Sean Williams
(seandwilliams@email.arizona.edu)
Michael Glenn
(mike1glenn@cox.net)
Aaron Valenzuela
(avalenzuela3@email.arizona.edu)

Cobol History

Common business-oriented language, or Cobol, is an English like computer language designed for business use. It was heavily based on the language FLOW-MATIC that was developed by Grace Hopper. Grace would later participate in the development of COBOL. COBOL is primarily used in business, finance, and administrative systems for companies and governments. COBOL was designed in 1959 by CODASYL(a steering committee sponsored by the US Department of Defense). It was designed to be a portable language for data processing. At the time, there was a need to lower the cost of programming and it was concluded that the creation of a business-oriented programming language would ease the cost of conversion. The CODASYL Executive Committee approved the specifications on 8 January 1960 and named it COBOL60. Since the government required computer manufacturers to provide it, it became available to everyone. It was standardized in 1968 when it was approved by ANSI as a standard commercial use language. There were several releases following the standardization that included fixing some early syntax ambiguity and implementing new features such as scope terminators and nested subprograms. Int 2002, Object-Oriented COBOL was published. Near the end of the 20th century COBOL

programming focused on solving the year 2000, or Y2K, problem. Many applications that relied on dates only stored 2 digits in decimal.

Control Structures

COBOL has some similar and some different control structures to c or java. However, the program structure is very different. Each COBOL program consists of 4 divisions. Divisions are specific areas of the program that contain information unique to each division. The 4 divisions are the Identification Division, Environment Division, Data Division, and Procedure Division. The Identification Division is the only mandatory division. It contains the paragraph PROGRAM-ID which is the only mandatory paragraph. PROGRAM-ID specifies the program name that can be 1 to 30 characters.

The Environment Division is used to specify input and output files to the program. It consists of two sections. The Configuration section which provides information about the system on which the program is written and executed. And the Input-Output section. This section provides information about the files to be used in the program.

The Data Division is used to define the variables used in the program. It consists of four sections. The File section, the Working-Storage section, the Local-Storage section and the Linkage section. The file section is used to define the record structure of the file. The Working-Storage section and the Local-Storage section are used to declare variables. the difference between the two sections is that variables in the Local-Storage section are initialized every time the program starts execution. The Linkage section is used to describe the data names that are received from an external program.

The Procedure division where the program possesses the variables it defined in the data division. It includes the logic and control structures of the program.

COBOL uses conditional statements in a similar way to modern programming languages. There is an if statement that tests a conditional and performs a block of code depending on the result of the conditional statement. There are several conditions an if statement can test. The most familiar is the relation condition. This includes things such as tests for equality, less than, and greater than. For numbers, relation conditions are calculated algebraically. For non-numeric operands, relation conditions are determined by the number of characters in each operand. In addition to the relation condition is the sign condition. It only tests numeric operands and tests if they are positive, negative, or zero. Class conditions test if an operand is a number or alphabetic. Negation of a condition is performed with the "NOT" keyword. Conditions can be combined with the "AND" and "OR" logical operators.

There is an additional type of condition called condition-name condition. It is a condition that is defined by the programmer. It is defined in the data division as a subordinate to another data item. It is defined with a name and a set of values. It can be used in an if statement by simply writing "IF <NAME>" where <NAME> is the name of the condition-name condition. It is evaluated by checking the value of the data item the condition is subordinate to against the values defined in the condition.

Loops are performed with the perform statement. Perform statements can be written in 4 ways and they always reference a paragraph to loop. Paragraphs are subsets of code that include at least one sentence. A sentence is a set of statements. Paragraphs when referenced by a perform statement act like labels of a goto statement. The first performance statement is

performed through. It specifies 2 paragraphs and runs code from the first paragraph specified thru to the last paragraph specified. Perform until behaves similarly to a while loop. It loops on a specified paragraph until a condition is met. Perform times performs a paragraph a set number of times. The last performance statement is perform varying. Perform varying behaves like a for loop. It includes a variable, a start value, an increment amount and an end condition. It loops on the paragraph and increments the variable until the variable has met the condition.

COBOL makes use of subroutines. There are two types of subroutines. Perform statements are actual Internal subroutines. There are also external subroutines that use the CALL verb. They behave similarly to functions and function calls. Variables to external subroutines can be called by reference or called by content. If a variable is called by reference and it's value is changed, the value will also be changed in the program that is called the subroutine. External subroutines are called by reference by default. If a variable is called by content and the value is changed then the change will not also occur in the program that is called the subroutine.

