UNIVERSIDAD AUTÓNOMA DE SINALOA

Facultad de Informática Culiacán

Estructura de un Programa en C#

Instructor:

MC. Gerardo Gálvez Gámez

gerardo.galvez@uas.edu.mx

Septiembre de 2017

Temas

- Estructura de un programa C#
- La clase
- El método Main
- La sentencia using y el espacio de nombres System
- Operaciones básicas de entrada/salida
- La clase Console
- Los métodos Write y WriteLine
- Formatos numéricos
- Los métodos Read y ReaLine
- Comentarios en aplicaciones
- Tratamiento de excepciones
- Compilación, ejecución y depuración
- Llamadas al compilador
- Depuración

Hola, mundo

```
using System;

class Hola//Nombre de la Clase
{//INICIO CLASE
  public static void Main()
  {//INICIO
    Console.WriteLine("Hola, mundo");
  }//FIN
}//FIN CLASE
```


La clase

- Una aplicación C# es una colección de clases, estructuras y tipos
- Una clase es un conjunto de datos y métodos
- Sintaxis

```
class nombre
{
      ...
}
```

- Una clase no puede abarcar más de un archivo

El método Main

- Al escribir Main hay que:
  - Utilizar una "M" mayúscula, como en "Main"
  - Designar un **Main** como el punto de entrada al programa
  - Declarar **Main** como **public static void Main**
- Un Main puede pertenecer a múltiple clases
- La aplicación termina cuando Main acaba o ejecuta un return

La sentencia using y el espacio de nombres System

- .NET Framework ofrece muchas clases de utilidad
  - Organizadas en espacios de nombres
- System es el espacio de nombres más utilizado
- Se hace referencia a clases por su espacio de nombres

```
System.Console.WriteLine("Hola, mundo");
```

- **La sentencia using**

```
using System;
…
Console.WriteLine("Hola, mundo");
```

Operaciones básicas de entrada/salida

- La clase Console
- Los métodos Write y WriteLine
- Los métodos Read y ReadLine

Introducción al Lenguaje C# • FIUAS

La clase Console

- Permite acceder a las secuencias estándar de entrada, salida y error.
- Sólo tiene sentido para aplicaciones de consola:
  - Entrada estándar: teclado
  - Salida estándar: Pantalla
  - Error estándar: Pantalla
- Es posible redireccionar todas las secuencias

Los métodos Write y WriteLine

- Console.Write y Console.WriteLine muestran información en la pantalla de la consola:
  - **WriteLine** envía un fin de línea/retorno de carro

```
•Console.WriteLine(99);
•Console.WriteLine("Hola, mundo");
•Console.WriteLine("La suma de {0} y {1} es {2}",
100, 130, 100+130);
```

- Ambos métodos son sobrecargados
- Es posible emplear una cadena de formato y parámetros:
  - Formatos de texto
  - Formatos numéricos

Introducción al Lenguaje C# • FIUAS

Formato

Se puede utilizar el parámetro de la cadena de formato para especificar anchuras de campos y para indicar si los valores en esos campos deben ir justificados a derecha o a izquierda, como se ve en el siguiente código:

```
Console.WriteLine("\"Justificación a la izquierda en un campo de anchura 10: {0, -10}\"", 99);
Console.WriteLine("\"Justificación a la derecha en un campo de anchura 10: {0,10}\"", 99);
```

Esto hará que en la consola aparezca lo siguiente:

"Justificación a la izquierda en un campo de anchura 10: 99 "
"Justificación a la derecha en un campo de anchura 10: 99"

Formatos numéricos

La sintaxis completa para la cadena de formato es:

{N,M: FormatString},donde:

- N es el número del parámetro,
- M es la anchura y justificación del campo, y
- FormatString indica cómo se deben mostrar los datos numéricos.

La tabla siguiente resume los valores que puede adoptar **FormatString**. Opcionalmente, en todos estos formatos se puede especificar el número de dígitos que se desea mostrar o al que se debe redondear.

Valor	Significado
C	Muestra el número como una unidad monetaria, usando el símbolo y las convenciones de la moneda local.
D	Muestra el número como un entero decimal.
E	Muestra el número como usando notación exponencial (científica).
F	Muestra el número como un valor en coma fija.
G	Muestra el número como un valor entero o en coma fija, dependiendo del formato que sea más compacto.
N	Muestra el número con comas incorporadas.
X	Muestra el número utilizando notación hexadecimal.

