Cobol

By: Steven Conner

History:

COBOL, Common Business-Oriented Language, one of the oldest programming languages, was designed in the last six months of 1959 by the CODASYL Committee, Conference on Data Systems Languages. COBOL’s specifications were strongly inspired by the language FLOW-MATIC. It was developed for the United States government and businesses during the arms race of the cold war, while the Soviet Union used APL, Address Programming Language. However, even though COBOL was developed by 1960, COBOL didn’t have a standardized version until 1968 when ANSI, American National Standard Institute, took over the responsibility of developing new COBOL standards and COBOL-68/ANS COBOL was developed. Since COBOL was first developed it has undergone four standardizations.

ANS COBOL/COBOL-68 mostly just fixed problems of incompatibility with different versions COBOL by making a standard version.

COBOL-74 just added a lot of new features not included in COBOL-68.

COBOL-85 also added in new features. Most importantly the
structured language constructs, END-IF, END-PERFORM, END-READ, etc.

**COBOL-2002/00 COBOL**, Object Oriented COBOL. This new version was approved by ISO, International Organization for Standardization, and like the name implies, it allows Object Oriented programming for COBOL. However, a new standardized library is being worked on but the expected time of completion was 2007 but the new standard has not yet been approved.

Even though COBOL is over fifty years old, it is still used in a majority of businesses in the United States and the United Kingdom. There are over 200 billion lines of running COBOL code with an annually addition of 5 billion lines of code.

**Overview:**

**Coding:** A COBOL program is divided into four division. These divisions are the IDENTIFICATION DIVISION, ENVIRONMENT DIVISION, DATA DIVISION, PROCEDURE DIVISION. In the IDENTIFICATION DIVISION the only thing required is the program name but this division also normally contains the programmer’s name, the date, and intended purpose. In the ENVIRONMENT DIVISION you tell the program what it will interact with. This
can include but not limited to files, printers, other computer, etc. This division is split into two sections, the CONFIGURATION SECTION which deals with the source and object computer, normally the same thing and doesn’t need to be included, and the INPUT-OUTPUT SECTION deals which all other interactions. Next is the DATA DIVISION which is also divide into two sections. The FILE SECTION which deals with how all info retrieved from a file or being sent to a file will be formatted and handled, and the WORKING-STOREAGE SECTION which deals with all other piece of data. The last division is called the PROCEDURE DIVISION and it is well all the actual programming takes place. In the PROCEDURE DIVISION the program is broken up into paragraphs or modules. You use Perform to call these paragraphs but may only call paragraphs below the call and once a paragraph has finish if no call was issued again the the program jumps back up to the last call and drops down one line.

**Name, Scope, Binding:** Names in COBOL must begin with a letter and can be a maximum of 30 characters long. The name can be composed of letters from A-Z (all uppercase), numbers 0-9, and hyphens (-). The scope of COBOL doesn’t exist. COBOL has no scope but if you had to give it a scope it would be
static. COBOL only deals with binding when accessing SQL for information.

**Data Types:** COBOL has only three basic data types declared after the ‘PICture’ clause, ‘A’ alpha, ‘X’ alpha-numeric, ‘9’ numeric. These data types can be declared as ‘DISPLAY’ or ‘COMPUTATION’ with the use of ‘USAGE’. Display is implied as long as the data type isn’t declared as Computation. Display is used to show the data, while Computation is used for computing the data in the form of Comp, or Comp-1 through Comp-5 most commonly used is Comp and Comp-3. Comp which is also Comp-4, stores binary data that is normally used in indexes, switches, or arithmetic and Comp3 being a packed decimal which is normally used for a fix value arithmetics. Comp-1 and Comp-2 are short and long, respectively, floating points that can be stored in IEEE format or hexadecimal format. While COMP-5 is used to store data values from files that might not fit PIC structure. A ‘S’ for sign and ‘V’ for an implied decimal point, can also be inserted in the pic clause if it being used as ‘DISPLAY’. ex. Weekly-Pay PIC $9(5)v99 USAGE DISPLAY

**Expressions and Assignment Statements:** COBOL allows a wide variety of ways to write expression and assignment
statements from using the familiar symbols like =,*,+,−,/, to keeping to its English syntax by using ADD, SUBTRACT, DIVIDE, MULTIPLY, MOVE, COMPUTE, BY, TO FROM, and INTO. COBOL also allows you to mix and match as you see fit but a good COBOL programmer uses the same statement format throughout the whole program.

