Practice with classes

William Hendrix

Lecture 17

Outline

- Short circuit evaluation
- Assertions
- Minilab

Short-circuit evaluation

- Some Boolean expressions can be evaluated without knowing both sides

```
true || any == true
false && any == false
```

- Modern compilers practice short-circuit evaluation
  - Will not evaluate the RHS of Boolean expression if outcome is predetermined

- Implications for pointers and error-checking
  - E.g., `if (ptr != NULL && *ptr > 0) //...`
    - If the pointer is `NULL`, it is never dereferenced
  - Can also be used to check for division by 0, etc.

Assertions

- C/C++ programming statements used for debugging and ensuring correctness of code
- Syntax

```
#include <assert.h>
assert(expression);
```

- If `expression` evaluates to `false`, program is terminated immediately
  - Prints out the expression that failed (e.g., `length >= 0`), file name, and line number
  - Can make finding problems easier, as it fails quickly and obviously
  - Intended for debugging, not end-users
- Assertions can be disabled by using

```
#define NDEBUG
```

  before including `assert.h`

Minilab

- Design and implement a MutableArray class that stores a array of integers. Your MutableArray should be able to:
  - Access and manipulate array members
    - *Hint:* Return an `int&`
  - Query its current size (occupancy, not allocated memory)
  - Resize the array
    - Fill new spaces with zeros if needed
  - Add an integer to the array
    - Should resize the array if full
    - *Optional*: allow syntax `arr[len] = -1;`
  - Delete an integer from the array given its index
  - Required constructors: default, copy, and a constructor that initializes a MutableArray of size *n* with all zeros

Tonight

Midterm is Feb. 11

Lab 4 is due Wednesday, Feb 12