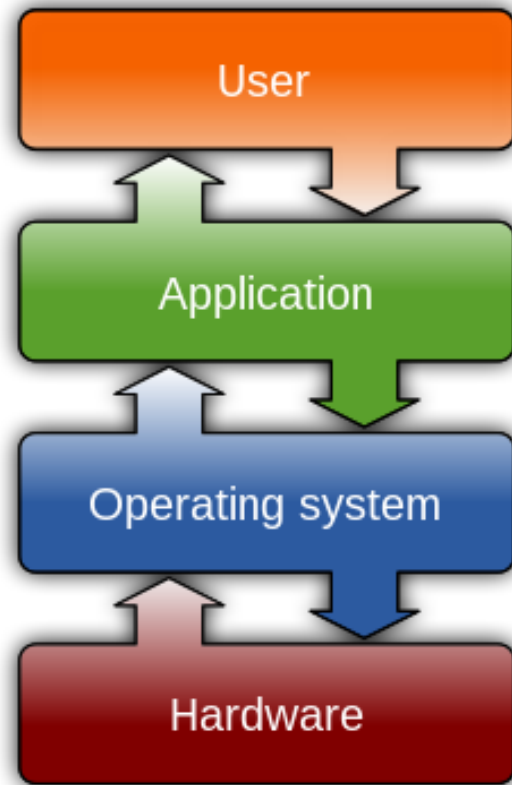
Ce este un programator

- Definiția clasică: o persoana care scrie cod, folosind unul sau mai multe limbaje de programare și / sau unelte software specifice, pentru a realiza aplicații care rezolva o anumită problemă, sau care automatizează un anumit flux de lucru
- Definitii alternative:
 - Dresor de calculatoare
 - Matematician esuat

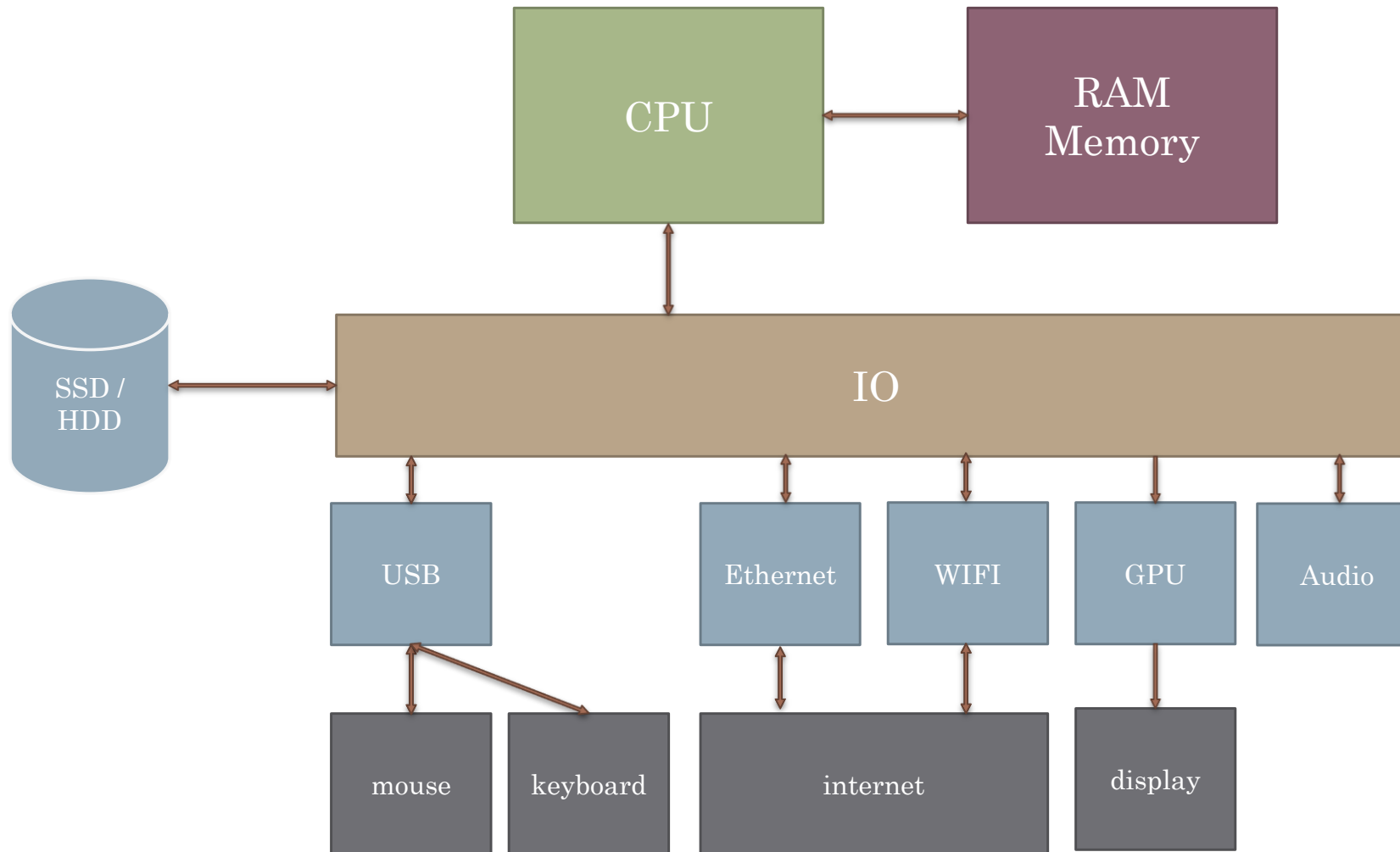
Ce este un programator – diagrama



Cum functioneaza un
program

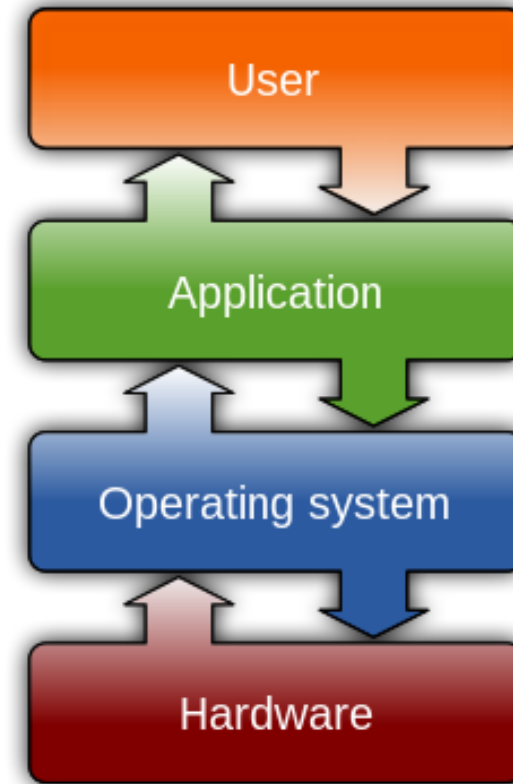


Hardware



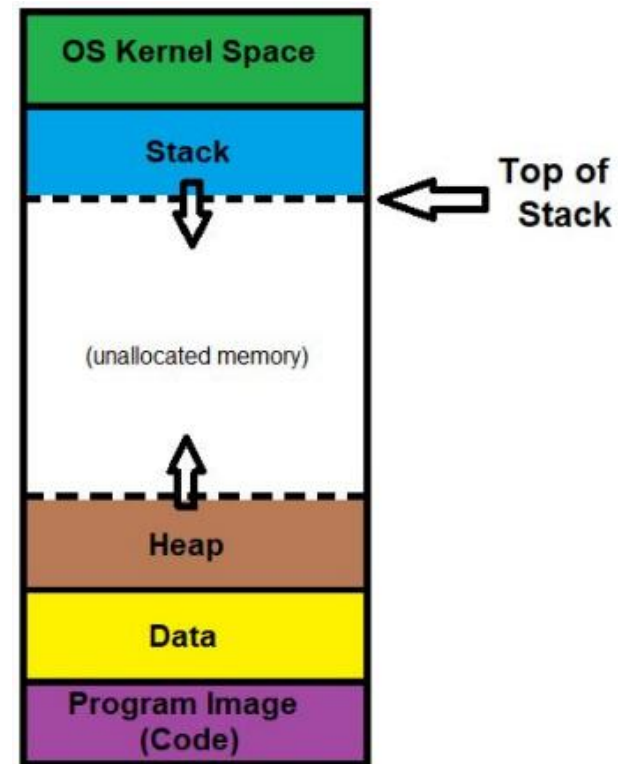
Sistemul de operare

- Gestioneaza resursele hardware
- Implementează politici de securitate pentru utilizatori și / sau aplicații
- Permite lansarea în execuție a aplicațiilor și gestionează execuția lor
- Lanseaza aplicatiile
 - La cererea utilizatorului
 - Ca urmare a unor evenimente



