UNIT 10
STRUCTURE

LH - 5HRS

PRESENTED BY: **ER. SHARAT MAHARJAN**

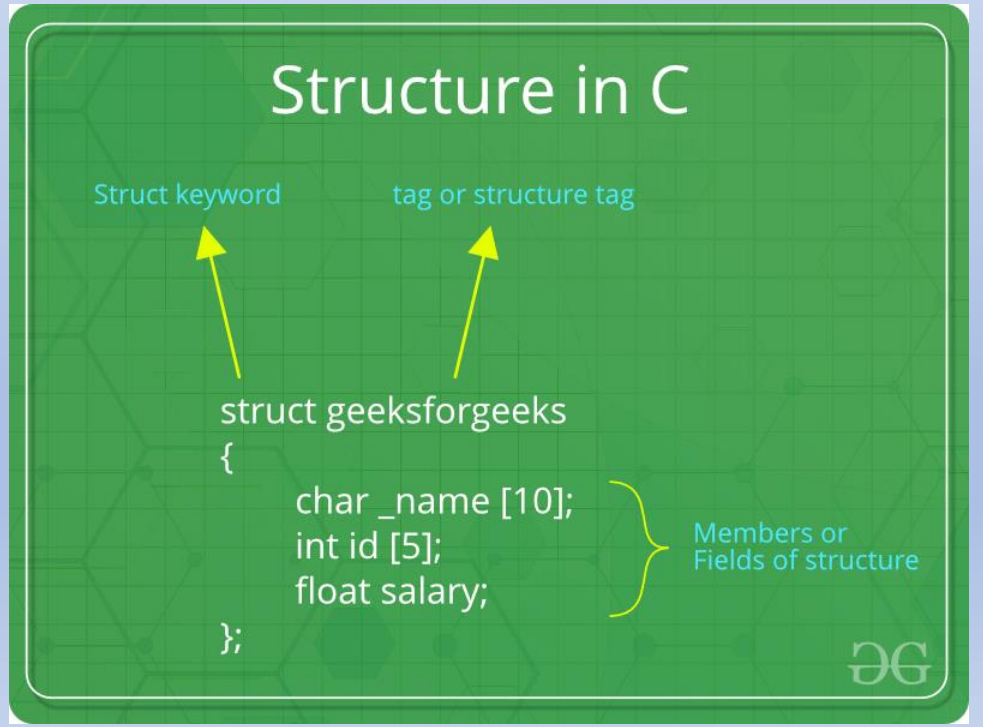
NETWORK PROGRAMMING

PRIME COLLEGE, NAYABAZAAR

CONTENTS (LH - 5HRS)

10.1 Introduction

- A structure is a key word that create user defined data type in C/C++.
- A structure creates a data type that can be used to group items of possibly different types into a single type.



- Following is an example:

```
struct employee
{
        int e_id;
        char name[50];
        float salary;
};
```

10.2 Declaration and Initialization

- A structure variable is a variable of a structure type.
- A structure variable can either be declared with structure declaration or as a separate declaration like basic types.

```
// A variable declaration with structure declaration.
struct Point{
int x, y;
} p1; // The variable p1 is declared with 'Point'

// A variable declaration like basic data types
struct Point{
int x, y;};
int main(){
struct Point p1; // The variable p1 is declared like a normal variable
}
```

- Structure members can be initialized using curly braces ‘{}’.
- For example:

```
struct bankaccount{
        int acct_no;
        char acct_type;
        char name[20];
        float balance;
}myaccount={1001, ’C’, “Sharat Maharjan”, 10000.0};
```

LAB 1: WAP to demonstrate initializing structure variable.

```
#include<stdio.h>
struct student{
        int id;
        char name[20];
        float marks;
        char gender;
};
int main(){
        struct student s={1,"Sharat Maharjan",80,'M'};
        printf("ID = %d",s.id);
        printf("\nName = %s",s.name);
        printf("\nMarks = %.2f",s.marks);
        printf("\nGender = %c",s.gender);
        return 0;
}
```

10.3 Nested Structure

- If a structure is defined as member of another structure then it is called nested structure.
- The outing structure is called nesting structure and inner is called nested.
- For example:

```
struct person{
        char name[20];
                struct{
                        int day;
                        int month;
                        int year;
                }birthday;
        float salary;
}person;
```

- The members contained within the inner structure can be accessed as:

person.birthday.day and so on.

