本対応表は、ESCRのルールと内容的に関係するMISRA・CERT・CWEのルールを挙げたものです。

ESCRのルールでグレーの欄はC++言語に対するものです。

作法詳細		ルール		MISRAルールとの関係			CERT C	CERT C++	CWE	
				C:2004	C:2012	C++:2008				
[信頼性 1] R1 領域は初期化し、大きさに気を付けて使用する。										
R1.1	領域は、初期化してから使用する。	R1.1.1	自動変数は宣言時に初期化する。または値を使用する直前に初期値を代入する。	9.1	R9.1	8-5-1	EXP33-C	EXP53-CPP	CWE-119 CWE-456 CWE665	
		R1.1.2	const 型変数は、宣言時に初期化する。						CWE-456	C版のみ
R1.2	初期化は過不足無いことが分かるように記述する。	R1.2.1	要素数を指定した配列の初期化では、初期値の数は、指定した要素数と一致させる。				ARR02-C STR11-C STR31-C		CWE-119 CWE-120 CWE-193	C版のみ
		R1.2.2	列挙型(enum型)のメンバの初期化は、定数を全く指定しない、すべて指定する、または最初のメンバだけを指定する、のいずれかとする。	9.3	R8.12	8-5-3	INT09-C		CWE-665	
R1.3	ポインタの指す範囲に気を付ける。	R1.3.1	(1)ポインタへの整数の加減算(++、-- も含む)は使用せず、確保した領域への参照・代入には [ ] を用いる配列形式で行う。 (2)ポインタへの整数の加減算(++、-- も含む)は、ポ インタが配列を指 している場合だけとし、結果は配列の範囲内を指すようにする。	17.1 17.4	R18.1 R18.4	5-0-15 5-0-16	ARR30-C ARR37-C ARR39-C	ARR30-C ARR37-C ARR39-C	CWE-119 CWE-122 CWE-129 CWE-468 CWE469 CWE-788 CWE-823	
		R1.3.2	ポインタ同士の減算は、同じ配列の要素を指すポイ ンタにだけ使用する。	17.2	R18.2	5-0-17	ARR36-C	ARR36-C	CWE-469	
		R1.3.3	ポインタ同士の比較は、同じ配列の要素、または同 じ構造体のメンバを指すポインタにだけ使用する。	17.3	R18.3	5-0-18	ARR36-C	ARR36-C	CWE-188 CWE-469	
		R1.3.4	restrict 型修飾子は使用しない。		R8.14		EXP43-C			C版のみ
R1.4	オブジェクトは完全に構築してから使用する。	R1.4.1	コンストラクタでは、すべてのデータメンバを初期化する。初期化の方法は次の通りとする。 (1)初期化は、コンストラクタ初期 化子で行う。ただし、クラス型以外の複数のメンバを同じ値 で 初期化する場合はこの限りではない。 (2)コンストラクタ初期化子では、基底クラス、データメンバをその宣言順に記述する。 (3)コンストラクタ初期化子では、初期化のために他のデータメンバを使用しない。または、他のデータメンバを使用する場合は、そのデータメンバより前に宣言されたデータメンバだけとする。			8-5-1		EXP53-CPP OOP53-CPP		
		R1.4.2	コピーコンストラクタとコピー代入演算子では、非静的データメンバすべてを複写する。							
		R1.4.3	データメンバを読取り参照するメンバ関数の呼出しは、コンストラクタではオブジェクトが完全に初期化されてからとし、デストラクタではオブジェクトの解体が始まる前とする。							
		R1.4.4	コンストラクタとデストラクタでは、仮想関数を呼び出さない。			12-1-1		OOP50-CPP		
		R1.4.5	コンストラクタとコピー代入演算子は、オブジェクト構築の失敗に対応する。			15-1-1 15-3-1		MEM51-CPP MEM52-CPP	CWE-590 CWE-415 CWE-404 CWE-762	
		R1.4.6	コンストラクタとデストラクタに記述した catch ハ ンドラでは、データメンバを参照しない。			15-3-3		ERR53-CPP		
R1.5	オブジェクトの生成・破棄に気を付ける。	R1.5.1	対応する new と delete は同じ形式([ ]付きかどうか)を使用する。					MEM51-CPP	CWE-590 CWE-415 CWE-404 CWE-762	
[信頼性 2] R2 データは、範囲、大きさ、内部表現に気を付けて使用する。										
R2.1	内部表現に依存しない比較を行う。	R2.1.1	浮動小数点式は、等価または非等価の比較をしない。	13.3	D1.1	6-2-2	FLP00-C			
		R2.1.2	浮動小数点型変数はループ カウンタとして使用しな い。	13.4	R14.1	6-5-1	FLP00-C FLP30-C	FLP30-C		
		R2.1.3	構造体や共用体の比較に memcmp を使用しない。 クラス型オブジェクトの比較に memcmp を使用しない。				EXP42-C FLP37-C	EXP62-CPP FLP37-C OOP57-CPP	CWE-188 CWE-469	
R2.2	論理値などが区間として定義されている場合、その中の一点（代表的な実装値）と等しいかどうかで判定を行ってはならない。	R2.2.1	真偽を求める式の中で、真として定義した値と比較しない。							C版のみ
R2.3	データ型を揃えた演算や比較を行う。	R2.3.1	符号なし整数定数式は、結果の型で表現出来る範囲内で記述する。	12.11	R12.4	5-19-1	INT30-C	INT30-C	CWE-190	
		R2.3.2	条件演算子(? : 演算子)では、論理式は括弧で囲み、戻 り値は2つとも同じ型にする。				INT02-C			
		R2.3.3	ループカウンタとループ継続条件の比較に使用する変数は、同じ型にする。				INT02-C			
R2.4	演算精度を考慮して記述する。	R2.4.1	演算の型と演算結果の代入先の型が異なる場合は、期待する演算精度の型へキャストしてから演算する。	10.3 10.4	R10.8	5-0-7 5-0-8	INT02-C INT18-C FLP06-C FLP36-C	FLP36-C		
		R2.4.2	符号付きの式と符号なしの式が混在した算術演算、比較を行う場合は、期待する型へ明示的にキャストする。				INT02-C API09-C		CWE-195	
R2.5	情報損失の危険のある演算は使用しない。	R2.5.1	情報損失を起こす可能性のあるデータ型への代入 (=演算、関数呼出しの実引数渡し、関数復帰)や演算を行う場合は、問題がないことを確認し、問題がないことを明示するためにキャストを記述する。	10.1 10.2	R10.3	5-0-3	INT02-C INT08-C INT18-C INT31-C INT32-C FLP34-C FLP36-C STR34-C STR37-C API09-C	INT31-C FLP34-C FLP36-C STR34-C STR37-C	CWE-192 CWE-197 CWE-681 CWE-686 CWE-704	
		R2.5.2	単項演算子 "-" は符号なしの式には使用しない。	12.9	R10.1	5-3-2	INT30-C	INT30-C		
		R2.5.3	unsigned char 型、または unsigned short 型のデータ を、ビット反転(~)、もしくは左シフト(<<)する場合、結果の型に明示的にキャストする。	10.5		5-0-10	INT30-C	INT30-C		
		R2.5.4	シフト演算子の右辺の項はゼロ以上、左辺の項のビット幅未満でなければならない。	12.8	R12.2	5-8-1	INT34-C	INT34-C		

