Polyspace Bug Finder

Detailed Report for Project: Algorithm

Report Author: HP

Polyspace Bug Finder: Detailed Report for Project: Algorithm

by Report Author: HP

Published 29-Sep-2019 23:25:16

Analysis Author(s): HP
Polyspace Version(s): 2.6 (R2018b)
Project Version(s): 1.0

Result Folder(s):
E:\git\Algorithm_Library\00_ploysoaces\Module_1\BF_Result

Table of Contents

Chapter 1. Polyspace Bug Finder Summary

Table 1.1. Project Summary

	Count	Reviewed	Unreviewed	Pass/Fail
MISRA:2012 Checker	5	0	5	NA
Defects	0	0	0	NA
Total	5	0	5	NA

Table 1.2. Summary By File

File	Defects (Reviewed)	MISRA C:2012 Guidelines (Reviewed)
E:\git\Algorithm_Library\queue\queue.c	0 (0)	5 (0)
E:\git\Algorithm_Library\queue\queue.h	0 (0)	0 (0)

Chapter 2. MISRA C:2012 Guidelines

MISRA C:2012 Summary - Violations by File

File	Total Violations
E:\git\Algorithm_Library\queue\queue.c	5
Total	5

MISRA C:2012 Violations

Table 2.1. E:\git\Algorithm_Library\queue\queue.c

ID	Guideline	Message	Function	Severity	Status	Comment
6	D4.14	The validity of values received from external sources shall be checked. Dereferenced pointer is from an unsecure source. Pointer may be NULL or may point to unknown memory.	Que_Init()	Unset	Unreviewed	
7	D4.14	The validity of values received from external sources shall be checked. Dereferenced pointer is from an unsecure source. Pointer may be NULL or may point to unknown memory.	Que_Write()	Unset	Unreviewed	
8	D4.14	The validity of values received from external sources shall be checked. Dereferenced pointer is from an unsecure source. Pointer may be NULL or may point to unknown memory.	Que_OverWrite()	Unset	Unreviewed	
9	D4.14	The validity of values received from external sources shall be checked. Dereferenced pointer is from an unsecure source. Pointer may be NULL or may point to unknown memory.	Que_Read()	Unset	Unreviewed	
10	D4.14	The validity of values received from external sources shall be checked. Dereferenced pointer is from an unsecure source. Pointer may be NULL or may point to unknown memory.	Que_OnlyRead()	Unset	Unreviewed	

Chapter 3. Defects

Defects

No defects were found.

Chapter 4. Appendix 1 - Configuration Settings

Polyspace Settings

Option	Value
-author	HP
-bug-finder	true
-c-version	c11
-compiler	generic
-date	29/09/2019
-dos	true
-I	E:\git\Algorithm_Library\queue
-import-comments	E:\git\Algorithm_Library\00_ploysoaces\Module_1\BF_Result\comments_bak
-lang	C
-misra3	mandatory-required
-prog	Algorithm
-results-dir	E:\git\Algorithm_Library\00_ploysoaces\Module_1\BF_Result
-target	i386
-to	Software Safety Analysis level 2
-verif-version	1.0
-checkers	ALIGNMENT_CHANGE, ASSERT, ATOMIC_VAR_ACCESS_TWICE, ATOMIC_VAR_SEQUENCE_NOT_ATOMIC, BAD_EQUAL_EQUAL_U... - See Ref #1

Ref #1:
ALIGNMENT_CHANGE, ASSERT, ATOMIC_VAR_ACCESS_TWICE, ATOMIC_VAR_SEQUENCE_NOT_ATOMIC, BAD_EQUAL_EQUAL_USE, BAD_EQUAL_USE, BAD_FREE, BAD_LOCK, BAD_PTR_SCALING, BAD_UNLOCK, CHARACTER_MISUSE, CHAR_EOF_CONFUSED, CLOSED_RESOURCE_USE, CONSTANT_OBJECT_WRITE, DATA_RACE, DATA_RACE_STD_LIB, DEADLOCK, DEAD_CODE, DECL_MISMATCH, DOUBLE_DEALLOCATION, DOUBLE_LOCK, DOUBLE_RESOURCE_CLOSE, DOUBLE_RESOURCE_OPEN, DOUBLE_UNLOCK, ERRNO_MISUSE, FILE_OBJECT_MISUSE, FLEXIBLE_ARRAY_MEMBER_STRUCT_MISUSE, FLOAT_ABSORPTION, FLOAT_CONV_OVFL, FLOAT_STD_LIB, FLOAT_ZERO_DIV, FREED_PTR, FUNC_CAST, IMPROPER_ARRAY_INIT, INLINE_CONSTRAINT_NOT_RESPECTED, INT_CONV_OVFL, INT_STD_LIB, INT_ZERO_DIV, INVALID_ENV_POINTER, INVALID_MEMORY_ASSUMPTION, INVALID_VA_LIST_ARG, IO_INTERLEAVING, LOCAL_ADDR_ESCAPE, MACRO_USED_AS_OBJECT,

