Structure

Course Code: CSE-121

Course Title: Structural Programming Language

Khandaker Jannatul Ritu, Lecturer, BAIUST

Outlines

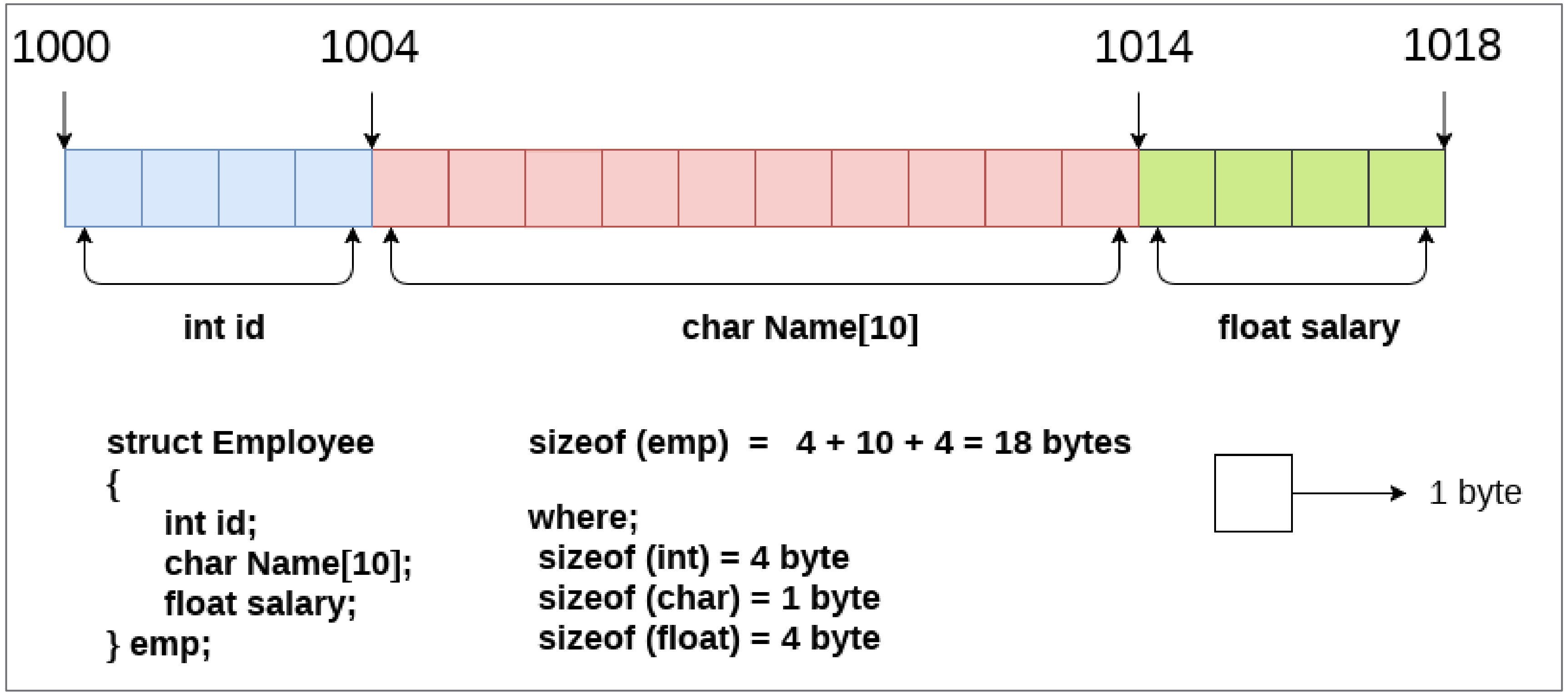
❑ What is Structure

Structure in c is a user-defined data type that enables us to store the collection of different data types. Each element of a structure is called a member. Structures ca; simulate the use of classes and templates as it can store various information
The **struct** keyword is used to define the structure. Let's see the syntax to define the structure in c.

```
struct structure_name
{
    data_type member1;
    data_type member2;
    .
    .
    data_type memeberN;
};
```

Let's see the example to define a structure for an entity employee in c.

```
struct employee
{   int id;
    char name[20];
    float salary;
};
```

