TECNOLÓGICO DE MONTERREY.

Estándar de Codificación para C#

Departamento de Ciencias Computacionales
Comité de Programación
Tecnológico de Monterrey Campus Monterrey

Antecedentes

Entre los principales propósitos de tener una convención de código están: entender el código más rápidamente; tener una visión consistente del código de manera que el lector pueda enfocarse en el contenido; facilitar el mantenimiento y aplicar las mejores prácticas del desarrollo.

Lineamientos

Los lineamientos de codificación son:

Propósito	Guía para la codificación de programas hechos con lenguaje C#
Nombre del archivo que almacena el programa	• Debe ser un nombre corto y descriptivo del propósito del programa. • Debe iniciar con mayúscula, formato Pascal[1]. • No debe incluir caracteres especiales, acentos o espacios. • Si requiere el uso de dos o más palabras cada una debe iniciar con mayúscula. • Se debe utilizar la extensión **.cs** • Preferiblemente hacer programas cortos, ya que dividir el código hace que las estructuras sean mas claras. • Poner todas las clases en archivos separados y nombrar el archivo con el mismo nombre de la clase.

	Ejemplos de nombres de archivos:
	`CalculoHipotenusa.cs`
	`ConversionesNumericas101.cs`
	
	Ejemplos de nombres de archivos incorrectos:
	`calculo.cs`
	`Calculoarea.cs`
	`Calculo Area.cs` (*recordar No poner espacios en el nombre*)
	`Calculo#2.cs`

Nombres de variables	• El nombre de una variable debe ser representativo, proporcionando información sobre su contenido.
	• Si se requiere que el nombre de la variable esté compuesto por varias palabras, se utilizará la notación *camelCase*.
	• Los nombres de variables:
	  ○ Sólo contienen letras, dígitos o el caracter _.
	  ○ Deben comenzar con una letra que representa el tipo de dato.
	  ○ No acentos, diéresis, ni letra ñ.

[1] Formato camelCase: cada palabra en Mayúscula, excepto la primera.
Formato Pascal: el primer caracter de cada palabra está en mayúscula.

<table>
<tr><td></td><td>

No debe ser palabra reservada.
Son case sensitive.
Utilizar mayúsculas para constantes.

Para conservar un estándar en los nombres de variables, se utilizará el formato Hungarian notation[2], para el cual se añade un prefijo al nombre de la variable, dependiendo del tipo de dato. Por ejemplo:
<table>
<tr><th>Prefijo</th><th>Tipo de dato</th><th>Ejemplo</th></tr>
<tr><td>b</td><td>booleano</td><td>bAprobado</td></tr>
<tr><td>by</td><td>byte</td><td>byContador</td></tr>
<tr><td>c</td><td>char</td><td>cSexo</td></tr>
<tr><td>d</td><td>double</td><td>dPrecio</td></tr>
<tr><td>de</td><td>decimal</td><td>dePesoEspecífico</td></tr>
<tr><td>f</td><td>float</td><td>fEstatura</td></tr>
<tr><td>i</td><td>int</td><td>iEdad</td></tr>
<tr><td>l</td><td>long</td><td>lGranTotal</td></tr>
<tr><td>s</td><td>string</td><td>sNombre</td></tr>
<tr><td>iArr</td><td>arreglo de enteros</td><td>iArrLista</td></tr>
<tr><td>dArr</td><td>arreglo de double</td><td>dArrPrecios</td></tr>
<tr><td>fArr</td><td>arreglo de float</td><td>fArrEstaturas</td></tr>
<tr><td>bArr</td><td>arreglo de boolen</td><td>bArrBoleanos</td></tr>
<tr><td>objArr</td><td>arreglo de Relojes</td><td>relArr1</td></tr>
<tr><td>objArr</td><td>arreglo de Ratones</td><td>ratArrRatonesBuenos</td></tr>
<tr><td>objMat</td><td>matriz de Relojes</td><td>relMatRelojes</td></tr>
<tr><td>objMat</td><td>matriz de Ratones</td><td>ratMatRatones</td></tr>
</table>
</td></tr>
<tr><td>Nombres de objetos para componentes gráficos</td><td>
Es recomendable utilizar un prefijo para cada tipo de objeto[2] de Componentes Gráficos. Ejemplos:
<table>
<tr><th>Objeto</th><th>Prefijo</th><th>Ejemplo</th></tr>
<tr><td>Button</td><td>btn</td><td>btnSubmit</td></tr>
<tr><td>CheckBox</td><td>chk</td><td>chkReadOnly</td></tr>
<tr><td>ComboBox</td><td>cbo</td><td>cboEnglish</td></tr>
<tr><td>Image</td><td>img</td><td>imgInicial</td></tr>
<tr><td>Label</td><td>lbl</td><td>lblHelpMessage</td></tr>
<tr><td>ListBox</td><td>lst</td><td>lstPaises</td></tr>
<tr><td>PictureBox</td><td>pic</td><td>picPaisaje</td></tr>
<tr><td>TextBox</td><td>txt</td><td>txtGetText</td></tr>
</table>
</td></tr>
<tr><td>Capitalización y nomenclatura</td><td>

