C - Week 8 - Notes

L1 Intro to Struct in C

- Basic C Data types
  int, char, float → fixed size
  char[] (strings), array → var. size

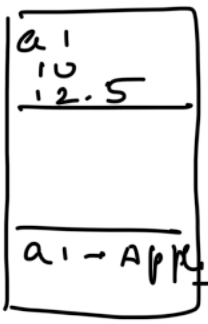
- Eg. fruit seller

name	quantity	unit price
Apple	10	12.5
⋮	⋮	⋮

- Arrays related by index

```
char* name[10];   int q[10];   float p[10];
0  "Apple"        10           12.5
```

- struct Fruit {
    char* name;
    int q;
    float p;
  }



→ Null terminated strings

- struct Fruit {
    char* name;
    int q;
    → Another struct (nesting)
    → pointer to same type of struct (recursion)
  }

- size of struct → gen. - sum of indi sizes

- Alignment - within struct, padding (compiler adds / removes)
- do not assume sizeof struct = sum of sizes.
- a = b; copies, permitted, copies all fields, doesn't perform deep copy.
- a == b; comparison is undefined. needs custom element-wise comparison

L2 - Egs. of Struct 1

Fruit

```
- struct {

  };

  struct Fruit apples;
  apples.name = "Apple";
  apples.q = 10;
  apples.p = 12.5;
- sizeof (Fruit);
  sizeof (fruit.name);
```

L3 - Egs. of Struct 2

```
struct Fruit apples = (struct Fruit)
                      {"Apples", 10, 12.5}
```

If it is a pointer →

(apples → name, apples ⇒ quantity)
- difference btw. shallow vs deep copy.

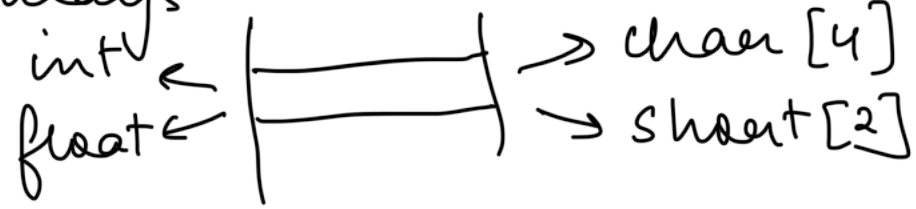
L4 - Typedef in C

- syntax elements to define custom data types. Reusable data types.
- Helps in cleaner, more readable code.

```
struct D1 {
    int x;
}
typedef struct D1 D1;
D1 x = (D1){10};
```

L5 - Union in C

- variable - data stored in memory
- some data can be interpreted in diff. ways




```
union foo1 {
    int x;
    int y;
```

it can in one place, have diff.

L6 Enum in C

- readability
- alternate representations for data (int.), not enforced by compiler or runtime, not printable, helps with debugging
- biggest usecase is in switch case statements.