

Програмування мовою C# .Net Framework

Клакович Л.М.

Що ми будемо вчити?

- Розробка консольних та десктопних (WPF) програм мовою C#
- Типи даних C#, оператори, інструкції
- Розробка типів-значень через enum та struct
- Розробка типів-посилань через class. Наслідування
- Інтерфейси, реалізація інтерфейсів класами та структурами
- Колекції та узагальнені колекції C#
- Введення-виведення, робота з файлами
- Сериалізація об'єктів у різних форматах
- Делегати і події
- LINQ
- Перехоплення винятків
- Юніт тестування засобами .Net
- Додаткові теми (async-await, TPL, PLINQ, ...)

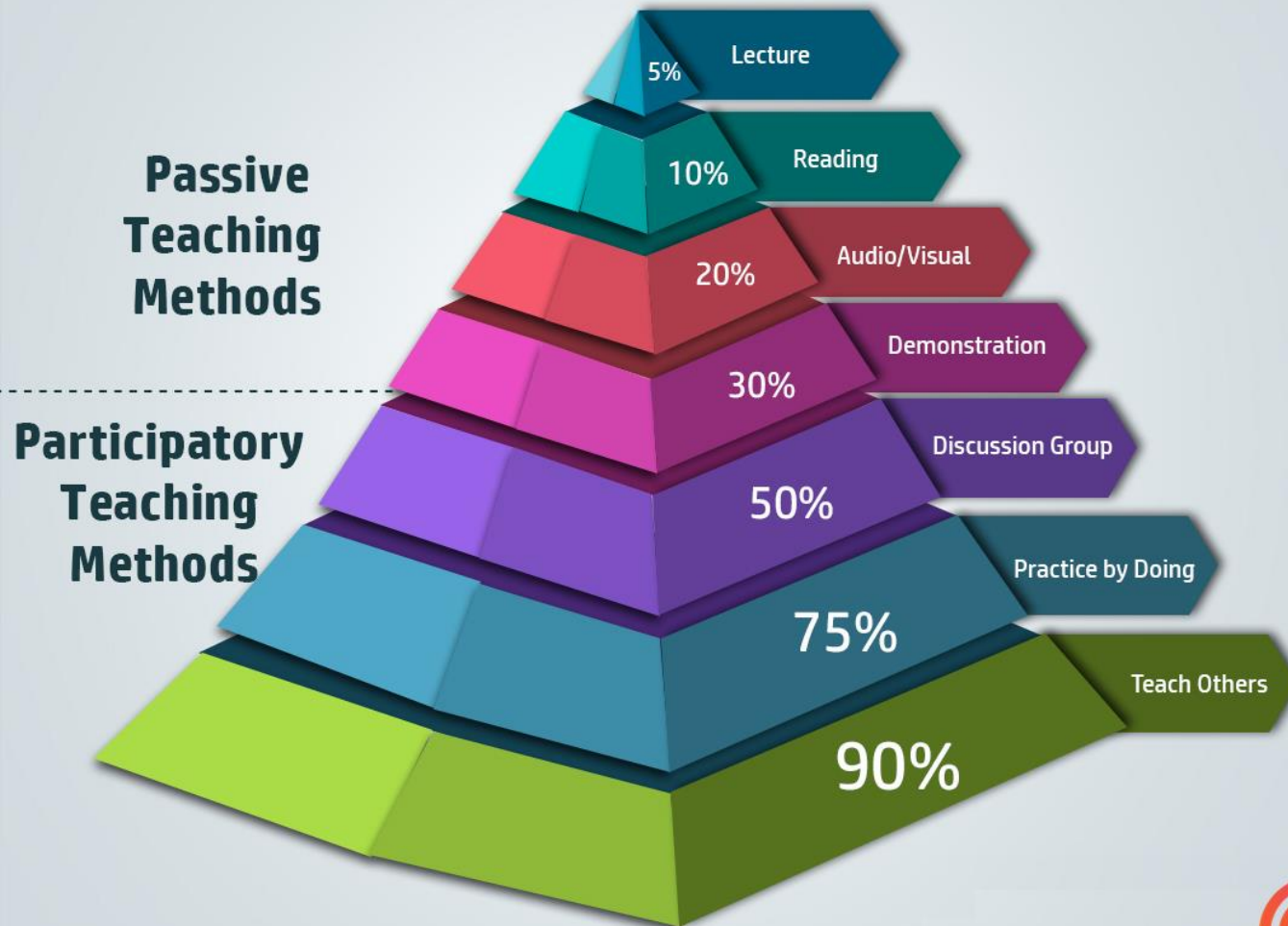
Очікувані цілі

➤ Навчитися робити:

