GRAMMATECH

MISRA C:2012 GUIDELINES FOR THE USE OF THE C LANGUAGE IN CRITICAL SYSTEMS (MARCH 2013) | CODESONAR® 7.2

INCLUDING MISRA C:2012 AMENDMENT 1 ADDITIONAL SECURITY GUIDELINES FOR MISRA C:2012 (APRIL 2016)

TRUSTED LEADERS OF SOFTWARE ASSURANCE AND ADVANCED CYBER-SECURITY SOLUTIONS

WWW.GRAMMATECH.COM

INTRODUCTION

The MISRA C:2012 standard aims to foster safety, reliability, and portability of programs written inISO C for embedded systems. It is used in a wide range of industries, including automotive, aero-space, medical devices, and industrial control.

CodeSonar 7.2 includes a large number of warning classes that support checking for the MISRA C:2012 guidelines. Every CodeSonar warning report includes the numbers of any MISRA C:2012 rules and directives that are closely mapped to the warning's class. (The close mapping for a warning class is the set of categories—including MISRA C:2012 rule and directive numbers—that most closely match the class, if any).

You can configure CodeSonar to enable and disable warning classes mapped to specific MISRA C:2012 rules and directives, or use build presets to enable all warning classes that are closely mapped to any MISRA C:2012 rules and directives. In addition, you can use the CodeSonar search function to find warnings related to specific MISRA C:2012 rules or directives, or to any MISRA C:2012 rule or directive.

For more information on MISRA C:

https://www.misra.org.uk/MISRAChome/tabid/181/Default.aspx

MISRA C:2012 CLOSE MAPPING (CODESONAR V7.2)

The following table contains CodeSonar classes that are closely mapped to specific MISRA C:2012 rules and directives.

Rule	Rule Name	Category	Decidability	Supported
Misra2012:1.1	The program shall contain no violations of the standard C syntax and constraints, and shall not exceed the implementation's translation limits	Required	Decidable	No
Misra2012:1.2	Language extensions should not be used	Advisory	Undecidable	Yes
Misra2012:1.3	There shall be no occurrence of undefined or critical unspecified behaviour	Required	Undecidable	Yes
Misra2012:1.4	Emergent language features shall not be used	Required	Decidable	No
Misra2012:2.1	A project shall not contain unreachable code	Required	Undecidable	Yes
Misra2012:2.2	There shall be no dead code	Required	Undecidable	Yes
Misra2012:2.3	A project should not contain unused type declarations	Advisory	Decidable	Yes
Misra2012:2.4	A project should not contain unused tag declarations	Advisory	Decidable	Yes
Misra2012:2.5	A project should not contain unused macro declarations	Advisory	Decidable	Yes
Misra2012:2.6	A function should not contain unused label declarations	Advisory	Decidable	Yes
Misra2012:2.7	There should be no unused parameters in functions	Advisory	Decidable	Yes
Misra2012:3.1	The character sequences /* and // shall not be used within a comment	Required	Decidable	Yes
Misra2012:3.2	Line-splicing shall not be used in // comments	Required	Decidable	Yes
Misra2012:4.1	Octal and hexadecimal escape sequences shall be terminated	Required	Decidable	Yes
Misra2012:4.2	Trigraphs should not be used	Advisory	Decidable	Yes
Misra2012:5.1	External identifiers shall be distinct	Required	Decidable	Yes
Misra2012:5.2	Identifiers declared in the same scope and name space shall be distinct	Required	Decidable	Yes
Misra2012:5.3	An identifier declared in an inner scope shall not hide an identifier declared in an outer scope	Required	Decidable	Yes
Misra2012:5.4	Macro identifiers shall be distinct	Required	Decidable	Yes
Misra2012:5.5	Identifiers shall be distinct from macro names	Required	Decidable	Yes
Misra2012:5.6	A typedef name shall be a unique identifier	Required	Decidable	Yes
Misra2012:5.7	A tag name shall be a unique identifier	Required	Decidable	Yes
Misra2012:5.8	Identifiers that define objects or functions with external linkage shall be unique	Required	Decidable	Yes
Misra2012:5.9	Identifiers that define objects or functions with internal linkage should be unique	Advisory	Decidable	Yes
Misra2012:6.1	Bit-fields shall only be declared with an appropriate type	Required	Decidable	Yes
Misra2012:6.2	Single-bit named bit fields shall not be of a signed type	Required	Decidable	Yes
Misra2012:7.1	Octal constants shall not be used	Required	Decidable	Yes
Misra2012:7.2	A "u" or "U" suffix shall be applied to all integer constants that are represented in an unsigned type	Required	Decidable	Yes
Misra2012:7.3	The lowercase character "l" shall not be used in a literal suffix	Required	Decidable	Yes
Misra2012:7.4	A string literal shall not be assigned to an object unless the object's type is "pointer to const-qualified char"	Required	Decidable	Yes
Misra2012:8.1	Types shall be explicitly specified	Required	Decidable	No
Misra2012:8.2	Function types shall be in prototype form with named parameters	Required	Decidable	Yes
Misra2012:8.3	All declarations of an object or function shall use the same names and type qualifiers	Required	Decidable	Yes
Misra2012:8.4	A compatible declaration shall be visible when an object or function with external linkage is defined	Required	Decidable	No
Misra2012:8.5	An external object or function shall be declared once in one and only one file	Required	Decidable	Yes
Misra2012:8.6	An identifier with external linkage shall have exactly one external definition	Required	Decidable	Yes
Misra2012:8.7	Functions and objects should not be defined with external linkage if they are referenced in only one translation unit	Advisory	Decidable	Yes

