**Project 5: Base64**

By Austin Kelly

March 24, 2017

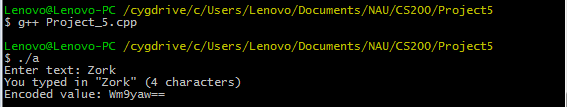
Overview

The purpose of the first part of this assignment is to understand how base64 conversion works, and how to encode and decode using it. We were tasked with creating a program that takes in a string and converts it to the corresponding base64 “word.” I decided to use C++ for this project.

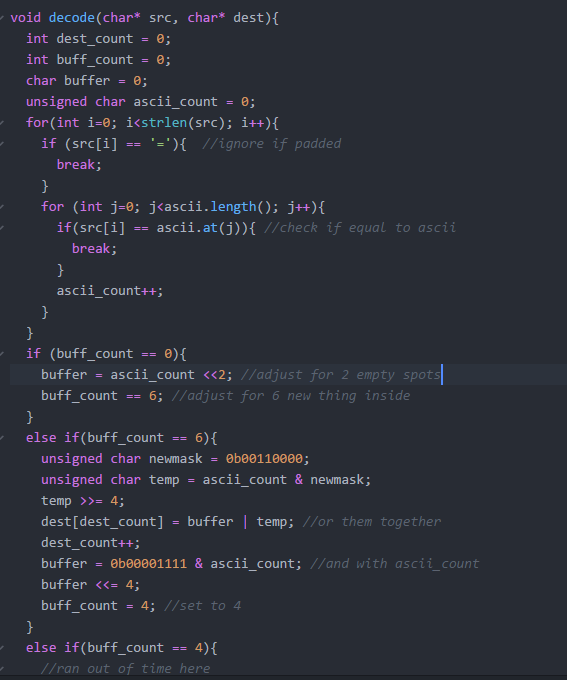
Approach

I approached this project by copying the skeleton code, analyzing it, and figuring out how it worked. I ended up completely scrapping the content of the “encode” function, but I did use the same variable naming conventions. I absolutely would not be able to complete this project if it wasn’t for the binary manipulation example in the project summary. Seeing the underlined sections how it takes the first 6 bits and divides into smaller sections was incredibly helpful. I ended up referencing that example constantly. For the encode function, I ended up adding a third parameter that takes in the number of bytes. This is to account for the overlap that can occur, as it originally would only print out part of the converted word. Based on the number of bytes, I decided what part of the encode function should initiate. To get the bits I needed, I created a mask with 1’s in the corresponding spots that I did need and 0’s for the places I didn’t. I then anded them together to get the correct values. If there was a part in the next section I needed, I would do the same with that in a temp variable and then or the 2 parts together to combine them. After that I adjusted the bits so that they were aligned to the correct order. Now that I had the correct value in binary, I set the destination point to the right value. For the decode section, I approached it a little differently. Since the word was padded with = signs every one was the same length, so I was able to go through using nested for loops and buffers. I broke the qhole process if the value was just padding, and if it wasn’t I iterated through the full list of ascii values checking to see if they matched. I then decoded using a very similar technique to encoding, except using the buffer count and setting it when appropriate.

Results

I was able to get the encode section working absolutely flawlessly. It even accounts for all situations and padding which I am extremely happy with. It was tricky working with this and I ran into a ton of problems since it reads it in groups of 3 and the overflow needed to be acounted for. Below is a screenshot of the encoding working perfectly with the example used in the project description, but this can be checked with anything else with the expected results. 

I ran out of time for the decoder part but it is 99% done. As of this submission I commented what I had and hopefully it is easy to see where I was going, as it is basically finished. The only part out of all of it that is missing is masking the last 2 sections in the if chain, and I also need to add the print statement for the decoded word. Below is a screenshot of the working code.



As you can see, I accounted for the padding, and I adjust the values correctly using bitwise operators in a reverse way of the encode, except using the counts from the for loop.

Conclusion

Overall this project was insanely difficult for me. I spent more hours on this one than probably all of the previous assignments combined. I am incredibly happy with my results, and I am proud of myself for getting everything working. I ran into a lot of trouble with the encoded word printing only some of the values or even not at all. I had to think outside the box to account for this without scrapping all of the work I was happy with for the bit adjustments. I am really happy with the result after adding if else statements for different parts of the process. This was definitely the hardest project I have worked on in this class and was a steep learning curve that taught me just about everything there is to know about base64 encoding.