CSci-3081W

SUMMARY OF INSTRUCTOR FEEDBACK ON WRITING / TOOLS FOR WRITING DOCUMENTATION

My Feedback on Your Project Update Emails

- Most successful elements:
  - The most inclusion of details rather than empty statements (e.g., everything is on schedule) that I have seen from any class yet! Excellent, extremely important!
  - Overall, also seems to be better grammar, organization, and writing style than I have seen in previous years — well done on these for the most part!

My Feedback on Your Project Update Emails

- Aspects that need more attention / revision:
  - Appropriate balance between past work and future work given that this is supposed to be a “project update” email — suggested ratio, perhaps 75% past, 25% future.
  - Organization, avoid long paragraphs that just list step 1, step 2, step 3…
  - There’s no need to make up imaginary detail to fill space (e.g., details of a company bid for the project, company policies). You have been working on a project for more than a month now, you should have no problem writing a page update about it without making things up.
  - Selection of “what information” to include in each email rather than just rephrasing each sentence from one email to the other.
  - Overall, in your writing to the client, cultivate honesty/integrity and also confidence/credibility, select the details to include based upon these.
    - e.g., how to handle challenges and lessons learned, how to handle difficulties in how your group worked as a team.
    - e..g., how to present your successes, such as “bug free” code.

TOOLS FOR WRITING DOCUMENTATION

From our earlier lecture on coding style…

- Documenting routines. If I had to put this in front of every routine I write, my program would consist of one routine!

```
'*********************************************************************
' Name: CopyString
'
' Purpose:      This routine copies a string from the source
'               string (source) to the target string (target).
'
' Algorithm:    It gets the length of "source" and then copies each
'               character, one at a time, into "target". It uses
'               the loop index as an array index into both "source"
'               and "target" and increments the loop/array index
'               after each character is copied.
'
' Inputs:       input    The string to be copied
'
' Outputs:      output   The string to receive the copy of "input"
'
' Interface Assumptions: None
'
' Modification History: None
'
' Author:       Dwight K. Coder
' Date Created: 10/1/04
' Phone:        (555) 222-2255
' SSN:          111-22-3333
' Eye Color:    Green
' Maiden Name:  None
' Blood Type:   AB-
' Mother's Maiden Name: None
' Favorite Car: Pontiac Aztek
' Personalized License Plate: "Tek-ie"
'*********************************************************************
```

Javadoc (Similar to Doxygen)

- This is more reasonable and still identifies parameters explicitly.
- Maybe a bit more heavy weight than the idea, but you get the huge benefit of the automatic javadoc generation.

```
/**
 * ... <description of the routine> ...
 *
 * @param dataToSort elements to sort in locations firstElement..lastElement
 * @param firstElement index of first element to sort (>=0)
 * @param lastElement  index of last element to sort (<= MAX_ELEMENTS)
 */
public void InsertionSort(
   int[] dataToSort,
   int firstElement,
   int lastElement
)
```

Doxygen

- A tool for auto-generating web-based documentation from C++ source code.
- Can also extract some structure from the code, e.g., superclasses and subclasses to help organize the documentation.
- You'll use this as part of your documentation for Iteration #3.

- www.doxygen.org
- Download from:
  - http://www.stack.nl/~dimitri/doxygen/download.html

Doxygen Hands-on

- http://www.stack.nl/~dimitri/doxygen/download.html
- on debian: sudo apt-get install doxygen
- on mac with brew: sudo brew install doxygen

1. Install doxygen (if not already) and determine the path to the doxygen command line program.
   - On OSX: /Applications/Doxygen.app/Contents/Resources/doxygen
2. Create a new "make doc" rule for your Makefile.
3. Create a simple config file for doxygen, called "Doxyfile":

```
PROJECT_NAME = 3081W FlashPhoto

INPUT = src
OUTPUT_DIRECTORY = web
HTML_OUTPUT = doxygen-html
GENERATE_LATEX = NO
```

Adding Doxygen-Readable Comments

```
/**
  ... comments for doxygen ...
 */
```

or

```
/*!
  ... comments for doxygen ...
 */
```

or

```
///  ... comments for doxygen ...
```

http://www.stack.nl/~dimitri/doxygen/manual/docblocks.html

```
/** This is the main class for testing functionality.
       Blah, blah, blah.  I could go on for lines…
*/
class Test {
  public:
   /// This enum type is really important to document…
   enum EnumType
   {
     int EVal1,     ///<  enum value 1
     int EVal2      ///<  enum value 2
   };

   /// A really important member function
   void member();

  protected:
   /// The variable myVar is used to accomplish XYZ.
   int myVar;
};
```

Creating a Main Project Page

The easiest method is to use “markdown” files.
http://www.stack.nl/~dimitri/doxygen/manual/markdown.html

Create a new file mainpage.md that has:

```
My Main Page     {#mainpage}
============

Documentation that will appear on the main page
```