Mark Watson

Anagram Maker

Gary Underwood

12/3/12

I approached this a lot like how I set it up in my PowerPoint since a lot of my code was done when I did my power point. I have 2 vectors, a initial string vector that only holds the starting dictionary, I then sort the dictionary into a vector of my Struct type that only had the letter count <= the input word. This struct holds the letter count, the word, and it makes a char alphabet array to keep track of how many of each letter is in the word for comparisons later. After it has done this the program calls the subtraction method which takes off letters and then recuses through the temp dictionary, if it finds an anagram then it adds the word to the soFar string variable and it adds 1 to the counter, if it doesn’t then it just recuses again to the next word. I prevented repeats because I kept a iterator that moved forward after each word had been tried and it never went backwards, it passed in its location at each recursion and it always moved forward, this prevented repeats. I tried to get it a little bit faster, I originally had 2 iterators I was passing into the helper function and I decided to pass in the actually temp dictionary by const reference and that made it a little faster but not by much. I also tried to mess around with c-strings a bit but could not get them to work because I ran out of time. My program is in between your slow no repeat and your good solution and I would of liked to get it a little faster but just don’t have the time to mess with it anymore.