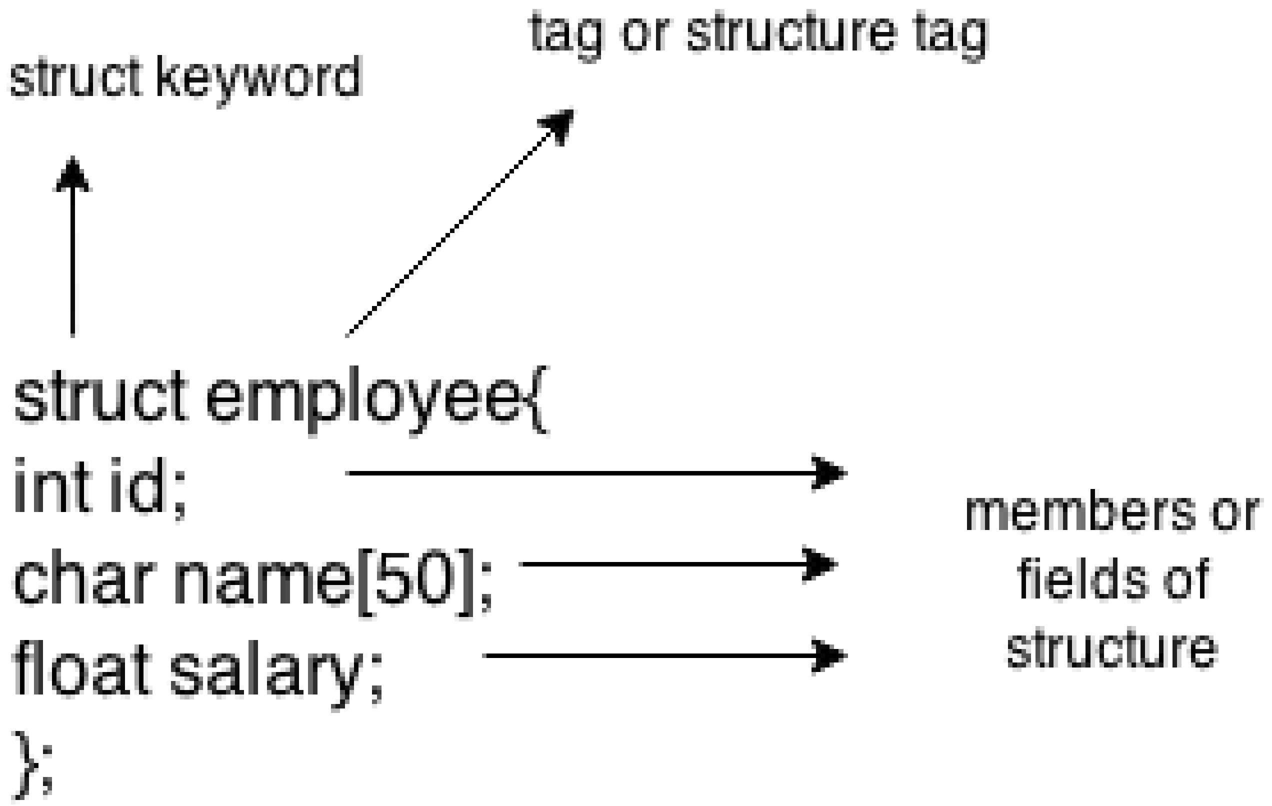


```
struct Employee
{
    int id;
    char Name[10];
    float salary;
} emp;
```