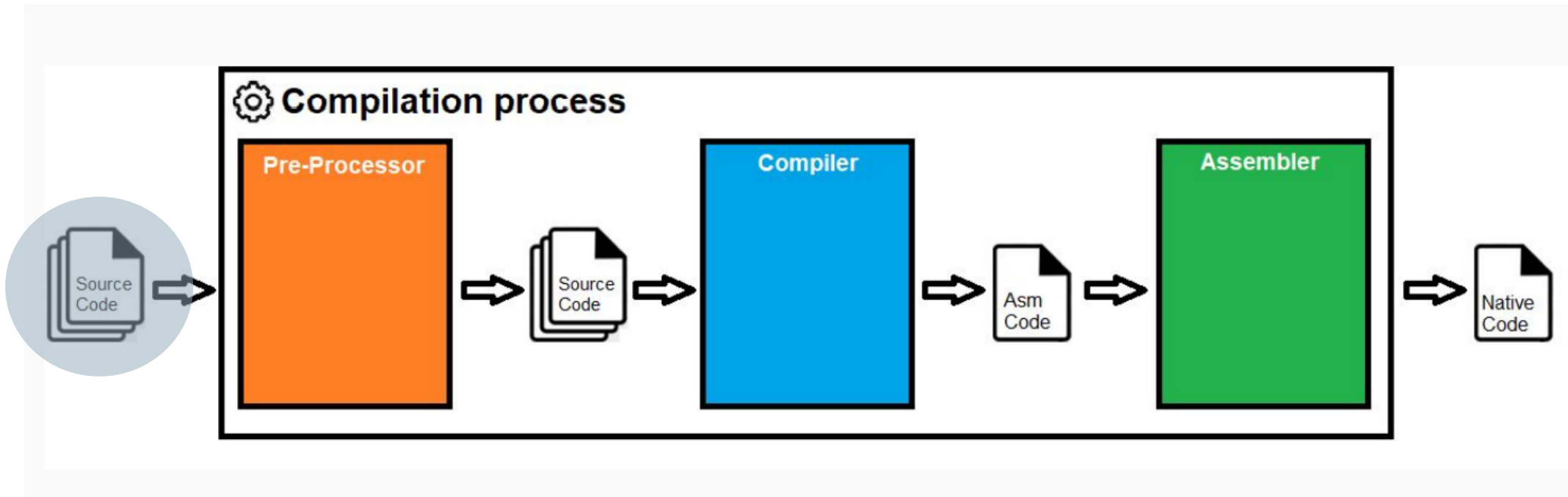
Executia

- La executarea unei aplicații
 - OS-ul citește de pe disk conținutul fișierului executabil (exe),
 - extrage codul în format nativ (direct executabil de către CPU)
 - încarcă codul în memoria de lucru (RAM) și îl
 - lansează programul în execuție (sub forma unui process)
- În memoria de lucru fiecărui process îi este alocată o zonă de memorie distinctă
- Fiecare process are o anumită structura în memorie - vezi imaginea de alături pentru un exemplu simplificat



De la cod sursa la cod
executabil

Limbaje de nivel inalt - compilarea



Limbaje de nivel inalt

- C#, Java, Python , Javascript, Typescript, Pascal,C, C++, Cobol, Fortran etc
- Ascund complexitatea și detaliile specifice (abstractizeaza) codului nativ, prin oferirea unor elemente ușor inteligibile (instrucțiuni, operatori, etc)
- Cat mai aproape de limbajul natural (limba engleza)
- Fac codul mai lizibil
- Fac codul mai mentenabil
- Simplifica procesul de dezvoltare al unei aplicații

Example: Sorting

```
var sortedStudents = from s in studentList
                      orderby s.StandardID, s.age
                      select new {
                          StudentName = s.StudentName,
                          Age = s.age,
                          StandardID = s.StandardID };

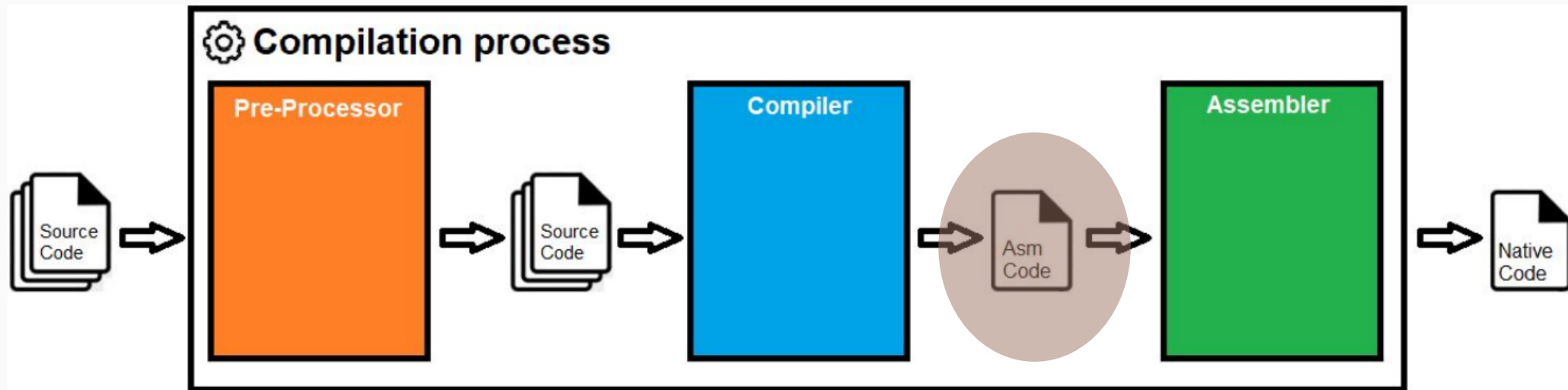
sortedStudents.ToList().ForEach(s => Console.WriteLine("Student Name: {0}, Age: {1}, StandardID: {2}", s.StudentName, s.Age, s.StandardID));
```

Try it

Output:

```
Student Name: Ron, Age: 21, StandardID: 0
Student Name: John, Age: 18, StandardID: 1
Student Name: Steve, Age: 21, StandardID: 1
Student Name: Bill, Age: 18, StandardID: 2
Student Name: Ram, Age: 20, StandardID: 2
```


Limbaje de nivel inalt - compilarea



Limbaaj de asamblare

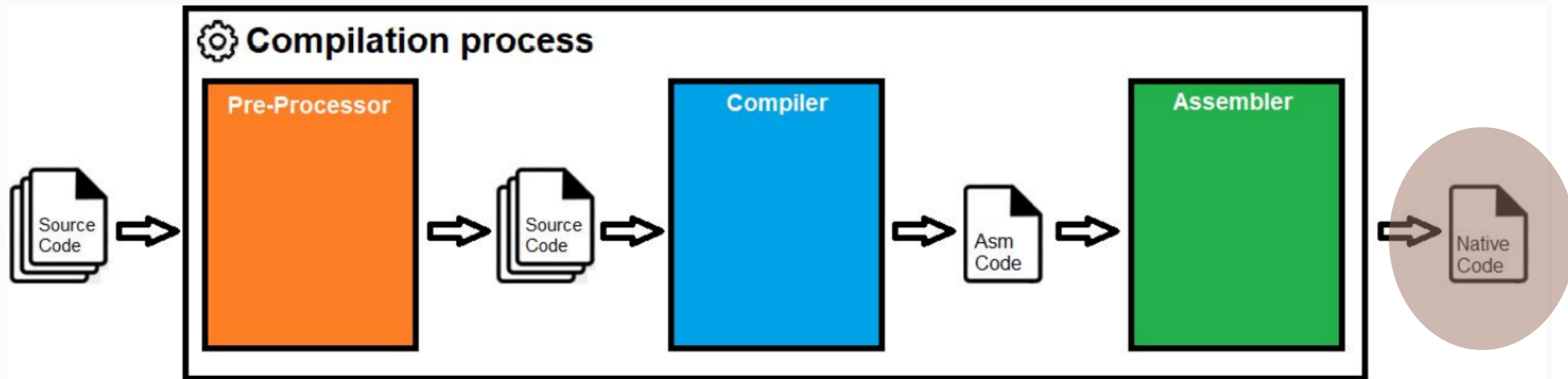
- Cea mai apropiata reprezentare cât de cât inteligibilă (de către om) a codului nativ
- Defineste niște simboluri, așa încat instrucțiunile CPU să fie mai ușor de identificat / recunoscut
- Suporta comentarii in cod, etichete
- Pentru a fi executat, trebuie convertit în cod nativ, lucru realizat de un tool denumit "assembler"

```
080483b4 <main>:
80483b4: 55          push    %ebp
80483b5: 89 e5       mov     %esp,%ebp
80483b7: 83 e4 f0    and     $0xffffffff0,%esp
80483ba: 83 ec 20    sub     $0x20,%esp
80483bd: c7 44 24 1c 00 00 00 movl    $0x0,0x1c(%esp)
80483c4: 00
80483c5: eb 11       jmp     80483d8 <main+0x24>
80483c7: c7 04 24 b0 84 04 08 movl    $0x80484b0,(%esp)
80483ce: e8 1d ff ff ff call    80482f0 <puts@plt>
80483d3: 83 44 24 1c 01 addl    $0x1,0x1c(%esp)
80483d8: 83 7c 24 1c 09 cmpl    $0x9,0x1c(%esp)
80483dd: 7e e8       jle     80483c7 <main+0x13>
80483df: b8 00 00 00 00 mov     $0x0,%eax
80483e4: c9         leave
80483e5: c3         ret
80483e6: 90         nop
80483e7: 90         nop
80483e8: 90         nop
80483e9: 90         nop
80483ea: 90         nop
```

