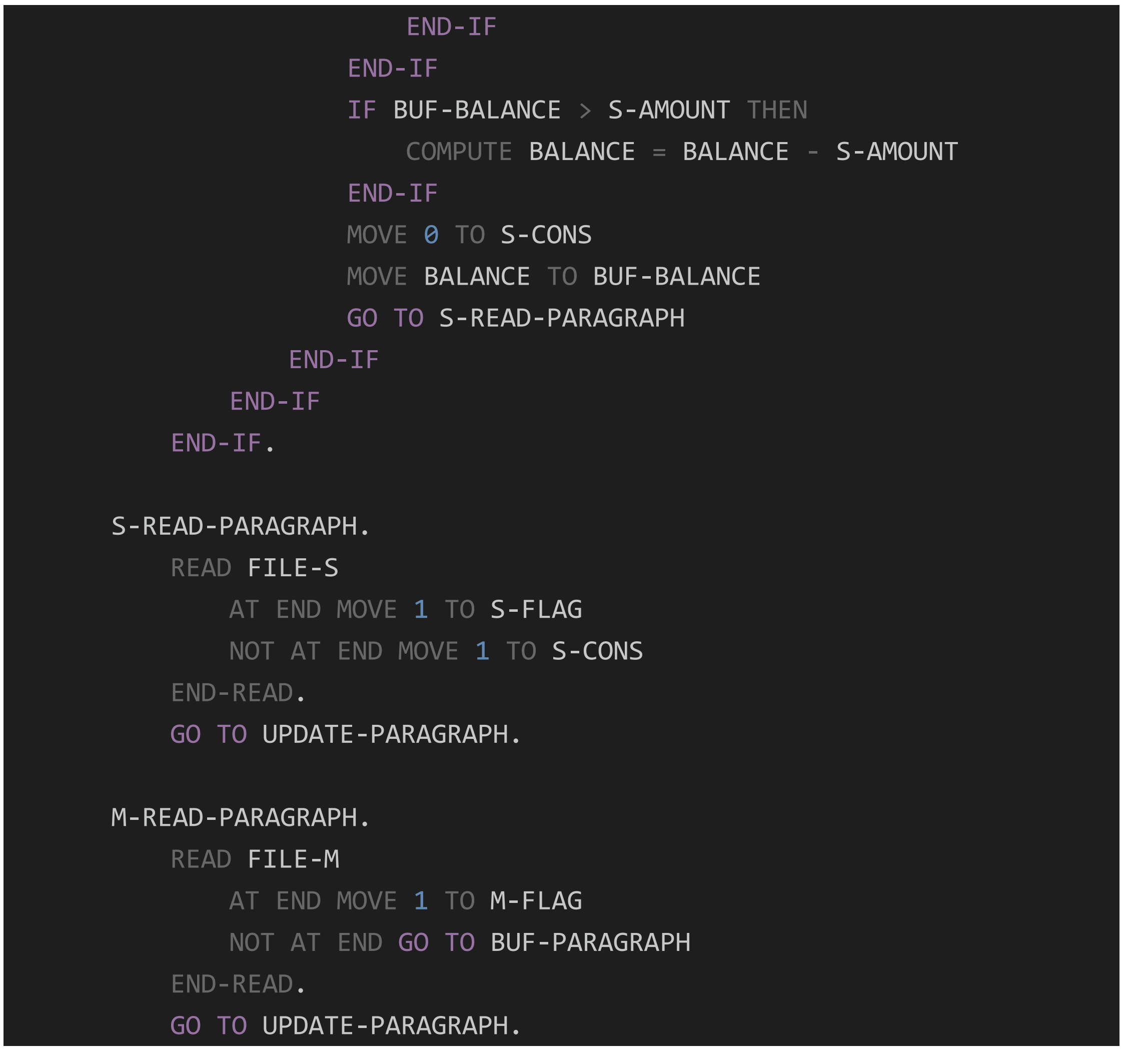
1. reading file: COBOL is better than C. Because in COBOL, I can read file items and no need to concern \n \r and author characters. What I just do is code READ FILE-TS, but in C, I should consider how many bytes are read, for example, fread(buf711acc,1,16,fd711). And in COBOL, it can find the end of file automatically using AT END. But in C, I need to locate it by myself using ftell and fseek.