- ✓ Розробляти консольні та десктопні програми мовою C#
- ✓ Створювати нові типи через `class`, `struct`, `enum`.
- ✓ Використовувати наслідування класів та реалізацію інтерфейсів, агрегацію та узагальнення
- ✓ Ефективно використовувати колекції та узагальнені колекції
- ✓ Передбачати та обробляти виняткові ситуації
- ✓ Користуватися LINQ
- ✓ Писати юніт-тести
- ✓ Працювати з файловою системою, файлами
- ✓ Сериалізувати об'єкти в різних форматах

THE LEARNING PYRAMID

KNOWLEDGE RETENTION RATES



Adapted from National Training Laboratories, Maine



Огляд архітектури .NET

1. Платформа .NET
2. .NET Framework - середовище для розробки і виконання прикладних програм
3. Виконання програм в середовищі CLR
4. Компілювання коду програми. Збірки
5. Загальна система типів
6. C# - мова для роботи з .NET Framework
7. Перша програма. Введення-виведення

Джерела

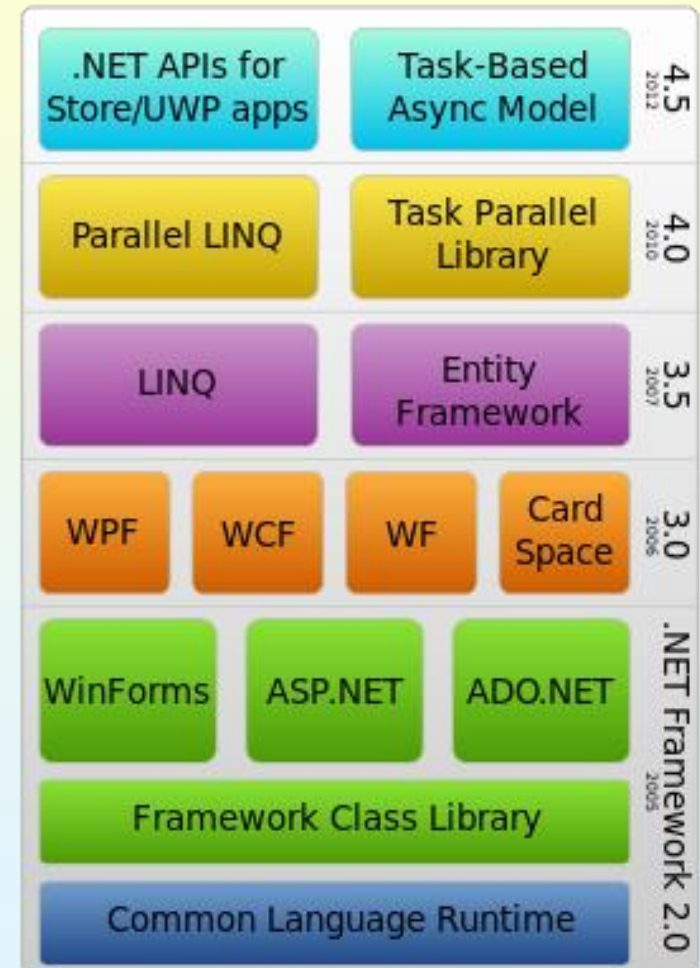
- Е. Троелсен C# і платформа .NET
- Дж. Ріхтер, CLR via C#. Програмування на платформі .NET Framework на C#.
- metanit.com/sharp/
- [C# programming guide](http://docs.microsoft.com/csharp/)- docs.microsoft.com

