	METODOLOGIA		F002-P002-GTI Versión 1 04.06.2013
	DOCUMENTACION CODIGO FUENTE		
	Código:	Versión:	Fecha:


Metodología documentación Código Fuente

Versión:

Fecha:


Realizado por:

Ing. Néstor José Gómez Lozano

	METODOLOGIA		F002-P002-GTI Versión 1 04.06.2013
	DOCUMENTACION CODIGO FUENTE		
	Código:	Versión:	Fecha:

Contenido

1	Objetivo.....	3
2	Nomenclatura.....	3
3	Comentarios al código fuente.....	4
3.1	Reglas	4
3.1.1	Preparare al lector sobre lo que sigue.....	5
3.1.2	Que cada comentario cuente.....	5
3.1.3	Evite el uso de las abreviaturas o siglas.....	5
3.1.4	Diferencie Comentarios principales vs secundarios	5
3.1.5	Documente las excepciones.....	5
3.1.6	Como Comentar el código complejo o confuso.....	6
3.1.7	Declaraciones de Datos	6
3.1.8	Datos globales	8
3.1.9	Estructuras de Control.....	8
3.1.10	Rutinas	9
3.1.11	Documentación de Archivos	12

	METODOLOGIA		F002-P002-GTI Versión 1 04.06.2013
	DOCUMENTACION CODIGO FUENTE		
	Código:	Versión:	Fecha:

1 Objetivo

El objetivo de esta metodología es establecer los lineamientos con base en las mejoras prácticas para documentar el código fuente y establecer una política uniforme para ejecutar esta actividad en todas las aplicaciones que se desarrollen en el SENA.

El código fuente debe estar documentado en forma sencilla, clara y eficiente de tal forma que facilite al programador que lo construyo o a cualquier otra persona que necesite hacer mantenimiento, ejecutar su labor en forma idónea, la escritura del código debe estar orientado a facilitar su entendimiento haciéndolo autoexplicativo.


2 Nomenclatura

Las especificaciones generales que se deben seguir en el nombramiento de los objetos archivos, variables, procedimientos y funciones deben regirse por los siguientes parámetros:

- Los nombres deben ser escritos en mayúscula.
- Los nombres deben ser siempre diferentes entre sí.
- Cuando se utilice más de una palabra estas deben estar separadas por “_”.
- Utilice nombres corto, fácil de leer, descriptivo y en plural.
- Los nombres deben corresponder al alfabeto español exceptuando vocales con acento, eñes, diéresis y todo tipo de carácter especial (#, /, :, %, +, -, ni espacios, etc).

La indentación del código deberá estar siempre entre 2 0 4 espacios.

Las líneas de código no deben ser superiores a 80 caracteres.

	METODOLOGIA		F002-P002-GTI Versión 1 04.06.2013
	DOCUMENTACION CODIGO FUENTE		
	Código:	Versión:	Fecha:

3 Comentarios al código fuente

Los comentarios erróneos al código confunden al lector, incluso si el código se ha escrito en forma estructurada. Se recomienda hacer los comentarios en español, en lenguaje claro.

Evite utilizar el comentario repetitivo y elimine los marcadores de código en la versión definitiva del mismo, a continuación se describe cada uno de los tipos de comentarios.

Repetitivos del Código: Replantan lo que el código dice en lenguaje común, dan al lector más líneas para leer sin agregar información adicional de valor, no corresponde a una codificación eficiente.

Explicativos del Código: Estos comentarios son utilizados para explicar piezas de código complejo o confuso. En tales situaciones son útiles, pero únicamente debido a que el código no es fácil de entender. Si el código es tan complicado que necesita ser explicado en detalle, es recomendable mejorar el código en lugar de añadir los comentarios.

Marcadores en el Código: Son los que se colocan a la izquierda en el código. Es una nota que el desarrollador deja para sí mismo con el objetivo de indicar el avance el trabajo que aún tiene pendientes. Puede ser una instrucción sintácticamente incorrecta para que el compilador la detecte, o bien un conjunto específico de caracteres como un comentario que pueda ser localizado fácilmente sin interferir con la compilación.


Resumen del Código: Es un comentario de una o dos líneas que resumen un pequeño grupo de sentencias. Son más valiosos que los comentarios repetitivos debido a que pueden leerse más rápidamente que al código. Son particularmente útiles cuando alguien diferente al autor trata requiere modificar el código.

Descriptivos del Propósito del Código: Son los comentarios que intentan explicar la intención de una sección de código, son utilizados más a nivel del problema que a nivel de la solución. Por ejemplo:

/* Capture la información de la factura actual */

3.1 Reglas

La regla más importante es introducir los comentarios a medida que se va codificando, los comentarios en exceso o la existencia mínima de ellos producen confusión y dificultan la lectura del código fuente, por ello lo importante es lograr un equilibrio, el término medio para no errar por exceso o por defecto.

