MATLAB CodeCount™ Counting Standard

University of Southern California

Center for Systems and Software Engineering

August , 2012

Revision Sheet

Date	Version	Revision Description	Author
7/26/2012	1.0	Original Release	CSSE

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, comma, or a carriage return. 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 an assembler or compiler to interpret other elements of the program.

MATLAB uses implicitly defined types so that there are no data declaration statements.

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 MATLAB keywords that denote compiler directive lines:

import

Table 1 Compiler Directives

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.

Matlab single line comment delimiter is "%". Anything included between "%{" and "%}" is considered part of a block comment. A whole comment line may span one line and does not contain any compliable source code. An embedded comment can co-exist with compliable 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, elseif, switch)
  - Iteration statements (for, while, parfor)
  - Error control statements (try-catch block)
  - Jump statements (break, continue)
  - Expression statements (function calls, assignment statements, operations, etc.)
  - Block 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			
Compiler Directives	2	One per line	Defined in 1.5
Comments			Defined in 1.7
On their own lines	3	Not Included (NI)	
Embedded	4	NI	
Banners	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	“for”, “while”, “parfor”, or “if” statement	1	Count Once	Loops and conditionals are independent statements.
R02	Statements ending by a semicolon or comma	2	Count Once	Semicolons and commas within matrix assignments are not counted.
R03	Line terminated by new line character and last symbol is not ellipsis “…”	3	Count Once	End of command
R04	Compiler Directive	4	Count once per directive	

3. Examples

EXECUTABLE LINES

SELECTION Statement

ESS1 – if, elseif, else and nested if statements

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
if <boolean expression> <statements> end	if rem(4, 2) == 0 disp('4 is even') end	1 1 0

if <boolean expression>
  <statements>
else
  <statements>
end
	if x > 0
	  disp ('x is positive')
	else
	  disp ('x is zero')
	end
		1
		1
		0
		1
		0

if <boolean expression>
  <statements>
elseif <boolean expression>
  <statements>
.
.
else
  <statements>
end
	if x > 0
	  disp ('x is positive')
	elseif x < 0
	  disp ('x is negative')
	else
	  disp ('x is zero')
	end
		1
		1
		1
		1
		0
		1
		0

NOTE: complexity is not considered, i.e. multiple "&&" or "||" as part of the expression.	if x != 0 && x > 0 disp ('x') end	1 1 0

ESS2 – switch and nested switch statements

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

switch <expression>
  case <constant 1>
  <statements>
  case <constant 2>
  <statements>
  case <constant 3>
  <statements>
  otherwise
  <statements>
end
	switch input_num
	  case -1
	  disp ('negative one');
	  case 0
	  disp ('zero');
	  case 1:
	  disp ('positive one');
	  otherwise
	  disp ('other value');
	end
		1
		0
		1
		0
		1
		0
		1
		0
		1
		0

ESS3 – try-catch

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

try
  <statements>
catch <exception-declaration>
  <statements>
end
	try
	  fid = fopen('abc', 'r');
	  d_in = fread(fid);
	catch exception
	  rethrow(exceptioin)
	end
		1
		1
		1
		1
		1
		0

ITERATION Statement

EIS1 – for

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
for <index> = <start>:<increment>:<end> <statements> end	for k = 1:2:24 C{k} = k * 2; end for x = 1:10 x end	1 1 0 2 1

EIS2 – while

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

while <boolean expression>
  <statements>
end
	n = 1;
	while prod(1:n) < 1e100
	  n = n + 1;
	end
		1
		1
		1
		0

EIS3 – parfor

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
parfor <index> = <start>:<end> <statements> end	parfor i = 1:length(A) B(i) = f(A(i)); end	1 1 0

JUMP Statement

EJS1 – return

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
return	If i == 0 return; end	1 1 0

EJS2 – break

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
break	if i > 10 break; end	1 1 0

EJS3 – continue

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
continue	if i < 5 continue; end	1 1 0

EXPRESSION Statement

EES1 – function call

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<function_name> (<parameters>)	surf(peaks)	1

EES2 – assignment statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<name> = <value>	X = [1 2 3 4]; Y = X;	1 1

COMPILER DIRECTIVES

CDL1 – directive types

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
import <package>	Import packagename.ClassName	1