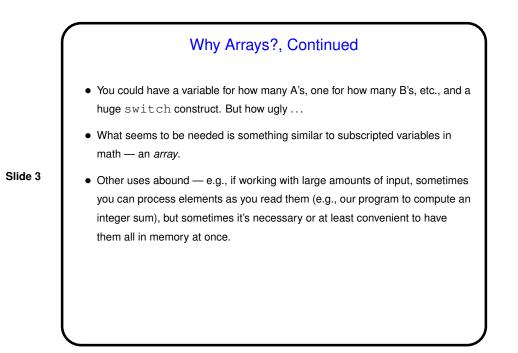
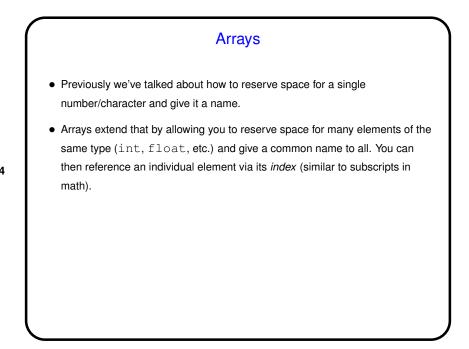


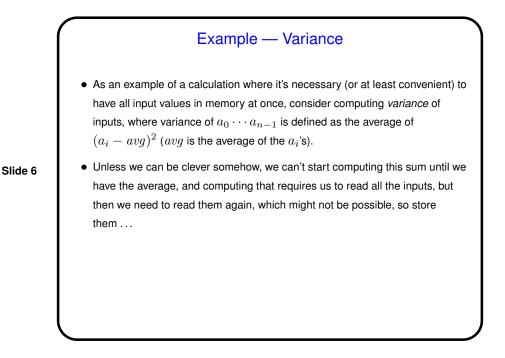
Why Arrays? • Suppose you wanted to write a program to count how many times each letter occurs in the program's input. What would you do? Is there an obvious way to solve this with what we've discussed so far?

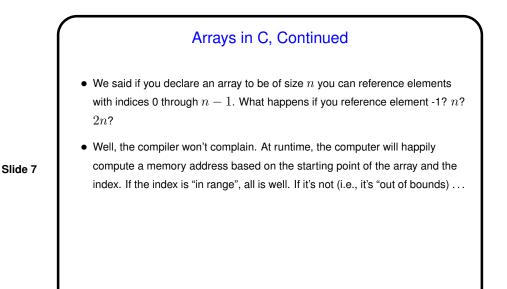
Slide 1





```
Arrays in C
• Declaring an array — give its type, name, and how many elements.
Examples:
    int nums[10];
    double stuff[N];
    (The second example assumes N is declared and given a value previously. In
    old C, it had to be a constant. In newer C, it can be a variable.)
• Referencing an array element — give the array name and an index (ranging
    from 0 to array size minus 1). Index can be a constant or a variable. Then use
    as you would any other variable. Examples:
    nums[0] = 20;
    printf("%d\n", nums[0]);
    (Notice that the second example passes an array element to a function. AOK!)
```





Arrays in C, Continued
(What happens if you try to access an array with an index that's out of bounds?)
"Results are unpredictable." Maybe it's outside the memory your program can access, in which case you probably get the infamous "Segmentation fault" error message.
Almost worse is if it's not — then what's at the computed memory address might be some other variable in your program, which will then be accessed/changed. (This is the essence of the *buffer overflows* you may hear mentioned in connection with security problems.)
What to do? *Be careful*. (Probably worth noting here that many more-recent languages, for example Java and Python, protect you from such errors by "throwing an exception", which by default crashes your program, but with information about what went wrong.)

