Structures and Unions

Unit 8

INTRODUCTION

In the previous unit 7 we have studied about C functions and their declarations, definitions, initializations. Also we have learned importance of local and global variables and their scope and life time. In this unit we will describe about another important topic in C language like structures. We have seen that, arrays can be used to represent a group of same data items. In the some cases, if we want to represent a collection of data items of different types using a single name, then we cannot use an array. For this reason, C language supports an assembled data type known as ***structure.*** In this unit, we will describe in detail how a structure is defined, initialized, and how structure works in a program.

Timeframe How long ?	We expect that this unit will take maximum 5 hours to complete.

Unit Structure	
Lesson- 1 : Introduction to C Structures Lesson- 2 : Structure Initialization	

Lesson-1 Introduction to C Structures

Learning Outcomes

Outcomes

Upon completion of this lesson you will be able to

- Understand basic idea about C structures.
- Learn necessity of structures in C program.
- Understand how a structures are declared and defined in program.

	Keywords	**Structure, Definition, Declaration, Member**

STRUCTURES IN C

We have studied that, an array is one kind of data structures that can be used to demonstrate a group of data items that belongs to the same type like *int, float, char, double* etc. But if we want to represent or demonstrate a collection of different types of data items using a single name, then we cannot use an array. For this reason, in C language supports a user defined data type known as ***structure.*** Actually, ***structure*** is user defined data type available in C that allows combining different kinds of data items using a single name.

On the other word, a structure is a well-situated tool for handling a collection of logically related data items. It is used to represent a record or a set of attributes, such as *student_name, roll_number, obtained_mark and cgpa* etc. It is a powerful concept that we can often need to use in our program design.

STRUCTURE DEFINITION/DECLARATION

A structure is a collection of deferent type variables under a single name. Structures assist to arrange complex data in a more meaningful. A structure declaration or definition generates a format that can be used to declare structure variables. To define a structure, we must use the keyword ***struct*** statement. The keyword ***struct*** statement defines a new data type, with more than one member. It declares a structure to hold the details of member fields. The general format of the structure definition is as follows:

```
struct    struct_tag_name
{
   data_type   member-1;
   data_type   member-2;
   data_type   member-3;
   …………   ….  ….
   ………..    ….  ….
   data_type   member-n;
} one or more structure variables;
```

$\underbrace{\text{member-1 … member-n}} \longrightarrow$ Structure fields or elements or member

Here, the ***struct_tag_name*** is called structure tag name and it is optional and each member definition is a normal variable definition, such as *int a, float b, char c or char c[]* etc; or any other valid variable definition. The tag name may be used subsequently to declare variables that have the tag structure. The fields are called structure elements or members. Each member may belong to different type of data. At the end of the structure's definition, before the final semicolon, you can specify one or more structure variables. Let us use an example to illustrate the process of structure definition and the creation of structure variables. Consider a book data base consisting of book title, author name, subject, number of pages, price and book id. We can define a structure to hold this information as follows:

```
struct Books
{
   char  title[50];
   char  author[50];
   char  subject[100];
   int pages;
   float price;
   int   book_id;
}book; ←———————— Structure variable
```

We can declare structure variables using structure tag name anywhere in the program. For instance, the statement

struct Books book, book1, book2, book3;

declares book, book1, book2 and book3 as variables of type ***struct Books***.

Note it!

A structure is a collection of deferent type variables under a single name. Structures assist to arrange complex data in a more meaningful.

ACCESSING STRUCTURE MEMBERS

If we want to access any member or field of a structure, we may use the **"member access operator (.)"** also called dot operator or period operator. The member access operator **(.)** is coded as a period between the structure variable name and the structure member that we wish to access.

