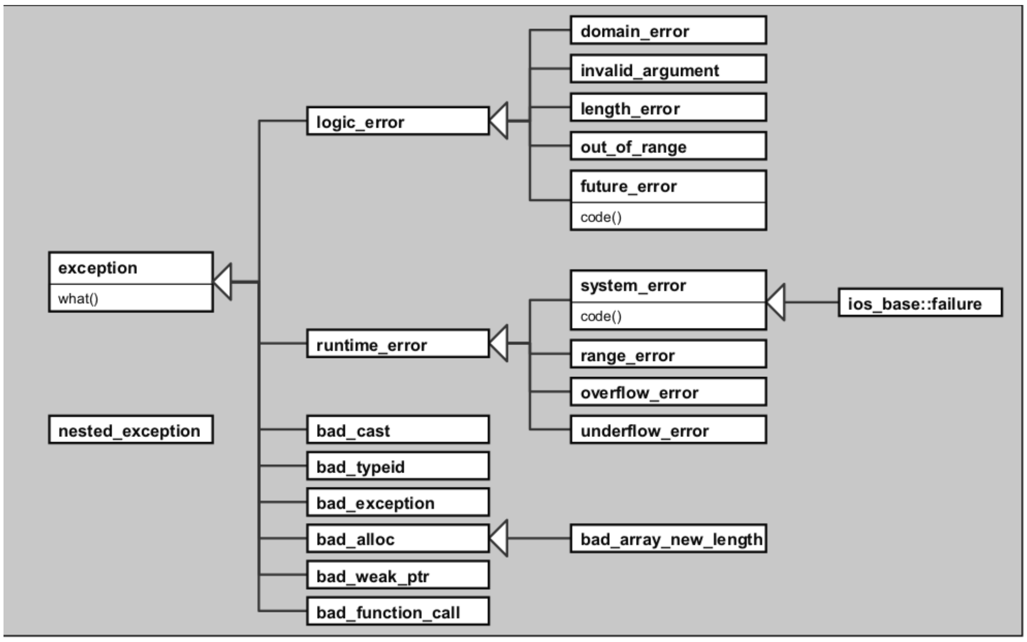
Error and Exception handling

★ Standard Exception Class

⇒ All exceptions thrown by the language or the library are derived from the base class exception, defined in <exception>.

⇒ This class is the root of several standard exception classes, which form a hierarchy.



⇒ These standard exception classes can be divided into three groups:

1. Language support
2. Logic errors
3. Runtime errors

1. Exception Classes for Language Support

⇒ Exceptions for language support are used by language features.

↳ So in a way they are part of the core language rather than the library.

⇒ These exceptions are thrown when the following operations fail:

1. bad_cast

   ↳ thrown by the dynamic_cast operator if a type conversion on a reference fails at runtime.

2. bad_typeid

   ↳ thrown by the typeid operator for runtime type identification.

   ↳ If the argument to typeid is zero or the null pointer, this exception gets thrown.

} defined in <typeinfo>

3. bad_exception

   ↳ used to handle unexpected exceptions.

} defined in <exception>

2. Exception Classes for Logic Errors

⇛ Exception classes for logic errors are usually derived from class logic_error.

⇛ Logic errors are errors that, at least in theory, could be avoided by the program.

⇛ The C++ standard library provides the following classes for logic errors:

1.invalid_argument
2. length_error
3. out_of_range
4. domain_error

} defined in <stdexcept>

5. future_error } defined in <future>

↳ This is used to report logical errors when using asynchronous system calls.

3. Exception Classes for Runtime Errors

⇛ Exceptions derived from runtime_error are provided to report events that are beyond the scope of a program and are not easily avoidable.

⇛ The C++ standard library provides the following classes for runtime errors:

1. range_error
2. overflow_error
   ↳ used to report an arithmetic overflow.
3. underflow_error
   ↳ used to report an arithmetic underflow.

} defined in <stdexcept>

4. system_error
   ↳ used to report errors caused by the underlying operating system. } defined in <system_error>
5. bad_alloc
   ↳ defined in <new>
   ↳ is thrown whenever the global operator new fails.
   ↳ bad_array_new_length, derived from bad_alloc, will be thrown by new if the size passed to new is less than zero or such that the size of the allocated object would exceed the implementation-defined limit
6. bad_weak_ptr
   ↳ defined in <memory>
   ↳ thrown whenever the creation of a weak pointer out of a shared pointer fails.
7. bad_function_call
   ↳ defined in <functional>
   ↳ thrown whenever a function wrapper object gets invoked but has no target.

⇛ In addition, for the I/O part of the library, a special exception class called ios_base::failure is provided in <ios>.

↳ thrown when a stream changes its state due to an error or end-of-file.

⇛ Any implementation of the standard library might also offer additional exception classes either as siblings or as derived classes.