

The idea with program style is to have a *readable* program. Poor style can lead to programs which are error-prone, difficult to read, and difficult to understand. The purpose of this recommended style is not to force you into a certain way of programming, but rather to give you a model of good style from which you can evolve your own style. If the word **must** appears in a style description that specific style will be required in *all* programs. In the other cases, if you don't use the recommended style it is up to you to use some other *good* (judged by the instructor) style. Be sure when adopting a programming style that it is not only readable, but also consistent over all statement types.

1. Every program must have a header as follows:

```
/******  
*                                                                 *  
* Programmer:           Tony Toledo                             *  
* Course/Lab Section:  CSCI203-1                               *  
* Date:                 9/29/02                               *  
*****
```

```
*
* Problem Statement: This program .....
*
*****/
```

The **Comment/Uncomment Block** feature of emacs makes this very easy to do.

2. Each logical segment of code must be preceded by a descriptive comment and a blank line.
3. Operators such as =, +, <, etc., must be surrounded by blank characters. For example:

```
x = x + 1;
```

4. Object and function names must be meaningful and start with a lower case letter. Each word in the name should begin with an upper case letter. Here is an example: numItems.

5. You must indent C++ statements that contain braces. Here is the recommended style.

```
for (int i = 0; i < n; i++) {  
    S1;  
    S2;  
}
```

```
while (i < n) {  
    S1;  
    S2;  
}
```

```
if (a < b + 1)  
    S1;  
else {  
    S2;
```

```
        S3;
    }

    if (a > b) {
        S1;
        S2;
    } else {
        S3;
        S4;
    }

    switch (x) {
    case 1:
        S1;
        break;
    case 2:
        S2;
```

```
        break;
case 3:
    S3;
    break;
default:
    S4;
    break;
}
```

6. Functions must be preceded by a comment block that states its purpose, pre-conditions, and post-conditions.
7. When possible, use assertions to verify pre-conditions and post-conditions.