



Tema 4. Programación modular y documentación de código fuente

Fundamentos de Programación II


Profesor: Baldomero Imbernón Tudela

Escuela Politécnica Superior
Grado en Ingeniería Informática

Contenidos

- El preprocesador de C
- Programación modular en C
 - Archivos de cabecera y de implementación
- Documentación de código en C
 - Introducción a Doxygen
 - Añadir comentarios al código
 - Generación automática de documentación

Contenidos

- **El preprocesador de C** 
- Programación modular en C
 - Archivos de cabecera y de implementación
- Documentación de código en C
 - Introducción a Doxygen
 - Añadir comentarios el código
 - Generación automática de documentación

El preprocesador de C (i)

- El **preprocesador de C**, previo al compilador, reconoce una serie de órdenes básicas que permiten:
 - Incluir ficheros
 - Definir macros
 - Dirigir la compilación posterior (compilación condicional)
- El preprocesador **tiene su propio lenguaje** independiente del lenguaje C y todas sus órdenes comienzan con el carácter **#**
- Comandos:
 - ❑ **include**
 - ❑ **define**
 - ❑ **ifdef / ifndef / undef**

#include

- **Incluye el contenido de un fichero** en el punto del programa donde se encuentra el comando.
- Normalmente será un fichero de cabecera **.h**, pero puede ser de cualquier tipo.

```
#include <stdio.h>
```

```
#include "sumar.h"
```

- Si el fichero se escribe entre **< >**, el fichero se busca en los directorios prefijados para ficheros cabeceras.
- Si el fichero se escribe entre **" "**, el fichero se busca en el directorio indicado (directorio actual por defecto)

#define (i)

- **Permite definir una constante, con o sin parámetros.**
- Su valor será sustituido textualmente en los sitios donde se encuentre el código.
- Estas constantes se llaman también **macros**.
- **Macro sin parámetros**
 - Normalmente se ponen en mayúsculas, para diferenciarlas de variables
- **Macro con parámetros**
 - Es parecido a una función, pero sigue siendo una constante.
 - Los parámetros no tienen tipo.
 - La expresión se sustituye textualmente en tiempo de compilación.

#define (ii)

MACRO SIN PARÁMETROS

```
#define NOMBRE VALOR
```

```
#define PI 3.1415926
```

```
#define MENSAJE "Mensaje predefinido"
```

MACRO CON PARÁMETROS

```
#define CUADRADO(N) (N)*(N)
```

```
#define MAX(A,B) ((A)>(B)?(A):(B))
```

```
...
```

```
int i, j, k;
```

```
k= CUADRADO(i+1);
```

```
j= MAX(E*PI, k);
```

#ifdef / ifndef / undef (i)

- **Comandos de compilación condicional:** permiten añadir o quitar trozos de código, en tiempo de compilación, según cierta condición.
- La condición es siempre del tipo “macro definido” o “macro no definido”
- Para indefinir un macro: **#undef**

```
#define DEBUG
...
#ifdef DEBUG
    printf("Pasa por aquí.");
#endif
```

```
#ifndef OPTIMIZAR
    i= i + 1;
    j= j*(i+1);
    i= j;
#else
    i= j*= (++i + 1);
#endif
```


#ifdef / ifndef / undef (ii)

- Ejemplo de uso:
 - Evitar la inclusión múltiple de un mismo fichero cabecera

Ejemplo.h

```
#ifndef _EJEMPLO_H
#define _EJEMPLO_H
...
#endif
```

Ejemplo.c

```
#include "Ejemplo.h"
```

- El preprocesador incluirá el fichero “**Ejemplo.h**” en el fichero “**Ejemplo.c**” solamente si no se ha incluido anteriormente.

