

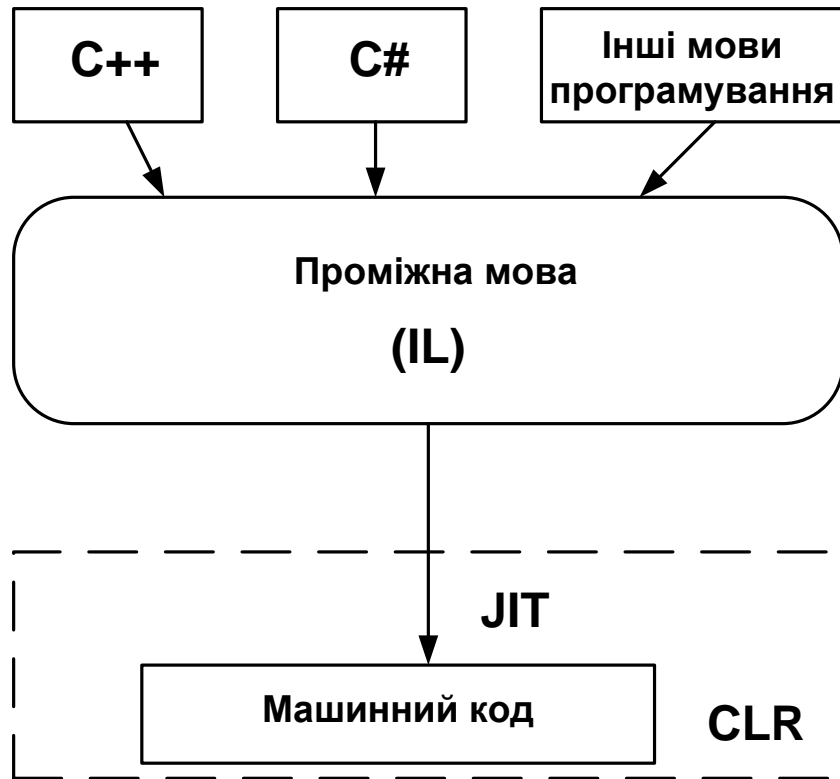
ТЕХНОЛОГІЯ .NET FRAMEWORK

Технологія .Net Framework представляє собою середовище, яке дає можливість розробляти і виконувати програми незалежно від операційної системи. Основною мовою програмування цієї технології є мова C#. У **цій технології єдиний спосіб опису програмного коду (метадані), одне середовище виконання (Common Language Runtime) і одна базова бібліотека (BCL).**

Міжмовна інтеграція досягається за рахунок того, що програма, написана на мові високого рівня (C#, C++ чи ін.) **не одразу компілюється у машинний код, а спочатку відбувається компіляція у команди (псевдокод) єдиної проміжкової мови Intermediate Language (IL)**

Під час виконання програми команди проміжкової мови перетворюються у машинні команди у середовищі Common Language Runtime (CLR) за допомогою відповідних компіляторів (JIT-компіляторів)

