Lesson 2: Exception Safety

Exception-safe code

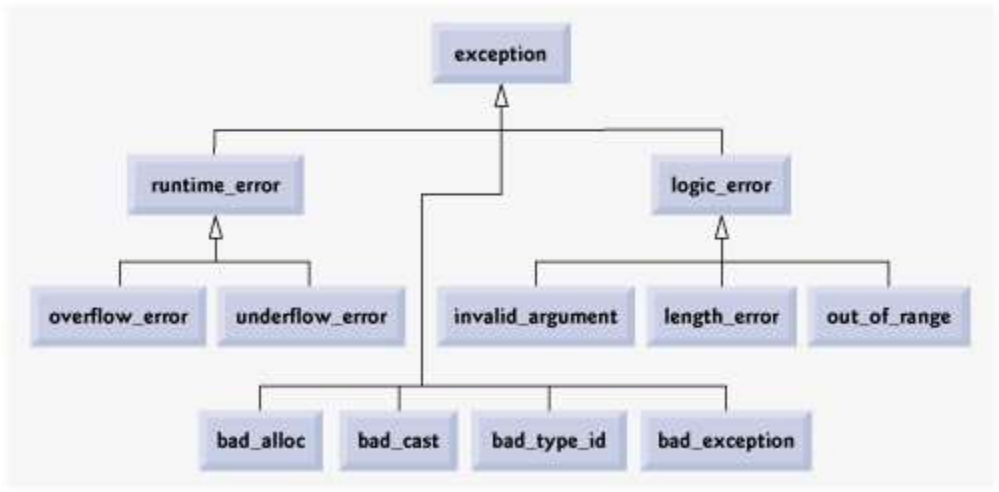
Exception-neutral code

2.1 Exception Safety Overview

- Definitions
- Our Task
- What We’ll Learn

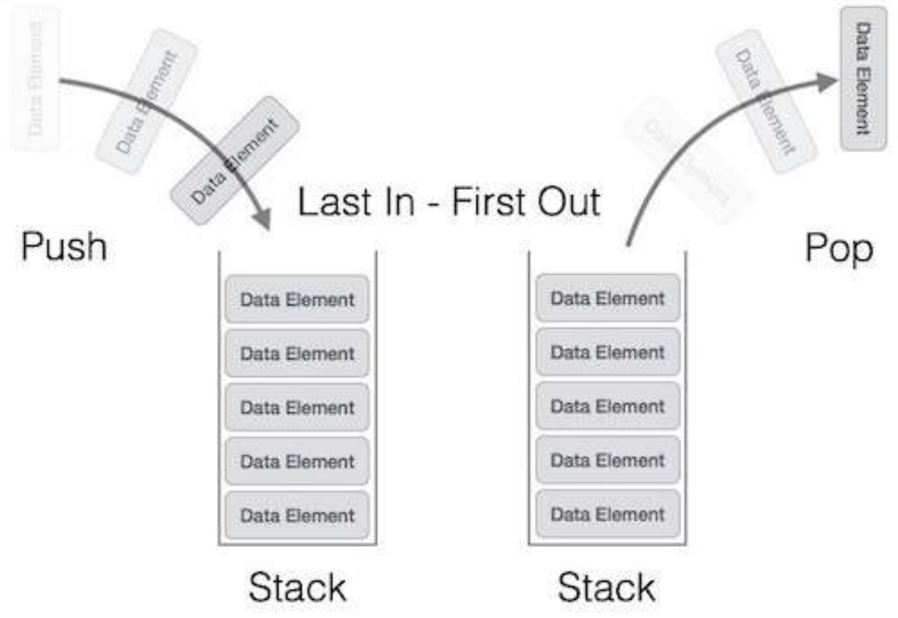
2.2 Exceptions Review

- Standard exceptions
- Standard exception usage
- Exception specifications



2.3 Stack Container

- Stack interface



2.4 Stack – Default Constructor

- Proposed default constructor
- Concerns
- Guidelines

2.5 Stack - Destructor

- Destructor implementation
- Destructors should never throw
- Guidelines

2.6 Stack – Helper “newCopy”

- newCopy implementation
- Exception neutral?
- Exception safe?

2.7 Stack – Copy Construction

- Copy constructor implementation
- Exception neutral?
- Exception safe?

2.8 Stack – Copy Assignment

- Copy assignment implementation
- Exception neutral?
- Exception safe?
- Guidelines

2.9 Stack - count

- count implementation
- Exception safe?
- Exception neutral?

2.10 Stack - push

- push implementation
- Exception safe?
- Exception neutral?
- Guidelines

2.11 Stack - pop

- pop implementation
- Exception safe?
- Exception neutral?
- A solution
- A better solution
- Guidelines

2.12 Stack – Full Code

- Full implementation of stack container

2.13 Stack – Analysis

- Exception safety guarantees
- Which guarantee does stack provide?
- What are requirements on type stored in stack?

2.14 Stack – Solution #2

- Goals
  - Reduce requirements on type stored in stack
  - Improve exception-safety guarantee

2.15 operators new() & delete()

- Operator new()
- Operator delete()
- Placement new
- Example

2.16 Stack – Solution #2 continued

- Approach
  - Encapsulate memory management
  - Eliminate useless constructions
- Implementation
- Benefits

2.17 Stack – Solution #3

- Approach (same as #2 except)
  - Uses composition instead of private inheritance
- Private inheritance or composition?
  - Both achieve same goal
  - Prefer composition unless edge cases private inheritance solves are present

2.18 Stack – Analysis of #2 & #3

- What requirements are put on T?
- Reusability improvements
- Guidelines

2.19 Destructors that Throw are Evil

- What happens if destructors can throw?
- Possible solution?
- Guidelines
- Conclusion
  - Destructors should never throw
  - Deallocators should never throw

2.20 Unit Test Frameworks

- Why 3rd party frameworks?
- UnitTest++
- Examples using UnitTest++
- Other unit testing frameworks