Tema 7:

Doxygen

Doxygen

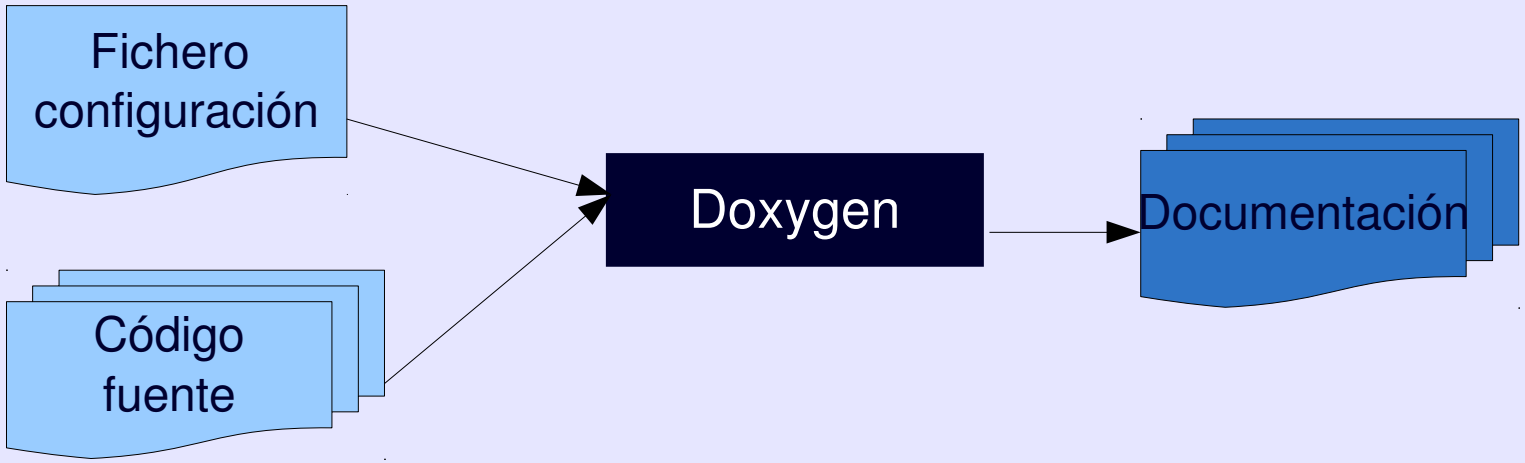
- Sistema generador de documentación
  - Genera documentación a partir de la estructura y los comentarios presentes en el código.
    - Páginas man
    - Documentación en formato html para visualizar
    - Documentación en formato latex para imprimir
    - Pdf
  - Se puede usar con diferentes lenguajes de programación
    - C++, Java, C, IDL, Python, PHP
  - Puede extraer la estructura del programa aún cuando no hay documentación
  - Configurable para generar diferentes tipos de documentación

Doxygen

- Utiliza un fichero de configuración para determinar
  - De qué código extraer documentación
  - Qué tipo de documentación crear
- Utiliza la estructura del código para construir
  - Diagramas y relaciones entre clases
  - Índice de métodos y funciones
  - Enlaces entre las clases y los métodos
- Utiliza comentarios especiales para
  - Proporcionar una descripción de los métodos, las clases, los parámetros, ...

Utilizar Doxygen

- Genera un fichero de configuración
  - `doxygen -g <fichero_configuracion>`
- Editar el fichero con nuestras preferencias
- Genera la documentación a partir del fichero de configuración y los códigos fuentes
  - `doxygen <fichero_configuracion>`



Fichero de configuración

- PROJECT_NAME
  - Nombre del proyecto
- PROJECT_NUMBER
  - Versión del proyecto
- OUTPUT_DIRECTORY
  - Directorio donde se almacenará el resultado
- OUTPUT_LANGUAGE
  - Idioma en el que se generará la documentación
    - Spanish /English
- INPUT
  - Directorio con los fuentes o lista de ficheros separados por comas.

Fichero de configuración

- `FILE_PATTERNS`
  - Tipos de archivos
- `SOURCE_BROWSER`
  - Muestra el código
- `GENERATE_HTML`
  - Si se quiere generar la documentación en html
- `GENERATE_LATEX`
  - Si se quiere generar la documentación en latex
- `GENERATE_MAN`
  - Si se quiere generar la documentación como páginas man

Comentarios en C

- Línea simple
  - `// Comentario de una línea`
- Varias líneas
  - `/* … línea 1 …`

    `… línea 2 … */`

Comentarios en Doxygen

- Línea simple
  - Tipo 1 (Estilo C)

```
///
///    Texto
///
```

  - Tipo 2 (Estilo Qt)

```
//!
//! Texto
//!
```

  - **Ejemplos**
    - `/// Comentario de una línea`
    - `//! Comentario de una línea`

Comentarios en Doxygen

- Bloque de varias líneas
  - Tipo 1 (Estilo C)

```
/**
*    Texto
*/
```

  - Tipo 2 (Estilo Qt)

```
/*!
* Texto
*/
```

  - **Ejemplos**

```
/** Esto es un comentario que ocupa
*    varias lineas
*/
```

Comentarios en Doxygen

- Los comentarios puede ir antes o después del bloque de código al que se refiere
  - Antes

```
///Funcion que suma dos enteros
int suma (int a, int b);
```

  - Después

```
int suma (int a, int b);
///< Funcion que suma dos numeros
```

¿Qué se comenta?