本対応表は、ESCRのルールと内容的に関係するMISRA・CERT・CWEのルールを挙げたものです。

ESCRのルールでグレーの欄はC++言語に対するものです。

作法詳細		ルール		MISRAルールとの関係			CERT C	CERT C++	CWE	
				C:2004	C:2012	C++:2008				
R2.6	対象データが表現可能な型を使用する。	R2.6.1	(1)ビットフィールドに使用する型は signed int と unsigned int だけとし、1ビット幅のビットフィールドが必要な場合は signed int 型でなく、unsigned int 型を使用する。 (2)ビットフィールドに使用する型は signed int、unsigned int、または _Bool とし、1ビット幅のビットフィールドが必要な場合は、unsigned int 型 または_Bool型を使用する。 (3)ビットフィールドに使用する型は signed int、unsigned int、_Bool または、処理系が許容している型のうち signed と unsigned を指定した型または enum 型を使用する。1ビット幅のビットフィールドが必要な場合は、unsigned を指定した型または _Bool 型を使用する。	6.4 6.5	R6.1 R6.2	9-6-2 9-6-3 9-6-4	INT12-C			
		R2.6.2	ビット列として使用するデータは、符号付き型ではなく、符号なし型で定義する。	12.7	R10.1	5-0-21	INT13-C			
R2.7	ポインタの型に気を付ける。	R2.7.1	(1) ポインタ型は、他のポインタ型及び整数型と相互に変換してはならない。ただし、データへのポインタ型におけるvoid* 型との相互変換は除く。 (1)ポインタ型は、他のポインタ型及び整数型に変換してはならない。また逆も行ってはならない。ただし、次の場合を除く。 ・データへのポインタ型から void*型への変換 ・派生関係にあるクラス型へのポインタ間の変換 (2) ポインタ型は、他のポインタ型、及びポインタ型のデータ幅未満の整数型と相互に変換してはならない。ただし、データへのポインタ型におけるvoid*型との相互変換は除く。 (2)ポインタ型は、他のポインタ型及びポインタ型のデータ幅未満の整数型に変換してはならない。ただし、次の場合を除く。 ・データへのポインタ型における void*型との相互の変換 ・派生関係にあるクラス型へのポインタ間の変換 (3) ポインタ型は、他のポインタ型、及びポインタ型のデータ幅未満の整数型と相互に変換してはならない。ただし、データへのポインタ型における、他のデータへのポインタ型及びvoid* 型との相互変換は除く。 (3)データへのポインタ型は、他のデータ型へのポインタ型及び void* 型に変換して良いが、関数型へのポインタは、他の関数型及びデータ型へのポインタ型に変換してはならない。ポインタ型を整数型に変換する場合、ポインタ型のデータ幅未満の整数型への変換を行ってはならない。	11.1 11.2 11.3 11.4	R11.1 R11.2 R11.3 R11.4 R11.5 R11.6 R11.7	5-2-6 5-2-7 5-2-8 5-2-9	EXP36-C EXP39-C INT36-C	EXP36-C EXP39-C INT36-C	CWE-119 CWE-466 CWE-587	
		R2.7.2	ポインタで指し示された型から const 修飾や volatile 修飾を取り除くキャストを行ってはならない。	11.5	R11.8	5-2-5	DCL13-C EXP05-C EXP08-C EXP32-C EXP40-C	EXP55-CPP		
		R2.7.3	ポインタが負かどうかの比較をしない。							
		R2.7.4	(1)派生クラスを指すポインタは、基底クラスを 指 すポインタに変換して良いが、基底クラスを指すポインタは派生クラスを指すポインタに変換してはならない。 (2)派生クラスを指すポインタは、基底クラスを指すポインタに変換して良い。また、基底クラスを指すポインタから派生クラスを指すポインタへは、dynamic_cast 演算子を利用すれば変換して良い。			5-2-2 5-2-3				
R2.8	宣言、使用、定義に矛盾が無いことをコンパイラがチェックできる書き方にする。	R2.8.1	引数をもたない関数は、引数の型を void として宣言する。				DCL20-C EXP37-C		CWE-628 CWE-686	C版のみ
		R2.8.2	(1)可変個引数を持つ関数を定義してはならない。 (2)可変個引数を持つ関数を使用する場合は、《処理系での動作を文書化し、使用する。》	16.1	R17.1	8-4-1	DCL10-C DCL11-C MSC39-C	DCL50-CPP EXP58-CPP MSC39-C	CWE-628	
		R2.8.3	1つのプロトタイプ宣言は1箇所に記述し、それが関数呼出し及び関数定義の両方から参照されるようにする。	8.1 8.3 8.4	R8.2 R8.3 R8.4 R17.3	3-2-1 3-2-2 3-3-1 3-9-1	DCL07-C DCL31-C DCL40-C EXP37-C	DCL40-C EXP37-C	CWE-628 CWE-686	
[信頼性 3] R3 動作が保証された書き方にする。										
R3.1	領域の大きさを意識した書き方にする。	R3.1.1	(1)配列の extern 宣言の要素数は必ず指定する。 (2)要素数が省略された初期化付き配列定義に対応した配列の extern 宣言を除き配列の extern 宣言の要素数は必ず指定する。	8.12	R8.11	3-1-3	ARR02-C		CWE-129	
		R3.1.2	配列を順次にアクセスするループの継続条件には、配列の範囲内であるかどうかの判定を入れる。				ARR30-C	ARR30-C	CWE-119 CWE-121 CWE-122 CWE-129 CWE-788	
		R3.1.3	指示付きの初期化子で初期化する配列のサイズは明示する。				ARR02-C			C版のみ
		R3.1.4	可変長配列型は使用しない。		R18.8		ARR32-C MEM05-C			C版のみ
		R3.1.5	(1)sizeof演算子はポインタ型の変数に用いてはならない。 (2)sizeof 演算子は配列型の引数に用いてはならない。							
R3.2	実行時にエラーになる可能性のある演算に対しては、エラーケースを迂回させる。	R3.2.1	除算や剰余算の右辺式は、0でないことを確認してから演算を行う。	21.1	D4.1	0-3-1	INT33-C	INT33-C	CWE-369	
		R3.2.2	ポインタは、ナルポインタでないことを確認してからポインタの指す先を参照する。	21.1	D4.1	0-3-1	EXP34-C	EXP34-C	CWE-476	
R3.3	関数呼出しではインターフェースの制約をチェックする。	R3.3.1	関数がエラー情報を戻す場合、エラー情報をテストしなければならない。	16.1	D4.7	0-3-2	EXP12-C FLP32-C ERR33-C	FLP32-C ERR33-C	CWE-252 CWE-390 CWE-391 CWE-682	
		R3.3.2	関数は、処理の開始前に引数の制約をチェックする。	20.3	D4.11	0-3-1	FLP32-C API00-C	FLP32-C	CWE-20 CWE-682	
R3.4	再帰呼出しは行わない。	R3.4.1	関数は、直接的か間接的かにかかわらず、その関数自身を呼び出してはならない。	16.2	R17.2	7-5-4	MEM05-C		CWE-674	

