Guion de prácticas

Documentación con Doxygen

Febrero de 2021

Metodología de la Programación

DGIM-GII-GADE

Curso 2020/2021

Índice

Esta sesión de prácticas está dedicada a aprender a utilizar el sistema de documentación Doxygen. Doxygen es un generador de documentación para C++, C, Java, Objective-C, Python, IDL (versiones Corba y Microsoft), VHDL y en cierta medida para PHP, C# y D. Te puede ayudar de varias formas:

1. Puede generar una documentación para ver en un navegador (en HTML) o un manual de referencia (en LaTeX) a partir de un conjunto ficheros fuentes documentados. También tiene soporte para generar salida en RTF (MS-Word), PostScript, PDF con hiperenlaces, HTML comprimido y páginas de manual de Unix.

2. La documentación se extrae directamente de los fuentes lo que hace más fácil mantener la documentación consistente con el código fuente.

3. Se puede, incluso, usar Doxygen para crear documentación normal (como se ha hecho con el manual de Doxygen y el sitio web)

1. Órdenes para realizar la documentación con Doxygen

Es de todos bien conocido la necesidad de documentar los programas de ordenador. Una buena forma de hacerlo es documentarlos en el mismo fichero fuente puesto que, de esta manera, la documentación se tiene siempre a la vista y se puede modificar fácilmente al tiempo que se modifica el código fuente.

Doxygen permite documentar los programas en el mismo fichero fuente y, posteriormente, extraer esa documentación para crear un manual en diferentes formatos. Para ello necesita identificar cual es la documentación para Doxygen y qué es lo que se está documentando.

La documentación de Doxygen se introduce en comentarios estándar de C++ y se identifican como documentación de Doxygen porque comienzan por `/**` o bien por `/*!`. Además se pueden usar `///` o `//!` para descripciones cortas que no necesitan más de una línea. Dentro de los bloques de documentación se escriben una serie de órdenes que Doxygen identifica y que le sirven para dar el formato correcto a la documentación. Estas órdenes comienzan con una `\` o bien con una `@`, por ejemplo, `@brief`, que sirve para hacer una descripción breve de un fichero o una función. Tras esta serie de órdenes pueden aparecer uno o varios párrafos con una descripción detallada del elemento que se documenta. Se deben documentar todos los elementos de un programa, es decir, el fichero, las estructuras, las constantes globales, las funciones, etc. La documentación de cada elemento se suele colocar justo antes del elemento aunque se puede colocar en cualquier otro sitio usando las órdenes apropiadas.

Así, la documentación de un programa tendrá la siguiente estructura:

```
/**
Documentación del fichero, usando las órdenes
referentes a  fichero

Documentación larga referente al fichero */

#include<los_necesarios>
/**
Documentación de cada constante global, si existen,
usando las órdenes correspondientes

Documentación larga referente a la constante */
const double constante=0;

/**
Documentación de las estructuras, si existen,
usando las órdenes correspondientes

Documentación larga referente a la estructura */
struct complejo{
   /// Documentación de r
   double r;

   /// Documentación de i
   double i;
}

/**
Documentación de las clases, si existen,
usando las órdenes correspondientes

Documentación larga referente a la calse */
class punto{
public:

    /**
    Documentación de los métodos, usando las órdenes
    correspondientes

    Documentación larga referente al método */
    punto(){...};
...
};

 /**
 Documentación de las funciones, usando las órdenes
 correspondientes

 Documentación larga referente a la función */

 int f(){
 ....
```

Cuando se documentan miembros de estructuras o clases o constantes globales a veces es cómodo poner la documentación de los miembros tras su declaración. Para ello se usa la marca <, como en el siguiente ejemplo:

```
const int DIM = 50; ///< Dimensión de los vectores ....

struct complejo{
   double r;    ///< Documentación de r
   double i;    ///< Documentación de i
}

class punto{
public:
   punto(){...}; ///< Documentación corta del constructor
...
};
```

Recordad que otras formas posibles de documentación son:

```
   //!< Documentación de x
   /**< Documentación de x */
   /*!< Documentación de x */
```

IMPORTANTE

- No es necesario documentar con Doxygen ni las variables locales a los módulos (ni al programa principal, por supuesto), ni los comentarios explicativos del código que se insertan dentro del mismo.
- Salvo excepciones se considerará que la documentación de Doxygen consiste en la guía de la programación de todo el proyecto, por lo que se documentan todas las clases, todos los métodos de la clase y todas las funciones.

