Objectives - Thu 2/28/2019

- ☐ C syntax
  - union
  - enum
  - compound initializers
- ☐ C99: compound literals
- ☐ C11: anonymous types

Collections

arrays (incl. strings)
struct objects
linked lists
BST

$\left.\begin{array}{l}\text{arrays (incl. strings)}\\ \text{struct objects}\\ \text{linked lists}\\ \text{BST}\end{array}\right\}$ one name <u>many</u> values

---

Not really collections

$\left.\begin{array}{l}\text{union}\\ \text{enum}\end{array}\right\}$ one name <u>one</u> value enum : more readable alternative to int constants for state or mode or type.

Union: just like a struct but only has one of the values, not all of them.

```
struct Place {
    struct Point {
        int x;
        int y
    } location;
    char* title;
};
```

optional always

optional in C11

↓

without it, you get an anonymous struct type

Current architectures
The Intel x86 and also AMD64 / x86-64 series of processors use the little-endian format, and for this reason, it is also known in the industry as the "Intel convention". Recently designed instruction set architectures typically follow this convention, either allowing only little-endian mode (e.g. RISC-V, Nios II, Andes Technology NDS32, or Qualcomm Hexagon), or running mostly little-endian software on a bi-endian architecture (e.g. ARM Aarch64, C-Sky).

Some big-endian architectures that remain popular include mostly older examples like the IBM z/Architecture, Freescale ColdFire (which is Motorola 68000 series-based) and Atmel AVR32, but also the more recent OpenRISC. The IBM AIX and Oracle Solaris operating systems on bi-endian Power ISA and SPARC run in big-endian mode, while Linux on Power has moved to little-endian mode for new distributions.

As a consequence of its original implementation on the Intel 8080 platform, the operating system-independent FAT file system is defined to use little-endian byte ordering, even on platforms using other endiannesses natively.

Credit: https://en.wikipedia.org/wiki/Endianness