For example

```
book.title
book.price
book. book_id
```

are the variables representing the title, price and book id of book and can be treated like any other ordinary variable. Here is how we should assign values to the members of **book:**

```
strcpy(book.title, " C Programming");
strcpy(book.author, " Amran Hossain");
book.pages=300;
book.price=500.00;
book.book_id=34521;
```

We can also use scanf() function to give the values through the keyword

```
scanf("%s\n",book.title);
scanf("%s\n",book.author);
scanf("%d\n",book.pages);
scanf("%f\n",book.price);
```

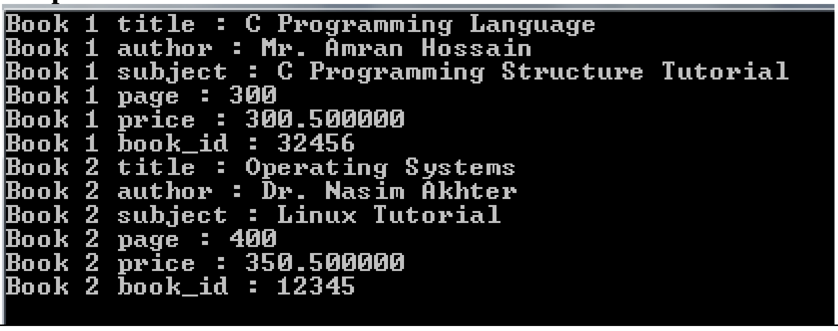
scanf(“%d\n”,book.book_id);
are valid input statements.

Program 8.1.1: Write a C program that defining and assigning value to structure members.

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

struct Books
{
   char  title[100];
   char  author[100];
   char  subject[100];
   int pages;
   float price;
   int   book_id;
} Book1,Book2; /* Declare Book1 and Book2 of type Books */

void main( )
{
  clrscr();

  /* Book 1 specifications */
  strcpy( Book1.title, "C Programming Language");
  strcpy( Book1.author, "Mr. Amran Hossain");
  strcpy( Book1.subject, "C Programming Structure Tutorial");
  Book1.pages=300;
  Book1.price=300.50;
  Book1.book_id = 32456;
  /* Book 2 specifications */
  strcpy( Book2.title, "Operating Systems");
  strcpy( Book2.author, "Dr. Nasim Akhter");
  strcpy( Book2.subject, "Linux Tutorial");
  Book2.pages=400;
  Book2.price=350.50;
  Book2.book_id = 12345;
  /* print Book1 information */
  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 page : %d\n", Book1.pages);
  printf( "Book 1 price : %f\n", Book1.price);
  printf( "Book 1 book_id : %d\n", Book1.book_id);
  /* print Book2 information */
  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 page : %d\n", Book2.pages);
  printf( "Book 2 price : %f\n", Book2.price);
  printf( "Book 2 book_id : %d\n", Book2.book_id);

  getch();
}
```

……………………………………………………………………………….

Output



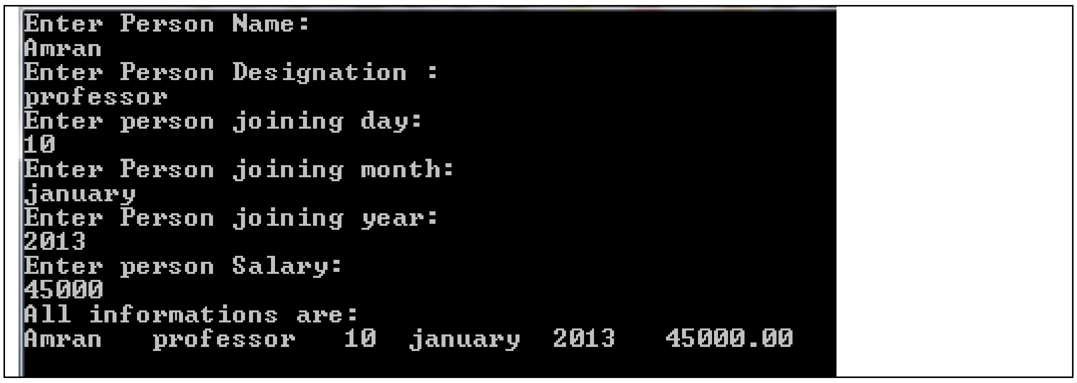
```
Book 1 title : C Programming Language
Book 1 author : Mr. Amran Hossain
Book 1 subject : C Programming Structure Tutorial
Book 1 page : 300
Book 1 price : 300.500000
Book 1 book_id : 32456
Book 2 title : Operating Systems
Book 2 author : Dr. Nasim Akhter
Book 2 subject : Linux Tutorial
Book 2 page : 400
Book 2 price : 350.500000
Book 2 book_id : 12345
```

Program 8.1.2: Define a structure type, struct personal that would contain person name, person designation, date of joining and salary. Using this structure, write a program to read this information for one person from the keyboard and print the same on the screen.

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

struct personal
{
  char person_name[50];
  char person_desig[30];
  int day;
  char month[12];
  int year;
  float salary;
};
void main()
{
  clrscr();
  struct personal person;
  printf("Enter Person Name:\n");
  scanf("%s",person.person_name);
  printf("Enter Person Designation :\n");
  scanf("%s",person.person_desig);
  printf("Enter person joining day:\n");
  scanf("%d",&person.day);
  printf("Enter Person joining month:\n");
  scanf("%s",person.month);
  printf("Enter Person joining year:\n");
  scanf("%d",&person.year);
  printf("Enter person Salary:\n");
  scanf("%f",&person.salary);
   printf("All informations are:\n");
  printf("%s   %s   %d  %s  %d   %.2f\n",person.person_name,
  person.person_desig,person.day,person.month,person.year,person.salary);
  getch();
}
```