Misra2012:8.8	The static storage class specifier shall be used in all declarations of objects and functions that have internal linkage	Required	Decidable	Yes
Misra2012:8.9	An object should be defined at block scope if its identifier only appears in a single function	Advisory	Decidable	Yes
Misra2012:8.10	An inline function shall be declared with the static storage class	Required	Decidable	Yes
Misra2012:8.11	When an array with external linkage is declared, its size should be explicitly specified	Advisory	Decidable	Yes
Misra2012:8.12	Within an enumerator list, the value of an implicitly-specified enumeration constant shall be unique	Required	Decidable	Yes
Misra2012:8.13	A pointer should point to a const-qualified type whenever possible	Advisory	Undecidable	Yes
Misra2012:8.14	The restrict type qualifier shall not be used	Required	Decidable	Yes
Misra2012:9.1	The value of an object with automatic storage duration shall not be read before it has been set	Mandatory	Undecidable	Yes
Misra2012:9.2	The initializer for an aggregate or union shall be enclosed in braces	Required	Decidable	Yes
Misra2012:9.3	Arrays shall not be partially initialized	Required	Decidable	Yes
Misra2012:9.4	An element of an object shall not be initialized more than once	Required	Decidable	Yes
Misra2012:9.5	Where designated initializers are used to initialize an array object the size of the array shall be specified explicitly	Required	Decidable	Yes
Misra2012:10.1	Operands shall not be of an inappropriate essential type	Required	Decidable	Yes
Misra2012:10.2	Expressions of essentially character type shall not be used inappropriately in addition and subtraction operations	Required	Decidable	Yes
Misra2012:10.3	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	Decidable	Yes
Misra2012:10.4	Both operands of an operator in which the usual arithmetic conversions are performed shall have the same essential type category	Required	Decidable	Yes
Misra2012:10.5	The value of an expression should not be cast to an inappropriate essential type	Advisory	Decidable	Yes
Misra2012:10.6	The value of a composite expression shall not be assigned to an object with wider essential type	Required	Decidable	Yes
Misra2012:10.7	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	Decidable	Yes
Misra2012:10.8	The value of a composite expression shall not be cast to a different essential type category or a wider essential type	Required	Decidable	Yes
Misra2012:11.1	Conversions shall not be performed between a pointer to a function and any other type	Required	Decidable	Yes
Misra2012:11.2	Conversions shall not be performed between a pointer to an incomplete type and any other type	Required	Decidable	Yes
Misra2012:11.3	A cast shall not be performed between a pointer to object type and a pointer to a different object type	Required	Decidable	Yes
Misra2012:11.4	A conversion should not be performed between a pointer to object and an integer type	Advisory	Decidable	Yes
Misra2012:11.5	A conversion should not be performed from pointer to void into pointer to object	Advisory	Decidable	Yes
Misra2012:11.6	A cast shall not be performed between pointer to void and an arithmetic type	Required	Decidable	Yes
Misra2012:11.7	A cast shall not be performed between pointer to object and a non-integer arithmetic type	Required	Decidable	Yes
Misra2012:11.8	A cast shall not remove any const or volatile qualification from the type pointed to by a pointer	Required	Decidable	Yes
Misra2012:11.9	The macro NULL shall be the only permitted form of integer null pointer constant	Required	Decidable	Yes
Misra2012:12.1	The precedence of operators within expressions should be made explicit	Advisory	Decidable	Yes
Misra2012:12.2	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	Undecidable	Yes
Misra2012:12.3	The comma operator should not be used	Advisory	Decidable	Yes
Misra2012:12.4	Evaluation of constant expressions should not lead to unsigned integer wrap-around	Advisory	Decidable	Yes

