CS11001/CS11002
Programming and Data Structures
(PDS) (Theory: 3-0-0)

Class Teacher: **Pralay Mitra**
                          Jayanta Mukhopadhyay
                          Soumya K Ghosh

Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur

Structures

What is a Structure?

- It is a convenient tool for handling a group of logically related data items.
  - Student name, roll number, and marks
  - Real part and complex part of a complex number

- This is our first look at a non-trivial data structure.
  - Helps in organizing complex data in a more meaningful way.

- The individual structure elements are called *members*.

Defining a Structure

- The composition of a structure may be defined as:

```
struct  tag  {
            member 1;
            member 2;
            :
            member m;
        };
```

  - struct is the required keyword.
  - tag is the name of the structure.
  - member 1, member 2, … are individual member declarations.

Defining a Structure

- The individual members can be ordinary variables, pointers, arrays, or other structures.
  - The member names within a particular structure must be distinct from one another.

  - A member name can be the same as the name of a variable defined outside of the structure.

- Once a structure has been defined, individual structure-type variables can be declared as:

  struct tag variable_1, variable_2, …, variable_n;

Example Structure

- A structure definition:

  ```
  struct  student  {
                      char  name[30];
                      int  roll_number;
                      int  total_marks;
                      char  dob[10];
                  };
  ```

- Defining structure variables:

  struct student a1, a2, a3;

  A new data-type

A Compact Form

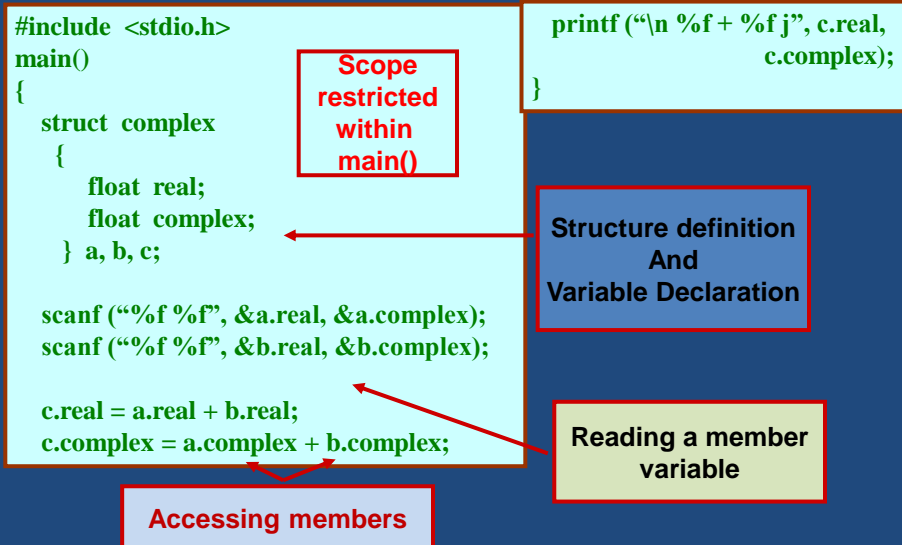- It is possible to combine the declaration of the structure with that of the structure variables:

```
struct  tag  {
                member 1;
                member 2;
                :
                member m;
            } variable_1, variable_2,…, variable_n;
```

- In this form, "tag" is optional.

Processing a Structure

- The members of a structure are processed individually, as separate entities.
- A structure member can be accessed by writing

  variable.member

  where variable refers to the name of a structure-type variable, and member refers to the name of a member within the structure.
- Examples:
  - a1.name, a2.name, a1.roll_number, a3.dob;

Example: Complex number addition

```
#include <stdio.h>
main()
{
   struct complex
    {
       float real;
       float complex;
   } a, b, c;

   scanf ("%f %f", &a.real, &a.complex);
   scanf ("%f %f", &b.real, &b.complex);

   c.real = a.real + b.real;
   c.complex = a.complex + b.complex;
```

Scope restricted within main()

```
   printf ("\n %f + %f j", c.real,
                            c.complex);
}
```

Structure definition And Variable Declaration

Reading a member variable

Accessing members

Arrays of Structures

• Once a structure has been defined, we can declare an array of structures.

   struct student class[50];

   – The individual members can be accessed as:
     • class[i].name
     • class[5].roll_number

Arrays within Structures

- A structure member can be an array:

```
struct  student  {
                        char  name[30];
                        int  roll_number;
                        int  marks[5];
                        char  dob[10];

                } a1, a2, a3;
```

- The array element within the structure can be accessed as:
  a1.marks[2]

Structure within Structures

- A structure member can be another structure:

```
struct college_info  {
    int college_id;
    char college_name[50];
};

struct stud_detail {
    int class;
    char name[20];
    float percentage;
    struct college_info college;
} stu_data;
```

Defining data type: using *typedef*

* One may define a structure data-type with a single name.

