PHP CodeCount™

Counting Standard

University of Southern California

Center for Systems and Software Engineering

April , 2010

Revision Sheet

Date	Version	Revision Description	Author
6/22/2007	1.0	Original Release	CSSE
11/8/2007	1.1	Updated	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, PHP) 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.

Each variable is defined with a dollar sign ($) before the variable's name. In addition, like many lines of PHP code, a semicolon is used. Semicolons do not, however, need to be placed at the end of commented lines. Strings, or a combination of characters, are defined with quotation marks around the value, while integers are not.

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

Data Declaration
$
Basic Data Types
boolean
integer
float
string
array
object
resource
NULL

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 PHP keywords that denote compiler directive lines:

define	declare
include	include_once
require	require_once

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.

PHP comment delimiters are "//", "#", 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, foreach, 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 (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”, “foreach”, “while” or “if” statement	1	Count Once	“while” is an independent statement.
R02	*do {…} while (…); statement*	2	Count Once	Braces {…} and semicolon ; used with this statement are not counted.
R03	Statements ending by a semicolon	3	Count once per statement, including empty statement	Semicolons within “for” statement are not counted. Semicolons used with R01 and R02 are not counted.
R04	Block delimiters, braces {…}	4	Count once per pair of braces {..}, except where a closing brace is followed by a semicolon, i.e. };or an opening brace comes after a keyword “else”.	Braces used with R01 and R02 are not counted. Function definition is counted once since it is followed by {…}.
R05	Compiler Directive	5	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>)	if ($x != 0)	1
<statements>;	echo “non-zero”;	1
if (<boolean expression>) :	if ($x != 0):	1
<statements>;	echo “non-zero”;	1
endif;	endif;	0
if (<boolean expression>)	if ($x == 0)	1
<statements>;	echo “zero”;	1
elseif (<boolean expression>)	elseif ($x > 0)	1
<statements>;.	echo “positive”;	1
.	else	0
.	echo “negative”;	1
else <statements>;		
if (<boolean expression>)	if ($x != 0)	1
{	{	0
<statements>;	echo “non-zero”;	1
}	}	0
else	else	0
{	{	0
<statements>;	echo “zero”;	1
}	}	0
if (<boolean expression>) :	if ($x != 0):	1
<statements>;	echo “non-zero”;	1
else:	else:	0
<statements>;	echo “zero”;	1
endif;	endif;	0

ESS2 – ? operator

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
Exp1?Exp2:Exp3	x > 0 ? echo “+” : echo “-”;	1

ESS3 – switch and nested switch statements

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
switch (<expression>)	switch (number)	1
{	{	0
case <constant 1> :	case 1:	0
<statements>;	foo1();	1
break;	break;	1
case <constant 2> :	case 2:	0
<statements>;	foo2();	1
break;	break;	1
default:	default:	0
<statements>;	echo “invalid case”;	1
}	}	0
switch (<expression>):	switch (number):	1
case <constant 1> :	case 1:	0
<statements>;	foo1();	1
break;	break;	1
case <constant 2> :	case 2:	0
<statements>;	foo2();	1
break;	break;	1
default:	default:	0
<statements>;	echo “invalid case”;	1
endswitch;	endswitch;	0

ESS4 – try-catch

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
try	try	0
{	{	0
// code that could throw	echo "Calling function";	1
// an exception	throw Exception(“Error”);	1
}		0
catch (exception-declaration)	MyFunc();	1
{	}	0
// code that executes when	catch (IOException $e)	1
// exception-declaration is thrown	{	0
// in the try block	echo “Error: ” . $e;	1
}	}	0

ITERATION Statement

EIS1 – for

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

for (initialization; condition; increment)
    <statements>;
	for (i = 0; i < 10; i++)
	   echo $i . "</br>";
		1
		1

	for (i = 0; i < 10; i++)
	{
	   echo $i . "</br>";
	}
		1
		0
		1
		0

for (initialization; condition; increment):
    <statements>;
endfor;
	for (i = 0; i < 10; i++):
	   echo $i . "</br>";
	endfor;
		1
		1
		0

EIS2 – empty statements (could be used for time delays)

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
for ($i = 0; $i < SOME_VALUE; $i++) ;	for ($i = 0; $i < 10; $i++) ;	2

EIS3 – while

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

while (<boolean expression>)
    <statements>;
	while ($i < 10)
	{
	   echo $i . "</br>";
	   $i++;
	}
		1
		0
		1
		1
		0

while (<boolean expression>):
    <statements>;
endwhile;
	while ($i < 10):
	   echo $i . "</br>";
	   $i++;
	endwhile;
		1
		1
		1
		0

EIS4 – do-while

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

do
{
   <statements>;
} while (*<boolean expression>*);
	do
	{
	   echo $i;
	   $i++;
	 } while ($i > 0);
		0
		0
		1
		1
		1

EIS5 – foreach

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

foreach (array_expression as $value)
   <statements>;
	$arr = array(1, 2, 3, 4);
	foreach ($arr as &$value) {
	   $value = $value * 2;
	}
		1
		1
		1
		0

foreach (array_expression as $value):
   <statements>;
endforeach;
	foreach ($arr as &$value):
	   $value = $value * 2;
	endforeach;
		1
		1
		0

	$employeeAges; $employeeAges["Lisa"] = "28"; $employeeAges["Grace"] = "34";	1 1 1

foreach (array_expression as $key => $value)
   <statements>;
	foreach($employeeAges as $key => $value)
	{
	   echo "Name: $key, Age: $value <br />";
	}
		1
		0
		1
		0

JUMP Statement

EJS1 - return

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
return expression	if ($i==0) return;	2

EJS2 – goto, label

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
goto label;	loop1: $x++; if ($x < $y) goto loop1;	0 1 2

EJS3 - break

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
break;	if ($i > 10) break;	2

EJS4 – exit function

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
void exit (int return_code);	if ($x < 0) exit (“Exit!”);	2

EJS5 – continue

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
continue;	while (list($key, $value) = each($arr)) {	1
	if (!($key % 2)) {	1
	continue;	1
	}	0
	do_something_odd($value);	1
	}	0

EXPRESSION Statement

EES1 – function call

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<function_name> (<parameters>);	read_file ($name);	1

EES2 – assignment statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<name> = <value>;	$x =$ y;	1
	$var = ‘Joe’;	1
	$a = 1; $b = 2; $c = 3;	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
/* start of block */	/* start of block */	0
{	{	0
<definitions>	$i = 0;	1
<statement>	echo $i;	1
}	}	1
/* end of block */	/* end of block */	0

Declaration or Data Lines

DDL1 – function prototype, variable declaration

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
functionname($*var1*,$*var2*,...,$*varX*) {	function prod($a,$b) {	1
<statements>	$hello = "Hello World!";	1
<statements>	$a_number = 4;	1
<statements>	$anotherNumber = 8;	1
}	}	0
$<name>;	$hello;	1

Compiler Directives

CDL1 – directive types

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
include <library_name>	include(test.php);	1
include_once<library_name>	include_once(foo.php);	1
require<library_name>	require(testfile.php);	1
require_once<library_name>	require_once(filename.php);	1
bool define (string $name, mixed $value [, bool $case_insensitive])	define("CONSTANT", "Hello");	1
declare (directive)	declare(ticks=2) {	1
statement	for ($x = 1; $x < 50; ++$x) {	1
	echo similar_text(md5($x), md5($x*$x)), "
;";	1
	}	0
	}	0