GRAMMATECH

MISRA C++:2008 GUIDELINES FOR THE USE OFTHE C++ LANGUAGE IN CRITICAL SYSTEMS CODESONAR® 6.2

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 writtenin ISO C for embedded systems. It is used in a wide range of industries, including automotive, aerospace, medical devices, and industrial control.

CodeSonar 6.2 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 mappingfor 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 anyMISRA C++:2008 rule or directive.

For more information on MISRA C++:2008:

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

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

Note-All CodeSonar MISRA mappings are close.

Misra C++ 2008 ID	Closely Mapped CodeSonar 6.2 Classes
0-1-1	Unreachable Data Flow Unexercised Call Unexercised Computation Unexercised Control Flow Unreachable Conditional Unexercised Conditional Unexercised Data Flow Unreachable Call Unreachable Computation Unreachable Control Flow

0-1-2	Unreachable Data Flow Unexercised Call Unexercised Computation Unexercised Control Flow Unreachable Conditional Unexercised Conditional Unexercised Data Flow Unreachable Call Unreachable Computation Unreachable Control Flow Unused Parameter
0-1-3	Unused Variable
0-1-4	Unused Variable
0-1-5	Unused Type
0-1-6	Unused Value
0-1-7	Ignored Return Value
0-1-8	Function Call Has No Effect
0-1-9	Function Call Has No EffectUseless Assignment Unused Value
0-1-11	Unused Parameter
0-1-12	Unused Parameter
0-2-1	Overlapping Memory Regions
0-3-2	Ignored Return Value
2-3-1	Trigraph
2-7-1	/* in Comment
2-7-2	Commented-out Code
2-7-3	Commented-out Code
2-10-1	Typographically Ambiguous Identifiers
2-10-2	Non-distinct Identifiers: Nested Scope
2-10-3	Non-unique Identifiers: Typedef
2-10-4	Non-unique Identifiers: Tag
2-10-5	Non-unique Identifiers: Internal NameLibrary Function Override
2-10-6	Non-distinct Identifiers: Same Scope
2-13-2	Octal Constant
2-13-3	Missing Literal Suffix
2-13-4	Confusing Literal Suffix

3-1-1	Object Defined in Header File Function Defined in Header File
3-1-2	Nested Function Declaration
3-1-3	Extern Array Without Size
3-2-1	Global Variable Declared with Different Types Inconsistent Object Declarations Inconsistent Function Declarations
3-2-2	Missing External Definition Multiple External Definitions
3-2-3	Multiple Declarations of a Global Missing External Declaration Multiple External Declarations
3-2-4	Missing External Definition Multiple External Definitions
3-4-1	Scope Could Be Local Static
3-9-1	Global Variable Declared with Different Types Inconsistent Object Declarations Inconsistent Function Declarations
3-9-2	Basic Numerical Type Used
3-9-3	Float Pointer Conversion
4-5-1	Inappropriate Operand Type
4-5-2	Inappropriate Operand Type
4-5-3	Inappropriate Character Arithmetic Inappropriate Operand Type
4-10-1	NULL Used as Integer
4-10-2	Coercion: Integer Constant to Pointer
5-0-2	Missing Parentheses
5-0-3	Expression Value Widened by Other Operand Inappropriate Assignment Type Expression Value Widened by Assignment Mismatched Operand Types
5-0-4	Coercion Alters Value
5-0-5	Expression Value Widened by Other Operand Inappropriate Assignment Type Expression Value Widened by Assignment Mismatched Operand Types
5-0-6	Coercion Alters Value Inappropriate Assignment Type
5-0-7	Inappropriate Cast Type: Expression
5-0-8	Inappropriate Cast Type: Expression
5-0-9	Cast Alters Value

5-0-10	Risky Integer Promotion
5-0-11	Inappropriate Character Arithmetic Inappropriate Operand Type Inappropriate Assignment Type Mismatched Operand Types
5-0-12	Inappropriate Character Arithmetic Inappropriate Operand Type Inappropriate Assignment Type Mismatched Operand Types
5-0-13	Condition Is Not Boolean
5-0-14	Inappropriate Operand Type
5-0-15	Pointer Arithmetic Subtraction of Unrelated Pointers
5-0-16	Type Underrun Buffer Underrun Buffer Overrun Pointer Past End of Object Type Overrun Tainted Buffer Access Pointer Before Beginning of Object
5-0-17	Subtraction of Unrelated Pointers
5-0-18	Comparison of Unrelated Pointers
5-0-19	Too Much Indirection in Declaration
5-0-20	Inappropriate Operand Type
5-0-21	Inappropriate Operand Type
5-2-4	C-style Cast Cast Alters Value Dangerous Function Cast
5-2-5	Cast Removes const Qualifier Cast Removes volatile Qualifier
5-2-6	Dangerous Function Cast Function Pointer Conversion Conversion from Function Pointer
5-2-7	Conversion: Pointer to Incomplete Conversion: Void Pointer to Object PointerConversion from Function Pointer Cast: Object Pointers Conversion: Pointer/Integer
5-2-8	Conversion: Void Pointer to Object PointerConversion: Pointer/Integer
5-2-9	Conversion: Pointer/Integer

