Structure
Union
Enumeration
Typedef

Course Title :- Structured Programming Language Sessional

Course Code :- CSE-122 [SECTION-B]

Level Term: 1-II-A(G1) & 1-II-B(G3,G4)

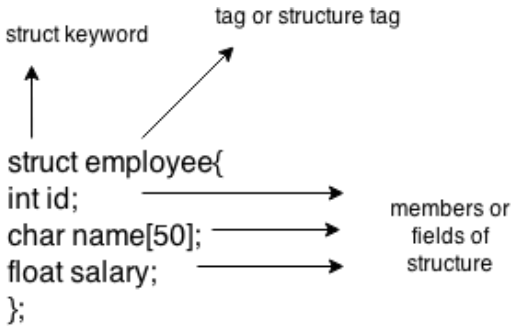
Khandaker Jannatul Ritu, Lecturer(CSE), BAIUST

Outlines

1. What is Structure
2. C Structure Declaration
3. Structure Initialization, Access Structure Members, Size of a structure
4. C program to illustrate Local & Global Structure
5. C program to illustrate Structure Comparison
6. C program to illustrate Array of Structure
7. C program to passing structure variable to function
8. Example of C Nested Structures
9. Embedded Definition for nested structure in C
10. Accessing Members in Nested Structures
11. Add two complex numbers using structure
12. Nested Structure to find a Rectangle Corners
13. Nested Structure to Report a student admission
14. How to Access a Structure Pointer in C ?
15. How to Pass a Structure Pointer in a function?
16. Exercise problems of structures
17. Introduction to union
18. Introduction to enum
19. Introduction to typedef

❑ What is Structure in C

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.




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

❑ C Structure 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

```
struct structure_name variable1, variable2, .......;
```

❑ Initialize Structure Members

```
struct Point
{
   int x = 0;  // COMPILER ERROR
   int y = 0;  // COMPILER ERROR
};
```

❑Default Initialization

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

❑ Size of a structure:

```
#include <stdio.h>
struct Student
{
        char name[32];
        float salary;
        int workerNo;
} s1;

int main()
{
     printf("\nsize of structure = %d bytes", sizeof(s1));
}
```

Output:-

```
size of structure = 40 bytes
```

❑ Access a structure members

```
#include <stdio.h>
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

```
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

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

     scanf("%d", &s1.age);
     scanf("%d", &s1.salary);
     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 Structure Comparison

```
#include <stdio.h>
//global structure
struct Student{
     int age;
     int salary;
};
int main(){
     //Initialize Structure
     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

```
#include <stdio.h>
struct Student{
      char name[50];
      int age;
      int salary;
};
int main(){
      int n;
      scanf("%d", &n);
      struct Student s[n];
      printf("Enter Age and Salary: \n");
      for(int i=0; i<n; i+.)
      {
          scanf("%s %d %d",&s[i].name, &s[i].age, &s[i].salary);
      }
      printf("-.. Information -.. \n");
      for(int i=0; i<n; i+.)
      {
          printf("Name : %s\n",s[i].name);
          printf("Age  : %d\n",s[i].age);
          printf("Salary : %d\n",s[i].salary);
      }
}
```

```
Output:-
3
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

```
#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
```

❑ Example of C Nested Structures – 1

```c
struct OuterStructure {
      // Members of the outer structure
      int outerMember1;
      float outerMember2;

      // Nested structure declaration
      struct InnerStructure {
            // Members of the inner structure
            int innerMember1;
            char innerMember2;
      } nestedStruct;
      // Declaration of a variable of the nested structure type
};
```

❑ Example of C Nested Structures – 2

```c
struct Address {
    char city[50];
    char state[50];
    int zip;
};

struct Employee {
    char name[50];
    int empID;
    struct Address address;
};
```

❑Embedded Definition for nested structure in C

```c
#include <stdio.h>

struct Address {
          char city[50];
          char state[50];
          int zip;
};

struct Employee {
          char name[50];
          int empID;
          struct Address address;
};

int main() {
          struct Employee emp = {"John Doe", 101, {"New York", "NY", 10001}};
          printf("Employee Details:\n");
          printf("Name: %s\n", emp.name);
          printf("ID: %d\n", emp.empID);
          printf("City: %s\n", emp.address.city);
          printf("State: %s\n", emp.address.state);
          printf("ZIP: %d\n", emp.address.zip);
          return 0;
}
```

```
Output:
Employee Details:
Name: John Doe
ID: 101
City: New York
State: NY
ZIP: 10001
```

❑Accessing Members in Nested Structures

Most of the nested structure in C allows access to the inner members by using the '.' on each of the nested levels.

