**Daniel Williamson**

**Project #6**

**4/1/16**

**CS 200**

Project Overview

Purpose:

The purpose of this project was to have a file scanned in and use RLE compression to compress any given text, symbol and then decompress said text, symbol’s. Also, once compression happens to create an output txt file containing said compression.

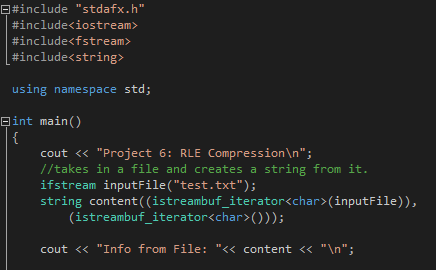
Approach:

To begin this project, I immediately went to the internet to gain a better understanding of how to program in C++. Also, how RLE Compression works. Once I read up, I began to plan how the program should run. I began by simply reading in a given txt file with a few repeated characters on it and print them out to the console as a string. Once this was complete, I took that string and used a while loop to loop through all the characters. It would run through the characters and after each time raising the counter, if it detected a character that was not equal to itself it would place said character down and the counter next to it. Then it would create an output file with the compressed values. As for the decoding, it would re-open the output file and iterate through every other value starting at 0 and multiplying it by every other value starting at 1. Which we would then print to the console.

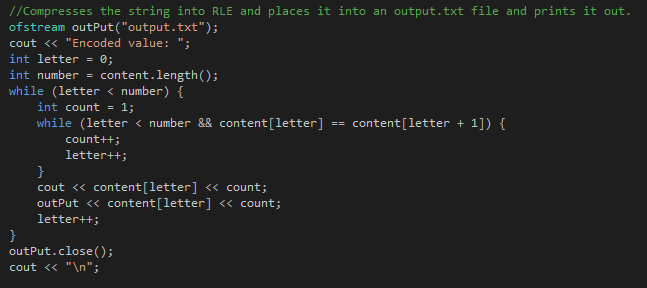
Results

These are my results:

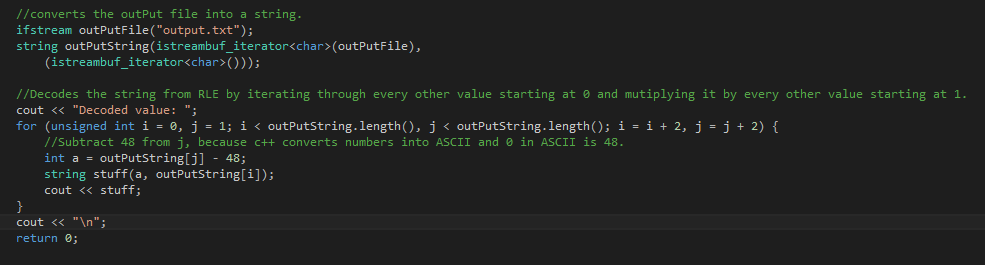
This is where we read in the test.txt file and convert it into a string and print it out to the console.



This is where we compress the information into RLE and place it into an output.txt file and print it out to the console.



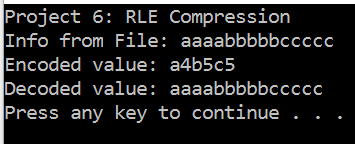
This is where we decompress the now RLE compressed information and print it out to the console.



Testing:

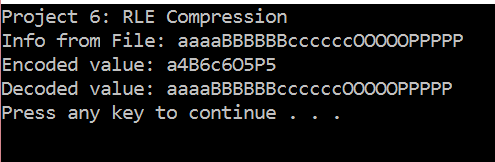
Test #1 using only lower case letters:

If we put into input txt file “aaaabbbbbccccc” we should get an RLE compression of a4b5c5.



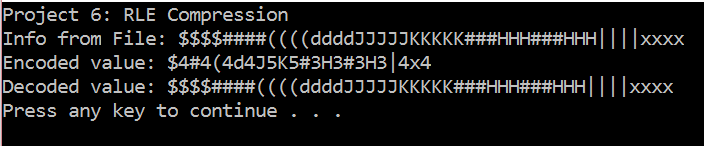
Test #2 using upper case and lower case letters:

If we put into the input txt file “aaaaBBBBBBccccccOOOOOPPPPP” we should get a RLE compression of a4B6c6O5P5.



Test #4 using characters and letters:

If we put into the input txt file “$$$$####((((ddddJJJJJKKKKK###HHH###HHH||||xxxx” we should get a RLE compression of $4#4(4d4J5K5#3H3#3H3|4x4.



Conclusion

In conclusion this was a rather difficult test. I relied heavily on the internet for how RLE compression work. However, as I read up on it, it become clear it was a rather simple compression. The hardest challenge I faced was figuring out why I need to subtract 48 from an int in C++. It’s because C++ converts numbers into ASCII and 0 in ASCII is 48.