

Exceptions

- Exceptions are unusual events
- They can be errors, or just unusual
- Can be detected by hardware or software
- When an exception is raised/thrown, an appropriate exception handler is invoked.

Error Handling Without Exceptions

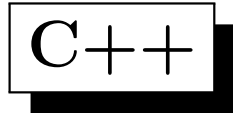
- Not all programming languages support exception handling
- One can use return values/parameters to indicate errors
- e.g. Unix system calls: very tricky to use
- Must explicitly check for errors, easy to ignore by mistake, laziness, etc.
- Makes it hard to read main algorithm
- Sometimes we need to exit multiple levels of loops and subprograms quickly (setjmp and longjmp)
- Can also pass error handlers as subprogram parameters

Advantages of Exception Handling

- Clearly separate error handling code from main algorithm
- Allows exception propagation: single handler can be used for exceptions raised in subprograms, exceptions can be handled at the appropriate place
- Encourages programmers to consider possible errors, and to prevent errors to be ignored

Design Issues

- Exceptions: are there predefined exceptions (implicitly raised)? Are there user-defined exceptions (explicitly raised)?
- How are different exceptions distinguished? Types? Any data attached to the exceptions?
- How are exception handlers bound to exceptions raised?
- How do exceptions propagate when there are no handlers?
- How to continue execution after exceptions are handled?
- Finalization support

The logo for C++ programming language, featuring the text "C++" in a white box with a black shadow.

- Use try-catch blocks
- Any type can be thrown, but preferably a subclass of `exception`
- No predefined exceptions
- `catch` comes with a number of forms for the type of exception to catch. Some require exact match, some require match of class or subclass.
- First handler that matches is executed
- `catch (...)` catches everything
- Can pass data in exception classes, can rethrow exception
- No `finalize`, continuation after the handler

Java

- Similar to C++
- Exceptions must be subclasses of `Throwable`
- Finalization support
- Some exceptions (anything other than `Error` and `RuntimeException`) must be checked: either handled or listed as a throwable exception from the function
- Some predefined exceptions are thrown implicitly (e.g. array out of bounds)

Finalization in Java

```
try {  
}  
catch (...) {  
}  
finally {  
}
```

- No exception thrown: finally clause executed after try block
- Exception thrown and caught: handler and then finally clause execute
- Exception thrown but not caught: finally clause execute, then exception propagate
- Other ways to exit try block (e.g. return, break, etc.): finally clause execute before exiting

Implementation

- Each time an exception is thrown, the local variables in the enclosing scope (e.g. try block) needs to be cleaned up/destroyed
- We can record on the stack all exception handlers defined.
- The stack is unwound until a matching exception handler is found. Variables/objects on the stack are cleaned up as the stack is unwound.
- This requires more memory usage on the stack.

Implementation

- We generally would like to avoid overhead if no exceptions are generated
- Each function has an exception frame, which contains a reference to an exception table describing how to process the exception
- The exception table usually describes for each exception handled, what action to take. The action usually passes control to a landing pad
- The landing pad is usually code to execute the catch statements (including selecting which catch to use)
- When an exception is thrown, a structure is created to hold the exception object. This object cannot be allocated on the stack. Why?

Implementation

- When a try block is entered, code to exit normally and code to landing pad are both inserted.
- When an exception is thrown, local variables are cleaned up. Then execution jumps to the landing pad which tests which catch to use.
- Some implementations do clean up at the landing pad before testing the catch clauses.
- Stack is unwound if needed
- The exception table can be stored elsewhere, even not loaded into memory unless needed.