Example:
#Accessing Members: Nested Structure in C

```
#include <stdio.h>

// Define the structures
struct Outer {
          int outerVar;
          struct Inner {
                  int innerVar;
          } innerStruct;
};

int main() {
    // Initialize the nested structure
    struct Outer obj = {10, {20}};

    // Access and print the values of the structure members
    printf("Outer Variable: %d\n", obj.outerVar);
    printf("Inner Variable: %d\n", obj.innerStruct.innerVar);

    return 0;
}
```

❑ Add two complex numbers using structure

```
#include <stdio.h>

struct complex
{
   int real, img;
}a,b,c;

int main()
{


    printf("Please enter first complex number\n");
    printf("Enter Real part of the 1st complex number\n");
    scanf("%d", &a.real);
    printf("Enter Imaginary part of the 1st complex number without i\n");
    scanf("%d", &a.img);

    printf("Please enter second complex number\n");
    printf("Enter Real part of the 2nd complex number\n");
    scanf("%d", &b.real);
    printf("Enter Imaginary part of the 2nd complex number without i\n");
    scanf("%d", &b.img);

   c.real = a.real + b.real;
   c.img = a.img + b.img;

   printf("Sum of the complex numbers: (%d) + (%di)\n", c.real, c.img);

   return 0;
}
```

❑Nested Structure to find a Rectangle Corners

```
#include <stdio.h>

// Define a structure for a point
struct Point {
      int x;
      int y;
};

// Define a structure for a rectangle
struct Rectangle {
      struct Point topLeft;
      struct Point bottomRight;
};

int main() {
      // Declare a variable of type Rectangle
      struct Rectangle rect;

      // Assign values to members of rect
      rect.topLeft.x = 10;
      rect.topLeft.y = 20;
      rect.bottomRight.x = 100;
      rect.bottomRight.y = 80;

      // Access and print the values
      printf("Top left corner: (%d, %d)\n", rect.topLeft.x, rect.topLeft.y);
      printf("Bottom right corner: (%d, %d)\n", rect.bottomRight.x, rect.bottomRight.y);

      return 0;
}
```

❑Nested Structure to Report a student admission

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

struct student
{
       int roll_no;
       char name[30];
       struct Date
       {
              int dd;
              int mm;
              int yyyy;
       }doj;
}s;

void main( )
{
       s.doj.dd=21;
       s.doj.mm=9;
       s.doj.yyyy=2014;
       s.roll_no = 1;
       strcpy(s.name, "Vibhuti Singh");

       printf( "Student Date of Joining (D/M/Y) : %d/%d/%d\n", s.doj.dd,s.doj.mm,s.doj.yyyy);
       printf( "Student Enrollment No. : %d\n", s.roll_no);
       printf( "Student Name : %s\n\n", s.name);
}
```

Output

```
Student Date of Joining (D/M/Y) : 21/9/2014
Student Enrollment No. : 1
Student Name : Vibhuti Singh
```

❑ How to Access a Structure Pointer in C ?

```
#include <stdio.h>

struct Point {
    int x, y;
};

int main()
{
    struct Point str = { 1, 2 };
    struct Point* ptr = &str;
    printf("%d %d", ptr→x, ptr→y);
}
```

❑How to Pass a Structure Pointer in a function?

```
#include <stdio.h>
struct Point {
    int x;
    int y;
};

void printPoint(struct Point *p) {
    printf("Point: (%d, %d)\n", p→x, p→y);
}

int main() {
     struct Point p1 = {10, 20};
     printPoint(&p1);
}
```

❑Exercise problems of structures

Example-1:

```
#include<stdio.h>
struct st
{
    int x;
    static int y;
};

int main()
{
    printf("%d", sizeof(struct st));
    return 0;
}
```

Example-2:

```
#include<stdio.h>
struct st
{
    int x;
    struct st next;
};
int main() {
    struct st temp;
    temp.x = 10;
    temp.next = temp;
    printf("%d", temp.next.x);
}
```

Example-3:

```
union test
{
    int x;
    char arr[8];
    int y;
};

int main()
{
    printf("%d", sizeof(union test));
    return 0;
}
```

❑Exercise problems of structures

Example-4:

```
#include <stdio.h>
struct da1{
     int a;
     int *b;
};
struct da2{
     int a;
     struct da1 *b;
};
int main(){
     int i=5;
     struct da2 p;
     struct da1 q;
     q.b=&i;
     p.b=&q;
     mystery(&p);
     printf("%d %d %d\n",i,p.a,q.a);
}
```

Example-5:

```
#include <stdio.h>
#include <string.h>
int main(){
    struct mystruct{
        char *name;
        unsigned int age;
    };
    struct mystruct st1 = {"Ram", 12};
    printf("%lu %u", strlen(st1.name), st1.age);
}
```

Example-6:

```
#include<stdio.h>
struct Ournode{
    char x, y, z;
};
int main() {
    struct Ournode p={'1', '0', 'a'+2};
    struct Ournode *q=&p;
    printf("%c, %c", *((char*)q+1), *((char*)q+2));
    return 0;
}
```

Example-7:

```
#include <stdio.h>
struct Point {
    int x;
    int y;
};

int main() {
     struct Point p1 = {.x = 5, .y = 10};
     printf("Coordinates of p1: (%d, %d)\n", p1.x, p1.y);
}
```

Example-8: Use of Character array in Structure

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

struct Contact {
    char name[50];
    char phoneNumber[15];
};

int main() {
    struct Contact contacts[3];

    // Populate the contact list
    strcpy(contacts[0].name, "John Doe");
    strcpy(contacts[0].phoneNumber, "555-1234");

    strcpy(contacts[1].name, "Alice Smith");
    strcpy(contacts[1].phoneNumber, "555-5678");

    strcpy(contacts[2].name, "Bob Johnson");
    strcpy(contacts[2].phoneNumber, "555-9876");

    // Display the contacts
    printf("Contacts:\n");
    for (int i = 0; i < 3; i++) {
        printf("Name: %s\n", contacts[i].name);
        printf("Phone Number: %s\n", contacts[i].phoneNumber);
        printf("\n");
    }
}
```

Example-9: C Program to illustrate bit fields in structures

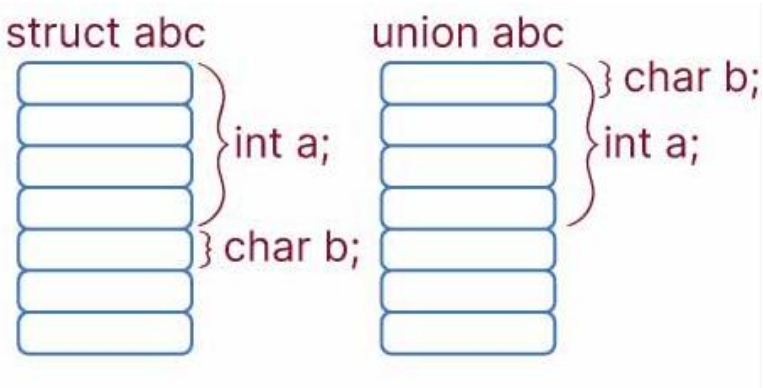
```
#include <stdio.h>
struct str1 {
    int a;
    char c;
};

// structure with bit fields
struct str2 {
    int a : 24; // size of 'a' is 3 bytes = 24 bits
    char c;
};

int main()
{
     printf("Size of Str1: %d\n", sizeof(struct str1));
     printf("Size of Str2: %d", sizeof(struct str2));
      return 0;
}
```

❑ C Unions

In C, union is a user-defined data type 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 point in time.

How to define a union?
We use the union keyword to define unions. Here's an example:

```
union car
{
        char name[50];
        int price;
};
```



H.W. Differentiate between structure and union.

➢ Solve the Exercise Problems from here:

✓ C Unions – GeeksforGeeks
✓ C Union – javatpoint

❑Create union variables

```
union car
{
        char name[50];
        int price;
};

int main()
{
        union car car1, car2, *car3;
        return 0;
}
```

❑Another way of creating union variables is:

```
union car
{
        char name[50];
        int price;
} car1, car2, *car3;
```

❑ Accessing Global Union Members

```
#include <stdio.h>
union Job {
   float salary;
   int workerNo;
}j;

int main() {
   j.salary = 12.3;
   j.workerNo = 100;
   printf("Salary = %.1f\n", j.salary);
   printf("Number of workers = %d", j.workerNo);
}
```

Output

```
Salary = 0.0
Number of workers = 100
```

❑ Accessing Global Union Members

```
union test{
    int x;
    char arr[4];
    int y;
};
int main(){
    union test t;
    t.x = 0;
    t.arr[1] = 'G';
    printf("%s", t.arr);
}
```

❑ Nested Union

```
#include <stdio.h>

struct Employee {
    char name[50];
    int id;
    union {
        float hourlyRate;
        float salary;
    } payment;
};

int main() {
    struct Employee e1;
    snprintf(e1.name, sizeof(e1.name), "Rahul");
    e1.id = 101;
    e1.payment.hourlyRate = 300;
    printf("Employee 1: %s (ID: %d)\n", e1.name, e1.id);
    printf("Hourly Rate: Rs %.2f", e1.payment.hourlyRate);
}
```

Output:

```
Employee 1: Rahul (ID: 101)
Hourly Rate: Rs 300.00
```

❑ Accessing Local Union Members

```
int main(){
    union {
        int i1;
        int i2;
    } myVar = {.i2 = 100};
     printf("%d %d", myVar.i1, myVar.i2);
}
```

❑ Can we store data in multiple union members at the same time?