   Simulating loops: C is much more better than COBOL. Each module in C is one in one out. If I use a loop, I can certainly know where it starts and where it ends. But in COBOL, using go to, when I debug, I have to track go tos from one to another and sometimes back to the first one, which makes COBOL hard to debug.

```
IF BUF-ACCOUNT = S-ACCOUNT THEN
               IF S-OPERATION = 'D' THEN
                   IF BUF-SIGN = '+' THEN
                       COMPUTE BALANCE = BALANCE + S-AMOUNT
                       MOVE 0 TO S-CONS
                       MOVE BALANCE TO BUF-BALANCE
                       GO TO S-READ-PARAGRAPH
                   END-IF
                   IF BUF-SIGN = '-' THEN
                       IF BUF-BALANCE > S-AMOUNT THEN
                           COMPUTE BALANCE = BALANCE - S-AMOUNT
                       END-IF
                       IF BUF-BALANCE <= S-AMOUNT THEN
                           COMPUTE BALANCE = S-AMOUNT - BALANCE
                           MOVE '+' TO BUF-SIGN
                       END-IF
                       MOVE 0 TO S-CONS
                       MOVE BALANCE TO BUF-BALANCE
                       GO TO S-READ-PARAGRAPH
                   END-IF
               END-IF
               IF S-OPERATION = 'W' THEN
                   IF BUF-SIGN = '-' THEN
                       COMPUTE BALANCE = BALANCE + S-AMOUNT
                       MOVE 0 TO S-CONS
                       MOVE BALANCE TO BUF-BALANCE
                       GO TO S-READ-PARAGRAPH
                   END-IF
                   IF BUF-SIGN = '+' THEN
                       IF BUF-BALANCE <= S-AMOUNT THEN
                           COMPUTE BALANCE = S-AMOUNT - BALANCE
                           MOVE '-' TO BUF-SIGN
                           IF BALANCE = 0 THEN
                               MOVE '+' TO BUF-SIGN
```



```
                    END-IF
                END-IF
                IF BUF-BALANCE > S-AMOUNT THEN
                    COMPUTE BALANCE = BALANCE - S-AMOUNT
                END-IF
                MOVE 0 TO S-CONS
                MOVE BALANCE TO BUF-BALANCE
                GO TO S-READ-PARAGRAPH
            END-IF
        END-IF
    END-IF.

S-READ-PARAGRAPH.
    READ FILE-S
        AT END MOVE 1 TO S-FLAG
        NOT AT END MOVE 1 TO S-CONS
    END-READ.
    GO TO UPDATE-PARAGRAPH.

M-READ-PARAGRAPH.
    READ FILE-M
        AT END MOVE 1 TO M-FLAG
        NOT AT END GO TO BUF-PARAGRAPH
    END-READ.
    GO TO UPDATE-PARAGRAPH.
```

Function call: C is more convenient. First I write COBOL program, I thought its go to just like a function call in C, it can come back to previous segment. But I was wrong, when it go to another paragraph, it will never come back. That is very inconvenient because sometimes I need to write something behind go to. If I can not do so, I can only use a more complex way.

2. Compare with java.
   Variable declaration: in java I can declare a variable anywhere I want, for example, I want to write a for loop, I can write for(int i;……) and for every loop not nest in my program, I can always use i to do loop. That can not cause fault. But in COBOL, there are only global variables. And there is only one zone to declare them. That means if I cant decide all variables I will use, I will come back to write declaration time and time again.
   Data type: in COBOL data types are so few. I can only use[AX90S] formats. But in java, any data type such as char, int, booleen can be combined to a new data type. That is really easy to code.

3 It is suitable. I really feel the power of while loop. Nowadays, almost all famous languages can use while loop. So I used to think it is very normal. Until I experience what is go to. I think loop is a very millstone in programing languages. But COBOL

has its advantage too. For very long numbers like 20 digits, COBOL is so powerful. Even the length of integer is longer than long long type, COBOL can still store and calculate easily. And in COBOL, the digits of a variable is fixed. So when display same type, it will Align automatically.