GRAMMATECH

MISRA C++:2008 GUIDELINES FOR THE USE OF THE C++ LANGUAGE IN CRITICAL SYSTEMS CODESONAR® 7.4

TRUSTED LEADERS OF SOFTWARE ASSURANCE AND ADVANCED CYBER-SECURITY SOLUTIONS

WWW.GRAMMATECH.COM

MISRA C++:2008 GUIDELINES FOR THE USE OF THE C++ LANGUAGE IN CRITICAL SYSTEMS

The MISRA C++:2008 standard aims to foster safety, reliability, and portability of programs written in ISO C for embedded systems. It is used in a wide range of industries, including automotive, aerospace, medical devices, and industrial control.

CodeSonar 7.4 includes a large number of warning classes that support checking for the MISRA C++:2008 guidelines. Every CodeSonar warning report includes the numbers of any MISRA C++:2008 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++:2008 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++:2008 rules and directives, or use build presets to enable all warning classes that are closely mapped to any MISRA C++:2008 rules and directives. In addition, you can use the CodeSonar search function to find warnings related to specific MISRA C++:2008 rules or directives, or to any MISRA C++:2008 rule or directive.

For more information on MISRA C++:2008:

https://www.misra.org.uk/MISRACHome/tabid/128/Default.aspx

MISRA C++:2008 CLOSE MAPPING (CODESONAR V7.4)

The following table contains CodeSonar classes that are closely mapped to specific MISRA C++:2008 rules and directives.

Rule	Rule Name	Category	Supported
MisraC++2008:0-1-1	A project shall not contain unreachable code.	Required	Yes
MisraC++2008:0-1-2	A project shall not contain infeasible paths.	Required	Yes
MisraC++2008:0-1-3	A project shall not contain unused variables.	Required	Yes
MisraC++2008:0-1-4	A project shall not contain non-volatile POD variables having only one use.	Required	Yes
MisraC++2008:0-1-5	A project shall not contain unused type declarations.	Required	Yes
MisraC++2008:0-1-6	A project shall not contain instances of non-volatile variables being given values that are never subsequently used.	Required	Yes
MisraC++2008:0-1-7	The value returned by a function having a non-void return type that is not an overloaded operator shall always be used.	Required	Yes
MisraC++2008:0-1-8	All functions with void return type shall have external side effect(s).	Required	Yes
MisraC++2008:0-1-9	There shall be no dead code.	Required	Yes
MisraC++2008:0-1-10	Every defined function shall be called at least once.	Required	No
MisraC++2008:0-1-11	There shall be no unused parameters (named or unnamed) in non-virtual functions.	Required	Yes
MisraC++2008:0-1-12	There shall be no unused parameters (named or unnamed) in the set of parameters for a virtual function and all the functions that override it.	Required	Yes
MisraC++2008:0-2-1	An object shall not be assigned to an overlapping object.	Required	Yes
MisraC++2008:0-3-1	Minimization of run-time failures shall be ensured by the use of at least one of:(a) static analysis tools/techniques; (b) dynamic analysis tools/techniques; (c) explicit coding of checks to handle run-time faults.	Document	No
MisraC++2008:0-3-2	If a function generates error information, then that error information shall be tested.	Required	Yes
MisraC++2008:0-4-1	Use of scaled-integer or fixed-point arithmetic shall be documented.	Document	No
MisraC++2008:0-4-2	Use of floating-point arithmetic shall be documented.	Document	No
MisraC++2008:0-4-3	Floating-point implementations shall comply with a defined floating-point standard.	Document	No
MisraC++2008:1-0-1	All code shall conform to ISO/IEC 14882:2003 "The C++ Standard Incorporating Technical Corrigendum 1."	Required	No
MisraC++2008:1-0-2	Multiple compilers shall only be used if they have a common, defined interface.	Document	No
MisraC++2008:1-0-3	The implementation of integer division in the chosen compiler shall be determined and documented.	Document	No
MisraC++2008:2-2-1	The character set and the corresponding encoding shall be documented.	Document	No
MisraC++2008:2-3-1	Trigraphs shall not be used.	Required	Yes
MisraC++2008:2-5-1	Digraphs should not be used.	Advisory	No
MisraC++2008:2-7-1	The character sequence /* shall not be used within a C-style comment.	Required	Yes
MisraC++2008:2-7-2	Sections of code shall not be "commented out" using C-style comments.	Required	Yes
MisraC++2008:2-7-3	Sections of code should not be "commented out" using C++ comments.	Advisory	Yes
MisraC++2008:2-10-1	Different identifiers shall be typographically unambiguous.	Required	Yes
MisraC++2008:2-10-2	Identifiers declared in an inner scope shall not hide an identifier declared in an outer scope.	Required	Yes
MisraC++2008:2-10-3	A typedef name (including qualification, if any) shall be a unique identifier.	Required	Yes
MisraC++2008:2-10-4	A class, union or enum name (including qualification, if any) shall be a unique identifier.	Required	Yes
MisraC++2008:2-10-5	The identifier name of a non-member object or function with static storage duration should not be reused.	Advisory	Yes
MisraC++2008:2-10-6	If an identifier refers to a type, it shall not also refer to an object or a function in the same scope.	Required	Yes