	METODOLOGIA		F002-P002-GTI Versión 1 04.06.2013
	DOCUMENTACION CODIGO FUENTE		
	Código:	Versión:	Fecha:

Debe seguir las siguientes reglas al construir los comentarios:

3.1.1 Preparare al lector sobre lo que sigue

Una excelente práctica es construir comentarios que explican qué esperar, así el lector con revisar únicamente los comentarios obtendrá una idea de lo que el código hace y encuentra en forma ágil un bloque particular de código.

3.1.2 Que cada comentario cuente

El exceso de comentarios enreda el código que se intenta clarificar. La mejor práctica es no escribir demasiados comentarios y utilizar este tiempo en construir código autoexplicativo.

3.1.3 Evite el uso de las abreviaturas o siglas


Los comentarios no deben ser ambiguos e ilegibles, por ello no se deben incluir abreviaturas y siglas, puede ocasionar diferentes interpretaciones, las abreviaturas van en detrimento de la claridad que se debe plasmar en el código mediante los comentarios.

3.1.4 Diferencie Comentarios principales vs secundarios

Cuando construya un comentario que es parte de otro más amplio es necesario que idente los comentarios secundarios para denotar subordinación a un comentario principal precedente.

3.1.5 Documente las excepciones

Si hay algo que es imposible deducir a partir de la lectura del código fuente, es recomendable aclararlo con un comentario, por ejemplo el uso de un truco para mejorar el desempeño o cualquier truco que evite un error o una funcionalidad no documentada en un lenguaje o entorno en particular.

	METODOLOGIA		F002-P002-GTI Versión 1 04.06.2013
	DOCUMENTACION CODIGO FUENTE		
	Código:	Versión:	Fecha:

También debe escribir un comentario cuando una excepción va en contravía del buen estilo de programación, es importante escribir el comentario y explicar el por qué.

3.1.6 Como Comentar el código complejo o confuso

Se recomienda reescribir el código complejo o confuso en lugar de escribir demasiadas líneas de comentario, aquí aplica la regla de Kernighan y Pike 1, “No documentes el mal código - rescríbelo.”

El código complejo o confuso es difícil de comprender y de mantener, lo mejor es buscar una forma de rescribirlo para disminuir su complejidad.

3.1.7 Declaraciones de Datos


Los comentarios para las declaraciones de variables describen aspectos de la variable cuyo nombre mismo no puede describir. Es importante documentar los datos cuidadosamente puesto que esa documentación es incluso más importante que las notas sobre los procesos en los cuales se utilizan los datos, se recomienda seguir los siguientes consejos:

- *Unidades de medida de datos numéricos*

Cada variable o campo que vaya a contener cantidades numéricas que representan unidades de medida deben tener documentada la unidad de medida que representa. Así, si una variable representará longitud debe documentarse si representará centímetros, pulgadas, metros, kilómetros, etc. Si representará peso, entonces su comentario debe especificar si serán gramos, onzas, libras, kilos, toneladas, etc. Si son coordenadas, si indicarán latitud, longitud o altitud, y si está en grados o radianes; si representarán una coordenada X, Y, X con su origen en el centro de la tierra, y cosas así.

¹ The Elements of Programming Style

http://en.wikipedia.org/wiki/The_Elements_of_Programming_Style_%28book%29

	METODOLOGIA		F002-P002-GTI Versión 1 04.06.2013
	DOCUMENTACION CODIGO FUENTE		
	Código:	Versión:	Fecha:

Ejemplo:

```
int XCursor; /* Pos. horizontal del cursor;
             desde 1 hasta MaxCols */
int YCursor; /* Pos. vertical del cursor;
             desde 1 hasta MaxFilas */
```

- *Rango de los valores numéricos permisibles*

Si una variable debe estar dentro de un rango de valores, se debe documentar este rango en la declaración de la variable, por ejemplo, si una variable representa calificaciones, se debe comentar cual es el rango para este valor (0 a 5, 0 a 10), este comentario no obvia incluir código que valide la entrada de datos para esa variable, por el contrario orienta esta exigencia.

Ejemplo:

```
long int longitudAntena; /* longitud de la antena en metros;
                        debe ser >=2 */
```

- *Rango de los valores numéricos permisibles*


Es recomendable utilizar comentarios para indicar que representa cada posible valor de una variable, por ejemplo si una variable representa tipos de corrientes eléctricas, es necesario construir un comentario que explique que 1 representa corriente alterna, 2 representa corriente directa y que 3 representa corriente indefinida.