Limbaaj de asamblare – programator tipic



Limbaje de nivel inalt - compilarea

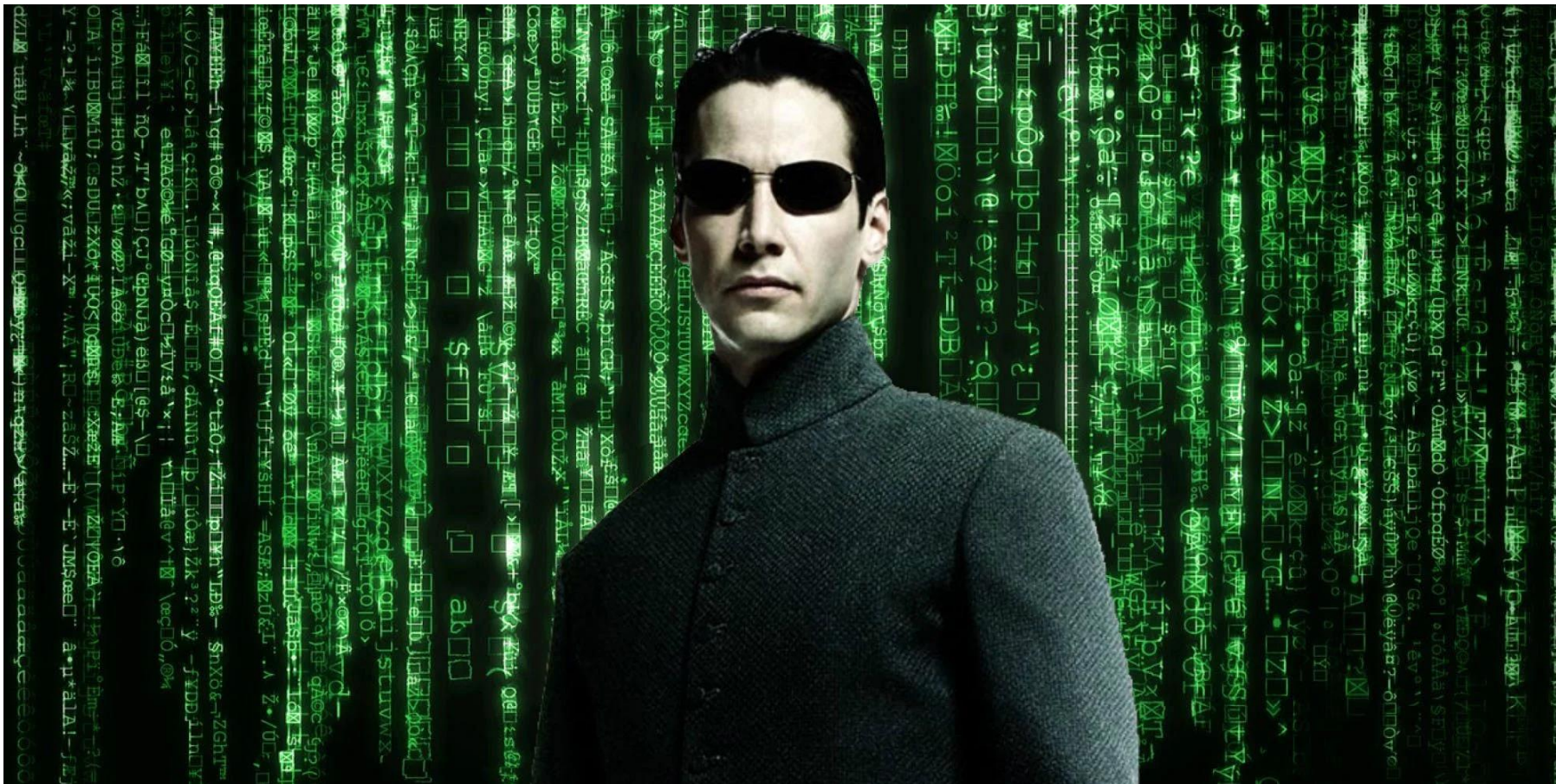


Cod nativ / cod masina

- Instrucțiuni direct executabile de către CPU.
- Fiecare instrucțiune are o anumită valoare binară 10100011 (în imagine se vad valorile simplificate in format hexa)
- Exemple instrucțiuni:
 - încarcă date în regiștrii CPU,
 - sari la o anumită adresa,
 - executa o anumită operațiune aritmetica, etc - vezi și x86 instruction set
- Instrucțiunile sunt executate în ordinea în care apar
- Fluxul de control al programului este implementat prin operațiunile de salt (condiționat, sau necondiționat)
- Greu (imposibil?) de citit / înțeles de către om

```
0749eb90 f0 32 7d 60 95 48 d0 62 08 80 4b 67 b4 4a 21 dc
0749eba0 80 3f 6c dd 4a f5 a3 d4 ce 32 8d e4 21 d7 a5 5a
0749ebb0 92 93 4b f1 ca 0a ce 3c b9 14 20 a5 00 a4 4a 3e
0749ebc0 bd 4b 8c b4 d1 90 2b 25 a9 c8 f4 c8 10 85 fb d6
0749ebd0 fc 2a 1f c6 8a 7f 25 e7 47 f4 95 01 e2 d7 82 fe
0749ebe0 22 95 fa 8e 49 e4 50 98 d3 84 95 a7 97 1d 97 92
0749ebf0 25 32 9f 90 0c a9 07 73 c2 2b 49 06 4c 1a 26 69
0749ec00 b2 75 3e 20 db 65 bf 22 68 cf 29 1b 8a 65 8d 54
0749ec10 91 ba 33 f3 05 59 07 39 cd 43 96 6f 5d 88 bb 7a
0749ec20 aa ae d2 04 b1 c6 33 25 8c 68 f7 c7 79 23 ef 66
0749ec30 7a aa 41 e7 99 55 1d 46 79 64 2a 6c 1f a9 64 63
0749ec40 ef f9 87 72 3f d9 5a 9f 48 0d 92 96 72 0d 1b a4
0749ec50 a6 2e 08 b0 96 cc e6 37 88 f0 57 32 3b 21 6d d9
0749ec60 e4 6b f1 ef 14 25 65 e3 3c b3 ee 60 bc a4 ea 44
0749ec70 64 49 0d 59 0b 45 3f f0 75 a4 24 be 41 f5 52 ad
0749ec80 32 65 33 4d 9c 83 8e 97 69 57 f2 5d 72 93 dd b1
0749ec90 d0 c6 dc c8 43 89 6e 1e 8b d9 2e 67 52 3e 26 3f
0749eca0 46 cc 92 a7 e1 f3 af 9c c8 b3 17 fe ff 8a bb 7a
0749ecb0 f6 e9 99 6d 8b 24 dc 84 97 67 b6 d5 5b 73 a6 fc
0749ecc0 50 a6 cf fe 92 7d c3 2f 2e 7e e8 b7 8f 9b 71 5f
0749ecd0 b0 43 79 5c f1 63 9d b7 2f 7e b1 f3 f6 87 5f b0
0749ece0 64 84 86 98 59 f7 d2 96 42 28 5a 96 8e d1 17 4f
0749ecf0 f4 2d a6 94 06 0f fb 57 83 fe 60 59 8e 32 70 23
0749ed00 c1 8a 98 43 0b 90 26 24 03 ce 3d 21 79 0b 75 f9
```