Data Types

Cobol has four data types available for the programmer to use: floats, ints, strings, and arrays. Arrays can hold items of the three other data types. The main difference between ints and floats in COBOL is that floats can be accurate to a decimal point of one’s liking as defined in the assignment with a 'V' followed by the number of integers being the level of precision. In COBOL, all variables that will be used in the program must be declared in the

DATA DIVISION WORKING-STORAGE SECTION. Below is an example of all four data types being initialized.

```
DATA
DIVISION.
                 WORKING-STORAGE SECTION.
                 01 ValueInt PIC 999.
                 01 ValueFloat PIC 9(3)V99.
                 01 ValueString PIC X(20) VALUE "Hello".
                 01 ARRAY-STRINGS.
                    05 ARRAY-ITEM PIC X(3) VALUE "Hi!" OCCURS 5 TIMES INDEXED
                BY I.
```

Integer and float data types are declared by reserving an amount of space for the amount of integers to be used for the data. In the first line, three spots are saved by writing PIC 999. This can also be done by 9(3) for the sake of simplicity when reading the code this will result in the value always being at least three characters in length even if the value is 1000 or 0 the result would in both cases be 000. The first variable, ValueInt, is an integer of up to three digits pre decimal and two digits post decimal. So, the largest value that can be stored in ValueInt is 999. Increasing the amount of 9s or the number in the parentheses increases the integer size of the variable. The same concept works for the float types except for the addition of the V character. In the second line, the number of 9s after V represents the number of decimal places to the right that is desired for this float variable. So ValueFloat will be calculated to the second decimal place (one hundredth place) reaching a max value of 999.99. There are two ways to perform arithmetic operations on integer and float data types:

```
ADD 35 TO ValueX GIVING ValueX.
COMPUTE ValueX = ValueX + 35
```

The two operations above both add 35 to a variable named ValueX. Dividing, multiplying and subtracting are done the same way for the COMPUTE function, just enter whatever operation you want done using the keyboard. ADD 35 TO ValueX is the operation that is being performed, and whatever variable is after the GIVING will save the result of that operation. There are also MULTIPLY, DIVIDE, and SUBTRACT functions following the same format, just replace ADD with the desired function and the order of operations are performed just as they are read.

The string data type above named ValueString contains the value “Hello”. Strings in COBOL can be concatenated. An example is shown below:

```
STRING ValueString DELIMITED BY SPACE "HELLO WORLD!" DELIMITED BY SIZE INTO
ValueString.
```

This is equivalent to doing: ValueString += “Hello World!”

So, the result of this statement would be: “HelloHello World!”

Arrays in COBOL have items placed in them when being initialized. In our array of strings above, “Hi!” is initialized as five times in the array by the line: VALUE “Hi!” OCCURS 5 TIMES INDEXED BY I. The ‘I’ character will be used as an index for the array. Arrays can be indexed using ARRAY-NAME(index_int). When indexing an array it is important to know that indices start at 1 and not 0 unlike other programming languages. The line below changes the first value in our string array to the string value “NEW”.

```
MOVE "NEW" TO ARRAY-ITEM(1)
```

To iterate through an array using a loop, use the variable name after the statement INDEXED BY when the array was first initialized to correctly start from the beginning of the array. An example of this is shown below.

```
PERFORM OUR-DISP VARYING I FROM 1 BY 1 UNTIL I>5
```

Cobol's Current Status

Although COBOL is over 60 years old, there are numerous banks that still use the language to this day. In the article, COBOL Is Everywhere, Who Will Maintain It?, David Cassel states that most ATM software is written in COBOL and that code written in COBOL handles about 3 trillion dollars daily to this day. The main reason for this is because COBOL is such an old language that numerous corporations and banks have been implementing COBOL into their systems for decades and switching to a different language can be expensive and time consuming. According to Cassel, even government agencies like the Department of Justice still use COBOL to this day for the purpose of tracking the population of inmates. The Veterans Association uses it to keep track of benefits claims, and the SSA uses it for the purpose of calculating retirement benefits. It seems ludicrous to run such important information on an outdated language but there's a good and expensive reason for doing so. According to Cassel, switching from COBOL to a different language cost the Commonwealth Bank of Australia $749.9 million dollars and took about five years to do so. This also causes the issue of a lack of COBOL programmers available to these companies because of the fact that it is such an old language. It is also difficult to read in that its control

flow can be similar to Assembly language, meaning the execution of the code can suddenly jump to a completely different part of the code making it non-sequential. Although it is outdated and the amount of people who know how to write COBOL is diminishing, COBOL still has a future ahead of it. IBM has made improvements to COBOL that allow it to be ran with JAVA on the same mainframe and the company also offers training programs for COBOL. Currently there are plenty of job offers for COBOL programmers that can be found on Google, so COBOL isn't going anywhere anytime soon.