Misra2012:12.5	The sizeof operator shall not have an operand which is a function parameter declared as "array of type"	Mandatory	Decidable	No
Misra2012:13.1	Initializer lists shall not contain persistent side effects	Required	Undecidable	Yes
Misra2012:13.2	The value of an expression and its persistent side effects shall be the same under all permitted evaluation orders	Required	Undecidable	No
Misra2012:13.3	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	Decidable	Yes
Misra2012:13.4	The result of an assignment operator should not be used	Advisory	Decidable	Yes
Misra2012:13.5	The right hand operand of a logical && or || operator shall not contain persistent side effects	Required	Undecidable	Yes
Misra2012:13.6	The operand of the sizeof operator shall not contain any expression which has potential side effects	Mandatory	Decidable	Yes
Misra2012:14.1	A loop counter shall not have essentially floating type	Required	Undecidable	Yes
Misra2012:14.2	A for loop shall be well-formed	Required	Undecidable	Yes
Misra2012:14.3	Controlling expressions shall not be invariant	Required	Undecidable	Yes
Misra2012:14.4	The controlling expression of an if statement and the controlling expression of an iteration-statement shall have essentially Boolean type	Required	Decidable	Yes
Misra2012:15.1	The goto statement should not be used	Advisory	Decidable	Yes
Misra2012:15.2	The goto statement shall jump to a label declared later in the same function	Required	Decidable	Yes
Misra2012:15.3	Any label referenced by a goto statement shall be declared in the same block, or in any block enclosing the goto statement	Required	Decidable	Yes
Misra2012:15.4	There should be no more than one break or goto statement used to terminate any iteration statement	Advisory	Decidable	Yes
Misra2012:15.5	A function should have a single point of exit at the end	Advisory	Decidable	Yes
Misra2012:15.6	The body of an iteration-statement or a selection-statement shall be a compound-statement	Required	Decidable	Yes
Misra2012:15.7	All if ... else if constructs shall be terminated with an else statement	Required	Decidable	Yes
Misra2012:16.1	All switch statements shall be well-formed	Required	Decidable	Yes
Misra2012:16.2	A switch label shall only be used when the most closely-enclosing compound statement is the body of a switch statement	Required	Decidable	Yes
Misra2012:16.3	An unconditional break statement shall terminate every switch-clause	Required	Decidable	Yes
Misra2012:16.4	Every switch statement shall have a default label	Required	Decidable	Yes
Misra2012:16.5	A default label shall appear as either the first or the last switch label of a switch statement	Required	Decidable	Yes
Misra2012:16.6	Every switch statement shall have at least two switch-clauses	Required	Decidable	Yes
Misra2012:16.7	A switch-expression shall not have essentially Boolean type	Required	Decidable	Yes
Misra2012:17.1	The features of shall not be used	Required	Decidable	Yes
Misra2012:17.2	Functions shall not call themselves, either directly or indirectly	Required	Undecidable	Yes
Misra2012:17.3	A function shall not be declared implicitly	Mandatory	Decidable	Yes
Misra2012:17.4	All exit paths from a function with non-void return type shall have an explicit return statement with an expression	Mandatory	Decidable	Yes
Misra2012:17.5	The function argument corresponding to a parameter declared to have an array type shall have an appropriate number of elements	Advisory	Undecidable	Yes
Misra2012:17.6	The declaration of an array parameter shall not contain the static keyword between the [ ]	Mandatory	Decidable	Yes
Misra2012:17.7	The value returned by a function having non-void return type shall be used	Required	Decidable	Yes
Misra2012:17.8	A function parameter should not be modified	Advisory	Undecidable	Yes
Misra2012:18.1	A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand	Required	Undecidable	Yes
Misra2012:18.2	Subtraction between pointers shall only be applied to pointers that address elements of the same array	Required	Undecidable	Yes

Misra2012:18.3	The relational operators >, >=, < and <= shall not be applied to objects of pointer type except where they point into the same object	Required	Undecidable	Yes
Misra2012:18.4	The +, -, += and -= operators should not be applied to an expression of pointer type	Advisory	Decidable	Yes
Misra2012:18.5	Declarations should contain no more than two levels of pointer nesting	Advisory	Decidable	Yes
Misra2012:18.6	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	Undecidable	Yes
Misra2012:18.7	Flexible array members shall not be declared	Required	Decidable	Yes
Misra2012:18.8	Variable-length array types shall not be used	Required	Decidable	Yes
Misra2012:19.1	An object shall not be assigned or copied to an overlapping object	Mandatory	Undecidable	Yes
Misra2012:19.2	The union keyword should not be used	Advisory	Decidable	Yes
Misra2012:20.1	#include directives should only be preceded by preprocessor directives or comments	Advisory	Decidable	Yes
Misra2012:20.2	The ', " or \ characters and the /* or // character sequences shall not occur in a header file name	Required	Decidable	Yes
Misra2012:20.3	The #include directive shall be followed by either a or "filename" sequence	Required	Decidable	Yes
Misra2012:20.4	A macro shall not be defined with the same name as a keyword	Required	Decidable	Yes
Misra2012:20.5	#undef should not be used	Advisory	Decidable	Yes
Misra2012:20.6	Tokens that look like a preprocessing directive shall not occur within a macro argument	Required	Decidable	Yes
Misra2012:20.7	Expressions resulting from the expansion of macro parameters shall be enclosed in parentheses	Required	Decidable	Yes
Misra2012:20.8	The controlling expression of a #if or #elif preprocessing directive shall evaluate to 0 or 1	Required	Decidable	Yes
Misra2012:20.9	All identifiers used in the controlling expression of #if or #elif preprocessing directives shall be #define'd before evaluation	Required	Decidable	Yes
Misra2012:20.10	The # and ## preprocessor operators should not be used	Advisory	Decidable	Yes
Misra2012:20.11	A macro parameter immediately following a # operator shall not immediately be followed by a ## operator	Required	Decidable	Yes
Misra2012:20.12	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	Decidable	No
Misra2012:20.13	A line whose first token is # shall be a valid preprocessing directive	Required	Decidable	Yes
Misra2012:20.14	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	Decidable	Yes
Misra2012:21.1	#define and #undef shall not be used on a reserved identifier or reserved macro name	Required	Decidable	Yes
Misra2012:21.2	A reserved identifier or macro name shall not be declared	Required	Decidable	Yes
Misra2012:21.3	The memory allocation and deallocation functions of shall not be used	Required	Decidable	Yes
Misra2012:21.4	The standard header file shall not be used	Required	Decidable	Yes
Misra2012:21.5	The standard header file shall not be used	Required	Decidable	Yes
Misra2012:21.6	The Standard Library input/output functions shall not be used	Required	Decidable	Yes
Misra2012:21.7	The atof, atoi, atol and atoll functions of shall not be used	Required	Decidable	Yes
Misra2012:21.8	The Standard Library termination functions of shall not be used	Required	Decidable	Yes
Misra2012:21.9	The library functions bsearch and qsort of shall not be used	Required	Decidable	Yes
Misra2012:21.10	The Standard Library time and date functions shall not be used	Required	Decidable	Yes
Misra2012:21.11	The standard header file shall not be used	Required	Decidable	Yes
Misra2012:21.12	The exception handling features of should not be used	Advisory	Decidable	Yes
Misra2012:21.13	Any value passed to a function in shall be representable as an unsigned char or be the value EOF	Mandatory	Undecidable	Yes
Misra2012:21.14	The Standard Library function memcmp shall not be used to compare null terminated strings	Required	Undecidable	No
Misra2012:21.15	The pointer arguments to the Standard Library functions memcpy, memmove and memcmp shall be pointers to qualified or unqualified versions of compatible types	Required	Decidable	No

