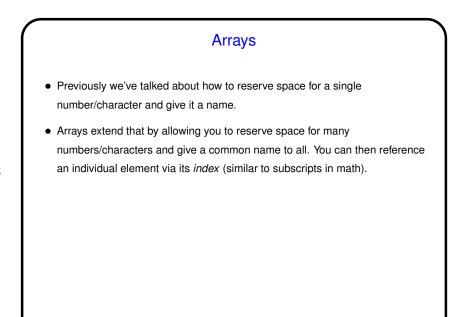


Why Arrays?, Continued
You could have a variable for how many A's, one for how many B's, etc., and a huge switch construct. But how ugly ...
What seems to be needed is something similar to subscripted variables in math — an array.

Slide 4

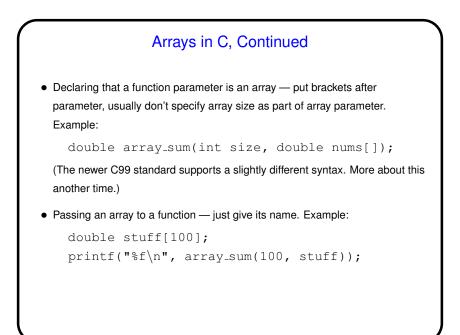


Slide 5

Slide 6

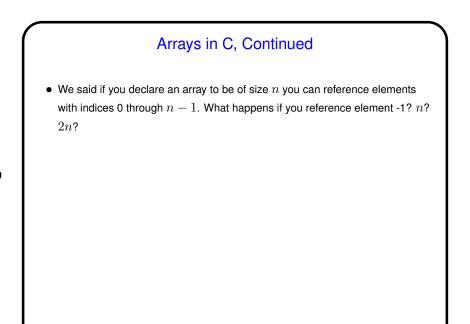
	Arrays in C
	Declaring an array — give its type, name, and how many elements. Examples:
	<pre>int nums[10]; double stuff[N];</pre>
	The second example assumes ${\rm N}$ is declared and given a value previously. I ld C, it had to be a constant. In newer C, it can be a variable.)
f	Referencing an array element — give the array name and an index (ranging rom 0 to array size minus 1). Index can be a constant or a variable. Then us is you would any other variable. Examples:
	<pre>nums[0] = 20; printf("%d\n", nums[0]);</pre>
(Notice that the second example passes an array element to a function. AOK

Slide 7



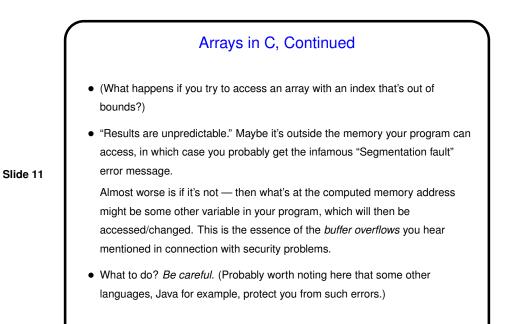
Arrays in C, Continued Using an array parameter within the function — as in previous examples. A difference between array parameters and other parameters — array elements can be changed. If your function isn't supposed to change the array, declare the parameter const, e.g. double array_sum(int size, const double nums[]); (Helps people using your function understand its effects, allows compiler to enforce that no changes are made.)

Slide 8



Slide 9

Arrays in C, Continued We said if you declare an array to be of size *n* you can reference elements with indices 0 through *n* – 1. What happens if you reference element -1? *n*? 2*n*? Well, the compiler won't complain. At runtime, the computer will happily compute a memory address based on the starting point of the array and the index. If the index is "in range", all is well. If it's not (i.e., it's "out of bounds)...



Example(s) • (Next time — instead look at second Homework 5 problem.) Slide 12