MisraC++2008:2-13-1	Only those escape sequences that are defined in ISO/IEC 14882:2003 shall be used.	Required	No
MisraC++2008:2-13-2	Octal constants (other than zero) and octal escape sequences (other than "\0") shall not be used.	Required	Yes
MisraC++2008:2-13-3	A "U" suffix shall be applied to all octal or hexadecimal integer literals of unsigned type.	Required	Yes
MisraC++2008:2-13-4	Literal suffixes shall be upper case.	Required	Yes
MisraC++2008:2-13-5	Narrow and wide string literals shall not be concatenated.	Required	No
MisraC++2008:3-1-1	It shall be possible to include any header file in multiple translation units without violating the One Definition Rule.	Required	Yes
MisraC++2008:3-1-2	Functions shall not be declared at block scope.	Required	Yes
MisraC++2008:3-1-3	When an array is declared, its size shall either be stated explicitly or defined implicitly by initialization.	Required	Yes
MisraC++2008:3-2-1	All declarations of an object or function shall have compatible types.	Required	Yes
MisraC++2008:3-2-2	The One Definition Rule shall not be violated.	Required	Yes
MisraC++2008:3-2-3	A type, object or function that is used in multiple translation units shall be declared in one and only one file.	Required	Yes
MisraC++2008:3-2-4	An identifier with external linkage shall have exactly one definition.	Required	Yes
MisraC++2008:3-3-1	Objects or functions with external linkage shall be declared in a header file.	Required	No
MisraC++2008:3-3-2	If a function has internal linkage then all re-declarations shall include the static storage class specifier.	Required	No
MisraC++2008:3-4-1	An identifier declared to be an object or type shall be defined in a block that minimizes its visibility.	Required	Yes
MisraC++2008:3-9-1	The types used for an object, a function return type, or a function parameter shall be token-for-token identical in all declarations and re-declarations.	Required	Yes
MisraC++2008:3-9-2	typedefs that indicate size and signedness should be used in place of the basic numerical types.	Advisory	Yes
MisraC++2008:3-9-3	The underlying bit representations of floating-point values shall not be used.	Required	Yes
MisraC++2008:4-5-1	Expressions with type bool shall not be used as operands to built-in operators other than the assignment operator = , the logical operators && , || , ! , the equality operators == and != , the unary & operator, and the conditional operator.	Required	Yes
MisraC++2008:4-5-2	Expressions with type enum shall not be used as operands to built-in operators other than the subscript operator [ ] , the assignment operator = , the equality operators == and != , the unary & operator, and the relational operators < , <= , > , >= .	Required	Yes
MisraC++2008:4-5-3	Expressions with type (plain) char and wchar_t shall not be used as operands to built-in operators other than the assignment operator = , the equality operators == and != , and the unary & operator.	Required	Yes
MisraC++2008:4-10-1	NULL shall not be used as an integer value.	Required	Yes
MisraC++2008:4-10-2	Literal zero (0) shall not be used as the null-pointer-constant.	Required	Yes
MisraC++2008:5-0-1	The value of an expression shall be the same under any order of evaluation that the standard permits.	Required	Yes
MisraC++2008:5-0-2	Limited dependence should be placed on C++ operator precedence rules in expressions.	Advisory	Yes
MisraC++2008:5-0-3	A cvalue expression shall not be implicitly converted to a different underlying type.	Required	Yes
MisraC++2008:5-0-4	An implicit integral conversion shall not change the signedness of the underlying type.	Required	Yes
MisraC++2008:5-0-5	There shall be no implicit floating-integral conversions.	Required	Yes
MisraC++2008:5-0-6	An implicit integral or floating-point conversion shall not reduce the size of the underlying type.	Required	Yes
MisraC++2008:5-0-7	There shall be no explicit floating-integral conversions of a cvalue expression.	Required	Yes
MisraC++2008:5-0-8	An explicit integral or floating-point conversion shall not increase the size of the underlying type of a cvalue expression.	Required	Yes
MisraC++2008:5-0-9	An explicit integral conversion shall not change the signedness of the underlying type of a cvalue expression.	Required	Yes
MisraC++2008:5-0-10	If the bitwise operators ~ and << are applied to an operand with an underlying type of unsigned char or unsigned short, the result shall be immediately cast to the underlying type of the operand.	Required	Yes

MisraC++2008:5-0-11	The plain char type shall only be used for the storage and use of character values.	Required	Yes
MisraC++2008:5-0-12	signed char and unsigned char type shall only be used for the storage and use of numeric values.	Required	Yes
MisraC++2008:5-0-13	The condition of an if-statement and the condition of an iteration-statement shall have type bool.	Required	Yes
MisraC++2008:5-0-14	The first operand of a conditional-operator shall have type bool.	Required	Yes
MisraC++2008:5-0-15	Array indexing shall be the only form of pointer arithmetic.	Required	Yes
MisraC++2008:5-0-16	A pointer operand and any pointer resulting from pointer arithmetic using that operand shall both address elements of the same array.	Required	Yes
MisraC++2008:5-0-17	Subtraction between pointers shall only be applied to pointers that address elements of the same array.	Required	Yes
MisraC++2008:5-0-18	> , >= , < , <= shall not be applied to objects of pointer type, except where they point to the same array.	Required	Yes
MisraC++2008:5-0-19	The declaration of objects shall contain no more than two levels of pointer indirection.	Required	Yes
MisraC++2008:5-0-20	Non-constant operands to a binary bitwise operator shall have the same underlying type.	Required	Yes
MisraC++2008:5-0-21	Bitwise operators shall only be applied to operands of unsigned underlying type.	Required	Yes
MisraC++2008:5-2-1	Each operand of a logical && or || shall be a postfix-expression.	Required	No
MisraC++2008:5-2-2	A pointer to a virtual base class shall only be cast to a pointer to a derived class by means of dynamic_cast.	Required	Yes
MisraC++2008:5-2-3	Casts from a base class to a derived class should not be performed on polymorphic types.	Advisory	No
MisraC++2008:5-2-4	C-style casts (other than void casts) and functional notation casts (other than explicit constructor calls) shall not be used.	Required	Yes
MisraC++2008:5-2-5	A cast shall not remove any const or volatile qualification from the type of a pointer or reference.	Required	Yes
MisraC++2008:5-2-6	A cast shall not convert a pointer to a function to any other pointer type, including a pointer to function type.	Required	Yes
MisraC++2008:5-2-7	An object with pointer type shall not be converted to an unrelated pointer type, either directly or indirectly.	Required	Yes
MisraC++2008:5-2-8	An object with integer type or pointer to void type shall not be converted to an object with pointer type.	Required	Yes
MisraC++2008:5-2-9	A cast should not convert a pointer type to an integral type.	Advisory	Yes
MisraC++2008:5-2-10	The increment (++) and decrement (--) operators should not be mixed with other operators in an expression.	Advisory	Yes
MisraC++2008:5-2-11	The comma operator, && operator and the || operator shall not be overloaded.	Required	Yes
MisraC++2008:5-2-12	An identifier with array type passed as a function argument shall not decay to a pointer.	Required	Yes
MisraC++2008:5-3-1	Each operand of the ! operator, the logical && or the logical || operators shall have type bool.	Required	Yes
MisraC++2008:5-3-2	The unary minus operator shall not be applied to an expression whose underlying type is unsigned.	Required	Yes
MisraC++2008:5-3-3	The unary & operator shall not be overloaded.	Required	Yes
MisraC++2008:5-3-4	Evaluation of the operand to the sizeof operator shall not contain side effects.	Required	Yes
MisraC++2008:5-8-1	The right hand operand of a shift operator shall lie between zero and one less than the width in bits of the underlying type of the left hand operand.	Required	Yes
MisraC++2008:5-14-1	The right hand operand of a logical && or || operator shall not contain side effects.	Required	Yes
MisraC++2008:5-17-1	The semantic equivalence between a binary operator and its assignment operator form shall be preserved.	Required	No
MisraC++2008:5-18-1	The comma operator shall not be used.	Required	Yes
MisraC++2008:5-19-1	Evaluation of constant unsigned integer expressions should not lead to wrap-around.	Advisory	No
MisraC++2008:6-2-1	Assignment operators shall not be used in sub-expressions.	Required	Yes
MisraC++2008:6-2-2	Floating-point expressions shall not be directly or indirectly tested for equality or inequality.	Required	Yes