Los nombres de variables se escriben en formato camelCase (cada palabra en Mayúscula, excepto la primera).
Los nombres de Clases y Métodos se escriben en formato Pascal (el primer caracter de cada palabra está en mayúscula).
Los nombres de constantes se escriben todas con mayúsculas.
El nombre del programa se escribe en formato Pascal.

Nomenclatura para las Clases.

El nombre de una clase debe componerse de sustantivos.
Utilizar la capitalización tipo Pascal.

</td></tr>
</table>

[2] Formato *Hungarian Notation*. Uso de prefijos para denotar el tipo de variable.

	• Ejemplos: Vehiculo, CamionPasajeros, CamionCarga **Nomenclatura para Archivos.** • El nombre lógico de un archivo debe corresponder con el tipo de archivo que sea, ya sea de "entrada" o de "salida". • Utilizar la capitalización tipo camelCase. • Ejemplos: StreamReader strEntrada = File.OpenText("c:\\temp\\" + sNomArchivoEntrada); StreamWriter stwSalida = File.CreateText("c:\\temp\\" + sNomArchivoSalida)
Literales numéricas	Al utilizar literales numéricas, usar 0.5 en lugar de .5
Declaraciones	**Numero de declaraciones por línea.** • Es recomendable declarar todas las variables locales al inicio de cada método, y las variables globales al inicio del programa. • Utilizar una declaración de variable por línea.

	Ejemplo:
	`int iEdad; // edad de la persona`
	`double dEstatura; // estatura de la persona`

Inicializaciones	Iniciar las variables locales tan pronto como sean declaradas.

	Ejemplo:
	`string sSaludo = "Hola";`
	`int iTiempoMaximo = 8;`

Orden dentro del programa	Todos los programas deben contener la siguiente estructura: 1. Estatutos *using* necesarios. 2. Inicio Namespace 3. Comentarios de inicio del programa o clase. 4. Inicio de la Clase 5. Variables de instancia, Constructores, Propiedades, Métodos.
Comentarios de inicio del programa o clase	Iniciar todos los programas con una documentación que tenga: • Nombre del programa, • Descripción • Autor/autores -matrícula, • Fecha de creación/modificación, y • Versión. Utilizar la documentación propia de C#, esto es, utilizando **///**

	Ejemplos de comentarios de inicio
	`/// <summary>`
	`/// EM2014_EjemploMetodos.`
	`/// Este programa calcula el area de un círculo, utilizando méto`
	`/// Autor: Donn Perfecto  Matrícula: 12345678`
	`/// </summary>`

Comentarios de métodos	Para cada método, escribir comentarios que contengan: nombre del método, propósito del método, parámetros que requiere y si tiene valor de retorno. Utilizar la documentación propia de C#, esto es, utilizando **///** antes del encabezado del método.