Cod nativ – programator tipic



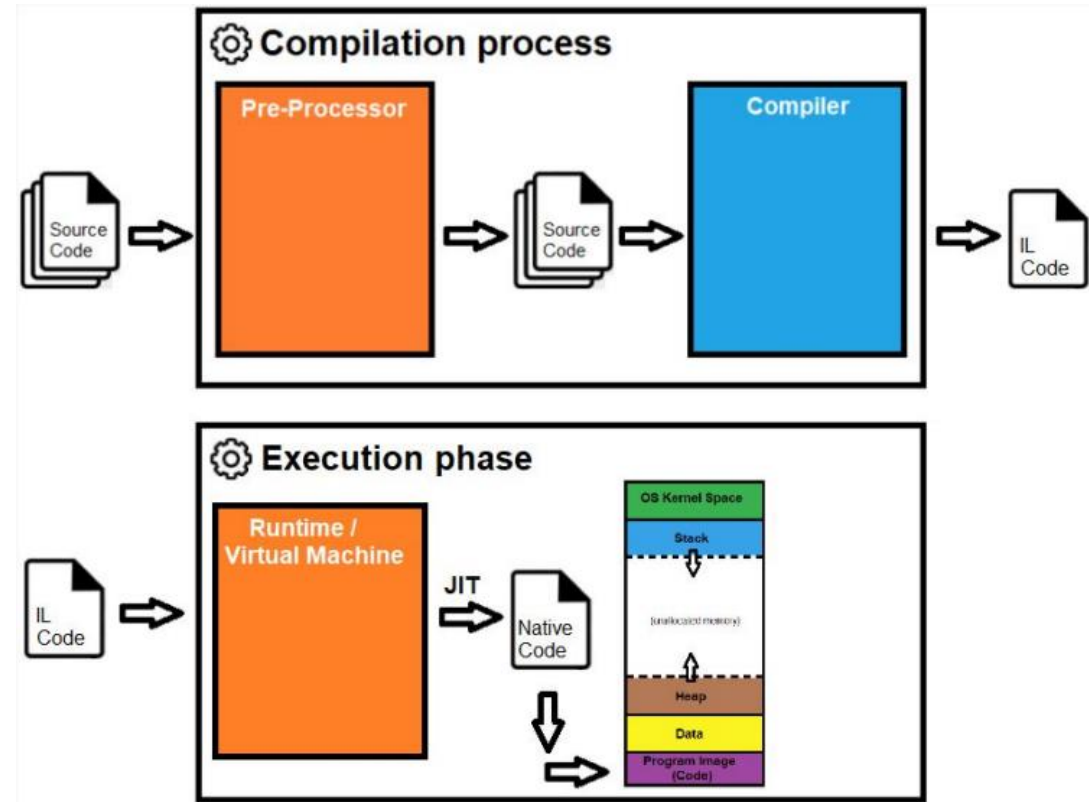
Limbaje managed

Limbaje native - probleme

- Portabilitate
 - Trebuie compilate pentru fiecare arhitectura hardware si sistem de operare in parte
 - Windows
 - Linux
 - MacOS
 - Android
 - X86, ARM, cuptor cu microunde
- Memory management
 - Resursele alocate dynamic – memory leaks
- Type checking
- Reference checking

Limbaje managed - solutii

- Portabilitate
 - Virtual machine
- Memory management
 - Garbage collector
- Type checking
 - At runtime
- Reference checking
 - At runtime



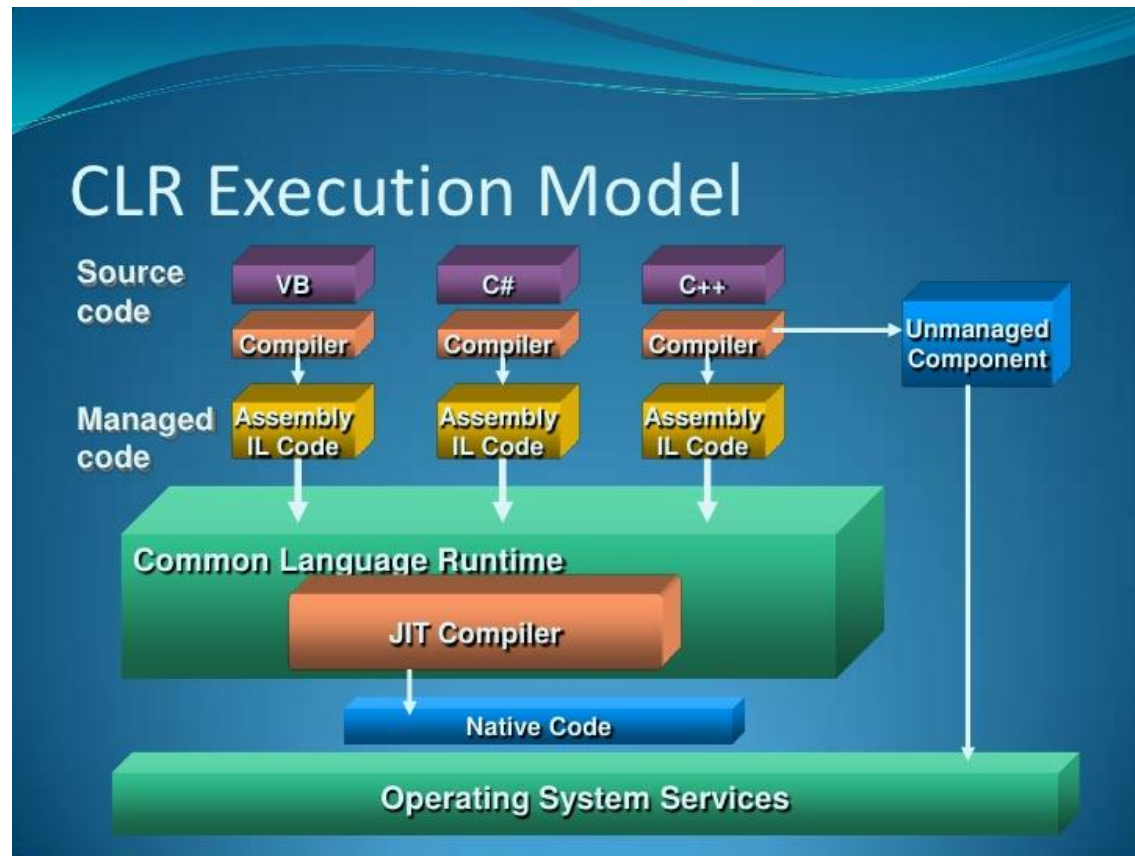
.Net framework

.Net platform

- Framework – colectie de tool-uri si librarii software
- Suporta mai multe limbaje de programare

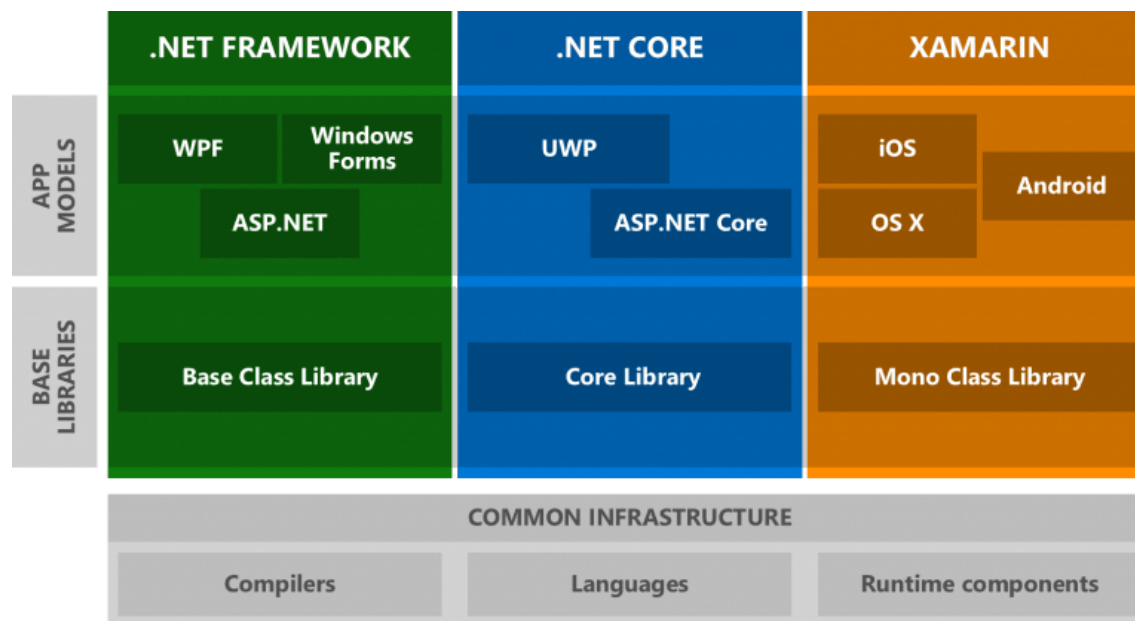
Componente ale framework-ului

- CIL – common interface language
- CLR – masina virtuala
 - Common language **runtime**
 - JIT – just in time compiler
 - Transpune bytecod-ul in cod nativ in timpul rularii programului
 - AOT – ahead of time compiler
 - Suportat doar pe windows



