HKMC Secure C++

Perforce QAC for C++ 2026.1

(HKMC Secure C++ Rev. Date '22.01 Release 4.1)

Rule Enforcement Summary

		Total
a	Total Number of Rules	80
b	Total Number of 'Not Statically Enforceable' Rules (Assisted/Unassisted)	0
c	Total Number of Enforceable Rules (a-b)	80
d	Total Number of Enforced Rules	80
e	Total Number of Unenforced Rules (c-d)	0
f	Enforced Rules Percentage (d/c)	100%
g	Unenforced Rules Percentage (e/c)	0%

Rule Enforcement

Id	Description	Severity	Enforced
Code Error			
3.1 Declarations and initialization			
P-DCL-001	Do not define a C-style variadic function	High	Yes
P-DCL-002	Do not declare or define a reserved identifier	Low	Yes
P-DCL-003	Do not qualify a reference type with const or volatile	Low	Yes
P-DCL-004	Do not write syntactically ambiguous declarations	Low	Yes
P-DCL-005	Overload allocation and deallocation functions as a pair in the same scope	Low	Yes
P-DCL-006	Attention of data leaks when passing class objects across trust boundaries	Low	Yes

Id	Description	Severity	Enforced
P-DCL-007	Do not reenter of function during initialization of static objects	Low	Yes
P-DCL-008	Prevent termination due to exceptions in destructors or deallocation functions	Low	Yes
P-DCL-009	Do not modify the standard namespaces	High	Yes
P-DCL-010	Do not define an unnamed namespace in a header file	Medium	Yes
P-DCL-011	Compliance of ODR (One-Definition Rule)	High	Yes
3.2 Integers and characters			
P-INT-001	Do not cast to an out-of-range enumeration value	Medium	Yes
P-STR-001	Guarantee that storage for strings has sufficient space for character data and the null terminator	High	Yes
P-STR-002	Do not create a std::string with a null pointer	High	Yes
P-STR-003	Do not use invalid references, pointers, and iterators when referencing to basic_string	High	Yes
P-STR-004	Range check required when accessing string	High	Yes
3.3 Expressions			
P-EXP-001	Attention to the order of evaluation that causes the side effects	Medium	Yes
P-EXP-002	Do not delete an array through a pointer of the incorrect type	Low	Yes
P-EXP-003	Attention to use expressions that do not evaluate(calculate) operands	Low	Yes
P-EXP-004	Do not reference memory before initialized	High	Yes
P-EXP-005	Do not access an object of its lifetime	High	Yes
P-EXP-006	Do not access a CV variable through a CV (const or volatile) unqualified variable	Medium	Yes
P-EXP-007	Calling va_start requires passing an object of the appropriate type	Medium	Yes
P-EXP-008	Use offsetof() on valid types and members	Medium	Yes
P-EXP-009	Assurance that a lambda object is shorter than lifecycle of reference captured objects	High	Yes
P-EXP-010	Do not access the bits that are not part of the object's value	High	Yes

Id	Description	Severity	Enforced
P-EXP-011	Do not rely on the value of moved from object	Medium	Yes
3.4 Containers			
P-CTR-001	Guarantee that container indices and iterators are within the valid range	High	Yes
P-CTR-002	Do not use valid references, pointer, and iterators to reference element of a container	High	Yes
P-CTR-003	Overflow protection required when copying data	High	Yes
P-CTR-004	Use valid iterator ranges	High	Yes
P-CTR-005	Do not subtract iterators that do not refer to the same container	High	Yes
P-CTR-006	Do not use an additive operator on an iterator if the result would overflow	High	Yes
P-CTR-007	Do not use pointer arithmetic on polymorphic objects	High	Yes
3.5 Exception handling			
P-ERR-001	Do not abruptly terminate the program	Low	Yes
P-ERR-002	Need to handle all exceptions	Low	Yes
P-ERR-003	Do not use setjmp() or longjmp()	Low	Yes
P-ERR-004	Do not reference base classes or class data members in a constructor or destructor function-try-block handler	Low	Yes
P-ERR-005	Exception handling in order from lowest class to highest class	Medium	Yes
P-ERR-006	Specify the exception exactly	Low	Yes
P-ERR-007	Guarantee exception safety	High	Yes
P-ERR-008	Prevent to leak resources when handling exceptions	Low	Yes
P-ERR-009	Handle all exceptions before main() begins executing	Low	Yes
P-ERR-010	Catch exceptions by lvalue reference	Low	Yes
P-ERR-011	Detect errors when converting a string to a number	Medium	Yes
P-ERR-012	Undetected exception handling	Medium	Yes
P-ERR-013	Catch declaration for common exceptions	Medium	Yes

Id	Description	Severity	Enforced
P-ERR-014	Throw declarations for general exceptions	Medium	Yes
Code quality			
4.1 Memory management			
P-MEM-001	Do not reference to deallocated memory	High	Yes
P-MEM-002	Properly deallocate dynamically allocated resources	High	Yes
P-MEM-003	Detect and handle memory allocation errors	High	Yes
P-MEM-004	Explicitly construct and destruct objects when manaing object lifecycle	High	Yes
P-MEM-005	Provide placement new with properly aligned pointers to sufficient storage capacity	High	Yes
P-MEM-006	Do not store an already-owned pointer value in an unrelated smart pointer	High	Yes
P-MEM-007	Memory release missing after lifetime	Medium	Yes
4.2 Object Oriented Programming			
P-OOP-001	Do not invoke virtual functions from constructors or destructors	Low	Yes
P-OOP-002	Do not slice derived objects	Low	Yes
P-OOP-003	Do not delete a polymorphic object without a virtual destructor	Low	Yes
P-OOP-004	Write constructor member initializers in the fixed order	Medium	Yes
P-OOP-005	Correctly handle self-copy assignment	Low	Yes
P-OOP-006	Do not use pointer-to-member operators to access nonexistent members	High	Yes
P-OOP-007	Prefer special member functions and overloaded operators over the C standard library	High	Yes
P-OOP-008	Do not modify the source object in copy operators	Low	Yes
P-OOP-009	Public static field do not mark as final	Medium	Yes
P-OOP-010	Important data element public declaration	Medium	Yes
P-OOP-011	Access critical private variables through public methods	High	Yes

Id	Description	Severity	Enforced
4.3 Input and output			
P-FIO-001	Do not alternately input and output operations without calling intermediate positioning functions from the file stream	Low	Yes
P-FIO-002	Close file pointers that are no longer needed	Medium	Yes
4.4 Concurrency			
P-CON-001	Do not destroy a mutex while it is locked	Medium	Yes
P-CON-002	Unlocking mutex on exception	Low	Yes
P-CON-003	Prevent data races when accessing bit-fields from multiple threads	Medium	Yes
P-CON-004	Prevent deadlock by locking mutex in predefined order	Low	Yes
P-CON-005	Wrap function that can wake up falsely in a loop	Low	Yes
P-CON-006	Protect thread safety and liveness when using condition variables	Low	Yes
P-CON-007	Do not lock a non-recursive mutex that is already owned within the calling thread	Low	Yes
Other function			
5.1 Using random number generator			
P-MSC-001	Do not use std::rand() for generating pseudo random number	Medium	Yes
P-MSC-002	Proper seeding for random number generator	Medium	Yes
5.2 Return value			
P-MSC-003	Value returning function require returning values from all exit paths	Medium	Yes
P-MSC-004	Do not return from function declared [[noreturn]]	Medium	Yes
P-MSC-005	Return of stack variable address	Medium	Yes