Slide 1

Administrivia

- Reminder: Homework 2 was due last week. Only two people have turned anything in. Problems?

Slide 2

C Basics and Arrays — Review/Recap

- C has the same control structures as most other procedural programming languages (assignment, if/then/else, loops), which should all be familiar except for syntax. Explicit variable declarations will be new to Python programmers.
- C arrays should also be a familiar idea, but they're low-level constructs, without some of the nice features of lists/arrays in other languages.

Slide 3

Strings in C

- Many languages have nice ways of working with text (character strings). What C provides is — no surprise — somewhat primitive.
- In C, strings are arrays of `char`s, with the convention that the actual text of interest is followed by a null character (8-bit zero, represented in code as `'\0'`.
- You can operate on individual characters however you see fit (accessing them as elements of the array). There are also standard library functions for some common operations (e.g., `strcmp` to compare two strings — similar to `compareTo` in Java).
- A significant source of potential trouble — most functions assume that strings are properly terminated, and (worse) many have no safety check to make sure you don't overflow a destination array.

Slide 4

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;
```

Pointers in C — Operators

Slide 5

- & 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

Slide 6

- 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 . . .

Slide 7

Pass By Reference (Sort Of)

- 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.

Slide 8

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`). 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.

Minute Essay

- For various reasons there is no required textbook for this course, only an online tutorial. Does it provide enough information, or would you rather have been asked to purchase a textbook?