* General syntax:
  typedef struct {
          member-variable1;
          member-variable2;
          ....
          member-variableN;
        } **tag**;

* **tag** is the name of the new data-type.

typedef : An example

typedef struct{
    float real;
    float imag;
    } COMPLEX;

COMPLEX a,b,c;

Structure Initialization

- Structure variables may be initialized following similar rules of an array. The values are provided within the second braces separated by commas.

- An example:

    COMPLEX a={1.0,2.0}, b={-3.0,4.0};

    ⬇

    a.real=1.0; a.imag=2.0;
    b.real=-3.0; b.imag=4.0;

Structure Initialization

- **Homework:**

1. How do you initialize nested structures?
2. How do you initialize arrays within structures?

Parameter Passing in a Function

- Structure variables could be passed as parameters like any other variable. Only the values will be copied during function invocation.

```
void swap(COMPLEX a, COMPLEX b)
{
   COMPLEX tmp;

   tmp=a;
   a=b;
   b=tmp;
}
```

An example program

```
#include <stdio.h>

typedef struct{
       float real;
       float imag;
       } COMPLEX;

void swap(COMPLEX a, COMPLEX b)
 {
   COMPLEX tmp;

   tmp=a;
   a=b;
   b=tmp;
 }
```

Example program: contd.

```c
void print(_COMPLEX a)
{
 printf("(%f , %f) \n",a.real,a.imag);
}

main()
{
 COMPLEX x={4.0,5.0},y={10.0,15.0};

 print(x); print(y);
 swap(x,y);
 print(x); print(y);
}
```

Returning structures

• It is also possible to return structure values from a function. The return data type of the function should be as same as the data type of the structure itself.

```c
COMPLEX add (COMPLEX a, COMPLEX b)
{
  COMPLEX tmp;

  tmp.real = a.real+b.real;
  tmp.imag = a.imag+b.imag;

  return(tmp);
}
```

> Direct arithmetic operations are not possible with Structure variables.

Example-1

Define a structure type **student** to store the **name, roll,** and **total-marks** of any student.

Write a program to read this information (from keyboard) for one student and print the same on the screen.

Example-1

CODE:

```c
#include <stdio.h>

//structure definition
struct student {
    char name[50];
    int roll;
    float marks;
};
```

```c
//main function
int main(){
    struct student s; //declaring structure variable

    //reading information from keyboard
    printf("Enter information of students:\n");
    printf("Enter name: ");
    scanf("%s",s.name);
    printf("Enter roll number: ");
    scanf("%d",&s.roll);
    printf("Enter marks: ");
    scanf("%f",&s.marks);

    //displaying information on screen
    printf("\nDisplaying Information\n");
    printf("Name: %s\n",s.name);
    printf("Roll: %d\n",s.roll);
    printf("Marks: %.2f\n",s.marks);
    return 0;
}
```

Example-2

Use the same *student* structure as described in the Example-1. Define a function to check whether two students are same or not.

- It returns 1, if the *student1* and *student2* are same
- It returns 0, if the *student1* and *student2* are NOT same

Example-3

Write a C program to perform *addition* and *multiplication* of any two complex numbers, taken as input from the terminal.

Example 4

Problem Statement:

Write a program which reads two timestamps (hour, minute, second separately in 23:59:59 format) and prints the time difference between them.

Example 5

Problem Statement:

Write a recursive C function to check whether a number is prime or not.

Sample output:

Return 1 if it is prime, 0 otherwise.

Example 6

Problem Statement:

Decimal number to binary conversion using recursion

Exercises

- **Exercise 1**: Find the LCM of two numbers using recursion.

Sample output
Enter any two positive integers 36 48
LCM of two integers is 144

- **Exercise 2**: Find the sum of the digits of a number using recursion

Sample output
Enter the number: 12345
Sum of digits in 12345 is 15

Exercise 3

Define a structure data type named *date* containing three integer members: *day, month,* and *year*. Write a program to perform the following tasks:

- To *read data* into structure members by a function
- To *print the date* in the format: July 11, 2013
- To *validate the date* by another function

> **Example Output:**
>
> Enter the day, month, and year: 10 9 2016
> The date is: September 10, 2016
> It is a VALID date
>
> Enter the day, month, and year: 31 4 2015
> The date is: April 31, 2015
> It is an INVALID date

Exercise 4

Use the same *date* structure as defined in Exercise 1 to store date of birth and current date. Calculate the age of the person.

Exercise 5

Define a structure called **cricket** that will describe the following information:

player-name
team-name
batting-average

Declare an array **player** of type **cricket,** with 50 elements. Write a program to read the information about all the 50 players and print a *team-wise list* containing names of players sorted (non-increasing) by their batting average.

Example Output:

```
TEAM: INDIA
-----------------------------------
Sachin Tendulkar    44.83
Sourav Ganguly      41.02
…
…
TEAM: NEW ZEALAND
-----------------------------------
Nathan Astle        34.92
Stephen Fleming     32.40
```