Documentación de programas con doxygen

1. Introducción al doxygen

doxygen es una aplicación que permite extraer y formatear documentación a partir de programas escritos en C++, Java, Python, etc. Su uso es muy fácil si no se aspira a personalizar demasiado el formato del documento resultante.

Si programamos con clases de C++, el input del Doxygen son nuestros ficheros .hh y .cc. El uso de Doxygen se basa en colocar estratégicamente una serie de comentarios especiales en dichos ficheros. Dentro de los comentarios se pueden usar unas ciertas palabras reservadas o comandos de formateo tipo html. En PRO2 usaremos el Doxygen para documentar especificaciones y también para documentar programas completos.

En el primer caso, los comentarios se colocarán solo en los .hh de las clases y en el .cc que contenga el main, pues lo que documentaremos principalmente serán las operaciones de las clases, ya sea para facilitar su uso, o para guiar su diseño, en el caso de la práctica. En el segundo caso, también intervendrán los ficheros .cc de las clases. La gran ventaja en ambos casos es que no tendremos que escribir un documento aparte del código y evitaremos mucha duplicidad de textos. Además, con muy poco esfuerzo por nuestra parte conseguiremos una documentación con un aspecto muy profesional.

2. Aspecto de la documentación generada por doxygen

Mostraremos la documentación de un proyecto basado en las clases Lavadora, Cubeta y Prenda. El material para realizar las actividades está en la carpeta /assig/pro2/sessio8.

La especificación de dichas clases, junto con la plantilla para un programa principal que las use, se puede encontrar en los formatos generados por doxygen en la subcarpeta DOC. Ésta, a su vez, contiene las carpetas html y latex. Si queréis ver la documentación en html, id a la carpeta correspondiente y abrid el fichero index.html. Si queréis ver una versión en pdf, id a la carpeta latex y abrid el fichero refman.pdf.

En la sesión 8 se propone un ejercicio consistente en obtener el programa principal de un proyecto que utilice dichas clases. Más adelante os pediremos actualizar la documentación correspondiente mediante doxygen.

3. Uso de doxygen

En esencia, documentar una clase o un programa mediante doxygen solo requiere introducir en éstos unos comentarios especiales, acotados por los caracteres /** y */. Dichos comentarios serán procesados por doxygen para producir la documentación en los formatos que habéis visto en las carpetas html y latex. Por ejemplo, el comentario típico de doxygen para una operación de una clase (en este caso, el afegir_nota de la clase Estudiant) tiene la forma:

```
/** @brief Modifica la nota d'un estudiant amb nota
    \pre el parametre implicit no te nota, 0 <= "nota" <= nota_maxima()
    \post la nota del parametre implicit passa a ser "nota"
*/
    void afegir_nota(double nota);
```

Notad que solamente se diferencia de un comentario normal en que hay dos asteriscos en lugar de uno al abrir comentario.

Lo que empieza por @ o por \ son palabras reservadas, que han de ir seguidas de textos. @brief viene seguida por una descripción breve de la operación, que será usada en varios lugares de la documentación. Si se quiere incluir una explicación más completa, ésta ha de ir separada por una línea en blanco. Dicha explicación adicional solo se usará en lo que se considera la "descripción detallada" de la operación.

Si no queremos usar las comillas para diferenciar el parámetro nota en los textos de este ejemplo, podemos usar , <b>, etc. como en html

```
/** @brief Modificadora de la nota, para estudiantes sin nota
    \pre El parámetro implícito no tiene nota, <em>nota</em> es una nota válida
    \post La nota del parámetro implícito pasa a ser <em>nota</em>
*/
    void afegir_nota(double nota);
```

La documentación generada por doxygen sale por defecto en los formatos html y latex. Se crean sendas carpetas con todos los ficheros necesarios. La carpeta de latex incluye un makefile para obtener una versión en pdf sin saber nada de latex.

Dicha documentación está estructurada alrededor de las clases y los ficheros que constituyen nuestro programa. Para cada clase sale un resumen y la ya mencionada "descripción detallada". Hay mecanismos para ocultar la implementación de la clase y otros detalles que no queramos que salgan en la documentación. Para cada fichero, se incluye información sobre la clase con la que está relacionado y, si se desea, un gráfico con las dependencias definidas por los #includes, que puede ser manipulado para parecerse a los diagramas modulares clásicos.

Las opciones de configuración de doxygen se definen en un fichero habitualmente llamado Doxyfile. El que os incluimos en la carpeta de la sesión contiene todas las opciones necesarias para documentar las especificaciones del laboratorio y de la práctica de la asignatura. En esta sesión solo debéis tocarlo para modificar los datos identificativos de vuestro programa, definir las rutas a vuestras carpetas de trabajo o el idioma de vuestros documentos.

