Coms 262 Program Style Guidelines Spring 2013

Comments

Block comments

Block comments should be placed at the beginning of each file and at the beginning of each function. Block comments should be surrounded by a box made with asterisks as a /*-style comment.

```
/*************************************************************************
*                                                                        *
*   Your comment goes here                                               *
*                                                                        *
*************************************************************************/
```

The first and last asterisks of the intermediate lines should be separated by 18 tabs. Do not use blank spaces. This guarantees that the box will fit nicely on a page when the file is printed. Furthermore, it acts as a guide; if any line exceeds the length of the box by more the a few characters, it will run off the right side of the page when the file is printed. The appropriate comments should be placed inside the box.

For a file header, the comments should include

- The name of the file
- The author of the file
- The date the file was created
- A brief description of the purpose of the file
- The Honor Code

```
/*************************************************************************
*                                                                        *
*   Program:    StockMarket.cpp                                          *
*                                                                        *
*   Author:     Robb T. Koether                                          *
*                                                                        *
*   Date:       Aug 28, 2012                                             *
*                                                                        *
*   Purpose:    This program will predict the value of the DJIA for      *
*               any given date in the future                             *
*                                                                        *
*   Pledge:     On my honor, I have neither given nor received any aid   *
*               on this work, nor am I aware of any breach of the Honor  *
*               Code that I shall not immediately report.                *
*                                                                        *
*************************************************************************/
```

For function headers, the comments should include

- The name of the function, including its signature
- A brief description of the purpose of the function

```
/*************************************************************************
*                                                                        *
*   Function:   DJIA(Date)                                               *
*                                                                        *
*   Purpose:    This function will return the DJIA for the given date    *
*                                                                        *
*************************************************************************/
```

Major comments

Major comments are used within a function to visually divide the function into logical segments. The major comment should describe the purpose of the next segment of the program. A major comment is a `//`-style comment. It should begin one tab-stop to the left of the current level of indentation. Type `//`, followed by a tab, followed by the comment. These comments should occur roughly every 4 to 10 lines throughout the function. If you find that you have written more than 10 lines without a major comment, then you probably ought to see if you can break it up into two or more segments.

Standard places to write major comments are immediately before

- Decisions
- Loops
- Sequences of 2 or more statements

In-line comments

In-line comments `//`-style comments. They are written on the same line as a program statement. An in-line comment should describe only the program statement on that line. The comment should be kept short, limited to one line.

Within a function, all in-line comments should be aligned by using tabs. Ideally, every line of a function would have an in-line comment. However, in practice they are used mostly only in statements which are not self-explanatory.

Indentation

The statements within a program block, delimited by braces {}, should always be indented one addition tab stop (4 spaces) from the current level of indentation. This applies to functions, which are always written within braces.

Other places where braces are commonly used are

- `if` statements
- `switch` statements
- `while` statements
- `for` statements
- `do` statements

In each of these types of statements, if the block consists of a single statement, then the braces are not required. However, the indentation rule should still be observed.

Excessive indentation

Excessive indentation should be avoided. If a line is indented too much, then it quickly will run off the right side of the screen. When the file is printed, the full statement will not appear on the paper.

Normally, a program block should not be indented more than 3 tab stops from the left margin. In no case should it be indented more than 5 tab stops. That much indentation indicates that the logic has become sufficiently complex that the function should be decomposed into two or more functions.

Identifiers

General remarks

Identifiers should be meaningful. The name should suggest what the identifier represents. Avoid excessive abbreviation, but also avoid excessively long names.

Constant names

The name of a constant should be written in all uppercase. If the name consists of two or more words, then the words should be separated by underscores. For example, if a constant represents the maximum length of a string, then the identifier `MAX_LENGTH` would be appropriate.

Object (variable) names

A variable name should begin with a lowercase letter. If the name consists of two or more words, then follow one of two conventions. Either each subsequent words begin with an uppercase letter, or underscores are used to separate the words. An example would be `numberOfStudents` or `number_of_students`.

Variable names should be meaningful. The name should suggest the what the variable represents. Avoid excessive abbreviation, but also avoid excessively long names.

Function names

The convention for naming functions is the same as the convention for objects.

There are further guidelines for member functions of classes. If the function is an inspector, then its name should be either the name of the attribute that it returns, or the word `get` followed by the name of the attribute. For example, an inspector that returns the length of a `Rectangle` object should be named `length()` or `getLength()`.

If the function is a mutator, then its name should be the word `set` followed by the name of the attribute that is being modified. For example, a mutator that sets the length of a `Rectangle` object should be named `setLength()`.

If the function is a facilitator, then its name should be the standard name of the operator that it is facilitating. For example, a facilitator that will facilitate the operator `+` should be named `add()`.

If the function returns a boolean value (`true` or `false`), then its name should begin with `is`. For example, if a function decides whether a value is valid, its name might be `isValid()`.

Class names

Class names should begin with a capital letter. If the class name consists of more than one word, then each word should begin with a capital letter. For example, a list class should be named `List` and a sorted list class should be named `SortedList`.