MisraC++2008:6-2-3	Before preprocessing, a null statement shall only occur on a line ny itself; it may be followed by a comment, provided that the first character following the null statement is a white-space character.	Required	No
MisraC++2008:6-3-1	The statement forming the body of a switch, while, do ... while or for statement shall be a compound statement.	Required	Yes
MisraC++2008:6-4-1	An if (condition) construct shall be followed by a compound statement. The else keyword shall be followed by either a compound statement, or another if statement.	Required	Yes
MisraC++2008:6-4-2	All if ... else if constructs shall be terminated with an else clause.	Required	Yes
MisraC++2008:6-4-3	A switch statement shall be a well-formed switch statement.	Required	Yes
MisraC++2008:6-4-4	A switch-label shall only be used when the most closely-enclosing compound statement is the body of a switch statement.	Required	Yes
MisraC++2008:6-4-5	An unconditional throw or break statement shall terminate every non-empty switch-clause.	Required	Yes
MisraC++2008:6-4-6	The final clause of a switch statement shall be the default-clause.	Required	Yes
MisraC++2008:6-4-7	The condition of a switch statement shall not have bool type.	Required	Yes
MisraC++2008:6-4-8	Every switch statement shall have at least one case-clause.	Required	Yes
MisraC++2008:6-5-1	A for loop shall contain a single loop-counter which shall not have floating type.	Required	Yes
MisraC++2008:6-5-2	If loop-counter is not modified by -- or ++ , then, within condition, the loop-counter shall only be used as an operand to <= , < , > or >= .	Required	No
MisraC++2008:6-5-3	The loop-counter shall not be modified within condition or statement.	Required	Yes
MisraC++2008:6-5-4	The loop-counter shall be modified by one of -- , ++ , -=n , or +=n; where n remains constant for the duration of the loop.	Required	Yes
MisraC++2008:6-5-5	A loop-control-variable other than the loop-counter shall not be modified within condition or expression.	Required	Yes
MisraC++2008:6-5-6	A loop-control-variable other than the loop-counter which is modified in statement shall have type bool.	Required	No
MisraC++2008:6-6-1	Any label referenced by a goto statement shall be declared in the same block, or in a block enclosing the goto statement.	Required	Yes
MisraC++2008:6-6-2	The goto statement shall jump to a label declared later in the same function body.	Required	Yes
MisraC++2008:6-6-3	The continue statement shall only be used within a well-formed for loop.	Required	Yes
MisraC++2008:6-6-4	For any iteration statement there shall be no more than one break or goto statement used for loop termination.	Required	Yes
MisraC++2008:6-6-5	A function shall have a single point of exit at the end of the function.	Required	Yes
MisraC++2008:7-1-1	A variable which is not modified shall be const qualified.	Required	Yes
MisraC++2008:7-1-2	A pointer or reference parameter in a function shall be declared as pointer to const or reference to const if the corresponding object is not modified.	Required	Yes
MisraC++2008:7-2-1	An expression with enum underlying type shall only have values corresponding to the enumerators of the enumeration.	Required	Yes
MisraC++2008:7-3-1	The global namespace shall only contain main, namespace declarations and extern "C" declarations.	Required	Yes
MisraC++2008:7-3-2	The identifier main shall not be used for a function other than the global function main.	Required	No
MisraC++2008:7-3-3	There shall be no unnamed namespaces in header files.	Required	Yes
MisraC++2008:7-3-4	using-directives shall not be used.	Required	Yes
MisraC++2008:7-3-5	Multiple declarations for an identifier in the same namespace shall not straddle a using-declaration for that identifier.	Required	Yes
MisraC++2008:7-3-6	using-directives and using-declarations (excluding class scope or function scope using-declarations) shall not be used in header files.	Required	Yes
MisraC++2008:7-4-1	All usage of assembler shall be documented.	Document	No
MisraC++2008:7-4-2	Assembler instructions shall only be introduced using the asm declaration.	Required	Yes
MisraC++2008:7-4-3	Assembly language shall be encapsulated and isolated.	Required	Yes
MisraC++2008:7-5-1	A function shall not return a reference or a pointer to an automatic variable (including parameters), defined within the function.	Required	Yes

MisraC++2008:7-5-2	The address of an object with automatic storage shall not be assigned to another object that may persist afer the first object has ceased to exist.	Required	Yes
MisraC++2008:7-5-3	A function shall not return a reference or a pointer to a parameter that is passed by reference or const reference.	Required	No
MisraC++2008:7-5-4	Functions should not call themselves, either directly or indirectly.	Advisory	Yes
MisraC++2008:8-0-1	An init-declarator-list or a member-declarator-list shall consist of a single init-declarator or member-declarator respectively.	Required	No
MisraC++2008:8-3-1	Parameters in an overriding virtual function shall either use the same default arguments as the function they override, or else shall not specify any default arguments.	Required	Yes
MisraC++2008:8-4-1	Functions shall not be defined using the ellipsis notation.	Required	Yes
MisraC++2008:8-4-2	The identifiers used for the parameters in a re-declaration of a function shall be identical to those in the declaration.	Required	Yes
MisraC++2008:8-4-3	All exit paths from a function with non-void return type shall have an explicit return statement with an expression.	Required	Yes
MisraC++2008:8-4-4	A function identifier shall either be used to call the function or it shall be preceded by &.	Required	Yes
MisraC++2008:8-5-1	All variables shall have a defined value before they are used.	Required	Yes
MisraC++2008:8-5-2	Braces shall be used to indicate and match the structure in the non-zero initialization of arrays and structures.	Required	Yes
MisraC++2008:8-5-3	In an enumerator list, the = construct shall not be used to explicitly initialize members other than the first, unless all items are explicitly initialized.	Required	Yes
MisraC++2008:9-3-1	const member functions shall not return non-const pointers or references to class-data.	Required	No
MisraC++2008:9-3-2	Member functions shall not return non-const handles to class-data.	Required	No
MisraC++2008:9-3-3	If a member function can be made static then it shall be made static, otherwise if it can be made const then it shall be made const.	Required	Yes
MisraC++2008:9-5-1	Unions shall not be used.	Required	Yes
MisraC++2008:9-6-1	When the absolute partitioning of bits representing a bit-field is required, then the behaviour and packing of bit-fields shall be documented.	Document	No
MisraC++2008:9-6-2	Bit-fields shall be either bool type or an explicitly unsigned or signed integral type.	Required	Yes
MisraC++2008:9-6-3	Bit-fields shall not have enum type.	Required	Yes
MisraC++2008:9-6-4	Named bit-fields with signed integer type shall have a length of more than one bit.	Required	Yes
MisraC++2008:10-1-1	Classes should not be derived from virtual bases.	Advisory	Yes
MisraC++2008:10-1-2	A base class shall only be declared virtual if it is used in a diamond hierarchy.	Required	Yes
MisraC++2008:10-1-3	An accessible base class shall not be both virtual and non-virtual in the same hierarchy.	Required	Yes
MisraC++2008:10-2-1	All accessible entity names within a multiple inheritance hierarchy should be unique.	Advisory	No
MisraC++2008:10-3-1	There shall be no more than one definition of each virtual function on each path through the inheritance hierarchy.	Required	No
MisraC++2008:10-3-2	Each overriding virtual function shall be declared with the virtual keyword.	Required	No
MisraC++2008:10-3-3	A virtual function shall only be overridden by a pure virtual function if it is itself declared as pure virtual.	Required	No
MisraC++2008:11-0-1	Member data in non-POD class types shall be private.	Required	No
MisraC++2008:12-1-1	An object's dynamic type shall not be used from the body of its constructor or destructor.	Required	Yes
MisraC++2008:12-1-2	All constructors of a class should explicitly call a constructor for all of its immediate base classes and all virtual base classes.	Advisory	No
MisraC++2008:12-1-3	All constructors that are callable with a single argument of fundamental type shall be declared explicit.	Required	No
MisraC++2008:12-8-1	A copy constructor shall only initialize its base classes and the non- static members of the class of which it is a member.	Required	Yes
MisraC++2008:12-8-2	The copy assignment operator shall be declared protected or private in an abstract class.	Required	Yes
MisraC++2008:14-5-1	A non-member generic function shall only be declared in a namespace that is not an associated namespace.	Required	No