1.1. Creación de listas

Aunque Doxygen tiene varias formas de crear listas, la más sencilla y cómoda a nuestro entender, es la siguiente: Poniendo una serie de signos menos, alineados por columnas, al comienzo de una línea automáticamente se ponen *topos* (marcas). Las listas numeradas se pueden generar poniendo una almohadilla (#) tras el guión. Se permite el anidado de las listas. Por ejemplo:

```
/**
Lista de valores:
 - lista 1
    -# primer valor de lista 1
    -# segundo valor de lista 1
    -# tercer valor de lista 1
 - lista 2
    -# primer valor de lista 2
    -# segundo valor de lista 2

 Texto tras las listas
*/
```

producirá:

```
Lista de valores:
* lista 1
   1. primer valor de lista 1
   2. segundo valor de lista 1
   3. tercer valor de lista 1
* lista 2
   1. primer valor de lista 2
   2. segundo valor de lista 2
Texto tras las listas
```

A continuación se describen brevemente las órdenes más comunes de Doxygen. Se puede consultar la documentación completa de las mismas, así como otras muchas, en la sección *Special Commands* del manual de Doxygen. Como notación, los argumentos entre corchetes `[ ]` son opcionales, entre llaves `{ }` significa que se extienden hasta el final del párrafo, entre paréntesis `( )` que se extienden sólo hasta el final de la línea y entre ángulos `< >` que son una única palabra.

1.2. Órdenes básicas

El texto que necesita la mayoría de estas órdenes continúa hasta que se encuentra una línea en blanco o aparece alguna otra orden de Doxygen.

`@brief {descripción breve}`

Comienza un párrafo que sirve como una descripción breve del cometido del elemento documentado. Se puede omitir en un bloque de documentación si la descripción breve es la primera línea del bloque y ocupa sólo una línea.

`@file [<nombre-fichero>]`

Indica que el bloque comentado contiene documentación de un fichero con el nombre nombre-fichero. Si se omite el nombre de fichero se entiende que la documentación se refiere al fichero actual. IMPORTANTE: Si no se especifica `@file` al comienzo de un fichero, en la salida no se incluirá la documentación de las funciones, estructuras, clases, constantes globales, etc. de ese fichero.

`@param <nombre-parametro> {descripción del parámetro}`

Comienza la descripción de un parámetro llamado `nombre-parametro` que pertenece a un función. La existencia o no existencia del parámetro no se comprueba.

`@return {descripción del valor de retorno}`

Se usa para describir el valor de retorno de una función de forma genérica. Por ejemplo:

```
@return la posición del máximo en el vector
```

`@retval <valor de retorno> {descripción}`

Se describe un valor de retorno concreto de una función. Por ejemplo:

```
@retval true si el número es primo
@retval false si el número no es primo
```

`@struct <nombre>`

Indica que un bloque de comentario contiene documentación sobre una estructura llamada nombre. También hay que documentar los miembros de las estructuras.

`@var (declaración de variable)`

Indica que un bloque de comentario contiene documentación sobre una constante o variable.

1.3. Órdenes adicionales

`@author {lista de autores}`

Comienza un párrafo donde se introducen los nombres de uno o más autores.

`@bug {descripción del fallo}`

Comienza un párrafo donde se describen los fallos conocidos de un módulo o programa.

