CST126 W19 Lab 1 Cyclomatic Complexity

# Introduction

Developers often use software metrics to pinpoint what functions or code sections are more complex than others. Metrics can help you decide what maintenance or extra testing needs to occur. Examples are LOC - lines of code (boo – almost never helpful), test coverage (helpful but sometimes drives odd behavior) and cyclomatic complexity (yea!). Cyclomatic Complexity is a measure of the number of paths through a program. You can learn more about Cyclomatic Complexity here: <https://en.wikipedia.org/wiki/Cyclomatic_complexity>.

In this lab we will do a simplified version of Cyclomatic Complexity. We will count the number of occurrences of the following keywords: *if*, *else*, *while*, *case*, and *for*.

Please note that to do a real calculation of Cyclomatic Complexity we would need to parse the file, something you will learn how to do in compilers. So, we will have some oddities, the most glaring is that it we are counting keywords that are in comments.

# Learning objectives

* Reviewing file IO
* Structures
* Includes
* String class

# Requirements

* Count the number of occurrences of: *if*, *else*, *while*, *case*, and *for* within a text file.
  + I’ve provided a sample for you, test.txt.
  + Another option is to use your lab code. (You can conveniently copy and paste the name from the first line.)
  + When we grade, we’ll use some sample .cpp from the class folder.
  + You can assume your keywords are lower case because that’s the way they would be in a .cpp.
* Use the string class for any text variables.
* Count the number of words in total in the file.
* Output your counts in a reasonably pretty way including telling the user the file name.
* Your main function can be no more than 15 lines long. Use functions instead. Don’t cram to get to the 15 lines. Mine is 11 lines including the int main and the closing }.
* Don’t wait to get started on this. You can do all the file IO with what you learned in 116.
* Do please do this incrementally. In the program, I added comments with most of the steps that I did. I kept in mind that you wouldn’t know about structures until the second week. They are only a suggestion. Just do a small bit and compile and test.
* Use a structure to store your data. This structure will have:
  + A print function
  + A data initialization function.
  + A total count of words.
  + A count for each of the keywords either as separate variables or as a single array of counts.
  + The name of the file read
* You must have a .h and .cpp for your structure. I know they will be very short.
  + Your .h must use #ifndef to ensure one time compilation. Next lab you will be able to explore #pragma once. This time use an ifndef.

# Output

Here is an example of the output from my program. You don’t have to copy it exactly. Just be self-documenting and not ugly.

