C++ Coding Style Rules

- Align all curly braces vertically to allow easy visual matching.
- Use four (4) spaces for each indentation.
- Indent all statements contained within compound statements (if, for, while, {}, etc).
- Use meaningful names for variables, parameters, subprograms, and classes. (You may use simple names like i, c, s, f, or p for general integers, characters, strings, floats, or pointers respectively.)
- Use the following conventions for identifiers:

Identifier	Convention	Examples
class names	capitalize first word, capitalize all other words (no underscores)	Circle, FilledCircle, ShadedFilledCircle
class member names	lowercase first word, capitalize all other words (no underscores)	draw(), drawLine(), centerX, centerY, radius
function names	lowercase first word, capitalize all other words (no underscores)	findIndex(), factorial(), toLowerCase(), isUpperCase()
named constants	all caps, use underscores to separate words	PI, MAX_BUFFER_SIZE

- Use whitespace around every binary operator (e.g, x + y instead of x+y) and around function arguments (e.g, put(x, y) instead of put(x,y)).
- Keep subprograms simple. At most five simple statements or one compound statement and two simple statements.
- Keep compound statements simple. At most three simple statements per statement list.
- Avoid over parenthesization, e.g, return x + 1; instead of return (x + 1);
- Comment each function, each class, and each class member (describing it's purpose) using a form similar to the example at the very end. (Note I only commented a few members to avoid cluttering up the Class example.)

Examples of Subroutines

A function

```
int factorial( int n )
{
   if ( n <= 0 )
      return 1;
   else
      return n * factorial( n - 1 );
}
```

A procedure

```
void print( int n )
{
   for ( int i = 0; i < n; i++ )
      cout << n << ' ';
   cout << endl;
}
```

An operator

```
ostream & operator << ( ostream & out, Circle c )
{
   return out << "Circle is " << c.radius;
}
```

The main routine and it's arguments

```
#include < iostream.h>

int main()

{
```

```
    while ( true )
    {
        cout << "this is the class that never ends,\n";
        cout << "yes it goes on and on my friend,\n";
        cout << "some students started taking it not knowing what it was,\n";
        cout << "and they'll continue taking it forever just because\n";
    }
    return 0;
}
```

Examples of statments

simple if/else

```
if ( x < y )
    y = 0;
else
    x = 1;
```

cascaded if/else

```
if ( var1 < var2 )
{
    var1 = ...;
    var2 = ...;
     ...
}
else if ( var1 == var2 )
{
    var1 = ...;
    var2 = ...;
    ...
}
else
{
    var1 = ...;
    var2 = ...;
     ...
}
```

switch statement

```
switch ( factorial( i ) )
{
    case 1:
    case 3:
    case 5:
        cout << "It's odd!\n";
        break;
    case 2:
    case 4:
    case 6:
        cout << "It's even!\n";
        break;
    default:
        cout << "It's less than 1 or greater than 6!\n";
        break;
 }
```

loop statements

```
for ( int i = 0; i < MAX_VAL; i++ )
{
    ...
    ...
```

```
}



while ( !cin.eof() )
{
    cin.get( ch );
    cout.put( ch );
}

do // read the first line (including the newline) only
{
    cin.get( ch );
    cout.put( ch );
} while ( ch != '\n' );
```

Examples of Class Definitions

```
/*
   classes Shape, Point, and Rectangle define a class hierarchy
   for geometrical shapes.
*/
class Shape
{
public:
    // draws this shape on the display
    virtual void draw() = 0;
    // inverts this shape along a horizontal line
    virtual void flip() = 0;
    // rotates this shape by number of degrees specified by angle
    virtual void rotate( int angle ) = 0;
};

class Point
{
public:
    // the origin of this point in 2-D space
    int x, y;
    Point( int newX, int newY )
        : x( newX ), y( newY )
    {
    }
};

class Rectangle
    : public Shape
{
private:
    Point tl;
    Point br;
    int angle;

protected:
    ...

public:
    Rectangle();
    virtual ~Rectangle();
    ...
```

```
    void draw();
    void flip();
    void rotate( int angle );
    float area();
};
```

Examples of REALLY BAD commenting

All the following comments are poor. Either they state the obvious about the programming language (typical from programmers just learning the language), or they don't add any information about the item they are commenting and the program would actually be more readable if they were deleted. The final offense is pointing out that a particular close brace matches some open brace. If you find yourself needing this, either you aren't aligning your braces properly, or your code is getting too complex.

```
/*
   This is a class defintion for Shape.
*/
class Shape
{
public:
    // These are pure virtual functions
    // draws shape
    virtual void draw() = 0;
    // flips shape
    virtual void flip() = 0;
    // rotates shape
    virtual void rotate( int angle ) = 0;
}; // end of Shape

// A Point class
class Point
{
public:
    // Two integers
    int x, y;
    // a constructor
    Point( int newX, int newY )
        : x( newX ), y( newY )  // construct x and y
    {
      // do nothing here
    } // end of Point
};

class Rectangle
    : public Shape // Rectangle is publically derived from Shape
{
private: // some private members
    Point tl;
    Point br;
    int angle;

protected: // some protected members
    ...

public: // the public members
    Rectangle(); // constructor
    virtual ~Rectangle(); // destructor
    ...
```

```
    void draw(); // some member functions
    void flip()
    {
      if (x != y)
      {
         swap(x,y); // swap the values of x and y
         flip(); // flip again
      } // end if
    } // end flip()
    void rotate( int angle );


  float area();
  }; // end Rectangle
```

Thanks To: http://doc.ece.uci.edu/~klefstad/s/C++-style.html

Courtesy: Fareed ul Hassan Baig