Introducción al Lenguaje C# • FIUAS

Ejemplos de formatos numéricos

```
Console.WriteLine("Formato de moneda - {0:C} {1:C4}", 88.8, □  -888.8);
Console.WriteLine("Formato entero - {0:D5}", 88);
Console.WriteLine("Formado exponencial - {0:E}", 888.8);
Console.WriteLine("Formato de punto fijo - {0:F3}", □  888.8888);
Console.WriteLine("Formato general - {0:G}", 888.8888);
Console.WriteLine("Formato de número - {0:N}", 8888888.8);
Console.WriteLine("Formato hexadecimal - {0:X4}", 88);
```

El resultado de ejecutar este código es el siguiente:

```
Formato de moneda - $88.80 ($888.8000)
Formato entero - 00088
Formado exponencial - 8.888000E+002
Formato de punto fijo - 888.889
Formato general - 888.8888
Formato de número - 8,888,888.80
Formato hexadecimal – 0058
```

Formato de Salida

Carácter	Descripción	Ejemplos	Resultados
C o c	Moneda	Console.Write("{0:C}", 2,5); Console.Write("{0:C}", -2,5);	$2.50 ($2.50)
D o d	Decimal	Console.Write("{0:D5}", 25);	00025
E o e	Científico	Console.Write("{0:E}", 250000);	2,500000E+005
F o f	Punto fijo	Console.Write("{0:F2}", 25); Console.Write("{0:F0}", 25);	25.00 25
G o g	General	Console.Write("{0:G}", 2,5);	2.5
N o n	Número	Console.Write("{0:N}", 2500000);	2,500,000.00
X o x	Hexadecimal	Console.Write("{0:X}", 250); Console.Write("{0:X}", 0xffff);	FA FFFF

Los métodos Read y ReadLine

- Console.Read y Console.ReadLine leen información introducida por el usuario
  - **Read** lee el siguiente carácter
  - **ReadLine** lee toda la línea introducida

```
•string input = Console.ReadLine( );
•Console.WriteLine("{0}", input);
```

Comentarios en aplicaciones

- Los comentarios son importantes:
  - Una aplicación con los comentarios adecuados permite a un desarrollador comprender perfectamente la estructura de la aplicación.
- Comentarios de una sola línea:

```
// Obtener el nombre del usuario
Console.WriteLine("¿Cómo se llama? ");
name = Console.ReadLine( );
```

- Comentarios de varias líneas:

```
/* Encontrar la  mayor raíz
de la ecuación cuadrática */
x = (…);
```


Tratamiento de excepciones

```
using System;
public class Hola
{
  public static void Main(string[ ] args)
  {
     try
      {

         Console.WriteLine(args[0]);


      }
      catch (Exception e) {
         Console.WriteLine("Excepción en
        ➥{0}", e.StackTrace);
     }
  }
}
```

Compilación, ejecución y depuración

- Llamadas al compilador
- Ejecución de la aplicación
- Demostración: Compilación y ejecución de un programa C#
- Depuración
- Demostración: Uso del depurador de Visual Studio
- Las herramientas del SDK
- Demostración: Uso del ILDASM

Conmutadores comunes del compilador

Es posible especificar distintos conmutadores para el compilador de C# usando el comando **csc**. La siguiente tabla describe los conmutadores más comunes.

Conmutador	**Significado :**
/?, /help	Muestra las opciones del compilador en la salida estándar.
/out	Especifica el nombre del ejecutable.
/main	Especifica la clase que contiene el método **Main** (si en la aplicación hay más de una clase que incluya un método **Main**).
/optimize	Activa y desactiva el optimizador de código.

Conmutadores comunes del compilador

Conmutador	Significado
/warn	Fija el nivel de aviso del compilador.
/warnaserror	Trata todos los avisos como errores que interrumpen la compilación.
/target	Especifica el tipo de aplicación generada.
/checked	Indica si un desbordamiento aritmético genera una excepción en tiempo de ejecución.
/doc	Procesa comentarios de documentación para crear un archivo XML.
/debug	Genera información sobre la depuración.

Ejecución de la aplicación

- Ejecución desde la línea de comandos
  - Escribir el nombre de la aplicación:

    - csc /debug+ /out:Saludo.exe Hola.cs

- Ejecución desde Visual Studio
  - Pulsar **Start Without Debugging** en el menú **Debug**

Depuración

- Excepciones y depuración JIT
- El Visual Studio Debugger
  - Configuración de puntos de interrupción e inspecciones
  - Seguimiento del código paso a paso
  - Examen y modificación de variables

Codificar en el Lenguaje **C#** el siguiente Algoritmo:

Configurar el Entornos de Desarrollo y depuración en nuestro equipo

- **Visual Studio**

- **SharpDevelop**
- **Mono**
- **MonoDevelop**
- **Eclipse y Emonic**
- **Notepad ++**

