Doxygen

Doxygen es un programa que interpreta la documentación del código fuente de un programa y genera informes sobre el mismo.

Una buena documentación es muy útil, principalmente en los casos siguientes:

- En el trabajo en equipo. Cuando varios módulos que interactúan entre sí son desarrollados por varios programadores, estos deben conocer el prototipo y la forma de uso de los demás módulos para elaborar su trabajo.
- En el mantenimiento del programa. Si se desea actualizar un programa, el código bien documentado facilita la labor.
- En el desarrollo de bibliotecas. Si se desea hacer una biblioteca para su uso general, el programador que la utilice deberá conocer cómo utilizar los tipos de dato y los módulos existentes en la misma.

Documentación con Doxygen

La forma más rápida de documentar un programa es en el propio código fuente, mediante el uso de comentarios.

Doxygen interpreta los comentarios de un fichero de código fuente, y genera un informe fácil de leer y con una buena presentación (por ejemplo: en HTML, en LaTEX, PDF...).

Inconveniente: El uso de Doxygen requiere que los comentarios se escriban en un formato concreto que pueda ser procesados por el intérprete.

Documentación con Doxygen

La documentación de Doxygen se introduce en comentarios estándar de C++. No obstante, para diferenciar comentarios del código de comentarios de Doxygen, estos tienen el siguiente formato:

- Para descripciones largas se utiliza `/** .... */`
- Para descripciones cortas de no más de una línea se utiliza `/// ...`
- Los comandos específicos suelen comenzar con `@comando`, dentro del comentario.

Documentación de ficheros con Doxygen

Un fichero de código fuente únicamente podrá ser procesado por Doxygen si comienza con un comentario largo e información sobre el nombre del fichero y su contenido:

```
/**
 * @file nombre del fichero
 * @brief Descripción del contenido del fichero
 * @author Autor/es del fichero
 * @date Fecha de última modificación
 * @version Versión del fichero
 */
```

El comando **@file es imprescindible** para que Doxygen procese el fichero; **@brief es muy recomendable de usar** pero opcional, y todos los demás son opcionales.

Ejemplo de documentación de ficheros con Doxygen

Ejemplo: Programa holamundo.cpp

```
/**
 * @file holamundo.cpp
 * @brief Programa que muestra el mensaje "Hola, Mundo" por pantalla
 * @author Paco López
 * @date 05/06/2008
 * @version 1.0
 */
#include <iostream>
using namespace std;

int main(){

  cout<<"Hola, Mundo";
  return  0;
}
```

Documentación de módulos con Doxygen

Para cada módulo, se debe especificar la información siguiente:

- Prototipo del módulo. Doxygen lo genera automáticamente, siempre y cuando el comentario se encuentre justo antes de la declaración del módulo.
- Descripción de la operación realizada por el módulo. Se realiza con el comando **@brief**.
- Descripción de los parámetros de entrada, salida y entrada y salida al módulo. Se realiza con el comando **@param**.
- Descripción del valor de retorno del módulo. Se realiza con el comando **@return**. Si se desea especificar concretamente qué valores devuelve, se utiliza el comando **@retval** para cada valor posible.
- Precondiciones del módulo. Condiciones que deben cumplirse en el programa y en los argumentos del módulo para su correcto funcionamiento. Se realiza con el comando **@pre**
- Postcondiciones del módulo. Efectos realizados por el módulo tras su ejecución (cambios en argumentos de salida, efectos colaterales, etc.). Se realiza con el comando **@post**

Doxygen y módulos: Uso de comandos

En la documentación del módulo también puede aparecer información adicional como autor, versión, fecha, etc.

Uso de **@param**

@param *identificador* Descripción.

Describe el parámetro con nombre *identificador* del módulo.

Uso de **@retval**

@retval *literal* Descripción.

Describe en qúe condiciones el módulo retorna el valor *literal*.

El resto de comandos se utilizan de la siguiente forma:

Demás comandos

@comando Descripción.

Describe qúe condiciones se dan en el módulo, según el comando utilizado.