ТЕХНОЛОГІЯ .NET FRAMEWORK



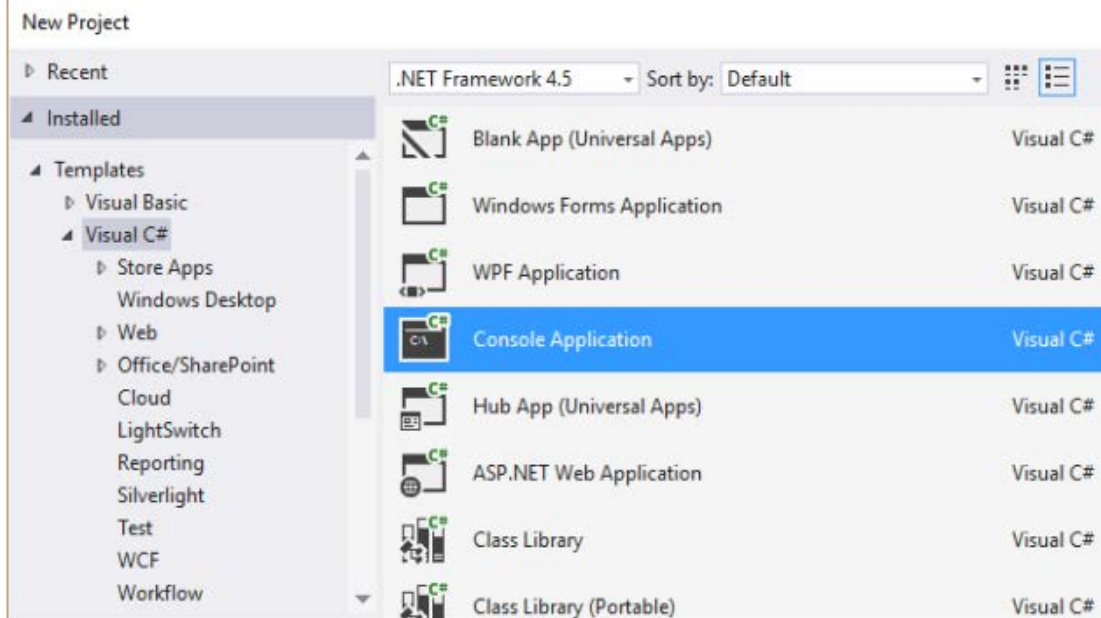
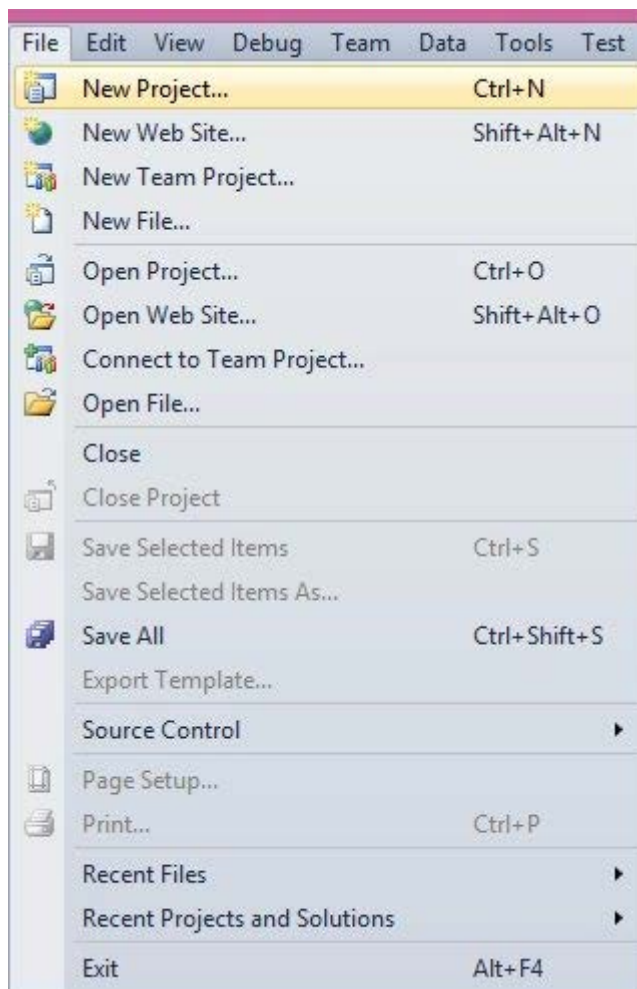
Технологія .Net Framework реалізована в середовищі Common Language Runtime CLR можна інтерпретувати, як препроцесор, який перетворює команди мови Intermediate Language (IL) в команди процесора

Використання CLR для розробки програм забезпечує:

- міжмовну інтеграцію;
- автоматичне керування пам'яттю (збирання сміття);
- контроль типів;
- контроль версій програм.