Платформа .NET

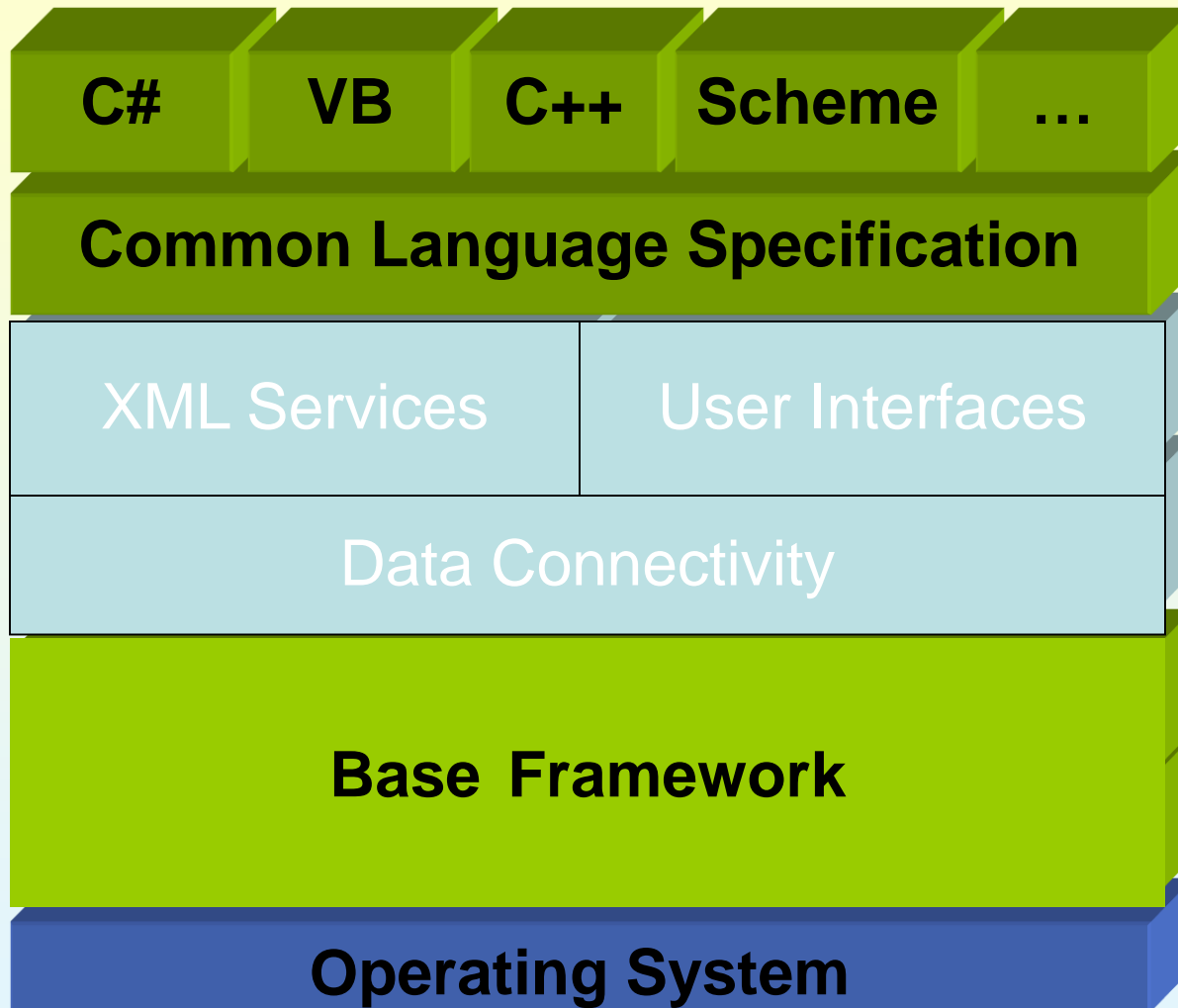
- **Microsoft .NET** - програмна технологія, запропонована фірмою [Microsoft](#) як платформа для створення десктопних, Web-програм, SOAP та REST сервісів, мобільних програм, кросплатформених програм, ігор (Unity)
- **Головна ідея .NET** - сумісність різних служб, написаних на різних мовах програмування
 - **Підтримка різних мов програмування** (C#, VB.NET, F#, C++) -При компіляції код різних мов компілюється в збірку на спільній мові CIL (Common Intermediate Language), тому можемо робити окремі модулі однієї програми на різних мовах.
 - **Кросплатформеність.** Використовуючи різні технології на платформі .Net, можна розробляти програми для різних платформ Windows, MacOS, Linux, Android, iOS, Tizen
 - **Потужна бібліотека класів.** Одна для всіх мов
 - **Різноманітність технологій.** Спільномовне середовище виконання CLR і базова бібліотека класів є основою для цілого стеку технологій:
 - для роботи з базами даних ADO.NET і Entity Framework Core.
 - для побудови графічних програм - WPF і UWP, Windows Forms.
 - для розробки мобільних додатків - Xamarin.
 - для створення веб-сайтів - ASP.NET і т.д.

