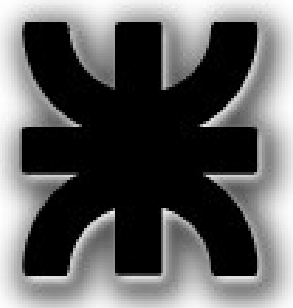


Herramientas de Documentación

Doxygen

Alejandro Furfaro

Marzo 2011

Temario

¿Que es doxygen?

Doxygen es una herramienta para convertir los comentarios de un programa escrito en C, C++, VHDL, PHP, C#, y Java (entre otros lenguajes), en documentación suficientemente prolija y presentable como para poser publicarse. Maneja la salida de documentación en formatos: html, LaTeX, pdf, rtf, entre otros. Y lo mas interesante es que la documentación se genera de manera automática, y a partir de los archivos fuente de modo que siempre será consistente con éstos.

Ventajas de usar Doxygen

1. Al escribir la documentación en el mismo archivo fuente en forma de comentarios, nos releva de mantener otra documentación separada de los fuentes.
2. La documentación se genera en forma automática agregando además diagramas de interacción entre los diferentes módulos
3. Permite documentar variables, estructuras de datos además de las funciones.
4. Es una herramienta de documentación automática. Hay otras pero su uso genera el hábito metodológico de trabajo.

Instalación

Hace falta instalar los siguientes paquetes

```
sudo apt-get install doxygen
sudo apt-get install graphviz
```

Opcionalmente (aunque no estaría de mas...)

```
sudo apt-get install kate
```

Instalación

Hace falta instalar los siguientes paquetes

```
sudo apt-get install doxygen
sudo apt-get install graphviz
```

Opcionalmente (aunque no estaría de mas...)

```
sudo apt-get install kate
```

¿Que mas hace falta para comenzar?

1. Escribir comentarios claros, y suficientes para describir cada función, antes de comenzar a “codearla”.
2. Una vez dentro de ella, insertar comentarios de modo de describir la operación (al menos los “big steps”)
3. Observar buenas prácticas de programación: Identar código, Usar convenciones para nombres de variables y funciones, Escribir código de manera “legible”.
4. Entender al comentario como parte escencial de la aplicación: Cada agregado o modificación debe ser documentado describiendo sus características, fecha y autor.

¿Que mas hace falta para comenzar?

1. Escribir comentarios claros, y suficientes para describir cada función, antes de comenzar a “codearla”.
2. Una vez dentro de ella, insertar comentarios de modo de describir la operación (al menos los “big steps”)
3. Observar buenas prácticas de programación: Identar código, Usar convenciones para nombres de variables y funciones, Escribir código de manera “legible”.
4. Entender al comentario como parte escencial de la aplicación: Cada agregado o modificación debe ser documentado describiendo sus características, fecha y autor.

¿Que mas hace falta para comenzar?

1. Escribir comentarios claros, y suficientes para describir cada función, antes de comenzar a “codearla”.
2. Una vez dentro de ella, insertar comentarios de modo de describir la operación (al menos los “big steps”)
3. Observar buenas prácticas de programación: Identar código, Usar convenciones para nombres de variables y funciones, Escribir código de manera “legible”.
4. Entender al comentario como parte escencial de la aplicación: Cada agregado o modificación debe ser documentado describiendo sus características, fecha y autor.

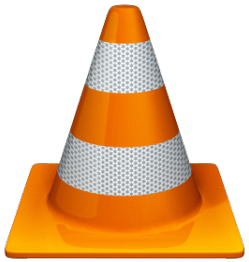
¿Que mas hace falta para comenzar?

1. Escribir comentarios claros, y suficientes para describir cada función, antes de comenzar a “codearla”.
2. Una vez dentro de ella, insertar comentarios de modo de describir la operación (al menos los “big steps”)
3. Observar buenas prácticas de programación: Identar código, Usar convenciones para nombres de variables y funciones, Escribir código de manera “legible”.
4. Entender al comentario como parte escencial de la aplicación: Cada agregado o modificación debe ser documentado describiendo sus características, fecha y autor.

¿Que mas hace falta para comenzar?

1. Escribir comentarios claros, y suficientes para describir cada función, antes de comenzar a “codearla”.
2. Una vez dentro de ella, insertar comentarios de modo de describir la operación (al menos los “big steps”)
3. Observar buenas prácticas de programación: Identar código, Usar convenciones para nombres de variables y funciones, Escribir código de manera “legible”.
4. Entender al comentario como parte escencial de la aplicación: Cada agregado o modificación debe ser documentado describiendo sus características, fecha y autor.

