Pascal CodeCount™ Counting Standard

University of Southern California

Center for Systems and Software Engineering

January , 2011

Revision Sheet

Date	Version	Revision Description	Author
1/5/2011	1.0	Original Release	CSSE
1/2/2013	1.1	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.

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 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 1 Data Declaration Types

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 Pascal keywords that denote compiler directive lines:

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

Table 2 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.

Pascal comment delimiters are "(*..*)", "{..}", and "//". 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, 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
- 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	
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", "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	

3. Examples

EXECUTABLE LINES

SELECTION Statement

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

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
if (<boolean expression>) then <statement>;	if (x <> 0) then writeln ('non-zero');	1 1

if (<boolean expression>) then
  <statement>
else
  <statement>;
	if (x > 0) then
	  writeln ('positive')
	else
	  writeln ('negative');
		1
		1
		0
		1

if (<boolean expression>) then
  begin
  <statements>
  end;
	if (x = 0) then
	  begin
	  writeln ('The answer is:');
	  writeln ('zero');
	  end;
		1
		0
		1
		1
		0

NOTE: complexity is not considered, i.e. multiple "and" or "or" as part of the expression.		

ESS2 – case statements

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

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

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

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

ITERATION Statement

EIS1 – for-do

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

for <control> := <initial> to <final> do
  <statement>;
	for i := 0 to 10 do
	  writeln ('at ', i);
	
	for i := 0 to 10 do
	  begin
	  DoSomething;
	  writeln ('at ', i);
	  end;
		1
		1
		
		1
		0
		1
		1
		0

EIS2 – while-do

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
while (<boolean expression>) do <statement>;	while (i < 10) do writeln (i);	1 1

EIS3 – repeat-until

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

repeat
  <statement>;
  <statement>
until (<boolean expression>);
	repeat
	  writeln (i);
	  i := i + 1
	until (i > 10);
		0
		1
		1
		1

EIS4 – with-do

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

with (<identifier>) do
  begin
  <statement>;
  end;
	with Info do
	  begin
	  Age := 18;
	  Zip := 90210;
	  end;
		1
		0
		1
		1
		0

JUMP Statement

EJS1 – goto, label

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

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

EJS2 – exit function

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
void exit (int return_code);	if (x < 0) then exit (1);	2

EXPRESSION Statement

EES1 – function call

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<function_name> (<parameters>);	readfile ('filename');	1

EES2 – assignment statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<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)

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

begin
<statements>;
end;
	begin
	i := 0;
	writeln (i);
	end;
		1
		1
		1
		0

DECLARATION OR DATA LINES

DDL1 – function prototype, variable declaration, record declaration

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
function <function_name> (<parameters>);	function readfile (name : string);	1
<name> : <type>;	amount : real;	1
<type> = record <statements>; end;	point = record x, y, z : real; end;	1 1 0

COMPILER DIRECTIVES

CDL1 – directive types

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
{$<directive>}	{$ifdef}	1