.NET Framework і альтернативи

- .Net початково розвивалась як платформа для Windows під назвою **.NET Framework**.
- В 2019 вийшла остання версія **.NET Framework 4.8**. Більше не розвивається
- В 2014 вийшла **.NET Core** - кросплатформенна open-source версія, яка замінила .Net Framework
- Версія .Net **Mono** створена в 2004 теж open-source-версія платформи .NET Framework для Linux і MacOS

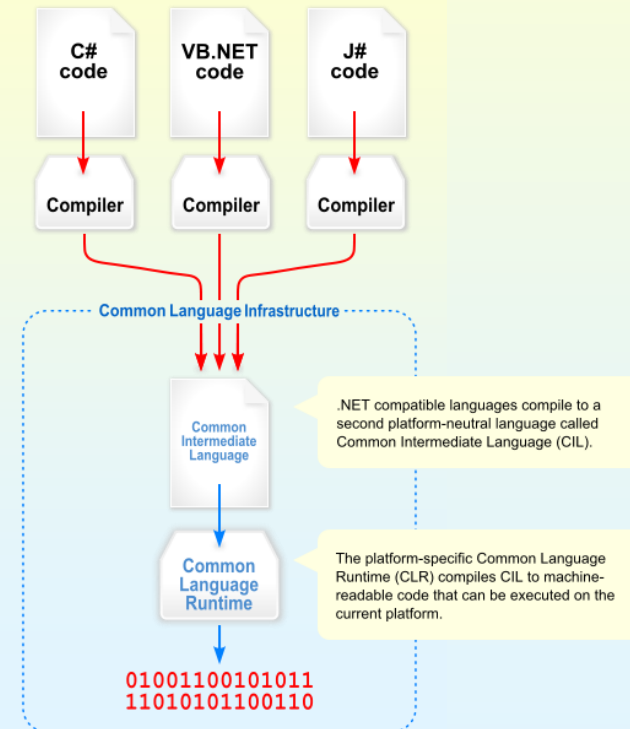


Архітектура .NET Framework



1. Середовище виконання програм - Common Language Runtime (CLR)- основа .NET Framework:

- Завантаження та строга перевірка типів
- Компілювання коду в **Intermediate Language (MSIL)**
- Компілювання **just-in-time (JIT)** в систему команд процесора
- Підтримка збірок (**assemblies**)
- Керування кодом під час виконання
- Організація віддаленої взаємодії
- Керування потоками
- Виділення пам'яті
- Збір сміття



2. Бібліотека класів - .NET Framework Class Library

- **System** - низькорівневі типи
- **System.Collections** - контейнери ArrayList, SortedList, Queue, Stack
- **System.ComponentModel** - компоненти і їх контейнери
- **System.Data** - доступ до баз даних
- **System.Drawing** - GDI+
- **System.EnterpriseServices** - середовище для програм рівня підприємства
- **System.IO** - файловий ввід-вивід
- **System.Math** – математика
- **System.Net** –протоколи і сервіси мережі
- **System.Reflection** - RTTI
- **System.Security** - криптографія, захист
- **System.Threading** – Багатопотоковість
- **System.Web** - взаємодія браузер-сервер
- **System.Windows.Forms** - стандартні програми, форми, контролю
- **System.Xml** – підтримка XML

Виконання програм в середовищі CLR

- **CLR-сумісні компілятори** текст програми конвертують в IL-код.
- **Intermediate Language (IL)**
 - високорівневий асемблер
 - невиконуваний код
 - апаратно незалежний
 - можливість зворотнього асемблерування
- **JIT** компілює IL-код в команди процесора - безпосередньо в часі виконанням
 - відсутність повторного компілювання
 - невикористаний IL-код не компілюється
 - JIT-компілятор є машинно-залежний.

C# source

```
Calc c = new Calc();  
int sum = c.Add(2, 4);
```



C# compiler



IL

```
.locals init ([0] class Calc c, [1] int32 sum)  
newobj instance void Calc::.ctor()  
stloc.0 // c = ptr to new object  
ldloc.0  
ldc.i4.2 // pass second arg  
ldc.i4.4 // pass first arg  
callvirt instance int32 Calc::Add(int32,int32)  
stloc.1 // sum = retval
```

Збірки (assembly)

- **Збірка** – це двійковий файл (exe чи dll), що містить:
 - IL код,
 - **метадані** (детально описують особливості кожного типу, що є всередині)
 - **маніфест** (поточна версія збірки; культура (локалізація); перелік посилань на всі зовнішні збірки)
- Файл з кодом, написаний на мові, яку підтримує .NET (C#, VB.NET...), з допомогою відповідного компілятора компілюється в збірку
- Збірка може виконуватися на Windows (exe) чи використовуватися іншими програмами (dll) при умові встановлення там однієї з версій .Net Framework

C# - мова, створена Microsoft для роботи з .NET Framework

- C# є простою, сучасною, об'єктно-орієнтованою мовою програмування, яка походить від C++ та Java.
- C# забезпечує підтримку основних принципів розробки програм:
 - ✓ Строга перевірка типів,
 - ✓ контроль виходу за межі масиву,
 - ✓ Упередження спроб використання неініціалізованих змінних
 - ✓ Автоматичних збір сміття
- Для розробки програм використовується **Visual Studio** – Інтегроване середовище розробки (IDE), яке є сукупність інструментів розробки, доступних через загальний користувальницький інтерфейс

C# 8.0 вересень 2019
з релізом .NET Core 3



Популярні мов програмування – в роках





























Very Long Term History

To see the bigger picture, please find below the positions of the top 10 programming languages of many years back. Please note that these are *average* positions for a period of 12 months.

Programming Language	2017	2012	2007	2002	1997	1992	1987
Java	1	2	1	1	15	-	-
C	2	1	2	2	1	1	1
C++	3	3	3	3	2	2	4
C#	4	5	7	11	-	-	-
Python	5	7	6	12	27	16	-
Visual Basic .NET	6	14	-	-	-	-	-
JavaScript	7	9	8	7	20	-	-
PHP	8	6	4	5	-	-	-
Perl	9	8	5	4	3	8	-
Delphi/Object Pascal	10	11	11	8	-	-	-
Lisp	31	12	15	13	8	4	2
Prolog	32	30	26	15	17	13	3

В динаміці <https://www.youtube.com/watch?v=Og847HVwRSI>

Популярність мов програмування, 2019

Rank	Language	Type	Score
1	Python	  	100.0
2	Java	  	96.3
3	C	  	94.4
4	C++	  	87.5
5	R		81.5
6	JavaScript		79.4
7	C#	   	74.5
8	Matlab		70.6
9	Swift	 	69.1
10	Go	 	68.0
11	Arduino		67.2
12	HTML, CSS		66.8
13	PHP		65.1
14	Assembly		63.7
15	SQL		63.4

- Журнал IEEE Spectrum

https://spectrum.ieee.org/ns/IEEE_TPL_2019/index/2019/1/1/1/1/1/50/1/50/1/50/1/30/1/30/1/20/1/20/1/5/1/50/1/100/1/50/