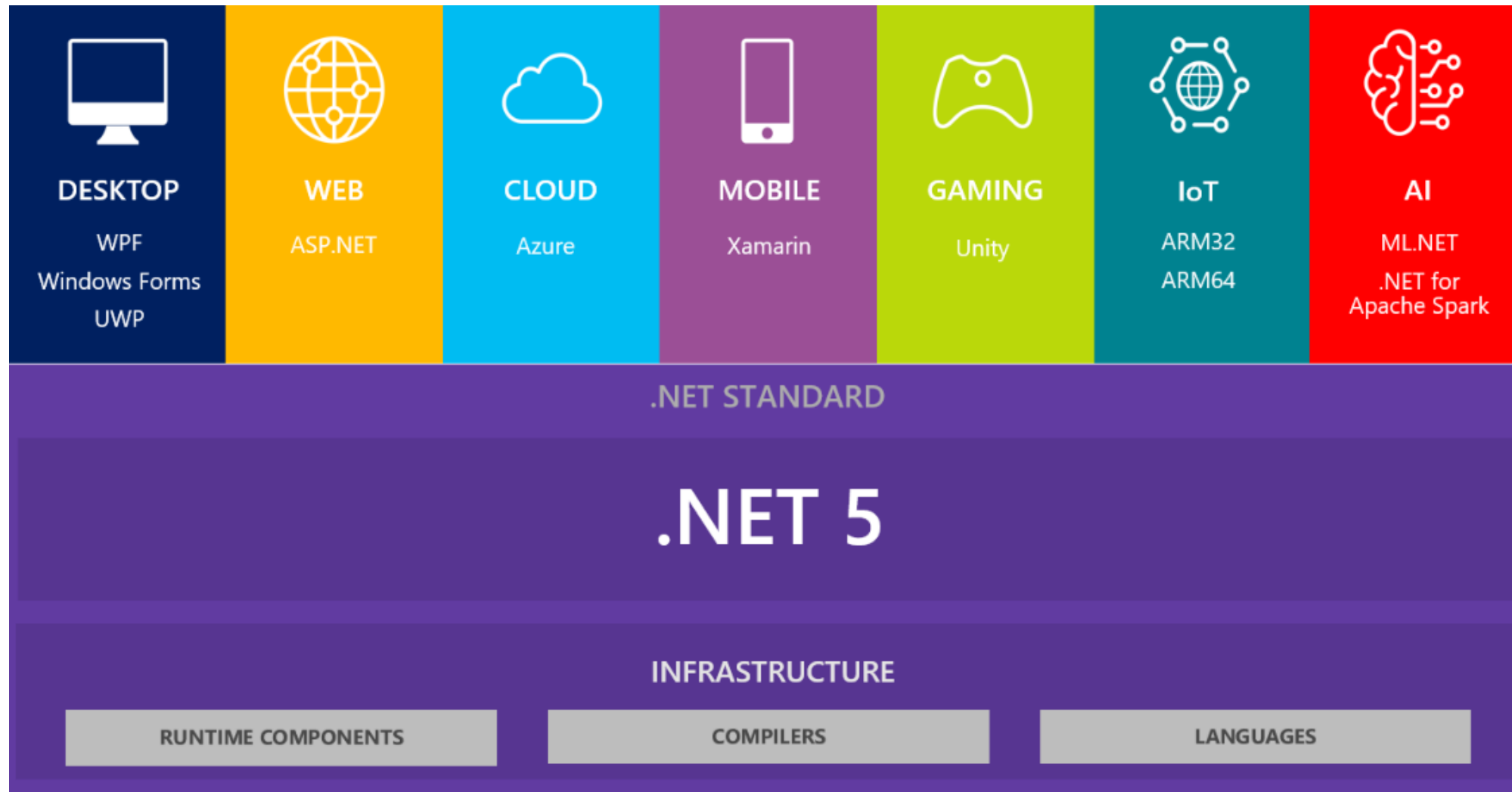
.Net platform – implementari

- .NET framework
 - 1995 – prima implementare
 - tehnologii desktop
 - tehnologii web de generatii mai vechi
 - Windows-specific
 - Closed source
- .Net core
 - 2014
 - Tehnologii web
 - Portabil (windwos, linux, macOS)
 - Mai performant
 - Open-source
 - Dezvoltat in mare masura de Microsoft
- Versiuni diferite
 - .net framework 4.5
 - .net core 3.1
- Limbajul C# - versionare
 - .net core 3.x – c# 8.0
 - .net framework – c# 7.3

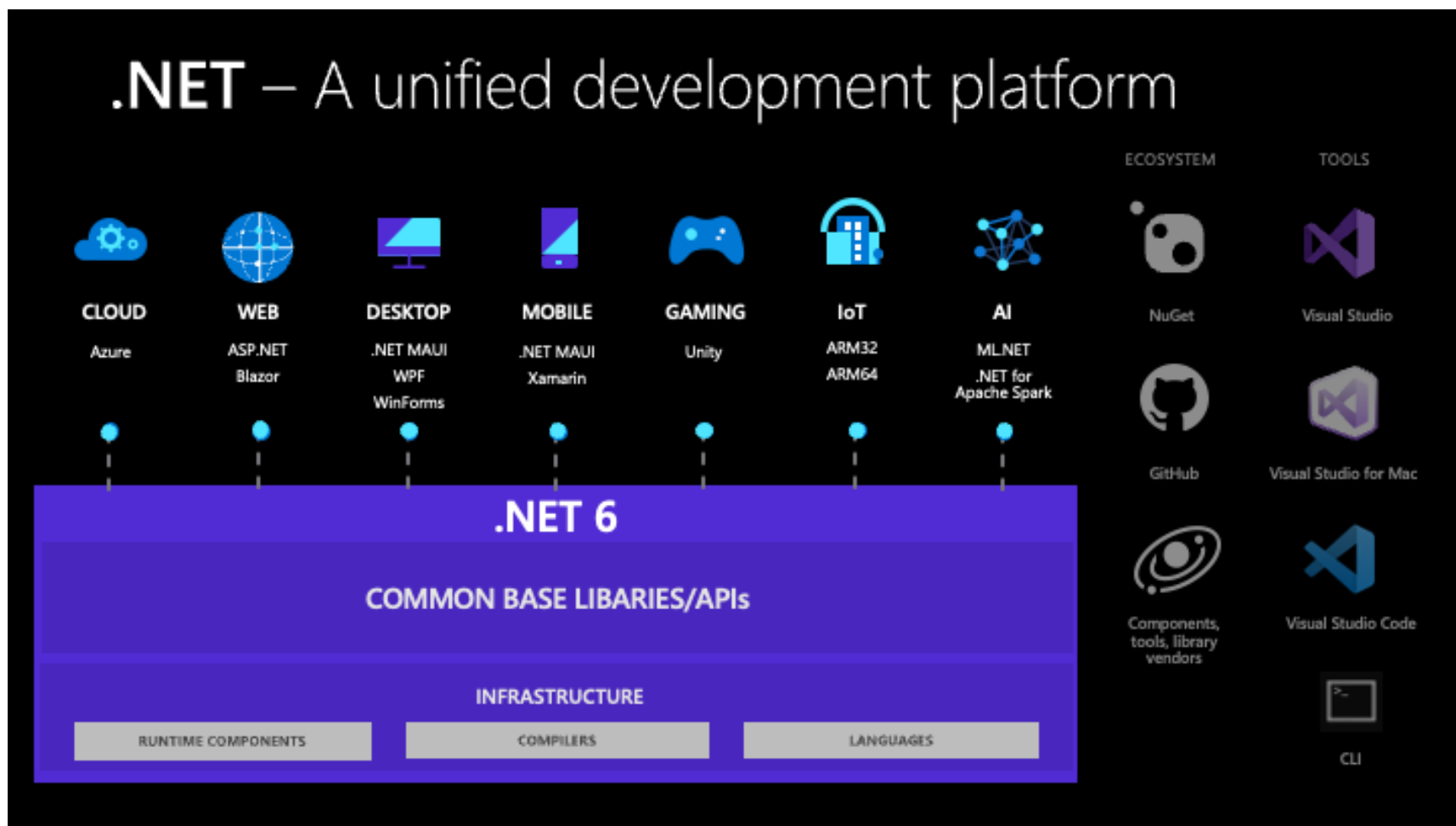


<https://www.youtube.com/watch?v=4olO9UjRiww>

.Net 5



.Net 6



.NET 5.0, .NET 6.0, .Net 7.0

- Inlocuiesc .net framework, xamarin si .net core
- Dezvoltate din .net core
- Unifica cele trei framework-uri
- C#9, C# 10,c# 11

