



Programación y Laboratorio II

Clase 01

Introducción a .NET y C#

Introducción a .NET

- ¿Qué es una plataforma de desarrollo?
- Componentes de .NET
- Proceso de compilación

Proyectos de consola

- Crear una aplicación de consola
- Entrada y salida de datos.
- Formato compuesto.

Introducción a C#

- Características de C#
- Gramática de C# (sintaxis y semántica)
- Common Type System
- Operadores
- Conversiones de tipos de datos
- Sentencias de selección
- Sentencias de iteración



01.

Introducción a .NET

¿Qué es una PLATAFORMA DE

DESARROLLO? Una **plataforma de desarrollo** es un entorno de software que cuenta con un conjunto de herramientas que nos permite construir determinadas aplicaciones de software.

Por ejemplo: editores de código, compiladores, entornos de ejecución, lenguajes de programación, bibliotecas, etc.



¿Qué es un proyecto?

Un **proyecto** de Visual Studio es una colección de archivos de código y recursos como iconos, imágenes, etc., que se compilan conjuntamente mediante el sistema de compilación de MSBuild. La generación de un proyecto suele producir un archivo ejecutable (.exe) o una biblioteca dll.



¿Qué es una **solución**?

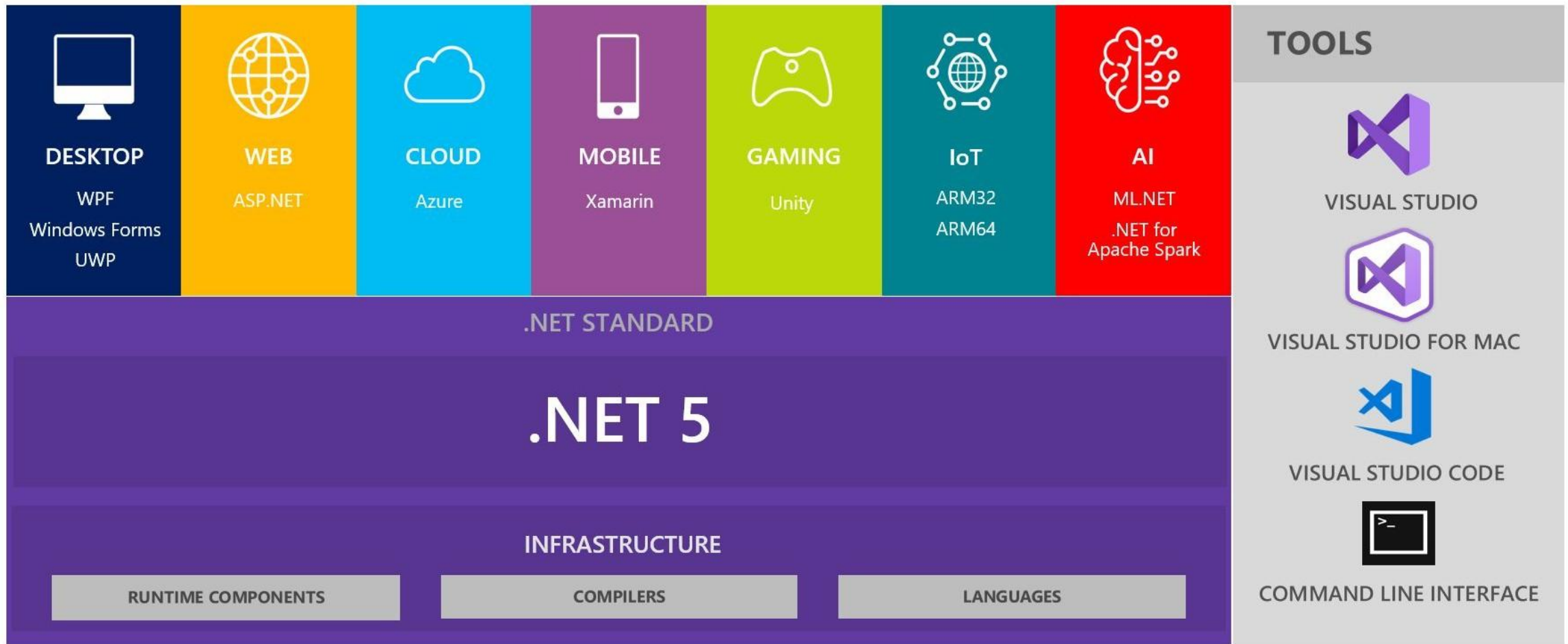
Una **solución** es un contenedor que se usa para organizar uno o más proyectos de código relacionados , por ejemplo, un proyecto de biblioteca de clases y un proyecto de consola.