MisraC++2008:14-5-2	A copy constructor shall be declared when there is a template constructor with a single parameter that is a generic parameter.	Required	No
MisraC++2008:14-5-3	A copy assignment operator shall be declared when there is a template assignment operator with a parameter that is a generic parameter.	Required	No
MisraC++2008:14-6-1	In a class template with a dependent base, any name that may be found in that dependent base shall be referred to using a qualified-id or this->	Required	No
MisraC++2008:14-6-2	The function chosen by overload resolution shall resolve to a function declared previously in the translation unit.	Required	No
MisraC++2008:14-7-1	All class templates, function templates, class template member functions and class template static members shall be instantiated at least once.	Required	No
MisraC++2008:14-7-2	For any given template specialization, an explicit instantiation of the template with the template-arguments used in the specialization shall not render the program ill-formed.	Required	No
MisraC++2008:14-7-3	All partial and explicit specializations for a template shall be declared in the same file as the declaration of their primary template.	Required	No
MisraC++2008:14-8-1	Overloaded function templates shall not be explicitly specialized.	Required	No
MisraC++2008:14-8-2	The viable function set for a function call should either contain no function specializations, or only contain function specializations.	Advisory	No
MisraC++2008:15-0-1	Exceptions shall only be used for error handling.	Document	No
MisraC++2008:15-0-2	An exception object should not have pointer type.	Advisory	No
MisraC++2008:15-0-3	Control shall not be transferred into a try or catch block using a goto or a switch statement.	Required	No
MisraC++2008:15-1-1	The assignment-expression of a throw statement shall not itself cause an exception to be thrown.	Required	No
MisraC++2008:15-1-2	NULL shall not be thrown explicitly.	Required	No
MisraC++2008:15-1-3	An empty throw (throw;) shall only be used in the compound- statement of a catch handler.	Required	No
MisraC++2008:15-3-1	Exceptions shall be raised only after start-up and before termination of the program.	Required	No
MisraC++2008:15-3-2	There should be at least one exception handler to catch all otherwise unhandled exceptions.	Advisory	No
MisraC++2008:15-3-3	Handlers of a function-try-block implementation of a class constructor or destructor shall not reference non-static members from this class or its bases.	Required	No
MisraC++2008:15-3-4	Each exception explicitly thrown in the code shall have a handler of a compatible type in all call paths that could lead to that point.	Required	No
MisraC++2008:15-3-5	A class type exception shall always be caught by reference.	Required	Yes
MisraC++2008:15-3-6	Where multiple handlers are provided in a single try-catch statement or function-try-block for a derived class and some or all of its bases, the handlers shall be ordered most-derived to base class.	Required	Yes
MisraC++2008:15-3-7	Where multiple handlers are provided in a single try-catch statement or function-try-block, any ellipsis (catch-all) handler shall occur last.	Required	Yes
MisraC++2008:15-4-1	If a function is declared with an exception-specification, then all declarations of the same function (in other translation units) shall be declared with the same set of type-ids.	Required	No
MisraC++2008:15-5-1	A class destructor shall not exit with an exception.	Required	No
MisraC++2008:15-5-2	Where a function's declaration includes an exception-specification, the function shall only be capable of throwing exceptions of the indicated type(s).	Required	No
MisraC++2008:15-5-3	The terminate() function shall not be called implicitly.	Required	No
MisraC++2008:16-0-1	#include directives in a file shall only be preceded by preprocessor directives or comments.	Required	Yes
MisraC++2008:16-0-2	Macros shall only be #define'd or #undef'd in the global namespace.	Required	No
MisraC++2008:16-0-3	#undef shall not be used.	Required	Yes
MisraC++2008:16-0-4	Function-like macros shall not be defined.	Required	Yes
MisraC++2008:16-0-5	Arguments to a function-like macro shall not contain tokens that look like preprocessing directives.	Required	Yes
MisraC++2008:16-0-6	In the definition of a function-like macro, each instance of a parameter shall be enclosed in parentheses, unless it is used as the operand of # or ## .	Required	Yes

MisraC++2008:16-0-7	Undefined macro identifiers shall not be used in #if or #elif preprocessor directives, except as operands to the defined operator.	Required	Yes
MisraC++2008:16-0-8	If the # token appears as the first token on a line, then it shall be immediately followed by a preprocessing token.	Required	Yes
MisraC++2008:16-1-1	The defined preprocessor operator shall only be used in one of the two standard forms.	Required	No
MisraC++2008:16-1-2	All #else, #elif and #endif preprocessor directives shall reside in the same file as the #if or #ifdef directive to which they are related.	Required	Yes
MisraC++2008:16-2-1	The pre-processor shall only be used for file inclusion and include guards.	Required	No
MisraC++2008:16-2-2	C++ macros shall only be used for include guards, type qualifiers, or storage class specifiers.	Required	No
MisraC++2008:16-2-3	Include guards shall be provided.	Required	No
MisraC++2008:16-2-4	The ', ", /* or // characters shall not occur in a header file name.	Required	Yes
MisraC++2008:16-2-5	The \ character should not occur in a header file name.	Advisory	Yes
MisraC++2008:16-2-6	The #include directive shall be followed by either a or "filename" sequence.	Required	Yes
MisraC++2008:16-3-1	There shall be at most one occurrence of the # or ## operators in a single macro definition.	Required	No
MisraC++2008:16-3-2	The # and ## operators should not be used.	Advisory	Yes
MisraC++2008:16-6-1	All uses of the #pragma directive shall be documented.	Document	No
MisraC++2008:17-0-1	Reserved identifiers, macros and functions in the standard library shall not be defined, redefined or undefined.	Required	Yes
MisraC++2008:17-0-2	The names of standard library macros and objects shall not be reused.	Required	Yes
MisraC++2008:17-0-3	The names of standard library functions shall not be overridden.	Required	Yes
MisraC++2008:17-0-4	All library code shall conform to MISRA C++.	Document	No
MisraC++2008:17-0-5	The setjmp macro and the longjmp function shall not be used.	Required	Yes
MisraC++2008:18-0-1	The C library shall not be used.	Required	Yes
MisraC++2008:18-0-2	The library functions atof, atoi and atol from library shall not be used.	Required	Yes
MisraC++2008:18-0-3	The library functions abort, exit, getenv and system from library shall not be used.	Required	Yes
MisraC++2008:18-0-4	The time handling functions of library shall not be used.	Required	Yes
MisraC++2008:18-0-5	The unbounded functions of library shall not be used.	Required	Yes
MisraC++2008:18-2-1	The macro offsetof shall not be used.	Required	Yes
MisraC++2008:18-4-1	Dynamic heap memory allocation shall not be used.	Required	Yes
MisraC++2008:18-7-1	The signal handling facilities of shall not be used.	Required	Yes
MisraC++2008:19-3-1	The error indicator errno shall not be used.	Required	No
MisraC++2008:27-0-1	The stream input/output library shall not be used.	Required	Yes

MISRA C++:2008 BROAD MAPPING (CODESONAR V7.4)

The following table contains CodeSonar warning classes that are broadly mapped to MISRA C++:2008 categories.

