Style Manual for Cosc-220
Spring 2003

Thomas A. Anastasio

February 4, 2003

1 Documentation

All program files are to be headed by comments giving the file name, its purpose, the author's name, the creation date, and the date of last revision. For example:

```
/*
 * foo.cpp
 * Definition of the foo class.
 * Author: Joe Blow
 * Created: February 2, 2003
 * Current: February 2, 2003
 */
```

Every function declaration is to be documented with comments giving the function's purpose, a description of the parameters, a description of the return value, any pre-conditions and any post-conditions. Example:

```
// cubeIt
// Prints and returns the cube of an integer
// Parameter: x is the integer to be cubed
// Returns: the cube of x
// Preconditions: x must be positive
// Postconditions: the cube of x is printed to cout
int cubeIt(int x);
```

It is not necessary to document the implementation of a function. Generally, comments should be minimized in a program. Well-named functions and variables and good structuring go a long way toward making a program "self-documenting." Do comment when clarification of an obscure algorithm is necessary.

2 Namespaces

- "using namespace std;" is called a namespace *directive*.
- "using std::XYZ;" is called a namespace *declaration*.
- `std::XYZ` is called a *fully-qualified* name.

Namespace directives and namespace declarations are **NEVER** to be used outside of a block. Fully-qualified names may be used anywhere.

> **If your code has "`using namespace std;`" or "`using std::XYZ`" outside of a block (curly braces), then it does not meet the style requirements of this course.**

Here are the guidelines for namespace std (and all other namespaces). Of course, it is still necessary to `#include` the appropriate standard header files such as `iostream` and `string`.

* Inside a block, if a name from std appears just once, qualify it with `std::` In the following example, cout occurs just once so it is qualified as `std::cout`

```
void printIt(int x)
{
  std::cout << "x = " << x;
}
```

* Inside a block, if a name appears more than once and it is the only name from a given namespace, use the "using" declaration specific for that name. In the following example, cout is used more than once so the declaration `using std::cout` is used.

```
void printIt(int x)
{
   using std::cout;

   cout << "x = ";
   cout << x;
}
```

* Inside a block, if there is more than one occurrence of name(s) from a given namespace, use the "using" directive. In the following example, `cout` and `endl` are both used so there is more than one occurrence of names from the `std` namespace.

```
void printIt(int x)
{
   using namespace std;

   cout << "x = " << x << endl;
}
```

* Outside a block qualify every name. For example, function prototypes occur outside a block. In the following, the `string` type (a standard namespace name) is qualified with `std::string`

```
void printString(std::string str);
```

3 Guarding header files

Every header file is to be guarded. To guard a file use compiler directives `#ifndef`, `#define`, and `#endif`. For example:

```
#ifndef FOO_H
#define FOO_H

the usual contents of foo.h

#endif
```

4 Program organization

* class declarations are to appear in guarded header files, one class to a file. If the class name is `foo`, the file is to be `foo.h`

* class definitions (implementation) are to appear in `.cpp` files, one class to a file. If the class name is `foo`, the file is to be `foo.cpp`

* A main function is to occupy its own implementation file, named appropriately.

* Free functions (non-member functions) are to be declared in an appropriately named, guarded header file and implemented in a `.cpp` file. For example, helper functions for a program might be declared in a file named `progHelper.h` and implemented in a file named `progHelper.cpp`