CSE 303 Concepts and Tools for Software Development

Richard C. Davis
UW CSE – 10/6/2006

Lecture 15 –
C++ Functions and Classes

Administravia

- Exams back on Wednesday
- Next HW5 due a week from Wednesday
  - It will be easier than HW4
- No office hours today
  - Instead, DRM lecture in Comm 126
  - Office hours Thursday 3:30-5:30

Tip of the Day

- Getting into C++ mode in Emacs
  - Doesn't enter this mode by default for `.h` files
  - use M-x c++-mode

Last Time

- Working with C++
- I/O & Strings
- Vectors
- Type casting

Today

- C++ Function details
- C++ Classes and Objects
  - Declaring
  - Using

C++ Functions

- Function Overloading Allowed (Like Java)
  - Functions with same name but different arguments
- Can supply default values for parameters
- Parameters passed by value (Like C)
  - New "reference" type.

```
void fun(int &i);
```

    - Ref must be a variable, not expression (or cast)
- Example in *CallByReference.cpp*

Remember C "Classes"?

- Class-like structure in C

```
struct MyPoint {
  // data
  int x; int y;
  // code
  int (*getX)(struct MyPoint *);
  void (*setX)(struct MyPoint *, int);
}
```

- C++: classes are like "structs" with methods

C++ Classes

- Very similar to Java
  - "public" and "private" specified for groups
  - Semicolon at end
  - this
    - *Pointer* to current object (not reference)
    - Can't use to invoke constructor
- Example in *IntCell.cpp*

Separation into .cpp and .h

- Why?
  - Avoid compiling multiple times
  - Faster
  - Avoids some compile problems
- How?
  - Class definition in .h
  - Method definitions in .cpp
- Example in *IntCell2.cpp and IntCell2.h*

Using Objects: Allocation

- Where are objects allocated?
  - Java: always on the heap
  - C++: stack or heap

Using Objects: Stack Allocation

```
IntCell m1;       :Calls Default (0-arg) constructor
IntCell m2 = 37;  :Calls 1-arg constructor
IntCell m3(55);   :Calls 1-arg constructor
IntCell m3(m2);   :Calls "Copy" constructor
m1 = m3;          :Calls "operator="
m1 = 42;          :Calls 1-arg constr. AND operator=
```

- Lots of Implicit copying
  - Hard to avoid in general
  - Book shows you ways to avoid
- Careful! Objects can go out of scope

Using Objects: Heap Allocation

```
IntCell *p = new IntCell(54);
delete p;
```

- `new` replaces `malloc`
- `delete` replaces `free`
  - Calls a *destructor*
- Each works with *pointers*

Reading

- C++ for Java Programmers
  - Chapter 3: Pointers and Reference Variables
    - Skip 3.2, 3.3.5
  - Chapter 4: Classes
    - Skip 4.6-4.8, 4.10, 4.12-4.14

Next Time

- More on C++ classes
- Inheritance