MISRA C++:2023

Perforce QAC for C++ 2026.1

(MISRA C++:2023 First Edition)

Rule Enforcement Summary

		Total
a	Total Number of Rules	179
b	Total Number of 'Not Statically Enforceable' Rules (Assisted/Unassisted)	4
c	Total Number of Enforceable Rules (a-b)	175
d	Total Number of Enforced Rules	175
e	Total Number of Unenforced Rules (c-d)	0
f	Enforced Rules Percentage (d/c)	100%
g	Unenforced Rules Percentage (e/c)	0%

Id	Description	Category	Analysis	Enforced
Rule-0.0.1	A function shall not contain unreachable statements	Required	Decidable	Yes
Rule-0.0.2	Controlling expressions should not be invariant	Advisory	Undecidable	Yes
Rule-0.1.1	A value should not be unnecessarily written to a local object	Advisory	Undecidable	Yes
Rule-0.1.2	The value returned by a function shall be used	Required	Decidable	Yes
Rule-0.2.1	Variables with limited visibility should be used at least once	Advisory	Decidable	Yes
Rule-0.2.2	A named function parameter shall be used at least once	Required	Decidable	Yes
Rule-0.2.3	Types with limited visibility should be used at least once	Advisory	Decidable	Yes

Id	Description	Category	Analysis	Enforced
Rule-0.2.4	Functions with limited visibility should be used at least once	Advisory	Decidable	Yes
Rule-4.1.1	A program shall conform to ISO/IEC 14882:2017 (C++17)	Required	Undecidable	Yes
Rule-4.1.2	Deprecated features should not be used	Advisory	Decidable	Yes
Rule-4.1.3	There shall be no occurrence of //undefined// or //critical unspecified behaviour//	Required	Undecidable	Yes
Rule-4.6.1	Operations on a memory location shall be sequenced appropriately	Required	Undecidable	Yes
Rule-5.0.1	Trigraph-like sequences should not be used	Advisory	Decidable	Yes
Rule-5.7.1	The character sequence /* shall not be used within a C-style comment	Required	Decidable	Yes
Rule-5.7.3	Line-splicing shall not be used in // comments	Required	Decidable	Yes
Rule-5.10.1	User-defined identifiers shall have an appropriate form	Required	Decidable	Yes
Rule-5.13.1	Within character literals and non raw-string literals, \\ shall only be used to form a defined escape sequence or universal character name	Required	Decidable	Yes
Rule-5.13.2	Octal escape sequences, hexadecimal escape sequences and universal character names shall be terminated	Required	Decidable	Yes
Rule-5.13.3	Octal constants shall not be used	Required	Decidable	Yes
Rule-5.13.4	Unsigned integer literals shall be appropriately suffixed	Required	Decidable	Yes
Rule-5.13.5	The lowercase form of "L" shall not be used as the first character in a literal suffix	Required	Decidable	Yes
Rule-5.13.6	An integer-literal of type long long shall not use a single L or l in any suffix	Required	Decidable	Yes
Rule-5.13.7	String literals with different encoding prefixes shall not be concatenated	Required	Decidable	Yes
Rule-6.0.1	Block scope declarations shall not be visually ambiguous	Required	Decidable	Yes

Id	Description	Category	Analysis	Enforced
Rule-6.0.2	When an array with external linkage is declared, its size should be explicitly specified	Advisory	Decidable	Yes
Rule-6.0.3	The only declarations in the global namespace should be main, namespace declarations and extern "C" declarations	Advisory	Decidable	Yes
Rule-6.0.4	The identifier main shall not be used for a function other than the global function main	Required	Decidable	Yes
Rule-6.2.1	The one-definition rule shall not be violated	Required	Decidable	Yes
Rule-6.2.2	All declarations of a variable or function shall have the same type	Required	Decidable	Yes
Rule-6.2.3	The source code used to implement an entity shall appear only once	Required	Decidable	Yes
Rule-6.2.4	A header file shall not contain definitions of functions or objects that are non-inline and have external linkage	Required	Decidable	Yes
Rule-6.4.1	A variable declared in an inner scope shall not hide a variable declared in an outer scope	Required	Decidable	Yes
Rule-6.4.2	Derived classes shall not conceal functions that are inherited from their bases	Required	Decidable	Yes
Rule-6.4.3	A name that is present in a dependent base shall not be resolved by unqualified lookup	Required	Decidable	Yes
Rule-6.5.1	A function or object with external linkage should be introduced in a header file	Advisory	Decidable	Yes
Rule-6.5.2	Internal linkage should be specified appropriately	Advisory	Decidable	Yes
Rule-6.7.1	Local variables shall not have static storage duration	Required	Decidable	Yes
Rule-6.7.2	Global variables shall not be used	Required	Decidable	Yes
Rule-6.8.1	An object shall not be accessed outside of its lifetime	Required	Undecidable	Yes
Rule-6.8.2	A function must not return a reference or a pointer to a local variable with automatic storage duration	Mandatory	Decidable	Yes

