Unidad 2

Fundamentos de los Lenguajes

Capítulo 1

Sintaxis C#

Contenidos

- C#
- Tipos, variables y constantes.
- Conversión de Tipos
- Operadores
- Asignación
- Estructuras de Control
  - Bifurcación
  - Repetición

C#

- Lenguaje creado especialmente para .NET.
- Totalmente OO.
- Sintaxis basada en C (como Java)
- Algo de influencia de VB (ej: propiedades)
- Algo de lenguajes funcionales (ej: lambda expressions)
- Case-sensitive
- Atributos accedidos por un punto
- Todo es tratado como objeto
- Manejo de memoria automatico

Elección del lenguaje

- .NET utiliza **UN** solo runtime (el CLR) y **TODO** lenguaje para .NET compila a MSIL
- No hay diferencias de performance entre C# u otro lenguaje .NET, todo es MSIL.
- El lenguaje usar, en general dependerá de su experiencia previa con otros lenguajes o decisión personal.
  - Si conoce Java, C, C++, JavaScript entonces C#
  - Si conoce Visual Basic 6.0 o VBScript entonces VB.NET
- El más popular en este momento es **C#**

Variables

- ¿Qué es una variable?
- ¿En qué situación se usa una variable?
- Variables en .NET
  - Declaradas en cualquier lugar del código
  - El contenido de la variable tiene que estar de acuerdo con su definición.

Variables - Declaración

- El tipo de variable precede al identificador

```
int x;
decimal y;
rectangle z;
Cliente cli;
```

Variables - Inicialización

- Toda variable debe ser inicializada EXPLICITAMENTE antes de ser usada

```
int tempBalance; //variable local
//ERROR: tempBalance NO ha sido inicializada
System.Console.WriteLine(tempBalance);
```

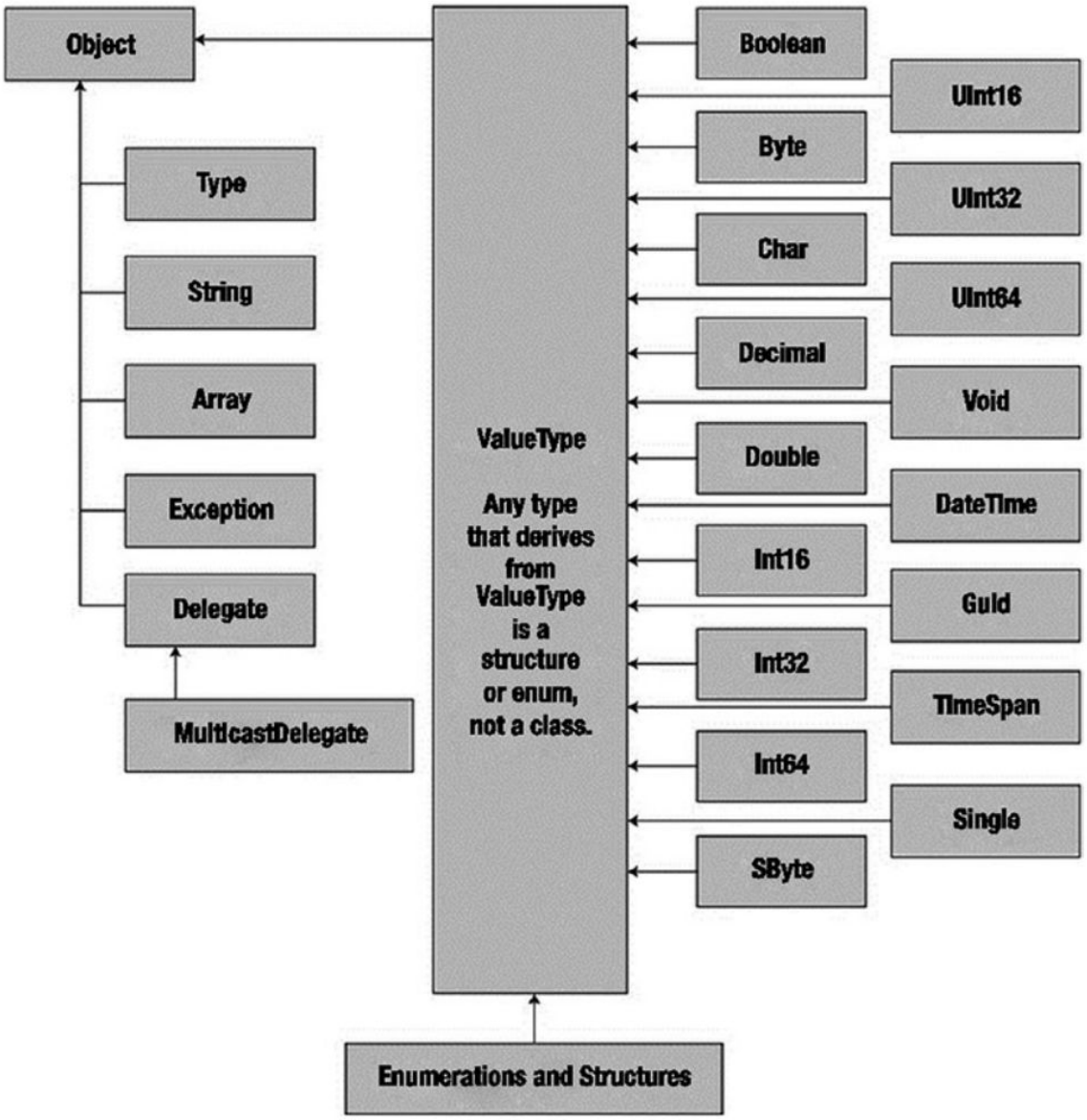
- Se puede inicializar con el literal **default** para un **int** por ejemplo, el default es 0.

