Desarrollador de Aplicaciones Web
Programación Web III

Departamento de Ingeniería e Investigaciones Tecnológicas

Introducción a C#

Ing. Matías Paz Wasiuchnik
Ing. Pablo Nicolás Sanchez
Ing. Mariano Juiz

Índice

- Entender los fundamentos de un programa C#.
- Usar tipos predefinidos.
- Escribir expresiones.
- Sentencias condicionales.
- Sentencias de iteración.

Entender los Fundamentos de un Programa C#

- Estructura de un programa C#.
- Formato del código en C#.

Estructura de un Programa C#

- La ejecución del programa comienza en el Main().

- La palabra clave using referencia a recursos de la librería de clases del .Net Framework.

- Un programa esta hecho de muchas sentencias separadas.

- Las sentencias están separadas por ";"

- Las llaves "{ }" son usadas para agrupar sentencias.

```
using System;
class HolaClase {
	static void Main() {
		Console.WriteLine ("Hola Clase");
	}
}
```

Formato del Código en c#

- C# es sensible a mayúsculas y minúsculas.
- Los espacios en blanco son ignorados.
- Para comentarios en una sola línea usar //.
- Para comentarios en múltiples líneas usar /* y */.

```
using System;
class HolaClase {
      static void Main() {

            //CONSOLE.WRITELINE(“No compila por las mayusculas");
            /* Esto es un comentario
              de varias lineas
            */
      }
}
```

Tipos Predefinidos

- ¿Qué son los tipos predefinidos?
- ¿Cómo declarar e inicializar variables.
- ¿Cómo crear y usar constantes.
- ¿Cómo crear y usar enumeraciones.
- Conversiones: Implícitas y explícitas.

¿Qué son los tipos predefinidos?

- Son tipos que se usan para declarar variables.
- Las variables almacenan diferentes tipos de datos.
- Los tipos predefinidos son provistos por C# y el Framework .Net (CTS).
    - Se pueden también definir propios tipos.
- Las variables deben ser declaradas antes de usarlas.

Que Son los tipos predefinidos?

- Tipos de referencia predefinidos: object y string
- Los tipos integrales con signo son sbyte (8 bits), short (16 bits), int (32 bits) y long (64 bits)
- Los tipos integrales sin signo son byte (8 bits), ushort (16 bits), uint (32 bits) y ulong (64 bits)
- Los tipos de punto flotante son float (precisión simple – 7 dígitos y 32 bits) y double (precisión doble 15 dígitos y 64 bits).
- El tipo bool permite representar valores booleanos
- El tipo char permite representar caracteres Unicode de 16 bits.
- El tipo decimal proporciona 28 dígitos significativos y 128 bits

Como Declarar e Inicializar Variables

- Declaración:
  - Asignar un tipo
  - Asignar un nombre
  - Finalizar con ";"

```
int numero;
```

```
string nombre;
```

- Asignación:
  - Usar el operador de asignación
  - Asignar un valor
  - Finalizar con ";"

```
string nombre = "Juana";
```

- Asignar valores literales:
  - Agregar un sufijo

```
decimal deposito = 100M;
```

Como Declarar e Inicializar Strings

- Ejemplo string

```
string s = "Hola Clase"; // Hola Clase
```

- Usando "escape character"

```
string s = "\"Hola\""; // "Hola"
```

```
string s = "Hola\nClase"; // nueva linea
```

- Usando @ en strings

```
string s = @"Hola\n"; // Hola\n
```

Parametros out y ref

Out

Utilizado para obtener valores de una función

```
int x;
Calcular(out x);
```

Ref

Utilizado como input y output de una función. Antes de utilizarlo en la función, se debe inicializar.

```
int x = 0;
Calcular(ref x);
```

Ejercicios

1) Crear una función que devuelva la suma de dos números recibidos por parámetros

2) Crear una función que reciba una cadena de 8 caracteres y retorne en el mismo parámetro la cadena cortada de izquierda a derecha en 4 caracteres.

3) Crear una función que devuelva la fecha y hora actual

4) Escribir un comentario con //

5) Escribir un comentario con /* */

Como Crear y Usar Constantes?

- Se declaran usando la palabra clave "const" seguida de un tipo
- Se debe asignar un valor al momento de la declaración.

```
const int radioTierra = 6378;//km

const long distanciaAlSol = 149600000;//km

const double velocidad = 29.79D;//km sec
```

¿Cómo Crear y Usar Enumeraciones?

Definiendo enumeraciones

```
enum Color {
        Rojo,
        Blanco,
        Verde,
        Marron
}
```

Usando enumeraciones

```
Color miColor = Color.Rojo;
```

Mostrando enumeraciones

```
Console.WriteLine("{0}", miColor);          //Muestra Rojo
```

Ejercicios

1) Crear una enumeración con los días de la semana, comenzando por Domingo con valor 1.

2) Agregar a la enumeración la posibilidad de Imprimir un Texto por cada día de la semana

Conversiones Implícitas y Explícitas

Implícitas

Realizadas por el compilador directamente. Esta operación garantiza que no habrá perdida de información.

```
int x = 123456; // int es un entero de 4-bytes
long y = x; // conversión implícita a long
```

Explícitas

El programador indica de forma explicita la conversión. Puede existir perdida de información.

```
int x = 65537;
short z = (short) x;
// conversión explicita a short, z == 1
```

Conversiones mediante Métodos

Convert

Convierte un tipo de datos base en otro tipo de datos base.