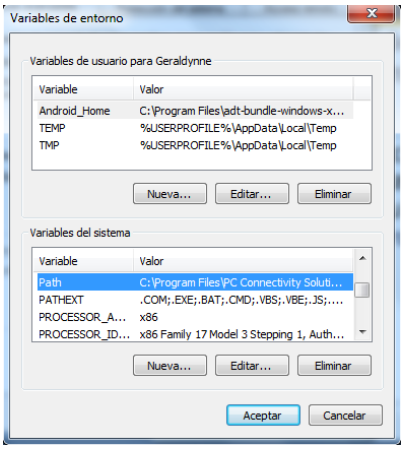
Microsoft.NET\Framework

Configuración del equipo para llamar al compilador **CSC**

1. Ubicar la dirección path, donde se encuentra instalado el compilador csc, en nuestro equipo:
    1. C:\Windows\Microsoft.NET\Framework /

Configuración del equipo para llamar al compilador **CSC**

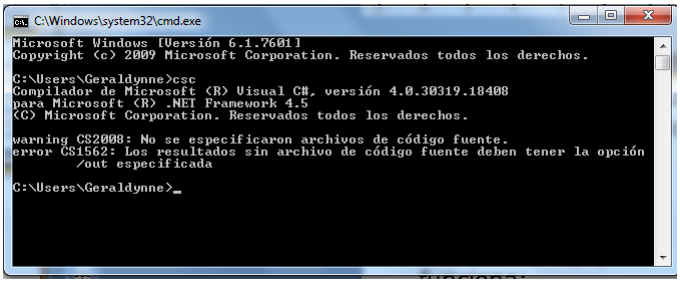
2. Agregar la dirección path, donde se encuentra instalado el compilador csc, como una variable del sistema en nuestro equipo:
   1. Panel de control/Sistema/Variables de Entorno/



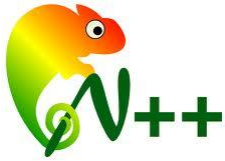
Introducción al Lenguaje C# • FIUAS

Configuración del equipo para llamar al compilador **CSC**

3. Comprobar que la variable del sistema en nuestro equipo funciona:
   1. Iniciar una ventana de comando (cmd).
   2. Escribir el comando csc.



Codificando el Algoritmo



Introducción al Lenguaje C# • FIUAS

Algoritmo en Pseudocódigo

```
//Objetivo: Determinar el área de un triangulo rectángulo
//Programador: MC. Gálvez Gámez Gerardo
//Fecha: __/Agosto/2016

INICIO
    //Definición de Constantes y Variables
    CONST ENTERO  Base=3
    REAL  Altura, Area
    //Lectura
    IMPRIMIR "Teclea la Altura del Triangulo: "
    LEER Altura
    //Procesamiento de los Datos, calcular el área
     Area = Base * Altura / 2
    //Impresión de Resultados
    IMPRIMIR "Área del Triangulo: ", Area
FIN
```

Compilar el código fuente para generar el .exe

Introducción al Lenguaje C# • FIUAS

Compilación

- Ejecución desde la línea de comandos
  - Escribir el nombre de la aplicación:

    - **csc** [/debug+ /out:Saludo.exe] **AreaTriangulo.cs**

- Resultado esperado:

  - AreaTriangulo.exe

Actividades
ExtraClases

Objetivo
El alumno demostrara la habilidad alcanzada en clases, para codificar el pseudocódigo del siguiente problemas, que utiliza procedimiento de solución secuencial.

Pseudocodigo

Objetivo: Determinar el promedio de un alumno .
Programador: Gerardo Gálvez Gámez
Fecha: __/Agosto/2016

```
INICIO
   //Definición de Constantes y Variables
   CONST ENTERO NumeroParciales=3
   REAL CalificacionParcial1, CalificacionParcial2, CalificacionParcial3
   REAL Promedio
   //Lectura
   IMPRIMIR "Teclea la Calificación del Parcial #1;"
   LEER CalificacionParcial1
   IMPRIMIR "Teclea la Calificación del Parcial #2;"
   LEER CalificacionParcial2
   IMPRIMIR "Teclea la Calificación del Parcial #3;"
   LEER CalificacionParcial3
```

Continuación

```
    //Proceso
    Promedio=(CalificacionParcial1  + CalificacionParcial2  +
                    CalificacionParcial3  )/NumeroParciales
    //Impresión
    IMPRIMIR "Su promedio es: ",Promedio
FIN
```

Introducción al Lenguaje C# - FIUAS

Preguntas?