本対応表は、ESCRのルールと内容的に関係するMISRA・CERT・CWEのルールを挙げたものです。

ESCRのルールでグレーの欄はC++言語に対するものです。

作法詳細		ルール		MISRAルールとの関係			CERT C	CERT C++	CWE
				C:2004	C:2012	C++:2008			
R3.5	分岐の条件に気を付け、所定の条件以外が発生した場合の処理を記述する。	R3.5.1	if-else if 文は、最後に else 節を置く。 通常、else 条件が発生しないことが分かっている場合は、次のいずれかの記述とする。 《(i)else 節には、例外発生時の処理を記述する。 (ii)else 節には、プロジェクトで規定したコメントを入れる。》	14.10	R15.7	6-4-2	MSC01-C		
		R3.5.2	switch 文は、最後に default 節を置く。 通常、default 条件が発生しないことが分かっている場合は、次のいずれかの記述とする。 《(i)default 節には、例外発生時の処理を記述する。 (ii)default 節には、プロジェクトで規程したコメントを入れる。》	15.3	R16.4 R16.5	6-4-6	MSC01-C		CWE-478
		R3.5.3	ループカウンタの比較に等価演算子(==)、不等価演算子(!=)は使用しない。(「<=, >=, <, >」を使用する)			6-5-2	MSC21-C		
R3.6	評価順序に気を付ける。	R3.6.1	変数の値を変更する記述をした同じ式内で、その変数を参照、変更しない。	12.13	R13.3	5-2-10	EXP10-C EXP30-C	EXP50-CPP	
		R3.6.2	実引数並び、及び2項演算式に、副作用を持つ関数呼出し、volatile 変数を、複数記述しない。	12.2	R13.2	5-0-1	EXP10-C EXP30-C	EXP50-CPP	
		R3.6.3	sizeof 演算子は、副作用がある式に用いてはならない。				EXP44-C	EXP52-CPP	
R3.7	クラスの動作に気を付ける。	R3.7.1	資源を管理するクラスでは、コピーコンストラクタ・コピー代入演算子・デストラクタを定義する。						
		R3.7.2	基底クラスには仮想デストラクタを宣言する。					OOP52-CPP	
		R3.7.3	コピー代入演算子は次の規則に従うように定義する。 (1)自身への参照を返す。 (2)「T &operator = (const T &)」または「T &operator = (T)」のいずれかの形式で宣言する。返却型に const は付けない。 (3)自身への代入を可能にする。					OOP54-CPP OOP58-CPP	
		R3.7.4	仮想関数をオーバーライドするときは、デフォルト引数の値を変更しない。			8-3-1			
		R3.7.5	派生クラスでは、非仮想関数を再定義しない。						
		R3.7.6	ポリモーフィックな動作をさせるオブジェクトを関数引数とする場合には、参照渡しまたはポインタ渡しを使用する。					OOP51-CPP	
R3.8	例外の動作に気を付ける。	R3.8.1	(1)例外処理を使用しない。 (2)例外処理を使用する場合は、《使用方法をプロジェクトで規定する。》			15-0-1			
		R3.8.2	例外指定を記述しない。			15-4-1 15-5-2		ERR55-CPP	
		R3.8.3	NULL を throw しない。			15-1-2			
		R3.8.4	ポインタを例外として送出しない。			15-0-2			
		R3.8.5	デストラクタから例外を送出しない。			15-5-1		DCL57-CPP	
		R3.8.6	再送出するときは、送出式に引数を記述しない。						
		R3.8.7	例外オブジェクトは参照で捕捉する。			15-3-5		ERR61-CPP	
		R3.8.8	例外ハンドラは派生クラス、基底クラス、...(すべての例外の捕捉)の順に記述する。			15-3-6 15-3-7		ERR54-CPP	
		R3.8.9	main 関数ではすべての例外を漏れ無く捕捉する。			15-3-2		ERR51-CPP ERR58-CPP	CWE-754
R3.9	テンプレートの動作に気を付ける。	R3.9.1	テンプレート仮引数がポインタで参照される場合は、テンプレートの特殊化を用意する。						
R3.10	ラムダ式の動作に気をつける。	R3.10.1	ラムダ式では、デフォルトのキャプチャーモードを利用せず、利用する局所名はすべて明示する。					EXP54-CPP EXP61-CPP	
R3.11	スレッドもしくはシグナルを用いたプログラムでは、共有データのアクセス法に気をつける。	R3.11.1	並行処理ではvolatile を同期プリミティブとして使用しない。				SIG30-C SIG31-C SIG34-C SIG35-C	SIG30-C SIG31-C SIG34-C SIG35-C	CWE-479 CWE-662
			並行処理ではvolatileではなくstd::atomicを利用する。						
		R3.11.2	同一領域に割り当てられる可能性のあるビットフィールドに対して複数スレッドによるアクセスは行わない。もしくは、適切な排他制御を行う。				CON32-C SIG30-C SIG31-C SIG34-C SIG35-C	CON52-CPP SIG30-C SIG31-C SIG34-C SIG35-C	CWE-479 CWE-662
【保守性 1】M1 他人が読むことを意識する。									
M1.1	使用しない記述を残さない。	M1.1.1	使用しない関数、変数、引数、typedef、タグ、ラベル、マクロなどは宣言(定義)しない。			0-1-3 0-1-5 0-1-10 0-1-11 0-1-12	MSC07-C		CWE-561
		M1.1.2	(1)コードの一部を"コメントアウト"すべきでない。 (2)コードの一部をコメントアウトする場合は、//コメントを使用する。	2.4	D4.4	2-7-2 2-7-3	MSC04-C		
M1.2	紛らわしい書き方をしない。	M1.2.1	(1)1つの宣言文で宣言する変数は、1つとする(複数宣言しない)。 (2)同じような目的で使用する同じ型の自動変数は、1つの宣言文で複数宣言してもよいが、初期化する変数と初期化をしない変数を混在させてはならない。				DCL04-C		
		M1.2.2	適切な型を示す接尾語が使用出来る定数記述には、接尾語を付けて記述する。long 型整数定数を示す接尾語は大文字の"L"のみ使用する。			2-13-3 2-13-4	DCL16-C		
		M1.2.3	長い文字列リテラルを表現する場合には、文字列リテラル内で改行を使用せず、連続した文字列リテラルの連結を使用する。						
		M1.2.4	《名前空間の使用方法を規定する。》			7-3-4			
		M1.2.5	ネストした名前空間定義を行わない。						
		M1.2.6	関数テンプレートの明示的な特殊化を使用しない。			14-8-1			
		M1.2.7	デフォルト値指定を持つ引数を除いた宣言がデフォルトコピーコンストラクタと同じであるようなコンストラクタを定義しない。						