Misra2012:21.16	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	Decidable	No
Misra2012:21.17	Use of the string handling functions from shall not result in accesses beyond the bounds of the objects referenced by their pointer parameters	Mandatory	Undecidable	Yes
Misra2012:21.18	The size_t argument passed to any function in shall have an appropriate value	Mandatory	Undecidable	Yes
Misra2012:21.19	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	Undecidable	No
Misra2012:21.20	The pointer returned by the Standard Library functions asctime, ctime, gmtime, localtime, localeconv, setlocale or strerror shall not be used following a subsequent call to the same function	Mandatory	Undecidable	Yes
Misra2012:21.21	The Standard Library function system of shall not be used	Required	Decidable	Yes
Misra2012:22.1	All resources obtained dynamically by means of Standard Library functions shall be explicitly released	Required	Undecidable	Yes
Misra2012:22.2	A block of memory shall only be freed if it was allocated by means of a Standard Library function	Mandatory	Undecidable	Yes
Misra2012:22.3	The same file shall not be open for read and write access at the same time on different streams	Required	Undecidable	No
Misra2012:22.4	There shall be no attempt to write to a stream which has been opened as read-only	Mandatory	Undecidable	Yes
Misra2012:22.5	A pointer to a FILE object shall not be dereferenced	Mandatory	Undecidable	Yes
Misra2012:22.6	The value of a pointer to a FILE shall not be used after the associated stream has been closed	Mandatory	Undecidable	Yes
Misra2012:22.7	The macro EOF shall only be compared with the unmodified return value from any Standard Library function capable of returning EOF	Required	Undecidable	Yes
Misra2012:22.8	The value of errno shall be set to zero prior to a call to an errno-setting-function	Required	Undecidable	Yes
Misra2012:22.9	The value of errno shall be tested against zero after calling an errno-setting-function	Required	Undecidable	Yes
Misra2012:22.10	The value of errno shall only be tested when the last function to be called was an errno-setting-function	Required	Undecidable	Yes
Misra2012:D.1.1	Any implementation-defined behaviour on which the output of the program depends shall be documented and understood	Required	Undecidable	No
Misra2012:D.2.1	All source files shall compile without any compilation errors	Required	Undecidable	No
Misra2012:D.3.1	All code shall be traceable to documented requirements	Required	Undecidable	No
Misra2012:D.4.1	Run-time failures shall be minimized	Required	Undecidable	Yes
Misra2012:D.4.2	All usage of assembly language should be documented	Advisory	Undecidable	No
Misra2012:D.4.3	Assembly language shall be encapsulated and isolated	Required	Undecidable	Yes
Misra2012:D.4.4	Sections of code should not be "commented out"	Advisory	Undecidable	Yes
Misra2012:D.4.5	Identifiers in the same name space with overlapping visibility should be typographically unambiguous	Advisory	Undecidable	Yes
Misra2012:D.4.6	typedefs that indicate size and signedness should be used in place of the basic numerical types	Advisory	Undecidable	Yes
Misra2012:D.4.7	If a function returns error information, then that error information shall be tested	Required	Undecidable	Yes
Misra2012:D.4.8	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	Undecidable	No
Misra2012:D.4.9	A function should be used in preference to a function-like macro where they are interchangeable	Advisory	Undecidable	Yes
Misra2012:D.4.10	Precautions shall be taken in order to prevent the contents of a header file being included more than once	Required	Undecidable	No
Misra2012:D.4.11	The validity of values passed to library functions shall be checked	Required	Undecidable	Yes
Misra2012:D.4.12	Dynamic memory allocation shall not be used	Required	Undecidable	Yes
Misra2012:D.4.13	Functions which are designed to provide operations on a resource should be called in an appropriate sequence	Advisory	Undecidable	Yes
Misra2012:D.4.14	The validity of values received from external sources shall be checked	Required	Undecidable	Yes