Proyectos documentados en Doxygen



Una vez instalado....

....Abrimos una terminal e ingresamos al directorio que contiene nuestros archivos de trabajo.
En ese directorio generamos el archivo de configuración para Doxygen.

Tipear:

```
doxygen -g
```

Doxyfile

- Es el archivo de configuración de doxygen.
- Es un archivo de texto.
- Todo lo que comienza con # se toma coo comentario
- Cada línea válida tiene el formato

```
PARAMETRO = VALOR
```

Parámetros de interés

Parámetro	Descripción / Valor
PROJECT_NAME	Nombre del Proyecto. Ejemplo: Trabajo Práctico Nº 4.
PROJECT_NUMBER	Número o versión del proyecto. Ej: Ejercicio 3.4..
OUTPUT_DIRECTORY	Es el directorio a partir del cual se generará el árbol de archivos y subdirectorios (ver opción siguiente) que componen la documentación. Generalmente usamos =./doxy, de modo que se genere dentro del directorio del proyecto.
CREATE_SUBDIRS	Valor default YES. En este caso crea dos niveles de subdirectorios a partir del de documentación para almacenar, los diferentes archivos que la componen.
OUTPUT_LANGUAGE	Lenguaje en el que queremos generar la documentación. Default English.

Parámetros de interés

Parámetro	Descripción / Valor
TAB_SIZE	Es la cantidad de espacios por los que reemplazará cada Tab encontrado en el código. Default: 8.
OPTIMIZE_OUTPUT_FOR_C	Si el proyecto es enteramente escrito en C conviene activar esta opción para que Doxygen optimice la salida para este lenguaje. En tal caso es = YES.
EXTRACT_ALL	Si está en YES, doxygen agregará a la documentación todo lo que encuentre (funciones, variables, etc), aunque no se hayan documentado.
INPUT	Es el directorio del proyecto, que se toma como entrada de información para Doxygen. En caso de poner Doxyfile en el mismo directorio del proyecto (como es nuestro caso) colocamos =./

Parámetros de interés

Parámetro	Descripción / Valor
INPUT_ENCODING	Establece el sistema de codificación de caracteres con el que se generará la documentación: Si se trabaja en inux colocamos UTF-8, para Windows ISO-8859.
GENERATE_LATEX	Conviene ponerlo en NO (el default es YES) si no queremos generar la documentación en LaTeX.

Es la parte mas fácil

Tipear en una consola dentro del directorio del proyecto:

```
doxygen Doxyfile
```

Luego, simplemente hay que abrir el archivo ./doxy/index.html con un navegador cualquiera y tenemos lista la documentación.

Como armar los comentarios

```
/**
* Estilo JavaDoc: Iniciar con '/**'
Apto para programas en C
Los asteriscos intermedios son opcionales
*/

/*!
* Estilo QT
Los asteriscos intermedios son opcionales
*/

///
// Estilo C++
```

Macros - Documentación general del programa

```
/**
\file programa.c
\brief Este archivo contiene el programa
       principal
\details Aqui nos explayamos sobre la tarea
         que reaizar  el programa o la
         funci n
\author Alejandro Furfaro afurfaro@ieee.org
\date 30 de Marzo de 2011
\version 1.0.0
*/
```

Macros - Documentación general de una función

```
/**
\fn void print_time (* struc tv)
\bref Funcion para obtener la fecha y la hora
      con formato.
\details En primer instancia se llama a gettimeofday,
         pasando como argumento el puntero a una
         estructura timeval (tv). Luego con strftime ()
         se le da un formato apto para presentar un
         time stamp con el formato
         a o-mes-dia horas:minutos:segundos.
\param [in] * struc tv: puntero a una
         estructura timeval (tv)
\return Nada (No regresa no regresa valores
\author Alejandro Furfaro afurfaro@ieee.org
\date 2011.05.08
\version 1.0.0
*/
```

Documentando variables

Versión compacta:

```
unsigned char Buffer [Buffersize]; //!< Buffer para recibir c
```

Versión extendida

```
/**
\var unsigned char Buffer [Buffersize];
\brief Buffer para recibir caracteres
\details Aqui si es necesario podemos explayarnos
         acerca de la variable.
*/
unsigned char Buffer [Buffersize];
```

Documentando variables

Estructuras

```
/**
\struct coordenada
\brief Variable para almacenar un par de coordenadas (x,y)
\details Aqui si es necesario podemos explayarnos
         acerca de la estructura.
*/
struct coordenada{
 int x; //!< Coordenada x;
 int y; //!< Coordenada y;
 };
```

Documentando variables

En general

- \struct
- \enum
- \union
- \class
- \def
- \typedef