COBOL CodeCount™

Counting Standard

University of Southern California

Center for Systems and Software Engineering

December , 2015

Revision Sheet

Date	Version	Revision Description	Author
10/01/2010	1.0	Initial Draft	Gurdeep Gahir, Virendra Chandak, Harsh Gupta
11/03/2010	1.1	Updated with examples	Gurdeep Gahir, Virendra Chandak, Harsh Gupta
12/228/2015	2.0	2015.12 first public release	Randy Maxwell

Table of Contents

No.	Contents	Page No.
1.0	Definitions	4
	1.1 Overview	4
	1.2 SLOC	4
	1.3 Physical SLOC	4
	1.4 Logical SLOC	4
	1.5 Data declaration line	4
	1.6 Compiler directive	5
	1.7 Executable keywords	5
	1.8 Blank line	5
	1.9 Comment line	5
	1.10 Executable line of code	6
2.0	Checklist for source statement counts	8
3.0	Examples of logical SLOC counting	9
	3.1 Executable Lines	9
	3.1.1 Selection Statements	9
	3.1.2 Iteration Statements	10
	3.1.3 Jump Statements	10
	3.1.4 Expression Statements	10
	3.1.5 Block Statements	11
	3.2 Declaration lines	11
	3.3 Compiler directives	11
4.0	Notes on Special Character Processing	13

1. Definitions

1.1. **Overview –** This document gives specific information of what the Unified Code Count (UCC) COBOL parser will do in terms of various keywords and other metrics. It is hoped that the metrics gathered by UCC will be of use to the COBOL community.

1.2. **SLOC –** Source Lines of Code is a unit used to measure the size of software program. SLOC counts the program source code based on a certain set of rules. SLOC is a key input for estimating project effort and is also used to calculate productivity and other measurements.

1.3. **Physical SLOC –** One physical SLOC is corresponding to one line starting with the first character and ending by a carriage return or an end-of-file marker of the same line, and which excludes the blank and comment line.

1.4. **Logical SLOC –** Lines of code intended to measure "statements", which normally terminate by a semicolon (C/C++, Java, C#) or a carriage return (VB, Assembly), etc. Logical SLOC are not sensitive to format and style conventions, but they are language-dependent. In COBOL, logical SLOC terminate by a period "." or carriage return.

1.5. **Data declaration line or data line –** A line that contains declaration of data and used by an assembler or compiler to interpret other elements of the program.

The following table lists the COBOL keywords that denote data declaration lines:

Figurative Constants		Picture Symbol
Space	High-Values	9
Spaces	Low-Value	X
Zero	Low-Values	A
Zeroes	All	V
Quote		S
Quotes		**Data Keywords**
High-Value		PIC
		PICTURE

Table 1 Data Declaration Types

1.6. Compiler Directives – A statement that tells the compiler how to compile a program, but not what to compile.

The following table lists the COBOL keywords that denote compiler directives:

BASIS	CBL	*CBL	CONTROL
*CONTROL	COPY	DEBUGGING	DELETE
EJECT	ENTER	INSERT	LABEL
PROCEDURE	PROCESS	READY	RELOAD
REPLACE	RESET	RESET TRACE	SERVICE
SERVICE RELOAD	SKIP1	SKIP2	SKIP3
TITLE	TRACE	USE	USING

Table 2 Compiler Directives

1.7. Executable Keywords – keywords that are reserved with predefined characteristics with respect to syntax and meaning (semantic or otherwise) to enable language specific features.

The following table lists the COBOL executable keywords:

break	case	catch	def	do
else	finally	for	if	match
new	return	super	this	throw
try	while			

Table 3 Executable Keywords

1.8. Blank Line – A physical line of code, which contains any number of white space characters (spaces, tabs, form feed, carriage return, line feed, or their derivatives).

1.9. Comment Line – A comment is defined as a string of zero or more characters that follow language-specific comment delimiter.

COBOL comment delimiters are ‘, *, and /. An asterisk (*) comment delimiter is printed in the output listing, immediately following the last preceding line. A slash (/) comment delimiter is printed on the first line of the next page, and the current page of the output listing is ejected.

1.10. Executable Line of code – A line that contains software instruction executed during runtime and on which a breakpoint can be set in a debugging tool. An instruction can be stated in a simple or compound form.

- An executable line of code may contain the following program control statements:
  - Arithmetic statements (ADD, COMPUTE, DIVIDE, MULTIPLY, SUBTRACT)
  - Data movement statements

ACCEPT	INITIALIZE	INSPECT	MOVE	SET
STRING	UNSTRING	XML GENERATE	XML PARSE	

  - Decision statements (IF, EVALUATE)
  - Input-output statements

ACCEPT	CLOSE	DELETE	DISPLAY	OPEN
READ	REWRITE	START	STOP	WRITE

  - Ordering statements (MERGE, RELEASE, RETURN, SORT)
  - Program or method linkage statements (CALL, INVOKE)
  - Program or method statements (CALL, CANCEL, INVOKE)
  - Table-handling statement (SEARCH)
  - Iteration statements (PERFORM, TIMES … PERFORM, UNTIL … PERFORM, VARYING … PERFORM)
  - Empty statements (CONTINUE)
  - Jump statements (EXIT, GO TO, PERFORM, ALTER)
  - Ending statements (STOP RUN, EXIT PROGRAM, EXIT METHOD, GOBACK)

