

Administrivia

- Homework 8 on the Web. Due a week from today. Short; there will be another assignment coming Wednesday, to be due in about a week.
- Next quiz next Monday. Likely topics include arrays and strings.

Slide 1

Text Data in C

- Single characters represented by type `char`. Character constants use single quotes. Can include “escape characters”, e.g., `'\n'`.
- Strings represented as arrays of `char`, of whatever size, with a character `'\0'` marking the end. “String” constants use double quotes. (“Aha!”? so now the first parameter to `printf` and `scanf` makes almost complete sense? except for `const`? next slide.)

Slide 2

Sidebar: `const` in C

- You'll notice that some parameters of library functions are declared `const`? In C, this keyword means "does not change".
- For parameters, means the function doesn't change it. (I haven't been using it in examples, but arguably I should.)
- For variables, means the value once assigned doesn't change. This might be a nicer alternative to `#define` for giving constant values a name.

Slide 3

Working With Text Strings in C

- Once you have a string, what can you do with it? can process it either as an array (using indices) or using pointers. Pointer arithmetic can be a help. (Review example.)

Slide 4

Working With Text Strings in C, Continued

- Surprisingly(?), getting string input *safely* is tricky. I recommend `fgets()`, when you can't just supply the string as a command-line argument.
- Perhaps surprisingly, normal(?) assignment and relational operators don't for the most part work, but there are library functions:

Slide 5

`strcpy` to copy (use instead of assignment).

`strcmp` to compare (use instead of relational operators).

Many other library functions ...

Working With Text Strings in C, Continued

- Significant problem in working with strings — no natural maximum size, so must decide how big to make the array of characters used to hold one — and then be sure you don't try to put in too many characters.
- Some library functions let you say how big the array is; some don't. *Always* be as careful as you can when working with strings; trying to store a string in an array not big enough is a source of "buffer overflows", which can lead to program crashes and more subtle problems, including security risks.

Slide 6

Command-Line Arguments in C — Review

Slide 7

- 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 — represented as pointers to their first elements(!).
- Reference individual arguments via `argv[0]`, `argv[1]`, etc.
- This should make more sense now that we know about arrays, and (more) about pointers? and we can write a general “echo arguments” program.

Converting Text Strings to Numeric Types

Slide 8

- You know about `scanf` (and `fscanf`) for converting text input to numeric types. But what if you have a text string (e.g., a command-line argument) and want to extract from it a command-line argument? You could use `sscanf`, or ...
- Functions `strtol` and `strtod` can help. (`atoi` and `atof` can also be used but do not provide any kind of error checking.)
Usage example:

```
char *endptr;  
long n = strtol(argv[1], &endptr, 10);  
if (*endptr != '\0') /* error */
```
- (Example — program to echo command-line arguments, revisited.)

2D Arrays Revisited — Conway's Game of Life

Slide 9

- Many years ago mathematician John Conway came up with a simple “game” in which you define a 2D grid of cells, each containing an organism, or not, and rules for updating the grid. (You may have seen this? if not, Wikipedia article seems good.)
- What would a C program to “play” this game look like? (Next time. Next homework will involve this.)

Minute Essay

Slide 10

- Have you seen Conway's Game of Life before?