DOS-Batch CodeCount™ Counting Standard

University of Southern California

Center for Systems and Software Engineering

December , 2016

Revision Sheet

Date	Version	Revision Description	Author
7/27/2016	1.0	Original Release	Derek Lengenfelder
Date	Version No.	Click here to Description.	Name

Table of Contents

1. Definitions

1.1. **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.

1.2. **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.

1.3. **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.

1.4. **Data declaration line or data line –** A line that contains declaration of data and used by a compiler or assembler to interpret other elements of the program. None exists in DOS batch file.

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

The following table lists the DOS Batch keywords that denote compiler directive lines:

NOTE: The "@" prefix to a command tells the interpreter how handle the output of the command.

@[DOS COMMAND]

Table 1 DOS Batch Compiler Directive

1.6. **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).

1.7. **Comment Line –** A comment is defined as a string of zero or more characters that follow language-specific comment delimiter. DOS Batch comment delimiters are "::" or "--". Batch adopts five ways of commenting. We count two of them. One is using REM command, this is the only documented way to insert a comment. REM this line should be counted as one SLOC line. The other way is using :: label. This is a non-documented comment line. A whole comment line may span one line and does not contain any compilable source code. An embedded comment can co-exist with compilable source code on the same physical line. Banners and empty comments are treated as types of comments.

1.8. **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:

- Selection statements (if)
- Iteration statements (for)
- Jump statements (goto, exit)
- Expression statements (assignment statements)

An executable line of code may not contain the following statements:

- Compiler directives
- 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	
Banners	6	NI	
Empty comments	7	NI	
Blank lines	8	NI	Defined in 1.6

LOGICAL SLOC COUNTING RULES				
NO.	STRUCTURE	ORDER OF PRECEDENCE	LOGICAL SLOC RULES	COMMENTS
R01	“for” or “if” statement	1	Count once	
R02	Compiler Directive	5	Count once per directive	

3. Examples

EXECUTABLE LINES

SELECTION Statement

ESS1 - if, else, and nested if statements

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
if (<boolean expression>) <statement>;	if (x != 0) echo “non-zero”	1 1
if (<boolean expression>) <statement>; else <statement>;	if (x > 0) echo “positive” else echo “negative”	2 1

if (<boolean expression>)
(
  <statements>;
)
else
(
  <statements>;
)
	if (x == 0)
	(
	  echo “zero”
	)
	else
	(
	  echo “not zero”
	)
		1
		0
		1
		0
		0
		0
		1
		0

NOTE: complexity is not considered, i.e. multiple “&&” or “||” as part of the expression.	if ((x != 0) && (x > 0)) echo %x%	1 1

ITERATION Statement

EIS1 – for loops

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
FOR %%A IN (list) DO command [parameters] <statement>	FOR %%A IN (1 2 3) DO ECHO %A	2
FOR %%A IN (list) DO command [parameters] (<statements>)	FOR %%A IN (1 2 3) DO (ECHO %A)	1 0 1 0

JUMP Statement

EJS1 – goto and label statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
GOTO label	loop1: set /a "x =x+1" IF x lt y GOTO loop1	0 1 2
EJS2 – exit function		
GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
EXIT	IF (x lt 0) EXIT	2
EXPRESSION Statement		
EES1 - assignment statement		
GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
SET _variable=one two three	SET var = abc efg	1
SET "_variable=one & two"	SET "var = abd & efg"	1
SET _variable= "one & two"	SET var = "abd & efg"	1
SET /A "_result=2+4"	SET /A "var = 2 + 4"	1

COMPILER DIRECTIVES

CDL1 – directive type

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
@[COMMAND]	@ECHO OFF	1

4. Complexity

Complexity measures the occurrences of different keywords in code baseline. Below table identifies the categories and their respective keywords that are counted as part of the complexity metrics.

Table 2 – Complexity Keywords List

Calculations	Conditionals	Logic	Pre-processor	Assignment
+	==	if	@[COMMAND]	[SET] [NAME] =
-	equ	else		
*	neq	for		
/	lss			
%	leq			
<<	gtr			
>>	geq			
&				
|				
~				