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.

Contd.

- **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

- **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.

Example

```
struct  student  {
                        char  name[30];
                        int  roll_number;
                        int  total_marks;
                        char  dob[10];
                    }  a1, a2, a3;
```

```
struct              {
                        char  name[30];
                        int  roll_number;
                        int  total_marks;
                        char  dob[10];
                    }  a1, a2, a3;
```

Equivalent declarations

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;
```

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

Scope restricted within main()

Structure definition
And
Variable Declaration

Reading a member variable

Accessing members

Comparison of Structure Variables

- Unlike arrays, group operations can be performed with structure variables.
  - A structure variable can be directly assigned to another structure variable of the same type.

    ```
    a1 = a2;
    ```

    - All the individual members get assigned.
  - Two structure variables can be compared for equality or inequality.

    ```
    if  (a1 = = a2)  …….
    ```

    - Compare all members and return 1 if they are equal; 0 otherwise.

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]
```

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;
```

Parameter Passing in a Function

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

```
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.

```
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.

```
_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.