本対応表は、ESCRのルールと内容的に関係するMISRA・CERT・CWEのルールを挙げたものです。

ESCRのルールでグレーの欄はC++言語に対するものです。

作法詳細		ルール		MISRAルールとの関係			CERT C	CERT C++	CWE	
				C:2004	C:2012	C++:2008				
M1.3	特殊な書き方はしない。	M1.3.1	switch (式) の式には、真偽結果を求める式を記述し ない。	15.4	R16.7	6-4-7				
		M1.3.2	switch 文の case ラベル及び default ラベルは、switch 文本体の複文（その中に入れ子になった複文は除く）にのみ記述する。	15.1	R16.2	6-4-4	MSC20-C			
		M1.3.3	関数や変数の定義や宣言では型を明示的に記述する。				DCL31-C			C版のみ
M1.4	演算の実行順序が分かりやすいように記述する。	M1.4.1	&& や || 演算の右式と左式は二項演算を含まない式か ( ) で囲まれた式を記述する。ただし && 演算が連続して結合している場合や、|| 演算が連続して結合している場合は、&&式や || 式を ( ) で囲む必要は無い。	12.5	R12.1	5-0-2 5-2-1	EXP00-C			
		M1.4.2	《演算の優先順位を明示するための括弧の付け方を規定する。》	12.1	R12.1	5-0-2	EXP00-C EXP13-C			
M1.5	省略すると誤解を招きやすい演算は、明示的に記述する。	M1.5.1	関数識別子（関数名）には、前に & を付けるか、括弧付きの仮引数リスト（空でも可）を指定して使用しなければならない。	16.9		8-4-4				
		M1.5.2	条件判定の式では、0との比較は明示的にする	13.2	R14.4	5-0-13	EXP20-C			
			if 文やループにおける条件式は、型が明示的に bool型になるようにする。							
M1.6	領域は１つの利用目的に使用する。	M1.6.1	目的ごとに変数を用意する。	18.3						
		M1.6.2	（1）共用体を使用してはならない。	18.4	R19.2	9-5-1	EXP39-C	EXP39-C	CWE-119 CWE-188	
			（2）共用体を使用する場合は、書き込んだメンバで参照する。							
M1.7	名前を再使用しない。	M1.7.1	名前の一意性は、次の規則に従う。 1. 内部のスコープで宣言された識別子は外側のスコープで宣言された識別子を隠してはならない。 2. typedef 名（修飾がある場合はそれも含む）は固有の識別子でなければならない。 3. タグ名は一意な識別子でなければならない。 3. クラス名、共用体名、列挙体名（修飾がある場合はそれも含む）は固有の識別子でなければならない。 4. 外部結合をもつオブジェクトや関数を定義する識別子は一意でなければならない。 5. 内部結合をもつオブジェクトや関数を定義する識別子は一意にするべきである。 6. 構造体及び共用体のメンバ名を除いて、あるネームスペースの識別子を、他のネームスペースの識別子と同じ綴りにしてはいけない。 5. あるカテゴリの識別子を他のカテゴリの識別子と同じ綴りにしてはいけない。	5.2 5.3 5.4 5.5 5.6	R5.3 R5.6 R5.7 R5.8 R5.9	2-10-2 2-10-3 2-10-4 2-10-5 2-10-6	DCL01-C DCL23-C			C版のみ
		M1.7.2	標準ライブラリの関数名、変数名、マクロ名は再定義・再利用してはならない。また定義を解除してはならない。	20.1 20.2	R21.1 R21.2	17-0-1 17-0-2 17-0-3	DCL37-C	DCL51-CPP		
		M1.7.3	下線で始まる名前（変数）は定義しない。			17-0-1 17-0-2	DCL37-C	DCL51-CPP		
M1.8	勘違いしやすい言語仕様を使用しない。	M1.8.1	論理演算子 && または || の右側のオペランドには、副作用があってはならない。	12.4	R13.5	5-14-1	EXP02-C EXP30-C	EXP50-CPP	CWE-768	
