C++ Style Guidelines

Introduction

This document contains a set of guidelines when programming in C++ for any classes you take from me. In general, the style notes are fairly well accepted, so there is nothing strange or radically different from what you'll see elsewhere. On the other hand, I realize different people and organizations have some differences. You should follow the guidelines put forth in this document for my classes and then where you have style preferences that are different, feel free to use those outside of this class.

Naming Guidelines

Variables

- All names should be descriptive of what value(s) the variable will store. Even loop indexes should be description, don't use indexes like: i,j,k; those are holdovers from FORTRAN and we aren't programming in FORTRAN.
- Two styles are accepted for capitalization:

```
string PartNumber
string partNumber
```

- Begin each new word with a Capitol letter; don't capitalize the whole name
- Don't use underscores

Functions (and Class members)

- Naming and capitolization style the same as variables
- Don't use underscores
- Get/Set type functions should follow the pattern: `getMethod/setMethod`

Constants

- Use all CAPS and underscores to separate words. For example: `MAX_VALUE, NUMBER_STUDENTS`
- Related groups of constants should have similar headings. For example: `FIELD_FIRSTNAME, FIELD_MIDDLENAME, FIELD_LASTNAME`

Use of Braces

- Braces should always, without exception, be placed on their own line. This rule follows for functions, structures, classes, or any other C++ structure. Some examples follow:

```
for (int city=0; city<CityCount; city++)
{
      //
      // Code is inside here
}

while (isActive)
{

}

struct MyStruct
{
      int Count;
      float Result;
}

void MyFunction()
{
      //
      // Code goes here
}

class CMyClass()
{
      //
      // Declaration goes here
}
```

- Braces should be one level on indentation less than the code it surrounds, for the most part. When writing code for a function, put the brace at the same level of the function name, for variable declarations they may optionally be indented with the code. An example is provided below:

```
void ExampleFunction()
{
int ThisVariable;
float AnotherVariable=0.0;

      for (int count=0; count<MAX_COUNT; count++)
      {
            AnotherVariable+=1.12;
      }

}
```

Comments

- Each function/member should begin with a comment section providing a brief description of what the purpose is. Don't write an essay, be brief, but descriptive.
- Each class should begin with the same kind of comment as noted for a function/member.

- Please use, exclusively, the C++ comment operator, i.e. //
- My personal commenting style looks like the following:

```
////////////////////////////////////////////////////////////////////
//
// This comment describes what this function does.
// All it does is demonstrate how I prefer to comment
//
void CommentDemo()
{

      //
      // This loop demonstrates how I like to see comments placed
      // before code
      for (int count=0; count<MAX_COUNT; count++)
      {
            //
            // Some kind of looping operation would normally go here
      }
}
```

Misc

- The main thing to keep in mind when coding is to write code that is obvious. Don't try to write code that looks cryptic, if a simpler, more readable, notation provides equivalent results, use the simpler notation. When you write code, think in your mind that someone else will have to look at your code, in this case, a grader. Don't make the grader guess at your code.
- *Magic Numbers*: These are hard-coded numbers sprinkled throughout code, don't do this ☺ Instead, use constants to define any hard-coded numbers