`@date {descripción de la fecha}`

Comienza un párrafo donde se puede introducir una o más fechas. El texto que se introduce no tiene una estructura especial.

`@fn (declaración de la función)`

Indica que el bloque comentado contiene documentación de una función. Sólo es necesario si la documentación no está situada justo antes de la cabecera de la función. Si la documentación está situada justo antes de la cabecera de la función es mejor no ponerlo (para evitar redundancias y posibles errores).

`@par [título] {párrafo}`

Comienza un párrafo nuevo titulado título. Se considera título hasta el final de la línea. El párrafo tras esta orden se sangra convenientemente. Si se omite el título sólo se comenzará un nuevo párrafo. Funciona dentro de otras órdenes que crean párrafos como `@param` o `@pre`.

`@pre {descripción de la precondición}`

Comienza un párrafo donde se describe una precondición. Si hay varias precondiciones se colocarán varias líneas.

`@post {descripción de la postcondición}`

Comienza un párrafo donde se describe una postcondición. Si hay varias postcondiciones se colocarán en varias líneas.

`@sa {referencias}` o `@see {referencias}`

Comienza un párrafo donde se puede especificar una o más referencias a funciones, variables, ficheros, o URLS.

`@version {versión}`

Comienza un párrafo donde se puede introducir uno o más números de versión. El texto que se introduce no tiene una estructura especial.

1.4. Órdenes para mejorar la presentación

`@a <palabra>`

Escribe la palabra usando un tipo de fuente de letra especial. Esta orden se utiliza para hacer referencia a argumentos dentro del texto.

`@b <palabra>`

Escribe la palabra en negrita.

`@c <palabra>`

Escribe la palabra usando una letra no proporcional. Se usa para referirse a una palabra de código (`void, for`, ...) o nombres de funciones.

`@e <palabra>`

Escribe la palabra en cursiva. Esta orden se utiliza para hacer énfasis en determinadas palabras

`@n`

Esta orden escribe un salto de línea.

`@p <palabra>`

Escribe la palabra usando una letra no proporcional. Se usa para referirse a parámetros en el texto.

`@\` Esta orden escribe el carácter especial \

`@@`

Esta orden escribe el carácter especial @

`@$`

Esta orden escribe el carácter especial $

`@&`

Esta orden escribe el carácter especial &

`@#`

Esta orden escribe el carácter especial #

`@<`

Esta orden escribe el carácter especial <

`@>`

Esta orden escribe el carácter especial >

2. Extracción de la documentación con Doxygen

Para extraer la documentación de un proyecto documentado con Doxygen es necesario crear un fichero de configuración de Doxygen para el proyecto documentado y ejecutar la orden `doxygen` usando ese fichero de configuración. El fichero de configuración, escrito en texto convencional, contendrá información sobre los ficheros de los que Doxygen debe extraer la documentación, el idioma de la documentación, los formatos en los que se desea la documentación, detalles sobre esos formatos y cientos de opciones adicionales que se pueden consultar en el sitio web de referencia de Doxygen.

Una vez creado este fichero de configuración la forma de generar la documentación es ejecutar `doxygen <fichero>`, es decir, si el fichero de configuración se llama `fichero.doxy`, ejecutar

```
doxygen fichero.doxy
```

Aunque el fichero de configuración puede contener muchas opciones, un contenido mínimo del fichero de configuración sería:

- `PROJECT_NAME = "(nombre del proyecto)"`
  Nombre del proyecto. Usualmente una palabra. Sirve para identificar el programa.
- `PROJECT_NUMBER = <número>`
  Número de la versión (si existe).

- `OUTPUT_LANGUAGE = SPANISH`
  Lenguaje en que estará escrita la documentación.
- `OUTPUT_DIRECTORY = ./doc`
  Directorio o carpeta en la que se almacenará la documentación.
- `INPUT = ./`
  Carpeta que contiene el proyecto.
- `RECURSIVE = YES`
  En caso de que el proyecto esté distribuido en varias subcarpetas.
- `FILE_PATTERNS = *.cpp *.h`
  Extensiones de los ficheros que contienen documentación útil para doxygen.
- `SOURCE_BROWSER = YES` y `INLINE_SOURCES = YES`
  Incluye el código fuente enlazado desde la documentación.