Ejemplo

```
#include <stdio.h>
#define MENSAJE "Mensaje predefinido"
#define MAX(A,B) ((A)>(B)?(A):(B))


int main (void) {
    int i, j;

    #ifdef DEBUG
        printf("%s\n", MENSAJE);
    #endif

    printf("Introduce dos enteros:");
    scanf("%d %d", &i, &j);
    printf("Maximo: %d\n", MAX(i, j));

    return 0;
}
```


Contenidos

- El preprocesador de C
- **Programación modular en C** 
 - Archivos de cabecera y de implementación
- Documentación de código en C
 - Introducción a Doxygen
 - Añadir comentarios el código
 - Generación automática de documentación

Programación modular

- En C, una aplicación puede estar compuesta por varios **módulos**:
 - Cada módulo ofrece un conjunto de recursos (funciones, tipos de datos, variables, etc.)
 - Los recursos de un módulo pueden ser utilizados por otros módulos.
- Un módulo es un fichero fuente con extensión **.c** el cual puede ser compilado por separado creando un fichero **.o**
- Una aplicación compuesta de varios módulos requiere:
 1. **Compilar todos los módulos**
 2. **Enlazarlos generando el fichero ejecutable .exe** (windows)

Contenidos

- El preprocesador de C
- Programación modular en C
 - **Archivos de cabecera y de implementación** 
- Documentación de código en C
 - Introducción a Doxygen
 - Añadir comentarios el código
 - Generación automática de documentación

Archivos de cabecera / implementación

- Los principios de la programación modular establecen que para cada fichero de módulo se deben crear:
 - **Fichero de cabecera (header): Extensión .h.** Contiene la definición de tipos de datos, declaración de todas la funciones y variables que se hacen públicas al resto de módulos.
 - **Fichero de implementación: Extensión .c.** Contiene la implementación de las funciones declaradas en el fichero de cabecera.
- Los módulos que vayan a utilizar recursos de un módulo determinado podrán hacerlo incluyendo el fichero **.h** asociado.
 - `#include "moduloX.h"`

Ejemplo de programación modular

multiplica.h

```
#ifndef _MULTIPLICA_H
#define _MULTIPLICA_H
int multiplicar (int a, int b);
#endif
```


multiplica.c

```
#include "multiplica.h"
int multiplicar (int a, int b){
    return a * b;
}
```

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "multiplica.h"
int main(void){
    int var1 = 10, var2 = 20;
    printf("var1 * var2 = %d\n", multiplicar(var1,var2));
    return 0;
}
```

Contenidos

- El preprocesador de C
- Programación modular en C
 - Archivos de cabecera y de implementación
- **Documentación de código en C** 
 - Introducción a Doxygen
 - Añadir comentarios al código
 - Generación automática de documentación

Documentación del código

- Para que los módulos de nuestro programa puedan ser empleados, es necesario disponer de:
 - **De la implementación** (ficheros .h y .c)
 - **Una documentación** que describa las funciones
- Vamos a analizar Doxygen
 - Herramienta que facilita la generación automática de código.
 - Válida para múltiples lenguajes de programación como C, C++, etc.
 - <http://www.stack.nl/~dimitri/doxygen/>

Añadiendo comentarios al código

- Doxygen procesa comentarios que comienzan por `/**`
- También procesa los siguientes comandos para describir una entidad (fichero, función, etc.):
 - `\file:` nombre del fichero
 - `\brief` : breve descripción
 - `\author:` indicar los autores
 - `\version:` indicar la versión
 - `\date:` fecha
 - `\param:` para explicar el significado de los parámetro
 - `\return:` explicar el valor devuelto

Ejemplo de documentación de fichero cabecera .h

```
#ifndef __ORDENACION_H
#define __ORDENACION_H
/**
\file ordenacion.h
\brief Archivo con métodos de ordenación iterativos
\author Baldomero Imbernón
\author Andrés Muñoz
\version 1.0
\date 2018
*/
/**
\brief Ordena un array por el método de Selección Directa
\param v El array a ordenar
\param n Tamaño del array (n>0)
*/
void ordenacion(int v[],int n);
#endif
```

Ejemplo de documentación de fichero de implementación .c