MISRA C:2012 BROAD MAPPING (CODESONAR V7.2)

The following table contains CodeSonar classes that are broadly mapped to specific MISRA C:2012 rules and directives.

Rule	Rule Name	Category	Decidability	Supported
Misra2012:1.1	The program shall contain no violations of the standard C syntax and constraints, and shall not exceed the implementation's translation limits	Required	Decidable	No
Misra2012:1.2	Language extensions should not be used	Advisory	Undecidable	Yes
Misra2012:1.3	There shall be no occurrence of undefined or critical unspecified behaviour	Required	Undecidable	Yes
Misra2012:1.4	Emergent language features shall not be used	Required	Decidable	No
Misra2012:2.1	A project shall not contain unreachable code	Required	Undecidable	Yes
Misra2012:2.2	There shall be no dead code	Required	Undecidable	Yes
Misra2012:2.3	A project should not contain unused type declarations	Advisory	Decidable	Yes
Misra2012:2.4	A project should not contain unused tag declarations	Advisory	Decidable	Yes
Misra2012:2.5	A project should not contain unused macro declarations	Advisory	Decidable	Yes
Misra2012:2.6	A function should not contain unused label declarations	Advisory	Decidable	Yes
Misra2012:2.7	There should be no unused parameters in functions	Advisory	Decidable	Yes
Misra2012:3.1	The character sequences /* and // shall not be used within a comment	Required	Decidable	Yes
Misra2012:3.2	Line-splicing shall not be used in // comments	Required	Decidable	Yes
Misra2012:4.1	Octal and hexadecimal escape sequences shall be terminated	Required	Decidable	Yes
Misra2012:4.2	Trigraphs should not be used	Advisory	Decidable	Yes
Misra2012:5.1	External identifiers shall be distinct	Required	Decidable	Yes
Misra2012:5.2	Identifiers declared in the same scope and name space shall be distinct	Required	Decidable	Yes
Misra2012:5.3	An identifier declared in an inner scope shall not hide an identifier declared in an outer scope	Required	Decidable	Yes
Misra2012:5.4	Macro identifiers shall be distinct	Required	Decidable	Yes
Misra2012:5.5	Identifiers shall be distinct from macro names	Required	Decidable	Yes
Misra2012:5.6	A typedef name shall be a unique identifier	Required	Decidable	Yes
Misra2012:5.7	A tag name shall be a unique identifier	Required	Decidable	Yes
Misra2012:5.8	Identifiers that define objects or functions with external linkage shall be unique	Required	Decidable	Yes
Misra2012:5.9	Identifiers that define objects or functions with internal linkage should be unique	Advisory	Decidable	Yes
Misra2012:6.1	Bit-fields shall only be declared with an appropriate type	Required	Decidable	Yes
Misra2012:6.2	Single-bit named bit fields shall not be of a signed type	Required	Decidable	Yes
Misra2012:7.1	Octal constants shall not be used	Required	Decidable	Yes
Misra2012:7.2	A "u" or "U" suffix shall be applied to all integer constants that are represented in an unsigned type	Required	Decidable	Yes
Misra2012:7.3	The lowercase character "l" shall not be used in a literal suffix	Required	Decidable	Yes
Misra2012:7.4	A string literal shall not be assigned to an object unless the object's type is "pointer to const-qualified char"	Required	Decidable	Yes
Misra2012:8.1	Types shall be explicitly specified	Required	Decidable	No
Misra2012:8.2	Function types shall be in prototype form with named parameters	Required	Decidable	Yes
Misra2012:8.3	All declarations of an object or function shall use the same names and type qualifiers	Required	Decidable	Yes
Misra2012:8.4	A compatible declaration shall be visible when an object or function with external linkage is defined	Required	Decidable	No
Misra2012:8.5	An external object or function shall be declared once in one and only one file	Required	Decidable	Yes
Misra2012:8.6	An identifier with external linkage shall have exactly one external definition	Required	Decidable	Yes