10.4 Array of Structure

- Like other primitive data types, we can create an array of structures. **For Example:**

```
#include<stdio.h>
struct Point{
int x, y;
};
int main(){
// Create an array of structures
struct Point arr[10];

// Access array members
arr[0].x = 10;
arr[0].y = 20;

printf("%d %d", arr[0].x, arr[0].y);
return 0;}
```

LAB 2: WAP creating structure named student that has name, roll, marks and remarks as its members. Assume appropriate types and use this structure to read and display records of 5 students.

```
#include<stdio.h>
struct student{
        char name[20];
        int roll;
        float marks;
        char remarks;
};
int main(){
        struct student s[5];
        int i;
        printf("Enter details of 5 students:\n");
        for(i=0;i<5;i++){
                printf("Name:");
                scanf("%s", s[i].name);
                printf("Roll:");
                scanf("%d", &s[i].roll);
                printf("Marks:");
                scanf("%f", &s[i].marks);
                printf("Remarks:");
                scanf(" %c", &s[i].remarks);
        }
        printf("Details of 5 students:\n");
        for(i=0;i<5;i++){
                printf("Name = %s \t Roll = %d \t Marks =%f \t Remarks = %c \n", s[i].name, s[i].roll, s[i].marks, s[i].remarks);
        }
        return 0;
}
```

10.5 Array within Structure

```
struct bankaccount{
        int acct_no;
        char acct_type;
        char name[20];      //array within structure
        float balance;
}myaccount={1001, 'C', "Sharat Maharjan", 10000.0};
```

LAB 3: WAP to read 100 students record with following fields and display the record of BCA faculty only.

```
#include<stdio.h>
#include<string.h>
struct student{
        int roll;
        char name[20];
        char faculty[20];
        struct{
                int day;
                int month;
                int year;
        }birthdate;
};
int main(){
        struct student s[100];
        int i;
        printf("Enter student details:\n");
        for(i=0;i<100;i++){
                printf("Roll:");
                scanf("%d",&s[i].roll);
                printf("Name:");
scanf("%s",s[i].name);
                printf("Faculty:");
                scanf("%s",s[i].faculty);
                printf("Enter day of birth:");
                scanf("%d",&s[i].birthdate.day);
                printf("Enter month of birth:");
                scanf("%d",&s[i].birthdate.month);
                printf("Enter year of birth:");
                scanf("%d",&s[i].birthdate.year);
        }
for(i=0;i<100;i++){
 if(strcmp(s[i].faculty,"BCA")==0){
  printf("Roll = %d \t Name = %s \t Faculty = %s \t Day = %d \t Month = %d
\tYear=%d\n",s[i].roll,s[i].name,s[i].faculty,s[i].birthdate.day,s[i].birthdate.
month,s[i].birthdate.year);
                }
        }
        return 0;
}
```

10.6 Passing Structure and Array of Structure to function

- Structure members can be passed to functions as actual arguments in function call like ordinary variables.

Passing Structure Array of Structure:

- It is similar to passing an array of any type to a function.
- That is, the name of the array of structure is passed by the calling function which is the base address of the array of structure.

LAB 4: WAP to read name, id and salary of 5 employee and display their details using function.

```
#include<stdio.h>
struct employee{
		char name[20];
		int id;
		float salary;
};
void display(struct employee []);
int main(){
		struct employee e[5];
		printf("Enter details of 5 employee:\n");
		for(int i=0;i<5;i++){
				printf("Name:");
				scanf("%s",e[i].name);
				printf("ID:");
				scanf("%d",&e[i].id);
				printf("Salary:");
				scanf("%f",&e[i].salary);
		}
		display(e);	//passing address of 1st element of an array of structure
		return 0;
}
void display(struct employee e[]){
		printf("The details of 5 students:\n");
		for(int i=0;i<5;i++){
				printf("\nName:%s",e[i].name);
				printf("\nID:%d",e[i].id);
				printf("\nSalary:%f",e[i].salary);
		}
}
```

10.7 Structure and pointer

LAB 5: Program to demonstrate the use of pointer to structure.

```
#include<stdio.h>
struct employee{
		char name[20];
		int id;
		float salary;
};
void display(struct employee []);
int main(){
		struct employee e[5],*ptr;//*ptr is pointer to structure
		ptr=e;
		printf("Enter details of 5 employee:\n");
		for(int i=0;i<5;i++){
				printf("Name:");
				scanf("%s",(ptr+i)->name);
				printf("ID:");
				scanf("%d",&(ptr+i)->id);
				printf("Salary:");
				scanf("%f",&(ptr+i)->salary);
		}
		display(e);
		return 0;
}
void display(struct employee *p){
		printf("The details of 5 students:\n");
		for(int i=0;i<5;i++){
				printf("\nName:%s",(p+i)->name);
				printf("\nID:%d",(p+i)->id);
				printf("\nSalary:%f",(p+i)->salary);
		}
}
```