C# - introdurre

Limbajul C#

- Relativ modern: anul 2000, Microsoft;
- Standardizat: ECMA-334 (v1 2002, v5 2017), ISO/IEC 23270
- De nivel înalt (nivel înalt de abstractizare față de codul mașina)
- Managed: rezultatul compilării este un bytecode (IL) rulat de un runtime software (CLR).
 - Permite însă și compilare nativă (doar pe Windows momentan).
- Multi-paradigma:
 - imperativ / OOP,
 - programare declarativă - funcțională (în special prin intermediul expresiilor lambda)
- Generalist: permite dezvoltarea oricărui tip de aplicație
- Parte a platformei .NET;

Primul program

Hello world

- Creearea proiectului
 - Mediul de lucru
- Git repository
- Citirea de la tastatura

Push to a new remote

GitHub

Other

Existing remote

Local only

Initialize a local Git repository

Local path C:\User\source\repos\

Create a new GitHub repository

Account (GitHub)

Owner

Repository name ConsoleApp4

Description Enter the description of the repository <Optional>

☒ Private repository

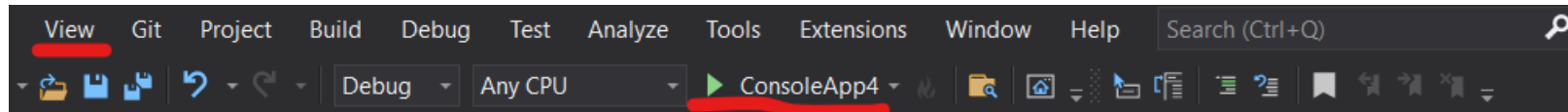
Push your code to GitHub

https://github.com/[redacted]/ConsoleApp4

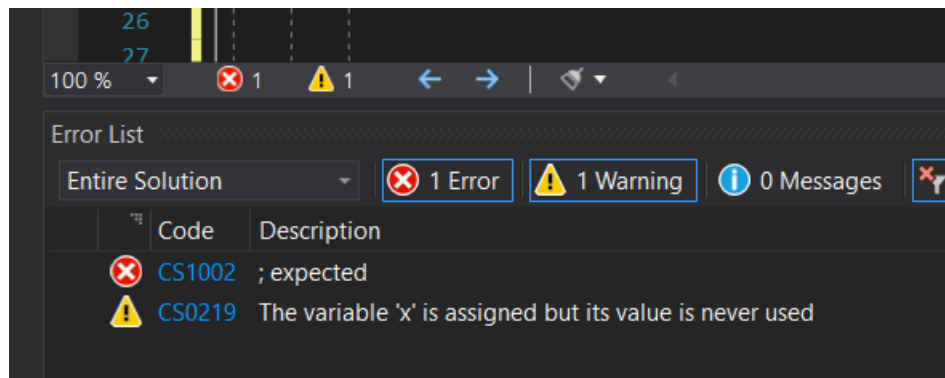
Create and Push Cancel

```
1  using System;
2
3  namespace ConsoleApp4
4  {
5      0 references
6      class Program
7      {
8          0 references
9          static void Main(string[] args)
10         {
11             Console.WriteLine("Hello World!");
12         }
13     }
```

Mediul de lucru Visual Studio



- Solution Explorer – View->Solution explorer
- Run Button – cu verde 😊
- Git Changes – View->Git Changes
- Error list – View->Error List



Mediul de lucru

Inceput de
code block

comentarii

Sfarsit de
code block

```
namespace ConsoleApp5
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            // this is a comment
            /*
             this is
             a multiline
             comment
            */
            Console.WriteLine("Hello World!");
        }
    }
}
```

Codul va rula aici

- Erori, exceptii

Cuvinte cheie

- Cuvant cheie: nume rezervat, care are o semnificatie speciala pentru limbajul C# si care nu poate fi in mod uzual folosit ca identificator.
- Cuvinte cheie C#
 - <https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/>
- **; Instructiunile c# sunt delimitate prin intermediul caracterului “;”**

Identificatori

- Nume pentru componente ale aplicatiilor - clasele/metodele/variabilele/etc
- Numele trebuie sa fie sugestive!
 - *"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."* – Martin Fowler
 - [Robert Martin – Clean Code](#)

```
var x = 3;  
var yy = 4;  
  
int p = x * yy;  
  
Console.WriteLine("Aria dreptunghiului este " + p);
```

```
var lungime = 3;  
var latime = 4;  
  
int ariaDreptunghiului = lungime * latime;  
  
Console.WriteLine("Aria dreptunghiului este " + ariaDreptunghiului);
```

- Casing - C#
 - UPPERCASE, lowercase
 - PascalCase - functii, nume de clase, proprietati, etc
 - camelCase – variabile locale, membri private
 - CONSTANT_NAMING_CONVENTION – doar pt. constante
 - skewer-case-identifiers – Angular/Typescript naming convention

Variabila


- identificador al unei zone de memorie
- camel case

```
/*  
tip: int  
nume: latime  
valoare 4  
*/  
int latime = 4;
```

```
// declararea variabilei  
int lungime;  
// initializarea variabilei  
lungime = 3;
```

```
// declarare si initializare  
int latime = 4;
```

```
// declarare si initializare  
// tipul este implicit  
// pentru ca se poate  
var inaltime = 6;
```

 readonly struct System.Int32
Represents a 32-bit signed integer.

Scope-ul variabilei

- class scope
- Function scope
- Code block scope
 - liniile de cod dintre o pereche de accolade {}
- Regula generala {}

```
static void Main(string[] args)
{
    {
        int lungime = 4;
        {
            lungime = 5;
        }
    }
    lungime = 30;
}
```

Tipuri de date

Numere intregi

Int32 - int

- Numere intregi cu semn, in intervalul [`int.MinValue` -> `int.MaxValue`]
- Operatori
 - + adunare
 - - scadere
 - * inmultire
 - / catul impartirii - quotient
 - % restul impartirii - remainder
 - +a valoarea numerica
 - -a opusul numeric
 - == comparare egalitate
 - != diferit
 - < > mai mic, mai mare
 - >=, <=

Int32 - int

- Functii utile
 - [int.Parse](#) – transforma un sir de caractere intr-un numar intreg
- Overflow – valoarea unui intreg depaseste `int.MaxValue`
 - Google : check for overflow in C#
 - <https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/>
- Precedenta operatorilor
 - -X,+X
 - *,/,%
 - +, -
- Google [C# operator precedence](#)
 - [Shortcut](#)
- Cand aveti dubii – folositi paranteze rotunde – ()

Alte tipuri intregi

byte	System.Byte	1 Byte	Numere intregi in intervalul 0 - 255 ($0 \rightarrow 2^8 - 1$)
sbyte	System.SByte	1 Byte	Numere intregi cu semn, in intervalul -128 -> 127 ($-2^7 \rightarrow 2^7 - 1$)
short	System.Int16	2 Bytes (=16 biti)	Numere intregi cu semn, in intervalul -32,768 -> 32,767 ($-2^{15} \rightarrow 2^{15} - 1$)
ushort	System.UInt16	2 Bytes (=16 biti)	Numere intregi fara semn, in intervalul 0 -> 65535 ($0 \rightarrow 2^{16} - 1$)
int	System.Int32	4 Bytes (=32 biti)	Numere intregi cu semn, in intervalul -2,147,483,648 -> 2,147,483,647 ($-2^{31} \rightarrow 2^{31} - 1$)

Alte tipuri intregi

uint	System.UInt32	4 Bytes (=32 biti)	Numere intregi fara semn, in intervalul 0 -> 4,294,967,295 (0 -> $2^{32} - 1$)
long	System.Int64	8 Bytes (=64 biti)	Numere intregi cu semn, in intervalul -922337203685477508 -> 922337203685477507 (-2^{63} -> $2^{63} - 1$)
ulong	System.UInt64	8 Bytes (=64 biti)	Numere intregi fara semn, in intervalul 0 -> 18446744073709551615 (0 -> 2^{64})

