DocuCode-AI
Documento de Estándares de Programación
Versión 1.0

Historia de Revisión

Historial de revisiones				
Ítem	**Fecha**	**Versión**	**Descripción**	**Equipo**
1	03/07/2025	1.0	Versión 1	Lupaca Mamani, Ronal Daniel

Estándares de Programación

1. OBJETIVO

Reglamentar la forma en que se implementará el código fuente del proyecto, pasando, por las variables, controles, clases, métodos, ficheros, archivos y todo aquello que esté implicado en el código,

Mejorar y uniformizar a través de las reglas que se proponen, el estilo de programación que tiene cada programador.

- 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.

Por tanto, se seguirán dichos patrones para un entendimiento legible del código y para facilitar el mantenimiento del mismo.

2. DECLARACIÓN DE VARIABLES

Se propone que la declaración de las variables, se ajusten al motivo para la que se requieran. El mnemotécnico definido se establece tomando en consideración principalmente lo siguiente:

- La longitud debe ser lo más recomendable posible. No debe ser tan grande de tal forma que el programador tenga la facilidad de manejo sobre la variable y ni tan corta que no pueda describirse claramente. Para el caso establecemos una longitud máxima de variable de 16 caracteres.
- Alcance de la variable

A medida que aumenta el tamaño del proyecto, también aumenta la utilidad de reconocer rápidamente el alcance de las variables. Esto se consigue al escribir un prefijo de alcance de una letra delante del tipo de prefijo propio, sin aumentar demasiado la longitud del nombre de las variables.

Alcance	Prefijo	Ejemplo
Global	G	gUsuario
Nivel de la clase	*M*	mblnProgresoDelCálculo
Local del procedimiento / método	Ninguno	dblVelocidad
Público	P	pCantidadUsuario
Privado	Pr	prCantidadVenta

- El tipo de dato al que pertenece la variable.

Por lo tanto la estructura de la variable es como sigue:

Estructura	Descripción de la Variable
LONGITUD. MAX.	1 16
FORMATO	*Minúscula la primera parte y luego la segunda con Mayúsculas*
EJEMPLO	nomUsuario

Siendo el nombre que identifica a la variable: **nomUsurio**

2.1 Descripción de la Variable.

Nombre que se le asignará a la variable para que se le identifique y deberá de estar asociada al motivo para la cual se le declara.

Ejemplos: idUsuario,nomUsuario,apeUsuario,teleUsuario,etc.

2.2 Variables de Tipo Arreglo

En el caso de las definiciones de arreglos de elementos se declarará la variable con el prefijo de “lista”,

el cual nos dará entender que se trata de una variable del tipo arreglo la cual contendrá de cero a más datos, según el tamaño declarado.

Ejemplos: arrayUsuarios,arrayEdad,arrayAreaSalud,etc.

Definición de Controles

Para poder determinar el nombre de un control dentro de cualquier aplicación de tipo visual, se procede a identificar el tipo al cual pertenece y la función que cumple dentro de la aplicación.

2.3 Tipo de datos

Tipo de variable	Mnemónico	Descripción
Byte	by	Entero de 8 bits sin signo.
Integer	in	Entero de 32 bits con signo.
Char	ch	Un carácter UNICODE de 16 bits
String	st	Cadena de caracteres
Date	dt	Formato de fecha/hora
Boolean	bl	Valor lógico: verdadero y falso
Float	fl	Coma flotantes, 11-12 dígitos significativos.
Double	db	Coma flotante, 64 bits (15-16 dígitos significativos)
Object	ob	Objeto genérico

2.4 Prefijo para el Control

El prefijo del control será determinado mediante tres caracteres que estarán conformados por las consonantes más representativas del control, es así, por ejemplo; el control Button, estará asociado al prefijo btn.

2.5 Nombre descriptivo del Control

Formado por la descripción de la función que lleva a cabo el control, esta debe ser descrita en forma específica y clara.

Tipo de control	Prefijo	Ejemplo
Label	lbl	lblNombre
Text	txt	txtApellido
Button	btn	btnLogin
RadioButton	rdo	rdoSeleccion
Image	img	imgUsuario
Password	pass	passUsuario

2.6 Declaración de variables, atributos y objetos

1. Se debe declarar una variable por línea.

Título	Descripción
Sintaxis	[TipoVariable] [Nombre de la Variable]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 20 caracteres. El nombre de la variable puede incluir más de un sustantivo los cuales se escribirán juntos. Si se tuvieran variables que puedan tomar nombres iguales, se le agregará un número asociado (si está dentro de un mismo método será correlativo).
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: ● Letra Ñ o ñ. ● Caracteres especiales ¡, ^, #, $, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. ● Caracteres tildados: á, é, í, ó, ú.
Ejemplo	$txtUsuario Indica una variable o atributo que guardará un tipo string.s