		M1.8.2	Cマク ロは、波括弧 ’{ }’ で囲まれた初期化子、定数、（ ）で囲まれた式、型修飾子、記憶域クラス指定子、do-while-zero 構造にのみ展開されなければならない。	19.4	R20.4	16-2-2	PRE10-C			
			（2）マクロは、ヘッダファイルの重複取り込みの防止、型修飾子、記憶域クラス指定子にのみ展開されなければならない。							
		M1.8.3	#line は、ツールによる自動生成以外では使用しない。							
		M1.8.4	?? で始まる3文字以上の文字の並び、及び代替される字句表記は使用しない。	4.2	R4.2	2-3-1 2-5-1	PRE07-C			
		M1.8.5	0で始まる長さ2 以上の数字だけの列を定数として使用しない。	7.1	R4.1 R7.1	2-13-2	DCL18-C			
		M1.8.6	&& || ,（コンマ） & 演算子をオーバーロードしない。			5-2-11 5-3-3				
		M1.8.7	変換関数を定義しない。							
		M1.8.8	引数が1つのコンストラクタには、explicit 指定子を付ける。			12-1-3				
M1.9	特殊な書き方は意図を明示する。	M1.9.1	意図的に何もしない文を記述しなければならない場合はコメント、空になるマクロなどを利用し、目立たせる。	14.3	R15.6	6-2-3				
		M1.9.2	《無限ループの書き方を規定する。》							
M1.10	マジックナンバーを埋め込まない。	M1.10.1	意味のある定数は const 定数として定義して使用する。				DCL06-C		CWE-547	
M1.11	領域の属性は明示する。	M1.11.1	参照しかしない領域はconst であることを示す宣言を行う。	16.7	R8.13	7-1-1 7-1-2	DCL00-C			
		M1.11.2	他の実行単位により更新される可能性のある領域は volatile であることを示す宣言を行う。				DCL22-C			
		M1.11.3	《ROM 化するための変数宣言、定義のルールを規定する。》				DCL00-C			
M1.12	コンパイルされない文でも正しい記述を行う。	M1.12.1	プリプロセッサが削除する部分でも正しい記述を行 う。	19.16	R20.13					
［保守性 2］M2 修正誤りのないような書き方にする。										
M2.1	構造化されたデータやブロックは、まとまりを明確化する。	M2.1.1	配列や構造体を0以外で初期化する場合は、構造を示し、それに合わせるために波括弧 ″{ }″を使用しなけ ればならない。また、すべてが0の場合を除き、デー タは漏れ無く記述する。	9.2	R9.2 R9.3	8-5-2				
		M2.1.2	if、else if、else、while、do、for、switch 文の本体はブロック化する。	14.8 14.9	R15.6	6-3-1 6-4-1	EXP19-C			
M2.2	アクセス範囲や関連するデータは局所化する。	M2.2.1	1つの関数内でのみ使用する変数は関数内で変数宣言する。	8.7	R8.9		DCL15-C DCL19-C			
		M2.2.2	同一ファイル内で定義された複数の関数からアクセスされる変数は、ファイルスコープで static 変数宣言する	8.10 8.11	R8.7 R8.8		DCL15-C DCL19-C			
			（2）無名名前空間で宣言する。							
		M2.2.3	同一ファイル内で定義された関数からのみ呼ばれる関数は、static 関数とする。	8.10 8.11	R8.7 R8.8		DCL15-C DCL19-C			
			（2）無名名前空間で宣言する。							
		M2.2.4	関連する定数を定義するときは、#define や const 定数より enum を使用する。							
		M2.2.5	データメンバは private とする。			11-0-1				