Приклад C# програми


```
public class App
{
    static public void Main(System.String[] args)
    {
        System.Console.WriteLine("Hello World!");
    }
}
```

- **App** - НОВИЙ ТИП
- **System.Console**, **System.String** - типи .Net Framework, IL код яких знаходиться в MSCorLib.dll

Class

- `class` використовується для визначення нового типу
- Більшість коду C# розміщується всередині класу
 - не допускається визначення глобальних змінних
 - не допускається визначення глобальних методів

class definition



```
class App
{
    ...
}
```

Main()

- Метод `Main` є точкою входження в програму
 - Повинен бути `static` методом деякого класу
- `Main` може взаємодіяти з оточенням
 - Може отримати аргументи командного рядка як масив рядків
 - Може повертати `int` для зазначення успішного/помилкового виконання

entry point



```
public class App
{
    static void Main()
    {
        ...
    }
}
```

Введення з консолі

- Введення з консолі здійснює статичний метод **ReadLine**
 - з класу **Console** простору імен **System**
 - повертає введений рядок символів (string)
- Часто такий рядок вимагає конвертування до потрібного типу
 - методи конвертування забезпечує клас **Convert**
 - Або методи **Parse()** та **TryParse()**, які є в кожному вбудованому типі

read entire line	→	<code>string s = System.Console.ReadLine ();</code>
convert string to int	→	<code>int i = System.Convert.ToInt32 (s);</code>
convert string to double	→	<code>double d = System.Convert.ToDouble (s);</code>
		<code>int number = Int32.Parse(s);</code>

Введення з консолі

- Використовуй **TryParse()** для уникнення винятків форматування

```
static bool TryParse(string s, out Int32 result);
```

```
string s = Console.ReadLine();  
int number;  
bool rez = Int32.TryParse(s, out number);  
Console.WriteLine("{0}-{1}", rez, number);
```

Виведення на консоль

- Виведення здійснюють статичні методи **Write** та **WriteLine**
 - з класу **Console** простору імен **System**
 - WriteLine** поміщає рядок в потік
 - перевизначені для всіх типів бібліотеки

```
int    i = 3;  
double d = 5.2;
```

int → System.Console.WriteLine(i);

double → System.Console.WriteLine(d);

multiple → System.Console.WriteLine("first {0} second {1}", i, d);

↑
format string

↑
placeholder

↑
value

Форматне виведення

{ index [:formatString] }

- **Index**: номер позиції в форматному рядку
- **formatString**: визначає формат.

FormatString	Опис
C або c	Форматування валюти. Додає валюту по замовченню вашої ОС.
D або d	Форматування десяткових чисел. Також може використовуватися вказання мінімальної кількості цифр для доповнення.
E або e	Використовується для експоненціального запису.
N або n	Використовується для форматування числових значень.
X або x	Форматування у шістнадцятковому вигляді.

Форматне виведення

```
Console.WriteLine("Currency format: {0:C}", 5555.5812);  
Console.WriteLine("Datetime format: {0:d}, {0:t}", DateTime.Now);  
Console.WriteLine("Float format (3 digits after point): {0:F3}", 1234.56789);  
Console.WriteLine("Numerical format: {0:N1}", 5555.5812);  
Console.WriteLine("16-X format: {0:X}", 5555);
```

```
Currency format: $5,555.58  
Datetime format: 22-Jan-17, 9:50 PM  
Float format (3 digits after point): 1234.568  
Numerical format: 5,555.6  
16-X format: 15B3
```

Завдання

- 1) В методі Main() визначити дві змінні цілого типу **a** і **b**. Зчитати їх значення з консолі, обчислити: $a+b$, $a-b$, $a*b$, a/b . Вивести на консоль результати.
- 2) Зчитати з консолі значення для 3 змінних: char, bool і double типів. Вивести повідомлення: "You entered (змінна char), (змінна bool), (змінна double)"

Ваш фідбек для мене важливий 😊

1. Що ви зрозуміли з лекції?
2. Що залишилося не зрозумілим?