UNIT 3
Preprocessor Directives

Prof. Keval Mehta, Senior Cyber Security Trainer
Cyber Security

Parul® University

DIGITAL LEARNING CONTENT

CHAPTER-3

Enumerators, Structures, Unions

Enumerators

Enumeration or Enum in C is a special kind of data type defined by the user. It consists of constant integrals or integers that are given names by a user. The use of enum in C to name the integer values makes the entire program easy to learn, understand, and maintain by the same or even different programmer.

Syntax to Define Enum in C:

enum enum_name{int_const1, int_const2, int_const3, …. int_constN};

Example: enum cars{BMW, Ferrari, Jeep, Mercedes-Benz};

enum cars{BMW=3, Ferrari=5, Jeep=0, Mercedes-Benz=1 };

Contd…

How to Create and Implement Enum in C Program:

```
#include <stdio.h>
    enum days{Sunday=1, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday};
void main(){
    // printing the values of weekdays
    for(int i=Sunday;i<=Saturday;i++){
        printf("%d, ",i);
    }
}
```

Contd…

Example 2: Assigning and Fetching Custom Values of Enum Elements

```
#include<stdio.h>
enum containers{
    cont1 = 5,
    cont2 = 7,
    cont3 = 3,
    cont4 = 8
};
```

```
void main(){
    // Initializing a variable to hold enums
    enum containers cur_cont = cont2;
printf("Value of cont2 is = %d \n", cur_cont);
cur_cont = cont3;
printf("Value of cont3 is = %d \n", cur_cont);
cur_cont = cont1;
printf("Value of hearts is = %d \n", cur_cont);
}
```

Structure

The structure in C is a user-defined data type that can be used to group items of possibly different types into a single type. The **struct keyword** is used to define the structure in the C programming language. The items in the structure are called its **member** and they can be of any valid data type.

Syntax

```
struct structure_name {
   data_type member_name1;
   data_type member_name1;
   ....
   ....
  };
```

Contd…

```
#include <stdio.h>
struct str1 {
        int i;
        char c;
        float f;
        char s[30];
};
void main() {
        struct str1 var1 = { 1, 'A', 1.00, "GeeksforGeeks" };
        printf("Struct 1:\n\ti = %d, c = %c, f = %f, s = %s\n",var1.i, var1.c, var1.f, var1.s);
        return 0;
}
```

Complex Structure

C provides us the feature of nesting one structure within another structure by using which, complex data types are created. For example, we may need to store the address of an entity employee in a structure. The attribute address may also have the subparts as street number, city, state, and pin code.

A **nested structure** in C is a structure within structure. One structure can be declared inside another structure in the same way structure members are declared inside a structure.