Misra2012:8.7	Functions and objects should not be defined with external linkage if they are referenced in only one translation unit	Advisory	Decidable	Yes
Misra2012:8.8	The static storage class specifier shall be used in all declarations of objects and functions that have internal linkage	Required	Decidable	Yes
Misra2012:8.9	An object should be defined at block scope if its identifier only appears in a single function	Advisory	Decidable	Yes
Misra2012:8.10	An inline function shall be declared with the static storage class	Required	Decidable	Yes
Misra2012:8.11	When an array with external linkage is declared, its size should be explicitly specified	Advisory	Decidable	Yes
Misra2012:8.12	Within an enumerator list, the value of an implicitly-specified enumeration constant shall be unique	Required	Decidable	Yes
Misra2012:8.13	A pointer should point to a const-qualified type whenever possible	Advisory	Undecidable	Yes
Misra2012:8.14	The restrict type qualifier shall not be used	Required	Decidable	Yes
Misra2012:9.1	The value of an object with automatic storage duration shall not be read before it has been set	Mandatory	Undecidable	Yes
Misra2012:9.2	The initializer for an aggregate or union shall be enclosed in braces	Required	Decidable	Yes
Misra2012:9.3	Arrays shall not be partially initialized	Required	Decidable	Yes
Misra2012:9.4	An element of an object shall not be initialized more than once	Required	Decidable	Yes
Misra2012:9.5	Where designated initializers are used to initialize an array object the size of the array shall be specified explicitly	Required	Decidable	Yes
Misra2012:10.1	Operands shall not be of an inappropriate essential type	Required	Decidable	Yes
Misra2012:10.2	Expressions of essentially character type shall not be used inappropriately in addition and subtraction operations	Required	Decidable	Yes
Misra2012:10.3	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	Decidable	Yes
Misra2012:10.4	Both operands of an operator in which the usual arithmetic conversions are performed shall have the same essential type category	Required	Decidable	Yes
Misra2012:10.5	The value of an expression should not be cast to an inappropriate essential type	Advisory	Decidable	Yes
Misra2012:10.6	The value of a composite expression shall not be assigned to an object with wider essential type	Required	Decidable	Yes
Misra2012:10.7	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	Decidable	Yes
Misra2012:10.8	The value of a composite expression shall not be cast to a different essential type category or a wider essential type	Required	Decidable	Yes
Misra2012:11.1	Conversions shall not be performed between a pointer to a function and any other type	Required	Decidable	Yes
Misra2012:11.2	Conversions shall not be performed between a pointer to an incomplete type and any other type	Required	Decidable	Yes
Misra2012:11.3	A cast shall not be performed between a pointer to object type and a pointer to a different object type	Required	Decidable	Yes
Misra2012:11.4	A conversion should not be performed between a pointer to object and an integer type	Advisory	Decidable	Yes
Misra2012:11.5	A conversion should not be performed from pointer to void into pointer to object	Advisory	Decidable	Yes
Misra2012:11.6	A cast shall not be performed between pointer to void and an arithmetic type	Required	Decidable	Yes
Misra2012:11.7	A cast shall not be performed between pointer to object and a non-integer arithmetic type	Required	Decidable	Yes
Misra2012:11.8	A cast shall not remove any const or volatile qualification from the type pointed to by a pointer	Required	Decidable	Yes
Misra2012:11.9	The macro NULL shall be the only permitted form of integer null pointer constant	Required	Decidable	Yes
Misra2012:12.1	The precedence of operators within expressions should be made explicit	Advisory	Decidable	Yes
Misra2012:12.2	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	Undecidable	Yes

Misra2012:12.3	The comma operator should not be used	Advisory	Decidable	Yes
Misra2012:12.4	Evaluation of constant expressions should not lead to unsigned integer wrap-around	Advisory	Decidable	Yes
Misra2012:12.5	The sizeof operator shall not have an operand which is a function parameter declared as "array of type"	Mandatory	Decidable	No
Misra2012:13.1	Initializer lists shall not contain persistent side effects	Required	Undecidable	Yes
Misra2012:13.2	The value of an expression and its persistent side effects shall be the same under all permitted evaluation orders	Required	Undecidable	No
Misra2012:13.3	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	Decidable	Yes
Misra2012:13.4	The result of an assignment operator should not be used	Advisory	Decidable	Yes
Misra2012:13.5	The right hand operand of a logical && or || operator shall not contain persistent side effects	Required	Undecidable	Yes
Misra2012:13.6	The operand of the sizeof operator shall not contain any expression which has potential side effects	Mandatory	Decidable	Yes
Misra2012:14.1	A loop counter shall not have essentially floating type	Required	Undecidable	Yes
Misra2012:14.2	A for loop shall be well-formed	Required	Undecidable	Yes
Misra2012:14.3	Controlling expressions shall not be invariant	Required	Undecidable	Yes
Misra2012:14.4	The controlling expression of an if statement and the controlling expression of an iteration-statement shall have essentially Boolean type	Required	Decidable	Yes
Misra2012:15.1	The goto statement should not be used	Advisory	Decidable	Yes
Misra2012:15.2	The goto statement shall jump to a label declared later in the same function	Required	Decidable	Yes
Misra2012:15.3	Any label referenced by a goto statement shall be declared in the same block, or in any block enclosing the goto statement	Required	Decidable	Yes
Misra2012:15.4	There should be no more than one break or goto statement used to terminate any iteration statement	Advisory	Decidable	Yes
Misra2012:15.5	A function should have a single point of exit at the end	Advisory	Decidable	Yes
Misra2012:15.6	The body of an iteration-statement or a selection-statement shall be a compound-statement	Required	Decidable	Yes
Misra2012:15.7	All if ... else if constructs shall be terminated with an else statement	Required	Decidable	Yes
Misra2012:16.1	All switch statements shall be well-formed	Required	Decidable	Yes
Misra2012:16.2	A switch label shall only be used when the most closely-enclosing compound statement is the body of a switch statement	Required	Decidable	Yes
Misra2012:16.3	An unconditional break statement shall terminate every switch-clause	Required	Decidable	Yes
Misra2012:16.4	Every switch statement shall have a default label	Required	Decidable	Yes
Misra2012:16.5	A default label shall appear as either the first or the last switch label of a switch statement	Required	Decidable	Yes
Misra2012:16.6	Every switch statement shall have at least two switch-clauses	Required	Decidable	Yes
Misra2012:16.7	A switch-expression shall not have essentially Boolean type	Required	Decidable	Yes
Misra2012:17.1	The features of shall not be used	Required	Decidable	Yes
Misra2012:17.2	Functions shall not call themselves, either directly or indirectly	Required	Undecidable	Yes
Misra2012:17.3	A function shall not be declared implicitly	Mandatory	Decidable	Yes
Misra2012:17.4	All exit paths from a function with non-void return type shall have an explicit return statement with an expression	Mandatory	Decidable	Yes
Misra2012:17.5	The function argument corresponding to a parameter declared to have an array type shall have an appropriate number of elements	Advisory	Undecidable	Yes
Misra2012:17.6	The declaration of an array parameter shall not contain the static keyword between the [ ]	Mandatory	Decidable	Yes
Misra2012:17.7	The value returned by a function having non-void return type shall be used	Required	Decidable	Yes
Misra2012:17.8	A function parameter should not be modified	Advisory	Undecidable	Yes

