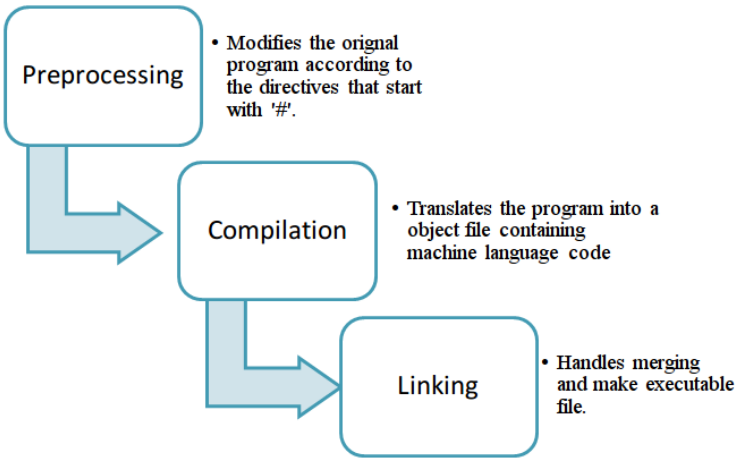
LECTURE 31: POINTERS

Today

- Reading
  - Weiss Chapter 3, 4
- Objectives
  - Compiling
  - Initializer lists
  - Pointers!

The C++ textbook is now on reserve at Honnold Mudd library

Compiling and Running C++



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

Compiling and Running C++

- To run the entire pipeline:

```
g++ -Wall -o executable_file first_src.cpp
second_src.cpp
```

- Use the –`Wall` flag to see all warnings

Initializer Lists

- Use in class constructor
- Comma-separated list of fields and initial values
- Don't forget the colon
- More efficient
- Initializes variables once rather than running default constructor and then updating

Back to Memory Management

- Java
  - Everything is an object, i.e. allocated from heap
  - Variables are reference variables
  - Garbage collector
- C++
  - Everything is a primitive, i.e. allocated from stack
  - Stack variables automatically de-allocated when scope exits
  - Allocate from heap using new keyword
  - You are the garbage collector!

Back to Memory Management

- Implications
  - Assignment (=) means copying
  - But now you have to think about *what* you're copying
  - Assignments happen more than you realize!

Pointers in C++

A pointer is a variable that stores the memory address of another entity of a given type

- To declare a pointer, use the * operator

```
int *ptr; // Uninitialized!
```

Pointers in C++: address operator

- The address operator & returns the address of a variable

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

string str = "hello";
string *str_ptr = &str;
```

Pointers in C++: dereference operator

- The dereference operator * goes from the pointer to the data being pointed to

```
int x = 5;
int y = 7;
int *ptr = &x;
cout << *ptr << endl;
*ptr = 10;
```

Pointers in C++

- *Do not dereference an uninitialized pointer!*
  - Always initialize pointers (to 0, NULL, or a known address)
- Precedent rules are important
  - *ptr++ will increment first then dereference
- Declaring multiple pointers on a line
  - int *ptr1, *ptr2; // each pointer needs a *

Pointers in C++

What are the values of the following?

```
int a = 5;
int *ptr = &a;

a. ptr
b. *ptr
c. ptr == a
d. ptr == &a
e. &ptr
f. *a
g. *&a
h. **&ptr
```