No. We can only store data in a single member at the same time.

For example, in the following C program, both x and y share the same location. If we change x, we can see the changes being reflected in y.

```
#include <stdio.h>
union test {
    int x, y;
};

int main(){
    union test t;
    t.x = 2;
    printf("After making x = 2:\n x = %d, y = %d\n\n", t.x, t.y);
    t.y = 10;
    printf("After making y = 10:\n x = %d, y = %d\n\n", t.x, t.y);
    return 0;
}

Output:
After making x = 2:
 x = 2, y = 2

After making y = 10:
 x = 10, y = 10
```

❑Size of a union

```
#include <stdio.h>
union Person
{
        char name[32];
        float salary;
        int workerNo;
} p1;

int main()
{
   printf("size of union = %d bytes", sizeof(p1));
}
```

Output: size of union = 32 bytes

❑ Size of a structure

```
#include <stdio.h>
struct Student
{
        char name[32];
        float salary;
        int workerNo;
} s1;

int main()
{
   printf("\nsize of structure = %d bytes", sizeof(s1));
}
```

Output: size of structure = 40 bytes

❑ Enumeration (or enum) in C

An enum is a special type that represents a group of constants (unchangeable values).
To create an enum, use the enum keyword, followed by the name of the enum, and separate the enum items with a comma:
enum_keyword *enum_name* {constant1, constant2, constant3, ... };


```c
#include <stdio.h>
enum day {sunday, monday, tuesday, wednesday, thursday, friday, saturday};

int main(){
    enum day d = thursday;
    printf("The day number stored in d is %d", d);
}
```

❑ Solve the exercise problems from here:

✓ https://www.geeksforgeeks.org/enumeration-enum-c/
✓ https://www.w3schools.com/c/c_enums.php

```c
// demonstrate working of enum in C
#include<stdio.h>

enum year{Jan=1, Feb, Mar, Apr, May, Jun, Jul,
          Aug, Sep, Oct, Nov, Dec};

int main(){
   int i;
   for (i=Jan; i≤Dec; i++)
      printf("%d ", i);
}
```

Output:

```
1 2 3 4 5 6 7 8 9 10 11 12
```

❑ typedef in C

The typedef is a keyword that is used to provide existing data types with a new name.

typedef existing_type new_type;

❑ typedef for Datatype

```
#include <stdio.h>

typedef long long ll;

int main() {
    ll a = 20;
    printf("%lld", a);
}
```

❑ typedef for Structures

```
#include <stdio.h>

typedef struct
{
    int a;
} str1;

typedef struct
{
    int x;
} str2;

int main()
{
    str1 var1 = { 20 };
    str2 var2 = { 314 };

    printf("var1.a = %d\n", var1.a);
    printf("var2.x = %d\n", var2.x);
}
```

- ❑ H.W: differences between the typedef and #define in C
- ❑ Solve the exercise problems from here:
- ✓ C typedef - GeeksforGeeks

❑ Video lectures:

❑ Structure:

C programming Bangla Tutorial 5.210 : Structure | Introduction to Structure
C programming Bangla Tutorial 5.211 : Structure | Local vs Global structure
C programming Bangla Tutorial 5.212 : Structure | Input structure element
C programming Bangla Tutorial 5.213 : Structure | initialize structure variables
C programming Bangla Tutorial 5.214 : Structure | structure comparison
C programming Bangla Tutorial 5.215 : Structure | Array of structure
C programming Bangla Tutorial 5.216 : Structure | Array within structure
C programming Bangla Tutorial 5.217 : Structure | passing structure variable to function

❑ Union:

C programming Bangla Tutorial 5.218 : union | Introduction to union
C programming Bangla Tutorial 5.219 : union | size of union and structure

❑ Enumeration:

C programming Bangla Tutorial 5.220 : enumeration | Introduction to enum

❑ Typedef:

C programming Bangla Tutorial 5.221 : typedef | Introduction to typedef