Doxygen y módulos: Ejemplo del uso de comandos

Ejemplo:

```
/**
 * @brief Comprueba si un número es primo o no
 * @param num Número a hallar si es primo
 * @retval true si el número es primo.
 * @retval false si el número no es primo.
 * @pre num debe ser mayor que 0
 */
bool esPrimo(int num);
```

Ejemplo:

```
/**
 * @brief Intercambia el valor de dos variables enteras
 * @param a Primera variable a intercambiar
 * @param b Segunda variable a intercambiar
 * @post a y b tienen sus valores intercambiados
 */
void Intercambia(int  &a, int  &b);
```

Doxygen y módulos: Ejemplo del uso de comandos

Ejemplo:

```
/**
 * @brief Busca un valor en un vector de enteros
 * @param v Vector donde se busca el valor
 * @param n Longitud del vector
 * @param valor Valor a buscar en el vector v
 * @return El índice de la componente del vector v donde está el valor, o
-1 si no se ha encontrado
 * @pre v debe tener al menos n componentes, con n mayor que 0
 */
int Busca(int v[], int n, int valor);
```

Documentación adicional con Doxygen

Existen comandos adicionales en Doxygen que permiten mejorar la presentación de la documentación final, o incluso la navegación a través de la misma. Por ejemplo, se pueden resaltar palabras, hacer referencias a otros módulos u objetos, o escribir caracteres especiales.

- Comando \a. Escribe la palabra siguiente en un tipo de letra especial. Se utiliza para resaltar nombres de parámetros o variables en el texto.
- Comando \b. Escribe la palabra siguiente en negrita.
- Comando \e. Escribe la palabra siguiente en cursiva.
- Comando \n. Escribe un salto de línea.
- Comandos \\, \@, \$, \&, \#, \ <, \ >. Escriben el caracter especial correspondiente.

Doxygen y módulos: Ejemplo del uso de comandos

Ejemplo:

```
/**
 * @brief Busca un valor en un vector de enteros
 * @param v Vector donde se busca el valor
 * @param n Longitud del vector
 * @param valor Valor a buscar en el vector \a v
 * @return El índice de la componente del vector \a v donde está el
valor, o \b -1 si no se ha encontrado
 * @pre \a v debe tener al menos \a n componentes, con \a n \ > 0
 */
int Busca(int v[], int n, int valor);
```

Doxygen y módulos: otros comandos

- Comando **@see**. Crea una referencia a otros puntos de la documentación que tienen relación con el módulo que se está definiendo.

Ejemplo de uso:

```
/**
 * @brief Suma todos los primos menores que \a n
 * @param n Entero para sumar los primos hasta n
 * @return La suma de todos los primos menores que n
 * @pre n mayor que 0
 * @see bool esPrimo(int n)
 */
int SumaPrimos(int n);
```

Doxygen y módulos: otros comandos

- Comando **@include**. Crea un enlace a un fichero con información adicional. Se utiliza cuando se dispone de ficheros de ejemplo de uso del módulo.

Ejemplo de uso:

```
/**
 * @brief Suma todos los primos menores que \a n
 * @param n Entero para sumar los primos hasta n
 * @return La suma de todos los primos menores que n
 * @pre n mayor que 0
 * @see bool esPrimo(int n)
 *
 * Un ejemplo de su uso:
 * @include ejemplosumaprimos.cpp
 */
int SumaPrimos(int n);
```

Instalación de Doxygen

Instalación

Doxygen puede descargarse desde su web principal, en http://www.doxygen.org . .

La instalación básica es bastante sencilla: Ejecutar el instalador **.exe** y seguir los pasos hasta el final.

Existen instalaciones avanzadas, que requieren de software adicional para generar la documentación en LaTEX y otros formatos, aunque estas quedan fuera de los límites de la asignatura.

Configuración de Doxygen

Para generar la documentación con Doxygen es necesario previamente crear un fichero de configuración. En este fichero se almacenarán las opciones de generación (formatos de salida -HTML, LaTEX, RTF...-, idioma, carpetas donde se guardarán los ficheros generados, carpetas donde se encuentra el código fuente, etc.).

Para crear el fichero de configuración, ejecutaremos el programa doxywizard.exe, incluido en la carpeta de instalación de Doxygen. Cuando se abra la ventana del programa, pulsaremos sobre el botón titulado wizard, para iniciar el asistente.

Configuración de Doxygen: Pestaña Project

La pestaña Project permite introducir cuál es el nombre del proyecto (programa) del que se genera la documentación, así como su versión.

