More Exceptions

Arnav Gupta

October 27, 2024

Contents

1 Derived Exception Type

A mechanism for inheritance of exception types.

Provides ability to handle an exception at different degrees of specificity along the hierarchy.

Higher-level code should catch general exception types to reduce tight coupling.

If a base and derived exception are present in different `catch` clauses, linear search requires that the derived one must be first.

Best to catch an exception by reference since it allows casting.

2 Catch Any

Mechanism to match any exception propagating through a guarding block.

For termination, catch any is used as a general cleanup when a non-specific exception occurs.

3 Exception Parameters

Allows passing information from the raise to a handler.

Inform a handler about details of the exception and to modify the raise site to fix an exceptional situation.

Parameters are defined inside the exception.

4 Exception List

Part of a routine's prototype specifying which exception types may propagate from the routine to its caller. Allows:

- static detection of a raised exception not handled locally or by its caller
- runtime detection where the exception may be converted into a special **failure exception** or the program terminated

Can be checked as exception types or routines. With exception types, less reuse is possible.

Determining an exception list for a routine can become impossible for concurrent exceptions since they can propagate at any time.

5 Destructor

Implictly `noexcept`, but can raise an exception if explicitly marked `noexcept(false)`.

6 Multiple Exceptions

An exception handler can generate an arbitrary number of nested exceptions.

Only destructor code can intervene during propagation, so a destructor cannot raise an exception during propagation, it can only start propagation.