……………………………………………………………………………………

Output:



```
Enter Person Name:
Amran
Enter Person Designation :
professor
Enter person joining day:
10
Enter Person joining month:
january
Enter Person joining year:
2013
Enter person Salary:
45000
All informations are:
Amran    professor    10  january  2013    45000.00
```

Note it!

If we want to access any member or field of a structure, we may use the "member access operator (.)" also called dot operator or period operator.

Study skills

1. Describe what is wrong in the following structure declaration:

```
struct
{
      int number;
      float price;
}
void main()
{
      ……………
}
```

2. Find the error(s) from the following code segments:

```
struct examtest
   {
        char name[];
         int year, day;
        double cgpa;
   }
void main()
   {
        struct examtest;
         printf(" Input Values: \n");
         scanf("%d  %d  %d  %ld", name, year, day, cgpa);
          ………………………
   }
```

3. Describe what is wrong in the following structure declaration:

```
struct
{
      int number;
      float price;
}
void main()
{
      ……………
 }
```

Summary Summary	
In this lesson we have ▪ Learned about C structures definition and declaration. ▪ Learned initialization procedure of structure. ▪ Also understood how structure variables are compared.	

ASSIGNMENT

Assignment

1. Write a program to determine the greatest common divisor (GCD) and least common multiple (LCM) of two integer number using structure.
   …..…………………………………………………………………………….…..
   ………………………………………………………………………………….
2. Define a structure that can describe a hotel. It should have members that include the hotel name, address, grade, room charge, room category and number of rooms.
   ……………………………………………………………………………..
   ……………………………………………………………………………..
3. Define a structure called company that will describe the following information:
   Company Name
   Company location
   Total employees
   Salary status
   Bonus system

Assessment

Assessment

Multiple Choice Questions (MCQ)

1. A structure is a well-situated tool for handling—

   a) A collection of logically related data items b) A collection of physical related data items
   c) Single logically related data items d) None of these.
2. To define a structure, we must use the keyword—

   a) *Structure* statement b) *struct* statement c) *Structure tag* statement d) None of these
3. If we want to access any member of a structure, we may use the

   a) (&) operator b) (%) operator c) (.) operator d) (| |) operator
4. Which of the following is correct?

   a) structure student1=(100,20,400.50 , “XXX”);
   b) structure student1={100,20,400.50 , “XXX”};
   c) struct student1= “100,20,400.50 , “XXX””;
   d) struct student1= {100,20,400.50 , “XXX”};

Exercises

1. What is structure? Explain structure declaration procedure with an example.
2. What do you mean by structure initialization? Explain with proper example.
3. How does a structure differ from an array?

Lesson-2 Structures Initialization

Learning Outcomes

Outcomes

Upon completion of this lesson you will be able to

- Explain procedure of structure initialization.
- Understand the comparison of structure variables.

Keywords **Structure, Initialization, Comparison, Variable**

STRUCTURE INITIALIZATION

Similar to any other data type, a structure variable can be initialized. When initializing an object or member of structure, it must be a non-empty, brace-enclosed, comma-separated list of initializers for the members. Consider the following example for structure initialization as follows:

```
struct student_record
 {
    int marks;
    int age;
    float cgpa;
  }student={70,24,3.20};
```

Here, the above initialization procedure, assigns the value 70 to ***student.marks,*** 24 to ***student.age*** and 3.20 to ***student.cgpa*** member variables respectively. There is one-to-one correspondence between the members and their initializing values. Various processes are possible in initializing a structure. The following statements initialize three structure variables. Here, it is necessary to use a structure tag name.