MEMCMP_PADDING_DATA, MEMCMP_STRINGS, MEM_STD_LIB, MISSING_ERRNO_RESET, MISSING_NULL_CHAR, MISSING_RETURN, NON_INIT_PTR, NON_INIT_VAR, NON_POSITIVE_VLA_SIZE, NULL_PTR, OPERATOR_PRECEDENCE, OTHER_STD_LIB, OUT_BOUND_ARRAY, OUT_BOUND_PTR, PARTIALLY_ACCESSED_ARRAY, PRE_DIRECTIVE_MACRO_ARG, PRE_UCNAME_JOIN_TOKENS, PTR_CAST, PTR_SIZEOF_MISMATCH, PTR_TO_DIFF_ARRAY, PUTENV_AUTO_VAR, READ_ONLY_RESOURCE_WRITE, RESOURCE_LEAK, SIDE_EFFECT_IGNORED, SIGN_CHANGE, SIG_HANDLER_CALLING_SIGNAL, SIG_HANDLER_COMP_EXCP_RETURN, SIG_HANDLER_ERRNO_MISUSE, SIG_HANDLER_SHARED_OBJECT, SIZEOF_MISUSE, STD_FUNC_ARG_MISMATCH, STREAM_WITH_SIDE_EFFECT, STRING_FORMAT, STRLIB_BUFFER_OVERFLOW, STRLIB_BUFFER_UNDERFLOW, STR_FORMAT_BUFFER_OVERFLOW, STR_STD_LIB, TEMP_OBJECT_ACCESS, TOO_MANY_VA_ARG_CALLS, TYPEDEF_MISMATCH, UINT_CONV_OVFL, UNPROTOTYPED_FUNC_CALL, UNREACHABLE, USELESS_IF, USELESS_WRITE, VAR_SHADOWING, VA_ARG_INCORRECT_TYPE

Coding Rules Configuration

Guidelines prefixed with a "D" are directives.

Table 4.1. MISRA C:2012 Configuration

Guideline	Description	Mode	Comment	Enabled
D1.1	Any implementation-defined behaviour on which the output of the program depends shall be documented and understood.	required		yes
D2.1	All source files shall compile without any compilation errors.	required		yes
D3.1	All code shall be traceable to documented requirements.	required	not enforceable	no
D4.1	Run-time failures shall be minimized.	required		yes
D4.2	All usage of assembly language should be documented.	advisory	not enforceable	no
D4.3	Assembly language shall be encapsulated and isolated.	required		yes
D4.4	Sections of code should not be "commented out".	advisory	not enforceable	no
D4.5	Identifiers in the same name space with overlapping visibility should be typographically unambiguous.	advisory		no
D4.6	typedefs that indicate size and signedness should be used in place of the basic numerical types.	advisory		no
D4.7	If a function returns error information, then that error information shall be tested.	required		yes
D4.8	If a pointer to a structure or union is never dereferenced within a translation unit, then the implementation of the object should be hidden.	advisory		no
D4.9	A function should be used in preference to a function-like macro where they are interchangeable.	advisory		no
D4.10	Precautions shall be taken in order to prevent the contents of a header file being included more than once.	required		yes
D4.11	The validity of values passed to library functions shall be checked.	required		yes
D4.12	Dynamic memory allocation shall not be used.	required		no

