


Lecture 35: More C++

The graphic consists of a large blue square on the left containing a white plus sign. To its right are four smaller squares arranged in a 2x2 grid: orange (top-left), green (top-right), purple (bottom-left), and red (bottom-right).

+ Today



- Reading
  - Weiss Ch. 8.1-8.3
- Objectives
  - Exception handling in C++

The slide features a blue plus sign followed by the word 'Today'. To the right is a vertical bar with a thin green line on the left and a wider blue section on the right. The main content is a bulleted list with blue square markers.

## + Access Modifiers in C++

- In Java: public, protected, private, and “default”
- In C++: public, private, and friend
- Friends are allowed access to the private section of class

## + Access Modifiers in C++

```
class Node{  
    private:  
        int element;  
        Node *next;  
  
        // the constructor is private!  
        Node(int theElement, Node*n)  
            : element(theElement), next(n){ }  
  
        // the integer queue class is the only  
        // class that can create Node variables  
        friend class IntegerQueue;  
};
```

## + Access Modifiers in C++

```
class Node{
    private:
        int element;
        Node *next;

        // the constructor is private!
        Node(int theElement, Node*n)
            : element(theElement), next(n){ }

        // only the enqueue function from the
        // IntegerQueue class is a friend
        friend void IntegerQueue::enqueue(int x);
};
```

## + Life Before Exceptions

- C++ was designed to be efficient and fast
- As a result, exception handling is not as sophisticated as Java
- Exception handling before “exceptions”
  - `abort` – immediately terminates the program (not graceful)
  - `exit` – slightly better, calls destructor of static objects, pass an int
  - `errno` – a bit that can be set to indicate various errors
- Assertions
  - Use `assert` keyword defined in `cassert` library
  - Use `#define NDEBUG` to turn off assertions Should still do this!

## + C++ Exceptions

- Use try-catch block like Java
- In C++, you can throw a variable of any type

```
#include <iostream>
using namespace std;

int main() {
    try{
        throw -1;
    }
    catch(int e) {
        cout << "Exception occurred: " << e << endl;
    }
    return 0;
}
```

type thrown and type caught must match

## + C++ Exceptions

- Can have multiple catch statements
- Use “...” to catch an exception of any type

```
try{
    // code here
}
catch(int e) {
    cout << "Integer exception: " << e << endl;
}
catch(char e) {
    cout << "Char exception: " << e << endl;
}
catch(...){
    cout << "Default exception" << endl;
}
```

## + C++ Exceptions

- C++ standard library includes `exception` class from which exception objects can be created
- The `exception` class includes the `what` function
  - Returns a `string` description
- All exceptions thrown by C++ standard library are derived from `exception` class
- See C++ documentation
  - <http://www.cplusplus.com/reference/exception/exception/>

## + Throw Lists

- Indicate what exceptions are thrown by function using throw lists

```
double sqrt(int x) throw(int);
```
- Should be in the function prototype
- No list means throws anything (for backward compatibility)
- Empty list means throws nothing
- Throw lists are deprecated as of C++11 (but still supported)

## + Exceptions with File I/O

- File I/O is always a promising place for things to go wrong!
- failbit means logical error in operations
  - e.g. Tried to read a number and got a string
- badbit means read/write error
  - e.g. Can't open file
- The function `file.exceptions()` alerts compiler that if these bits are set an exception should be raised
- Good article on reading files in C++:
  - <http://gehrcke.de/2011/06/reading-files-in-c-using-ifstream-dealing-correctly-with-badbit-failbit-eofbit-and-perror/>