

# ENUMS/UNIONS

# RECALL: STRUCTS

- Way to group multiple data of different types into a single thing
- Kind of like an object (not really, but as close as you get in C)

```
struct Student {  
    char *name;  
    double *scores;  
    int grad_year;  
}
```

# RECALL: TYPEDEF

- If we define like the previous page, create by
  - `struct Student st1`
  - `struct Student *st1` - then malloc
- To prevent needing to say `struct`, use `typedef`
  - Can be it's own line `typedef struct Student Student;`
  - Can lump into struct definition

```
typedef struct Student {  
    // same stuff as previous slide  
} Student;
```



# ENUMS

- Basically a way of storing a named element
- Increases clarity of the program, but doesn't actually add any functionality
- Each enum element is simply a constant stored as an integer underneath
- Default stores as 0 ... (N-1) in order listed
- Can (if you want to):
  - Define specific integer value
  - Have 2 elements with same integer value

# ENUMS - EXAMPLE

```
enum grade {A, B, C, D, F};  
enum grade g1;  
g1 = B;
```

# ENUMS - TYPEDEF

- Like with struct, can typedef to avoid having to use  
`enum day_of_week day;`
- Can make it a separate line or encapsulate into  
enum definition

```
typedef enum grade {A, B, C, D, F} grade;
```

## ENUMS (CONT.)

- Enums can be a member of struct
- Can also have an array of enum

```
typedef struct Student {  
    char *name;  
    double *scores;  
    grade *grades;  
    int grad_year;  
} Student;
```



# UNIONS

- struct = store multiple different types of data in one thing
- union = can store one of multiple different types of data
  - share the same memory
  - you are responsible for interpreting stored values

```
union idc {  
    int i;  
    double d;  
    char c;  
};  
union idc var;  
var.d = 16.8;
```



# UNIONS (CONT.)

- Like structs/enums, can typedef unions

```
typedef union idc {  
    int i;  
    double d;  
    char c;  
} idc;  
idc var;  
var.i = 16;
```

## UNIONS (CONT.)

- Can also combine unions with structs/enums
- Example:
  - type enum to define possible types
  - struct containing type and union
  - union has multiple different options
  - type stores type currently being stored