Componentes de .NET

- Common Language Runtime (CLR)
- Base Class Library (BCL)
- Componentes de infraestructura común (lenguajes, compiladores, sistema de proyectos, etc)
- Frameworks (Windows Forms, WPF, ASP. NET)
- Herramientas de desarrollo (editores de código, IDEs, línea de comandos)

Componentes de .NET

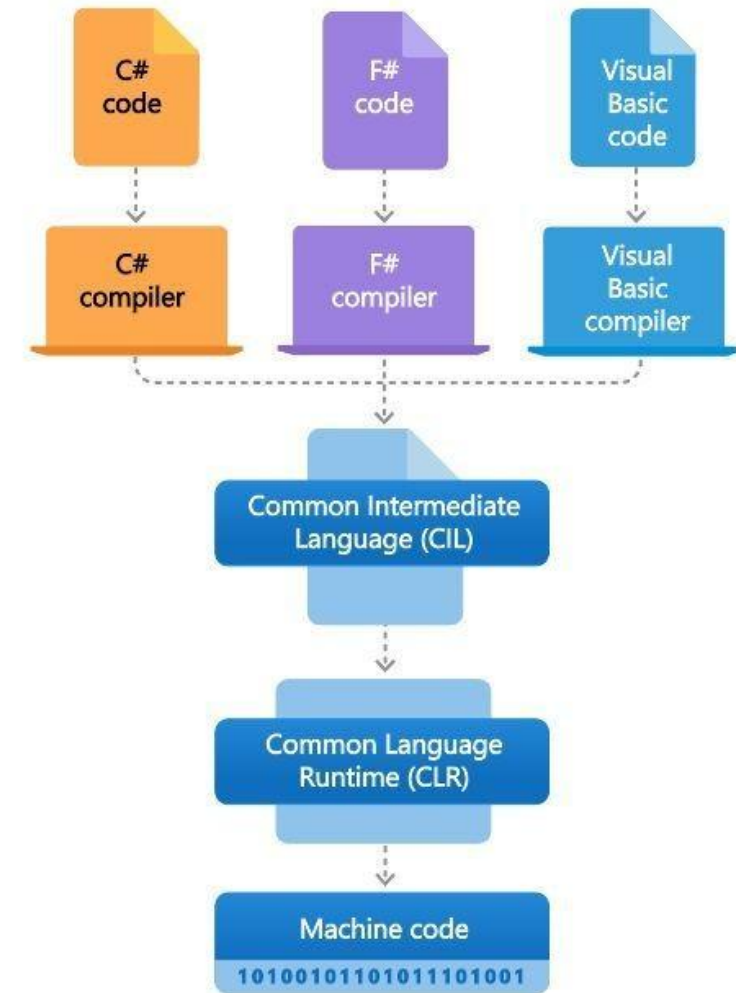


Proceso de compilación

1. Se compilan los archivos que contienen el **código fuente a lenguaje intermedio**.
2. Al ejecutarse la aplicación, el lenguaje intermedio se compila a **lenguaje nativo (máquina)** por el CLR.

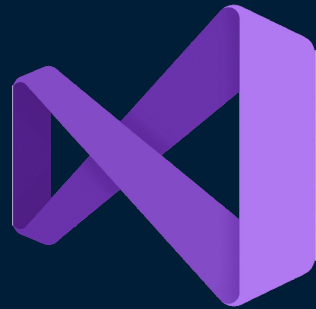
Conceptos clave:

- Archivos con lenguaje intermedio (.exe, .dll)
- Tiempo de ejecución y tiempo de compilación.