本対応表は、ESCRのルールと内容的に関係するMISRA・CERT・CWEのルールを挙げたものです。
ESCRのルールでグレーの欄はC++言語に対するものです。

作法詳細		ルール		MISRAルールとの関係			CERT C	CERT C++	CWE	
				C:2004	C:2012	C++:2008				
[保守性 3] M3 プログラムはシンプルに書く。										
M3.1	構造化プログラミングを行う。	M3.1.1	繰返し文では、ループを終了させるための break 文の使用を最大でも1つだけに留めなければならない。	14.6	R15.4	6-6-4				
		M3.1.2	(1)goto 文を使用しない。 (2)goto 文を使用する場合、、飛び先は同じブロック、またはgoto 文を囲むブロック内で、かつgoto 文の後方に宣言されているラベルとする。	14.4	R15.1 R15.2 R15.3	6-6-2				
		M3.1.3	continue 文を使用してはならない。	14.5		6-6-3				欠番
		M3.1.4	(1)switch 文の case 節、default 節は、必ず break文で終了させる。 (2)switch 文の case 節や default 節を break 文で終了させない場合は、《プロジェクトでコメントを規定し》そのコメントを挿入する。	15.2	R16.3	6-4-5	MSC17-C			
		M3.1.5	(1)関数は、1つの return 文で終了させる。 (2)処理の途中で復帰する return 文は、異常復帰の場合のみとする。	14.7	R15.5	6-6-5				
M3.2	1つの文で1つの副作用とする。	M3.2.1	(1)コンマ式は使用しない。 (2)コンマ式は for 文の初期化式や更新式以外では使用しない。	12.10	R12.3	5-18-1				
		M3.2.2	1つの文に代入を複数記述しない。ただし、同じ値を複数の変数に代入する場合を除く。							
M3.3	目的の違う式は、分離して記述する。	M3.3.1	for 文の3つの式には、ループ制御にかかわるもののみを記述しなければならない。	13.5	R14.2					
		M3.3.2	for ループの中で繰返しカウンタとして用いる数値変数は、ループの本体内で変更してはならない。	13.6	R14.2	6-5-3 6-5-5				
		M3.3.3	(1)真偽を求める式の中で代入演算子を使用しない。 (2)真偽を求める式の中で代入演算子を使用しない。ただし、慣習的に使う表現は除く。	13.1	R13.4	6-2-1	EXP45-C	EXP45-C	CWE-480	
M3.4	複雑なポインタ演算は使用しない。	M3.4.1	3段階以上のポインタ指定は使用しない。	17.5	R18.5	5-0-19				
M3.5	複雑なクラス構造は使用しない。	M3.5.1	同じ階層構造で、アクセス可能な基底クラスに仮想継承と非仮想継承を混在させない。			10-1-3				
[保守性 4] M4 統一した書き方にする。										
M4.1	コーディングスタイルを統一する。	M4.1.1	《波括弧（{ }）や字下げ、空白の入れ方などのスタイルに関する規約を規定する。》							
		M4.1.2	キャストは、C++スタイルを使用する。ただし、void キャストは許す。			5-2-4				
M4.2	コメントの書き方を統一する。	M4.2.1	《ファイルヘッダコメント、関数ヘッダコメント、行末コメント、ブロックコメント、コピーライトなどの書き方に関する規約を規定する。》				MSC04-C			
M4.3	名前の付け方を統一する。	M4.3.1	《外部変数、内部変数などの命名に関する規約を規定 する。》				DCL02-C DCL16-C MSC09-C			
		M4.3.2	《ファイル名の命名に関する規約を規定する。》				PRE04-C MSC09-C			
M4.4	ファイル内の記述内容と記述順序を統一する。	M4.4.1	《ヘッダファイルに記述する内容（宣言、定義等）とその記述順序を規定する。》							
		M4.4.2	《ソースファイルに記述する内容（宣言、定義等）とその記述順序を規定する。》	8.6						
		M4.4.3	外部変数や関数（ファイル内でのみ使用する関数を除く）を使用したり定義したりする場合、宣言を記述したヘッダファイルをインクルードする。	8.8	R8.5	3-3-1	DCL36-C DCL40-C EXP37-C	DCL40-C EXP37-C	CWE-628 CWE-686	
		M4.4.4	外部変数は、複数箇所で定義しない。				DCL36-C			C版のみ
		M4.4.5	ヘッダファイルには、変数定義や関数定義を記述しない。							C版のみ
		M4.4.6	ヘッダファイルは重複取り込みに耐えうる作りとする。《そのための記述方法を規定する。》	19.15	D4.10	16-2-3	PRE06-C			
		M4.4.7	ソースファイルの#include の前やヘッダファイル内に、using 指令や名前空間の using 宣言を記述しない。ただしクラススコープや関数スコープに using 宣言を記述する場合を除く。			7-3-5 7-3-6				
		M4.4.8	《クラスメンバの記述順序を規定する。》							
M4.5	宣言の書き方を統一する。	M4.5.1	(1)関数プロトタイプ宣言では、すべての引数に名前を付けない（型だけとする）。 (2)関数プロトタイプ宣言では、すべての引数に名前を付ける。さらに引数の型と名前、及び戻り型は、関数定義とのじ通りに同じにする。	16.3 16.4	R8.2 R8.3	3-9-1 8-4-2	API08-C			
		M4.5.2	構造体タグの宣言と変数の宣言は別々に行う。 クラス・列挙の定義と変数の宣言は別々に行う。							
		M4.5.3	(1)構造体・共用体・配列の初期値式のリスト、及び列挙子リストの最後の“ }” の前に “,” を記述しない。 (2)構造体・共用体・配列の初期値式のリスト、及び列挙子リストの最後の“ }” の前に “,” を記述しない。ただし配列の初期化の初期値リストの最後の “ }”の前に “,” を書くことは許す。							
M4.6	ナルポインタの書き方を統一する。	M4.6.1	(1)ナルポインタには 0 を使用する。NULL はいかなる場合にも使用しない。 (2)ナルポインタには NULL を使用する。NULL はナルポインタ以外に使用しない。			4-10-1 4-10-2				
M4.7	前処理指令の書き方を統一する。	M4.7.1	演算子を含むマクロは、マクロ本体とマクロ引数を括弧で囲む。	19.10	R20.7	16-0-6	PRE01-C PRE02-C			
		M4.7.2	#ifdef、#ifndef、#if に対応する#else や #endif は、同一ファイル内に記述し、《プロジェクトで規定したコメントを入れ、対応関係を明確にする。》	19.17	R20.14	16-1-2				
		M4.7.3	#if や #elif で、マクロ名が定義済みかを調べる場合は、defined (マクロ名) または defined マクロ名 により定義済みかを調べる。			16-0-7				
		M4.7.4	#if や#elif で使用する defined 演算子は、defined（マクロ名）または defined マクロ名 という書き方以外では書かない。	19.14		16-1-1				欠番
		M4.7.5	マクロは、ブロック内で #define または #undef してはならない。	19.5		16-0-2				
		M4.7.6	#undef は使用してはならない。	19.6	R20.5	16-0-3				
		M4.7.7	#if または #elif 前処理指令の制御式は、0または1に評価されなければならない。		R20.8					

