



Unit VII – Part 01

(Structure in C)

Lecture Slide



AS2023



Operations on Individual Member



- Individual members are identified using the member operator
- A member with the *dot operator* along with its name can be treated like any other variable name
- And therefore can be manipulated using expression and operations

Example

```
if (student1.mark==111) {  
    student1.mark+=10;  
}
```

- We can also apply increment and decrement operators to numeric type members



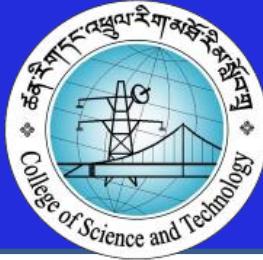
Arrays of Structure

- We use structure to describe the format of a number of related variables
- For example, in analyzing the marks obtained by a class of students, we may use template to describe student name and marks obtained in various subjects and declare all the students as structure variables
- In such cases, we may declare an array of structure, where each element of array represents a structure variable



Arrays of Structure

```
struct marks{  
    int number;  
    float subject1;  
    float subject2;  
    float subject3;  
} student[103];
```



Array within Structures

- C permits the usage of arrays as structure members
- We have already used arrays of characters inside a structure. Similarly, we can use one dimensional or multi dimensional arrays of type float or int

```
struct marks{  
    int number;  
    float subject[3];  
} student[103];
```

- Elements of this array can be accessed using appropriate subscript



Royal University of Bhutan



Keyword **typedef**

- The **typedef** keyword to create an alias name for data types.
- It is commonly used with structures to simplify the syntax of declaring variables.

```
struct Distance{  
    int feet;  
    float inch;  
};  
int main() {  
    struct Distance d1, d2;  
}
```

```
typedef struct Distance{  
    int feet;  
    float inch;  
} distances;  
int main() {  
    distances d1, d2;  
}
```



Royal University of Bhutan

Ways of Nesting Structure



Two Ways:

1. Embedded nested Structure
2. Separate nested Structure



1. Embedded nested Structure

- Structure within Structure
- Means nesting of structures
- Consider the following structure to store information about the salary of employees

```
struct salary{  
    char name[30];  
    int basic;  
    int teaching_allowance;  
    int house_rent_allowance;  
}employee;
```



1. Embedded nested Structure



- We can group all the items related to allowance together and declare them under a substructure as shown

```
struct salary{  
    char name[30];  
    int basic;  
    struct{  
        int teaching;  
        int house_rent;  
    }allowance;  
}employee;
```



1. Embedded nested Structure



- An inner most member in a nested structure can be accessed by chaining all the concerned structure variable with the member using *dot operator*

```
employee.allowance.house_rent;
```



2. Separate nested Structure



- Two structures are created
- The dependent structure should be used inside the main structure as a member

Dependent Structure :

```
struct allowance
{
    int teaching;
    int house_rent;
};
```



2. Separate nested Structure



Main Structure :

```
struct salary
{
    char name[30];
    int basic;
    struct allowance al;
};
```



Royal University of Bhutan



Struct and Pointers

- Here's how you can create pointers to structs.

```
struct name {  
    member1;  
    member2;  
    . . .  
};  
int main() {  
    struct name *ptr, Dorji;  
}
```

Here, *ptr* is a pointer to struct.



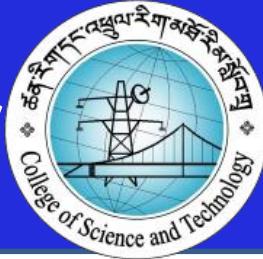
Royal University of Bhutan



Access members using Pointer

- To access members of a structure using pointers, we use the `->` operator.

```
#include <stdio.h>
struct person {
    int age; float weight;
};
int main() {
    struct person *personPtr, person1;
    personPtr = &person1;
    printf("Enter age: ");
    scanf("%d", &personPtr->age);
    printf("Enter weight: ");
    scanf("%f", &personPtr->weight);
    printf("Displaying:\n");
    printf("Age: %d\n", personPtr->age);
    printf("weight: %f", personPtr->weight);
    return 0;
}
```



Access members using Pointer

- In the example, the address of `person1` is stored in the `personPtr` pointer using

```
personPtr = &person1; .
```

- Now, you can access the members of `person1` using the `personPtr` pointer.
- By the way,
 - `personPtr->age` is equivalent to `(*personPtr).age`
 - `personPtr->weight` is equivalent to `(*personPtr).weight`



Thank you