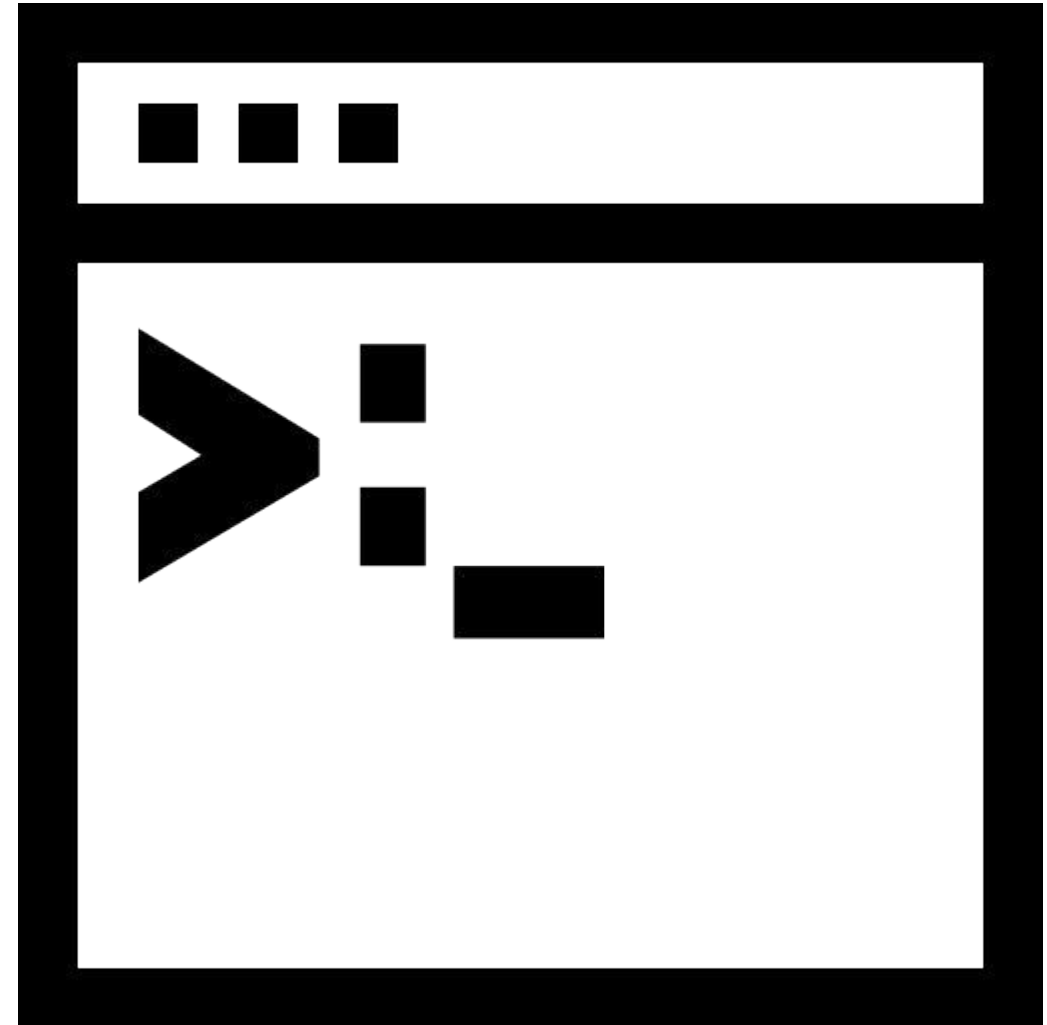
Creando una aplicación de consola

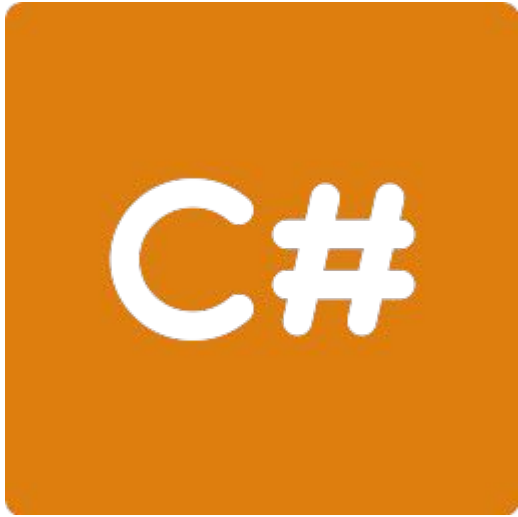
> Hello, world!_



Trabajando con la consola

- Clase System.Console
- Salida de datos
 - Write y WriteLine
 - Formato compuesto
 - Estandar
 - Personalizado
- Secuencias de escape. Prefijo @.
- Entrada de datos
