Practical-9

To study about Structures and its operations

Structure

- A structure is a **collection** of one or more variables, possibly of different types, grouped under a single name.
- It is a **user-defined** data type.
- Arrays allow to define type of variables that can hold several data items of the same kind.
- Similarly, **structure** is another user defined data type available in C that allows to combine data items of different kinds.

Example

- structure allow a group of logically related variables to be treated as one.
- For example, a student can have properties of name, age, gender and marks.
- We could create a character array for name, an integer variable for roll, a character variable for gender, and an integer array for marks.
- But if there are 20 or 100 students, it will be difficult to handle those variables.

Structure Declaration

- We can declare a structure using the struct keyword
- struct structureName {

```
datatype memberVariable1;
datatype memberVariable2; .....
};
```

- Example
- struct student {

```
char name[20];      int roll;
char gender;   int marks[5];
};
```

Structure Declaration

- Example
- struct address {

```
        char name[50];
        char street[100];
        char city[50];
        char state[20];
        int pin;
};
```

How to create variable ?

- When a struct type is declared, no storage or memory is allocated.
- To allocate memory of a given structure type and work with it, we need to create variables.
- Syntax
- `struct structureName structureVariable;`
- Example
- `struct student st1;`
- `struct student st2,st3,st4;`

How to create variable ?

- Example
- struct Person

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

- int main()

```
{
    struct Person person1, person2, p[20];
    return 0;
}
```

```
struct Person
{
    char name[50];
    int id;
    float salary;
} person1, person2, p[20];
```

Keyword typedef

Using the typedef keyword in the structure declaration, we can prevent having to write struct again.

- typedef struct students

```
{
char name[20];
int roll;
char gender;
int marks[5];
} STUDENT;
```

- typedef struct

```
{
    char name[20];
    int roll;
    char gender;
    int marks[5];
} STUDENT;
STUDENT st1,st2,st3,st4;
```

Keyword typedef

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

=

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

Structure

- The use of structure Name is optional.
- C doesn't allow variable initialization inside a structure declaration.
- struct point {

  int x = 0; int y = 0; //error};
- The reason for above error is simple, when a datatype is declared, no memory is allocated for it.
- Memory is allocated only when variables are created.

Initialization Members of a Structure

- `struct structureName = { value1, value2,...};`
- `typedef struct {`

```
        char name[20]; int roll;
        char gender; int marks[5];
            } STUDENT;
void main() {
    STUDENT st1 = { "Alex", 43, 'M', {76, 78, 56, 98, 92}};
    STUDENT st2 = { "Max", 33, 'M', {87, 84, 82, 96, 78}};
    }
```

How to access structure members ?

- There are two types of operators used for accessing members of a structure.

  1. . - Member operator
  2. **->** - Structure pointer operator

- Struct student { int roll_no; char name; }
- Student.roll_no ;
- Student.name;

Example

- printf("Name: %s\n", st1.name);
- printf("Roll: %d\n", st1.roll);
- printf("Gender: %c\n", st1.gender);
- for(i = 0; i < 5; i++)

```
    {
        printf("Marks in %dth subject: %d\n", i, st1.marks[i]);
    }
```

Example

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

struct Books {
    char  title[50];
    char  author[50];
    char  subject[100];
    int   book_id;
};

int main( ) {

    struct Books Book1;          /* Declare Book1 of type Book */
    struct Books Book2;          /* Declare Book2 of type Book */

    /* book 1 specification */
    strcpy( Book1.title, "C Programming");
    strcpy( Book1.author, "Nuha Ali");
    strcpy( Book1.subject, "C Programming Tutorial");
    Book1.book_id = 6495407;

    /* book 2 specification */
    strcpy( Book2.title, "Telecom Billing");
    strcpy( Book2.author, "Zara Ali");
    strcpy( Book2.subject, "Telecom Billing Tutorial");
    Book2.book_id = 6495700;
```

```
   /* print Book1 info */
   printf( "Book 1 title : %s\n", Book1.title);
   printf( "Book 1 author : %s\n", Book1.author);
   printf( "Book 1 subject : %s\n", Book1.subject);
   printf( "Book 1 book_id : %d\n", Book1.book_id);

   /* print Book2 info */
   printf( "Book 2 title : %s\n", Book2.title);
   printf( "Book 2 author : %s\n", Book2.author);
   printf( "Book 2 subject : %s\n", Book2.subject);
   printf( "Book 2 book_id : %d\n", Book2.book_id);

   return 0;
}
```

When the above code is compiled and executed, it produces the following result –

