http://www.informatizate.net

Dime como Programas y te diré Quien Eres

Helkyn R. Coello Costa

Ing. Informatico
AccountTECH Peru - Project Manager
MCSD.NET / MCDBA / MCT
Catedratico UPN
Trainer/ Expositor en diversas instituciones

informatizate(at)informatizate(dot)net
Agosto 23 del 2004.

Un aspecto muy importante para un programador es definir el "estilo" de programación que este utiliza. Algunos, los más principiantes, usan nombres de sus seres queridos para nombrar objetos y variables en el programa, otros, que no desea pensar mucho, usan nombres aleatorios para sus variables de código, y así podemos seguir con una interminable lista de "estilos" o "formas" de programación.

La pregunta clave que todos nos hacemos es: ¿Cual es el estilo adecuado?, ¿que terminología es la mas apropiada para mi?

A decir verdad, no existe "terminología" o "estilo" que sea mejor que otro. La valoración de dichas "terminologías" se basa no en lo que al programador le guste, sino primordialmente en el uso adecuado de un "terminología" específica. Esto es lo que denominamos "estándares de programación", que no es mas que el usar y seguir ciertas reglas de notación y nomenclatura durante la fase de implementación (codificación) de una aplicación.

Criterios de un buen estándar

Hay muchos estándares de programación que podemos usar. Debemos elegir aquel que se adecue más a nuestro estilo de programación. Si aun no se tiene uno, pues en este artículo veremos brevemente algunos de ellos. Un buen estándar de programación generalmente considerará los siguientes factores:

- Factor mnemotécnico: Para que el programador pueda recordar el nombre de una variable fácilmente
- Factor sugestivo: Para que otros programadores puedan leer y entender rápidamente nuestro código
- Consistencia: Tiene que ver con usar las mismas convenciones de nomenclatura en todo el programa y hacer que el texto del código sea "legible"

Ventajas del uso de estándares

Establecer un estándar de programación y nomenclatura puede tomar mucho tiempo. De allí la necesidad de "encontrar" o "elaborar" aquel que sea ajuste más a nosotros (según nuestra experiencia). Pero una vez establecido este estándar, los beneficios son muchos:

- Los nombres de variables serán mnemotécnicos con lo que se podrá saber el tipo de dato de cada variable con sólo ver el nombre de la variable
- Los nombres de variables serán sugestivos, de tal forma que se podrá saber el uso y finalidad de dicha variable o función fácilmente con solo ver el nombre de la variable
- La decisión de poner un nombre a una variable o función será mecánica y automática, puesto que seguirá las reglas definidas por nuestro estándar.
- Permite el uso de herramientas automáticas de verificación de nomenclaturas

¿Porque los estándares son usados muy poco?

Si los estándares tienes tantos beneficios, entonces la pregunta es ¿porque los programadores los usan muy pocas veces? La razón tiene que ver más con los seres humanos que con la tecnología:

- Trabajan en un proyecto que no ha adoptado ningún estándar
- No entiende o no pueden recordar el estándar
- No ven el beneficio
- Están muy apurados o cansados
- Prefieren creatividad y consistencia arbitraria
- Piensan que es divertido usar nombres "bonitos" en código
- Son "artistas del software" y no pueden estar regidos por convenciones

Cabe recalcar que el buen programador debe estar en la capacidad de adaptarse a cualquier Standard de programación que establezca el equipo de desarrollo a la que pertenece dicho programador. Esto es común cuando se trabajo en equipos de programadores de 2 o más personas.

¿Qué comprende un estándar?

Un estándar de programación no solo se busca definir la nomenclatura de las variables, objetos, métodos y funciones, sino que también tiene que ver con el orden y legibilidad del código escrito. Siguiendo esta idea, podemos definir 3 partes principales dentro de un estándar de programación:

- Convención de nomenclatura: Como nombrar variables, funciones, métodos, etc.
- Convenciones de legibilidad de código: Como identar el código, etc.
- Convenciones de documentación: Como establecer comentarios, archivos de ayuda, etc.

