The ‘other’ aspects of Programming

Ravikishore Kommajosyula

Overview

- Programming is more than writing code
- TODAY: The ‘other’ aspects of programming
- Coding guidelines
- The need for good documentation
- Doxygen
- Concluding Remarks

Coding guidelines

- NEVER re-define standards

```
#define TRUE FALSE //Happy debugging  people
```

Coding guidelines

- NEVER re-define standards

```
#define TRUE FALSE //Happy debugging  people
```

- Industry standard coding guidelines:
  - Google C++ style guide
  - id Software coding style
- Need for standard coding guidelines:
  - Uniform format and look
  - Higher quality code and less misunderstandings
  - Someone else CAN look at your code

Coding guidelines - example

- Indentation: 4 spaces per level
- Each code line at most 100 characters long
- 1 space before and after all operators and after every comma
- Use UTF-8 formatting
- Function names
  - Should be descriptive
  - Use a verb, or a verb followed by a noun
  - Use underscores (_) between words
- Variable names
  - Same as function names
  - Can use underscore (_) before start..

Comments – Why they are important?

- Use comments wisely to give reader information about what is not obvious
- Should enable people to:
  - Re-use your code
  - Understand your code for maintenance and debugging

Comments – Why they are important?

- Some useless / dangerous comments:

```
return 1;  // returns 1
```

```
/** * Always returns true. */
public boolean isAvailable() {
return false;
}
```

```
const int TEN=10;  // Store value of constant 10
```

- Don’t say what you are doing. Say why you are doing it that way

Comments – Why they are important?

- Some funny / yet useful comments:

```
// Dear maintainer:
// Once you are done trying to 'optimize' this routine,
// and have realized what a terrible mistake that was,
// please increment the following counter as a warning
// to the next guy:
// total_hours_wasted_here = 42
```

```
// somedev1 - 6/7/02 Adding temporary tracking of Login screen
// somedev2 - 5/22/07 Temporary my ###
```

```
// Magic. Do not touch
```

```
// drunk, fix later
```

Documentation Generators

- A documentation generator is a *tool* that generates documentation for programmers or end users, from a set of specially commented source codes
- Doxygen is a documentation generator for C, C++, Java, Objective-C, Python and to some extent PHP, C# and D
- Portable (Windows/ Linux/ Mac)

Doxygen

- Doxygen is very useful for creating useful documentation for large projects without additional effort (except the special format commenting)
- Eclipse has plugins that helps in creating Doxygen compliant comments
- Not too much additional work to create Doxygen compliant comments