- ¿Qué hay que comentar en estilo doxygen?
  - Información sobre el fichero – programa
  - Las funciones
  - Las estructuras definidas
  - Los defines
  - Las variables globales
- ¿Qué información incluir en la documentación?
  - Descripción breve y/o detallada
  - Autor, fecha,
  - Información específica sobre lo que estemos documentando
    - Parámetros y valor devuelto para las funciones
    - Campos para las estructuras, ...

Descripciones breves y detalladas

- Bloque de código (main o función) → bloque de comentarios
  - Descripción breve
    - @brief <descripción breve>
  - Descripción detalla
    - Lo que aparezca a continuación de una línea en blanco
  - **Ejemplos**

```
/**
 @brief Descripción breve

 Explicación detallada
*/
```

Descripciones breves y detalladas

- Bloque de código (main o función) → bloque de comentarios
  - Descripción breve
    - ///< <descripción breve>
  - Descripción detalla
    - /**< <descripción detallada>*/
  - **Ejemplos**

```
struct punto{
 float x; /**<Coordenada x */
 float y; ///<Coordenada y
};
```

Comandos

- Los comandos pueden estar precedidos por @ o \
  - @author
    - Comentarios sobre el autor
  - @date
    - Fecha de creación
  - @file
    - Nombre del fichero a comentar
  - @brief
    - Descripción breve
  - @struct
    - Documentar una estructura

Comandos

- Los comandos pueden estar precedidos por @ o \
  - @fn
    - Documentar una función
  - @var
    - Documentar una variable
  - @def
    - Documentar un define
  - @typedef
    - Documentar un typedef
  - @version
    - Versión del documento

Comandos

- Comandos relacionados con la documentación de una función
  - @param
    - Documentar los parámetros de una función
  - @return
    - Documentar el resultado de una función
  - @post
    - Documentar las postcondiciones de una función
  - @pre
    - Documentar las precondiciones de una función

Comandos

- Comandos que modifican el aspecto de la documentación
  - @n
    - Salto de línea
  - @b
    - Pone en negrita la siguiente palabra
  - @em
    - Pone en cursiva la siguiente palabra
  - @p
    - Pone en estilo curier la siguiente palabra
  - \@, \\, \$, \&, \#, \%, \<, \>
    - Escribe los símbolos @, \, $, &, #, %, <, >

Comandos

- Comandos que modifican el aspecto de la documentación
  - @li <descripción de un item>
    - Genera un lista de argumentos. Cada elemento de la lista empieza con @li
- Comandos para insertar fórmulas
  - @f$
    - Incluir una fórmula de texto
  - \f[
    - Comienzo de una fórmula larga
  - \f]
    - Fin de una fórmula larga

Comandos(Referencias)

- Descripción de los comandos doxygen
  - http://www.stack.nl/~dimitri/doxygen/commands.html
- Como incluir fórmulas
  - http://www.stack.nl/~dimitri/doxygen/formulas.html

Ejemplos

- Documentación inicial de un fichero

```
/** ---------------------------------------------
 @file imagenes.c
 @brief Descripción de las funciones
 @author Maria Luque Rodriguez
 @date 13-04-2012
 @version 1.0

 Este fichero contiene el código de las funciones
 necesarias para trabajar con imagenes:
 @li Cargar imagen
 @li Grabar imagen
 @li Espejo horizontal
 ---------------------------------------------*/
```

Ejemplos

```
/* ----------------------------------------------
   Nombre: restaCuadrados
   Tipo: entero (int)
   Objetivo: calcula la diferencia de los cuadrados
             de dos números (n1^2 -n2^2)
   Parámetros entrada:
      int n1: primer numero
      int n2: segundo numero
   Precondiciones: Ninguna
   Valor devuelto: n1^2-n2^2
   Utiliza: cuadrado
   Autor: Maria
   Fecha: 20-11-2008
----------------------------------------------*/

int restaCuadrados(int n1, int n2);
```

Ejemplos

- Documentación de una función

```
/** ----------------------------------------
 @fn restaCuadrados(int n1, int n2)
 @brief Calcula la diferencia de dos cuadrados
 @param n1 Primer numero
 @param n2 Segundo numero
 @pre  Ninguna
 @return \f$ n_1^2 - n_2^2

 @author Maria Luque Rodriguez
 @date 13-04-2012

 Esta funcion calcula la diferencia entre los
cuadrados de dos números pasados como argumentos
 ----------------------------------------*/

int restaCuadrados(int n1, int n2);
```

Ejemplos

- Documentación de una estructura

```
/**
 @struct punto
 @brief Definicion de una estructura de tipo punto
*/
struct punto
{
   float x; /**<Coordenada x del punto (D. larga)*/
   float y; ///<Coordenada y del punto (D. breve)
};
```

- Documentación de un typedef

```
/**
 @typedef stPunto
 Prueba de typedef
*/
typedef struct punto stPunto;
```

Ejemplos

- Documentación de una macro

```
/*!
   @def MAX(x,y)
   Calcula el máximo de \a x and \a y.
*/
#define MAX(x,y) ((x)>(y)?(x):(y))
```

- Documentación de una constante

```
/** @def PI
 Constante con el valor de PI (3.1416) */
#define PI 3.1416
```