Id	Description	Category	Analysis	Enforced
Rule-6.8.3	An assignment operator shall not assign the address of an object with automatic storage duration to an object with a greater lifetime	Required	Decidable	Yes
Rule-6.8.4	Member functions returning references to their object should be ref-qualified appropriately	Advisory	Decidable	Yes
Rule-6.9.1	The same type aliases shall be used in all declarations of the same entity	Required	Decidable	Yes
Rule-6.9.2	The names of the standard signed integer types and standard unsigned integer types should not be used	Advisory	Decidable	Yes
Rule-7.0.1	There shall be no conversion from type bool	Required	Decidable	Yes
Rule-7.0.2	There shall be no conversion to type bool	Required	Decidable	Yes
Rule-7.0.3	The numerical value of a character shall not be used	Required	Decidable	Yes
Rule-7.0.4	The operands of bitwise operators and shift operators shall be appropriate	Required	Decidable	Yes
Rule-7.0.5	Integral promotion and the usual arithmetic conversions shall not change the signedness or the type category of an operand	Required	Decidable	Yes
Rule-7.0.6	Assignment between numeric types shall be appropriate	Required	Decidable	Yes
Rule-7.11.1	nullptr shall be the only form of the null-pointer-constant	Required	Decidable	Yes
Rule-7.11.2	An array passed as a function argument shall not decay to a pointer	Required	Decidable	Yes
Rule-7.11.3	A conversion from function type to pointer-to-function type shall only occur in appropriate contexts	Required	Decidable	Yes
Rule-8.0.1	Parentheses should be used to make the meaning of an expression appropriately explicit	Advisory	Decidable	Yes
Rule-8.1.1	A non-transient lambda shall not implicitly capture this	Required	Decidable	Yes

Id	Description	Category	Analysis	Enforced
Rule-8.1.2	Variables should be captured explicitly in a non-transient lambda	Advisory	Decidable	Yes
Rule-8.2.1	A virtual base class shall only be cast to a derived class by means of dynamic_cast	Required	Decidable	Yes
Rule-8.2.2	C-style casts and functional notation casts shall not be used	Required	Decidable	Yes
Rule-8.2.3	A cast shall not remove any const or volatile qualification from the type accessed via a pointer or by reference	Required	Decidable	Yes
Rule-8.2.4	Casts shall not be performed between a pointer to a function and any other type	Required	Decidable	Yes
Rule-8.2.5	reinterpret_cast shall not be used	Required	Decidable	Yes
Rule-8.2.6	An object with integral, enumerated, or pointer to void type shall not be cast to a pointer type	Required	Decidable	Yes
Rule-8.2.7	A cast should not convert a pointer type to an integral type	Advisory	Decidable	Yes
Rule-8.2.8	An object pointer type shall not be cast to an integral type other than std::uintptr_t or std::intptr_t	Required	Decidable	Yes
Rule-8.2.9	The operand to typeid shall not be an expression of polymorphic class type	Required	Decidable	Yes
Rule-8.2.10	Functions shall not call themselves, either directly or indirectly	Required	Undecidable	Yes
Rule-8.2.11	An argument passed via ellipsis shall have an appropriate type	Required	Decidable	Yes
Rule-8.3.1	The built-in unary - operator should not be applied to an expression of unsigned type	Advisory	Decidable	Yes
Rule-8.3.2	The built-in unary + operator should not be used	Advisory	Decidable	Yes
Rule-8.7.1	Pointer arithmetic shall not form an invalid pointer	Required	Undecidable	Yes
Rule-8.7.2	Subtraction between pointers shall only be applied to pointers that address elements of the same array	Required	Undecidable	Yes

