Pascal CodeCount™ Counting Standard

University of Southern California

Center for Systems and Software Engineering

January, 2011

Revision Sheet

Date	Version	Revision Description	Author
1/5/11	1.0	Original Release	CSSE

1.0 CHECKLIST FOR SOURCE STATEMENT COUNTS

PHYSICAL AND LOGICAL SLOC COUNTING RULES

Measurement Unit	Order of Precedence	Physical SLOC	Logical SLOC	Comments
Executable lines	1	One per line	See table below	Defined in 2.9
Non-executable lines				
Declaration (Data) lines	2	One per line	See table below	Defined in 2.4
Compiler directives	3	One per line	See table below	Defined in 2.5
Comments				Defined in 2.8
On their own lines	4	Not included (NI)	NI	
Embedded	5	NI	NI	
Banners	6	NI	NI	
Empty comments	7	NI	NI	
Blank lines	8	NI	NI	Defined in 2.7

Table 1 Physical and Logical SLOC Counting Counts

LOGICAL SLOC COUNTING RULES

No.	Structure	Order of Precedence	Logical SLOC Rules	Comments
R01	"*for*", "*while*" or "*if*" statement	1	Count once.	"*while*" is an independent statement.
R02	*repeat* {…} *until* (…); statement	2	Count once.	
R03	Statements ending by a semicolon	3	Count once per statement, including empty statement.	
R04	Block delimiters, begin..end;	4	Count once per set	
R05	Compiler directive	5	Count once per directive.	

Table 2 Logical SLOC Counting Rules

2.0 DEFINITIONS

2.1 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.

2.2 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.

2.3 Logical SLOC – Lines of code intended to measure “statements”, which normally terminate by a semicolon. Logical SLOC are not sensitive to format and style conventions, but they are language-dependent.

2.4 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 Pascal keywords that denote data declaration lines:

Simple Data Types	Compound and User Defined Data Types	Access Specifiers	Type Qualifiers
`boolean`	`array`	`private`	`const`
`byte`	`type`	`protected`	`volatile`
`bytebool`		`public`	
`cardinal`			
`char`			
`comp`			
`complex`			
`double`			
`extended`			
`integer`			
`int64`			
`longint`			
`real`			
`shortint`			
`single`			
`smallint`			
`string`			
`word`			

Table 3 Data Declaration Types

NOTE: See Section 3 of this document for examples of data declaration lines.

2.5 Compiler directive - A statement that tells the compiler how to compile a program, but not what to compile.

A list of common Pascal directives is presented in the table below:

`$define`	`$ifndef`	`$include`	`$I`
`$undef`	`$else`	`$CSDefine`	`$M`
`$if`	`$W`	`$macro`	`$setc`
`$ifdef`	`$endif`	`$error`	

Table 4 Compiler Directives

NOTE: See Section 3 of this document for examples of compile directive lines.

2.6 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).

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

Pascal comment delimiters are "(*..*)", "{..}", and "***//***". A whole comment line may span one or more lines 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.

2.8 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, case)
  - Iteration statements (for, while, repeat, with)
  - Empty statements (one or more ";")
  - Jump statements (goto, exit function)
  - Expression statements (function calls, assignment statements, operations, etc.)
  - Block statements

  NOTE: See Section 3 of this document for examples of control 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

3.0 EXAMPLES OF LOGICAL SLOC COUNTING

EXECUTABLE LINES

SELECTION STATEMENTS

ID	STATEMENT DESCRIPTION	GENERAL FORM	SPECIFIC EXAMPLE	SLOC COUNT

ESS1	if, else and nested if statements	if (<boolean expression>) then
		      <statement>;
		
		if (<boolean expression>) then
		   <statement>
		else
		   <statement>;
		
		if (<boolean expression>) then
		   begin
		      <statements>
		   end;
		