Doxygen – How to use

The entire process at a glance

- doxygen –g <filename>
- Create the configuration file for the project

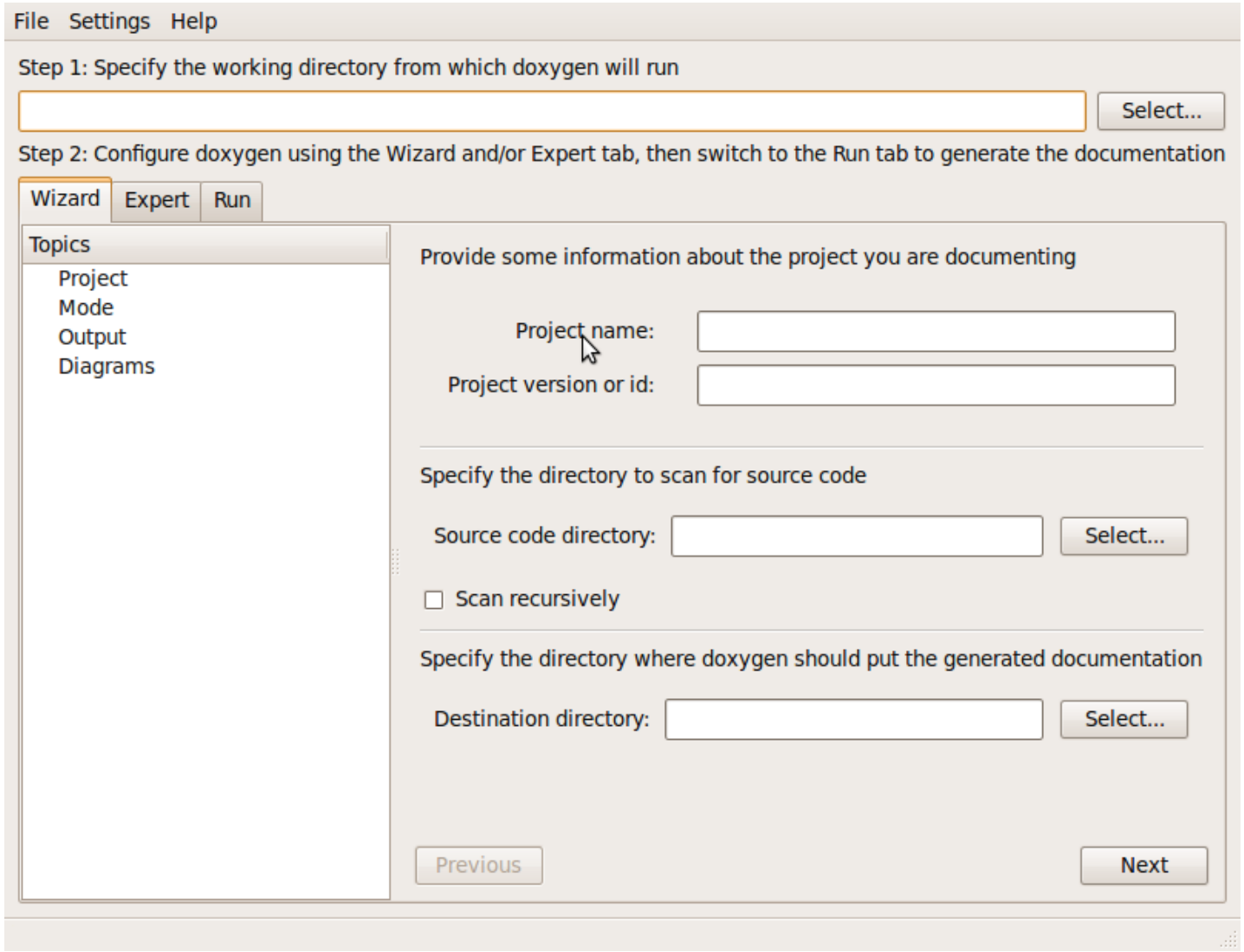
- Special Doxygen compliant comments
- Starts with /** (or) /// (or) /*!

- doxygen <configfile>
- Creates the documentation in specified format (html, pdf, latex)

Doxygen

- Well documented, you just need to fill in the blanks (GUI also available)
- Main things to set:
  - PROJECT_NAME = Adv_Prog
  - OUTPUT_DIRECTORY = ./doc
  - INPUT = ./src ./include
  - FILE_PATTERNS = *.cpp *.hpp
  - GENERATE_HTML = YES
  - EXTRACT_ALL = YES

Doxygen - GUI


File Settings Help
Step 1: Specify the working directory from which doxygen will run
Select...
Step 2: Configure doxygen using the Wizard and/or Expert tab, then switch to the Run tab to generate the documentation
Wizard
Expert
Run
Topics
Project
Mode
Output
Diagrams
Provide some information about the project you are documenting
Project name:
Project version or id:
Specify the directory to scan for source code
Source code directory:
Select...
Scan recursively
Specify the directory where doxygen should put the generated documentation
Destination directory:
Select...
Previous
Next

Doxygen

Beginning of file:

Start of file	/*! \file dpoint.hpp
Brief description	\brief d-dimensional point class
Detailed Description	A d-dimensional point class which is written carefully using templates. It allows for basic operations on points in any dimension. Orientation tests for 2 and 3 dimensional points are supported using
HTML allowed	Jonathan's code. This class forms the building block of other classes like dplane, dsphere etc.
Author	\author Piyush Kumar
Bugs	\bug No known bugs. */

Doxygen

Beginning of function:

Brief description	/*! \brief Prints character ch at the current location * of the cursor.
Detailed Description	* If the character is a newline ('\n'), the cursor should * be moved to the next line (scrolling if necessary). If * the character is a carriage return ('\r'), the cursor * should be immediately reset to the beginning of the current * line, causing any future output to overwrite any existing * output on the line. If backsapce ('\b') is encountered, …
Parameter	\param ch the character to print
Return	* \return The input character */

```
int putbyte( char ch );
```

Doxygen

- Create a main page using *@mainpage*
- Each file/function should have a header block
- Use meaningful names for variables, constants and functions (They come up in the documentation with *@params*)
- To document a class function, you first have to document the class (Hierarchy)
- Doxygen plugins makes your life easy

Doxygen - Demo

- Demo on how to generate documentation using Doxygen for a C++ project with Eclipse
- Reference to Doxygen Documentation (generated using Doxygen) - http://www.stack.nl/~dimitri/doxygen/manual/docblocks.html