```
/**
\file pruebaOrdenacion.c
\brief Prueba de método de ordenación iterativo
\author Baldomero Imbernón
\author Andrés Muñoz
\version 1.0
\date 2018
*/
#include "ordenacion.h"
#include <stdio.h>
#include <stdlib.h>
/**
\brief Prueba de ordenacion(vector,tamaño)
*/
int main()
{
//PRUEBA ORDENACION
...
system("pause");
return 0;
}
```

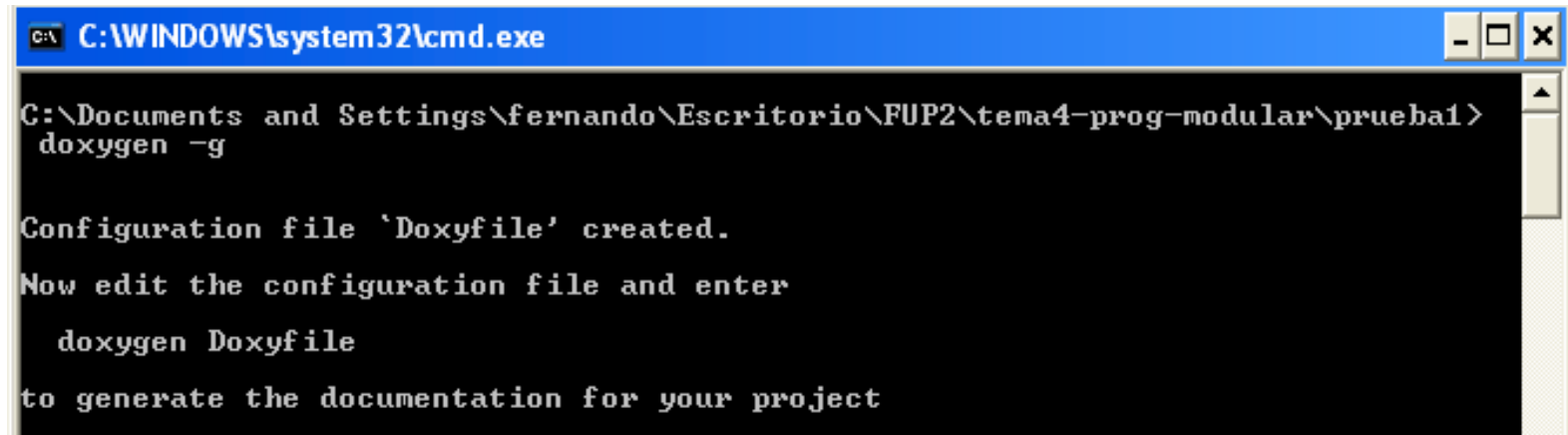
Generación automática de documentación con Doxygen

Pasos a seguir:

1. **Documentar** el código fuente usando el formato Doxygen.
2. **Generar documentación** con Doxygen:
 - Generar un fichero de configuración básico
 - Indicar localización código fuente
 - Configurar idioma
 - Especificar el formato salida
 - Ejecutar *Doxygen* y visualizar la documentación (html)

Generando un fichero de configuración inicial

- Doxygen genera la documentación a partir de un **fichero de configuración**
- Es recomendable generar un fichero por defecto usando el comando: `> doxygen -g`