Id	Description	Category	Analysis	Enforced
Rule-8.9.1	The built-in relational operators >, >=, < and <= shall not be applied to objects of pointer type, except where they point to elements of the same array	Required	Undecidable	Yes
Rule-8.14.1	The right-hand operand of a logical && or || operator should not contain persistent side effects	Advisory	Undecidable	Yes
Rule-8.18.1	An object or sub-object must not be copied to an overlapping object	Mandatory	Undecidable	Yes
Rule-8.18.2	The result of an assignment operator should not be used	Advisory	Decidable	Yes
Rule-8.19.1	The comma operator should not be used	Advisory	Decidable	Yes
Rule-8.20.1	An unsigned arithmetic operation with constant operands should not wrap	Advisory	Decidable	Yes
Rule-9.2.1	An Explicit type conversion shall not be an expression statement	Required	Decidable	Yes
Rule-9.3.1	The body of an iteration-statement or a selection-statement shall be a compound-statement	Required	Decidable	Yes
Rule-9.4.1	All if ... else if constructs shall be terminated with an else statement	Required	Decidable	Yes
Rule-9.4.2	The structure of a switch statement shall be appropriate	Required	Decidable	Yes
Rule-9.5.1	Legacy for statements should be simple	Advisory	Decidable	Yes
Rule-9.5.2	A for-range-initializer shall contain at most one function call	Required	Decidable	Yes
Rule-9.6.1	The goto statement should not be used	Advisory	Decidable	Yes
Rule-9.6.2	A goto statement shall reference a label in a surrounding block	Required	Decidable	Yes
Rule-9.6.3	The goto statement shall jump to a label declared later in the function body	Required	Decidable	Yes
Rule-9.6.4	A function declared with the [[noreturn]] attribute shall not return	Required	Undecidable	Yes

Id	Description	Category	Analysis	Enforced
Rule-9.6.5	A function with non-void return type shall return a value on all paths	Required	Decidable	Yes
Rule-10.0.1	A declaration should not declare more than one variable or member variable	Advisory	Decidable	Yes
Rule-10.1.1	The target type of a pointer or lvalue reference parameter should be const-qualified appropriately	Advisory	Decidable	Yes
Rule-10.1.2	The volatile qualifier shall be used appropriately	Required	Decidable	Yes
Rule-10.2.1	An enumeration shall be defined with an explicit underlying type	Required	Decidable	Yes
Rule-10.2.2	Unscoped enumerations should not be declared	Advisory	Decidable	Yes
Rule-10.2.3	The numeric value of an unscoped enumeration with no fixed underlying type shall not be used	Required	Decidable	Yes
Rule-10.3.1	There should be no unnamed namespaces in header files	Advisory	Decidable	Yes
Rule-10.4.1	The asm declaration shall not be used	Required	Decidable	Yes
Rule-11.3.1	Variables of array type should not be declared	Advisory	Decidable	Yes
Rule-11.3.2	The declaration of an object should contain no more than two levels of pointer indirection	Advisory	Decidable	Yes
Rule-11.6.1	All variables should be initialized	Advisory	Decidable	Yes
Rule-11.6.2	The value of an object must not be read before it has been set	Mandatory	Undecidable	Yes
Rule-11.6.3	Within an enumerator list, the value of an implicitly specified enumeration constant shall be unique	Required	Decidable	Yes
Rule-12.2.1	Bit-fields should not be declared	Advisory	Decidable	Yes
Rule-12.2.2	Bit-fields shall have an appropriate type	Required	Decidable	Yes
Rule-12.2.3	A named bit-field with signed integer type shall not have a length of one bit	Required	Decidable	Yes
Rule-12.3.1	The union keyword shall not be used	Required	Decidable	Yes
Rule-13.1.1	Classes should not be inherited virtually	Advisory	Decidable	Yes