<table>
<tr><td></td><td>Ejemplos de comentario de método:

```
/// <summary>
/// CalculaArea.
/// Calcula el area de un circulo, dado el radio.
/// </summary>
/// <param name="iRadio">Radio del circulo de tipo entero.</param>
/// <returns>Regresa el area del circulo de tipo double.</returns>
```

</td></tr>
<tr><td>Comentarios en general</td><td>• Se debe documentar el código para que pueda ser entendido por desarrolladores externos y por uno mismo.
• Cada comentario debe iniciar con // o /* ... */ o ///
• Los comentarios se deben colocar antes de la instrucción referenciada, o a un lado del estatuto que se quiere comentar.
• Se deben hacer comentarios de valor y evitar hacer comentarios que son obvios al leer el estatuto referenciado.</td></tr>
<tr><td></td><td>Ejemplo de comentarios correctos.

```
// Se lee el nombre
sNombre = Console.ReadLine();

// Se calcula el area del circulo
dArea = 3.1416 * dRadio * dRadio;

iContador = 0;   // inicializar el contador de alumnos
```

Ejemplo de comentarios que no tienen valor.

```
// leo
sNombre = Console.ReadLine();

// calculo
dArea = 3.1416 * dRadio * dRadio;

iContador = 0;  // hacer contador igual a cero
```

</td></tr>
<tr><td>Espacios y líneas en blanco</td><td>Los elementos del programa se deben escribir con suficiente espacio para que no aparezcan amontonados.
• En las operaciones aritméticas y estatutos de asignación se debe dejar un espacio en blanco entre cada operador.
• Antes de cada encabezado de método se deberá dejar una línea en blanco.
• Cada línea debe de contener solo un estatuto.</td></tr>
<tr><td></td><td>Ejemplo correcto de espacios en blanco entre operadores:

```
dD = dB * dB - 4.0 * dA * dC;
```

</td></tr>
<tr><td>Longitud de línea de código</td><td>Evitar líneas mayores de 80 caracteres.
Cuando una frase no quepa en una sola línea, pasarla a la siguiente línea siguiendo estos principios:
o Hacer un salto de línea después de una coma
o Hacer un salto de línea después de un operador
o Alinear la nueva línea con el principio de la expresión para que queden al mismo nivel que la línea anterior.</td></tr>
</table>

	Ejemplo de romper línea con llamada a método:
	
	`LlamadaAMetodo(iDato1, iDato2,`
	`iDato3, iDato4, iDato5);`
	
	Ejemplo de romper línea en una expresión aritmética:
	
	`dRes = dA * dB / (dC - dG + dF) +`
	`4 * dZ;`

Sangría	• Utilizar tabuladores para la sangría. • Si se quiere aumentar la sangría, marcar el párrafo y aumentar el nivel de sangría con tab, y con Shift + tab se disminuye la sangría. Esto funciona con cualquier tipo de editor de texto. Aquí se define Tab como el carácter estándar de sangría. • No utilizar espacios para la sangría, utilizar los tabuladores. • Incluir sangría de un Tab por cada nivel de anidamiento de las estructuras de control (if / for / while). • Incluir sangría de un Tab en todas las instrucciones que se encuentran dentro de un método.

	Ejemplo de uso de sangría:
	`while (iContador < 10){`
	`if (iContador % 2 == 0)`
	`iTotal = iTotal + iContador;`
	`iContador = iContador + 2;`
	`}`

Nombre de métodos	• Los nombres de métodos deben de Iniciar con mayúscula. • Si se requieren dos palabras o más para el identificador, deben iniciar con mayúscula (formato *Pascal*)
	Ejemplo de nombre de método correcto: `void CalculaImpuesto(int iValor)`

Sentencia If, if-else, if else-if else.