Contd…

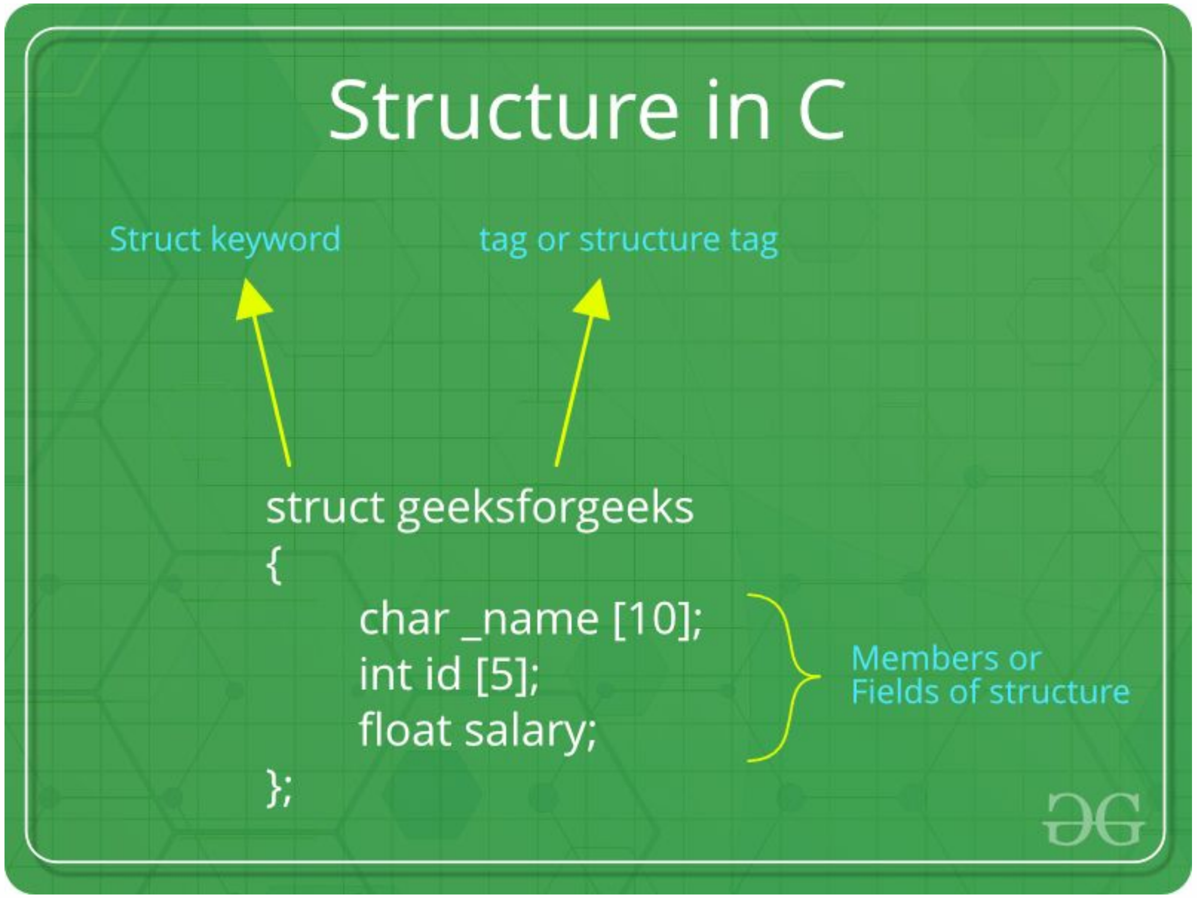
Syntax:

```
struct name_1 {
member1;
member2;
membern;
struct name_2 {
member_1;
member_2;
member_n;
}, var1
} var2;
```

structure and functions

The structure in C is a user-defined data type that can be used to group items of possibly different types into a single type. The struct keyword is used to define the structure in the C programming language. The items in the structure are called its member and they can be of any valid data type.

Contd…



```
struct geeksforgeeks
{
        char _name [10];
        int id [5];
        float salary;
};
```

Parul® University

Array of Structures

An array whose elements are of type structure is called array of structure. It is generally useful when we need multiple structure variables in our program.

Declaration of Array of Structures

```
struct structure_name array_name [number_of_elements];
```

Initialization of Array of Structures

```
struct structure_name array_name [number_of_elements] = {
            {element1_value1, element1_value2, ....},
            {element2_value1, element2_value2, ....},
             ......
};
```

Parul® University

Contd….

```
#include <stdio.h>
struct Employee {
    char Name[20];
    int employeeID;
    int WeekAttendence[7];
};
void main()  {
    struct Employee emp[5];
    for (int i = 0; i < 5; i++) {
            emp[i].employeeID = i;
            strcpy(emp[i].Name, "XYZ");
            int week;
            for (week = 0; week < 7; week++) {
                    int attendence;
    emp[i].WeekAttendence[week] = week;
            }
    }
    printf("\n");

    // printing data
    for (int i = 0; i < 5; i++) {
            printf("Emplyee ID: %d - Employee
Name: %s\n",
                    emp[i].employeeID,
emp[i].Name);
            printf("Attendence\n");
            int week;
            for (week = 0; week < 7; week++) {
                    printf("%d ",
emp[i].WeekAttendence[week]);
            }
            printf("\n");
    }
}
```

Arrays within structures

In C, a structure is a user-defined data type that allows us to combine data of different data types. While an array is a collection of elements of the same type. In this article, we will discuss the concept of an array that is declared as a member of the structure.

Syntax to Declare Array Within Structure

```
struct StructureName {
// Other members
dataType arrayName[arraySize];
};
```

Initialize Array Within Structure in C

```
struct StructureName variableName = { ...
   {element1_value1, element1_value2 ...} };
```

Contd…

Accessing Elements of an Array within a Structure

```
structureName.arrayName[index]
#include <string.h>
#include <stdio.h>
struct Employee {
    char Name[20];
    int employeeID;
    int WeekAttendence[7];
};
```

```
void main()      {
    struct Employee emp;
    emp.employeeID = 1;
    strcpy(emp.Name, "Rohit");
    int week;
    for (week = 0; week < 7; week++) {
        int attendence;
        emp.WeekAttendence[week] =
week;
    }
    printf("\n");
```