 - ReadLine y ReadKey
- Modificando la consola





02.

Introducción a C#

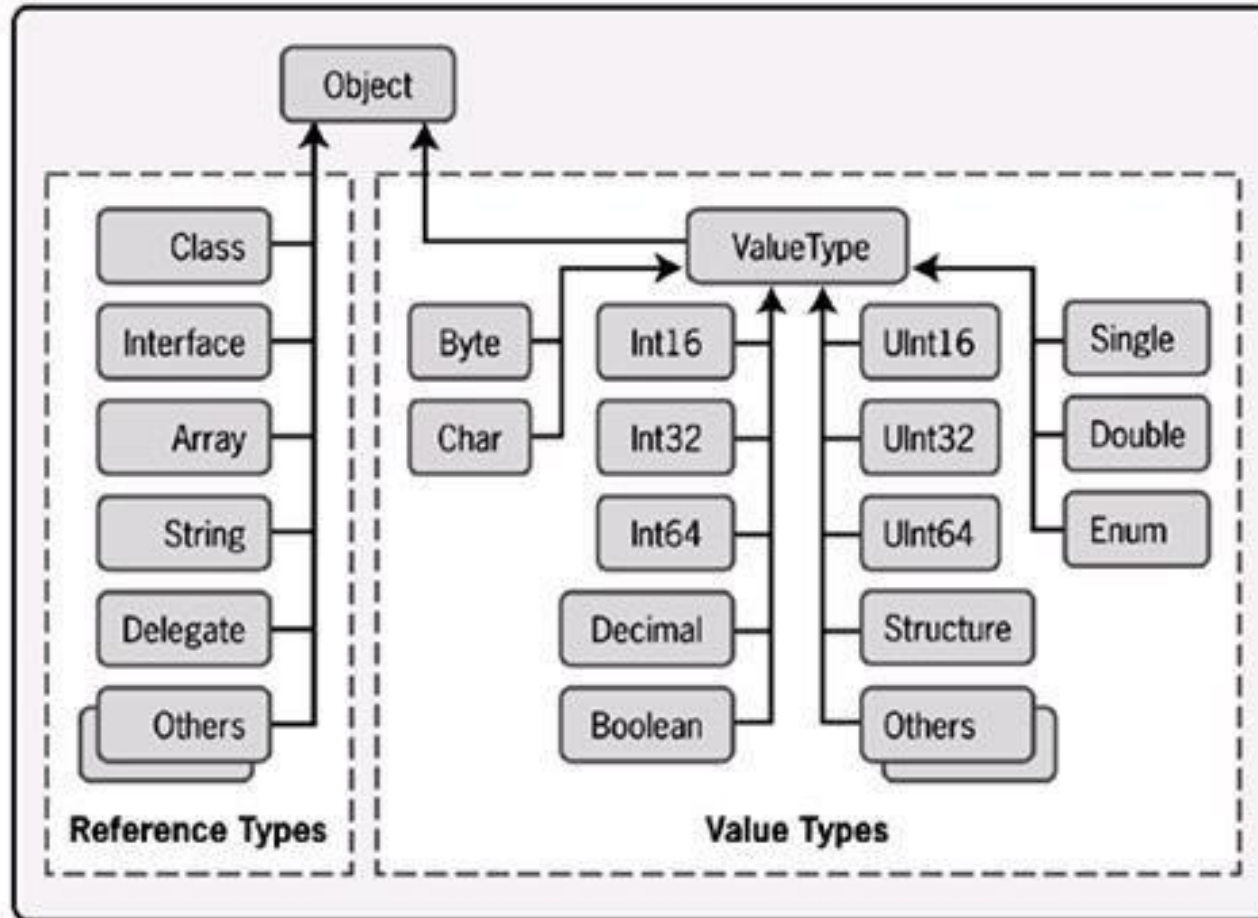
Características de C#

- Compilación híbrida (CIL & CLR)
- Orientado a objetos
- Orientado a componentes
- Seguridad de tipos (tipado estático)
- Garbage collection
- Sistema de tipos unificado
- Case Sensitive

