CISC 2200/5220—Data Structures
Spring, 2016

Using the doxygen Documentation System

There are three steps to using doxygen:

1. Embed doxygen markup into your C++ source code.
2. Configure doxygen.
3. Run doxygen.

Let's look at these in turn.

Note: In what follows, instructions are given for students in CISC 2200. If you're taking CISC 5220, modify accordingly.

Embedding doxygen markup into your C++ source code: This is described pretty well in Chapter 1 of the text, as well as in one of the Appendices. By the way, you know that coding is an iterative process; the same is often true for documentation. It's a good idea to create the (at least the initial version of the) documentation (and related stub versions of the functions you'll need to use) early on, *before* you do the full implementation.

The text's material leaves out one useful feature. If you want to have documentation appearing on the "main page", you'll need to add the @mainpage command to the first stanza. So the beginning of your file will look something like the following:

```
/** @mainpage
 * CISC 2200, Data Structures<br>
 * Project 192: Whatever was the instructor thinking?
 *
 * I can't believe I've spent this much time on this course.
 * But I'd better not complain too much, since this is on a web site.
 *
 * @author Joseph L. User
 * @date 31 January 2016
 * @file proj192.cc
 */
```

Configuring doxygen: Run the command "doxygen -g" from the command line. This will create a configuration file Doxyfile. The default behavior of Doxyfile isn't quite what we want, and so you'll need to make a few changes:

- The line PROJECT_NAME="My Project" should be changed to PROJECT_NAME="CISC 2200 Project 1" (or whatever the project number happens to be).
- The line "OUTPUT_DIRECTORY=" should be changed to

  OUTPUT_DIRECTORY=/u/erdos/students/joeuser/public_html/datastr

  (of course, you should substitute your own login ID for "joeuser").
- The line JAVADOC_AUTOBRIEF=NO should be changed to JAVADOC_AUTOBRIEF=YES.
- If you're doing Project 1, then the line "HTML_OUTPUT=html" should be changed to "HTML_OUTPUT=proj1". (Make the obvious change for other projects.)
- Make sure that the GENERATE_HTML line says GENERATE_HTML=YES.

- The line `GENERATE_LATEX=YES` should be changed to `GENERATE_LATEX=NO`.
- The line `HAVE_DOT=NO` should be changed to `HAVE_DOT=YES`.

You'll also want to make sure that the directory `~/public_html/datastr/proj1` exists. Use the `mkdir` command to create this directory, if necessary; you may need to use the `-p` option (read the `mkdir` manpage for more details).

By the way, it's actually easier to reuse an existing `Doxyfile` for an old project than it is to generate a fresh `Doxyfile` for a new project. It's a matter of copying the old `Doxyfile` into the working directory for the new project[1] and changing the `PROJECT_NAME` and `HTML_OUTPUT` to reflect the new project.

Running `doxygen`: Simply run the shell command "`doxygen`". You can now access the HTML documentation via the URL

`http://www.dsm.fordham.edu/~joeuser/datastr/proj1`

(of course, substituting your login ID for `joeuser`). If you find that you're not happy with the documentation, go back into the relevant file and repair same; you can then re-run "`doxygen`". Eventually, you'll be satisfied with your documentation, which means that you're ready to start coding.

If your appetite has been whetted, go to the URL `http://www.stack.nl/~dimitri/doxygen` for more information about `doxygen`.

[1] Remember, each of your programming projects should be in its own directory. For example, Project 1 should be done in `~/private/datastr/proj1`.