Commenting standards for the BenchmarkXPRT benchmarks

Introduction

This document summarizes the coding guidelines for the BenchmarkXPRT benchmarks. The goal of these guidelines is to make the code easier to evaluate without imposing an undue burden on the developer.

The XPRTs are written in a number of languages. The list includes C, C++, C#, Java, JavaScript, HTML5, XML, and more. Each one of these languages has its own set of best practices and styles. Therefore, this document does not discuss the particulars of coding style in any detail. We do assume that any code you submit for the benchmarks will use best coding practices for the language in which it is coded. We also assume that you will follow the universal principles for making the code as readable as possible, such as having descriptive variable names and having clear and consistent indentation.

Commenting style

As noted above, the benchmarks are coded in a number of languages. For consistency, the examples in this document are in C#. However, you should format the comments as appropriate for the language in which you are coding.

All comments should be in English.

The comments are aimed at other programmers. Therefore, you should assume the reader has a base level of technical competence. Comments should explain the purpose of the code, rather than restating the code itself. For example, this comment for a line of C# states the obvious:

```
// Zero the high order bits.
x = x & 0x00FF;
```

A better comment would be:

```
// High order bits are not used. This will simplify later comparisons.
x = x & 0x00FF;
```

Remember that your comments will be seen by other people. Avoid obscenity and insulting remarks. For example, a bad comment would be:

```
// If the user is stupid enough to enter a negative number
If (i < 0)
```

Don't write more in comments than is needed to convey the idea. Avoid jokes, quotes, and overly wordy explanations. Keep the comments simple and direct.

Comments should be up to date. When the code changes, comments should be updated at the same time.

File comments

Every source file must have a copyright header. Source files include the actual code, the include files, and the text-based file that the compiler uses to compile the application. The header will contain the following text, formatted as appropriate for the language in which you are coding:

```
Copyright (C) 2014 BenchmarkXPRT Development Community
Licensed under the BENCHMARKXPRT DEVELOPMENT COMMUNITY MEMBER
LICENSE AGREEMENT (the "License"); you may not use this file
except in compliance with the License.  You may obtain a copy of
the License by contacting Principled Technologies, Inc.
Unless required by applicable law or agreed to in writing,
software distributed under the License is distributed on an "AS
IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express or implied.
See the License for the specific language governing grants and
restrictions under the License.
```

In addition, every source file should have a header containing the following information:

- The name of the file and its overall purpose
- The names of the authors
- The name of and a brief (one- or two-sentence) description for every externally visible or persistent entity in the module. This includes, but is not limited to, static variables, global variables, macro definitions, functions, classes, methods, and so on.
- Any other pertinent notes

Code comments

Every entity that can be called, invoked, or expanded in-line should have a header. These include, but are not limited to, subroutines, classes, methods, and macros. The header should contain the following information:

- Its name and purpose
- An explanation of any input parameters
- An explanation of any output parameters or return values
- An explanation of any external dependencies it may have, such as global symbols, external libraries, shared memory, and so on

Any block of code more than a few lines long should have a brief header explaining what it does. Blocks of code include, but are not limited to, loops, if-then constructs, and sections of code that have a local scope.

A static/global variable should have a comment that explains its purpose and scope.