PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN
IIC 1102 INTRODUCCIÓN A LA PROGRAMACIÓN
Profesor: Rodrigo Sandoval U.

Capítulo II – Introducción al Lenguaje C#

Material preparado por Rodrigo Sandoval U en Marzo-Agosto 2004
Basado en la documentación del curso IIC1102, año 2003, de Miguel Nussbaum y Marcos Sepúlveda et. al.

1 El Lenguaje C# y el Framework de .NET

C# (pronunciado “C Sharp”) es el nuevo lenguaje de propósito general orientado a objetos creado por Microsoft para su nueva plataforma .NET.

Microsoft.NET es el conjunto de nuevas tecnologías en las que Microsoft ha estado trabajando estos últimos años con el objetivo de mejorar tanto su sistema operativo como su arquitectura de desarrollo anterior, para obtener una plataforma con la que sea sencilla la construcción de software.

La plataforma .NET ofrece numerosos servicios a las aplicaciones que para ella se escriban, como son un recolección de basura, independencia de la plataforma, total integración entre lenguajes (por ejemplo, es posible escribir una clase en C# que derive de otra escrita en Visual Basic.NET que a su vez derive de otra escrita en Cobol)

Como se deduce del párrafo anterior, es posible programar la plataforma .NET en prácticamente cualquier lenguaje, pero Microsoft ha decidido sacar uno nuevo porque ha visto conveniente poder disponer de un lenguaje diseñado desde 0 con vistas a ser utilizado en .NET, un lenguaje que no cuente con elementos heredados de versiones anteriores e innecesarios en esta plataforma y que por tanto sea lo más sencillo posible para programarla aprovechando toda su potencia y versatilidad.

C# combina los mejores elementos de múltiples lenguajes de amplia difusión como C++, Java, Visual Basic o Delphi. De hecho, su creador Anders Heljsberg fue también el creador de muchos otros lenguajes y entornos como Turbo Pascal, Delphi o Visual J++. La idea principal detrás del lenguaje es combinar la potencia de lenguajes como C++ con la sencillez de lenguajes como Visual Basic, y que además la migración a este lenguaje por los porgramadores de C/C++/Java sea lo más inmediata posible.

Además de C#, Microsoft propociona Visual Studio.NET, la nueva versión de su entorno de desarrollo adaptada a la plataforma .NET y que ofrece una interfaz común para trabajar de manera cómoda y visual con cualquiera de los lenguajes de la plataforma .NET (por defecto, C++, C#, Visual Basic.NET y JScript.NET, aunque pueden añadirse nuevos lenguajes mediante los plugins que proporcionen sus fabricantes).

1.1 Microsoft .NET Framework 1.1

El Framework de .Net es una infraestructura sobre la que se reúne todo un conjunto de lenguajes y servicios que simplifican enormemente el desarrollo de aplicaciones. Mediante esta herramienta se ofrece un entorno de ejecución altamente distribuido, que permite crear aplicaciones robustas y escalables. Los principales componentes de este entorno son:

- Lenguajes de compilación
- Biblioteca de clases de .Net
- CLR (Common Language Runtime)

Actualmente, el Framework de .Net es una plataforma no incluida en los diferentes sistemas operativos distribuidos por Microsoft, por lo que es necesaria su instalación previa a la ejecución de programas creados mediante .Net. El Framework se puede descargar gratuitamente desde la web oficial de Microsoft (ver link de descarga en los recursos del final).

1.1.1 Lenguajes de Programación

.Net Framework soporta múltiples lenguajes de programación y aunque cada lenguaje tiene sus características propias, es posible desarrollar cualquier tipo de aplicación con cualquiera de estos lenguajes. Existen más de 30 lenguajes adaptados a .Net, desde los más conocidos como C# (C Sharp), Visual Basic o C++ hasta otros lenguajes menos conocidos en el mundo microsoft como son Perl o Cobol.

1.1.2 Common Language Runtime (CLR)

El CLR es el verdadero núcleo del Framework de .Net, ya que es el entorno de ejecución en el que se cargan las aplicaciones desarrolladas en los distintos lenguajes, ampliando el conjunto de servicios que ofrece el sistema operativo estándar Win32. La herramienta de desarrollo compila el código fuente de cualquiera de los lenguajes soportados por .Net en un mismo código, denominado código intermedio (MSIL, Microsoft Intermediate Language). Para generar dicho código el compilador se basa en el Common Language Specification (CLS) que determina las reglas necesarias para crear código MSIL compatible con el CLR.

1.1.3 Biblioteca de clases de .Net

Cuando se está programando una aplicación muchas veces se necesitan realizar acciones como manipulación de archivos, acceso a datos, conocer el estado del sistema, implementar seguridad, etc. El Framework organiza toda la funcionalidad del sistema operativo en un espacio de nombres jerárquico de forma que a la hora de programar resulta bastante sencillo encontrar lo que se necesita.