Id	Description	Category	Analysis	Enforced
Rule-13.1.2	An accessible base class shall not be both virtual and non-virtual in the same hierarchy	Required	Decidable	Yes
Rule-13.3.1	User-declared member functions shall use the virtual, override and final specifiers appropriately	Required	Decidable	Yes
Rule-13.3.2	Parameters in an overriding virtual function shall not specify different default arguments	Required	Decidable	Yes
Rule-13.3.3	The parameters in all declarations or overrides of a function shall either be unnamed or have identical names	Required	Decidable	Yes
Rule-13.3.4	A comparison of a potentially virtual pointer to member function shall only be with nullptr	Required	Decidable	Yes
Rule-14.1.1	Non-static data members should be either all private or all public	Advisory	Decidable	Yes
Rule-15.0.1	Special member functions shall be provided appropriately	Required	Decidable	Yes
Rule-15.0.2	User-provided copy and move member functions of a class should have appropriate signatures	Advisory	Decidable	Yes
Rule-15.1.1	An object's dynamic type shall not be used from within its constructor or destructor	Required	Undecidable	Yes
Rule-15.1.2	All constructors of a class should explicitly initialize all of its virtual base classes and immediate base classes	Advisory	Decidable	Yes
Rule-15.1.3	Conversion operators and constructors that are callable with a single argument shall be explicit	Required	Decidable	Yes
Rule-15.1.4	All direct, non-static data members of a class should be initialized before the class object is accessible	Advisory	Decidable	Yes
Rule-15.1.5	A class shall only define an initializer-list constructor when it is the only constructor	Required	Decidable	Yes
Rule-16.5.1	The logical AND and logical OR operators shall not be overloaded	Required	Decidable	Yes
Rule-16.5.2	The address-of operator shall not be overloaded	Required	Decidable	Yes

Id	Description	Category	Analysis	Enforced
Rule-16.6.1	Symmetrical operators should only be implemented as non-member functions	Advisory	Decidable	Yes
Rule-17.8.1	Function templates shall not be explicitly specialized	Required	Decidable	Yes
Rule-18.1.1	An exception object shall not have pointer type	Required	Decidable	Yes
Rule-18.1.2	An empty throw shall only occur within the compound-statement of a catch handler	Required	Decidable	Yes
Rule-18.3.1	There should be at least one exception handler to catch all otherwise unhandled exceptions	Advisory	Decidable	Yes
Rule-18.3.2	An exception of class type shall be caught by const reference or reference	Required	Decidable	Yes
Rule-18.3.3	Handlers for a function-try-block of a constructor or destructor shall not use non-static members from their class or its bases	Required	Decidable	Yes
Rule-18.4.1	Exception-unfriendly functions shall be noexcept	Required	Decidable	Yes
Rule-18.5.1	A noexcept function should not attempt to propagate an exception to the calling function	Advisory	Undecidable	Yes
Rule-18.5.2	Program-terminating functions should not be used	Advisory	Decidable	Yes
Rule-19.0.1	A line whose first token is # shall be a valid preprocessing directive	Required	Decidable	Yes
Rule-19.0.2	Function-like macros shall not be defined	Required	Decidable	Yes
Rule-19.0.3	#include directives should only be preceded by preprocessor directives or comments	Advisory	Decidable	Yes
Rule-19.0.4	#undef should only be used for macros defined previously in the same file	Advisory	Decidable	Yes
Rule-19.1.1	The defined preprocessor operator shall be used appropriately	Required	Decidable	Yes
Rule-19.1.2	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

Id	Description	Category	Analysis	Enforced
Rule-19.1.3	All identifiers used in the controlling expression of #if or #elif preprocessing directives shall be defined prior to evaluation	Required	Decidable	Yes
Rule-19.2.1	Precautions shall be taken in order to prevent the contents of a header file being included more than once	Required	Decidable	Yes
Rule-19.2.2	The #include directive shall be followed by either a <filename> or "filename" sequence	Required	Decidable	Yes
Rule-19.2.3	The ', " or \\ characters and the /* or // character sequences shall not occur in a header file name	Required	Decidable	Yes
Rule-19.3.1	The # and ## preprocessor operators should not be used	Advisory	Decidable	Yes
Rule-19.3.2	A macro parameter immediately following a # operator shall not be immediately followed by a ## operator	Required	Decidable	Yes
Rule-19.3.3	The argument to a mixed-use macro parameter shall not be subject to further expansion	Required	Decidable	Yes
Rule-19.3.4	Parentheses shall be used to ensure macro arguments are expanded appropriately	Required	Decidable	Yes
Rule-19.3.5	Tokens that look like a preprocessing directive shall not occur within a macro argument	Required	Decidable	Yes
Rule-19.6.1	The #pragma directive and the _Pragma operator should not be used	Advisory	Decidable	Yes
Rule-21.2.1	The library functions atof, atoi, atol and atoll from <cstdlib> shall not be used	Required	Decidable	Yes
Rule-21.2.2	The string handling functions from <cstring>, <cstdlib>, <cwchar> and <cinttypes> shall not be used	Required	Decidable	Yes
Rule-21.2.3	The library function system from <cstdlib> shall not be used	Required	Decidable	Yes
Rule-21.2.4	The macro offsetof shall not be used	Required	Decidable	Yes
Rule-21.6.1	Dynamic memory should not be used	Advisory	Undecidable	Yes
Rule-21.6.2	Dynamic memory shall be managed automatically	Required	Decidable	Yes

