Alex Childers

PA #5 Reflection Essay

04/24/2019

I rate my work on this assignment a 6.4/10. The content of the code itself merits a 6, because I think it could use some serious refactoring, but I did at least make an attempt to break down the code into smaller functions this time, instead of having it all in main. In the future, I would want to refactor the code, implement the SQLite functionality, and possibly record user-typed words in words.txt.

Oddly, one of the bigger challenges of this assignment was deciding how to remove punctuation from words before processing them. There’s many different uses of punctuation in the English language—for instance: ‘twas, “hello,” y’all’d’v’f’I’d’ve, back-to-back, ‘n’. I had a difficult time deciding which bits of punctuation mattered to the word content, and which bits didn’t. For instance, the apostrophes in ‘hi’ are less significant to its spelling than in the word wasn’t. However, I did have a lot of fun brainstorming out what rules I would programmatically enforce, and I eventually settled on just removing punctuation from either end of the word, letting the user deal with any exceptional cases. In the process, I also learned about several C++ functions that determine if characters are punctuation, alphanumeric, and so on, so it was a fun little learning experience.

I also struggled with how to implement storage for autocorrected words throughout the program. I wanted to use an unordered\_map, but I couldn’t figure out how to use them, since the order of words based on edit distances mattered. In the end, I fell back on using vectors, since I couldn’t figure out what other data structure could hold words in an ordered manner. We had a brief discussion today about using parallel vectors as an alternative to vectors of pairs, since the pair.first/pair.second notation is clumsy. I do heavily rely on vectors of pairs in this program, so if I had thought of parallel vectors from the beginning, I would try storing words and edit distances in that format instead. In my search for appropriate data structures, I also learned a little bit about deques. One of the great things about struggling with these design decisions is that, as I scour the Internet for answers, I get exposure to many more features of C++ than I would have learned about in class alone.

Overall, this PA wasn’t too difficult. I knew the general tasks that I needed to complete for the PA, and I was confident that I had the knowledge I needed to implement each feature of this program. The most difficult part of this assignment was, as ever, making design decisions, abstracting out code, and figuring out the program logic. However, knowing how to use the data structures I implemented, while trying out C++ features I’d never used before, made debugging a little bit easier and more fun, since I could observe step-by-step how all the variables of the program interacted with each other and passed data back and forth.