Estándares más comunes

A continuación daremos algunas pautas sobre los principales estándares de programación:

Notación húngara

Esta convención se basa en definir prefijos para cada tipo de datos y según el ámbito de loas variables. También es conocida como notación: REDDICK (por el nombre de su creador).

La idea de esta notación es la de dar mayor información al nombre de la variable, método o función definiendo en ella un prefijo que identifique su tipo de dato y ámbito. A continuación un ejemplo:

intEdad: Según la definición vemos que esta variable es de tipo INTEGER y que representa la edad de alguna persona
prStrNombre: En este caso la variable tiene el prefijo: "prInt", lo cual significa que es un parámetro por referencia (pr) de tipo STRING que representa un nombre

gStrConexion: En este caso se trata de una variable global (g) de tipo STRING que representa cierta información de conexión

Para una referencia completa de este estándar de programación consulte la siguiente dirección: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnvsgen/html/hunganotat.asp

Notación PascalCasing

Pascal-Casing es como la notación húngara pero sin prefijos. En este caso, los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula. A continuación un ejemplo:

DoSomething: Este nombre de método esta compuesto por 2 palabras, ambas iniciando con letra mayúscula.

Notación camelCasing

Camel-Casing es común en Java. Es parecido al Pascal-Casing con la excepción que la letra inicial del identificador no debe estar en mayúscula. A continuación un ejemplo:

doSomething: Este nombre de método esta compuesto por 2 palabras, la primera todo en minúsculas y la segunda iniciando con letra mayúscula

Crear tu propio estándar

También, como se dijo anteriormente, podemos establecer nuestros propios estándares de programación, los cuales pueden basarse en los ya existente, extenderlos o modificarlos. A continuación les presentare un ejemplo de un estándar de programación personalizado

	AMBITO		
	Local o a nivel de procedimiento	**Nivel de modulo o form**	**Global**
Controles			
Classes	cls< ClassName>	cls< ClassName>	cls< LassName>
Módulos	mod< ModuleName>	mod< ModuleName>	mod< ModuleName>
Formularios	frm< FormName>	lFrm< FormName>	gFrm< FormName>
Combobox	cbo< ComboName>	lCbo< ComboName>	gCbo< ComboName>
Command	cmd< commandName>	lCmd< CommandName>	gCmd< CommandName>
Datagrid	grd< GridName>	lGrd< GridName>	gGrd< GridName>
ListBox	Lst< ListboxName>	lLst< ListboxName>	gLst< ListboxName>
Option buttons	opt< OptionName>	lOpt< OptionName	gOpt< OptionName>
checkBoxes	chk< CheckName>	lChk< CheckName>	gChk< CheckName>
Textboxes	txt< TextName>	lTxt< TextName>	gTxt< TextName>
Tipos primitivos			
Integer	int< Nombre >	lInt< Nombre >	gInt< Nombre >
Long	lng< Nombre >	lLng< Nombre >	GLng< Nombre >
Bolean	bln< Nombre >	lBln< Nombre >	gBln< Nombre >

Object	obj< Nombre >	lObj< Nombre >	Gob..< Nombre >
String	str< Nombre >	lStr< Nombre >	gStr< Nombre >
Double	dbl< Nombre >	lDbl< Nombre >	gDbl< Nombre >
Constantes	C_< NOMBRE>	LC_< NOMBRE >	GC_< NOMBRE >

Legibilidad del código

Debemos poner énfasis especial en la legibilidad del código. Cualquier que sea el proyecto, los miembros del proyecto pasarán mucho tiempo escribiendo, leyendo y revisando el código fuente. Se calcula que un programador pasa la mitad del tiempo tratando de entender "que es lo que tal o cual bloque de código hace". Podemos hacer este trabajo mucho más fácil siguiendo las convenciones de nomenclatura de los estándares y a la vez haciendo nuestro código legible y bien documentado.