Rule	Rule Name	Category	Supported
MisraC++2008:0-1-1	A project shall not contain unreachable code.	Required	Yes
MisraC++2008:0-1-2	A project shall not contain infeasible paths.	Required	Yes
MisraC++2008:0-1-3	A project shall not contain unused variables.	Required	Yes
MisraC++2008:0-1-4	A project shall not contain non-volatile POD variables having only one use.	Required	Yes
MisraC++2008:0-1-5	A project shall not contain unused type declarations.	Required	Yes
MisraC++2008:0-1-6	A project shall not contain instances of non-volatile variables being given values that are never subsequently used.	Required	Yes
MisraC++2008:0-1-7	The value returned by a function having a non-void return type that is not an overloaded operator shall always be used.	Required	Yes
MisraC++2008:0-1-8	All functions with void return type shall have external side effect(s).	Required	Yes
MisraC++2008:0-1-9	There shall be no dead code.	Required	Yes
MisraC++2008:0-1-10	Every defined function shall be called at least once.	Required	Yes
MisraC++2008:0-1-11	There shall be no unused parameters (named or unnamed) in non-virtual functions.	Required	Yes
MisraC++2008:0-1-12	There shall be no unused parameters (named or unnamed) in the set of parameters for a virtual function and all the functions that override it.	Required	Yes
MisraC++2008:0-2-1	An object shall not be assigned to an overlapping object.	Required	Yes
MisraC++2008:0-3-1	Minimization of run-time failures shall be ensured by the use of at least one of:(a) static analysis tools/techniques; (b) dynamic analysis tools/techniques; (c) explicit coding of checks to handle run-time faults.	Document	No
MisraC++2008:0-3-2	If a function generates error information, then that error information shall be tested.	Required	Yes
MisraC++2008:0-4-1	Use of scaled-integer or fixed-point arithmetic shall be documented.	Document	No
MisraC++2008:0-4-2	Use of floating-point arithmetic shall be documented.	Document	No
MisraC++2008:0-4-3	Floating-point implementations shall comply with a defined floating-point standard.	Document	No
MisraC++2008:1-0-1	All code shall conform to ISO/IEC 14882:2003 "The C++ Standard Incorporating Technical Corrigendum 1."	Required	No
MisraC++2008:1-0-2	Multiple compilers shall only be used if they have a common, defined interface.	Document	No
MisraC++2008:1-0-3	The implementation of integer division in the chosen compiler shall be determined and documented.	Document	No
MisraC++2008:2-2-1	The character set and the corresponding encoding shall be documented.	Document	No
MisraC++2008:2-3-1	Trigraphs shall not be used.	Required	Yes
MisraC++2008:2-5-1	Digraphs should not be used.	Advisory	No
MisraC++2008:2-7-1	The character sequence /* shall not be used within a C-style comment.	Required	Yes
MisraC++2008:2-7-2	Sections of code shall not be "commented out" using C-style comments.	Required	Yes
MisraC++2008:2-7-3	Sections of code should not be "commented out" using C++ comments.	Advisory	Yes
MisraC++2008:2-10-1	Different identifiers shall be typographically unambiguous.	Required	Yes
MisraC++2008:2-10-2	Identifiers declared in an inner scope shall not hide an identifier declared in an outer scope.	Required	Yes
MisraC++2008:2-10-3	A typedef name (including qualification, if any) shall be a unique identifier.	Required	Yes
MisraC++2008:2-10-4	A class, union or enum name (including qualification, if any) shall be a unique identifier.	Required	Yes
MisraC++2008:2-10-5	The identifier name of a non-member object or function with static storage duration should not be reused.	Advisory	Yes
MisraC++2008:2-10-6	If an identifier refers to a type, it shall not also refer to an object or a function in the same scope.	Required	Yes
MisraC++2008:2-13-1	Only those escape sequences that are defined in ISO/IEC 14882:2003 shall be used.	Required	No
MisraC++2008:2-13-2	Octal constants (other than zero) and octal escape sequences (other than "\0") shall not be used.	Required	Yes
MisraC++2008:2-13-3	A "U" suffix shall be applied to all octal or hexadecimal integer literals of unsigned type.	Required	Yes