Control Structures: COBOL has fewer control structures than most other languages but enough to get the job done. Just like the expression and assignment statements control structures in COBOL have both a word and symbol form, </LESS, >/GREATER, </LESS OR EQUAL, >/GREATER OR EQUAL, </EQUAL, NOT=>/NOT EQUAL. These statements are used in COBOL just like in most other languages and are used to compare values. COBOL also has three logical operators, AND, NOT, OR. These logic operators are used inside IF statement to allow more control and easier coding due to only having to write one IF statement instead of two or more. The AND statement makes it so all values must be true while an OR statement allows an IF statement to pass as long as one of the values is true. The NOT statement makes a true value false and a false value true. The selection statements in COBOL are, IF, THEN, ELSE, END-IF, ACCEPT, EVALUATE, WHEN, WHEN OTHER, END-EVALUATE, EXIT, and
UNTIL. These statement are actually a huge part of the COBOL language. They are what allows the program to handle different situations of data. The EXIT statement is used to return to the line right after the last PERFORM statement. The IF statement is the biggest exception handling and allows for the program to head in different directions. The EVALUATE statement is used whenever multiple actions, more than three, can be executed because of a piece of data. For example taxes on paygrades.

    ACCEPT PAY
    EVALUATE
    WHEN PAY IS GREATER OR EQUAL TO 45000
        MOVE 32 INTO TAXES
    WHEN PAY IS GREATER OR EQUAL TO 30000
        MOVE 27 INTO TAXES
    WHEN OTHER
        MOVE 23 INTO TAXES
END-EVALUATE.

    Subprograms: In COBOL subprograms are written as an entirely separate program and include a linkage section which is what accepts data from the main program and can return data to the main program. You can pass data to a subprogram two
ways, REFERENCE or CONTENT. All local variables in a subprogram are static and are only refreshed if the subprogram call is preceded by a CANCEL or an INITIAL statement. 

    CALL ‘SUBPROGRAM’ USAGE BY CONTENT FNAME, MINT, LNAME

**Abstract Data Types and Encapsulation Constructs:** Cobol doesn’t have them.

**Support for Object-Oriented Programming:** COBOL has support for object-oriented programming through its newest update called Object-Oriented COBOL, however it’s as different as C/C++ and C#. OO COBOL gives you Classes, Objects, Methods, Inheritance, and Polymorphism. But since OO COBOL is still being changed/added on to and it’s like its own branch off language, that’s as far as OO COBOL will be discussed.

**Concurrency:** COBOL uses LOCKS and TRANSACTIONS. LOCKS are used to keep data from being changed while doing something else in the program or to allow only certain things, the statements READ WITH NO LOCK and START WITH NO LOCK, the rights to change the data. LOCK is generally used in companies to stop co-workers from changing the data while you are working with it, but still leaves access for workers above you to change the data. TRANSACTIONS are used to remember how the data was before the process started and is used to restore the
data if the TRANSACTION fails.

**Exception Handling:** COBOL has no integrated exception handling. A user can add in any exception handler that the user wants into the source code but the user has to self-implement them. Examples of this are ON-OVERFLOW, ON-????-ERROR, and AT-END*, each of this can be used for exception handling. However, the easiest way for exception handling is by using the IF and END-IF statement to catch any problem where you think one could happen. This does increase the lengthiness of COBOL programs but allows for specific handling and the IF and END-IF statement still has to be used with above stated handlers if you use the above handlers.

ex: IF ON-OVERFLOW(VALUE)
    DISPLAY OVERFLOW-ERROR
    STOP RUN
ELSE
    ...
END-IF.

**Evaluation:**

**Read ability:** While the data variables can be a little hard to read with the PIC clause. The rest of the COBOL language is very readable as it can be typed using words and
not symbols.

**Writability:** AS said before COBOL can have a near english syntax making writing in it feel more natural. You can even use periods to end a loop, if, when, etc statements. (All but END_PROGRAM) can be replaced. However it doesn’t have functions so you get repeated code unless you know the language really well, but you would lose some read ability.

**Reliability:** COBOL might not be the easiest or shortest or most powerful language but it does get the job done. Which is why after 50 years it is still going strong and is still used in most big businesses in the United States and the United Kingdom.

**Cost:** While COBOL is a relatively easy language to learn, not many people are learning it anymore. So some companies have to train their employees in how to write COBOL. But even with the additional cost of training some employees, the cost is still drastically below what it would cost to convert COBOL code to a more modern language.
Reference:
