Slide 1

Administrivia

- Reminder: Homework 4 due today.

Slide 2

Pointers, Characters, and Strings in C — Review

- Pointers are, roughly speaking, memory addresses. Useful in many contexts. If they don’t make sense to you yet, be patient?
- Characters in C are small integers, big enough to represent ASCII characters but possibly not much more.
- Strings in C are arrays of characters, ending with “null character” (`\0`). Can operate on them as arrays or using pointers.

Characters and Strings in C — Library Functions

- C's standard library is pretty limited but does contain some useful functions for operating on character/string data.
- Some useful ones are `isdigit` etc. for characters and `strlen`, `strcmp`, and `strchr` for strings.

Slide 3

Strings in C — Pitfalls

- Most functions assume that strings are properly terminated. (What do you think happens if they're not?)
- Many functions that store into a string have no way to check that it's big enough.

  So getting text input from standard input *safely* is surprisingly difficult! `scanf` can be made to check, but not (in my opinion) nicely, and it stops on whitespace anyway. `gets` gets a full line, but notice what `gcc` says when you use it. `fgets` is maybe better but has its limitations too.

Slide 4

Slide 5

Pointers and Strings in C — Example

- An interesting(?) example might be a function that determines whether a string is a palindrome, defining “palindrome” such that non-letter characters don’t matter nor does case.

Slide 6

Another Way to Get Input — Command-Line Arguments

- Now that we know about arrays, pointers, and text strings, we can talk about command-line arguments. What are they? text that comes after the name of the program on the command line (e.g., when you write `gcc -Wall myprogram.c`, there are are two command-line arguments), possibly modified by the shell (e.g., for filename wildcards).
- Most programming languages provide a way to access this text, often via some sort of argument to the main function/method.

Slide 7

Command-Line Arguments in C

- In C, command-line arguments are passed to `main` as an array of text strings. So if you define `main` as

  ```
  int main(int argc, char * argv[]) { .... }
  ```

  `argc` is the number of arguments, plus one, and `argv` is an array of strings containing the arguments.

  ("Plus one"? yes, `argv[0]` is something system-dependent, often the path for the program's executable.)

- What if you want to get numeric input? you must convert string pointed to by `argv[i]` to the type you want (more next time).

Slide 8

Command-Line Arguments and UNIX Shells

- Be aware that most UNIX shells do some preliminary parsing and conversion of what you type — e.g., splitting it up into "words", expanding wildcards, etc., etc.
- If you don't want that — enclose in quotation marks or use escape character (backslash).

Minute Essay

- Anything noteworthy about Homework 4 (interesting, difficult, etc.)?
- How has the pace/workload of this class been so far? do you feel like it's about right for a one-unit course (which is supposed to represent about three hours of work per week, in and out of class)?