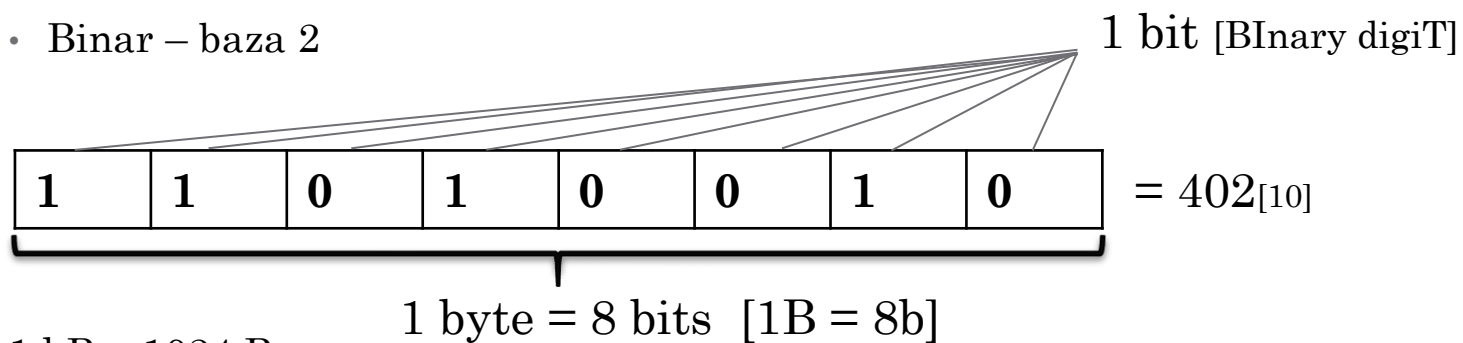
Tipuri de date

Numere intregi

Numere intregi – reprezentare

- Informatia reprezentata binar
 - 2 nivele de tensiune
 - 0 – 0V
 - 1 – 3-5V

- Binar – baza 2



1 kB = 1024 B

1MB = 1024 kB

1GB = 1024 MB

1TB = 1024 GB

Diferenta dintre un programator amator si unul profesionist

Numere intregi – Conversie

Conversia unui număr N în baza 2:

$$R_0 = N \% 2; C_0 = N/2;$$

$$R_1 = C_0 \% 2; C_1 = C_0/2;$$

$$R_2 = C_1 \% 2; C_2 = C_1/2;$$

.

.

.

$$R_n = C_{(n-1)} \% 2; C_n = C_{(n-1)}/2 == 0;$$

$$N_{[2]} = R_n R_{(n-1)} \dots R_2 R_1 R_0$$

$$N_{[10]} = R_0 * 2^0 + R_1 * 2^1 + R_2 * 2^2 + \dots + R_{n-1} * 2^{n-1} + R_n * 2^n$$

1	1	0	1	0
---	---	---	---	---

$$1 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0 =$$

$$1 * 16 + 1 * 8 + 0 * 4 + 1 * 2 + 0 * 1 =$$

$$26$$

Alte tipuri de date intregi

byte	System.Byte	1 Byte	Numere intregi in intervalul 0 - 255 ($0 \rightarrow 2^8 - 1$)
sbyte	System.SByte	1 Byte	Numere intregi cu semn, in intervalul -128 -> 127 ($-2^7 \rightarrow 2^7 - 1$)
short	System.Int16	2 Bytes (=16 biti)	Numere intregi cu semn, in intervalul -32,768 -> 32,767 ($-2^{15} \rightarrow 2^{15} - 1$)
ushort	System.UInt16	2 Bytes (=16 biti)	Numere intregi fara semn, in intervalul 0 -> 65535 ($0 \rightarrow 2^{16} - 1$)
int	System.Int32	4 Bytes (=32 biti)	Numere intregi cu semn, in intervalul -2,147,483,648 -> 2,147,483,647 ($-2^{31} \rightarrow 2^{31} - 1$)

Alte tipuri de date intregi

uint	System.UInt32	4 Bytes (=32 biti)	Numere intregi fara semn, in intervalul 0 -> 4,294,967,295 (0 -> $2^{32} - 1$)
long	System.Int64	8 Bytes (=64 biti)	Numere intregi cu semn, in intervalul -922337203685477508 -> 922337203685477507 (-2^{63} -> $2^{63} - 1$)
ulong	System.UInt64	8 Bytes (=64 biti)	Numere intregi fara semn, in intervalul 0 -> 18446744073709551615 (0 -> 2^{64})

Numere intregi – probleme

- Division by 0

```
var numarCaSir = Console.ReadLine();  
var convertit = int.Parse(numarCaSir);  
var impartire = 123 / convertit;
```

Exception Unhandled
System.DivideByZeroException: 'Attempted to divide by zero.'
[View Details](#) | [Copy Details](#) | [Start Live Share session...](#)
▸ Exception Settings

- Conversie

```
var numarCaSir = Console.ReadLine();  
var convertit = int.Parse(numarCaSir);
```

Exception Unhandled
System.FormatException: 'Input string was not in a correct format.'
[View Details](#) | [Copy Details](#) | [Start Live Share session...](#)
▸ Exception Settings

Complicat?

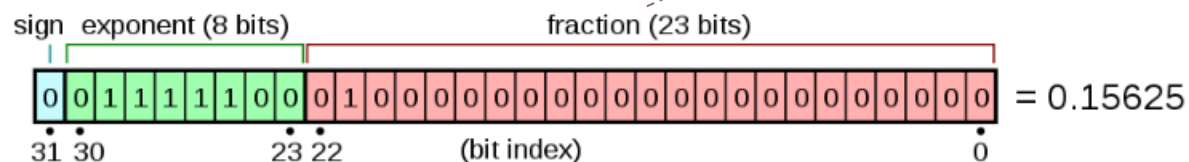
Tipuri de date

Numere reale

Numere in virgula flotanta

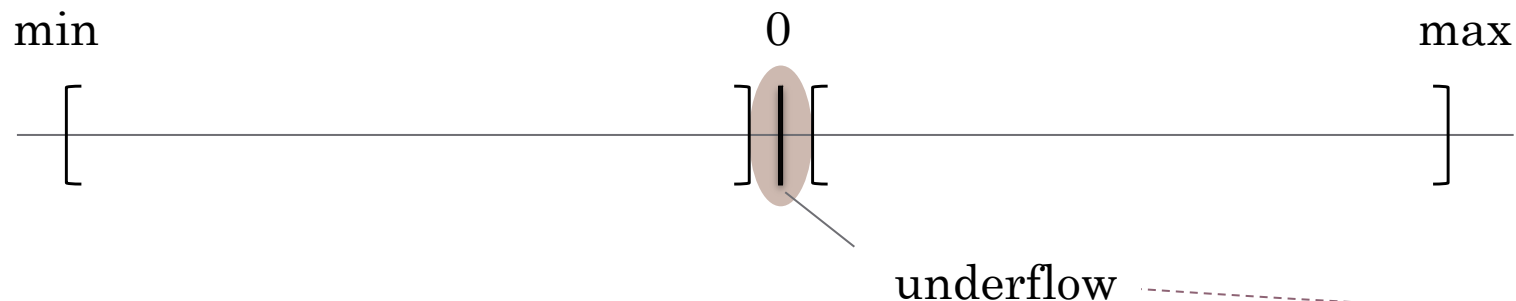
- Sunt numere reale, cu virgula – floating point

- [Documentatie](#)



- $N = sign * e^{exponent_{10}} * 1.fraction$

- Fraction – mantisa



Alte tipuri de numere cu virgula flotanta

float	System.Single	4 Bytes	~ 6 -> 9 cifre dupa virgula	Numere reale (simpla precizie) in intervalul $\pm 1.5 \times 10^{-45}$ -> $\pm 3.4 \times 10^{38}$
double	System.Double	8 Bytes	~ 15 -> 17 cifre dupa virgula	Numere reale (dubla precizie) in intervalul $\pm 5.0 \times 10^{-324}$ -> $\pm 1.7 \times 10^{308}$
decimal	System.Decimal	16 Bytes	28 -> 29 cifre dupa virgula	Numere reale in intrevalul $\pm 1.0 \times 10^{-28}$ to $\pm 7.9228 \times 10^{28}$

- Float – precizie simpla, double – dubla precizie (standard), decimal – precizie ridicata, banking.
- [type].MaxValue, [type].MinValue
- [type].Epsilon - cea mai mica valoare pozitiva reprezentabila
- [type].PositiveInfinity, [type].NegativeInfinity – rezultatul impartirii cu +0, -0
- [type].NaN – rezultatul impartirii la 0; nu arunca exceptie!