Para cada "proyecto", es decir, cada programa que queramos documentar, ha de prepararse un fichero que contendrá las rutas a los inputs y a los outputs, todas las opciones de formateo, ocultación de información, uso de gráficos, idioma, etc. Si ya hemos definido una "norma" para nuestra documentación, podemos utilizar copias de un mismo fichero, modificando solamente la identificación del proyecto (nombre, versión, etc) y las rutas a los inputs y a los outputs.

Supongamos que nuestro fichero de configuración se llama `Doxyfile` (nombre típico y habitual) y que estamos en el directorio donde lo hemos copiado. Si escribimos esto en la línea de comandos:

```
> doxygen Doxyfile
```

el programa irá a la carpeta donde le hemos dicho que está nuestro código C++, lo procesará y creará las carpetas `html` y `latex` en la ruta que le hayamos configurado.

Dentro de la carpeta `html` encontraremos el fichero `index.html`. Lo abrimos con un navegador y veremos la documentación generada. Dentro de la carpeta `latex` encontraremos el fichero `Makefile`. Ejecutando el comando

```
> make
```

se generará el fichero `refman.pdf` con la misma documentación, organizada como informe.

Como ejemplo, estas son la primeras líneas del `Doxyfile` que hemos usado

```
DOXYFILE_ENCODING      = utf-8
PROJECT_NAME           = "Laboratori de PRO2. Exercici de prova del Doxygen."
PROJECT_NUMBER         = "versio oct-2015"
OUTPUT_DIRECTORY       = "./DOC"
...
OUTPUT_LANGUAGE        = Catalan
```

varias líneas más abajo aparece la otra parte que puede ser necesario retocar

```
INPUT                  = "."
INPUT_ENCODING         = utf-8
```

Obviamente, cada uno tendrá que introducir los datos pertinentes de cada proyecto: identificación, carpetas de input y output e idioma base de sus documentos.

Notad que hay dos lugares donde se especifican formatos de caracteres. Hay que tener mucho cuidado de usar los valores adecuados a nuestro entorno de trabajo, ya que de lo contrario pueden salir documentaciones con caracteres ilegibles en lugar de letras acentuadas, ñ, ç, etc. Por ejemplo, si vuestro editor de textos utiliza el formato `unicode`, deberéis usar el juego de caracteres `utf-8`; si utiliza `ascii`, tendréis que usar `iso-8859-15`.

Por último, existe la posibilidad de seleccionar las extensiones y los nombres de los ficheros que `doxygen` ha de procesar dentro de la carpeta `INPUT`: por ejemplo para documentar la especificación de un proyecto basta con tratar ficheros `.hh` o, todo lo más, ficheros `.cc` cuyos nombres respondan a los siguientes esquemas: `pro2*.cc`, `main*.cc`, `principal*.cc`. Obviamente, para

documentar proyectos completos se han de procesar todos los ficheros `.hh` y `.cc`. Veremos como indicárselo a `doxygen` en una sesión posterior.

Observad que la documentación resultante incluye un diagrama modular de toda la solución y otro por cada clase. Dichos diagramas se construyen a partir de los `#include` de los ficheros implicados. Si queremos añadir o eliminar elementos tendremos que usar algún `#include` redundante o decirle a `doxygen` que no procese los `#include` que no deseamos ver reflejados en los diagramas.

4. Ejercicio: añadir un programa principal a la documentación de un proyecto

Siguiendo la instrucciones de la sección anterior, generad una nueva documentación para nuestro proyecto que incluya también el programa principal completo del primer ejercicio de la sesión 8, contenido en el fichero `pro2_s8.cc`. Tanto éste como los ficheros `Lavadora.hh`, `Cubeta.hh` y `Prenda.hh` ya contienen instrucciones para ser procesados por `doxygen`. Notad, por lo tanto, que para usar `doxygen` no es necesario que los programas y las clases sean definitivos, porque de momento no estamos usando los ficheros `.cc` de las mismas. Sí que es deseable que el programa principal junto a los ficheros `.hh` se compile sin errores.

5. Ejercicio: actualizar la documentación de una clase

Siguiendo las pautas del ejercicio alternativo de la sesión 8, modificad la especificación de la operación `completar_lavadora` y actualizad la documentación usando `doxygen`. Examinad el resultado y comprobad que los cambios quedan reflejados correctamente.

6. Apéndice

La “biblia” del `doxygen` es el siguiente sitio web:

`http://www.stack.nl/~dimitri/doxygen/index.html`

Es muy exhaustivo pero puede resultar un poco árido. Buscando en google se pueden encontrar fácilmente ejemplos sencillos, reportes de experiencias, etc.

La lista de palabras reservadas se puede encontrar en

`http://www.stack.nl/~dimitri/doxygen/commands.html`

Programas auxiliares:

- Para que sean visibles los diagramas modulares, se ha de instalar el paquete gráfico `dot`, lo podéis encontrar en `http://www.graphviz.org`.
- Para obtener la versión pdf de la documentación se ha de instalar el sistema de edición `latex` y, en particular, los paquetes `pdflatex` y `epstools`.