本対応表は、ESCRのルールと内容的に関係するMISRA・CERT・CWEのルールを挙げたものです。

ESCRのルールでグレーの欄はC++言語に対するものです。

作法詳細		ルール		MISRAルールとの関係			CERT C	CERT C++	CWE
				C:2004	C:2012	C++:2008			
M4.8	オーバーロードの書き方を統一する。	M4.8.1	演算子のオーバーロードは、演算子の本来の意味で同類の演算子を併せて定義する。			5-17-1			
		M4.8.2	クラスに対する new と delete を定義する場合は、標準で提供されている形式すべてを定義する。						
		M4.8.3	new と delete は対で定義する。					DCL54-CPP MEM51-CPP	CWE-590 CWE-415 CWE-404 CWE-762
[保守性 5] M5 試験しやすい書き方にする。									
M5.1	問題発生時の原因を調査しやすい書き方にする。	M5.1.1	《デバッグオプション設定時のコーディング方法と、リリースモジュールにログを残すためのコーディング方法を規定する。》				DCL03-C ERR06-C MSC11-C		
		M5.1.2	(1)前処理演算子 # と ## を使用してはならない。 (2)# 演算子の直後に続くマクロパラメータの直後に ## 演算子を続けない。	19.13 19.12	R20.10 R20.11	16-3-1 16-3-2	PRE05-C		
		M5.1.3	関数形式のマクロよりも、インライン関数を使用する。	19.7	D4.9	16-0-4	PRE00-C		
M5.2	動的なメモリ割り当ての使用に気を付ける。	M5.2.1	(1)動的メモリを使用しない。 (2)動的メモリを使用する場合は、《使用するメモリ量の上限、メモリ不足の場合の処理、及びデバッグ方法等を規定する。》	20.4	R21.3 D4.12	18-4-1	EXP33-C MEM30-C MEM31-C MEM35-C CON30-C	EXP53-CPP EXP54-CPP MEM30-C MEM31-C MEM35-C MEM50-CPP MEM51-CPP MEM52-CPP	CWE-119 CWE-131 CWE-190 CWE-401 CWE-404 CWE415 CWE-416 CWE-467 CWE-590 CWE-665 CWE-762
[移植性 1] P1 コンパイラに依存しない書き方にする。									
P1.1	拡張機能や処理系定義の機能は使用しない。	P1.1.1	(1)言語標準の規格外の機能は使用しない。 (2)言語標準の規格外の機能を使用する場合は、《使用する機能とその使い方を文書化する。》	1.1	R1.1 R1.2	1-0-1			
		P1.1.2	《使用する処理系定義の動作はすべて文書化しなければならない。》	3.1	D1.1				
		P1.1.3	他言語で書かれたプログラムを利用する場合、《そのインタフェースを文書化し、使用方法を規定する。》	1.3	D1.1	1-0-2		EXP56-CPP	
P1.2	言語規格で定義されている文字や拡張表記のみを使用する。	P1.2.1	プログラムの記述において、言語規格で規定している文字以外の文字を使用する場合、コンパイラの仕様を確認し《その使い方を規定する。》	3.2	D1.1	2-2-1			
		P1.2.2	言語規格で定義されている拡張表記だけを使用する。	4.1	R4.1	2-13-1			
P1.3	データ型の表現、動作仕様の拡張機能、及び処理系依存部分を確認し、文書化する。	P1.3.1	単なる(符号指定のない) char 型は、文字の値の格納(処理)にだけ使用し、符号の有無(処理系定義)に依存する処理が必要な場合は、符号を明記した unsigned char 型または signed char 型を利用する。	6.1 6.2	R10.1 R10.2 R10.3 R10.4	5-0-11 5-0-12	INT07-C STR00-C STR04-C		
		P1.3.2	列挙(enum)体のメンバは、int 型で表現可能な値で定義する。						
		P1.3.3	(1)ビットフィールドは使用しない。 (2)ビット位置が意識されたデータに対してはビットフィールドは使用しない。 (3)ビットフィールドの処理系定義の動作とパッキングに(プログラムが)依存している場合、《それは文書化しなければならない。》	3.5	D1.1	9-6-1	EXP11-C		
P1.4	ソースファイル取り込みについて、処理系依存部分を確認し、依存しない書き方にする。	P1.4.1	#include 指令の後には、<filename>または "filename" が続かなければならない。	19.3	R20.3	16-2-6			
		P1.4.2	《#include のファイル指定で、<> 形式と " " 形式の使い分け方を規定する。》	19.3	R20.3				
