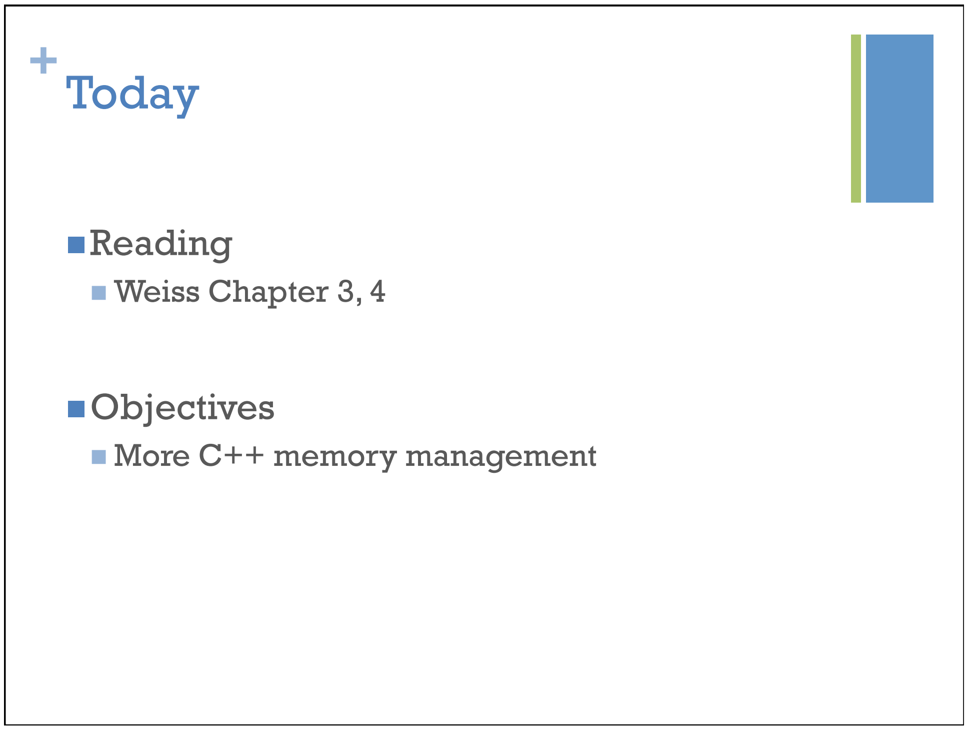
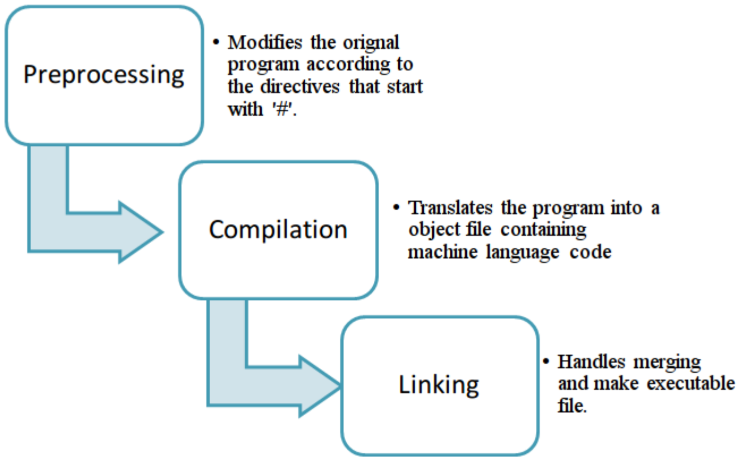
Lecture 31: More C++



Today

- Reading
  - Weiss Chapter 3, 4
- Objectives
  - More C++ memory management

Compiling and Running C++


Preprocessing
• Modifies the orignal program according to the directives that start with '#'.
Compilation
• Translates the program into a object file containing machine language code
Linking
• Handles merging and make executable file.

http://www.lifengadget.com/lifengadget/compiling-linking-cplusplus/

Compiling and Running C++

- `g++ -o executable_file first_src.cpp second_src.cpp`
- Use the –`Wall` flag to see all warnings
- Common errors
  - bool type: 0 is false and any non-zero value is considered true

```
    if(x = 0){…}   // will always be false
```

  - C++ does not check if variable initialized before use
  - C++ does not check to see if return value from function
  - C++ requires declaration before first use (use prototypes)

Constructors in C++

- Original constructor for IntCell:

```
IntCell::IntCell(int initValue) {
      value = initValue;
}
```

- Better to use an initializer list

```
IntCell::IntCell(int initValue) : value(initValue) {
      // no code needed now
}
```

Constructors in C++

- Initializer Lists
  - Comma-separated list of instance variables and values after a colon
  - Why? Because it's more efficient
  - Only initialize variables once rather than running the default constructor and then update in constructor

Memory Management in C++

- In C++, everything is a primitive
  - Objects are allocated from the stack
- The implications of this are,
  - No need to use the new keyword to create objects

```
vector<int> nums;
nums.push_back(5);
```

  - The equals operator (=) creates a copy

```
vector<int> nums2;
nums2 = nums;
```

Pointers in C++

- A pointer is a variable that stores the memory address of another entity (e.g. variable)
- Use the * operator to declare a pointer

```
int *ptr; // a pointer to an integer
          // Beware! ptr doesn't point to valid address
```

- Use the & operator to get the address of a variable

```
int x = 5;
int y = 7;
ptr = &x;
```

+ Pointers in C++

- Use the * operator to go from the pointer to the data being pointed to

```
int x = 5;
int y = 7;
int *ptr = &x;     // declare and initialize
cout << *ptr << endl; // prints 5
*ptr = 10;            // changes x to 10
```

- Weiss (Ch. 3) gives lots of examples of all that could go wrong!

+ Pointers in C++

```
int x, y;             // declare two ints
x = 10;
int *p, *q;           // p, q pointers to ints
p = new int(3);       // p points to value 3
*p = 47;              // p now points to 47
q = p;                // q points to value 47
*q = 23;              // both p,q point to 23
delete p;             // memory is recycled
*q = 17;              // ERROR! memory was recycled
p = NULL;             // p points to nowhere
p = &x;               // p holds address of x - points to x
cout << *p << endl;   // prints value of x
```