Project #1 – Scope Analysis

due Tuesday, February 11

version 1.3

Purpose:

This project requires you to analyze the scope structure of a set of C++ source code files. You will then use this information to display the size and complexity of each function definition in each file analyzed[1].

You will find code facilities, provided here: http://www.lcs.syr.edu/faculty/fawcett/handouts/CSE687/code/Project1HelpS14, to be useful for implementing the requirements stated below.

Requirements:

Your Scope Analysis project:

1. **shall** use standard C++[2] and the standard library, compile and link from the command line, using Visual Studio 2013, as provided in the ECS clusters and operate in the environment provided there[3].

2. **shall** use services of the C++ std::iostream library for all input and output to and from the user's console and C++ operator new and delete for all dynamic memory management.

3. **(3) shall** identify a set of files for analysis by supplying, on the command line, a path, one or more file patterns, and a switch /s which, if present, indicates that the entire directory tree rooted at the path is searched for matching files. If the switch is not present on the command line only the directory at that path is searched.

4. **(5) shall**, for each scope[4] in every file in the file set, display the scope type with line numbers for the opening brace and closing brace. If the scope type is namespace, class, struct, enum, or function you will also supply the scope name.

5. **(5) shall,** for each function analyzed, create an in-memory tree structure that contains a node for each scope encountered in the function, and display the tree's node count after the output from the previous requirement. At the end of processing each file, **shall** display an XML representation of the tree with the largest number of scope nodes.

6. **(3) shall** provide a command line option, /b, to show for each analyzed function only function size and complexity, where we use the number of scopes defined in the function as its complexity.

7. **(4) shall** provide a test executive package and a display package that, combined with the analysis facility, demonstrates you meet all the requirements of this specification. It is important that your demonstration is accurate, complete, and clear. Your score for meeting requirements will be based on this display. You will get no credit for requirements met but not accurately demonstrated.

We will use the results of this Project to develop a Code Similarity tool in Project #2. Finding parts of source code files that have a high degree of similarity is very useful for code refactoring. The idea is to implement a functionality in only one place so there is only one piece of code to understand and fix if there are any latent errors in its implementation. So, when similar code is detected we can replace the duplicated processing with calls to a single function.

[1] We will use this analyzer in Project #2 to detect similarities between C++ source code packages.
[2] This means, for example that you may not use the .Net managed extensions to C++.
[3] VC++ 2012 is available in all the ECS clusters.
[4] For this project we will define scope as being determined by an opening brace "{" and its matching closing brace "}". One can argue that an "if", "while", "else", "for", or "catch" with a single line expression is a new scope – this is true. But, to keep things simple you are not required to detect the "braceless" scopes. If that offends your coding principles you may count them as additional scopes PROVIDED THAT YOU ANNOUNCE THAT IN ONE PROMINENT PLACE IN YOUR PROJECT OUTPUT AND COUNT CORRECTLY. Note that we count the "if" and "else" statements with braces as two distinct scopes.