LDRA

This page was automatically generated and should not be edited.

The information on this page was provided by outside contributors and has not been verified by SEI CERT.

The table below can be re-ordered, by clicking column headers.

Tool Version: 9.7.1

Checker	Guideline
1 J	EXP54-CPP. Do not access an object outside of its lifetime
1 Q	EXP50-CPP. Do not depend on the order of evaluation for side effects
2 D	MSC52-CPP. Value-returning functions must return a value from all exit paths
6 D	DCL56-CPP. Avoid cycles during initialization of static objects
9 S	EXP50-CPP. Do not depend on the order of evaluation for side effects
35 D	EXP50-CPP. Do not depend on the order of evaluation for side effects
36 S	MSC52-CPP. Value-returning functions must return a value from all exit paths
41 S	DCL50-CPP. Do not define a C-style variadic function
42 D	EXP54-CPP. Do not access an object outside of its lifetime
43 S	ERR52-CPP. Do not use setjmp() or longjmp()
44 S	OOP57-CPP. Prefer special member functions and overloaded operators to C Standard Library functions
44 S	MSC50-CPP. Do not use std::rand() for generating pseudorandom numbers
45 D	CTR50-CPP. Guarantee that container indices and iterators are within the valid range
45 D	MEM52-CPP. Detect and handle memory allocation errors
47 S	CTR50-CPP. Guarantee that container indices and iterators are within the valid range
50 D	ERR57-CPP. Do not leak resources when handling exceptions
53 D	EXP53-CPP. Do not read uninitialized memory
53 D	EXP54-CPP. Do not access an object outside of its lifetime
54 S	EXP52-CPP. Do not rely on side effects in unevaluated operands
56 D	ERR55-CPP. Honor exception specifications
56 D	ERR56-CPP. Guarantee exception safety
64 D	MEM51-CPP. Properly deallocate dynamically allocated resources
64 X	CTR50-CPP. Guarantee that container indices and iterators are within the valid range
66 X	CTR50-CPP. Guarantee that container indices and iterators are within the valid range
66 X	STR50-CPP. Guarantee that storage for strings has sufficient space for character data and the null terminator
67 D	EXP50-CPP. Do not depend on the order of evaluation for side effects
68 X	CTR50-CPP. Guarantee that container indices and iterators are within the valid range
69 D	EXP53-CPP. Do not read uninitialized memory
69 X	CTR50-CPP. Guarantee that container indices and iterators are within the valid range
70 S	CTR54-CPP. Do not subtract iterators that do not refer to the same container
70 X	CTR50-CPP. Guarantee that container indices and iterators are within the valid range
70 X	STR50-CPP. Guarantee that storage for strings has sufficient space for character data and the null terminator
71 D	ERR56-CPP. Guarantee exception safety

71 S	EXP54-CPP. Do not access an object outside of its lifetime
71 X	CTR50-CPP. Guarantee that container indices and iterators are within the valid range
71 X	STR50-CPP. Guarantee that storage for strings has sufficient space for character data and the null terminator
72 D	EXP50-CPP. Do not depend on the order of evaluation for side effects
77 D	EXP54-CPP. Do not access an object outside of its lifetime
79 X	CTR50-CPP. Guarantee that container indices and iterators are within the valid range
86 S	DCL51-CPP. Do not declare or define a reserved identifier
87 S	CTR54-CPP. Do not subtract iterators that do not refer to the same container
92 D	OOP50-CPP. Do not invoke virtual functions from constructors or destructors
112 D	MEM51-CPP. Properly deallocate dynamically allocated resources
122 S	ERR50-CPP. Do not abruptly terminate the program
133 S	EXP52-CPP. Do not rely on side effects in unevaluated operands
134 S	EXP50-CPP. Do not depend on the order of evaluation for side effects
169 S	EXP57-CPP. Do not cast or delete pointers to incomplete classes
203 S	EXP55-CPP. Do not access a cv-qualified object through a cv-unqualified type
206 S	OOP53-CPP. Write constructor member initializers in the canonical order
218 S	DCL51-CPP. Do not declare or define a reserved identifier
219 S	DCL51-CPP. Do not declare or define a reserved identifier
232 S	MEM51-CPP. Properly deallocate dynamically allocated resources
236 S	MEM51-CPP. Properly deallocate dynamically allocated resources
239 S	MEM51-CPP. Properly deallocate dynamically allocated resources
242 S	EXP55-CPP. Do not access a cv-qualified object through a cv-unqualified type
286 S	DCL59-CPP. Do not define an unnamed namespace in a header file
286 S	DCL60-CPP. Obey the one-definition rule
287 S	DCL60-CPP. Obey the one-definition rule
296 S	DCL53-CPP. Do not write syntactically ambiguous declarations
303 S	OOP52-CPP. Do not delete a polymorphic object without a virtual destructor
344 S	EXP55-CPP. Do not access a cv-qualified object through a cv-unqualified type
407 S	MEM51-CPP. Properly deallocate dynamically allocated resources
437 S	CTR54-CPP. Do not subtract iterators that do not refer to the same container
438 S	CTR54-CPP. Do not subtract iterators that do not refer to the same container
453 S	DCL57-CPP. Do not let exceptions escape from destructors or deallocation functions
455 S	ERR61-CPP. Catch exceptions by lvalue reference
467 S	OOP50-CPP. Do not invoke virtual functions from constructors or destructors
469 S	MEM51-CPP. Properly deallocate dynamically allocated resources
470 S	MEM51-CPP. Properly deallocate dynamically allocated resources
476 S	CTR50-CPP. Guarantee that container indices and iterators are within the valid range
483 S	MEM50-CPP. Do not access freed memory
483 S	MEM51-CPP. Properly deallocate dynamically allocated resources
484 S	MEM50-CPP. Do not access freed memory
484 S	MEM51-CPP. Properly deallocate dynamically allocated resources
485 S	MEM51-CPP. Properly deallocate dynamically allocated resources
489 S	CTR50-CPP. Guarantee that container indices and iterators are within the valid range
489 S	STR50-CPP. Guarantee that storage for strings has sufficient space for character data and the null terminator
512 S	DCL59-CPP. Do not define an unnamed namespace in a header file
527 S	ERR51-CPP. Handle all exceptions
527 S	ERR56-CPP. Guarantee exception safety

541 S	ERR54-CPP. Catch handlers should order their parameter types from most derived to least derived
549 S	ERR53-CPP. Do not reference base classes or class data members in a constructor or destructor function-try-block handler
554 S	EXP57-CPP. Do not cast or delete pointers to incomplete classes
556 S	ERR54-CPP. Catch handlers should order their parameter types from most derived to least derived
565 S	EXP54-CPP. Do not access an object outside of its lifetime
567 S	CTR55-CPP. Do not use an additive operator on an iterator if the result would overflow
567 S	CTR56-CPP. Do not use pointer arithmetic on polymorphic objects
580 S	DCL51-CPP. Do not declare or define a reserved identifier
597 S	MEM54-CPP. Provide placement new with properly aligned pointers to sufficient storage capacity
618 S	EXP62-CPP. Do not access the bits of an object representation that are not part of the object's value representation
631 S	EXP53-CPP. Do not read uninitialized memory
652 S	EXP53-CPP. Do not read uninitialized memory