CS 161 Assignment 7.b - Comparisons
Christopher Matian
ID: 933308644

08/04/2017

To start, I want to say that a lot of my comparisons will be pedantic and minute in nature due to how similar the comparisons are to my own code. I will break down the comparisons similar to how it's structured in the rubric.

Comparison One: Audrey Young's Submission:

Taking a look at the header file **Person.hpp** the structure of both files is largely the same. The only minor difference is that no namespace is declared and is instead prepended when a certain library is called like so:

```
std::string name;
```

Out preference, I like to declare all of the namespace's I use in advance because it saves me from having to type out **std::** each time I want to import the proper library - saves time. The other thing of note is the lack of comments breaking up variable declarations, constructors, and methods. Again, minor things but having comments personally helps me a great deal when bouncing around through documents and looking at code. If I know a method is screwing up, I can quickly scan or search a document for the method's keyword and take a look at what's wrong.

Looking into the **Person.cpp** I see that our functions and class constructor's are all structure in the same manner. The major difference is once again the namespace. My code (unnecessarily) declares specific libraries for use like cout (even though cout is never used) and Audrey's code imports the *entire* namespace library. Both are unnecessary. Lastly, there is a lack of comments in Audrey's code whereas mine includes a break-up of section indicating where the class constructors are and where the getter functions are located.

Finally, looking into **stdDev.cpp** and going down the document I can see an import of the entire namespace library in Audrey's code (unnecessary). Additionally, the ***int main()*** isn't commented out (doesn't really matter in the scope of the group assignment) and there is a lack of comments breaking up various parts of the ***stdDev()*** function. Again, I think it's important to have some comments that explain code because it's a boon to anyone that takes a look into the code.

Comparison Two: Benjamin Jone's Submission:

Comparing my **Person.hpp** file to Ben's, I found that our files were extremely similar except Ben avoids including the *<iostream>.* During our group assignment, Ben explained that it wasn't necessary to import the cout/cin libary by virtue of the fact that they're not used in that particular document at all.

In **Person.cpp** Ben omits the use of unnecessary namespace libraries. In fact, Ben doesn't import any namespace libraries and instead only has an include for:

```
#include "Person.hpp"
```

He explained that because **Person.hpp** already imported the <string> libraries, and that **Person.cpp** was including the **.hpp** file, it wasn't necessary to declare the namespaces a second time - the library is already included because it's referencing that file. Other than that, Ben's **Person.cpp** file had more long-form comments that broke and explained what each constructor and getter did. My code simply include two lines of comments that specified where the class constructor began, and where the getter function began.

Finally, in the **stdDev.cpp** file, the major difference is that Ben's code doesn't really explain or outline the structure of the ***stdDev()*** function. Again, this is entirely a personal thing, but for the sake of the project I found it helpful to comment what each for loop did in order to break up the function and make sense of what was happening. It made it easier when it came down to debugging in that I could easily scan my document and find what might have gone wrong. Only other minor difference is how Ben structured his code. Namely, his **return** statement returned the square root of **variance**, whereas mine returned the square root of **standardDeviation** divided by **sampleSize**. See context below...

Context:

```
for (int i = 0; i < size; i++) {
	variance += pow((ar[i].getAge() - mean), 2.0) / size;
}
return sqrt(variance);
```

versus

```
for (int i = 0; i < sampleSize; i++) {
	// This effectively takes the age, subtracts the mean, and squares the result.
	// It then adds each result to standardDeviation
	standardDeviation += pow(peopleArray[i].getAge() - mean, 2);
}
// I return the sqrt of standardDeviation divided by n, which in this case is n = sampleSize.
return sqrt(standardDeviation/sampleSize);
```

Comparison Three: Donghao Lin's Submission:

There were again some minor differences between my code and Donghao's (abbreviated D) code. First, D's **Person.hpp** file had the same includes. However, similar to Audrey's code he opted to import the entire namespace library (not needed). Secondly, he has no comments to break up the code that he uses or defines. It's just one giant block whereas my own separates declarations, constructors, and methods. Additionally, the public and private data sections are reversed in D's file.

In **Person.cpp** the one thing of note is that D's code had individual comments for each constructor and getter function. In hindsight, this is probably the better route than my own (only denotes the code by section). The other minor thing is that D's code included a default constructor which was unnecessary given the scope of the assignment. It's never really used and there are no setter functions that can be called to manipulate the data.

Lastly, **stdDev.cpp** is mostly the same in structure. I keep seeing the same minor differences (namespace inclusions, comments, etc). The one thing that did stand out however was that D's code initialized the variable N to size (which is passed in as a parameter). To my understanding, this was done because the actual formula for calculating standard deviation uses the character ***N*** to denote the initial size. D opted to use the same format, but it can be slightly confusing because the rest of the function makes no reference to ***N***, and instead just uses the variable ***size*** directly. Only D's **return** statement makes use of ***N:***

```
return sqrt(totalSquare/N);
```

I personally felt this was a waste of time to declare, and could have simply been handled by using ***size*** directly.

Closing thoughts and what can be gleaned from the comparisons:

Having gone through the comparisons I found that it was massively helpful looking and speaking directly with Ben regarding how importing files works. He explained that when importing files, it's redundant to keep importing the same namespace's when it's already being handled and referenced during the import stage. I was pretty ignorant of this and as such I will be keeping this in mind when I structure my documents in the future.

The other thing that I ended up learning was from a question Audrey asked during the discussions where she pointed out that D's **Person.hpp** file had swapped around the public and private data fields. I went ahead and researched if this made a difference and the stack overflow consensus was that it is better to declare the guts of the class first (i.e. the public data fields) and then have the private data fields below all of that. This is because in longer files where there are tons of classes and lots of constructors/methods, it's better to have that information high up in the hierarchy so that it's easier to reference and work on. I think that on future assignments, I will declare public data members first and then private data members.

Even though the scope of most assignments is relatively concentrated and short form, I find it's best to reinforce and hammer home programming concepts and standards while I'm learning. I don't want to turn it in to a habit which ends up causing frustration for myself and other engineers in professional work environments.