Consider the following initialization example:

```
struct student_record
  {
     int marks;
     int age;
     float cgpa;
   };
struct student_record student1={80,20,4.00};
struct student_record student2={60,22,3.50};
```

Another process is to initialize a structure variable inside the main function as follows:

```
struct student_record
 {
    int marks;
    int age;
   float cgpa;
  } student1={80,20,4.00};
void main()
```

```
{
   struct student_record student2={60,22,3.50};
    ………………………..
    ………………………..
 }
```

COMPARISON OF STRUCTURE VARIABLES

Two variables of the same structure type can be compared the same way as normal variables. If **student1** and **student2** belong to the same structure, then the following operations are possible:

Operation	**Description**
student1= student2	Assign **student1** values to **student1**
student1= =student2	Compare all members of **student1** and **student2**. If they are equal then return 1, otherwise return 0.
student1 != student2	If all the members of **student1 and student2** are not equal then return 1 otherwise return 0.

Figure 8.2.1: Comparison of structure variables

Program 8.2.1: Write a C program to illustrate the comparison of structure variables.

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

struct student_record
{
   char student_name[30];
   int marks;
   int age;
   float cgpa;
};
void main()
{
     clrscr();
      int record;
      struct student_record student1={"Amran",78,24,3.89};
      struct student_record student2={"Mamun",70,30,3.50};
      struct student_record student3;

     student3=student2;
    record=((student3.marks= =student2.marks)&&(student3.cgpa= =student2.cgpa))?1:0;

    if(record = =1)
     {
       printf("\n Student2 and Student3 marks and cgpa are same!!\n\n\n ");
       printf("%s    %d    %d    %f", student3.student_name, student3.marks,
       student3.age,student3.cgpa);
     }
    else
     {
         printf("\n Student 2 and student 3 are different\n\n");
     }
    getch();
}
…………………………………………………………………………………………….…………..
```

Output

```
Student2 and Student3 marks and cgpa are same!!

Mamun    70    30    3.500000
```

Note it!

Two variables of the same structure type can be compared the same way as normal variables.

Activity

1. Mention the significance of using structure in C language by your own idea.
   ……………………………………………………………………………….……
   .………………………………………………………………………………………

Study skills

1. Describe what is wrong in the following structure declaration:
```
 struct test_rec
   {
        int val1;
        int val2;
        float price;
      }test={70,2.4,320};
```
2. Find the error(s) from the following code segments:
```
 struct product
     {
             int pro_id;
             char pro_name;
             float pro_price;
             int quantity;
       } prolist={T80,20,47.50,5.00};
 void main()
     {
        struct prolist2={600, 'B',22,3.50};
             ………………………..
             ………………………..
      }
```

Summary Summary	
In this lesson we have ▪ Learned initialization procedure of structure. ▪ Also understood how structure variables are compared.	

ASSIGNMENT

Assignment

1. Define a structure called company that will describe the following information and initialize all information's using structure initialization procedure:
   Company Name
   Company location
   Total employees
   Salary status
   Bonus system

Assessment

Assessment

Multiple Choice Questions (MCQ)

1. To define a structure, we must use the keyword—
   a) Structure statement
   b) struct statement
   c) Structure tag statement
   d) None of these
2. Which of the following is correct?
   a) structure student1=(100,20,400.50 , "XXX");
   b) structure student1={100,20,400.50 , "XXX"};
   c) struct student1= "100,20,400.50 , "XXX"";
   d) struct student1= {100,20,400.50 , "XXX"};
3. Two variables of the same structure type can be compared the-
   a) Same way as formal variables of a function
   b) Same way as normal variables
   c) Same way as array initialization
   d) None of these
4. When initializing an object of structure, it must be-
   a) Non-empty, brace-enclosed, comma-separated
   b) Empty, brace-enclosed
   c) Only non-empty and brace-enclosed
   d) Only comma-separated

Exercises

1. What do you mean by structure initialization? Explain with proper example.
2. How does a structure differ from an array?
3. Mention and describe the various comparisons of structure variables with an example.