ІНТЕГРОВАНЕ СЕРЕДОВИЩЕ РОЗРОБКИ

Microsoft Visual Studio 2019 Community Edition



Декларація простору імен

Namespaces in C#

```
using System;  
using System.Threading;  
using SchoolA;  
using SchoolB;
```

Простір імен проектів

Project Namespace

```
namespace TEPPProject  
{  
}
```

Програмний клас

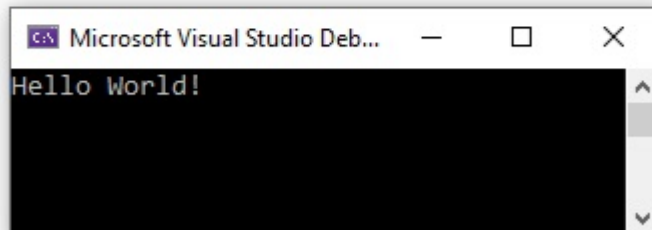
```
class Program  
{  
    static void Main(string[] args)  
    {  
        Console.WriteLine("Hello World!");  
    }  
}
```

C# Class & Method

```
using System;  
  
namespace TEPPProject  
{  
    References  
    class Program  
    {  
        References  
        static void Main(string[] args)  
        {  
            Console.WriteLine("Hello World!");  
        }  
    }  
}
```

Introduction to C#

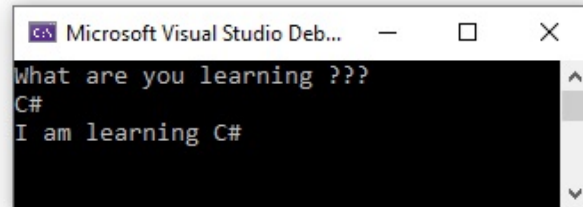
```
using System;  
  
namespace TEPPProject  
{  
    public delegate Boolean StudentsDelegate(int score);  
    References  
    class Program  
    {  
        References  
        static void Main(string[] args)  
        {  
            Console.WriteLine("Hello World!\n\n\n");  
        }  
    }  
}
```



First Code in C#

```
using System;
```

```
namespace TEPPProject
{
    public delegate Boolean StudentsDelegate(int score);
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("What are you learning ???");
            string subject = Console.ReadLine();
            Console.WriteLine("I am learning {0} \n", subject);
        }
    }
}
```

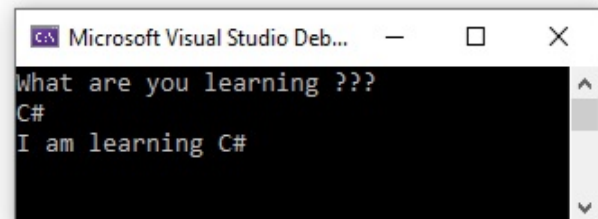


A screenshot of a console window titled "Microsoft Visual Studio Deb...". It shows the program's output: "What are you learning ???", "C#", and "I am learning C#".

First Code in C#

```
using System;
```

```
namespace TEPPProject
{
    public delegate Boolean StudentsDelegate(int score);
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("What are you learning ???");
            string subject = Console.ReadLine();
            Console.WriteLine("I am learning " + subject + "\n");
        }
    }
}
```



A screenshot of a console window titled "Microsoft Visual Studio Deb...". It shows the program's output: "What are you learning ???", "C#", and "I am learning C#".

```
static void Main(string[] args)
{
    Console.WriteLine("Hello World!");
}
```

```
static void Main(string[] args)
{
    int x = 89;
    Console.WriteLine(x);
}
// Outputs 89
```

```
static void Main(string[] args)
{
    int x = 10;
    double y = 20;

    Console.WriteLine("x = {0}; y = {1}", x, y);
}
// Output: x = 10; y = 20
```

```
static void Main(string[] args)
{
    string yourName;
    Console.WriteLine("What is your name?");

    yourName = Console.ReadLine();

    Console.WriteLine("Hello {0}", yourName);
}
```

```
using System;

class MainClass {
    public static void Main() {
        double v = 17688.65849;
        double v2 = 0.15;
        int x = 21;

        Console.WriteLine("{0:F2}", v);

        Console.WriteLine("{0:N5}", v);

        Console.WriteLine("{0:e}", v);

        Console.WriteLine("{0:r}", v);

        Console.WriteLine("{0:p}", v2);

        Console.WriteLine("{0:X}", x);

        Console.WriteLine("{0:D12}", x);

        Console.WriteLine("{0:C}", 189.99);
    }
}
```

```
17688.66
17,688.65849
1.768866e+004
17688.65849
15.00 %
15
000000000021
$189.99
```

```
static void Main(string[] args)
{
    int age = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("You are {0} years old", age);
}
```