Misra2012:18.1	A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand	Required	Undecidable	Yes
Misra2012:18.2	Subtraction between pointers shall only be applied to pointers that address elements of the same array	Required	Undecidable	Yes
Misra2012:18.3	The relational operators >, >=, < and <= shall not be applied to objects of pointer type except where they point into the same object	Required	Undecidable	Yes
Misra2012:18.4	The +, -, += and -= operators should not be applied to an expression of pointer type	Advisory	Decidable	Yes
Misra2012:18.5	Declarations should contain no more than two levels of pointer nesting	Advisory	Decidable	Yes
Misra2012:18.6	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	Undecidable	Yes
Misra2012:18.7	Flexible array members shall not be declared	Required	Decidable	Yes
Misra2012:18.8	Variable-length array types shall not be used	Required	Decidable	Yes
Misra2012:19.1	An object shall not be assigned or copied to an overlapping object	Mandatory	Undecidable	Yes
Misra2012:19.2	The union keyword should not be used	Advisory	Decidable	Yes
Misra2012:20.1	#include directives should only be preceded by preprocessor directives or comments	Advisory	Decidable	Yes
Misra2012:20.2	The ', " or \ characters and the /* or // character sequences shall not occur in a header file name	Required	Decidable	Yes
Misra2012:20.3	The #include directive shall be followed by either a or "filename" sequence	Required	Decidable	Yes
Misra2012:20.4	A macro shall not be defined with the same name as a keyword	Required	Decidable	Yes
Misra2012:20.5	#undef should not be used	Advisory	Decidable	Yes
Misra2012:20.6	Tokens that look like a preprocessing directive shall not occur within a macro argument	Required	Decidable	Yes
Misra2012:20.7	Expressions resulting from the expansion of macro parameters shall be enclosed in parentheses	Required	Decidable	Yes
Misra2012:20.8	The controlling expression of a #if or #elif preprocessing directive shall evaluate to 0 or 1	Required	Decidable	Yes
Misra2012:20.9	All identifiers used in the controlling expression of #if or #elif preprocessing directives shall be #define'd before evaluation	Required	Decidable	Yes
Misra2012:20.10	The # and ## preprocessor operators should not be used	Advisory	Decidable	Yes
Misra2012:20.11	A macro parameter immediately following a # operator shall not immediately be followed by a ## operator	Required	Decidable	Yes
Misra2012:20.12	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	Decidable	No
Misra2012:20.13	A line whose first token is # shall be a valid preprocessing directive	Required	Decidable	Yes
Misra2012:20.14	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	Decidable	Yes
Misra2012:21.1	#define and #undef shall not be used on a reserved identifier or reserved macro name	Required	Decidable	Yes
Misra2012:21.2	A reserved identifier or macro name shall not be declared	Required	Decidable	Yes
Misra2012:21.3	The memory allocation and deallocation functions of shall not be used	Required	Decidable	Yes
Misra2012:21.4	The standard header file shall not be used	Required	Decidable	Yes
Misra2012:21.5	The standard header file shall not be used	Required	Decidable	Yes
Misra2012:21.6	The Standard Library input/output functions shall not be used	Required	Decidable	Yes
Misra2012:21.7	The atof, atoi, atol and atoll functions of shall not be used	Required	Decidable	Yes
Misra2012:21.8	The Standard Library termination functions of shall not be used	Required	Decidable	Yes
Misra2012:21.9	The library functions bsearch and qsort of shall not be used	Required	Decidable	Yes
Misra2012:21.10	The Standard Library time and date functions shall not be used	Required	Decidable	Yes
Misra2012:21.11	The standard header file shall not be used	Required	Decidable	Yes
Misra2012:21.12	The exception handling features of should not be used	Advisory	Decidable	Yes