Existe también una herramienta gráfica, llamada `doxywizard`, que nos permite crear el fichero de configuración y ejecutar Doxygen sobre él. Puede consultar la documentación de Doxygen para obtener más información sobre ella.

3. Grafos de llamadas

En Linux, el uso de doxygen puede generar información gráfica como las dependencias de ficheros de cabeceras (Figura 1) o los grafos de llamadas entre funciones y/o métodos (Figura 2). Para ello es necesario tener instalado el paquete **graphviz**

```
./sudo apt-get install graphviz
```

y añadir las siguientes líneas al fichero doxy

- `HAVE_DOT = YES`
- `CALL_GRAPH = YES`

4. Problemas más frecuentes y soluciones

1. No se genera la documentación (ni directorios ni nada).

   Si al ejecutar doxygen nos dice que no puede encontrar un fichero fuente, comprobar que el fichero está correctamente introducido en la pestaña de Input y que el nombre o el camino no contiene espacios, acentos o caracteres especiales. Si nos dice que no puede crear el directorio de salida, comprobar que el directorio está correctamente introducido el fichero de configuración y que el nombre

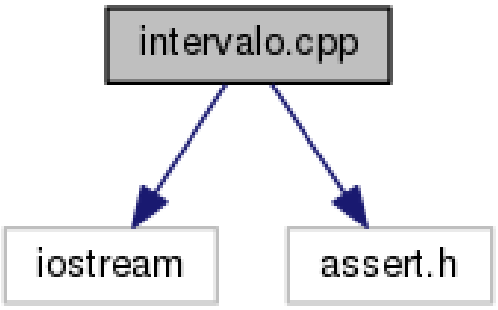

Figura 1: Depedencias de ficheros de cabeceras

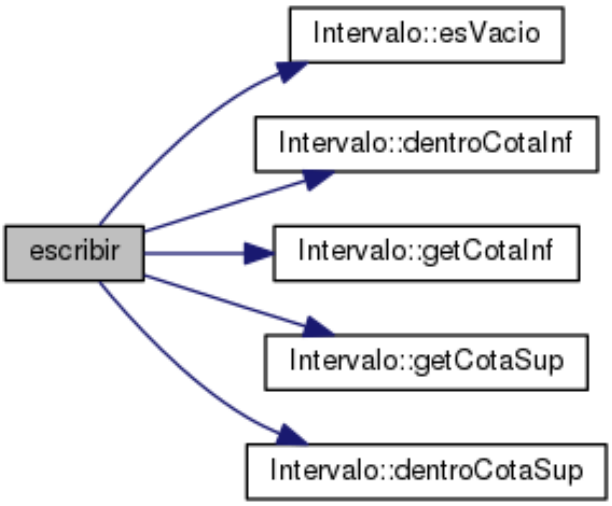

Figura 2: Grafo de llamadas de una función

o el camino no contiene espacios, acentos o caracteres especiales (normalmente no debe dar problemas, pero no está de más evitar caracteres especiales). Si todo está correcto, crear el directorio de salida a mano.

2. Se generan los directorios de documentación pero no aparece la documentación del fichero

   Comprobar que el fichero está documentado, es decir, que tiene una línea con `@file` al principio del fichero. Si no la tiene, introducirla. Si las tiene, comprobar que el nombre del fichero que sigue a `@file` (si hay alguno) corresponde exactamente (incluyendo mayúsculas y minúsculas) con el nombre del fichero.

3. No se genera la documentación de algunas funciones

   Comprobar que, si existe la línea `@fn cabecera`, esta se corresponde exactamente, incluyendo espacios, mayúsculas y minúsculas con la cabecera de la función.