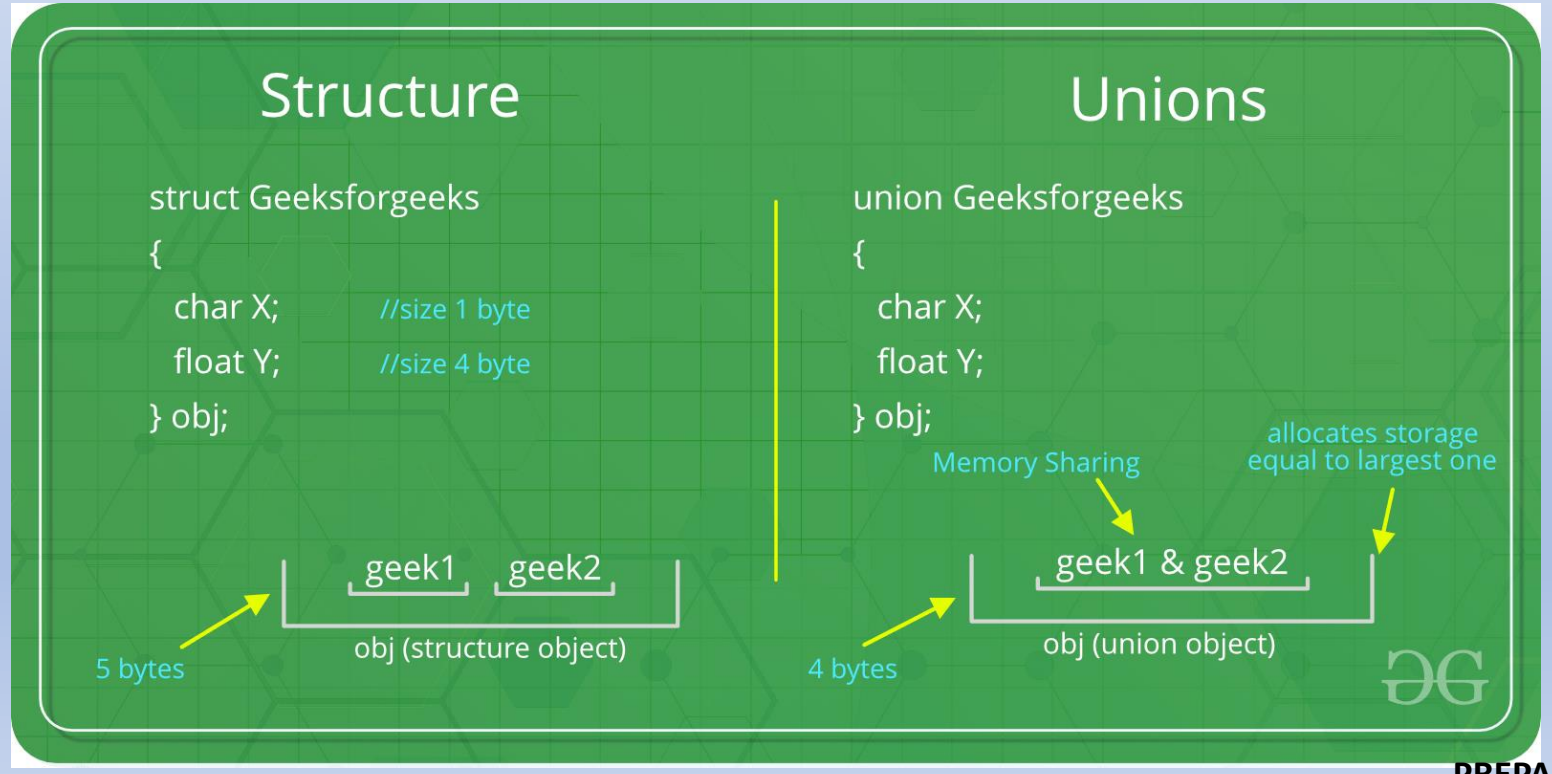
10.8 Bit fields

- In C, we can specify size (in bits) of structure members.
- The idea is to use memory efficiently when we know that the value of a field will never exceed a limit or is within a small range.
- For example, the value of day is always from 1 to 31, the value of month is from 1 to 12, we can optimize the space using bit fields.

```
int day:5;      //will use 5bits
int month:4;//will use 4bits
```

10.9 Union and Its importance

- Like Structures, union is a user defined data type.
- In union, all members share the same memory location.



Example:

```
#include <stdio.h>
// Declaration of union is same as structures
union test {
        int x, y;
};
int main()
{
        // A union variable t
        union test t;

        t.x = 2; // t.y also gets value 2
        printf("x = %d, y = %d\n\n",t.x, t.y);

        t.y = 10; // t.x is also updated to 10
        printf("x = %d, y = %d\n\n",t.x, t.y);
        return 0;
}
```

Importance of Union:

- C unions are used to save memory.
- Quite important when memory is valuable, such as in embedded systems.

	STRUCTURE	UNION
Keyword	The keyword **struct** is used to define a structure	The keyword **union** is used to define a union.
Size	When a variable is associated with a structure, the compiler allocates the memory for each member. The size of structure is **greater than or equal to the sum of sizes of its members.**	when a variable is associated with a union, the compiler allocates the memory by considering the size of the largest memory. So, size of **union is equal to the size of largest member.**
Memory	Each member within a structure is assigned unique storage area of location.	Memory allocated is shared by individual members of union.
Value Altering	Altering the value of a member will not affect other members of the structure.	Altering the value of any of the member will alter other member values.
Accessing members	Individual member can be accessed at a time.	Only one member can be accessed at a time.
Initialization of Members	Several members of a structure can initialize at once.	Only the first member of a union can be initialized.

THANK YOU FOR YOUR ATTENTION