

C (&c., II)

UMass CS 201
February 25, 2010
Ben Ransford <ransford@cs>

1

Last time

- History of C: 1969–present
- Low- and high-level programming langs.
- Interpreted and compiled langs.
- Elements of the compiler toolchain

2

This time

- Some real C programs
- Reasons to feel strongly about C
- Goal: demystify stuff you'll see while reading C code
- Assumption: familiarity with Java
 - Will skip *lots* of Chapter 11

3

But first

- Pop quiz I hinted strongly about last time
 - Elements of toolchain
 - One diff. btwn. compiler, interpreter
- **5 minutes**

4

Exercise: cookie compilation

- Adapted from *Chocolate Chip Cookies*, New York Times, July 9, 2008

5

Hello world

- C & Java versions of the old standard “Hello world” program
- `int main()` vs. `class Foo { void main() }`

6

Procedural vs. OOP

- C is *procedural*
- Java is *object-oriented*
- What’s the difference?
 - `foo.bar()` vs. `bar(foo)`

7

Making things in C

- Simple data types, like in Java
 - `int`, `char`, `long`, `float`, `double`, ...
- No `String` type; must use `char[]`
 - `char[]` “strings” NUL-terminated

8

Making things in C

- Compound data structures (structs)
- `struct foo { int a; int b; };`
 - Access `a` like `foo.a`

9

Making things do things

- `foo.bar()` vs. `bar(foo)` [again]

10

Header files & libraries

- Standard headers include `stdio.h` (for I/O) and `math.h`
- Libraries usually ship with linkable libraries and header files that describe them
- Dynamic linking vs. static linking

11

Memory management

- No “new Foo” -- no constructors
- `malloc` to allocate memory
- stack vs. heap allocation

12

Memory management

- No destructors either
- *free* to free memory

13

Pointers

- Don't worry, actually not that scary
- Pointer type: struct foo *x
- Referencing: &x
- Dereferencing: *x

14

When are pointers useful?

- Imagine a function that takes a large struct as input
- Pass-by-value, pass-by-reference

15

#define, preprocessing

- Preprocessor is the first compiler phase
- Preprocessor is what handles #include
- #define and conditionals

16

Pointers vs. arrays

- Sort of the same thing
- `char a[4]`
- 'a' gives the location of `a[0]`

17

n-th degree pointers

- `char** argv`

18

Debugging with printf

- `printf` is like Java's `System.out.println`
- Format strings:
 - `%d` = decimal
 - `%f` = float
 - `%s` = string (NUL-terminated)

19