MisraC++2008:2-13-4	Literal suffixes shall be upper case.	Required	Yes
MisraC++2008:2-13-5	Narrow and wide string literals shall not be concatenated.	Required	No
MisraC++2008:3-1-1	It shall be possible to include any header file in multiple translation units without violating the One Definition Rule.	Required	Yes
MisraC++2008:3-1-2	Functions shall not be declared at block scope.	Required	Yes
MisraC++2008:3-1-3	When an array is declared, its size shall either be stated explicitly or defined implicitly by initialization.	Required	Yes
MisraC++2008:3-2-1	All declarations of an object or function shall have compatible types.	Required	Yes
MisraC++2008:3-2-2	The One Definition Rule shall not be violated.	Required	Yes
MisraC++2008:3-2-3	A type, object or function that is used in multiple translation units shall be declared in one and only one file.	Required	Yes
MisraC++2008:3-2-4	An identifier with external linkage shall have exactly one definition.	Required	Yes
MisraC++2008:3-3-1	Objects or functions with external linkage shall be declared in a header file.	Required	No
MisraC++2008:3-3-2	If a function has internal linkage then all re-declarations shall include the static storage class specifier.	Required	No
MisraC++2008:3-4-1	An identifier declared to be an object or type shall be defined in a block that minimizes its visibility.	Required	Yes
MisraC++2008:3-9-1	The types used for an object, a function return type, or a function parameter shall be token-for-token identical in all declarations and re-declarations.	Required	Yes
MisraC++2008:3-9-2	typedefs that indicate size and signedness should be used in place of the basic numerical types.	Advisory	Yes
MisraC++2008:3-9-3	The underlying bit representations of floating-point values shall not be used.	Required	Yes
MisraC++2008:4-5-1	Expressions with type bool shall not be used as operands to built-in operators other than the assignment operator = , the logical operators && , || , ! , the equality operators == and != , the unary & operator, and the conditional operator.	Required	Yes
MisraC++2008:4-5-2	Expressions with type enum shall not be used as operands to built-in operators other than the subscript operator [ ] , the assignment operator = , the equality operators == and != , the unary & operator, and the relational operators < , <= , > , >= .	Required	Yes
MisraC++2008:4-5-3	Expressions with type (plain) char and wchar_t shall not be used as operands to built-in operators other than the assignment operator = , the equality operators == and != , and the unary & operator.	Required	Yes
MisraC++2008:4-10-1	NULL shall not be used as an integer value.	Required	Yes
MisraC++2008:4-10-2	Literal zero (0) shall not be used as the null-pointer-constant.	Required	Yes
MisraC++2008:5-0-1	The value of an expression shall be the same under any order of evaluation that the standard permits.	Required	Yes
MisraC++2008:5-0-2	Limited dependence should be placed on C++ operator precedence rules in expressions.	Advisory	Yes
MisraC++2008:5-0-3	A cvalue expression shall not be implicitly converted to a different underlying type.	Required	Yes
MisraC++2008:5-0-4	An implicit integral conversion shall not change the signedness of the underlying type.	Required	Yes
MisraC++2008:5-0-5	There shall be no implicit floating-integral conversions.	Required	Yes
MisraC++2008:5-0-6	An implicit integral or floating-point conversion shall not reduce the size of the underlying type.	Required	Yes
MisraC++2008:5-0-7	There shall be no explicit floating-integral conversions of a cvalue expression.	Required	Yes
MisraC++2008:5-0-8	An explicit integral or floating-point conversion shall not increase the size of the underlying type of a cvalue expression.	Required	Yes
MisraC++2008:5-0-9	An explicit integral conversion shall not change the signedness of the underlying type of a cvalue expression.	Required	Yes
MisraC++2008:5-0-10	If the bitwise operators ~ and << are applied to an operand with an underlying type of unsigned char or unsigned short, the result shall be immediately cast to the underlying type of the operand.	Required	Yes
MisraC++2008:5-0-11	The plain char type shall only be used for the storage and use of character values.	Required	Yes
MisraC++2008:5-0-12	signed char and unsigned char type shall only be used for the storage and use of numeric values.	Required	Yes
MisraC++2008:5-0-13	The condition of an if-statement and the condition of an iteration-statement shall have type bool.	Required	Yes
MisraC++2008:5-0-14	The first operand of a conditional-operator shall have type bool.	Required	Yes
MisraC++2008:5-0-15	Array indexing shall be the only form of pointer arithmetic.	Required	Yes

MisraC++2008:5-0-16	A pointer operand and any pointer resulting from pointer arithmetic using that operand shall both address elements of the same array.	Required	Yes
MisraC++2008:5-0-17	Subtraction between pointers shall only be applied to pointers that address elements of the same array.	Required	Yes
MisraC++2008:5-0-18	> , >= , < , <= shall not be applied to objects of pointer type, except where they point to the same array.	Required	Yes
MisraC++2008:5-0-19	The declaration of objects shall contain no more than two levels of pointer indirection.	Required	Yes
MisraC++2008:5-0-20	Non-constant operands to a binary bitwise operator shall have the same underlying type.	Required	Yes
MisraC++2008:5-0-21	Bitwise operators shall only be applied to operands of unsigned underlying type.	Required	Yes
MisraC++2008:5-2-1	Each operand of a logical && or || shall be a postfix-expression.	Required	No
MisraC++2008:5-2-2	A pointer to a virtual base class shall only be cast to a pointer to a derived class by means of dynamic_cast.	Required	Yes
MisraC++2008:5-2-3	Casts from a base class to a derived class should not be performed on polymorphic types.	Advisory	No
MisraC++2008:5-2-4	C-style casts (other than void casts) and functional notation casts (other than explicit constructor calls) shall not be used.	Required	Yes
MisraC++2008:5-2-5	A cast shall not remove any const or volatile qualification from the type of a pointer or reference.	Required	Yes
MisraC++2008:5-2-6	A cast shall not convert a pointer to a function to any other pointer type, including a pointer to function type.	Required	Yes
MisraC++2008:5-2-7	An object with pointer type shall not be converted to an unrelated pointer type, either directly or indirectly.	Required	Yes
MisraC++2008:5-2-8	An object with integer type or pointer to void type shall not be converted to an object with pointer type.	Required	Yes
MisraC++2008:5-2-9	A cast should not convert a pointer type to an integral type.	Advisory	Yes
MisraC++2008:5-2-10	The increment (++) and decrement (--) operators should not be mixed with other operators in an expression.	Advisory	Yes
MisraC++2008:5-2-11	The comma operator, && operator and the || operator shall not be overloaded.	Required	Yes
MisraC++2008:5-2-12	An identifier with array type passed as a function argument shall not decay to a pointer.	Required	Yes
MisraC++2008:5-3-1	Each operand of the ! operator, the logical && or the logical || operators shall have type bool.	Required	Yes
MisraC++2008:5-3-2	The unary minus operator shall not be applied to an expression whose underlying type is unsigned.	Required	Yes
MisraC++2008:5-3-3	The unary & operator shall not be overloaded.	Required	Yes
MisraC++2008:5-3-4	Evaluation of the operand to the sizeof operator shall not contain side effects.	Required	Yes
MisraC++2008:5-8-1	The right hand operand of a shift operator shall lie between zero and one less than the width in bits of the underlying type of the left hand operand.	Required	Yes
MisraC++2008:5-14-1	The right hand operand of a logical && or || operator shall not contain side effects.	Required	Yes
MisraC++2008:5-17-1	The semantic equivalence between a binary operator and its assignment operator form shall be preserved.	Required	No
MisraC++2008:5-18-1	The comma operator shall not be used.	Required	Yes
MisraC++2008:5-19-1	Evaluation of constant unsigned integer expressions should not lead to wrap-around.	Advisory	No
MisraC++2008:6-2-1	Assignment operators shall not be used in sub-expressions.	Required	Yes
MisraC++2008:6-2-2	Floating-point expressions shall not be directly or indirectly tested for equality or inequality.	Required	Yes
MisraC++2008:6-2-3	Before preprocessing, a null statement shall only occur on a line ny itself; it may be followed by a comment, provided that the first character following the null statement is a white-space character.	Required	No
MisraC++2008:6-3-1	The statement forming the body of a switch, while, do ... while or for statement shall be a compound statement.	Required	Yes
MisraC++2008:6-4-1	An if (condition) construct shall be followed by a compound statement. The else keyword shall be followed by either a compound statement, or another if statement.	Required	Yes
MisraC++2008:6-4-2	All if ... else if constructs shall be terminated with an else clause.	Required	Yes
MisraC++2008:6-4-3	A switch statement shall be a well-formed switch statement.	Required	Yes
MisraC++2008:6-4-4	A switch-label shall only be used when the most closely-enclosing compound statement is the body of a switch statement.	Required	Yes