Para ello, el Framework posee un sistema de tipos universal, denominado Common Type System (CTS). Este sistema permite que el programador pueda interactuar los tipos que se incluyen en el propio Framework (biblioteca de clases de .Net) con los creados por él mismo (clases). De esta forma se aprovechan las ventajas propias de la programación orientada a objetos, como la herencia de clases predefinidas para crear nuevas clases, o el polimorfismo de clases para modificar o ampliar funcionalidades de clases ya existentes.

Este resumen fue extraído de: http://www.desarrolloweb.com/articulos/1328.php?manual=48

1.2 Estructura de un programa en C#

La declaración de un programa en C# se centra en la definición de clases que estarán involucradas en la lógica de la solución. Este programa puede contener cuantas clases se estime necesarias, y al menos una de ellas debe contener el algoritmo principal o Main().

En pocas palabras, la estructura general y simplificada de un programa en C# es la siguiente: se declaran las librerías cuyas clases se referencian en el programa, luego se declaran las clases requeridas con todos sus detalles, y finalmente, la clase principal (que en muchos casos es la más simple y de corta declaración), cuyo principal y a veces único contenido es el método Main(), también conocido como algoritmo principal. Este Main() es obligatorio en todo programa, y será llamado en forma implícita cuando la ejecución del programa dé inicio.

```
[declaración de namespaces o librerías de clases (externas) requeridas]


[declaración clase 1]
[declaración clase 2]
...
[declaración clase N o principal]
      [algoritmo principal o Main()]
```

En la declaración de un programa, todos los elementos definidos se etiquetan con nombres propios, que se conocen como identificadores. Estos identificadores son asignados arbitrariamente por el programador, siempre cumpliendo algunas reglas que se describen a continuación.

1.2.1 Ejemplo de un programa en C#

El programa más simple de todos: “Hola Mundo”, el cual sólo contiene una clase principal.

```
using System;  // Referencia al namespace de clases y métodos más usado

// Se declara una clase única y exclusivamente para alojar el Main().
class MainApp {

   // Función principal, Main()
   public static void Main() {

      // Escribe el texto a la consola.
      Console.WriteLine("Hola Mundo, estoy hecho en C#!");
      Console.ReadLine(); // Espera un ENTER para cerrar
   }
}
```

Como se ve en el trozo de código anterior, todo en C# pertenece a una clase, y por ende, obliga totalmente a tener un enfoque orientado a estos objetos.

1.3 Identificadores

Los identificadores son los **nombres con que se identifica a los distintos objetos dentro de un programa**, como ser:

- Clases.
- Instancias.
- Namespaces.
- Funciones o métodos.
- Variables, campos o propiedades.
- Constantes.
- Estructuras de datos.

En C#, los **identificadores** siguen las reglas para identificadores recomendados en el Anexo 15 del Unicode Standard, es decir, cada identificador puede contener:

- Letras (las del alfabeto mayúsculas y minúsculas, **menos la ñ, Ñ y las acentuadas**). No es posible emplear acentos o caracteres especiales del alfabeto español.
- Dígitos numéricos (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
- Símbolo de subrayado (_)
- Adicionalmente, en casos especiales, se puede utilizar el símbolo @ al comienzo de un identificador, cuando éste es una de las palabras claves. Sin ese símbolo, ningún identificador puede ser igual al conjunto de palabras reservadas del lenguaje C#.
- Nótese, que a diferencia de otros lenguajes, C# así como sus antecesores (C, Java, etc.) son “sensibles a las mayúsculas”, es decir, se considera “var1” y “VAR1”, e incluso “Var1”, son identificadores totalmente distintos.

Las palabras reservadas en C# son las siguientes.

abstract	**event**	**New**	**struct**
as	**explicit**	**Null**	**switch**
base	**extern**	**object**	**this**
bool	**false**	**operator**	**throw**
break	**finally**	**out**	**true**
byte	**fixed**	**override**	**try**
case	**float**	**params**	**typeof**
catch	**for**	**private**	**uint**
char	**foreach**	**protected**	**ulong**
checked	**goto**	**public**	**unchecked**
class	**if**	**readonly**	**unsafe**
const	**implicit**	**ref**	**ushort**
continue	**in**	**return**	**using**
decimal	**int**	**sbyte**	**virtual**
default	**interface**	**sealed**	**volatile**
delegate	**internal**	**short**	**void**
do	**is**	**sizeof**	**while**
double	**lock**	**stackalloc**	
else	**long**	**static**	
enum	**namespace**	**string**	

1.3.1 Algunas consideraciones al definir identificadores

En la práctica, **la letra de subrayado (_) se emplea para dar mayor legibilidad a nombres** compuestos por varias palabras. Además, se acostumbra emplear letras minúsculas para nombrar a las variables y las mayúsculas se usan al comienzo de una palabra en funciones declaradas con identificadores compuestos.

Se recomienda **elegir nombres que sean representativos del elemento (atributo o método)** que se defina o del valor que la variable guarde. Por ejemplo, una variable que guardará la edad de un empleado podría llamarse edad_empleado o *edad*, y no simplemente xyz, aunque éste es un identificador válido y el programa funcionará correctamente este nombre.

1.3.2 Ejemplos de identificadores

Los siguientes son todos identificadores válidos en C#.