CSE 303 Concepts and Tools for Software Development

Richard C. Davis
UW CSE – 11/3/2006

Lecture 14 – Introduction to C++

Administravia

- Midterm Exam
  - Too Long
  - Mostly due to problem 4(b)
    - It way too much to write out
    - Some haven't had data structures yet
  - We'll try to correct the situation
    - Try to be fair to those who invested time in 4(b)
    - Try to get back to you on Wednesday

Where We Are

- C language
  - Good for low-level programming work
    - Operating Systems
    - Embedded systems
  - Unsafe
  - Not Object-oriented
  - Primitive Standard library
    - Very few built-in types (arrays, structs)
    - Awkward strings and file I/O

Introducing C++

	Java	C	C++
Ideal for what tasks?	High-level	Low-level	Low-level
Safety	Safe	Unsafe	Unsafe
Object-Oriented	Yes	No	Yes
Standard Library	Large	Small	Medium

Today

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

C++ In a Nutshell

- Contains all of C
  - Almost backward-compatible (see *CHello.cpp*)
  - Some new keywords (not many!)
  - Compiled/Linked differently
- Like Java, but…
  - Distinguishes objects and pointers-to-objects
  - Reference Parameters
  - More OO features
    - Multiple Inheritence, non-virtual functions…
  - Templates, Operator Overloading, etc…

Working with C++

- File names end in "`.cpp`"
  - But other conventions abound
  - `.cxx`, `.C`, `.cc`
- Compiling
  - Use `g++` instead of `gcc`
  - `g++ -g -Wall -o Hello Hello.cpp`
- Managing Files
  - Use headers and source files as in C
  - No "one-class-per-file" restriction as in Java

C++ Namespaces

- Allows grouping of function/class names
  - Not found in C (all names global!)
  - Similar to Java Packages
  - C++ Standard Library is in namespace "`std`"
- Compare *Hello.java* with *Hello.cpp*

C++ Strings, I/O

- Strings
  - Like in Java, but mutable
- I/O
  - New objects for stdin, stdout, stderr
    - `cin`, `cout`, `cerr`
  - `<<` and `>>` have automatic type conversion
- We'll go into more depth later
- Example in *StringsIO.cpp* & *Vector.cpp*

C++ Vectors

- Vectors
  - An object that works like an array
    - Like Java ArrayList
  - Parameterized by type (using templates)
  - Can grow and shrink dynamically
- We'll go into more depth later
- Example in *Vector.cpp*

C++ Type Casting

- New type casting syntax (primitive types)
  - C way: `int i = (int)d;`
    - C way still works in C++
  - New C++ way #1: `int i = int(d);`
    - Can't use for type names like `unsigned int`
  - New C++ way #2:
    - `int i = static_cast<int>(d);`
    - Helpful, because it's more explicit
    - We'll cover `dynamic_cast` later.

Summary

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

Reading

- C++ for Java Programmers
  - Chapter 1: Basic Types and Control Structs.
  - Chapter 2: Strings and Parameter Passing
- Remember, it doesn't know you know C!

Next Time

- C++ Parameter Passing
- Debuggers