Guideline	Description	Mode	Comment	Enabled
D4.13	Functions which are designed to provide operations on a resource should be called in an appropriate sequence.	advisory		no
D4.14	The validity of values received from external sources shall be checked.	required	MISRA C:2012 Amendment 1	yes
1.1	The program shall contain no violations of the standard C syntax and constraints, and shall not exceed the implementation's translation limits.	required		yes
1.2	Language extensions should not be used.	advisory		no
1.3	There shall be no occurrence of undefined or critical unspecified behaviour.	required		yes
2.1	A project shall not contain unreachable code.	required		yes
2.2	There shall be no dead code.	required		yes
2.3	A project should not contain unused type declarations.	advisory		no
2.4	A project should not contain unused tag declarations.	advisory		no
2.5	A project should not contain unused macro declarations.	advisory		no
2.6	A function should not contain unused label declarations.	advisory		no
2.7	There should be no unused parameters in functions.	advisory		no
3.1	The character sequences /* and // shall not be used within a comment.	required		yes
3.2	Line-splicing shall not be used in // comments.	required		yes
4.1	Octal and hexadecimal escape sequences shall be terminated.	required		yes
4.2	Trigraphs should not be used.	advisory		no
5.1	External identifiers shall be distinct.	required		yes
5.2	Identifiers declared in the same scope and name space shall be distinct.	required		yes
5.3	An identifier declared in an inner scope shall not hide an identifier declared in an outer scope.	required		yes
5.4	Macro identifiers shall be distinct.	required		yes
5.5	Identifiers shall be distinct from macro names.	required		yes
5.6	A typedef name shall be a unique identifier.	required		yes
5.7	A tag name shall be a unique identifier.	required		yes
5.8	Identifiers that define objects or functions with external linkage shall be unique.	required		yes
5.9	Identifiers that define objects or functions with internal linkage should be unique.	advisory		no
6.1	Bit-fields shall only be declared with an appropriate type.	required		yes
6.2	Single-bit named bit fields shall not be of a signed type.	required		yes
7.1	Octal constants shall not be used.	required		yes
7.2	A "u" or "U" suffix shall be applied to all integer constants that are represented in an unsigned type.	required		yes

Guideline	Description	Mode	Comment	Enabled
7.3	The lowercase character "l" shall not be used in a literal suffix.	required		yes
7.4	A string literal shall not be assigned to an object unless the object's type is "pointer to const-qualified char".	required		yes
8.1	Types shall be explicitly specified.	required		no
8.2	Function types shall be in prototype form with named parameters.	required		yes
8.3	All declarations of an object or function shall use the same names and type qualifiers.	required		yes
8.4	A compatible declaration shall be visible when an object or function with external linkage is defined.	required		yes
8.5	An external object or function shall be declared once in one and only one file.	required		yes
8.6	An identifier with external linkage shall have exactly one external definition.	required		yes
8.7	Functions and objects should not be defined with external linkage if they are referenced in only one translation unit.	advisory		no
8.8	The static storage class specifier shall be used in all declarations of objects and functions that have internal linkage.	required		yes
8.9	An object should be defined at block scope if its identifier only appears in a single function.	advisory		no
8.10	An inline function shall be declared with the static storage class.	required		yes
8.11	When an array with external linkage is declared, its size should be explicitly specified.	advisory		no
8.12	Within an enumerator list, the value of an implicitly-specified enumeration constant shall be unique.	required		yes
8.13	A pointer should point to a const-qualified type whenever possible.	advisory		no
8.14	The restrict type qualifier shall not be used.	required		yes
9.1	The value of an object with automatic storage duration shall not be read before it has been set.	mandatory		yes
9.2	The initializer for an aggregate or union shall be enclosed in braces.	required		yes
9.3	Arrays shall not be partially initialized.	required		yes
9.4	An element of an object shall not be initialized more than once.	required		yes
9.5	Where designated initializers are used to initialize an array object the size of the array shall be specified explicitly.	required		yes
10.1	Operands shall not be of an inappropriate essential type.	required		yes
10.2	Expressions of essentially character type shall not be used inappropriately in addition and subtraction operations.	required		yes
10.3	The value of an expression shall not be assigned to an object with a narrower essential type or of a different essential type category.	required		yes
10.4	Both operands of an operator in which the usual arithmetic conversions are performed shall have the same essential type category.	required		yes
10.5	The value of an expression should not be cast to an inappropriate essential type.	advisory		no

