Computer Programming Using C
COP 3275 - Summer 2017
Lecture 21: Structures, Unions, andEnumerations (cont.)

C

/* Programming */

By Ahmed E. Khaled
Declaring a Structure Tag

• A *structure tag* is a name used to identify a *particular kind of structure*.
• The declaration of a structure tag named `part`:

```c
struct part {
    int number;
    char name[NAME_LEN+1];
    int on_hand;
};
```
Recap to the previous lecture:

```c
#include <stdio.h>
#include <string.h>

struct Books {
    char title[50];    // char array to store title
    int book_id;       // integer to store book id
};

void main() {
    struct Books Book1;  // Declare Book1 of type Book

    strcpy( Book1.title, "C Programming");
    Book1.book_id = 6495407;

    printf( "Book 1 title : %s\n", Book1.title);
    printf( "Book 1 book_id : %d\n", Book1.book_id);
}
```
Structures as Arguments and Return Values

• Functions may have structures as arguments and return values.

• A function with a structure argument:

```c
void print_part(struct part p){
    printf("Part number: %d\n", p.number);
    printf("Part name: %s\n", p.name);
    printf("Quantity on hand: %d\n", p.on_hand);
}
```

• A call of print_part:

```c
print_part(part1);
```
- A function that returns a part structure:

```c
struct part build_part(int NM, char *NA, int onH) {
    struct part p;
    p.number = NM;
    strcpy(p.name, NA);
    p.on_hand = onH;
    return p;
}
```

- A call of `build_part`:

```c
part1 = build_part(528, "Disk drive", 10);
```
Nested Arrays and Structures

- Structures and arrays can be combined without restriction.

- Arrays may have structures as their elements, and structures may contain arrays and structures as members.
Nested Structures

• Nesting one structure inside another is often useful.

• Suppose that `person_name` is the following structure:

```c
struct person_name {
    char first[FIRST_NAME_LEN+1];
    char middle_initial;
    char last[LAST_NAME_LEN+1];
};
```
- We can use `person_name` as part of a larger structure:

```c
struct student {
    struct person_name name;
    int id, age;
    char sex;
} student1, student2;
```

- Accessing `student1`'s first name, middle initial, or last name requires two applications of the `. operator`:

```c
strcpy(student1.name.first, "Fred");
student1.name.middle_initial = 'E';
```