MisraC++2008:6-4-5	An unconditional throw or break statement shall terminate every non-empty switch-clause.	Required	Yes
MisraC++2008:6-4-6	The final clause of a switch statement shall be the default-clause.	Required	Yes
MisraC++2008:6-4-7	The condition of a switch statement shall not have bool type.	Required	Yes
MisraC++2008:6-4-8	Every switch statement shall have at least one case-clause.	Required	Yes
MisraC++2008:6-5-1	A for loop shall contain a single loop-counter which shall not have floating type.	Required	Yes
MisraC++2008:6-5-2	If loop-counter is not modified by -- or ++ , then, within condition, the loop-counter shall only be used as an operand to <= , < , > or >= .	Required	No
MisraC++2008:6-5-3	The loop-counter shall not be modified within condition or statement.	Required	Yes
MisraC++2008:6-5-4	The loop-counter shall be modified by one of -- , ++ , -=n , or +=n; where n remains constant for the duration of the loop.	Required	Yes
MisraC++2008:6-5-5	A loop-control-variable other than the loop-counter shall not be modified within condition or expression.	Required	Yes
MisraC++2008:6-5-6	A loop-control-variable other than the loop-counter which is modified in statement shall have type bool.	Required	No
MisraC++2008:6-6-1	Any label referenced by a goto statement shall be declared in the same block, or in a block enclosing the goto statement.	Required	Yes
MisraC++2008:6-6-2	The goto statement shall jump to a label declared later in the same function body.	Required	Yes
MisraC++2008:6-6-3	The continue statement shall only be used within a well-formed for loop.	Required	Yes
MisraC++2008:6-6-4	For any iteration statement there shall be no more than one break or goto statement used for loop termination.	Required	Yes
MisraC++2008:6-6-5	A function shall have a single point of exit at the end of the function.	Required	Yes
MisraC++2008:7-1-1	A variable which is not modified shall be const qualified.	Required	Yes
MisraC++2008:7-1-2	A pointer or reference parameter in a function shall be declared as pointer to const or reference to const if the corresponding object is not modified.	Required	Yes
MisraC++2008:7-2-1	An expression with enum underlying type shall only have values corresponding to the enumerators of the enumeration.	Required	Yes
MisraC++2008:7-3-1	The global namespace shall only contain main, namespace declarations and extern "C" declarations.	Required	Yes
MisraC++2008:7-3-2	The identifier main shall not be used for a function other than the global function main.	Required	No
MisraC++2008:7-3-3	There shall be no unnamed namespaces in header files.	Required	Yes
MisraC++2008:7-3-4	using-directives shall not be used.	Required	Yes
MisraC++2008:7-3-5	Multiple declarations for an identifier in the same namespace shall not straddle a using-declaration for that identifier.	Required	Yes
MisraC++2008:7-3-6	using-directives and using-declarations (excluding class scope or function scope using-declarations) shall not be used in header files.	Required	Yes
MisraC++2008:7-4-1	All usage of assembler shall be documented.	Document	No
MisraC++2008:7-4-2	Assembler instructions shall only be introduced using the asm declaration.	Required	Yes
MisraC++2008:7-4-3	Assembly language shall be encapsulated and isolated.	Required	Yes
MisraC++2008:7-5-1	A function shall not return a reference or a pointer to an automatic variable (including parameters), defined within the function.	Required	Yes
MisraC++2008:7-5-2	The address of an object with automatic storage shall not be assigned to another object that may persist afer the first object has ceased to exist.	Required	Yes
MisraC++2008:7-5-3	A function shall not return a reference or a pointer to a parameter that is passed by reference or const reference.	Required	No
MisraC++2008:7-5-4	Functions should not call themselves, either directly or indirectly.	Advisory	Yes
MisraC++2008:8-0-1	An init-declarator-list or a member-declarator-list shall consist of a single init-declarator or member-declarator respectively.	Required	No
MisraC++2008:8-3-1	Parameters in an overriding virtual function shall either use the same default arguments as the function they override, or else shall not specify any default arguments.	Required	Yes
MisraC++2008:8-4-1	Functions shall not be defined using the ellipsis notation.	Required	Yes

MisraC++2008:8-4-2	The identifiers used for the parameters in a re-declaration of a function shall be identical to those in the declaration.	Required	Yes
MisraC++2008:8-4-3	All exit paths from a function with non-void return type shall have an explicit return statement with an expression.	Required	Yes
MisraC++2008:8-4-4	A function identifier shall either be used to call the function or it shall be preceded by &.	Required	Yes
MisraC++2008:8-5-1	All variables shall have a defined value before they are used.	Required	Yes
MisraC++2008:8-5-2	Braces shall be used to indicate and match the structure in the non-zero initialization of arrays and structures.	Required	Yes
MisraC++2008:8-5-3	In an enumerator list, the = construct shall not be used to explicitly initialize members other than the first, unless all items are explicitly initialized.	Required	Yes
MisraC++2008:9-3-1	const member functions shall not return non-const pointers or references to class-data.	Required	No
MisraC++2008:9-3-2	Member functions shall not return non-const handles to class-data.	Required	No
MisraC++2008:9-3-3	If a member function can be made static then it shall be made static, otherwise if it can be made const then it shall be made const.	Required	Yes
MisraC++2008:9-5-1	Unions shall not be used.	Required	Yes
MisraC++2008:9-6-1	When the absolute partitioning of bits representing a bit-field is required, then the behaviour and packing of bit-fields shall be documented.	Document	No
MisraC++2008:9-6-2	Bit-fields shall be either bool type or an explicitly unsigned or signed integral type.	Required	Yes
MisraC++2008:9-6-3	Bit-fields shall not have enum type.	Required	Yes
MisraC++2008:9-6-4	Named bit-fields with signed integer type shall have a length of more than one bit.	Required	Yes
MisraC++2008:10-1-1	Classes should not be derived from virtual bases.	Advisory	Yes
MisraC++2008:10-1-2	A base class shall only be declared virtual if it is used in a diamond hierarchy.	Required	Yes
MisraC++2008:10-1-3	An accessible base class shall not be both virtual and non-virtual in the same hierarchy.	Required	Yes
MisraC++2008:10-2-1	All accessible entity names within a multiple inheritance hierarchy should be unique.	Advisory	No
MisraC++2008:10-3-1	There shall be no more than one definition of each virtual function on each path through the inheritance hierarchy.	Required	No
MisraC++2008:10-3-2	Each overriding virtual function shall be declared with the virtual keyword.	Required	No
MisraC++2008:10-3-3	A virtual function shall only be overridden by a pure virtual function if it is itself declared as pure virtual.	Required	No
MisraC++2008:11-0-1	Member data in non-POD class types shall be private.	Required	No
MisraC++2008:12-1-1	An object's dynamic type shall not be used from the body of its constructor or destructor.	Required	Yes
MisraC++2008:12-1-2	All constructors of a class should explicitly call a constructor for all of its immediate base classes and all virtual base classes.	Advisory	No
MisraC++2008:12-1-3	All constructors that are callable with a single argument of fundamental type shall be declared explicit.	Required	No
MisraC++2008:12-8-1	A copy constructor shall only initialize its base classes and the non- static members of the class of which it is a member.	Required	Yes
MisraC++2008:12-8-2	The copy assignment operator shall be declared protected or private in an abstract class.	Required	Yes
MisraC++2008:14-5-1	A non-member generic function shall only be declared in a namespace that is not an associated namespace.	Required	No
MisraC++2008:14-5-2	A copy constructor shall be declared when there is a template constructor with a single parameter that is a generic parameter.	Required	No
MisraC++2008:14-5-3	A copy assignment operator shall be declared when there is a template assignment operator with a parameter that is a generic parameter.	Required	No
MisraC++2008:14-6-1	In a class template with a dependent base, any name that may be found in that dependent base shall be referred to using a qualified-id or this->	Required	No
MisraC++2008:14-6-2	The function chosen by overload resolution shall resolve to a function declared previously in the translation unit.	Required	No
MisraC++2008:14-7-1	All class templates, function templates, class template member functions and class template static members shall be instantiated at least once.	Required	No

