Documenting Using Doxygen

Objective

The objective of this activity is to transform our code in a well-documented application. In order to achieve this goal in addition to following the programming style guide introduce before, part of the documentation of the source code should be produced automatically from the source code using Doxygen.

Introduction to Doxygen

Doxygen is a tool to generate documentation from the comments in the source code. The tool supports different programming languages such as Fortran, C, C++, Java, VHDL, PHP, C# or Python, among the most popular.

Including comments that can be processed by Doxygen is performed using a special syntax that includes tags, which are predefined texts with some sense for the tool.

An example of a comment in a file header `.h` is:

```
/**
 * @brief Characters management utility
 *
 * This module contains the prototypes for the characters management functions.
 *
 * @file utilcad.h
 * @author PPROG Professors
 * @version 1.0
 * @date 23-01-2015
 */
```

Comments that are interpreted by Doxygen must start with `/**` and end with `*/`. Doxygen tags begin with `@`, and define fields within the comments that will be processed by the documentation tool. `@brief` tag serves to include a summary of the utility module/file purpose. `@file` indicates the name of the file. `@author` the author names of the authors, `@version` the document version, and `@date` the last modification date.

Functions comments follow a similar syntax:

```
/**
 * @brief compare two char arrays
 *
 * strcmp() compares lexicographically two character arrays s1 and s2.
 *
 * @param s1 first char string to be compared.
 * @param s2 second char string to be compared.
 * @return an integer higher, equal or less than zero
 *       if s1 is greater, equal or small than s2
```

```
 * @author PPROG Professors
 */
int strcmp(const char *s1, const char *s2);
```

The `@brief` tag is used at the beginning of the header files, structures and data types, a apart from the header files comments, for defining a summary sentence for each of them. For functions, `@param` tag describes the input parameters, while `@return` tag describes the return values.

Constants, global variables, enumerated types and data structures are commented in a slightly different way:

```
const double NA = 6.02214179E23; /*!< Avogrado constant */


/**
 * @brief Possible errors list
 */
enum Errors {
    OK,          /*!< No error */
    ERROR_MEM,   /*!< Memory allocation error */
    ERROR_FICH   /*!< File I/O error */
};

/**
 * @brief 2D point
 *
 * This struct defines a point in two dimensions.
 */
typedef struct
{
    float x;       /*!< x coordinate */
    float y;       /*!< y coordinate */
} Point;
```

Note that the individual fields are commented with `/*!< … */`. Keep in mind that Doxygen support several syntax styles, this document introduces only one of the possibilities following a similar documentation to Java javadoc.

Execution

Before running Doxygen it is recommended to create an auxiliary configuration file called `Doxyfile`. To simplify the creation of a configuration file, Doxygen can create a template configuration file for you. To do this, call doxygen from the command line with the -g option:

```
> doxygen -g
```

To generate the documentation you can now enter:

```
> doxygen Doxyfile
```

Using Doxygen

The Doxygen tool includes a fairly comprehensive set of tags. However, as it has been seen, including these labels reduces the readability of the source code, so its moderate use is recommended. The labels included herein are among the most popular[1].

The parts of the source code that makes sense commenting using Doxygen syntax are the public parts, that is, those that correspond to `.h` files and header comments of `.c` files. Its use in the rest of the code is discouraged; in particular, within the code of the functions.

Interesting Links

- http://www.doxygen.nl/manual/index.html
- http://plugins.netbeans.org/plugin/14326/doxygen-integration
- http://doxymacs.sourceforge.net/

[1] In some projects, `@date` and `@version` are omitted as there exists control version tools for generating automatically this information.