

Exception Safety Solutions

- What is meant by "exception safe" code?
 - Code is “exception safe” if it behaves correctly in the presence of exceptions
- What is meant by "exception neutral" code?
 - Code is “exception neutral” if it does not handle exceptions itself but leaves them to be processed by the caller

- What is meant by the basic exception guarantee?
 - Code provides the basic exception guarantee if no resources are leaked when an exception is thrown
- What is meant by the strong exception guarantee?
 - Code provides the strong exception guarantee if the program reverts to its previous state when an exception is thrown

- What is meant by the no-throw guarantee?
 - Code provides the no-throw guarantee if it does not throw any exceptions which need to be handled by the caller
- To what extent does C++ provide these guarantees?
 - All STL containers provide the strong guarantee (except for a few special cases with insert operations)
 - All built-in operators and functions provide the no-throw guarantee (again, with a few exceptions)
 - C++ allows us to provide these guarantees with our own code

- In C++11, which keyword is used to promise that a function does not throw any exceptions?
 - `noexcept`
- How can we do this in earlier versions of C++?
 - `throw()` with an empty argument list
- Apart from providing the no-throw guarantee, how does this improve our code?
 - Makes the purpose of the function clearer
 - Helps the compiler optimize calls to this function