CODING STANDARD
FOR COMPUTER PROJECTS

When you convert a design document into source code, one of your primary goals should be to write the source code and internal documentation in such a way that it is easy to verify that the source code conforms to the design, and that it is easy to debug, test, and maintain the source code. The coding conventions listed below are designed to assist you in achieving this goal.

1. File Prologue

Each source code file will contain an introductory block of comments that identifies the author and assignment. For example:

```
/*************************************************************************
  John Doe
  Computer Project #5
*************************************************************************/
```

2. Function Prologue

Each function will contain an introductory block of comments that explains the purpose of the function and describes all information passed to the function (parameters and global data objects). For example:

```
/*-----------------------------------------------------------------------
  Name:  celsius2fahrenheit

  Purpose:  Convert a temperature from Celsius to Fahrenheit
  Receive:  A temperature in degrees Celsius
  Return:   The equivalent temperature in degrees Fahrenheit
-----------------------------------------------------------------------*/
```

3. Mnemonic Identifiers

Meaningful identifiers that are taken from the problem domain will be used for all names (data objects, functions, and programmer-defined data types). In addition, a name will not be used for more than one purpose. For example:

```
double fahrenheitTemp;   // Temperature in degrees Fahrenheit
double celsiusTemp;      // Temperature in degrees Celsius
```

4. Symbolic Constants

Symbolic constants will be used instead of embedding arbitrary numeric and character constants in the body of functions. For example:

```
const double HEAT_OF_FUSION = 79.71;   // Calories to melt one gram of ice
```

5. Descriptive Comments

Comments that describe the functionality of blocks of code will be included within the body of each function. Comments are particularly important before all blocks of code that perform major data manipulations or error processing. For example:

```
// Convert a temperature from the Fahrenheit scale to the Celsius scale

celsiusTemp = (fahrenheitTemp - 32.0) / 1.8;
```

6. Source Code Format

All source lines will be less than 80 characters in length.

Only one statement will be included on a given line of the source code.

Blank spaces and blank lines will be used to enhance the readability of both source statements and comments.

In addition, all C/C++ statements will be indented to show levels of nesting. Normally, two, three or four spaces are used for indenting. Examples of two styles of indenting are given below.

```
while (expression)
{
   statement;
      .
      .
   statement;
}
```

```
while (expression) {
   statement;
      .
      .
   statement;
}
```

Other control structures will be indented in a similar fashion. A single style of indentation will be chosen and used consistently within a program.

Control structures will not be nested too deeply. If the clarity of the source code is impaired by too many levels of nesting, the statements that are nested most deeply will be removed and placed in a separate function.