Egalitatea numerelor reale

- Egalitatea numerelor reale
 - Insumam 0.1 de 10 ori
 - $0.1 * 10 == \sum_1^{10} 0.1$

```
var suma = 0.1;
suma = suma + 0.1;
suma = suma + 0.1;
suma = suma + 0.1;
suma = suma + 0.1;
suma = suma + 0.1;
suma = suma + 0.1;
suma = suma + 0.1;
suma = suma + 0.1;
suma = suma + 0.1;

if (suma == 0.1 * 10) {
    Console.WriteLine("egale");
}
else {
    Console.WriteLine("inegale");
}
```

Egalitatea numerelor reale

- Egalitatea numerelor reale

- Insumam 0.1 de 10 ori

- $0.1 * 10 == \sum_1^{10} 0.1$

Solutii

- Marim precizia
 - Folosim o toleranta
 - $|a - b| \leq \text{tol}$

```
var suma = 0.1;  
suma = suma + 0.1;  
suma = suma + 0.1;  
suma = suma + 0.1;  
suma = suma + 0.1;  
suma = suma + 0.1;  
suma = suma + 0.1;  
suma = suma + 0.1;  
suma = suma + 0.1;  
suma = suma + 0.1;
```

```
if (suma == 0.1 * 10) {  
    Console.WriteLine("egale");  
}  
else {  
    Console.WriteLine("inegale");  
}
```

Tipuri de date numerice - operatii

- Initializari implicite

```
var intregImplicit = 1; // nerecomandat
var longExplicit = 1L; // nerecomandat
var doubleImplicit = 0.0; // nerecomandat
var floatExplicit = 0.0f; // nerecomandat
var decimalExplicit = 0.0m; // nerecomandat
```

- Operatorul Cast

- Orice operatie cu un numar cu virgula flotanta va avea rezultat un numar cu virgula flotanta!
- Cast
 - Transformare din virgula flotanta in intreg
 - Fara rotunjire

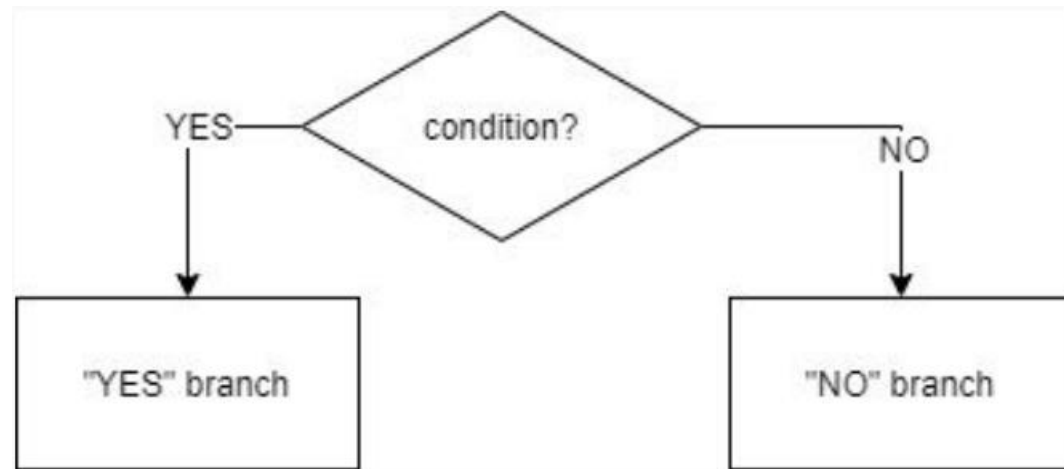
```
double numarReal = 3.99;
int numarIntreg = (int)numarReal;
//numarIntreg va avea valoarea 3
```

If-then-else

Instruțiune condițională

If-then-else

- Instructiunea "if" permite programatorului sa ofere 2 ramuri de continuare a executiei, in functie de rezultatul evaluarii unei conditii:
- Adevarat – ramura YES
- False – ramura NO



If-then-else – combinarea instructiunilor

- If-else

```
if (condition)
{
    // YES branch
}
else
{
    // NO branch
}
```

- Fara else

```
if (condition)
{
    // YES branch
}

// do something else
// yet another thing to do
```

- Inlantuirea instructiunilor if-else

```
if (condition1)
{
    // do something
}
else if (condition2)
{
    // do something else
}
else if (condition3)
{
    // do something else one more time
}
else {
    // do something else AGAIN
}
```

Tipuri de date

Boolean

Boolean

- Bool
- Permite doua valori
 - true
 - false
- 1byte de memorie
- Operatii logice
 - `5==5` -> true
 - `5==2` -> false
- `bool.Parse`
- Comparatiile `=>` bool

Operatori

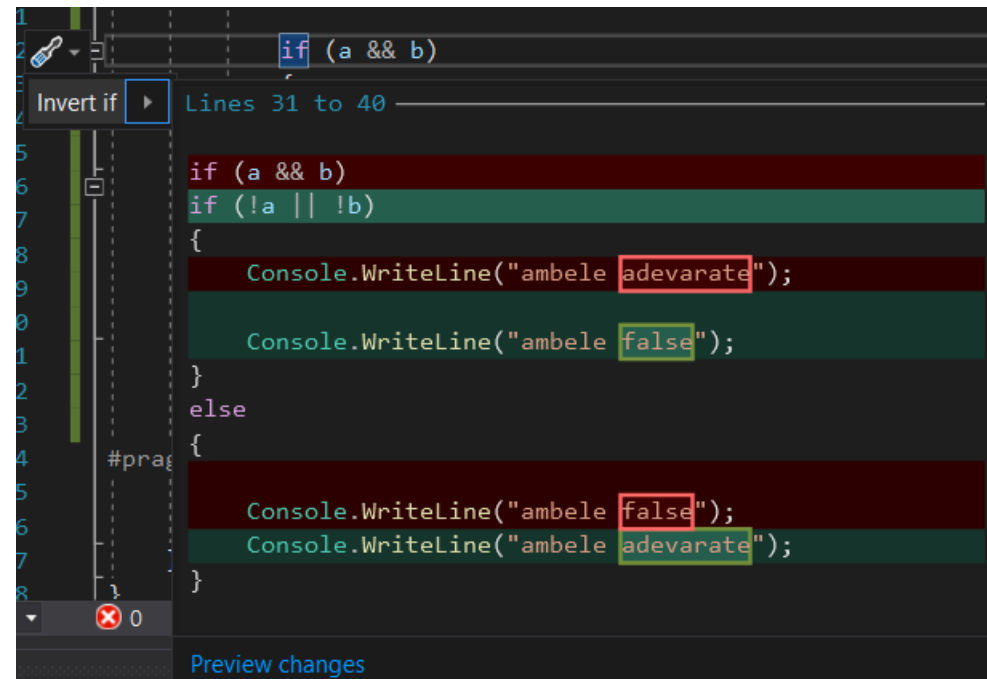
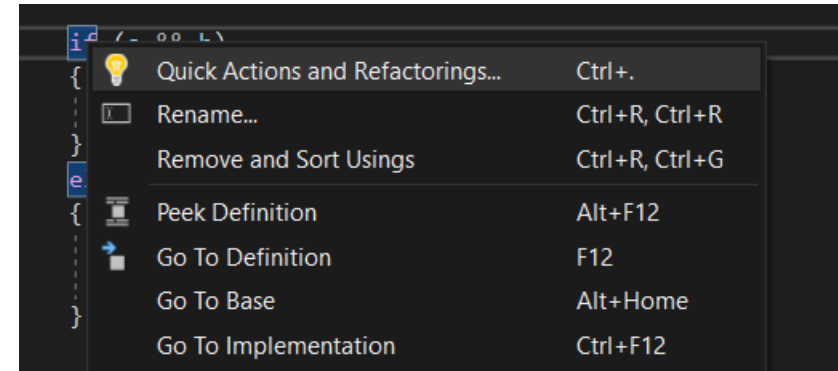
- Negatia logica (!)
 - !true -> false
 - !false -> true
- Logical / Bitwise AND (&&)
 - false && false == false && true == false
- Logical / Bitwise XOR (^) = mutual exclusiv
 - True^false == false^true == true
 - True^true == false^false == false
- Conditional OR (||)
 - true || false == true || true == true
 - false || false == false
- Logical / Bitwise OR (|)
- Logical AND (&)
- Precedenta operatorilor
- Uzual
 - ==, &&, ||

Inversarea logica

Inversarea logica

$!(x \mid \mid y) == (!x \&\& !y)$

$!(x \&\& y) == (!x \mid \mid !y)$



Exercitii

- Scrieti un program care va calcula valoarea urmatoarei functii pentru x citit de la tastatura

$$\begin{cases} 7x^2, \text{ daca } x \in (-\infty, -2] \\ 4x - 5, \text{ daca } x \in \left(-2, \frac{1}{2}\right] \\ 14x - 7, \text{ daca } x \in \left(\frac{1}{2}, \infty\right) \end{cases}$$

- Scrieti un program care va determina daca un numar negativ citit de la tastatura este divizibil cu 2 si nu este divizibil cu 6

Instructione switch

Instructional switch

- Google it 😊

C# programmer's best
friends

C# programmer's best friends



- <https://docs.microsoft.com/en-us/dotnet/csharp/>
- <https://www.c-sharpcorner.com/>
- <https://stackoverflow.com/>



stackoverflow

Va multumesc!