CS - 114 : Computer Workshop

Prof. Chamakuri Nagaiah
Mahindra-École Centrale, Hyderabad
nagaiah.chamakuri@mechyd.ac.in

What is a Structure?

- Arrays allow to define type of variables that can hold several data items of the same kind.

What is a Structure?

- 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.
- Examples :
  - Student name, roll number, and marks.
  - Real part and complex part of a complex number.
  - Keep track of your books in a library.
- It helps in organizing complex data in a more meaningful way.
- The individual structure elements are called members.

Defining a Structure

- To define a structure, you must use the **struct** statement.
- The composition of a structure may be defined as:

```
struct [tag] {

 member 1;
 member 2;
 ...
 member n;
} [one or more structure variables];
```

  - struct is the required keyword.
  - tag is the name of the structure (it is optional).
  - member 1, member 2, ... are individual member declarations.
- **Note**: Don't forget the semicolon }; in the ending line.

Defining a Structure: 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, the individual structure-type variables can be declared as:

  ```
  struct tag var_1, var_2, ..., var_m;
  ```

Example

- A structure definition:

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

- Defining structure variables (new data-type):
  struct student a1, a2, a3;

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;
}  var_1, var_2,..., var_n;
```

- In this form, “tag” is optional.

Equivalent Declarations

-
  ```
  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;
  ```

Accessing Structure Members

- The members of a structure are processed individually, as separate entities.
- To access any member of a structure, we use the member access operator (.)
- A structure member can be accessed by writing

Accessing Structure Members

- The members of a structure are processed individually, as separate entities.
- To access any member of a structure, we use the member access operator (.)
- 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
- Structure pointer operator(->) (will be discussed in structure and pointers class)

Example 1:

```
#include <stdio.h>
#include <string.h>
struct Books {
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
};
void 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, "Dennis Ritche");
   strcpy( Book1.subject, "C Programming Tutorial");
   Book1.book_id = 6495407;
```

Example 1:

```
  /* book 2 specification */
   strcpy( Book2.title, "Numerical Analysis");
   strcpy( Book2.author, "David Kincaid");
   strcpy( Book2.subject, "Numerical Methods");
   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);
}
```

Example 2:

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

  scanf ("%f %f", &a.real, &a.imag);
  scanf ("%f %f", &b.real, &b.imag);
  c.real = a.real + b.real;
  c.imag = a.imag + b.imag;
  printf ("\n %f + %f j", c.real, c.imag);
 }
```

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)...... ???

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.
- Example

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

COMPLEX a, b, c;

Defining data type: using **typedef**

- **typedef** keyword is used in creating a type **comp** (which is of type as **struct complex**).

```
typedef struct COMPLEX
{
  int imag;
  float real;
} comp;

int main()
{
  comp comp1, comp2;
}
```

- Two structure variables *comp1 and comp2* are created by this **COMPLEX** type.

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};
```

Equivalent to

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

Parameter Passing in a Function

- Structure variables can be passed as parameters like any other variables. 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

```
 #include <stdio.h>

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

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

An Example

```
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);
 }
```

Output

```
(4.000000, 5.000000)
(10.000000, 15.000000)
(4.000000, 5.000000)
(10.000000, 15.000000)
```

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.

Passing structures to a function

- There are mainly two ways to pass structures to a function:
  1. Passing by value (passing actual value as argument)
  2. Passing by reference (passing address of an argument)

Passing structure by value : An example

```
#include <stdio.h>

struct student
{
    char name[50];
    int roll;
};

void display(struct student stu);
// function prototype should be below to the structure
//declaration otherwise compiler shows error
```

Passing structure by value: An example

```
int main()
{
    struct student stud;
    printf("Enter student’s name: ");
    scanf("%s", &stud.name);
    printf("Enter roll number:");
    scanf("%d", &stud.roll);
    display(stud);   // passing structure variable stud as a
    return 0;
}
void display(struct student stu)
{
  printf("Output\nName: %s",stu.name);
  printf("\nRoll: %d",stu.roll);
}
```

Passing structure by reference : An example

```
main()
{
    struct dist dist1, dist2, dist3;
    ....
    add(dist1, dist2, &dist3); ...
}
void add(struct dist d1,struct dist d2, struct dist *d3) {
     //Adding distances d1 and d2 and storing it in d3
     d3->feet = d1.feet + d2.feet;
     d3->inch = d1.inch + d2.inch;

     if (d3->inch >= 12) { /* if inch is greater or equal */
         d3->inch -= 12;   /* to 12, converting it to feet*/
         ++d3->feet;
    }
}
```