- Sintaxis
 - Sentencias
 - Variables
 - Expresiones
 - Constantes
 - Bloques
 - Comentarios
 - Vocabulario
 - Operadores
 - Palabras reservadas
 - Métodos
 - Tipos, atributos y variables
- Semántica

```
1  if (condicion)
2  {
3      Console.WriteLine("Entiendo sintaxis y semántica.");
4  }
5  else
6  {
7      Console.WriteLine("No entiendo nada.");
8  }
```

Common Type System (CTS)



- Definición Common Type System.
 - Tipos de valor
 - Aliases
-
- Literales y declaración
 - Tipos char, string y bool
 - Numéricos. *sizeof()*. *MinValue*. *MaxValue*.
 - Valores por defecto

Operadores

Operadores aritméticos

Operador	Nombre
+	Suma
-	Resta
*	Multiplicación
/	División
%	Módulo / Resto
++	Incremento
--	Decremento

Operadores de asignación

Operador	Nombre
=	Asignación
+=	Suma y asignación
-=	Resta el valor de la izquierda al de la variable de la derecha y almacena el resultado en la misma variable.
*=	Multiplica el valor de la izquierda por el de la variable de la derecha y almacena el resultado en la misma variable.
/=	Divide el valor de la izquierda por el de la variable de la derecha y almacena el resultado en la misma variable.

Operadores

Operadores de comparación

Operador	Nombre
<	Menor que
>	Mayor que
<=	Menor o igual a
>=	Mayor o igual a

Operadores de igualdad

Operador	Nombre
==	Igualdad
!=	Desigualdad

Operadores lógicos

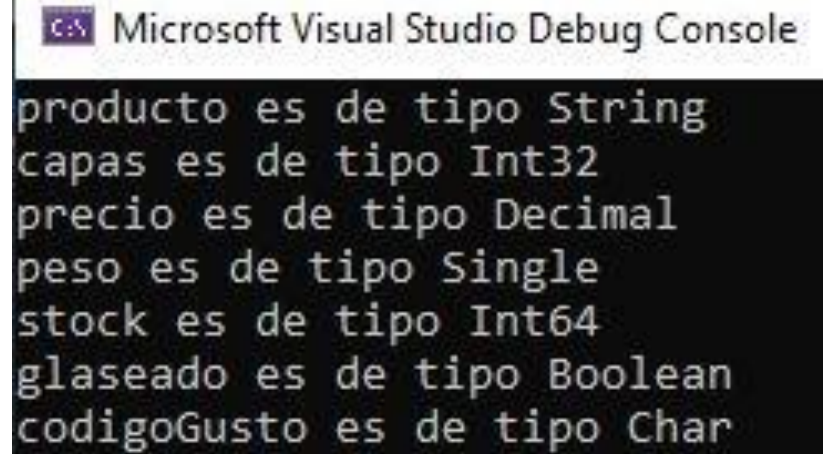
Operador	Nombre
!	Negación lógica
&	AND lógico
&&	AND condicional lógico / cortocircuito
	OR lógico
	OR condicional lógico / cortocircuito



```
1 //Si MetodoUno() es True, entonces NO se evalua MetodoDos()
2
3 if (MetodoUno() || MetodoDos())
4 { }
5
6 //Si MetodoUno() es False, entonces NO se evalua MetodoDos()
7
8 if (MetodoUno() && MetodoDos())
9 { }
```

