Basic

Commentary

Wow… I must say this language is the best one I've used so far in this assignment, that doesn't say much with COBOL and Fortran as what I'm comparing with. It really isn't far off from today's modern languages. With the other "journal entries," I had written them as I went, but here, there wasn't enough friction to stop me from just coding all the way through. Half of the time spent getting this program to run was just finding out what version of BASIC I should use and setting up the compiler. I went with FreeBASIC as it seemed to have the best docs and was listed on your website. With the other languages, finding help was very hard compared to BASIC. It probably has something to do with how this was a common language for students to use so there are plentiful beginner documentations. COBOL being business oriented doesn't have an audience that is normally beginners. I was able to find a list of the string methods, and a way to loop over them and everything just kinda made sense based on my existing knowledge of high-level languages.

Google Searches

- basic compiler
- basic getting started
- basic hello world
- basic ubuntu apt
- basic for loop
- freebasic string addition
- wordle
- freebasic string index
- freebasic modulus
- freebasic functions
- freebasic sub

Caesar Implementation

```
'compile and run with fbc caesar.bas && ./caesar

declare function encrypt(word as String, shift as Integer) as String
declare function decrypt(word as String, shift as Integer) as String
declare Sub solve(word as String, maxShiftValue as Integer)

print encrypt("ATTACK AT ONCE", 4)
print decrypt("EXXEGO EX SRGI", 4)

solve("abcdeFGHIJKLmnopqrstuvwxyz ,?;{[()]}", 26)

function encrypt(word as String, shift as Integer) as String
    dim i as Integer
    word = ucase(word)

    for i = 0 TO len(word) - 1
```

```
        'print i, word[i], chr(word[i])

        if word[i] >= 64 and word[i] <= 90 then
            word[i] = (word[i] - 65 + shift) mod 26 + 65
        end if
    next

    Return(word)

end function

function decrypt(word as String, shift as Integer) as String
    Return(encrypt(word, 26-shift))
end function

sub solve(word as String, maxShiftValue as Integer)
    dim i as Integer

    for i = 1 to maxShiftValue
        'strings are modified in place so always shift by 1
        print "shift: " & i, decrypt(word, 1)
    next


end sub
```

Output

```
EXXEGO EX SRGI
ATTACK AT ONCE
shift: 1      ZABCDEFGHIJKLMNOPQRSTUVWXY ,?;{[()]}
shift: 2      YZABCDEFGHIJKLMNOPQRSTUVWX ,?;{[()]}
shift: 3      XYZABCDEFGHIJKLMNOPQRSTUVW ,?;{[()]}
shift: 4      WXYZABCDEFGHIJKLMNOPQRSTUV ,?;{[()]}
shift: 5      VWXYZABCDEFGHIJKLMNOPQRSTU ,?;{[()]}
shift: 6      UVWXYZABCDEFGHIJKLMNOPQRST ,?;{[()]}
shift: 7      TUVWXYZABCDEFGHIJKLMNOPQRS ,?;{[()]}
shift: 8      STUVWXYZABCDEFGHIJKLMNOPQR ,?;{[()]}
shift: 9      RSTUVWXYZABCDEFGHIJKLMNOPQ ,?;{[()]}
shift: 10     QRSTUVWXYZABCDEFGHIJKLMNOP ,?;{[()]}
shift: 11     PQRSTUVWXYZABCDEFGHIJKLMNO ,?;{[()]}
shift: 12     OPQRSTUVWXYZABCDEFGHIJKLMN ,?;{[()]}
shift: 13     NOPQRSTUVWXYZABCDEFGHIJKLM ,?;{[()]}
shift: 14     MNOPQRSTUVWXYZABCDEFGHIJKL ,?;{[()]}
shift: 15     LMNOPQRSTUVWXYZABCDEFGHIJK ,?;{[()]}
shift: 16     KLMNOPQRSTUVWXYZABCDEFGHIJ ,?;{[()]}
shift: 17     JKLMNOPQRSTUVWXYZABCDEFGHI ,?;{[()]}
shift: 18     IJKLMNOPQRSTUVWXYZABCDEFGH ,?;{[()]}
shift: 19     HIJKLMNOPQRSTUVWXYZABCDEFG ,?;{[()]}
shift: 20     GHIJKLMNOPQRSTUVWXYZABCDEF ,?;{[()]}
shift: 21     FGHIJKLMNOPQRSTUVWXYZABCDE ,?;{[()]}
```

```
shift: 22     EFGHIJKLMNOPQRSTUVWXYZABCD ,?;{[()]}
shift: 23     DEFGHIJKLMNOPQRSTUVWXYZABC ,?;{[()]}
shift: 24     CDEFGHIJKLMNOPQRSTUVWXYZAB ,?;{[()]}
shift: 25     BCDEFGHIJKLMNOPQRSTUVWXYZA ,?;{[()]}
shift: 26     ABCDEFGHIJKLMNOPQRSTUVWXYZ ,?;{[()]}
```

Log

Prediction: 2 hours

Date	Hours Spent	Accomplishments
2/3	1	Figure which version of basic and install compiler
2/7	.5	Paste hello world example and find helper method docs
2/7	.5	Implement encrypt, decrypt, solve
2/21	.5	Test thoroughly

Discrepancy of time

I still went over in time here, but not in the initial pass through the program. I was pretty close to my estimate. The biggest loss in time was just figuring out which version of basic to use. I kept seeing results of visual basic and the likes. Also, there wasn't a package on apt for basic, I had to manually install a linux binary, which I should've just done and not tried to find a different version on apt.

Overall Review

I really enjoyed this language, there were a few aspects which confused me. For example, declaring the function signatures at the top of the file and then having to completely re-write them when I want to actually use it. This made the program less writable and I don't see much of a benefit from doing it this way. The way the language requires all variables at the top of the function reminds me of Dr. Schwartz's SD1 class. He would always take off points when they weren't declared at the top and I never understood why.

Ratings

Readability: 8/10

Writability: 7/10

Ranking: 2/5