		P1.4.3	#include のファイル指定では、文字 ' 、 \ 、 " 、 /*、//、及び : は使用しない。	19.2	R20.2	16-2-4 16-2-5			
P1.5	コンパイル環境に依存しない書き方にする。	P1.5.1	#include のファイル指定では、絶対パスは記述しない。						
		P1.5.2	型や変数のサイズはsizeofを使って求める。						
[移植性 2] P2 移植性に問題のあるコードは局所化する。									
P2.1	移植性に問題のあるコードは局所化する。	P2.1.1	C言語からアセンブリ言語のプログラムを呼び出す場合、インラインアセンブリ言語のみが含まれる C言語の関数やインライン関数として表現する、またはマクロで記述するなど、《局所化する方法を規定する。》	2.1	D4.2 D4.3	7-4-3			
		P2.1.2	処理系が拡張しているキーワードは、《マクロを規定して》局所化して使用する。						
		P2.1.3	(1)char、int、long、long long、float、double 及び long double という基本型は使用しない。代わりに typedef した型を使用する。《プロジェクトで利用する typedef した型を規定する。》 (2)char、int、long、long long、float、double 及び long double という基本型を、そのサイズに依存する形式で使用する場合、各基本型を typedef した型を使用する。《プロジェクトで利用する typedef 型を規定する。》	6.3	D4.6	3-9-2			
[効率性 1] E1 資源や時間の効率を考慮した書き方にする。									
E1.1	資源や時間の効率を考慮した書き方にする。	E1.1.1	マクロ関数は、速度性能にかかわる部分内に閉じて使用する。				PRE00-C		
		E1.1.2	繰返し処理内で、変化のない処理を行わない。						
		E1.1.3	関数の引数として構造体ではなく、構造体ポインタを使用する。						
		E1.1.4	《switch 文とするか if 文とするかは、可読性と効率性を考えて選択方針を決定し、規定する。》						
		E1.1.5	ヘッダファイルには、変数定義・関数定義を記述しない。	8.5		3-1-1		DCL59-CPP	
		E1.1.6	データメンバの初期化は、コンストラクタ初期化子を使う。						
		E1.1.7	暗黙の型変換を起こさないように、型に対応したオーバーロード関数を定義する。						
		E1.1.8	インクリメント演算子とデクリメント演算子は、前置形式を使う。						
		E1.1.9	仮想継承は同じ基底クラスを持つクラスを多重継承している場合だけ使用して良い。			10-1-2			
		E1.1.10	テンプレート定義中に、テンプレート仮引数と関係 の無いコードを書かない。						

本対応表は、ESCRのルールと内容的に関係するMISRA・CERT・CWEのルールを挙げたものです。
ESCRのルールでグレーの欄はC++言語に対するものです。

作法詳細		ルール		MISRAルールとの関係			CERT C	CERT C++	CWE
				C:2004	C:2012	C++:2008			
組込みソフトウェアにありがちなコーディングミス									
1	意味のない式や文	例1	実行されない文を記述						
		例2	実行結果が使用されない文を記述						
		例3	実行結果が使用されない式を記述						
		例4	実引数で渡した値が使用されない						
2	誤った式や文	例1	誤った範囲指定						
		例2	範囲外の比較						
		例3	文字列の比較は== 演算では行えない						
		例4	関数の型とreturn 文の不整合				MSC37-C		
		例5	ポインタへの加減算の誤り				ARR39-C		
3	誤ったメモリの使用	例1	配列の範囲外の参照・更新				ARR30-C		
		例2	自動変数の領域のアドレスを呼出し元に渡してしまう誤り				DCL21-C DCL30-C EXP35-C MEM30-C		
		例3	動的メモリ解放後のメモリ参照				MEM30-C		
		例4	文字列リテラルを書き込む誤り				STR30-C		
		例5	複写サイズの指定誤り						
4	論理演算の勘違いによる誤り	例1	論理和とするところを論理積とした誤り						
		例2	論理積とするところを論理和とした誤り						
		例3	論理演算とするところをビット演算とした誤り				EXP17-C EXP46-C		
5	タイプミスによる誤り	例1	演算子を記述するべきところに= 演算子を記述				EXP45-C		
6	コンパイラによってはエラーにならないケースがある記述	例1	同名マクロの多重定義						
		例2	const 領域に書き込む誤り						