Contd…

```
printf("Emplyee ID: %d - Employee Name: %s\n", emp.employeeID,
emp.Name);
      printf("Attendence\n");
      for (week = 0; week < 7; week++) {
         printf("%d ", emp.WeekAttendence[week]);
      }
      printf("\n");
```

Anonymous structures

Anonymous unions/structures are also known as unnamed unions/structures as they don’t have names. Since there is no names, direct objects(or variables) of them are not created and we use them in nested structure or unions.

```
// Anonymous structure example
struct
{
 char alpha;
 int num;
};
```

Contd….

```
#include <stdio.h>
struct Scope {
        // Anonymous union
        union {
                char alpha;
                int num;
        };
};
void main()    {
        struct Scope x;
        x.num = 65;
// Note that members of union are accessed directly
        printf("x.alpha = %c, x.num = %d", x.alpha, x.num);
}
```

Nested structures

A **nested structure** in C is a structure within structure. One structure can be declared inside another structure in the same way structure members are declared inside a structure.

Syntax:

```
struct name_1 {
                member1;
                member2;
  membern;
struct name_2  {
       member_1;
       member_2;
      member_n;
   }, var1
} var2;
```

Contd…

```
#include <stdio.h>
#include <string.h>
struct Employee   {
int employee_id;
char name[20];
int salary;
};
struct Organisation {
char organisation_name[20];
char org_number[20];
struct Employee emp;
};
```

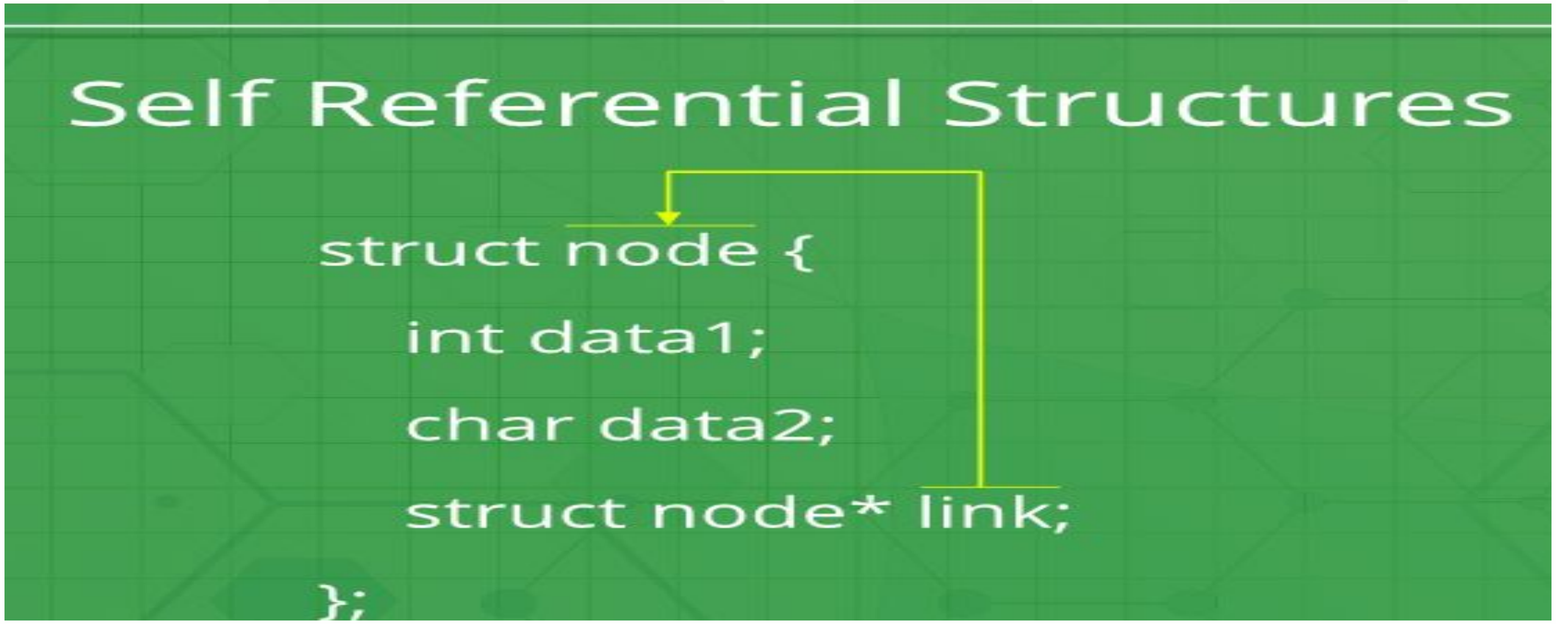
```
// Driver code
int main()      {
struct Organisation org;
printf("The size of structure organi :%ld\n",
              sizeof(org));
org.emp.employee_id = 101;
strcpy(org.emp.name, "Robert");
org.emp.salary = 400000;
strcpy(org.organisation_name, "XYZ");
strcpy(org.org_number, "GFG123768");
```

