Fortran CodeCount™ Counting Standard

University of Southern California

Center for Systems and Software Engineering

March , 2010

Revision Sheet

Date	Version	Revision Description	Author
12/3/2007	1.0	Original Release	CSSE
3/23/2010	1.1	UCC Update	CSSE
1/2/2013	1.2	Updated document template	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 (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 the case of FORTRAN, there is no semicolon terminating every single executable line.

The number of logical SLOC within a source file is defined to be the sum of the number of logical SLOCs classified as compiler directives, data lines, or executable lines. It excludes comments (whole or embedded) and blank lines. Thus, a line containing two or more source statements count as multiple logical SLOCs. A single logical statement that extends over five physical lines counts as one logical SLOC. Specifically, the logical SLOC found within a file containing software written in the FORTRAN 77 programming language is equivalent to the number of physical lines less the number of physical continuation lines.

The logical SLOC found within a file containing software written in the FORTRAN 90 programming language may be computed by

(1) Counting the number of separator semicolons, i.e., non-terminal semicolons used to separate multiple logical statements on the same physical line,
(2) Adding the number of physical lines,
(3) Subtracting the number of physical continuation lines, then
(4) Subtracting the number of physical lines that only contain the following keywords USE, CYCLE, CASE, CONTAINS, PUBLIC, PRIVATE, ELSE END INTERFACE, END TYPE, END CASE, END IF, END DO, END SELECT, END WHERE, END SUBROUTINE, END FUNCTION, END PROGRAM, and END MODULE.

The logical SLOC definition was selected due to

(1) Compatibility with parametric software cost modeling tools, and
(2) Ability to support software metrics collection.

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.

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

Data Types	**FORTRAN 77**	**FORTRAN 90**
SUBROUTINE	X	X
FUNCTION	X	X
VIRTUAL	X	
BLOCK DATA	X	
DOUBLE COMPLEX	X	
COMMON	X	X
NAMELIST	X	X
IMPLICIT	X	X
INTEGER	X	X
REAL	X	X
COMPLEX	X	X
CHARACTER	X	X
DATA	X	X
PARAMETER	X	X
DIMENSION	X	X
DOUBLE PRECISION	X	X
EQUIVALENCE	X	X
RECORD	X	
STRUCTURE	X	
BYTE	X	
LOGICAL	X	X
EXTERNAL	X	X
INTRINSIC	X	X
UNION	X	
MAP	X	
VOLATILE	X	
SAVE	X	X
PROGRAM		X
MODULE		X
CONTAINS		X
ASSIGN		X
USE		X
TYPE	X	X
INTERFACE		X
ALLOCATE		X
REALLOCATE		X
DEALLOCATE		X
NULLIFY		X
OPTIONAL		X
PUBLIC		X
PRIVATE		X
POINTER		X
TARGET		X

Table 1 Data Type Keywords

Execute Keywords	FORTRAN 77	FORTRAN 90
GOTO	X	X
IF	X	X
ELSE WHERE		X
ELSE IF		X
ELSE		X
DO	X	X
CYCLE		X
SELECT		X
CASE		X
WHERE		X
CALL	X	X
FORMAT	X	X
READ	X	X
PRINT	X	X
WRITE	X	X
INQUIRE		X
OPEN		X
CLOSE	X	X
REWIND	X	X
REWRITE	X	
BACKSPACE		X
ENTRY	X	X
EXIT		X
PAUSE	X	X
RETURN	X	X
STOP	X	X
ACCEPT	X	
CONTINUE	X	

Table 2 Executable Statements Keywords

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 Fortran keywords that denote complier directive lines:

Compiler Directive Keywords	FORTRAN 77	FORTRAN 90
OPTIONS	X	X
INCLUDE	X	X
DICTIONARY	X	X

Table 3 Compiler Directive Keywords

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.

There are several ways to indicate comments in Fortran. The table below illustrates the different methods.

COMMENTS	
Rule	Example
A line that begins with the letter "c" or "C" or an asterisk "*" in the first column of the line	C This is a comment c This is a comment * This is a comment
All characters following an exclamation mark "**!**", except in a character string, are comments.	Year = Year + 1 ! This is a comment
Blank lines are not counted	

Table 4 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 (do-enddo)
  - Jump statements (return, goto, break, continue, exit function)
  - Expression statements (function calls, assignment statements, operations, etc.)
  - Block statements
- 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 2.9
Non-executable lines			
Declaration (Data) lines	2	One per line	Defined in 2.4
Compiler directives	3	One per line	Defined in 2.5
Comments			Defined in 2.7
On their own lines	4	Not included (NI)	
Embedded	5	NI	
Empty comments	6	NI	
Blank lines	7	NI	Defined in 2.7