5-2-10	Side Effects in Expression with Decrement Side Effects in Expression with Increment
5-2-11	Confusing Operator Overload
5-2-12	Array to Pointer Decay
5-3-1	Inappropriate Operand Type
5-3-2	Inappropriate Operand Type
5-3-3	Confusing Operator Overload
5-3-4	Side Effects in sizeof
5-8-1	Negative Shift Amount Shift Amount Exceeds Bit Width
5-14-1	Side Effects in Logical Operand
5-18-1	Use of Comma Operator
6-2-1	Assignment Result in Expression
6-2-2	Floating Point Equality
6-3-1	Body Is Not Compound Statement
6-4-1	Body Is Not Compound Statement
6-4-2	Missing Final else
6-4-3	Malformed switch Statement
6-4-4	Misplaced case
6-4-5	Missing break
6-4-6	Missing default Misplaced default
6-4-7	Boolean switch Expression
6-4-8	Too Few Cases in switch
6-5-1	Float-typed Loop Counter
6-5-3	Malformed for-loop Condition Malformed for-loop Initialization
6-5-4	Malformed for-loop Step
6-5-5	Malformed for-loop ConditionMalformed for- loop Step
6-6-1	Label Not In Enclosing Block
6-6-2	Backwards goto
6-6-3	Continue Statement
6-6-4	Multiple Abnormal Loop Exits
6-6-5	Misplaced Return StatementMultiple Return Statements
7-1-1	Variable Could Be const

7-1-2	Pointed-to Type Could Be const
7-2-1	Cast Alters Value Coercion Alters Value
7-3-1	Inappropriate Declaration in Global Namespace
7-3-3	Anonymous Namespace in Header File Misplaced Using Declaration
7-3-4	Using Directive Using Directive in Header File
7-3-6	Using Declaration in Header File Using Directive in Header File
7-4-2	Assembly Pragma
7-4-3	Inline Assembly Code Mixed Assembly and Code
7-5-1	Return Pointer to Local
7-5-4	Recursion
8-3-1	Method Default Value Mismatch
8-4-1	Ellipsis
8-4-2	Inconsistent Function Declarations
8-4-3	Missing Return Statement
8-4-4	Implicit Address of Function Varargs Function Cast Dangerous Function Cast Conversion from Function Pointer Function Pointer Conversion Conversion to Function Pointer
8-5-1	Uninitialized Variable
8-5-2	Missing Braces in Initialization
8-5-3	Inconsistent Enumerator Initialization
9-3-3	Member Function Could Be const Member Function Could Be static
9-5-1	Union Type
9-6-2	Inappropriate Bit-field Type Bit-field Signedness Not Explicit
9-6-3	Inappropriate Bit-field Type Bit-field Signedness Not Explicit
9-6-4	Bit-field Too Short
10-1-1	Virtual Base Class
10-1-2	Virtual Base Class not In Diamond
10-1-3	Virtual and Non-Virtual Base Class

12-1-1	Virtual Call in Constructor Virtual Call in Destructor
12-8-2	Object Slicing
15-3-5	Object Slicing
15-3-6	Unreachable Catch
15-3-7	Unreachable Catch

16-1-2	No Matching #endif No Matching #if
16-2-4	Dangerous Include File Name
16-2-5	Dangerous Include File Name
16-2-6	Malformed #include
16-3-2	Macro Uses # Operator Macro Uses ## Operator
17-0-1	Macro Undefinition of Reserved NameMacro Definition of Reserved Name Library Function Override
17-0-2	Declaration of Reserved Name
17-0-3	Declaration of Reserved Name
17-0-5	Use of setjmp Use of longjmp

18-0-1	Use of <setjmp.h> Use of <tgmath.h> Use of <signal.h>
18-0-2	Use of atoll Use of atoi Use of atof Use of atol
18-0-3	Use of exit Use of abort Use of system Use of getenv
18-0-4	Use of <time.h> Time/Date Function Potential Timebomb
18-0-5	Use of strcmp Use of strlen Use of strcpy Use of strstr Use of strpbrk Use of strrchr Use of strcat Use of strchr Use of strcoll Use of strtok Use of strspn Use of strcspn
18-2-1	Use of offsetof

18-4-1	Dynamic Allocation After Initialization Use of <stdlib.h> Allocator/Deallocator Macro Use of <stdlib.h> Allocator/Deallocator
18-7-1	Use of signal
27-0-1	Use of <wchar.h> Input/Output Use of <stdio.h> Input/Output Macro Use of <wchar.h> Input/Output Macro Use of <stdio.h> Input/Output

GrammaTech is a leading global provider of application testing (AST) solutions used by the world'smost security conscious organizations to detect, measure, analyze and resolve vulnerabilitiesfor 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.