```
C:\WINDOWS\system32\cmd.exe

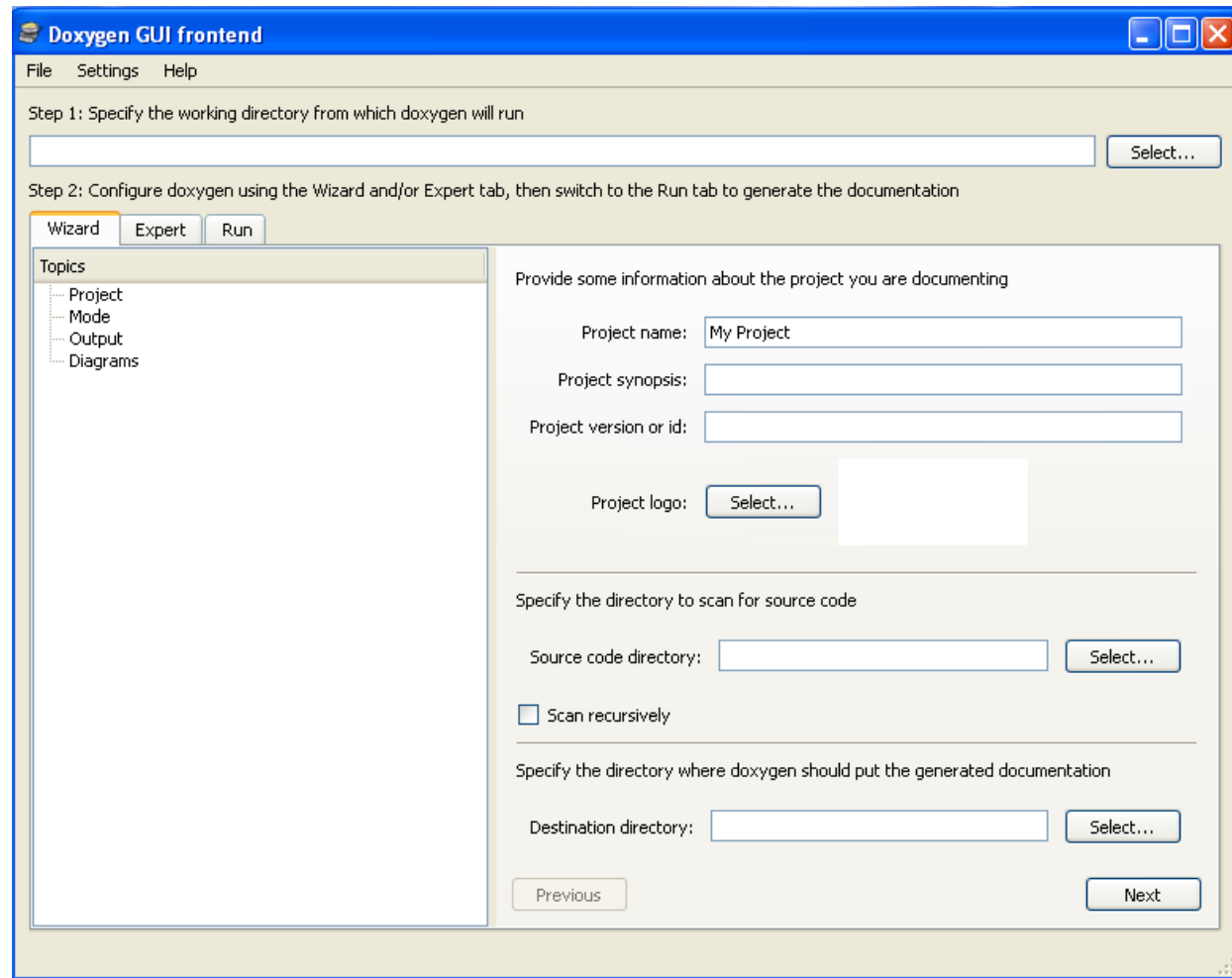
C:\Documents and Settings\fernando\Escritorio\FUP2\tema4-prog-modular\prueba1>
doxygen -g

