

User-Defined Types in C — typedef

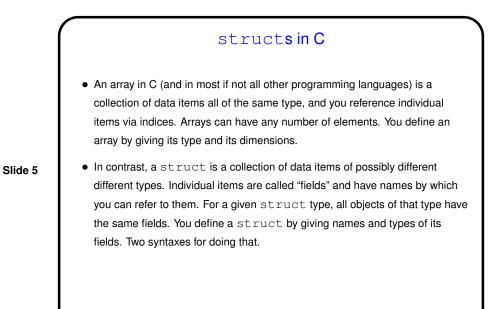
• typedef just provides a way to give a new name to an existing type, e.g.:

```
typedef charptr char *;
```

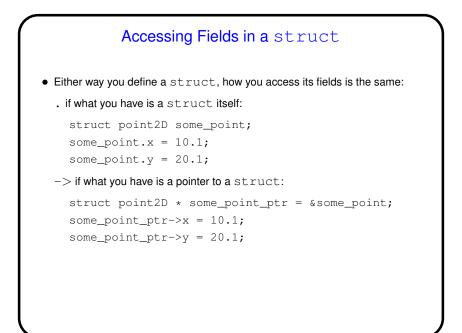
• This can make your code more readable, or allow you to isolate things that might be different on different platforms (e.g., whether to use float or double in some application) in a single place.

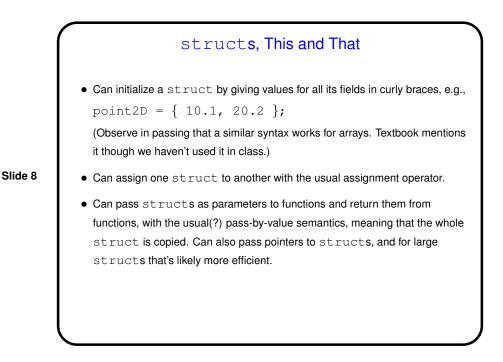
Slide 3

User-Defined Types in C — struct It's also useful to have ways of representing more-complex "types". Simple examples include rational numbers (with integer numerator and denominator) and points in 2D or 3D space (with two or three coordinates). More-complex examples ... Well, it's almost "the sky's the limit", but as one example think about what might be involved in writing a program with a GUI: It would seem to make sense to have some way of representing "windows" and "buttons" and so forth. "Object-oriented" languages provide a nice way to do this. C doesn't provide a really nice way, but it does provide a way, via structs.



Defining a struct • One way to define uses typedef: typedef struct { double x; double y; } point2D; point2D some_point; • Another way doesn't: struct point2D { double x; double y; }; struct point2D some_point;





Slide 9



- All the programs we've written have been single files. Fine for small programs but seems like it might be unwieldy for large ones, no?
- So usually code for large programs is split into multiple files, which can be separately compiled and then "linked" together to form an executable. This also allows reuse of the same function(s) in multiple programs. We'll do examples of both, later.
- "Library" functions build on this idea: Someone writes code for them, typically splitting it into a . h file with just declarations of functions (e.g., stdio.h) and a . c file with code for functions. The code is compiled and turned into something that can be linked into user programs. What's distributed might be just those . h files and the compiled code.

structs - Example(s)

- As an example we might define a struct to represent three-dimensional vectors and write some functions for it ...
- We might even usefully put the struct definition in a . h file, so we could use it in more than one program (with #include "3d-vec.h").