С или с	Валюта
Д или d	Decimal
Е или e	Экспоненциальный
Ф или f	С фиксированной запятой
Г или g	Общее
Н или n	Numeric
Р или р	Процент
Р или г	Приемо-передача
Х или х	Шестнадцатеричный

ВБУДОВАНІ ТИПИ ДАНИХ

Тип	Опис	Область значень
object	Базовий клас для всіх інших типів	
string	Рядковий тип, послідовність символів Unicode	
sbyte	8-розрядне ціле число з знаком	-128 до 127
short	16-розрядне ціле число з знаком	-32768 до 32767
int	32-розрядне ціле число з знаком	-2147483648 до 2147483647
long	64-розрядне ціле число з знаком	-9223372036854775808 до 9223372036854775807
byte	8-розрядне ціле число без знака	0 до 255
ushort	16-розрядне ціле число без знака	0 до 65535
uint	32-розрядне ціле число без знака	0 до 4294967295
ulong	64-розрядне ціле число без знака	0 до 18446744073709551615
float	Число з плаваючою крапкою 4 байти, точність — 7 розрядів	$\pm 1,5 \cdot 10^{-45}$ до $\pm 3,4 \cdot 10^{33}$
double	Число з плаваючою крапкою 8 байт, точність — 16 розрядів	$\pm 5 \cdot 10^{-324}$ до $\pm 1,7 \cdot 10^{306}$
bool	Логічний тип	true або false
char	Тип символу Unicode	U+0000 до U+ffff
decimal	Тип десяткового числа 12 байт, точність — 28 розрядів	(Від $-7,9 \times 10^{28}$ до $7,9 \times 10^{28}$) / (10^{0-28})

Загальне правило опису:

- без початкової ініціалізації
<тип змінної> <ідентифікатор змінної>;
- з початковою ініціалізацією
<тип змінної> <ідентифікатор змінної> = <значення>;

Загальне правило опису констант:

const <тип константи> <ідентифікатор константи> = <значення>;

const int a = 10;

```
int x = 42;
double pi = 3.14;
char y = 'Z';
bool isOnline = true;
string firstName = "David";
```


ПЕРЕТВОРЕННЯ ТИПІВ

Неявне перетворення

`float f = 1.23;`

`double d = f;`

Явне перетворення:

<змінна> = (<тип>) <вираз>;

`double d = 2.9;`

`float f = (float)d;`

VAR

*С # надає зручну функцію, яка дозволяє компілятору автоматично визначати тип змінної на основі виразу, якому вона призначена. **var** використовується ключове слово для цих сценаріїв:*

```
var num = 15;
```

*Змінні, оголошені за допомогою ключового слова **var**, називаються неявно введеними змінними. Неявно введені змінні повинні бути ініціалізовані зі значенням. Наприклад, наступна програма призведе до помилки:*

```
var num;  
число = 42;
```


ПЕРЕЛІКОВНИЙ ТИП

Переліковний тип це тип, який задається повним переліком своїх значень. Значеннями перелікового типу можуть бути константи типів **byte, sbyte, short, ushort, int, uint, long, ulong**. Якщо при описанні перелікового типу в явному вигляді не вказано тип його констант, то вважається, що константи типу **int**. Всі константи повинні бути одного типу, який називається базовим типом. Тип **char** не може використовуватися в якості базового типу. В переліковному типі можна задавати довільну кількість іменованих констант.

Загальне правило опису:

```
<атрибути> <модифікатори> enum <ім'я типу> [: <базовий тип>]  
{  
    <атрибути> <конст.1> [= <знач.1>],  
    <атрибути> <конст.2> [= <знач.2>],  
    .....  
    <атрибути> <конст.N> [= <знач.N>]  
}
```

ПЕРЕЛІКОВНИЙ ТИП

```
enum Week : Byte
```

```
{  
    Monday,    // =0  
    Tuesday,   // =1  
    Wednesday, // =2  
    Thursday,  // =3  
    Friday,    // =4  
    Saturday,  // =5  
    Sunday     // =6  
}
```

```
enum Week : Byte
```

```
{  
    Monday =2,  
    Tuesday =35,  
    Wednesday =2,  
    Thursday, // =3  
    Friday, // =4  
    Saturday, // =5  
    Sunday =1,  
    Sabbath =Saturday // =5  
}
```