Configuration file 'Doxyfile' created.
Now edit the configuration file and enter
    doxygen Doxyfile
to generate the documentation for your project
```

- Genera un fichero llamado “**Doxyfile**” con configuración por defecto.

Doxywizard: interfaz interactiva

- El fichero de configuración creado anteriormente se puede editar de forma interactiva a través de *DoxyWizard* (instalado por defecto)
- *File-Open*



Doxywizard: aspectos a configurar (i)

- **PESTAÑA WIZARD (Project)**

- **Project Name:** título de la documentación generada ()
- **Project version or id:** versión de la documentación ()
- **Source Code Directory:** directorio donde se encuentra el código fuente.
- **Destination Directory:** directorio donde Doxygen generará la documentación ()

- **PESTAÑA EXPERT->Project**

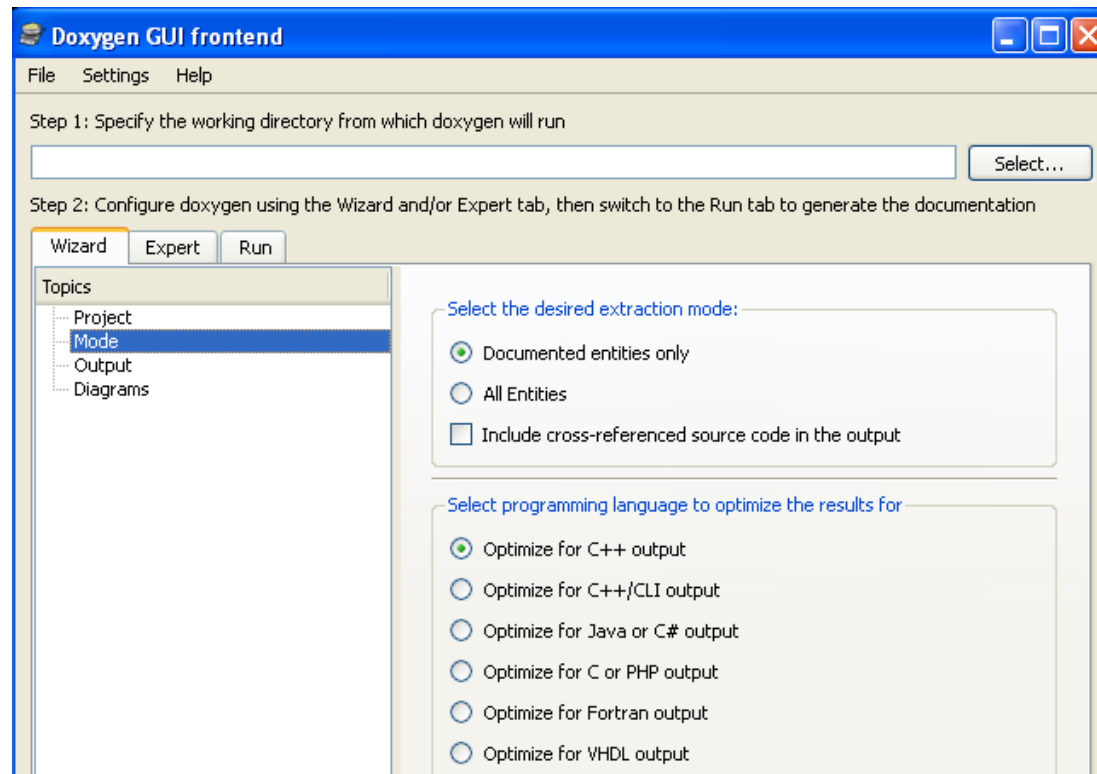