Los tipos base que se admiten son Boolean, Char, SByte, Byte, Int16, Int32, Int64, UInt16, UInt32, UInt64, Single, Double, Decimal, DateTime y String.

```
double numeroReal = 23.15;

int    numeroEntero = System.Convert.ToInt32(numeroReal);
```

Conversiones mediante Métodos

Parse

Convierte un tipo de datos string en otro tipo de datos base. Si no puede lograr la conversión, arroja una excepción

```
int    numeroEntero = int.Parse(“10”);    //ok

int    numeroEntero = int.Parse(“Hola”);    //error - excepción
```

TryParse

Convierte un tipo de datos string en otro tipo de datos base. Si no puede lograr la conversión, devuelve false (booleano)

```
int    numeroEntero;
int.TryParse(“10”, out numeroEntero);    //true y devuelve 10

int.TryParse(“Hola”, out numeroEntero);    //false y devuelve 0
```

Conversiones mediante Métodos

¿Cúal uso?

TryParse es el mejor por el manejo de excepciones que realiza Convert.ToInt32 es mejor que Int32.Parse

```
string s1 = "1234";
string s2 = "1234.65";
string s3 = null;
string s4 = "12345678912345678912345678912345678912345678";

int result;
bool success;

result = Int32.Parse(s1); //-- 1234
result = Int32.Parse(s2); //-- FormatException
result = Int32.Parse(s3); //-- ArgumentNullException
result = Int32.Parse(s4); //-- OverflowException

result = Convert.ToInt32(s1); //-- 1234
result = Convert.ToInt32(s2); //-- FormatException
result = Convert.ToInt32(s3); //-- 0
result = Convert.ToInt32(s4); //-- OverflowException

 success = Int32.TryParse(s1, out result); //-- success => true; result => 1234
 success = Int32.TryParse(s2, out result); //-- success => false; result => 0
 success = Int32.TryParse(s3, out result); //-- success => false; result => 0
 success = Int32.TryParse(s4, out result); //-- success => false; result => 0
```

http://www.codeproject.com/Articles/32885/Difference-Between-Int32-Parse-Convert-ToInt32-and

Escribir Expresiones

- ¿Qué son las expresiones y operadores?

- ¿Cómo determinar la precedencia en operadores?

Que Son las Expresiones y Operadores?

Los operadores son símbolos usados en expresiones.

Operadores comunes	Ejemplo
• Incremento / decremento	++ --
• Aritméticos	* / % + -
• Relacionales	< > <= >=
• Igualdad	== !=
• Condicionales	&& || ?:
• Asignación	= *= /= %= += -= <<= >>= &= ^= |=

Como Determinar la Precedencia de Operadores

Las expresiones son evaluadas de acuerdo a la precedencia del operador:

```
10 + 20 / 5                    el resultado es 14
```

Los paréntesis pueden usarse para controlar el orden de evaluación:

```
(10 + 20) / 5          el resultado es 6
10 + (20 / 5)          el resultado es 14
```

La precedencia de operadores también es determinada por la asociatividad:

- Los operadores binarios son asociativos a la izquierda.
- Los operadores condicionales y de asignación son asociativos a la derecha.

Sentencias Condicionales

Sentencia if.

Sentencia switch.

Sentencia if

if

```
if ( ventas > 10000 ) {
        bono += .05 * ventas;
}
```

if else

```
if ( ventas > 10000 ) {
        bono += .05 * ventas;
}
else {
        bono = 0;
}
```

if
else if

```
if ( ventas > 10000 ) {
        bono += .05 * ventas;
}
else if ( ventas > 5000 ) {
        bono = .01 * ventas;
}
else    {
        bono = 0;
        if ( stock == 0 ) {
         //avisar a compras;
        }
}
```

Sentencia switch

```
switch (miColor){

    case Color.Rojo:
            Console.WriteLine("Rojo");
            break;

    case Color.Verde:
            Console.WriteLine("Verde");
            break;

    case Color.Blanco:
            Console.WriteLine("Blanco");
            break;

    default:
            Console.WriteLine("Color no encontrado");
}
```

Sentencias de Iteración

- Bucle for.
- Bucle foreach.
- Bucle while.
- Bucle do while.

Bucle for

```
for (inicializador; condición; iterador) {
    sentencias;
}
```

Ejemplo

```
for (int i = 0; i < 10; i++) {
    Console.WriteLine("i = {0}",i);
}

for ( int j = 100; j > 0; j -= 10 ) {
    Console.WriteLine("j = {0}", j);
}
```

Bucle foreach

Ejemplo

```
Console.Writeline("Ingrese un nombre");
string nombre = Console.ReadLine();

Console.WriteLine("Deletreo {0}: \n", nombre);

foreach( char c in nombre){

 Console.WriteLine("{0} - ", c);
}
```

Bucle while

Ejemplo

```
bool leerTabla,error;

// . . .

while ( leerTabla == true ) {
    ObtenerRegistro();

    if(error == true)
        break;
}
```

- **continue, break**

Bucle do while

Ejemplo 1

```
do {
    Console.Writeline("Ingrese una opcion, s para salir");
    string opcion = Console.ReadLine();
    //hacer algo

} while (opcion != "s");
```

Ejemplo 2

```
int i = 1;
do {
    Console.WriteLine ("{0}", i++);
} while (i <= 10);
```

Desarrollador de Aplicaciones Web
Programación Web III

Departamento de Ingeniería e Investigaciones Tecnológicas

Muchas gracias

Ing. Matías Paz Wasiuchnik
Ing. Pablo Nicolás Sanchez
Ing. Mariano Juiz