Inferencia de tipos

```
1 var producto = "Alfajor Capitán del Espacio";
2
3 Console.WriteLine("{0} es de tipo {1}", nameof(producto), producto.GetType().Name);
4
5 var capas = 3;
6
7 Console.WriteLine("{0} es de tipo {1}", nameof(capas), capas.GetType().Name);
8
9 var precio = 99.99M;
10
11 Console.WriteLine("{0} es de tipo {1}", nameof(precio), precio.GetType().Name);
12
13 var peso = 40F;
14
15 Console.WriteLine("{0} es de tipo {1}", nameof(peso), peso.GetType().Name);
16
17 var stock = 1000L;
18
19 Console.WriteLine("{0} es de tipo {1}", nameof(stock), stock.GetType().Name);
20
21 var glaseado = true;
22
23 Console.WriteLine("{0} es de tipo {1}", nameof(glaseado), glaseado.GetType().Name);
24
25 var codigoGusto = 'C';
26
27 Console.WriteLine("{0} es de tipo {1}", nameof(codigoGusto), codigoGusto.GetType().Name);
```



Microsoft Visual Studio Debug Console

```
producto es de tipo String
capas es de tipo Int32
precio es de tipo Decimal
peso es de tipo Single
stock es de tipo Int64
glaseado es de tipo Boolean
codigoGusto es de tipo Char
```

Conversiones de tipos de datos

Implícitas

No interviene el programador (no requieren casteo).

No deberían implicar pérdida de datos.

```
1 // Los float pueden almacenar números más grandes que los int.  
2 // No hay pérdida de datos.  
3  
4 float entero = 15;
```

Explícitas

Interviene el programador (se quiere un casteo).

Podrían implicar pérdida de datos (En el ejemplo, se pierden los decimales)

```
1 // Los double pueden almacenar números más grandes que los int.  
2 // Además los enteros no guardan los decimales.  
3 // Puede haber pérdida de datos.  
4  
5 int entero = (int)15.2;
```

Sentencias de selección



```
1 // Si x es menor a 10 se ejecuta MetodoUno() de lo contrario no pasa nada.  
2  
3 if (x < 10)  
4 {  
5     MetodoUno();  
6 }
```

Sentencias de selección



```
1 // Si x es menor a 10 se ejecuta MetodoUno().
2 // Si x está entre 10 y 20 incluido se ejecuta MetodoDos().
3 // Si ninguno de los casos se cumple se ejecuta MetodoTres().
4
5 if (x < 10)
6 {
7     MetodoUno();
8 }
9 else if (x >= 20)
10 {
11     MetodoDos();
12 }
13 else
14 {
15     MetodoTres();
16 }
```



```
1 // Si x es menor a 10 se ejecuta MetodoUno().
2 // Si x es mayor o igual a 10 se ejecuta MetodoDos().
3
4 if (x < 10)
5 {
6     MetodoUno();
7 }
8 else
9 {
10     MetodoDos();
11 }
```

Sentencias de selección

```
1  int a = 0;
2
3  switch (a)
4  {
5      case 0:
6          MetodoUno();
7          break;
8
9      case 1:
10         MetodoDos();
11         break;
12 }
```

```
1  int a = 9;
2
3  switch (a)
4  {
5      case 0:
6          MetodoUno();
7          break;
8
9      case 1:
10         MetodoDos();
11         break;
12
13     default:
14         MetodoTres();
15         break;
16 }
```

Sentencias de iteración

```
// Partes: declaración, prueba, acción
for (int i = 1; i < 10; i++)
{
}
```

```
string[] nombres = new string[5];

foreach (string auxNombre in nombres)
{
    //auxNombre es un elemento de nombres.
}
```

Sentencias de iteración

```
bool condicion = true;

while (condicion == true)
{
    //En algún momento poner condicion = false
}
```

```
bool condicion = true;

do
{
    //En algún momento poner condicion = false
} while (condicion == true);
```


Debugging en Visual Studio

debugging



a debugger

```
printf("I got here\n");
```

Ejercicios



- I01 - Máximo, mínimo y promedio
- I02 - Error al cubo
- I03 - Los primos

https://codeutnfra.github.io/programacion_2_laboratorio_2_apuntes/