Ejemplo:

```
int atributoDelCaracter; /* 0=Plano; 1=Cursiva;
                        2=Negrita; 3=Negrita Cursiva */
```

- *Limitaciones de los datos de entrada*

Los datos de entrada pueden venir de distintas fuentes: de un archivo, digitados, como un parámetro desde o hacia una rutina, etc., aplican tanto a los parámetros de entrada para las rutinas como para cualquier otro tipo de datos. Es importante explicar los datos esperados y

	METODOLOGIA		F002-P002-GTI Versión 1 04.06.2013
	DOCUMENTACION CODIGO FUENTE		
	Código:	Versión:	Fecha:

los datos no esperados, en un comentario, es una forma de asegurar que una rutina reciba los datos que corresponden. Seguir esta recomendación hace el código además de autoexplicativo, autoverificable.

- *Actualice los comentarios relacionados a las variables*

Si se tienen comentarios específicos para una variable, es importante actualizarlos cada vez que la variable es actualizada, especialmente en las modificaciones al código y cuando la variable, por alguna razón, cambie de nombre durante la programación, esto garantiza consistencia en las modificaciones.

3.1.8 Datos globales


Cuando se utiliza una variable o una estructura de datos global, se recomienda comentarla en donde es declarada. El comentario deben explicar el propósito de ese elemento y por qué se define como global, igualmente cada que se utiliza se recomendable documentar que variable o estructura de datos es global. El usar una convención de nombres (por ejemplo la letra g) ayuda a distinguir las variables y estructuras de datos globales de las que no lo son.

3.1.9 Estructuras de Control

Se recomienda documentar las sentencias condicionales o de selección (if y case) y los ciclos al inicio y al fin de cada una de ellas:

La estructura de control por regla general necesita explicación y la mejor práctica es incluir el comentario al inicio, en el caso de una sentencia de selección, se recomienda incluir la razón para la decisión y un resumen del resultado, si es un ciclo, puede indicar su propósito y aspectos importantes de la ejecución.

Igualmente un comentario al final del bloque de sentencias de una estructura de control resalta la construcción lógica del programa, es particularmente útil en ciclos largos o en el anidamiento de estructuras de control del mismo tipo, porque clarifica el anidamiento.

	METODOLOGIA		F002-P002-GTI Versión 1 04.06.2013
	DOCUMENTACION CODIGO FUENTE		
	Código:	Versión:	Fecha:

Ejemplo:

```
for (indiceArray=0; indiceArray < contadorArray; indiceArray++)
{
    while (indiceRegistro < conteoRegistros)
    {
        if (BuscarRegistro(indiceRegistro))
        {
            . . .
        } /* if */
    } /* while */
} /* for */
```

El anterior ejemplo corresponde a una estructura corta y el comentario refuerza la explicación de la estructura, por razones de espacio no se incluye el ejemplo largo pero funciona en igual forma, sin embargo no sobra la recomendación de tratar de reducir la estructura larga, construyendo código menos extenso, en lo posible.

3.1.10 Rutinas

Los comentarios deben escribirse en la(s) línea(s) que precede al código que describen, así se facilita su lectura, entendimiento y actualización durante la fase de mantenimiento, conservando su utilidad.


Una buena práctica es crear una plantilla que incluya las piezas de información que se sugieren en los siguientes ítems:

- *Rutina con una o dos oraciones en la línea anterior a ella*

Un comentario descriptivo breve permitirá conocer rápidamente el propósito de la rutina. Es buena práctica describir cada rutina en máximo dos líneas. Si se dificulta crear una descripción corta para la rutina es posible que el diseño del programa sea susceptible de mejorar.

Las rutinas cortas y de fácil entendimiento pueden o no contar con un comentario breve, pero de acuerdo a su importancia y objetivo es posible incluirlo.

- *Parámetros de entrada y salida*

	METODOLOGIA		F002-P002-GTI Versión 1 04.06.2013
	DOCUMENTACION CODIGO FUENTE		
	Código:	Versión:	Fecha:

La mejor práctica para documentar variables de entrada y salida de una rutina es escribir el comentario bajo la línea de encabezado de la rutina misma.

Ejemplo:

```
void InsertarEnOrden(int datoAInsertar, int *ArregloDeDatos[])
/*
datoAInsertar: Número a ingresar en la lista ordenada
ArregloDeDatos: Array de enteros que contiene la lista
de datos ordenados
*/
{
    ...
} /* InsertarEnOrden */
```

Aunque esta es una excepción a la regla de no usar comentarios de fin de línea, es una buena práctica.

Cuando los nombres de los parámetros son adecuadamente seleccionados y hacen ver a los comentarios como redundantes, si se mantienen estos pueden complementar la información necesaria para comprender su propósito.