- An executable line of code may not contain the following statements:
    - Compiler directives
    - Data declaration (data) lines
    - Whole line comments, including empty comments and banners
    - Blank lines

2. Checklist for source statement counts

PHYSICAL SLOC COUNTING RULES			
MEASUREMENT UNIT	ORDER OF PRECEDENCE	PHYSICAL SLOC	COMMENTS
Executable Lines	1	One Per line	Defined in 1.8
Non-executable Lines			
Declaration (Data) lines	2	One per line	Defined in 1.4
Compiler Directives	3	One per line	Defined in 1.5
Comments			Defined in 1.7
On their own lines	4	Not Included (NI)	
Embedded	5	NI	
Empty Comments	6	NI	
Blank Lines	7	NI	Defined in 1.6

LOGICAL SLOC COUNTING RULES				
NO.	STRUCTURE	ORDER OF PRECEDENCE	LOGICAL SLOC RULES	COMMENTS
R01	IF-THEN-ENDIF, IF-THEN-ELSE-ENDIF, IF-THEN-ELSEIF-ENDIF	1	COUNT ONCE	
R02	“PERFORM ..TIMES”, “PERFORM.. UNTIL” STATEMENT	2	COUNT ONCE	
R03	STATEMENTS ENDING BY “.”	3	COUNT ONCE PER STATEMENT	
R04	COMPILER DIRECTIVES	4	COUNT ONCE PER DIRECTIVE	

3. Examples

Executable Lines

SELECTION Statement

ESS1 – if, else if, else and nested if statements

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

IF <CONDITION>
THEN
<STATEMENTS>
END-IF
	IF VarG = 14 THEN
	DISPLAY "First"
	END-IF
		1
		1
		0

IF <CONDITION>
THEN
<STATEMENTS>
ELSE
<STATEMENTS>
END-IF
	IF VarG = 14 THEN
	DISPLAY "First"
	ELSE
	DISPLAY "Second"
	END-IF
		1
		1
		0
		1
		0

IF <CONDITION>
THEN
<STATEMENTS>
ELSEIF <CONDITION2>
<STATEMENTS>
ELSE
<STATEMENTS>
END-IF
	IF VarG = 14 THEN
	DISPLAY "First"
	ELSE IF VarG = 15
	DISPLAY "Second"
	ELSE
	DISPLAY “Third”
	END-IF
		1
		1
		1
		1
		0
		1
		0

ESS2 – EVALUATE and nested EVALUATE statements

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
EVALUATE <EXPRESSION> ALSO <EXPRESSION> WHEN <CONDITIONS> END-EVALUATE	EVALUATE TRUE ALSO Position WHEN L-Arrow ALSO 2 THRU 10 SUBTRACT 1 FROM Position WHEN R-Arrow ALSO 1 THRU 9 ADD 1 TO Position END-EVALUATE	1 1 1 1 1 0

ITERATION Statement

EIS1 – perform

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
PERFORM <PROC> <COUNT> TIMES END PERFORM	PERFORM S1 3 TIMES END PERFORM	1 0
PERFORM <COUNT> TIMES <STATEMENTS> END PERFORM	PERFORM 3 TIMES DISPLAY "Finished in” END PERFORM	1 1 0
PERFORM WITH TEST <BEFORE/AFTER> UNTIL <CONDITION> END PERFORM	PERFORM WITH TEST BEFORE UNTIL EndOfStudentFile DISPLAY "Finished in” END-PERFORM	1 1 1 0

JUMP Statement

EJS1 – GOTO

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
GOTO <LABEL>	GOTO READX	1

EJS2 – EXIT

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
EXIT	EXIT	0

EXPRESSION Statement

EES1 – perform

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
PERFORM <PROC>	PERFORM S1	1

EES2 – perform thru

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
PERFORM <PROC1> THRU <PROC2>	PERFORM S1 THRU S3	1

EES3 – empty statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
one or more "." in succession	.	1 per each

BLOCK Statement

EBS1 – block = related statements treated as a unit

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
COBOL does not have block statements		

COBOL Declarations or Data Lines

DDL1 – elementary items

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
01 <NAME> PIC <RANGE> VALUE <VALUE>	01 GrossPay PIC 9(5)V99 VALUE ZEROS.	1

DDL2 – group items

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<LEVEL> <NAME> <LEVEL> <NAME> (RANGE) . . . . . <LEVEL> <NAME> (RANGE)	01 StudentDetails. 02 StudentId PIC 9(7). 02 StudentName. 03 FirstName PIC X(1). 03 MiddleInitial PIC X. 03 Surname PIC X(5). 02 DateOfBirth. 03 DayOfBirth PIC 99. 03 MonthOfBirth PIC 99. 03 YearOfBirth PIC 9(4). 02 CourseCode PIC X(4).	0 1 0 1 1 1 0 1 1 1 1

COBOL Compiler Directives

CDL1 – USE FOR

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
USE FOR DEBUGGING	USE FOR DEBUGGING	1

CDL2 – USE AFTER

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
USE <ITEM> AFTER <EXCEPTIONTYPE> EXCEPTION/ERROR	USE GLOBAL AFTER STANDARD EXCEPTION	1

4. Notes on Special Character Processing

<Extra points related to special character processing for the current language document>