- **OUTPUT_LANGUAGE:** idioma en que se generará la documentación
- **FULL_PATH_NAMES:** desmarcar la opción para que no se muestre el path completo de los ficheros.

- **PESTAÑA EXPERT->Build**

- **EXTRACT_ALL:** Marcar

Doxywizard: aspectos a configurar (ii)

- **PESTAÑA WIZARD (Mode)**
 - Marcar las siguientes opciones:
 - *Documented entities only*
 - *Optimize for C or PHP Output*



Doxywizard: aspectos a configurar (ii)

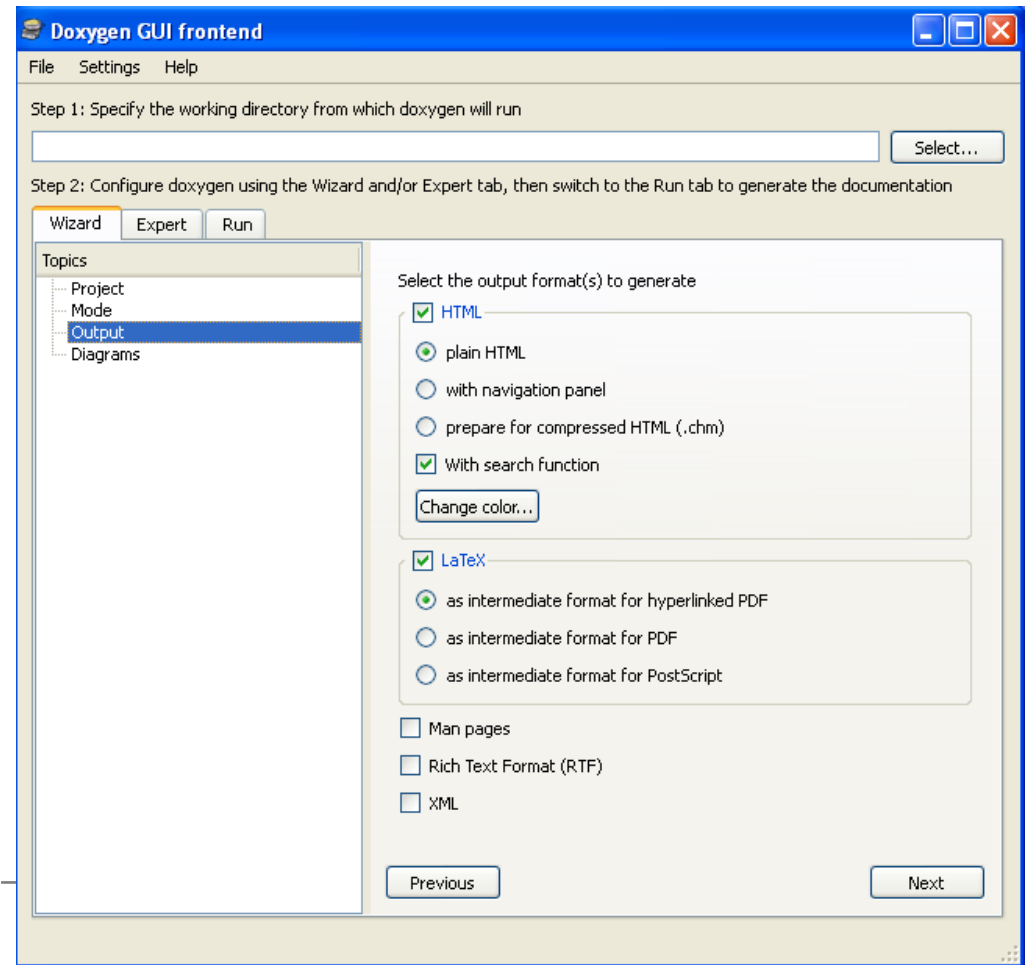
- **PESTAÑA
WIZARD (Output)**

- Marcar la siguiente opción:

- ***HTML***

- **Guardar la configuración**

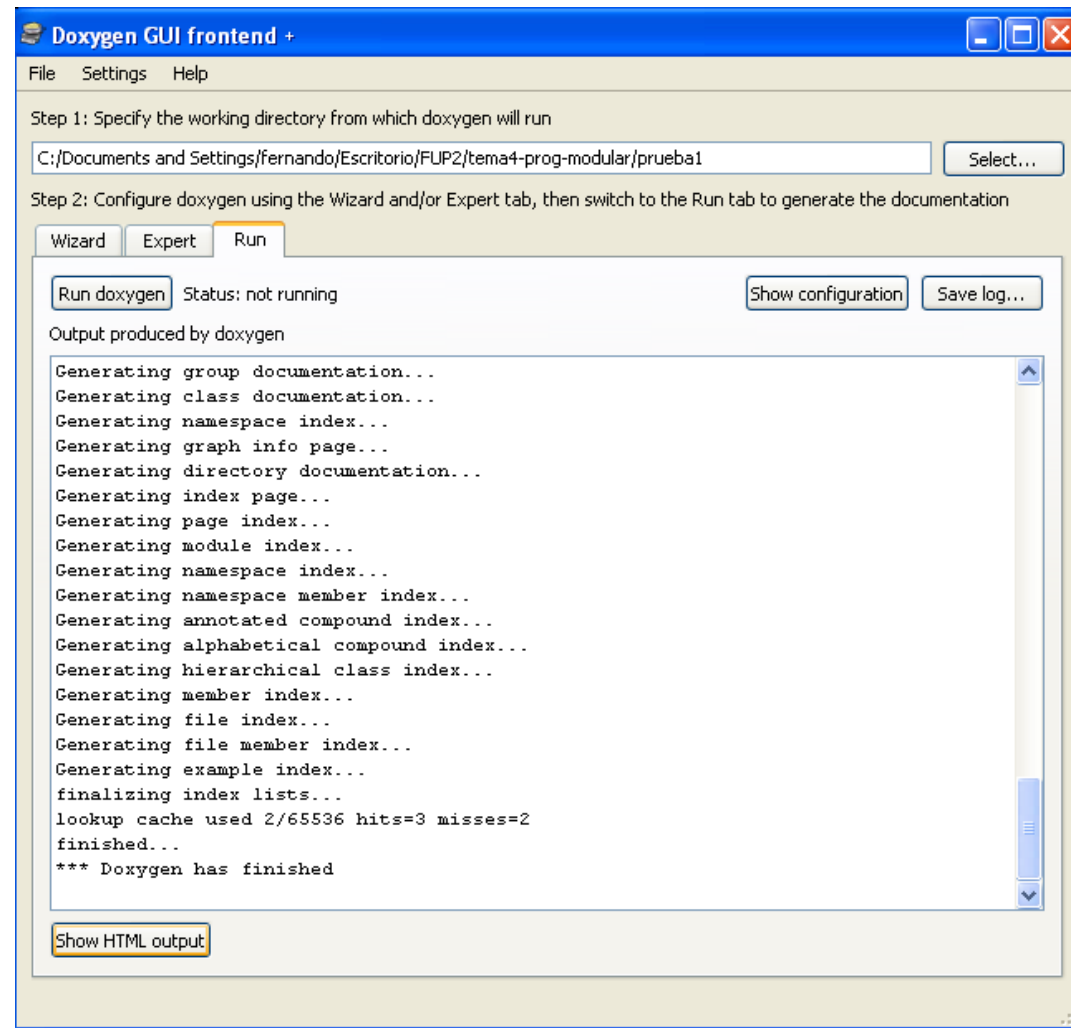
- ***File-Save***



Doxywizard: ejecutar (ii)

• PESTAÑA RUN

1. Generar documentación
 - Run doxygen
2. Mostar documentación resultante:
 - Show HTML output



Documentación

- Comprobar la documentación generada.
- Directorio **html**
- Fichero **index.html**

prueba 1.0

Página principal Archivos Buscar

Lista de archivos Globales

Lista de archivos

Lista de todos los archivos documentados y con descripciones breves:

main.c	Fichero principal de ejecución del programa
multiplica.c	Archivo implementación del módulo de multiplicación
multiplica.h	Archivo cabecera del módulo de multiplicación

Generado el Lunes, 24 de Febrero de 2014 13:39:29 para prueba por **doxygen** 1.8.6

Bibliografía básica

- King, K.N. **C Programming. A modern approach.** 2ªed. Ed. W.W. Norton & Company. Inc. 2008.Chapter 2, 3.
- JOYANES, L. **Fundamentos de programación.** 4ªed. Ed. McGraw-Hill. 2008.Capítulo 3.
- <http://www.stack.nl/~dimitri/doxygen/>
- Documentación de código con C++.
<http://jbgarcia.webs.uvigo.es/asignaturas/FP/index.html>