- *Distinguir los parámetros de entrada de los de salida*

Es útil saber cuáles datos se utilizan como entrada para la rutina y cuáles para generar los resultados.


Ejemplo:

```
void InsertarEnOrden(int datoAInsertar, int *ArregloDeDatos[])
/*
datoAInsertar: Entrada - Número a ingresar en la lista ordenada
ArregloDeDatos: Salida - Array de enteros que contiene la lista de datos ordenados
*/
{
    ...
} /* InsertarEnOrden */
```

Es una buena práctica distinguir los parámetros de entrada de los de salida en las rutinas.

- *Suposiciones*

Cuando se hace suposiciones sobre el estado de los argumentos que se reciben, valores legales e ilegales, estructuras que debe conservar un orden, variables y estructuras que

	METODOLOGIA		F002-P002-GTI Versión 1 04.06.2013
	DOCUMENTACION CODIGO FUENTE		
	Código:	Versión:	Fecha:

contienen sólo datos válidos, entre otras, es recomendable incluir el comentario al inicio de la rutina o en la declaración de los parámetros, es muy importante escribir antes de que olvide. También se debe documentar las variables y estructuras de datos globales que se reciben u utilizan al interior de una rutina, para facilitar su diferenciación del resto de parámetros, porque cualquier cambio en los valores que ellas almacenan afecta directamente a todo el programa.

- *Limitaciones de la rutina*


Si una rutina devuelve un resultado numérico, se debe incluir un comentario indicando su precisión. Si los cálculos son indefinidos bajo algunas condiciones, se debe explicar en un comentario. Si la rutina para presenta un determinado comportamiento o funciona con valores por defecto cuando se da un excepción, se debe documenta este particular comportamiento. Si la rutina funciona sólo para estructuras de datos de cierto tamaño, se debe explicar en un comentario. Si se tiene claro que modificaciones harán que la rutina deje de funcionar, se deben aclarar en un comentario, y así cualquier limitación que la rutina pudiese tener.

- *Efectos globales de las modificaciones que realiza la rutina*

Si dentro de la rutina se modifica una variable o estructura de datos global, debes describir de forma explícita esta modificación, pero cuando la documentación de las variables y estructuras de datos globales se torna demasiado compleja, es recomendable rescribir el código para reducir o evitar el uso de tales elementos, en lo posible.

- *Fuente de los algoritmos*

Si has empleado un algoritmo tomado de un libro, un texto web o una revista, es recomendable documentar el código escribiendo la fuente con el número de página de donde se tomó o su dirección URL exacta. Si el algoritmo es un trabajo personal, indica al lector en dónde puede encontrar las notas de referencia, en particular los algoritmos o ideas sobre las que se hizo la construcción.

	METODOLOGIA		F002-P002-GTI Versión 1 04.06.2013
	DOCUMENTACION CODIGO FUENTE		
	Código:	Versión:	Fecha:

3.1.11 Documentación de Archivos

En los archivos se recomienda el uso de un comentario de bloque que describa el contenido del archivo:

- *Describir el propósito y contenido de cada archivo*

En el encabezado del archivo se debe describir las clases o rutinas contenidas en el archivo. Si por el contrario contiene una clase específica es más fácil construir el comentario, pero si por el contrario todas las rutinas para un programa están en un único archivo, este contiene el programa completo y debe explicar el por qué se combinan en un único archivo.

Esta explicación es de suma importancia para el mantenimiento posterior del programa, sea o no una persona diferente a quien lo construyo.

- *Datos del autor*

Es importante sobre todo en los proyectos grandes, porque seguramente se tiene más de un programador y en algunos casos es necesario hacer consultas a quien construyó el programa, por esto es importante incluir nombre completo, dirección de correo, número telefónico en el encabezado del archivo.


- *Copyright en el comentario de bloque*

Cuando se necesita incluir los derechos de autor de la entidad o del programador, es importante hacerlo en la primera línea del comentario.

- *Nombre relacionado con su contenido*

Es un detalle muy importante que facilita la lectura y entendimiento del código fuente contenido en él.

- *Indexación de la Documentación de Programas*

	METODOLOGIA		F002-P002-GTI Versión 1 04.06.2013
	DOCUMENTACION CODIGO FUENTE		
	Código:	Versión:	Fecha:

La construcción de un índice permite dividir el código en secciones y generar un listado que permite a los programadores tener una visión previa a la revisión del código.

Esta organización del código permite reducir el tiempo de mantenimiento del código por cuanto permite a los programadores comprender y recordar más fácilmente los distintos componentes de un programa y excluir la revisión de código que no pertenece a un proyecto en particular.