```
Book 1 title : C Programming
Book 1 author : Nuha Ali
Book 1 subject : C Programming Tutorial
Book 1 book_id : 6495407
Book 2 title : Telecom Billing
Book 2 author : Zara Ali
Book 2 subject : Telecom Billing Tutorial
Book 2 book_id : 6495700
```

Practical-9.1

- Write a program using structure to read following data of 3 students:

  [a] Roll no.

  [b] Student name

  [c] Marks in subject-1

  [d] Marks in subject-2

  Compute and print total marks and percentage of all students.

Array of structure

```
struct student
    {
        char name[20];
        double roll;
        char gender;
        int marks[5];
    };

struct student stu[4];
```

```
/* Taking values for the user */

for(int i = 0; i < 4; i++)
    {
        printf("Enter name:\n");
        scanf("%s",&stu[i].name);
        printf("Enter roll:\n");
        scanf("%d",&stu[i].roll);
        printf("Enter gender:\n");
        scanf(" %c",&stu[i].gender);

        for( int j = 0; j < 5; j++)
        {
            printf("Enter marks of %dth subject:\n",j);
            scanf("%d",&stu[i].marks[j]);
        }

        printf("\n-------------------\n\n");
    }
```

Practical-9.1

- struct student
  {
  int rollno,sub1,sub2,total;
  float per;
  char name[20];
  };

- Int main()
  {
  struct student st[3];
  // struct student s1,s2,s3;
  for(i=0;i<3;i++)
  {
  scanf("%d%d%d%s",&st[i].rollno,…
  //read values for rollno ,sub1,sub2 and name
  //calculate total marks
  //percentage =total*100/200
  }

Practical-9.2

Write a structure called Game that will describe the following information:

[a] player name

[b] team name

[c] maximum score

Using Game, declare an array Player with 3 elements and

write a program to read the information about all the 3 players and print a team-wise containing names of players with their maximum scores.

Practical-9.2

- struct game
  {
    char p_name[3][20];
    int max_score[3];
    char team[20];
  } info[3];

```
for (i=0 to i<3)
    {
     //Read team name info[i].name
for(j=0 to j<3)
     {
      //read player name and maximum score info[i].p_name[j]

      }
```

Practical-9.3

- There are 3 students in the class.
- The student's records as shown below:
  Roll no [integer] , Marks of sub-1 [integer] ,Marks of sub-2 [integer] ,Marks of sub-3 [integer]
- Student fails if he gets less than 50 marks in any subject. [maximum marks are 100]
  Passing class is given below:
  First class :marks >=70%
  Second class :marks < 70% and >=55%
  Pass class : marks < 55% and >=50%
- Print the result of each student with heading Roll No. Marks1 Marks2 Marks3
- Also print....
  * Total number of students in each class
  * Total number of student failing
  * Number of student failing in more than one subject
  * Print the report on sorted order of student name

Practical-9.3

- struct student { int rollno,sub1,sub2,sub3; };
- Main(){

  Struct student s[3];

  For(i=0 to i<3)

  {

  //read student rollno,sub1,sub2 and sub3

  //count no of fails who get < 50 in any subject (logical OR)

  //count student fail in more than one(logical AND)

  }

  For loop { print rollno,marks1,marks2,marks3}

Practical-9.4

- Write a program to enter and display the employee data using nested structure.
- Take three structures department , address, employee .
- Take department and address as a member of employee .

Nesting of Structure

- Nesting a structure means having one or more structure variables inside another structure

```
struct birth
    {
        int date;
        int month;
        int year;
    };
```

```
struct student
    {
        char name[20];
        int roll;
        char gender;
        int marks[5];
        struct birth birthday;
    };

void main(){
 struct student stu1;
```

Nesting of Structure

- structure to be nested has to be declared first
- (. Dot) is used to access the members contained within the inner structure as well as other members.
- Example

```
 /* Example */
stu1.birthday.date
stu1.birthday.month
stu1.birthday.year
stu1.name
```

Practical-9.4

- struct employee

```
    {
        char name[20];  //e1.name
        struct dept
            {
                char dpt[20]; //e1[i].d1.dpt
            }  d1;
        struct address
            {
                int no, char street[30]; char city[20];  //e1.ad1.city
                 char state[15];
            } ad1;
    };
```

Practical-9.4

- int main()

```
{
struct employee e[3];
int i;
for(i=0;i<3;i++)
{
  //read values e[i].name…….
  //e[i].ad1.street …….
}
For loop
{ //print values }
```