Id	Description	Category	Analysis	Enforced
Rule-21.6.3	Advanced memory management shall not be used	Required	Decidable	Yes
Rule-21.6.4	If a project defines either a sized or unsized version of a global delete operator, then both shall be defined	Required	Decidable	Yes
Rule-21.6.5	A pointer to an incomplete class type shall not be deleted	Required	Decidable	Yes
Rule-21.10.1	The features of <cstdarg> shall not be used	Required	Decidable	Yes
Rule-21.10.2	The standard header file <csetjmp> shall not be used	Required	Decidable	Yes
Rule-21.10.3	The facilities provided by the standard header file <csignal> shall not be used	Required	Decidable	Yes
Rule-22.3.1	The assert macro shall not be used with a constant-expression	Required	Decidable	Yes
Rule-22.4.1	The literal value zero shall be the only value assigned to errno	Required	Decidable	Yes
Rule-23.11.1	The raw pointer constructors of std::shared_ptr and std::unique_ptr should not be used	Advisory	Decidable	Yes
Rule-24.5.1	The character handling functions from <cctype> and <cwctype> shall not be used	Required	Decidable	Yes
Rule-24.5.2	The C++ Standard Library functions memcpy, memmove and memcmp from <cstring> shall not be used	Required	Decidable	Yes
Rule-25.5.1	The setlocale and std::locale::global functions shall not be called	Required	Decidable	Yes
Rule-25.5.2	The pointers returned by the C++ Standard Library functions localeconv, getenv, setlocale or strerror must only be used as if they have pointer to const-qualified type	Mandatory	Decidable	Yes
Rule-25.5.3	The pointer returned by the C Standard Library functions asctime, ctime, gmtime, localtime, localeconv, getenv, setlocale or strerror must not be used following a subsequent call to the same function	Mandatory	Undecidable	Yes
Rule-26.3.1	std::vector should not be specialized with bool	Advisory	Decidable	Yes

Id	Description	Category	Analysis	Enforced
Rule-28.3.1	Predicates shall not have persistent side effects	Required	Undecidable	Yes
Rule-28.6.1	The argument to std::move shall be a non-const lvalue	Required	Decidable	Yes
Rule-28.6.2	Forwarding references and std::forward shall be used together	Required	Decidable	Yes
Rule-28.6.3	An object shall not be used while in a potentially moved-from state	Required	Decidable	Yes
Rule-28.6.4	The result of std::remove, std::remove_if, std::unique and empty shall be used	Required	Decidable	Yes
Rule-30.0.1	The C Library input/output functions shall not be used	Required	Decidable	Yes
Rule-30.0.2	Reads and writes on the same file stream shall be separated by a positioning operation	Required	Undecidable	Yes

Directive Enforcement

'A directive is a guideline for which it is not possible to provide the full description necessary to perform a check for compliance.' MISRA C++:2023 Section 3.1. Due to the nature of directives, they are, in theory, not statically enforceable, however, the use of the tools can supply some assistance in checking.

Id	Description	Assisted
Dir-0.3.1	Floating point arithmetic should be used appropriately	Assisted
Dir-0.3.2	A function call shall not violate the function's preconditions	Assisted
Dir-5.7.2	Sections of code should not be 'commented out'	Assisted
Dir-15.8.1	User-provided copy assignment and move assignment operators shall handle self-assignment	Assisted