COBOL

Commentary

For COBOL, I knew it was a much more structured language and that a simple hello world would not be simple, similar to Java, but worse. I started by looking for a youtube video to get a general overview. The only real video for beginners was COBOL in 100 seconds by fireship, a channel I subscribe to and watch regularly. This allowed me to understand the tip of the iceberg, and nothing more, and from there I copied a hello world example into IDEone. Doesn't compile. After debugging this for way longer than I should have the problem was related to how the code was spaced. Little do I know how much time I would save if I was to just abandon IDEone at this point. I end up counting out the correct number of spaces for each line and eventually get it to compile. *woot!* My next problem was to find any sort of documentation of the keywords. It seems IBM is the only real provider of this. I also hate IBM's website, I've never seen such a bloated website for documentation, only for it to be hard to find any useful info, no code snippets, no getting started tutorials, a long way from geeksforgeeks.org. Maximum line length was a big blocker for a while, I couldn't figure out how to override it or anything so I eventually came up with breaking it up into smaller statements and hold the intermediate information in temp variables. This problem wasn't made any easier by the amount of text it takes to call a function. Why should I have to write FUNCTION in front of everything I want to use? Also, I was never able to find a way to use functions, I'm just kinda guessing the way I did it is reasonable. I am basically using a combination of GOTOs and using some global variables. They have to exist... right??

Google Searches

- COBOL getting started
- Fireship cobol 100 seconds
- GET ASCII CODE FROM CHARACTER IN COBOL
- COBOL TO UPPERCASE
- CREATE A FUNCTION IN COBOL (spent like 30 minutes here just trying to figure out if they exist or not)
- FIX COBOL LINE LENGTH TOO LONG
- COBOL LOOP OVER STRING
- Split cobol code onto multiple lines – couldn't find an answer to this
- COBOL compiler ubuntu

Caesar Implementation

NOTE: This language sucks so much it doesn't have syntax highlighting

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. CAESAR.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
```

```
        WORKING-STORAGE SECTION.

        01 INP PIC X(45) VALUE "ATTACK AT ONCE".
        01 SHIFT PIC 99 VALUE 4.
        01 OUT PIC X(45).
        01 LEN PIC 999.
        01 TMP1 PIC 999.
        01 TMP2 PIC 999.
        01 I PIC 999.
        01 S PIC 999.
        01 TMPSHIFT PIC 99.

        PROCEDURE DIVISION.

            MOVE FUNCTION UPPER-CASE(INP) TO INP.

            PERFORM ENCRYPT.

            DISPLAY FUNCTION TRIM(OUT).

            MOVE OUT TO INP.

            PERFORM DECRYPT.

            DISPLAY FUNCTION TRIM(OUT).

            MOVE 'abcdeFGHIJKLmnopqrstuvwxyz ,?;{[()]}' TO INP.
            MOVE FUNCTION UPPER-CASE(INP) TO INP.

            PERFORM SOLVE.

            STOP RUN.

            ENCRYPT.
            MOVE FUNCTION LENGTH(INP) TO LEN

            PERFORM VARYING I FROM 1 BY 1 UNTIL I > LEN
                MOVE FUNCTION ORD(INP(I:1)) TO TMP1
                IF TMP1 > 65 AND TMP1 < 92
                    MOVE FUNCTION MOD(TMP1 - 66 + SHIFT, 26) TO TMP2
                    MOVE FUNCTION CHAR(TMP2 + 66) TO OUT(I:1)
                ELSE
                    MOVE INP(I:1) TO OUT(I:1)
            END-PERFORM.

            DECRYPT.
            MOVE FUNCTION LENGTH(INP) TO LEN

            PERFORM VARYING I FROM 1 BY 1 UNTIL I > LEN
                MOVE FUNCTION ORD(INP(I:1)) TO TMP1
                IF TMP1 > 65 AND TMP1 < 92
                    MOVE FUNCTION MOD(TMP1 - 66 - SHIFT, 26) TO TMP2
                    MOVE FUNCTION CHAR(TMP2 + 66) TO OUT(I:1)
```

```
               ELSE
                   MOVE INP(I:1) TO OUT(I:1)
           END-PERFORM.

           SOLVE.
           DISPLAY "SOLVING...".
           MOVE SHIFT TO TMPSHIFT.
           PERFORM VARYING S FROM 1 BY 1 UNTIL S > 26
               MOVE S TO SHIFT
               PERFORM DECRYPT
               DISPLAY "SHIFT " SHIFT " " FUNCTION TRIM(OUT)
           END-PERFORM.
           MOVE TMPSHIFT TO SHIFT.
```

Output

```
EXXEGO EX SRGI
ATTACK AT ONCE
SOLVING...
SHIFT 01 ZABCDEFGHIJKLMNOPQRSTUVWXY ,?;{[()]}
SHIFT 02 YZABCDEFGHIJKLMNOPQRSTUVWX ,?;{[()]}
SHIFT 03 XYZABCDEFGHIJKLMNOPQRSTUVW ,?;{[()]}
SHIFT 04 WXYZABCDEFGHIJKLMNOPQRSTUV ,?;{[()]}
SHIFT 05 VWXYZABCDEFGHIJKLMNOPQRSTU ,?;{[()]}
SHIFT 06 UVWXYZABCDEFGHIJKLMNOPQRST ,?;{[()]}
SHIFT 07 TUVWXYZABCDEFGHIJKLMNOPQRS ,?;{[()]}
SHIFT 08 STUVWXYZABCDEFGHIJKLMNOPQR ,?;{[()]}
SHIFT 09 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

Date	Hours Spent	Accomplishments
1/27	1	RESEARCH COBOL
1/27	2	GET AROUND MAXIMUM LINE LENGTH, I NEVER WOULD’VE EXPECTED THIS PROBLEM
1/28	1	Install cobol compiler and set up local dev environment
1/28	1	Transform encrypt function into decrypt and solve
2/21	.5	Test thoroughly

Discrepancy of time

The biggest setbacks I had in this were just the very limited amount of help I could get from the internet. I know there is a meme going around on the internet that programmers are just good at google, but I guess COBOL programmers must be a different breed. I couldn't understand any resources that were provided or they were just way too wordy. I was caught in the endless loop of googling, clicking links, having too many words on the page, closing out and doing it again after 5 minutes.

Overall Review

This took me by far the most time to implement. One of my biggest problems was finding good documentation. It just doesn’t exist, at least for a basic program like caesar cipher. The language was designed for business applications, so one shouldn’t expect COBOL to make sense here, but at the same time, a simple program shouldn’t ever be difficult to implement. By far the least writable of the languages, way too verbose and structured. It somewhat makes up for it with the readability which is very close to english. Overall, a terrible language but I can’t blame the creators of it because it suits the needs of that time period very well. Having specific columns of code for certain things makes no sense nowadays, but at the time, when punch cards were being used, I’m sure it made a lot of sense. Having to correctly space the columns was a huge downside, both for readability and writability.

Ratings

Readability: 4/10

Writability: 1/10

Ranking: 5th