CMPE 1666- Guidelines for Commenting Code

Comments

- Commenting involves placing **Human Readable Descriptions** inside of computer programs detailing what the **Code** is doing.
- Proper use of commenting can make code maintenance much easier, as well as help finding bugs faster.
- Further, commenting is very important when writing functions that other people will use.
- Remember, well documented code is as important as correctly working code.

Courtesy: https://www.cs.utah.edu/~germain/PPS/Topics/commenting.html

Comments

- All programs should be commented in such a manner as to easily describe (in easily understood terms) the purpose of the code and any algorithms used to accomplish the purpose.
- A user should be able to utilize a previously written program (or function) without ever having to look at the code, simply by reading the comments.
- Commenting is the "art" of describing what your program is going to do in "high level" statements.
- Commenting is best done **before** actually writing the code for your program.

Courtesy: https://www.cs.utah.edu/~germain/PPS/Topics/commenting.html

Comments

- Comments are specially marked lines of text in the program that are **not evaluated**.
- There are usually two syntactic ways to comment.
- The first is called a *single line comment* and, as implied, only applies to a single line in the "source code" (the program).
- The second is called a *Block* comment and refers usually refers to a paragraph of text.
- A block comment has a start symbol and an end symbol and everything between is ignored by the computer.

Where to Comment:

- Comments should occur in the following places:
  - The top of any program file. This is called the **"Header Comment"**. It should include all the defining information about who wrote the code, and why, and when, and what it should do. (See File Header Comment below)
  - Above every function. This is called the function header and provides information about the purpose of this "sub-component" of the program.
  - When and if there is only one function in a file, the function header and file header comments should be merged into a single comment. (See Function Header below)
  - In line. Any "tricky" code where it is not immediately obvious what you are trying to accomplish, should have comments right above it or on the same line with it.

How not to comment:

- Comments should be useful high level descriptions of what the program is doing.
- They should not restate something that is "obvious".
- By using appropriate variable names, much of a program can be (almost) as easy to read as English.

Example of Poor Comments

```
int x = 5; // this sets x to 5
int y = 2 * x; // here we double the value of x and save it in the variable  y
int average = (x + y) /2; // compute the average by dividing the sum by the  number of values....
```

How to comment Code:

- Primarily, a single "block" comment should be placed at the top of the function (or file) and describe the purpose the code and any algorithms used to accomplish the goal.
- In-line comments should be used sparingly, only where the code is not "self-documenting".
- When you do use "in-line" comments, you should place them before (or next to) any code that is not self explanatory.
- This comment should detail the "idea" behind the code and what is to be accomplished.
- It may also say how this is to be accomplished if the "algorithm" is complex.

Example of In Line Comments

- In line comments are those that are found in the general body of the program.
- They are usually very short (a line or two) comments making a "note" about what is going on.
- In line comments are usually made using the "single line" commenting syntax of the language.
- You should add "in-line" comments wherever you think a little bit of English explanation would be useful to either yourself or someone else (like a TA) who is going to read your code.
- Such in-line comments should be used whenever the code is not "transparent" (easy to follow simply from the names of the variables).

```
float
solve_quadratic_equation(int A, int B, int C)
 {
    return (-B + sqrt(B*B - 4*A*C)) / (2*A); // NOTE:  we only return the positive value
 }
```

Self Documenting Code

- Self documenting code uses well chosen variable names (and function names) to make the code read as close to English as possible.
  - This should be your goal.
- For example, naming a variable g has little meaning, but naming a variable gravity gives a much better description of what the variable should contain.
- By using proper variable and function names, you should minimize the amount of "external" documentation that is necessary. For example, compare the following two pieces of code?

Comparison of poorly written and self-documented code in Matlab

```
% Poorly written/Cryptic code in Matlab
[a1, a2] = slvqd(A,B,C);

fprintf('we got %f, %f', a1, a2);
```

A self-documented version of the above code

```
[answer1, answer2] = solve_the_quadratic_formula(A,B,C);

fprintf('The solutions to Ax^2 + Bx + C = 0, are %f, %f', answer1, answer2);
```

File and Function Header Comments

- File Header comments are used to identify what is in a file, who wrote it, the date it was written, and a description of what is being solved by the code in the file.
- All program files should have header comments and it should be located at the **TOP** of the file!
- The file header comment details what is in a file. Among other things it should have:
  - The author, date and company info. (course number for student programs)
  - A description of what the code in the file accomplishes
  - A list of any modifications (bug fixes) to the file. Note this is not as important for programs written in class, but important in the real world.

File Header Comments

- A good file header comment should fully describe the project and purpose of the code in the file.
- A programmer (or non-programmer for that matter) should be able to read the file header and be able to understand what is the high level idea and/or purpose behind the code, as well as what data-structures and methods are used.
- This can save many hours of time getting a new project member up to speed.

File Header Comments - Example

```
/**
 * File:    compute_blackjack_odds.C
 *
 * Author1:  H. James de St. Germain (germain@eng.utah.edu)
 * Author2:  Dav de St. Germain (dav@cs.utah.edu)
 * Date:     Spring 2007
 * Partner:  I worked alone
 * Course:   Computer Science 1000
 *
 * Summary of File:
 *
 *   This file contains code which simulates a blackjack game.
 *   Functions allow the user of the software to play against the
 *   "casino", or to simulate the odds of successfully "hitting"
 *   given any two cards.
 *
 */
```

Function Header Comments

- Function Header comments are used to describe the purpose of a function.
- Every function must have a separate header comment.
- Function headers serve to describe the algorithm which the function implements without forcing the reader to interpret code.
- Further, it serves to visually separate each function (e.g., in C, multiple functions are written in a single file).

Function Header Comments

- Short and simple functions can have only a few lines of description. As a rule of thumb, the longer the function the longer the header comment.
- Remember, always use appropriate amounts of whitespace and good formatting styles. This is as important in coding as in writing technical papers.
- By using a function header, you will need to use fewer comments in the actual code segment of the function. This will make your program cleaner and more readable.

Function Header Comments- Example

```
/**
 *
 * void sort( int array[] )
 *
 * Summary of the Sort function:
 *
 *    The Sort function, rearranges the given array of
 *    integers from highest to lowest
 *
 * Parameters    : array: containing integers
 *
 * Return Value : Nothing -- Note: Modifies the array "in place".
 *
 * Description:
 *
 *    This function utilizes the standard bubble sort algorithm...
 *    Note, the array is modified in place.
 *
 */

void
sort( int array[] )
{
  // code
}
```