Ініціалізація змінної перелікового типу значенням здійснюється через ім'я типу та оператор "." (крапка). Також між змінними перелікового типу та змінними відповідного базового типу існує явне перетворення типів.

```
Week d = Week.Monday;  
byte b = (byte)d; // b = 2  
byte i = 1;  
Week s = (Week)i; // s = Week.Sunday
```

ВИРАЗИ ТА ОПЕРАЦІЇ

Арифметичні вирази

Операція	Позначення	Приклад
+	додавання	$z = x + y$
-	віднімання	$z = x - y$
*	множення	$z = x * y$
/	ділення	$z = x / y$
%	остача від ділення	$z = x \% y$

Операція	Аналог
Постфіксна форма	
<code>i++</code>	<code>i=i+1</code>
<code>i--</code>	<code>i=i-1</code>
Префіксна форма	
<code>++i</code>	<code>i=i+1</code>
<code>--i</code>	<code>i=i-1</code>

Операція	Аналог з бінарними операціями
<code>int i = 5;</code>	<code>int i = 5;</code>
<code>int j = ++i; // j=6 i=6</code>	<code>i=i+1;</code> <code>int j = i; // j=6 i=6</code>
<code>int i = 5;</code>	<code>int i = 5;</code>
<code>int j = i++; // j=5 i=6</code>	<code>int j = i;</code> <code>i=i+1; // j=5 i=6</code>
<code>int i = 5;</code>	<code>int i = 5;</code>
<code>int j = --i; // j=4 i=4</code>	<code>i=i-1;</code> <code>int j = i; // j=4 i=4</code>
<code>int i = 5;</code>	<code>int i = 5;</code>
<code>int j = i--; // j=5 i=4</code>	<code>int j = i;</code> <code>i=i-1; // j=5 i=4</code>

ВИРАЗИ ТА ОПЕРАЦІЇ

Побітові операції

Операція	Позначення	Приклад
&	побітове «і»	<code>int x=10; //x=1010₍₂₎</code> <code>int y=7; //y=0111₍₂₎</code> <code>z=x&y // z=2=0010₍₂₎</code>
	побітове «або»	<code>int x=10; //x=1010₍₂₎</code> <code>int y=7; //y=0111₍₂₎</code> <code>z=x y //z=15=1111₍₂₎</code>
^	Побітове «виключаюче або»	<code>int x=10; //x=1010₍₂₎</code> <code>int y=7; //y=0111₍₂₎</code> <code>z=x^y //z=13=1101₍₂₎</code>

Операції зсуву

Операція	Позначення	Приклад
>>	зсув розрядів вправо (змінна) =(змінна)>>(кільк. розрядів)	<code>int i=4; //i=100₍₂₎</code> <code>i=i>>1; //i=2=10₍₂₎</code>
<<	зсув розрядів вліво (змінна) =(змінна)<<(кільк. розрядів)	<code>int i=4; //i=100₍₂₎</code> <code>i=i<<2; //i=16=10000₍₂₎</code>

ВИРАЗИ ТА ОПЕРАЦІЇ

Операції присвоєння

Операція	Приклад	Аналог з бінарними операціями
<code>+=</code>	<code>int i = 5; i += 3; // i=8</code>	<code>int i = 5; i=i+3; // i=8</code>
<code>-=</code>	<code>int i = 5; i -= 3; // i=2</code>	<code>int i = 5; i=i-3; // i=2</code>
<code>*=</code>	<code>int i = 5; i *= 3; // i=15</code>	<code>int i = 5; i=i*3; // i=15</code>
<code>/=</code>	<code>int i = 6; i /= 3; // i=2</code>	<code>int i = 6; i=i/3; // i=2</code>
<code>>>=</code>	<code>int i = 5; i >>= 1; // i=2</code>	<code>int i = 5; i=i>>1; // i=2</code>
<code><<=</code>	<code>int i = 5; i <<= 1; // i=10</code>	<code>int i = 5; i=i<<1; // i=10</code>
<code>&=</code>	<code>int i = 5; int j = 7; i &= j; // i=5</code>	<code>int i = 5; int j = 7; i=i&j; // i=5</code>
<code>^=</code>	<code>int i = 5; int j = 7; i ^= 1; // i=2</code>	<code>int i = 5; int j = 7; i=i^j; // i=2</code>
<code> =</code>	<code>int i = 5; int j = 7; i = 1; // i=7</code>	<code>int i = 5; int j = 7; i=i j; // i=7</code>

