Exceptions

• In C, standard library functions indicate errors by returning appropriate numeric values
• Retrieving useful information about the error is cumbersome
• So is recovering from errors that occur several function calls deep
• C++ allows throwing exceptions to facilitate error reporting and recovery
Throwing Exceptions

• Use the throw keyword:
  \texttt{throw object;}
• The thrown object can be any object that can be copied
• So far we have used an object of type \texttt{std::exception}
• Can create your own exception class that contains information about the error leading to the exception

Consequences

• \texttt{throw} statement diverts execution outside smallest enclosing \texttt{try} block
• If no \texttt{try} block inside current function, calling function is checked, and so on
• Variables in each scope de-allocated
• Dynamically allocated memory is not de-allocated, so don't lose track of it!
• Once an error is detected, perform any necessary cleanup before throwing
Catching Exceptions

- Once a `try` block is found, execution proceeds with following `catch` blocks.
- Argument list of each `catch` block is checked against thrown object.
- If a match is found, statements in `catch` block are executed.
- Thrown object is a local variable within the `catch` block where it is declared.
- `catch(...)` matches anything.

Exception Do's and Don'ts

- **DO** throw an exception within a constructor if anything goes wrong.
- **DO NOT** throw an exception within a destructor; might crash program!
- **DO NOT** let an exception go uncaught, or execution will terminate immediately.
- **DO NOT** use throw specifications like `type `fname`(args)` `throw(type);`
because they limit flexibility.
std::runtime_error

• Instead of throwing an object of type std::exception, can use the derived class std::runtime_error
• Declared in <stdexcept> header
• Construct with your own error message:
  using namespace std;
  runtime_error ex("Idiot!");
  throw ex;
• When caught: ex.what() is a const char array with the error message

Next Time

• Basics of Computer Organization
• What goes on under the hood
  – Function calls
  – Memory management
  – Linking and loading