sizeof (emp) = 4 + 10 + 4 = 18 bytes

where;
sizeof (int) = 4 byte
sizeof (char) = 1 byte
sizeof (float) = 4 byte

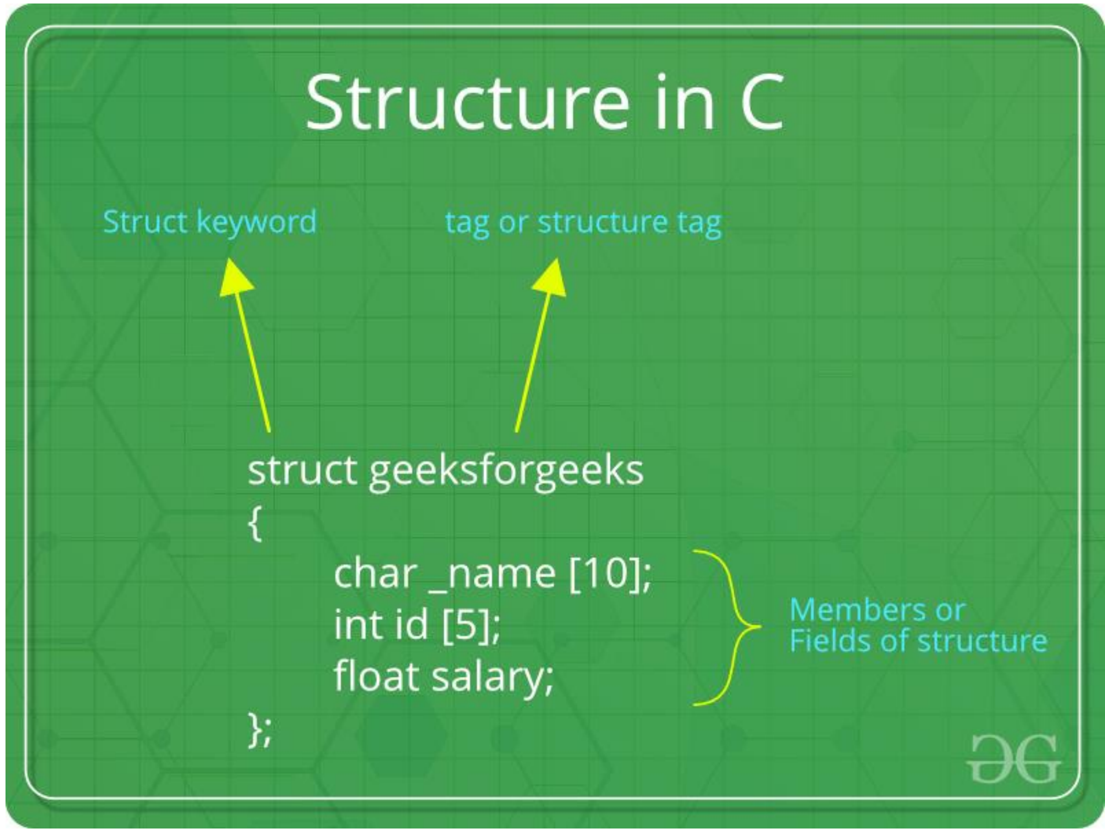
Here, **struct** is the keyword; **employee** is the name of the structure; **id**, **name**, and **salary** are the members or fields of the structure. Let's understand it by the diagram given below:



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

❑ C Structures

- ✓ 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.
- ✓ Additionally, the values of a structure are stored in contiguous memory locations.



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

❑ C Structure Declaration

We have to declare structure in C before using it in our program. In structure declaration, we specify its member variables along with their datatype. We can use the struct keyword to declare the structure in C using the following syntax:

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

The above syntax is also called a structure template or structure prototype and no memory is allocated to the structure in the declaration.

1. Structure Variable Declaration with Structure Template

```
struct structure_name {
    data_type member_name1;
    data_type member_name1;
    ....
    ....
}variable1, varaible2, ... ;
```

2. Structure Variable Declaration after Structure Template

```
// structure declared beforehand
struct structure_name variable1, variable2, .......;
```

❑Access Structure Members

We can access structure members by using the (.) dot operator.

Syntax

```
        structure_name.member1;
        strcuture_name.member2;
```

N.B. :-

```
struct Point
{
   int x = 0;  // COMPILER ERROR:  cannot initialize members here
   int y = 0;  // COMPILER ERROR:  cannot initialize members here
};
```

The reason for the above error is simple. When a datatype is declared, no memory is allocated for it. Memory is allocated only when variables are created.

❑Default Initialization

By default, structure members are not automatically initialized to 0 or NULL. Uninitialized structure members will contain garbage values.

However, when a structure variable is declared with an initializer, all members not explicitly initialized are zero-initialized.

```
struct Point
{
     int x;
     int y;
};
struct Point p = {0}; // Both x and y are initialized to 0
```

❑ C program to illustrate use of structure:- Print Person’s Age & Salary

```c
#include <stdio.h>
///global structure
struct Person{
    int age;
    int salary;
};

int main(){
    struct Person p1, p2;
    p1.age = 22;
    p1.salary = 2200;

    printf("Age = %d \n", p1.age);
    printf("Salary = %d \n\n", p1.salary);

    p2.age = 44;
    p2.salary = 2220;

    printf("Age = %d \n", p2.age);
    printf("Salary = %d \n", p2.salary);
}
```