- También se puede usar **int variable = new int()** para definir e inicializar una variable con el valor default o directamente se puede hacer **int variable = new()**

Variables - Tipos

C# Shorthand	CLS Compliant?	System Type	Range	Meaning in Life
bool	Yes	Boolean	true or false	Represents truth or falsity
sbyte	No	SByte	–128 to 127	Signed 8-bit number
byte	Yes	Byte	0 to 255	Unsigned 8-bit number
short	Yes	Int16	–32,768 to 32,767	Signed 16-bit number
ushort	No	UInt16	0 to 65,535	Unsigned 16-bit number
int	Yes	Int32	–2,147,483,648 to 2,147,483,647	Signed 32-bit number
uint	No	UInt32	0 to 4,294,967,295	Unsigned 32-bit number
long	Yes	Int64	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	Signed 64-bit to number
ulong	No	UInt64	0 to 18,446,744,073,709,551,615	Unsigned 64-bit number
char	Yes	Char	U+0000 to U+ffff	Single 16-bit Unicode character
float	Yes	Single	$-3.4\ 10^{38}$ to $+3.4\ 10^{38}$	32-bit floating-point number
double	Yes	Double	$\pm 5.0\ 10^{-324}$ to $\pm 1.7\ 10^{308}$	64-bit floating-point number
decimal	Yes	Decimal	$(-7.9 \times 10^{28}$ to $7.9 \times 10^{28})/(10^{0 \text{ to } 28})$	128-bit signed number
string	Yes	String	Limited by system memory	Represents a set of Unicode characters
object	Yes	Object	Can store any data type in an object variable	The base class of all types in the .NET universe

Jerarquía de tipos en .NET



- Todos los tipos heredan de Object
- Por referencia y por valor

Estructuras

- Son Tipos por Valor
- Son un compuesto de otros tipos que facilitan el manejo de datos.
- Pueden ser convertidos a Tipos por Referencia cambiando palabra reservada **struct** por **class**
- Pueden usarse para representar elementos un rango de valores fijos tales como grados, ciclos, cuartos

Enumeraciones

- Son Tipos por Valor.
- Son un conjunto de símbolos relacionados que tienen un valor fijo.
- Se los puede considerar como una colección o arreglo de constantes.
- Permiten proveer una lista de opciones que pueden ser elegibles y auto-descriptibles en reemplazo del uso de banderas.
- Existen en diferentes clases de la BCL y pueden crearse otras para uso especifico.