ВИРАЗИ ТА ОПЕРАЦІЇ

Назва	Опис	Результат	Пояснення
Тригонометричні функції			
Sin	Синус	double	Math.Sin(double x)
Cos	Косинус	double	Math.Cos(double x)
Tan	Тангенс	double	Math.Tan(double x)
Обернені тригонометричні функції			
ASin	Арксинус	double	Math.ASin(double x)
ACos	Арккосинус	double	Math.ACos(double x)
ATan	Арктангенс	double	Math.ATan(double x)
ATan2	Арктангенс	double	Math.ATan2(double x, double y) – кут, тангенс якого є результатом ділення y на x
Гіперболічні функції			
Tanh	Тангенс гіперболічний	double	Math.Tanh(double x)
Sinh	Синус гіперболічний	double	Math.Sinh(double x)
Cosh	Косинус гіперболічний	double	Math.Cosh(double x)
Експонента і логарифмічні функції			
Exp	Експонента	double	Math.Exp(x)
Log	Логарифм натуральний	double	Math.Log(x)
Log10	Логарифм десятковий	double	Math.Log10(x)
Модуль(абсолютна величина), корінь квадратний, знак			
Abs	Модуль	Залежить від типу аргументу	Math.Abs(x)
Sqrt	Квадратний корінь	double	Math.Sqrt(x)
Sign	Знак числа	int	Math.Sign(x)
Заокруглення			
Ceiling	Заокруглення до більшого цілого	double	Math.Ceiling(double x)
Floor	Заокруглення до меншого цілого	double	Math.Floor(double x)
Round	Заокруглення	Залежить від типу аргументу	Math.Round(x)
Мінімум, максимум			
Min	Мінімум двох чисел	Залежить від типу аргументу	Math.Min(x,y)
Max	Максимум двох чисел	Залежить від типу аргументу	Math.Max(x,y)

Математичні
функції

ВИРАЗИ ТА ОПЕРАЦІЇ

Степінь, остача			
Pow	Піднесення до степеня	double	Math.Pow(x,y)
IEEERemainder	Остача від ділення	double	Math.IEEERemainder(double x, double y)
Добуток двох цілих величин			
BigMul	Добуток	long	Math.BigMul(int x,int y)
Ділення і остача від ділення			
DivRem	Ділення і остача	Залежить від типу аргументу	Math.DivRem(x,y, rem)
Константи			
E	База натурального логарифма	double	Math.E
PI	Значення числа π	double	Math.PI

Математичні
функції

Операція	Позначення	Приклад
==	рівність	$x==y$
>	більше	$x>y$
<	менше	$x<y$
>=	більше або рівно	$x>=y$
<=	менше або рівно	$x<=y$
!=	не рівно	$x!=y$

Логічні вирази

x	y	$x \& y$ (логічне «і»)	$x \parallel y$ (логічне «або»)	$!x$ (заперечення)
false	false	false	false	true
false	true	false	true	true
true	false	false	true	false
true	true	true	true	false

Логічні операції

ВИРАЗИ ТА ОПЕРАЦІЇ

Пріоритет операцій

1. `()`, `[]`, `..`, (постфікс)++, (постфікс)--, `new`, `sizeof`, `typeof`, `unchecked`
2. `!`, `~`, (ім'я типу), +(унарний), -(унарний), ++(префікс), --(префікс)
3. `*`, `/`, `%`
4. `+`, `-`
5. `<<`, `>>`
6. `<`, `>`, `<=`, `>=`, `is`
7. `==`, `!=`
8. `&`
9. `^`
10. `|`
11. `&&`
12. `||`
13. `?:`
14. `=`, `+=`, `-=`, `*=`, `/=`, `%=`, `&=`, `|=`, `^=`, `<<=`, `>>=`