Administrivia

- Reminder: Homework 3 due today (accepted without penalty through end of the day tomorrow).

- Note revision to reading for 10/10.

Slide 1

Pointers in C

- C, in contrast to Python and Scala, makes an explicit distinction between things and pointers-to-things. As I understand things, in Python and Scala variables are pointers/references to objects, and you deal with them fairly abstractly. In C, you can have variables that are "things" (integers, floating-point numbers, etc.) and variables that are "pointers to things" (in some ways more like variables in Python and Scala, but very low-level and with fewer safety checks).

- That is, in C, pointers are basically just memory addresses, though declared to point to variables (or data) of a particular type. Example:

```
int * pointer_to_int;
double * pointer_to_double;
```

Slide 2

Pointers in C — Operators

- `&` gets a pointer to something in memory. So for example you could write

  ```
  int x;
  int * x_ptr = &x;
  ```

- `*` "dereferences" a pointer. So for example you could change `x` above by writing

  ```
  *x_ptr = 10;
  ```

- You can also perform arithmetic on pointers (e.g., `++x_ptr`) — something not allowed in languages more concerned with safety.

Parameter Passing in C — Review

- In C, all function parameters are passed "by value" — which means that the value provided by the caller is copied to a local storage area in the called function. The called function can change its copy, but changes aren't passed back to the caller.

- An apparent exception is arrays — no copying is done, and if you pass an array to a function the function can change its contents (as we did in the sort program). Why "apparent exception"? because really what's being passed to the function is not the array but a pointer! so the copying produces a second pointer to the same actual data.

- This is at least simple and consistent, but has annoying limitations . . .

Pass By Reference (Sort Of)

Slide 5

- A significant potential limitation on functions is that a function can only return a single value. Pointers provide a way to get around this restriction: By passing a pointer to something, rather than the thing itself, we can in effect have a function return multiple things.

- To make this work, typically you declare the function's parameters as pointers, and pass addresses of variables rather than variables.

- (The "sort of" of the title means that this isn't true pass by reference, as it exists in some other languages such as C++, but it can be used to more or less get the same effect.)

Pointers — Examples

- (Simple examples.)

- Calls to scanf should now make sense — the function is supposed to store values into variable(s), and with pass-by-value we can't do that unless we pass a pointer.

Slide 6

Pointers Versus Arrays

- In C, pointers and arrays are in some sense(s) equivalent — not identical, but in many contexts interchangeable.

- This is reflected in the `man` pages for many functions (e.g., `printf` — strings are arrays of characters, as we will discuss next time). It also means that when you pass an array to a function, what you're actually passing is a pointer — so the array is not copied.

Slide 7

Minute Essay

- Standard question — what was interesting about Homework 3? difficult?

Slide 8