Además, se nos permite especificar cuál es la carpeta que contiene el código fuente (*Source Code Directory*). Si el código está organizado en varias subcarpetas, deberemos pulsar sobre el botón Scan Recursively, para que incluya todo el código fuente de las subcarpetas.

Por último, podremos especificar la carpeta destino donde Doxygen guardará los ficheros de documentación de salida.

Configuración de Doxygen: Pestaña Project

La pestaña Project permite introducir cuál es el nombre del proyecto (programa) del que se genera la documentación, así como su versión.

Además, se nos permite especificar cuál es la carpeta que contiene el código fuente (*Source Code Directory*). Si el código está organizado en varias subcarpetas, deberemos pulsar sobre el botón Scan Recursively, para que incluya todo el código fuente de las subcarpetas.

Por último, podremos especificar la carpeta destino donde Doxygen guardará los ficheros de documentación de salida.

Avanzado

La pestaña Project, si pulsamos en el botón expert en la ventana principal del programa, además permite configurar otras opciones, como el idioma de la documentación de salida.

Configuración de Doxygen: Pestaña Mode

La pestaña Mode permite seleccionar qué elementos del programa son susceptibles de ser documentados, y el lenguaje en el que se encuentra el código fuente.

Para la asignatura, seleccionaremos que deseamos generar la documentación para los elementos que hemos comentado en el código (opción *Documented Entities Only*), y el lenguaje de programación *C++*.

Configuración de Doxygen: Pestaña Output

La pestaña Output permite seleccionar el formato de salida de la documentación: HTML, LaTEX, RTF, XML o páginas de manual de Linux.

Para la asignatura, seleccionaremos que deseamos generar la documentación en HTML o en RTF, a gusto del alumno.

Configuración de Doxygen: Pestaña Diagrams

Doxygen permite generar diagramas de relaciones entre clases y de dependencias. No obstante, para utilizar esta opción es necesario tener instalado en el PC el software GraphViz.

La pestaña Diagrams permite seleccionar qué tipo de diagramas generará Doxygen. En la asignatura, no generaremos ningún diagrama dada la simplicidad de los programas desarrollados. Marcaremos, por tanto, la opción No Diagrams.

Configuración de Doxygen: Guardar fichero de configuración

Tras haber configurado Doxygen, pulsaremos el botón Ok para volver a la pestaña principal.

En la ventana principal, el botón Save nos permitirá guardar la configuración establecida para su uso en futuros programas.

Ejecución de Doxygen

Doxygen puede ejecutarse de dos formas: por línea de comandos (facilitando así la generación de documentación desde ***makefiles***, y de forma visual.

- Ejecución desde línea de comandos:

  doxygen fichero_configuración_doxygen

- Ejecución desde doxywizard:

  Seleccionaremos el directorio desde el cual debe ejecutarse Doxygen (Working directory), seguido del botón start.

En ambos casos, la documentación se generará en la carpeta especificada en el fichero de configuración.

Documentar clases con Doxygen

La documentación de una clase con Doxygen conlleva la implementación de comentarios en 3 partes fundamentales:

- En la declaración de la clase. Se debe hacer una breve descripción de la clase, junto con una descripción un poco más larga de qué representa y cómo utilizarla (métodos, atributos públicos, etc.)
- En la declaración de atributos de la clase. Se debe comentar cada atributo por separado, indicando qué representa dentro de la clase, para qué se utiliza y sus restricciones.
- En la declaración de métodos de la clase. Se realiza de igual forma que la **documentación de módulos con Doxygen**.

Documentar clases con Doxygen

Ejemplo: Documentación general de la clase T.D.A. Complejo

```
/**
 * @brief T.D.A. Complejo
 *
 * Una instancia @a c de la clase @b Complejo es un objeto
 *  que representa un elemento del conjunto de los números complejos.
 * Lo representaremos como a+b*i (parte real e imaginaria). @n
 *
 * Un ejemplo de su uso:
 * @include ejemplocomplejo.cpp
 */
class Complejo {
...


...
};
```

Documentar clases con Doxygen: Atributos

La documentación de los atributos se realiza a la derecha de su declaración, y utilizando el comando siguiente: */**< Comentario del atributo */*

Ejemplo: Comentario de los atributos del T.D.A. Complejo

```
...
private:
  double  real; /**< Parte real del complejo */
  double  imag; /**< Parte imaginaria del complejo */
...
```