MisraC++2008:14-7-2	For any given template specialization, an explicit instantiation of the template with the template-arguments used in the specialization shall not render the program ill-formed.	Required	No
MisraC++2008:14-7-3	All partial and explicit specializations for a template shall be declared in the same file as the declaration of their primary template.	Required	No
MisraC++2008:14-8-1	Overloaded function templates shall not be explicitly specialized.	Required	No
MisraC++2008:14-8-2	The viable function set for a function call should either contain no function specializations, or only contain function specializations.	Advisory	No
MisraC++2008:15-0-1	Exceptions shall only be used for error handling.	Document	No
MisraC++2008:15-0-2	An exception object should not have pointer type.	Advisory	No
MisraC++2008:15-0-3	Control shall not be transferred into a try or catch block using a goto or a switch statement.	Required	No
MisraC++2008:15-1-1	The assignment-expression of a throw statement shall not itself cause an exception to be thrown.	Required	No
MisraC++2008:15-1-2	NULL shall not be thrown explicitly.	Required	No
MisraC++2008:15-1-3	An empty throw (throw;) shall only be used in the compound- statement of a catch handler.	Required	No
MisraC++2008:15-3-1	Exceptions shall be raised only after start-up and before termination of the program.	Required	No
MisraC++2008:15-3-2	There should be at least one exception handler to catch all otherwise unhandled exceptions.	Advisory	No
MisraC++2008:15-3-3	Handlers of a function-try-block implementation of a class constructor or destructor shall not reference non-static members from this class or its bases.	Required	No
MisraC++2008:15-3-4	Each exception explicitly thrown in the code shall have a handler of a compatible type in all call paths that could lead to that point.	Required	No
MisraC++2008:15-3-5	A class type exception shall always be caught by reference.	Required	Yes
MisraC++2008:15-3-6	Where multiple handlers are provided in a single try-catch statement or function-try-block for a derived class and some or all of its bases, the handlers shall be ordered most-derived to base class.	Required	Yes
MisraC++2008:15-3-7	Where multiple handlers are provided in a single try-catch statement or function-try-block, any ellipsis (catch-all) handler shall occur last.	Required	Yes
MisraC++2008:15-4-1	If a function is declared with an exception-specification, then all declarations of the same function (in other translation units) shall be declared with the same set of type-ids.	Required	No
MisraC++2008:15-5-1	A class destructor shall not exit with an exception.	Required	No
MisraC++2008:15-5-2	Where a function's declaration includes an exception-specification, the function shall only be capable of throwing exceptions of the indicated type(s).	Required	No
MisraC++2008:15-5-3	The terminate() function shall not be called implicitly.	Required	No
MisraC++2008:16-0-1	#include directives in a file shall only be preceded by preprocessor directives or comments.	Required	Yes
MisraC++2008:16-0-2	Macros shall only be #define'd or #undef'd in the global namespace.	Required	No
MisraC++2008:16-0-3	#undef shall not be used.	Required	Yes
MisraC++2008:16-0-4	Function-like macros shall not be defined.	Required	Yes
MisraC++2008:16-0-5	Arguments to a function-like macro shall not contain tokens that look like preprocessing directives.	Required	Yes
MisraC++2008:16-0-6	In the definition of a function-like macro, each instance of a parameter shall be enclosed in parentheses, unless it is used as the operand of # or ## .	Required	Yes
MisraC++2008:16-0-7	Undefined macro identifiers shall not be used in #if or #elif preprocessor directives, except as operands to the defined operator.	Required	Yes
MisraC++2008:16-0-8	If the # token appears as the first token on a line, then it shall be immediately followed by a preprocessing token.	Required	Yes
MisraC++2008:16-1-1	The defined preprocessor operator shall only be used in one of the two standard forms.	Required	No
MisraC++2008:16-1-2	All #else, #elif and #endif preprocessor directives shall reside in the same file as the #if or #ifdef directive to which they are related.	Required	Yes
MisraC++2008:16-2-1	The pre-processor shall only be used for file inclusion and include guards.	Required	No
MisraC++2008:16-2-2	C++ macros shall only be used for include guards, type qualifiers, or storage class specifiers.	Required	No
MisraC++2008:16-2-3	Include guards shall be provided.	Required	No
MisraC++2008:16-2-4	The ', ", /* or // characters shall not occur in a header file name.	Required	Yes
MisraC++2008:16-2-5	The \ character should not occur in a header file name.	Advisory	Yes
MisraC++2008:16-2-6	The #include directive shall be followed by either a or "filename" sequence.	Required	Yes

MisraC++2008:16-3-1	There shall be at most one occurrence of the # or ## operators in a single macro definition.	Required	No
MisraC++2008:16-3-2	The # and ## operators should not be used.	Advisory	Yes
MisraC++2008:16-6-1	All uses of the #pragma directive shall be documented.	Document	No
MisraC++2008:17-0-1	Reserved identifiers, macros and functions in the standard library shall not be defined, redefined or undefined.	Required	Yes
MisraC++2008:17-0-2	The names of standard library macros and objects shall not be reused.	Required	Yes
MisraC++2008:17-0-3	The names of standard library functions shall not be overridden.	Required	Yes
MisraC++2008:17-0-4	All library code shall conform to MISRA C++.	Document	No
MisraC++2008:17-0-5	The setjmp macro and the longjmp function shall not be used.	Required	Yes
MisraC++2008:18-0-1	The C library shall not be used.	Required	Yes
MisraC++2008:18-0-2	The library functions atof, atoi and atol from library shall not be used.	Required	Yes
MisraC++2008:18-0-3	The library functions abort, exit, getenv and system from library shall not be used.	Required	Yes
MisraC++2008:18-0-4	The time handling functions of library shall not be used.	Required	Yes
MisraC++2008:18-0-5	The unbounded functions of library shall not be used.	Required	Yes
MisraC++2008:18-2-1	The macro offsetof shall not be used.	Required	Yes
MisraC++2008:18-4-1	Dynamic heap memory allocation shall not be used.	Required	Yes
MisraC++2008:18-7-1	The signal handling facilities of shall not be used.	Required	Yes
MisraC++2008:19-3-1	The error indicator errno shall not be used.	Required	No
MisraC++2008:27-0-1	The stream input/output library shall not be used.	Required	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.