

Introduction to C

CS 113: Introduction to C

Instructor: Saikat Guha

Cornell University

Spring 2007, Lecture 1

Administrivia

- ▶ Instructor: Saikat Guha, 331 Upson (5-1008)
- ▶ Email: saikat@cs.cornell.edu
- ▶ Lectures: MWF 12:20–1:10p, UP 205
- ▶ Lab: PH 318 (when announced)
- ▶ Office Hours: Tuesday 4:30–5:30p (or by appointment)
- ▶ <http://courses.cs.cornell.edu/cs113/>

Pre-requisites

- ▶ Basic programming knowledge (variables, functions, loops)
- ▶ Lots of composure
 - ▶ Your programs won't compile
 - ▶ Your programs won't run
 - ▶ Your programs will crash
 - ▶ You'll have no idea what happened
 - ▶ ... but at least it'll happen fast!

History of C

- ▶ Writing code in an assembler gets real old real fast
 - ▶ Really low level (no loops, functions, if-then-else)
 - ▶ Not portable (different for each architecture)
- ▶ BCPL (by Martin Richards): Grandparent of C
 - ▶ Close to the machine
 - ▶ Procedures, Expressions, Statements, Pointers, ...
- ▶ B (by Ken Thompson): Parent of C
 - ▶ Simplified BCPL
 - ▶ Some types (int, char)

History of C

- ▶ C (by Kernighan and Ritchie)
 - ▶ Much faster than B
 - ▶ Arrays, Structures, more types
- ▶ Standardization
- ▶ Portability enhanced
- ▶ Parent of Objective C, Concurrent C, C*, C++

When to use C

- ▶ Working close to hardware
 - ▶ Operating System
 - ▶ Device Drivers
- ▶ Need to violate type-safety
 - ▶ Pack and unpack bytes
 - ▶ Inline assembly
- ▶ Cannot tolerate overheads
 - ▶ No garbage collector
 - ▶ No array bounds check
 - ▶ No memory initialization
 - ▶ No exceptions

When not to use C

Use JAVA or C# for ...

- ▶ Quick prototype
- ▶ Compile-once Run-Everywhere
- ▶ Reliability is critical, but performance is not
 - ▶ C can be very reliable, but requires tremendous programmer discipline