ECED3403 Programming Best Practices

Dalhousie University

Last Edited: 2020/MAY/28

1 Overview

When writing code, it is important to follow some best practices to ensure your code can be read by others, and even yourself in the future. This document will summarize some (but not all) of the best practices you should follow (and will be marked on) when writing code for ECED3403.

2 Comments

Comments are used to help **humans** understand what is **intended** by the code it is associated with. It is important to understand that comments only indicate intent because code can have bugs, or maybe it was updated and the comment wasn't updated with it. This section outlines the C and C++ comment syntax and guidelines that will be enforced when marking your assignments.

2.1 Syntax

Both the C and C++ programming languages have two types of comments: A block comment starting with \⋆ and ending with ⋆\ as shown below,

```
/*
 * This is a block comment.
 */
```

and an inline comment which starts with

and ends at the end of a line.

```
// This is an inline comment.
```

2.2 File header comments

File header comments go at the top of every code file (.c, .cpp, .h, etc.). The comment should be a block style comment that includes the following:

- File name
- Author
- School (or place of employment)
- Department
- Course number
- A description of the file's purpose
- The date it was last modified

Example:

```
/* File: symbolTable.c
 * Author: Gary Hubley
 * School: Dalhousie University
 * Dept: Electrical and Computer Engineering
 * Course: ECED3403 Computer Architecture
 *
 * Purpose: This file is the contains function
 *          definitions associated with the symbol
 *          table struct defined in struct.h
 *
 * Last Modified: 2020.05.28
 */
```

2.3 Function comments

Function comments are comments that appear on the line before a function **declaration** as a block comment. This is usually in the header file but could also be in a source file. The function comments will include:

- Function name
- Function description
- A list of arguments, each with its own description
- A description of what is returned

Example (link):

```
/*
 * Function:  approx_pi
 * --------------------
 * computes an approximation of pi using:
 *    pi/6 = 1/2 + (1/2 x 3/4) 1/5 (1/2)^3  + (1/2 x
   3/4 x 5/6) 1/7 (1/2)^5 +
 *
 *  n: number of terms in the series to sum
 *
 *  returns: the approximate value of pi obtained by
   suming the first n terms
 *           in the above series
 *           returns zero on error (if n is non-
   positive)
 */
double approx_pi(int n) {
    ...
}
```

2.4 Inline comments

Inline comments are comments that describe something you're doing that might not be obvious by simply reading the code. Inline comments should not exist where the code is self descriptive. Inline comments can appear as block comments or as an line comment - the choice is yours. See the examples below.

BAD example:

```
x++; //increment x by 1.
y = getInstructionIndex("ld"); /* get the index of ld
                                 from inst. table */
```

Note that the first line in the bad example is blatantly obvious and a comment is clearly not needed. The second line is only obvious becuase the programmer named the function being called in a descriptive manner. If the function being called was 'getIdx()' then the comment might be necessary. There will be a section on naming added later.

GOOD example:

```
sum = num * (num + 1) / 2; /* Get sum of first <num>
                              integers */
```

Note that this inline comment is placed where the code is not obvious to the average reader (did you remember that formula from first year statistics?). Therefore, an inline comment is warranted and welcomed by the reader. Keep this in mind when writing your code.

3 Formatting

Coming Soon!

4 Naming

Coming soon!

5 General

Coming soon!