		NOTE: complexity is not considered, i.e. multiple “and” or “or” as part of the expression.
			if (x <> 0) then
			   writeln (‘non-zero’);
			
			if (x > 0) then
			   writeln (‘positive’)
			else
			   writeln (‘negative’);
			
			if (x = 0) then
			   begin
			      writeln (‘The answer is:’);
			      writeln (‘zero’);
			   end;
				1
				1
				
				1
				1
				0
				1
				
				1
				0
				1
				1
				0

ESS2	case statements	case <expression> of
		   <constant 1> : <statement>;
		   <constant 2> : <statement>;
		   else (or otherwise) <statement>;
		end;
			case Number of
			     1..10 : writeln (‘small num’);
			   11..100 : writeln (‘large num’);
			   else writeln (‘HUGE num’);
			end;
				1
				1
				1
				1
				0

ESS3	try-except/finally	try
		   <statements>;
		except or finally
		   <handlers>;
		end;
			try
			   z := doDiv (X,Y);
			except
			   On EDivException do z := 0;
			end;
				1
				1
				1
				1
				0

ITERATIONS STATEMENTS				
ID	**STATEMENT DESCRIPTION**	**GENERAL FORM**	**SPECIFIC EXAMPLE**	**SLOC COUNT**
EIS1	for-do	for <control> := <initial> to <final> do	for i := 0 to 10 do	1
		<statement>;	writeln (‘at ‘, i);	1
			for i := 0 to 10 do	1
			begin	0
			DoSomething;	1
			writeln (‘at ‘, i);	1
			end;	0
EIS2	while-do	while (<boolean expression>) do	while (i < 10) do	1
		<statement>;	writeln (i);	1
EIS3	repeat-until	repeat	repeat	0
		<statement>;	writeln (i);	1
		<statement>	i := i + 1	1
		until (<boolean expression>);	until (i > 10);	1
EIS4	with-do	with (<identifier>) do	with Info do	1
		begin	begin	0
		<statement>;	Age := 18;	1
		end;	Zip := 90210;	1
			end;	0

JUMP STATEMENTS
(ARE COUNTED AS THEY INVOKE ACTION – PASS TO THE NEXT STATEMENT)

ID	STATEMENT DESCRIPTION	GENERAL FORM	SPECIFIC EXAMPLE	SLOC COUNT

EJS1	goto, label	goto *label*;
		.
		.
		label:
			loop1:
			x := x + 1;
			if (x < y) then goto loop1;
				0
				1
				2

EJS2	exit function	void exit (int return_code);	if (x < 0) then exit (1);	2

EXPRESSION STATEMENTS

ID	STATEMENT DESCRIPTION	GENERAL FORM	SPECIFIC EXAMPLE	SLOC COUNT
EES1	function call	<function_name> (<parameters>);	readfile (‘filename’);	1

EES2	assignment statement	<name> := <value>;	x := y;
			a := 1; b := 2; c := 3;
				1
				3

EES3	empty statement (is counted as it is considered to be a placeholder for something to call attention)	one or more “;” in succession	;	1 per each

BLOCK STATEMENTS

ID	STATEMENT DESCRIPTION	GENERAL FORM	SPECIFIC EXAMPLE	SLOC COUNT

EBS1	block = related statements treated as a unit	begin
		<statements>;
		end;
			begin
			i := 0;
			writeln (i);
			end;
				1
				1
				1
				0

DECLARATION (DATA) LINES

ID	STATEMENT DESCRIPTION	GENERAL FORM	SPECIFIC EXAMPLE	SLOC COUNT

DDL1	function prototype, variable declaration,
	
	record declaration
		function <function_name> (<parameters>);
		
		<name> : <type>;
		
		<type> = record
		    <statements>;
		end;
			function readfile (name : string);
			
			amount : real;
			
			point = record
			   x, y, z : real;
			end;
				1
				
				1
				
				1
				1
				0

COMPILER DIRECTIVES

ID	STATEMENT DESCRIPTION	GENERAL FORM	SPECIFIC EXAMPLE	SLOC COUNT
CDL1	directive types	{$<directive>}	{$ifdef}	1