2.7 Declaración de clases

Título	Descripción
Sintaxis	Class [Nombre de Clase]
Descripción	El nombre de las clases tendrá una longitud máxima de 30 caracteres y las primeras letras de todas las palabras estarán en mayúsculas. Tipo se refiere a si la clase será: Private, Public o Protected.
Observaciones	En la declaración de clases no se deberá utilizar caracteres como: ● Letra Ñ o ñ. ● Caracteres especiales ¡, ^, #, $, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. ● Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Class Usuario Indica una clase Usuario

2.8 Declaración de métodos

Título	Descripción
Sintaxis	nombreProcedim(ListaParámetros)
Descripción	El nombre del método constará hasta de 25 caracteres. La primera letra de la primera palabra del nombre será escrita en minúscula y las siguientes palabras empezarán

	con letra mayúscula.
Observaciones	En la declaración de métodos no se deberá utilizar caracteres como: ● Letra Ñ o ñ. ● Caracteres especiales ¡, ^, #, $, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,], _. ● Caracteres tildados: á, é, í, ó, ú.
Ejemplo	citaMédica($medico,$paciente) Indica un método citaMédica que recibe dos variables uno es el medico y otro paciente.

2.9 Declaración de funciones

Título	Descripción
Sintaxis	[Tipo] nombreFuncion(ListaParámetros)
Descripción	El nombre del objeto constará hasta de 25 caracteres, no es necesario colocar un nombre que indique la clase a la cual pertenece. La primera letra de la primera palabra del nombre será escrita en mayúsculas El tipo de dato de retorno se coloca al final y será obligatorio colocarlo.
Observaciones	En la declaración de objetos no se deberá utilizar caracteres como: ● Letra Ñ o ñ. ● Caracteres especiales ¡, ^, #, $, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,], _. ● Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Public function agregar($nomUsario,$ApeUsuario) Indica un método agregar usuario donde en otro archivo php llamo al método y escribo las variables que quiero guardar.

2.10 Control de versiones de código fuente

Cada modificación realizada será guardada de la forma:

Título	Descripción
Formato	[NOMBRE DOCUMENTO][_][FECHA][_][HORA] donde y la fecha estará en formato yyyymmdd y la hora en formato HHMM.
Descripción	Se generarán archivos con las siguientes extensiones:.zip o .rar. Por ejemplo: WSTENNIS_20070421_2056.zip

3. Clases.

El nombre de las clases debe ser autodescriptivo de manera que no se requiera, en lo posible, entrar

al código de la función para saber qué es lo que realiza.

El estándar para nombres de clases es usar iniciar con las siglas **cls,** la cual debe estar escrita en minúscula seguido del nombre que identifica la clase, la primera letra del nombre debe iniciar con mayúscula

- Ejemplos: Class Cuenta, Class Médico, Class Cita

Nota:
- Se hará uso de los caracteres: Espacio en blanco " ".

4. Métodos, Procedimientos y Funciones definidos por el Usuario.

El nombre de las funciones y procedimientos debe ser autodescriptivo de manera que no se requiera, en lo posible, entrar al código de la función para saber qué es lo que realiza.
verbo-Sustantivo

El estándar para nombres de procedimiento es usar un Verbo que describa la acción realizada seguida por un sustantivo (objeto sobre el cual actúa). Se recomienda:

- Usar un nombre que represente una acción y un objeto. El nombre del procedimiento debe indicar qué hace el procedimiento a... o qué hace el procedimiento con....

- El verbo debe estar en infinitivo.

- Ser consistente en el orden de las palabras. Si se va a usar ***verboNombre***, siempre usar ***verboNombre***.

- Ser consistente en los verbos y sustantivos usados. Por ejemplo, si tiene un procedimiento ***asignarNombre***, en vez de ***colocarNombre***.

- Para la acción **modificar cuentas del cliente** se define:

modificar Cuenta
Verbo: modificar
Sustantivo: Cuenta

Nota:
- No se hará uso de los caracteres: Espacio en blanco " ", Carácter de subrayado "_".
- La nomenclatura de argumentos o parámetros pasados a los procedimientos/funciones así como para valores devueltos por funciones sigue las mismas convenciones que la nomenclatura para variables.

5. Beneficios

- La documentación hace más legible un programa.
- Al documentar bien un programa desde un principio se evita que para cada modificación deba estudiarse profundamente el funcionamiento del programa, redescubriendo todo lo no documentado, con la ventaja adicional de que generalmente quién modifica el programa no es siempre quién lo escribió.
- Facilita la reutilización de módulos y rutinas desde cualquier otro programa o el mismo.

- Ayuda a determinar cuándo debe ser reescrito un código. Si existen problemas para explicar el código con un comentario, probablemente el código esté mal escrito.

6. Conclusiones

- Una buena programación e implementación legible, sólo se logra usando y llevando de la mano un buen estándar o patrón de programación.
- Es muy importante que el programador tenga un conocimiento previo del estándar o en su defecto que lea el documento para prever diferencias.
- Al documentar se obtienen dos cosas fundamentales, un documento legible y segundo una buena base para los futuros desarrollos de mantenimiento del código.