Misra2012:21.13	Any value passed to a function in shall be representable as an unsigned char or be the value EOF	Mandatory	Undecidable	Yes
Misra2012:21.14	The Standard Library function memcmp shall not be used to compare null terminated strings	Required	Undecidable	No
Misra2012:21.15	The pointer arguments to the Standard Library functions memcpy, memmove and memcmp shall be pointers to qualified or unqualified versions of compatible types	Required	Decidable	No
Misra2012:21.16	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	Decidable	No
Misra2012:21.17	Use of the string handling functions from shall not result in accesses beyond the bounds of the objects referenced by their pointer parameters	Mandatory	Undecidable	Yes
Misra2012:21.18	The size_t argument passed to any function in shall have an appropriate value	Mandatory	Undecidable	Yes
Misra2012:21.19	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	Undecidable	No
Misra2012:21.20	The pointer returned by the Standard Library functions asctime, ctime, gmtime, localtime, localeconv, setlocale or strerror shall not be used following a subsequent call to the same function	Mandatory	Undecidable	Yes
Misra2012:21.21	The Standard Library function system of shall not be used	Required	Decidable	Yes
Misra2012:22.1	All resources obtained dynamically by means of Standard Library functions shall be explicitly released	Required	Undecidable	Yes
Misra2012:22.2	A block of memory shall only be freed if it was allocated by means of a Standard Library function	Mandatory	Undecidable	Yes
Misra2012:22.3	The same file shall not be open for read and write access at the same time on different streams	Required	Undecidable	No
Misra2012:22.4	There shall be no attempt to write to a stream which has been opened as read-only	Mandatory	Undecidable	Yes
Misra2012:22.5	A pointer to a FILE object shall not be dereferenced	Mandatory	Undecidable	Yes
Misra2012:22.6	The value of a pointer to a FILE shall not be used after the associated stream has been closed	Mandatory	Undecidable	Yes
Misra2012:22.7	The macro EOF shall only be compared with the unmodified return value from any Standard Library function capable of returning EOF	Required	Undecidable	Yes
Misra2012:22.8	The value of errno shall be set to zero prior to a call to an errno-setting-function	Required	Undecidable	Yes
Misra2012:22.9	The value of errno shall be tested against zero after calling an errno-setting-function	Required	Undecidable	Yes
Misra2012:22.10	The value of errno shall only be tested when the last function to be called was an errno-setting-function	Required	Undecidable	Yes
Misra2012:D.1.1	Any implementation-defined behaviour on which the output of the program depends shall be documented and understood	Required	Undecidable	No
Misra2012:D.2.1	All source files shall compile without any compilation errors	Required	Undecidable	No
Misra2012:D.3.1	All code shall be traceable to documented requirements	Required	Undecidable	No
Misra2012:D.4.1	Run-time failures shall be minimized	Required	Undecidable	Yes
Misra2012:D.4.2	All usage of assembly language should be documented	Advisory	Undecidable	No
Misra2012:D.4.3	Assembly language shall be encapsulated and isolated	Required	Undecidable	Yes
Misra2012:D.4.4	Sections of code should not be "commented out"	Advisory	Undecidable	Yes
Misra2012:D.4.5	Identifiers in the same name space with overlapping visibility should be typographically unambiguous	Advisory	Undecidable	Yes
Misra2012:D.4.6	typedefs that indicate size and signedness should be used in place of the basic numerical types	Advisory	Undecidable	Yes
Misra2012:D.4.7	If a function returns error information, then that error information shall be tested	Required	Undecidable	Yes
Misra2012:D.4.8	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	Undecidable	No
Misra2012:D.4.9	A function should be used in preference to a function-like macro where they are interchangeable	Advisory	Undecidable	Yes
Misra2012:D.4.10	Precautions shall be taken in order to prevent the contents of a header file being included more than once	Required	Undecidable	No

Misra2012:D.4.11	The validity of values passed to library functions shall be checked	Required	Undecidable	Yes
Misra2012:D.4.12	Dynamic memory allocation shall not be used	Required	Undecidable	Yes
Misra2012:D.4.13	Functions which are designed to provide operations on a resource should be called in an appropriate sequence	Advisory	Undecidable	Yes
Misra2012:D.4.14	The validity of values received from external sources shall be checked	Required	Undecidable	Yes

GrammaTech is a leading global provider of application testing (AST) solutions used by the world's most security conscious organizations to detect, measure, analyze and resolve vulnerabilities for software they develop or use. The company is also a trusted cybersecurity and artificial intelligence research partner for the nation's civil, defense, and intelligence agencies.