LOGICAL SLOC COUNTING RULES

NO.	STRUCTURE	ORDER OF PRECEDENCE	LOGICAL SLOC RULES	COMMENTS
R01	"*do*-x", "while-do" combination, "*if*", "elseif" statement	1	Count once per structure	
R02	Compiler directive	2	Count once per directive	
R03	data declaration and data assignment	3	Count once per declaration/assignment	
R04	Jump statement	4	Count once per keyword	
R05	Function/Subroutine call	5	Count once per call	
R06	Semicolon in statement	6	Count once per semicolon	
R07	Multiple statements in a line	7	Count one per executable statement	

3. Examples

EXECUTABLE LINES

SELECTION Statement

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

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
if (*logical expression*) *executable statement*	if (x .lt. 0) x = -x	2
or	or	
if (*logical expression*) then	if (x .ge. y) then	1
statements	write(*,*) 'x'	1
else	else	0
statements	write(*,*) 'x'	1
endif	endif	0
or	or	
if (*logical expression*) then	if (x .gt. 0) then	1
statements	if (x .ge. y) then	1
elseif (*logical expression*) then	write(*,*) 'x'	1
statements	else	0
:	write(*,*) 'x'	1
:	endif	0
:	elseif (x .lt. 0) then	1
:	write(*,*) 'x is neg'	1
else	else	0
statements	write(*,*) 'x is zero'	1
endif	endif	0

ESS2 – select and nested select statements

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
Datatype :: selector	integer :: Class	1
select case (selector)	select case (Class)	1
case (label-list-1)	case (1)	0
statements-1	write(*,*) 'Freshman'	1
case (label-list-2)	case (2)	0
statements-2	write(*,*) 'Sophomore'	1
:	case (3)	0
:	write(*,*) 'Junior'	1
case (label-list-n)	case (4)	0
statements-n	write(*,*) 'Senior'	1
case default	case default	0
statements-default	write(*,*) "no class"	1
end select	end select	0

ITERATION Statement

EIS1 – do-enddo

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
do *label var = expr1, expr2, expr3*	do 20 i = 10, 1, -2	1
statements	write(*,*) 'i =', i	1
label continue	20 continue	1
or	or	
do *var = expr1, expr2, expr3*	do i = 10, 1, -2	1
statements	write(*,*) 'i =', i	1
enddo	enddo	0
or	or	
	i = 10	1
do	do	1
statements	write(*,*) 'i =', i	1
if *(logical expr)* exit	if (i < 1) exit	2
statements	i = i - 2	1
enddo	enddo	0
where *expr1* specifies the initial value of *var*, *expr2* is the terminating bound, and *expr3* is the increment (step).		

EIS2 – do-while

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
do *label while (logical-expr)*	do 20 while (i == 10)	1
statements	write(*,*) 'i =', i	1
label continue	20 continue	1
or	or	
do while *(logical expr)*	do while (i == 10)	1
statements	write(*,*) 'i =', i	1
enddo	enddo	0

JUMP Statement

EJS1 – goto label

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
label if *(logical expr)* then	10 if (n .le. 100) then	1
statements	n = 2 * n	1
:	write (*,*) n	1
goto *label*	goto 10	1
endif	endif	0

EJS2 – cycle

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
cycle *label*	outer: do i = 1, n	1
	middle: do j = 1, m	1
Or	inner: do k = 1, l	1
	:	
cycle	:	
	:	
	if (a(i, j, k) < 0) exit outer	2
	if (j == 5) cycle middle	2
	if (i == 5) cycle	2
	:	
	:	
	:	
	enddo inner	0
	enddo middle	0
	enddo outer	0

EJS3 – exit

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
exit *label*	exit 10	1
or	or	
exit	exit	1

EJS4 – continue

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
label continue	*20* continue	1
statements	k = 3	1
if *(logical expr)* goto *label*	if *(k==3)* goto *20*	2

EXPRESSION Statement

EES1 – function call or procedure call

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<function_name> (<parameters>)	cos(4)	1
call <subroutine_name> (<parameters>)	call avg(a, b, c)	1

EES2 – assignment statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<name> = <value>	a = 174.5	1
<name> = <value>; <name> = <value>;	a = 2; b = 7; c = 3	3

EES3 – empty statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
one or more “;” in succession	a = 2;	1 per each

Declaration or Data Lines

DDL1 – function declaration subroutine declaration variable declaration type declaration

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

type function *name (list-of-variables)*
   declarations
   statements
    :
    :
   return
end
	function fact(n)
	   fact = 1
	   do 10 j = 2, n
	      fact = fact * j
	   10 continue
	   return
	end
		1
		1
		1
		1
		1
		1
		0

subroutine *name (list-of-arguments)*
   declarations
   statements
   return
end
	subroutine iswap(a, b)
	   integer a, b
	   integer tmp
	   tmp = a
	   a = b
	   b = tmp
	   return
	end
		1
		1
		1
		1
		1
		1
		1
		0

<type> <name>	real a	1
type ( )	type (person) chairman	1

Compiler Directives

CDL1 – directive types

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
options *options*	options /CHECK=ALL/F77	1
include *character-literal-constant*	include 'my_common_blocks'	1
dictionary *character-literal-constant*	dictionary 'salary'	1