Variables – Alcance

- Tiempo de vida de una variable

```
string fuera = "Declarada fuera";
string temp = "";

if ( mostrarValores )
{
    string dentro = "Mostrar Dentro";
    temp = dentro;
}
else
{
    temp = fuera;
}
```

Case Sensitive

C# es case-sensitivity

```
system.console.writeline("HOLA");  INCORRECTO
System.Console.WriteLine("HOLA");  CORRECTO
```

Terminaciones de línea

La línea finaliza con un ;

```
//Una linea con mas de un renglon
string sName = sFirstName +
               sLastName;
//El punto y coma indica FINAL de linea
```

Comentarios

C# soporta dos tipos de comentarios

```
// Comentario de una sola linea
string sName = "Juan";
/* Comentario con mas
   de un renglon */
```

Estructuras de Decisión - IF

IF

if (*condición*)

    …

else if (*condición*)

    …

else

    …

Operadores Logicos

C#	Operador
&&	Operador lógico Y
II	Operador lógico O
!	Negación lógica
==	Igual
!=	Distinto

- En C# todas las evaluaciones se hacen por “cortocircuito”

```
//Si Hacer1() es True, entonces
//NO se evalua Hacer2()
if (Hacer1() || Hacer2())
{
}
```

```
//Si Hacer1() es False, entonces
//NO se evalua Hacer2()
if (Hacer1() && Hacer2())
{
}
```

If Else – Ejemplo I

sentencia *if* con varios formatos

```
if (x > 10)
   HacerAlgo();
```

```
if (x < 10)
{
    Hacer1();
    Hacer2();
}
```

```
if (x < 10)
{
     Hacer1();
}
else
{
     Hacer2();
}
```

```
if (x < 10)
{
    Hacer1();
}
else if (x > 20)
{
    Hacer2();
}
else
{
    Hacer3();
}
```

If Else – Ejemplo II

Condiciones con strings

```
if (nombre ==“Juan”)      if (nombre != “Carlos”)     if ( nombre==“Juan” ||
    HacerAlgo();              {                           nombre==“Carlos)
                                 Hacer1();                {
                                 Hacer2();                   Hacer1();
                              }
```

Estructuras de decisión - Case

switch

```
switch (condición)
{
    case valor:
        …
        break;
    default:
        …
        break;
}
```

switch

ejemplos:

```
string Pais = valor;
string Deporte = "";
switch (Pais)
{
    case "Brasil":
     Deporte = "Futbol";
     break;
    case "USA":
     Deporte = "Basquet";
    default:
     Deporte = "Tenis"
     break;
}
```

```
int opcion = valor;
string Deporte = "";
switch (opcion)
{
    case 1:
    Deporte = "Futbol";
    break;
    case 2:
    Deporte = "Basquet";
    default:
    Deporte = "Tenis"
    break;
}
```

Arreglos (arrays)

C# utiliza corchetes [ ] para definir arrays

```
string[] telefonos; //Definicion de un Arreglo de strings
telefonos = new string[3]; //De 3 elementos
telefonos[0] = "1245"; //Seteo del 1er elemento del arreglo

//Definicion y asignacion de una vez
telefonos = new string[] {"1","2","3"};
```

Estructuras de iteración – For

La sentencia for consta de tres partes

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

For

for (contador; expresion; incremento)

{

statements

}

```
for (int item = 1; i<=10; i++)
    Console.WriteLine(i);
```

```
for (int item = 1; i<=10; i++)
{
    Console.WriteLine(i);
}
```

Estructuras de Iteración – for each

- C#: usa la palabra foreach para recorrer arreglos y colecciones

```
string[] nombres = new string[5];
foreach(string auxNombre in nombres)
{
    //auxNombre es de SOLO LECTURA
}
```

foreach

foreach (elemento in grupo)

{

statements

}

```
int multDos = 0;
int noMult = 0;
int[] arrayData = {1, 5, 8, 45, 25};
foreach (int numero In arrayData)
{
    if (numero % 2 ==  0)
        multDos += 1;
    else
        noMult +=1;
}
```

Estructuras de Iteración – While

- C#: usa las palabras while o do - while

```
bool condicion = true;
while (condicion)
{
    //codigo que haga que cambie la condicion
}
```

While - Todas las opciones

C#

```
while (condicion)
{
    …
}

do
{
    …
} while (condicion);
```

Demostracion

Demostración 1

Ejemplos varios en C#

Laboratorios

Laboratorio 1

Estructuras de decisión

Estructuras de iteración

Ejercicios para pensar