Guideline	Description	Mode	Comment	Enabled
10.6	The value of a composite expression shall not be assigned to an object with wider essential type.	required		yes
10.7	If a composite expression is used as one operand of an operator in which the usual arithmetic conversions are performed then the other operand shall not have wider essential type.	required		yes
10.8	The value of a composite expression shall not be cast to a different essential type category or a wider essential type.	required		yes
11.1	Conversions shall not be performed between a pointer to a function and any other type.	required		yes
11.2	Conversions shall not be performed between a pointer to an incomplete type and any other type.	required		yes
11.3	A cast shall not be performed between a pointer to object type and a pointer to a different object type.	required		yes
11.4	A conversion should not be performed between a pointer to object and an integer type.	advisory		no
11.5	A conversion should not be performed from pointer to void into pointer to object.	advisory		no
11.6	A cast shall not be performed between pointer to void and an arithmetic type.	required		yes
11.7	A cast shall not be performed between pointer to object and a non-integer arithmetic type.	required		yes
11.8	A cast shall not remove any const or volatile qualification from the type pointed to by a pointer.	required		yes
11.9	The macro NULL shall be the only permitted form of integer null pointer constant.	required		yes
12.1	The precedence of operators within expressions should be made explicit.	advisory		no
12.2	The right hand operand of a shift operator shall lie in the range zero to one less than the width in bits of the essential type of the left hand operand.	required		yes
12.3	The comma operator should not be used	advisory		no
12.4	Evaluation of constant expressions should not lead to unsigned integer wrap-around.	advisory		no
12.5	The sizeof operator shall not have an operand which is a function parameter declared as "array of type".	mandatory	MISRA C:2012 Amendment 1	yes
13.1	Initializer lists shall not contain persistent side effects.	required		yes
13.2	The value of an expression and its persistent side effects shall be the same under all permitted evaluation orders.	required		yes
13.3	A full expression containing an increment (++) or decrement (--) operator should have no other potential side effects other than that caused by the increment or decrement operator.	advisory		no
13.4	The result of an assignment operator should not be used.	advisory		no
13.5	The right hand operand of a logical && or || operator shall not contain persistent side effects.	required		yes
13.6	The operand of the sizeof operator shall not contain any expression which has potential side effects.	mandatory		yes
14.1	A loop counter shall not have essentially floating type.	required		yes
14.2	A for loop shall be well-formed.	required		yes
14.3	Controlling expressions shall not be invariant.	required		yes

Guideline	Description	Mode	Comment	Enabled
14.4	The controlling expression of an if statement and the controlling expression of an iteration-statement shall have essentially Boolean type.	required		yes
15.1	The goto statement should not be used.	advisory		no
15.2	The goto statement shall jump to a label declared later in the same function.	required		yes
15.3	Any label referenced by a goto statement shall be declared in the same block, or in any block enclosing the goto statement.	required		yes
15.4	There should be no more than one break or goto statement used to terminate any iteration statement.	advisory		no
15.5	A function should have a single point of exit at the end.	advisory		no
15.6	The body of an iteration-statement or a selection-statement shall be a compound-statement.	required		yes
15.7	All if ... else if constructs shall be terminated with an else statement.	required		yes
16.1	All switch statements shall be well-formed.	required		yes
16.2	A switch label shall only be used when the most closely-enclosing compound statement is the body of a switch statement.	required		yes
16.3	An unconditional break statement shall terminate every switch-clause.	required		yes
16.4	Every switch statement shall have a default label.	required		yes
16.5	A default label shall appear as either the first or the last switch label of a switch statement.	required		yes
16.6	Every switch statement shall have at least two switch-clauses.	required		yes
16.7	A switch-expression shall not have essentially Boolean type.	required		yes
17.1	The features of <stdarg.h> shall not be used.	required		yes
17.2	Functions shall not call themselves, either directly or indirectly.	required		yes
17.3	A function shall not be declared implicitly.	mandatory		no
17.4	All exit paths from a function with non-void return type shall have an explicit return statement with an expression.	mandatory		yes
17.5	The function argument corresponding to a parameter declared to have an array type shall have an appropriate number of elements.	advisory		no
17.6	The declaration of an array parameter shall not contain the static keyword between the [ ].	mandatory		yes
17.7	The value returned by a function having non-void return type shall be used.	required		yes
17.8	A function parameter should not be modified.	advisory		no
18.1	A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand.	required		yes
18.2	Subtraction between pointers shall only be applied to pointers that address elements of the same array.	required		yes