Contd…

```
// Printing the details
printf("Organisation Name : %s\n",
            org.organisation_name);
printf("Organisation Number : %s\n",
            org.org_number);
printf("Employee id : %d\n",
            org.emp.employee_id);
printf("Employee name : %s\n",
            org.emp.name);
printf("Employee Salary : %d\n",
            org.emp.salary);
}
```

Self Referential Structure

Self Referential structures are those structures that have one or more pointers which point to the same type of structure, as their member.



Self Referential Structures

```
struct node {
    int data1;
    char data2;
    struct node* link;
};
```

Contd…

```
struct node {
	int data1;
	char data2;
	struct node* link;
};

int main()
{
	struct node ob;
	return 0;
}
```

Structure padding

Structure padding is the addition of some empty bytes of memory in the structure to naturally align the data members in the memory. It is done to minimize the CPU read cycles to retrieve different data members in the structure.

```
// structure A
typedef struct structa_tag
{     char c;
      short int s;
} structa_t;
// structure B
typedef struct structb_tag
{     short int s;
      char c;
      int i;
} structb_t;
```

```
// structure C
typedef struct structc_tag {
      char c;
      double d;
      int s;
} structc_t;
```

```
// structure D
typedef struct
structd_tag {
      double d;
      int s;
      char c;
} structd_t;
```

Contd…

- **Size of Structure A** = Size of (char + short int) = 1 + 2 = **3.**
- **Size of Structure B** = Size of (short int + char + int) = 2 + 1 + 4 = **7.**
- **Size of Structure C** = Size of (char + double + int) = 1 + 8 + 4 = **13.**
- **Size of Structure A** = Size of (double + int + char) = 8 + 4 + 1= **13**

```
void main()   {
   printf("sizeof(structa_t) = %lu\n", sizeof(structa_t));
   printf("sizeof(structb_t) = %lu\n", sizeof(structb_t));
   printf("sizeof(structc_t) = %lu\n", sizeof(structc_t));
   printf("sizeof(structd_t) = %lu\n", sizeof(structd_t));
}
```

Unions

The Union is a user-defined data type in C language that can contain elements of the different data types just like structure. But unlike structures, all the members in the C union are stored in the same memory location. Due to this, only one member can store data at the given instance.

Syntax of Union in C

```
union union_name {
    datatype member1;
    datatype member2;
     };
```

Parul® University

Contd…

Different Ways to Define a Union Variable

1. With Union Declaration
2. After Union Declaration

Defining Union Variable with Declaration

```
union union_name {
    datatype member1;
    datatype member2;
    ...
  } var1, var2, ...;
```

Defining Union Variable after Declaration

```
union union_name var1,
var2, var3...;
```

Contd…

```
#include <stdio.h>
// union template or declaration
union un {
        int member1;
        char member2;
        float member3;
};
```

```
// driver code
void main()   {
        union un var1;
        var1.member1 = 15;
        printf("The value stored in member1 = %d", var1.member1);
}
```

Example of C Bit Fields

Structure Without Bit Fields

```
#include <stdio.h>

struct date {
    unsigned int d;
    unsigned int m;
    unsigned int y;
};
void main(){
    printf("Size of date is %lu bytes\n", sizeof(struct date));
    struct date dt = { 31, 12, 2014 };
    printf("Date is %d/%d/%d", dt.d, dt.m, dt.y);
}
```

Bit Fields

Structure with Bit Field

```
struct date
{
// month has value between 0 and 15,
// so 4 bits are sufficient for month variable.
    int month : 4;
};
```

Typedef

The typedef is a keyword that is used to provide existing data types with a new name. The C typedef keyword is used to redefine the name of already existing data types.

C typedef Syntax

```
typedef existing_name alias_name;
```

Example of typedef in C

```
typedef long long x;
```

Example:

```
#include <stdio.h>

// defining an alias using typedef

typedef long long ll;

// Driver code

void main()       {

    // using typedef name to declare
variable

    x var = 20;

    printf("%ld", var);

}
```

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in