Exception Handling in C++

OUTLINE

• What is an exception?
• Programming without exception handling.
• How to handle exceptions?
• Example
• Standard Library Exception
WHAT IS AN EXCEPTION?

• Exceptions
  – Indicate problems that occur during a program’s execution
  – Occur infrequently

PROGRAMMING WITHOUT EXCEPTION HANDLING

• Error handling without exceptions:
  – Requires a lot of tedious and error prone programming
  – All functions must agree on “error” return values
  – Complicated flow control
  – Must remember to release resources
• Constructors errors are especially problematic
  – No return value, so must use flags
  – Must remember to check flags

• Need flexible ways to deal with error conditions

EXCEPTION HANDLING

• Can resolve exceptions
  – Allow a program to continue executing
  OR
  – Notify the user of the problem and terminate the program in a controlled manner

• Makes programs robust and fault-tolerant
• A standard mechanism for processing errors
  – Especially important when working on a project with a large team of programmers

• C++ exception handling is much like Java’s

---

**try BLOCKS**

• Keyword `try` followed by braces (`{}`).

• Should enclose
  – Statements that might cause exceptions
catch HANDLERS

- Immediately follow a `try` block

- Contains code that should be executed whenever an exception of a specific type is caught

- Exception parameter enclosed in parentheses

```java
try {
    // code to try
}
catch (exceptionClass1 &name1) {
    // handle exceptions of exceptionClass1
}
catch (exceptionClass2 &name2) {
    // handle exceptions of exceptionClass2
}
catch (exceptionClass3 &name3) {
    // handle exceptions of exceptionClass3
}
```
throw HANDLE

• Serves two purposes
  – When used within the body of a function
    • used to throw an exception
  – When used in the header of a function
    • used to specify what types of exceptions the function may throw.

NOTE

• Catching an exception object by reference eliminates the overhead of copying the object that represents the thrown exception
throw - IN FUNCTION SIGNATURE

• A function can **throw** only exceptions of types in its specification

• Absence of exception specification indicates that the function can **throw** any exception

• An empty exception specification, **throw()**, indicates the function **cannot throw** any exceptions

---

**Standard Library exception**

- exception
  - runtime_error
    - overflow_error
    - underflow_error
  - logic_error
    - invalid_argument
    - length_error
    - out_of_range
      - bad_alloc
      - bad_cast
      - bad_type_id
      - bad_exception
Questions?