Output:-

```
Age = 22
Salary = 2200

Age = 44
Salary = 2220
```

❑C program to illustrate Local & Global Structure

```c
struct Student{   ///global structure
    int age;
    int salary;
};
struct Student s1; ///Global structure variable
int main(){
    ///local structure
    struct Person{
        int age;
        int salary;
    };
    struct Person p1; ///Local structure variable
    p1.age = 22;
    p1.salary = 2200;
    printf("Age = %d \n", p1.age);
    printf("Salary = %d \n\n", p1.salary);
    s1.age = 44;
    s1.salary = 2220;
    printf("Age = %d \n", s1.age);
    printf("Salary = %d \n", s1.salary);
}
```

Output:-

```
Age = 22
Salary = 2200

Age = 44
Salary = 2220
```

❑C program to illustrate input Structure element

```c
#include <stdio.h>
///global structure
struct Student{
    int age;
    int salary;
};

int main(){
    struct Student s1;
    printf("Enter Age : ");
    scanf("%d", &s1.age);
    printf("Enter Salary : ");
    scanf("%d", &s1.salary);

    printf("\nPerson Information : \n");
    printf("Age : %d \n", s1.age);
    printf("Salary : %d \n", s1.salary);
}
```

Output:

```
Enter Age : 23
Enter Salary : 34

Person Information :
Age : 23
Salary : 34
```

❑C program to illustrate Initialize Structure Variables

```c
#include <stdio.h>
///global structure
struct Student{
    int age;
    int salary;
};

int main(){
    ///Initialize
    struct Student s1 = {23, 45000};
    printf("\nPerson Information : \n");
    printf("Age : %d \n", s1.age);
    printf("Salary : %d \n", s1.salary);
}
```

```
Output:

Person Information :
Age : 23
Salary : 45000
```

❑C program to illustrate Structure Comparison

```
#include <stdio.h>
///global structure
struct Student{
    int age;
    int salary;
};
int main(){
    ///Initialize
    struct Student s1 = {23, 45000};
    struct Student s2 = {23, 45001};

    if(s1.age == s2.age && s1.salary==s2.salary){
        printf("Equal");
    }
    else{
        printf("Not Equal");
    }
}
```

Output:

```
Not Equal
```

❑C program to illustrate Array of Structure

```c
#include <stdio.h>

struct Student{
    int age;
    int salary;
};
int main(){
    struct Student s[3];
    printf("Enter Age and Salary: \n");
    for(int i=0; i<3; i++)
    {
        scanf("%d %d", &s[i].age, &s[i].salary);
    }

    printf("--- Information --- \n");
    for(int i=0; i<3; i++)
    {
        printf("Age : %d    Salary : %d\n", s[i].age, s[i].salary);
    }
}
```

Output:

```
Enter Age and Salary:
34 5000
32 4000
45 9000
--- Information ---
Age : 34    Salary : 5000
Age : 32    Salary : 4000
Age : 45    Salary : 9000
```

❑C program to illustrate Array within the Structure

```c
#include <stdio.h>

struct Student{
    char name[50];
    int age;
    int salary;
};
int main(){
    struct Student s[3];
    printf("Enter Age and Salary: \n");
    for(int i=0; i<3; i++)
    {
        scanf("%s %d %d",&s[i].name, &s[i].age, &s[i].salary);
    }

    printf("--- Information --- \n");
    for(int i=0; i<3; i++)
    {
        printf("Name : %s   Age : %d    Salary : %d\n",s[i].name, s[i].age, s[i].salary);
    }
}
```

```
Output:-

Enter Age and Salary:
karim 30 5500
rahim 31 4500
john 35 10000
--- Information ---
Name : karim   Age : 30    Salary : 5500
Name : rahim   Age : 31    Salary : 4500
Name : john   Age : 35    Salary : 10000
```

❑C program to passing structure variable to function

```c
#include <stdio.h>

struct Student{
    char name[50];
    int age;
    int salary;
};

void show(struct Student ss)
{
    printf("Name : %s \n", ss.name);
    printf("Age : %d \n", ss.age);
    printf("Salary : %d \n", ss.salary);
}
int main(){
    struct Student st = {"karim", 65, 1234567890};
    show(st);
}
```

Output:

```
Name : karim
Age : 65
Salary : 1234567890
```

Resources to Learn Structure:

$\Longrightarrow$ Anisul Islam C-programming playlist