Pascal CodeCount™ Counting Standard

University of Southern California

Center for Systems and Software Engineering

December , 2016

Revision Sheet

Date	Version	Revision Description	Author
7/13/2016	1.0	Original Release	Derek Lengenfelder
8/12/2016	1.1	Updated appendix and table contents	Matthew Swartz

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:

ansistring	array	boolean	byte	bytebool
cardinal	char	class	comp	complex
const	double	extended	file	integer
interface	Int64	longbool	longint	longword
object	pchar	qword	real	record
set	shortint	shortstring	single	smallint
string	type	widestring	word	wordbool
private	protected	public	volatile	

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 data declaration lines:

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

Table 2 Compiler Directives

A full list of compiler directives can be found here http://www.freepascal.org/docs-html/prog/progch1.html

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 "(*...*)" and "{…}". 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, ? operator, switch)
  - Iteration statements (for, while, do-while)
  - Empty statements (one or more ";")
  - 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 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	
Embedded	5	Not Included	
Banners	6	Not included	
Empty comments	7	Not included	
Blank lines	8	Not Included	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 Statements

ESS1 – if-else if-else and nested if statements

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

if (<boolean expression>)
  <statement>;
 	
if (<boolean expression>) then
  <statement>
else
  <statement>;
 	
if (<boolean expression>) then
  begin
  <statement>;
  end;
 	
if (<boolean expression>)
  <statements>;
else if (<boolean expression>)
  <statements>;
 	
NOTE: complexity is not considered, i.e. multiple "&&" or "||" as part of the expression.
	if () then
	  writeln('non-zero');
	
	if (x > 0) then
	  writeln('non-negative');
	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

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

case <expression> of
  <constant 1> : <statement>;
  break;
  <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…or negative');
	end;
		1
		1
		1
		1
		
		0

ESS3 – try-except/finally blocks

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;
		0
		1
		0
		2
		0

ITERATION Statements

EIS1 – for-do Loop

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

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

EIS2 – While-do Loop

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

EIS3 – repeat until Loop

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 Loop

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 Statements

(are counted as they invoke action-pass to the next statement)

EJS1 – goto,label statements

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 statements

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

EXPRESSION Statements

EES1 – function and procedure 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

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
one or more ";" in succession	;;;;	0

BLOCK Statements

EBS1 – simple blocks

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

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

DECLARATION OR DATA LINES

DDL1 – function prototype, variable declaration, record declaration

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
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

CDL1 – directive types

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

Math Functions	Trig	Log	Calculations	Conditionals	Logic	Pre-processor	Assignment
abs	arcos	ln	+	if	=	NOTE:	:=
arg	arcsin		-	else	<>	See	
cmplx	arctan		*	for	>	“Compiler	
dec	cos		/	repeat	<	directive”	
exp	sin		div	while	>=		
im			mod	with	=<		
inc					not		
min					and		
max					or		
polar					xor		
pow					shl		
re					shr		
round							
sqr							
sqrt							