Guideline	Description	Mode	Comment	Enabled
18.3	The relational operators >, >=, < and <= shall not be applied to objects of pointer type except where they point into the same object.	required		yes
18.4	The +, -, += and -= operators should not be applied to an expression of pointer type.	advisory		no
18.5	Declarations should contain no more than two levels of pointer nesting.	advisory		no
18.6	The address of an object with automatic storage shall not be copied to another object that persists after the first object has ceased to exist.	required		yes
18.7	Flexible array members shall not be declared.	required		yes
18.8	Variable-length array types shall not be used.	required		yes
19.1	An object shall not be assigned or copied to an overlapping object.	mandatory		yes
19.2	The union keyword should not be used.	advisory		no
20.1	#include directives should only be preceded by preprocessor directives or comments.	advisory		no
20.2	The ', " or \ characters and the /* or // character sequences shall not occur in a header file name.	required		yes
20.3	The #include directive shall be followed by either a <filename> or "filename"sequence.	required		yes
20.4	A macro shall not be defined with the same name as a keyword.	required		yes
20.5	#undef should not be used.	advisory		no
20.6	Tokens that look like a preprocessing directive shall not occur within a macro argument.	required		yes
20.7	Expressions resulting from the expansion of macro parameters shall be enclosed in parentheses.	required		yes
20.8	The controlling expression of a #if or #elif preprocessing directive shall evaluate to 0 or 1.	required		yes
20.9	All identifiers used in the controlling expression of #if or #elif preprocessing directives shall be #define'd before evaluation.	required		yes
20.10	The # and ## preprocessor operators should not be used.	advisory		no
20.11	A macro parameter immediately following a # operator shall not immediately be followed by a ## operator.	required		yes
20.12	A macro parameter used as an operand to the # or ## operators, which is itself subject to further macro replacement, shall only be used as an operand to these operators.	required		yes
20.13	A line whose first token is # shall be a valid preprocessing directive.	required		yes
20.14	All #else, #elif and #endif preprocessor directives shall reside in the same file as the #if, #ifdef or #ifndef directive to which they are related.	required		yes
21.1	#define and #undef shall not be used on a reserved identifier or reserved macro name.	required		yes
21.2	A reserved identifier or macro name shall not be declared.	required		yes
21.3	The memory allocation and deallocation functions of <stdlib.h> shall not be used.	required		yes
21.4	The standard header file <setjmp.h> shall not be used.	required		yes
21.5	The standard header file <signal.h> shall not be used.	required		yes

Guideline	Description	Mode	Comment	Enabled
21.6	The Standard Library input/output functions shall not be used.	required		yes
21.7	The atof, atoi, atol, and atoll functions of <stdlib.h> shall not be used.	required		yes
21.8	The library functions abort, exit and system of <stdlib.h> shall not be used.	required		yes
21.9	The library functions bsearch and qsort of <stdlib.h> shall not be used.	required		yes
21.10	The Standard Library time and date functions shall not be used.	required		yes
21.11	The standard header file <tgmath.h> shall not be used.	required		yes
21.12	The exception handling features of <fenv.h> should not be used.	advisory		no
21.13	Any value passed to a function in <ctype.h> shall be representable as an unsigned char or be the value EOF.	mandatory	MISRA C:2012 Amendment 1	yes
21.14	The Standard Library function memcmp shall not be used to compare null terminated strings.	required	MISRA C:2012 Amendment 1	yes
21.15	The pointer arguments to the Standard Library functions memcpy, memmove and memcmp shall be pointers to qualified or unqualified versions of compatible types.	required	MISRA C:2012 Amendment 1	yes
21.16	The pointer arguments to the Standard Library function memcmp shall point to either a pointer type, an essentially signed type, an essentially unsigned type, an essentially Boolean type or an essentially enum type.	required	MISRA C:2012 Amendment 1	yes
21.17	Use of the string handling functions from <string.h> shall not result in accesses beyond the bounds of the objects referenced by their pointer parameters.	mandatory	MISRA C:2012 Amendment 1	yes
21.18	The size_t argument passed to any function in <string.h> shall have an appropriate value.	mandatory	MISRA C:2012 Amendment 1	yes
21.19	The pointers returned by the Standard Library functions localeconv, getenv, setlocale or, strerror shall only be used as if they have pointer to const-qualified type.	mandatory	MISRA C:2012 Amendment 1	yes
21.20	The pointer returned by the Standard Library functions asctime, ctime, gmtime, localtime, localeconv, getenv, setlocale or strerror shall not be used following a subsequent call to the same function.	mandatory	MISRA C:2012 Amendment 1	yes
22.1	All resources obtained dynamically by means of Standard Library functions shall be explicitly released.	required		yes
22.2	A block of memory shall only be freed if it was allocated by means of a Standard Library function.	mandatory		yes
22.3	The same file shall not be open for read and write access at the same time on different streams.	required		yes
22.4	There shall be no attempt to write to a stream which has been opened as read-only.	mandatory		yes
22.5	A pointer to a FILE object shall not be dereferenced.	mandatory		yes
22.6	The value of a pointer to a FILE shall not be used after the associated stream has been closed.	mandatory		yes
22.7	The macro EOF shall only be compared with the unmodified return value from any Standard Library function capable of returning EOF.	required	MISRA C:2012 Amendment 1	yes
22.8	The value of errno shall be set to zero prior to a call to an errno-setting-function.	required	MISRA C:2012 Amendment 1	yes
22.9	The value of errno shall be tested against zero after calling an errno-setting-function.	required	MISRA C:2012 Amendment 1	yes
22.10	The value of errno shall only be tested when the last function to be called was an errno-setting-function.	required	MISRA C:2012 Amendment 1	yes

Chapter 5. Appendix 2 - Definitions

Table 5.1. Abbreviations

Abbreviation	Definition
NA	Not Available