A continuación les daré algunas pautas de cómo puede dar mayor legibilidad a su código fuente:

```
1)
Public Sub Procedure(parameters list)
On Error GoTo errCatch
If bSomeCondition = False Then
DoSomething
Else
For each object in colObjects
If bSomeOtherCondition = false then
DoOtherThings
End If
Next
End If
Exit Sub
errCatch:
MsgBox Err.Description, .......
End Sub
```

```
2)

Public Sub Procedure(parameters list)
   On Error GoTo errCatch
   If bSomeCondition = False Then
       DoSomething
   Else
       For each object in colObjects
        If bSomeOtherCondition = false then
                 DoOtherThings
        End If
       Next
   End If
   Exit Sub
   errCatch:
        MsgBox Err.Description, .......
End Sub
```

Claramente podemos notar que el Segundo bloque de código es mucho más entendible que el primero.

Documentación del código

Esto tiene que ver con los comentarios explicatorios y aclaratorios que establecemos en nuestro código para futura referencia. Muchas veces podemos olvidar la finalidad de un proceso con complicada algoritmia. Los comentarios nos ayudan a recordar los puntos claves de cada parte de nuestro código (sobre todo cuando ha pasado un tiempo largo de haberlo codificado). Además sirve como que los demás miembros del equipo de desarrollo puedan entender también dichos bloques de código.

Podemos establecer una convención de documentación que puede darse de muchas formas. A continuación una forma sencilla y practica de comentar nuestro código.

```
'Created by: Helkyn Coello
'Date of creation: 18/08/2004
'Function description: Descripción de lo que hace la función
'Parameters description: descripción de la finalidad de cada parámetro
Function Process1 (par1 as integer, par2 as integer) as Integer
    code goes here
    ...
    ...
End Function
```

Qué herramientas usar

Algo también importante a la hora de usar estándares es usar una herramienta de verificación de estos estándares. Para aquellos programadores que no estén muy acostumbrados a uso de estas convenciones, estas herramientas pueden resultar muy útiles, puesto que nos harán aprender "a la mala"

Por lo general estas herramientas analizar nuestro archivos de código fuente y verifica que partes de el no cumple con nuestras convenciones establecidas.

Lógicamente estas herramientas incluyen motores de verificación de distintos tipos de estándares y esto es configurable según nuestras necesidades.

Incluso algunos de estas herramientas nos permiten definir nuestras propias reglas y convenciones, como es el caso de FxCop, un software libre de la comunidad gotDotNet, que ha sido incluido como parte del Visual Studio.NET 2005. FxCop tiene una interfaz muy amigable en la cual podemos elegir que reglas queremos sean verificadas y además podemos crear nuevas reglas con el Introspection Engine de FxCop.

Otra herramienta interesante es "Standard Master" el cual se adhiere al editor de código de VS.NET y permite verificar el código mientras este se escribe.

Conclusiones

Nombres crípticos y de libre imaginación eran permitidos hace décadas, dado el tamaño limitado de los programas y las herramientas. Pero hoy en día, el programador que persiste en usar variables con nombre triviales o sin un orden establecido traerá muchas dificultades en un equipo de desarrollo. Se puede ahorrar algunos segundos al no usar ningún estándar, pero se perderán mucho tiempo después.

La esencia de los estándares de programación en mantener la consistencia del código siguiente una determinada convención de nombres. Esto le ayudará a escribir, mantener y re-usar código en una forma mas eficaz y eficiente.

Puede elegir el estándar que más le guste, e incluso puede crear y personalizar su propio estándar. Elija aquel que establezca nombres claros, descriptivos y significativos, y que sean fáciles de recordar. Sea que hablemos de nombre de un negocio, producto, programa o de un bebe recién nacido, un buen nombre siempre es esencial.
Para lograr "acostumbrarse" mejor a cualquiera fuere el estándar que eligió, puede además usar herramientas de verificación de estas convenciones que puede detectar automáticamente errores de nomenclatura.

De esta manera, conforme usted programe y use esta nomenclatura, podremos decir: "Dime como programas y te diré quien eres"