Las sentencias con if, if-else and if else-if else deben estar alineadas así:

```
if (condición) {
    HazAlgo();
    ...
}


if (condición) {
    HazAlgo();
    ...
} else {
    HazOtraCosa();
    ...
}


if (condición) {
    HazAlgo ();
    ...
} else if (condición) {
    HazOtraCosa();
    ...
} else {
    HazOtraCosaDistinta();
    ...
}
```

Sentencia con switch

Una sentencia con switch debe escribirse:

```
switch (condición) {
case A:
    ...
break;
case B:
    ...
break;
default:
    ...
break;
}
```

Sentencia con For

Una sentencia con *for* debe tener la siguiente forma general:

```
for (inicialización; condición; actualización)  {
    ...
}
```

Ejemplo:

```
for (int iK = 0; iK < 5; iK++) {
    ...
}
```

Sentencia con Try-catch	Una sentencia con try-catch debe seguir la siguiente forma:
	
	`try {`
	`  ...`
	`} catch (Exception e) {}`
	
	`try {`
	`  ...`
	`} catch (Exception e) {`
	`  ...`
	`}`
	
	`try {`
	`  ...`
	`} catch (Exception e) {`
	`  ...`
	`} finally {`
	`  ...`
	`}`

Sobre el uso de llaves	• La llave que abre se debe colocar en una sola línea y alineada a la izquierda, de acuerdo al nivel de indentación que corresponda. • La llave que cierra se debe colocar en una sola línea, y alineada a su correspondiente llave de apertura. • Es válido no utilizar llaves cuando se tenga una sola instrucción dentro de los estatutos de control (if, while, for).
Mejores prácticas	A continuación se mencionan algunas de las mejores prácticas de programación: • Crear un directorio para cada proyecto o ejercicio, con nombre corto y descriptivo. Utilizar formato Pascal. o Ejemplo: EneMay2016_CalculoArea.cs • Es recomendable que los parámetros de los constructores tengan el mismo nombre que los atributos de la clase. • Dentro de un método, si el valor de retorno es booleano, es preferible regresar una condición, en lugar de hacer un if. • Evitar ifs innecesarios dentro de un ciclo. • En una condición, una variable booleana no es necesario compararla con los valores de true y false. • En la medida de lo posible evitar que las variables de instancia sean públicas. • En la medida de lo posible evitar el uso de variables globales. • Promover la modularidad y la Programación Orientada a Objetos. • Buscar siempre la eficiencia en el código y en los recursos de memoria utilizados. • Siempre reconocer explícitamente los autores del código reutilizado, incluso si se reutiliza código encontrado en Internet, o código inspirado en el desarrollado por el profesor u otro estudiante.

Referencias bibliográficas

C# Coding Conventions - MSDN - Microsoft. msdn.microsoft.com/en-us/library/ff926074.aspx
C# Coding Style. Mike Krüger. icsharpcode.net

Documento elaborado por:

Comité del Área de Programación, Ciencias Computacionales, Campus Monterrey.
Profesores responsables: Ing. Jakeline Marcos Abed, Luis Humberto González.

Ejemplo completo

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace EM2014_EjemploMetodos
{
    /// <summary>
    /// EM2014_EjemploMetodos.
    /// Este Programa calcula el area de un circulo, utilizando métodos.
    /// Autor: Donn Perfecto   Matrícula: 12345678
    /// </summary>
    class Program
    {

        /// <summary>
        /// CalculaArea.
        /// Calcula el area de un circulo, dado el radio.
        /// </summary>
        /// <param name="iRadio">Radio del circulo de tipo entero.</param>
        /// <returns>Area del circulo de tipo double.</returns>
        static double CalculaArea(int iRadio) {
            return (Math.PI * Math.Pow(iRadio, 2));
        }

        /// <summary>
        /// Asteriscos.
        /// Despliega una línea de asteriscos.
        /// No tiene parámetros.
        /// No tiene valor de retorno.
        /// </summary>
        static void Asteriscos() {
            Console.WriteLine("******************************");
            Console.WriteLine("******************************");
            Console.WriteLine("******************************");
        }

        static void Main(string[] args)
        {
            string sUsuario;
            int iRadio;

            do {
                Asteriscos();
                Console.WriteLine("Dime el radio: ");
                iRadio = int.Parse(Console.ReadLine());
                Console.WriteLine("Area = " + CalculaArea(iRadio));
                Console.WriteLine("Deseas ejecutarlo otra vez? si/no");
                sUsuario = Console.ReadLine();
            } while(sUsuario == "si");

            Console.WriteLine("Adios!...");
            Console.ReadLine();
        }
    }
}
```