Introduction

Structure : Mechanism for packing data of different types

A structure is a convenient tool for handling a group of ***logically related*** data items.

Ex :	time	: seconds, hours, minutes;
	customer	: name, telephone, city, category
	book	: author, title, price, year

Defining a Structure

Syntax :

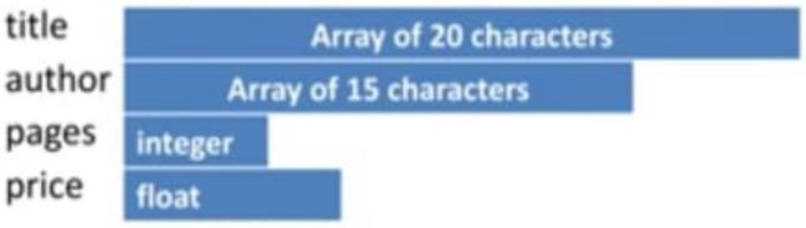
```
struct  tag_name
{
        data-type member1;
        data-type member2;
        ---------- ----------
        ---------- ----------
};
```

```
Ex : struct book
       {
         char title[20];
         char author[15];
         int   pages;
         float  price;
       };
```

Continued ..

```
struct book
   {
   char title[20];
   char author[15];
   int   pages;
   float  price;
   };
```

struct : keyword to define structure
book : is the name of structure; structure-tag
title, **author**, **pages** and **price** are the structure members or elements.
The above definition describes a following format to represent information.



Some observations on syntax of structure

- The definition is terminated with a semicolon.
- Each member is declared independently.
- The tag-name can be used to declare variable of its type.

Declaring Structure Variable

Syntax:

struct tag-name var*1*, var*2*,….var*n*;

A variable of structure can be defined after defining structure as shown in syntax, it has

- Keyword struct
- Structure tag name
- List of variables separeted with commas
- A terminating semicolon.

Ex : struct book book1, book2, book3;

book1, book2 and book 3 are the variables of type struct book.

The complete declaration looks like :

```
struct book
{
      char title[20];
      char author[15];
      int   pages;
      float  price;
};
struct   book  book1, book2, book3;
```

Note : structure members occupy memory only after associating with structure variables.

Combining structure definition and structure variable declaration

Use of tag name is optional to declare variable.

```
struct tag-name
{
  . . . . . . .
  . . . . . . .
} var1,  var2,  var3 ;
```

```
Ex :    struct book
         {
           char title[20];
           char author[15];
           int   pages;
           float  price;
         } book1, book2, book3 ;
```

Accessing Structure Members

Dot operator (•) or period operator is used to access structure member

By linking structure member with structure variable.

```
Ex:     book1.price

                    // refers to the member price linked with
                       variable book1.
```

this way it can be accessed in initialization, processing, input functions and output functions.

// C program to demonstrate structure (to read name, salary, date of joining for employees)

```
struct employee
  {
      char name[20];
      int dd;
      int mm;
      int yy;
      float sal;
  };
main()
{
  struct employee e1;
  printf("\n Enter name, date(dd mm yy) and salary");
  scanf("%s%d%d%d%f",e1.name,&e1.dd,&e1.mm,&e1.yy,&e1.sal);
  printf("\n Name : %s",e1.name);
  printf("\n Date of joining : %d-%d-%d",
  e1.day,e1.month,e1.year);
  printf("\n Salary : %f",e1.salary);
}
```

Structure Initialization

Structure variables can be initialized at compile time in following way

```
main()
   {
       struct st_records
       {
              int weight;
              float height;
       };
       struct st_records s1={ 60,175.5 };
       struct st_records s2={ 53,170.60 };
       . . . .
       . . . .
}
```

Note : Initialization of individual members inside structure is not allowed in C.

Compile-time initilzation of a structure variable must have following elements

- Keyword struct
- Structure tag name
- Name of the variable to be declared
- The assignment operator
- A set of values seperated by commas
- Terminating semicolon

Copying and Comparing structure variables

Any two variables of same structure type can be copied like ordinary variables.

Ex : If person1 and person2 belong to same structure, then

```
person1=person2;  // copies all values of person2 to person1
person2=person1;  // both will have same values.
```

Comparison is not possible

```
person1= = person2; // invalid
person2 != person1;  // invalid
```

```
// C program to copy and compare structure variables.
struct class
   {
       int rollno;
       char name[20];
       float avg;
   };
main()
   {
       int x;
       struct class s1={101,"Amon",53.55};
       struct class s2={102,"Seema",73.81};
       struct class s3;
              s3=s2;
   x=((s3.rollno==s2.rollno) && (s3.marks==s2.marks) &&
   (strcmp(s2.name,s3.name)==0))?1:0;
       if(x==1)
       {      printf("\ student2 and student3 are same");
              printf("%d %s %f",s3.rollno,s3.name,s3.avg);
       }
else
       printf("\n student2 and student3 are different");
   }
```

Operations on individual members

using dot operator, structure members can be used in expressions along with operators.

Ex :

```
if(s1.rollno==102)
      s1.avg + = 5.00;
```

to increment

```
      s1.rollno++;
```

Arrays of Structure

A structure variable can be declared as an array.

Ex: `struct class s[10];`

this defines array called s, which consists 10 elements, and each of 10 has all elements of struct class.

initialising:

```
struct class s[3]={{1,"Rahul",52.6}, {2,"Jyoti",46.4}, {3, "Abdal",72.5}}
```

This initializes each member in this way :

```
s[0].rollno=1; s[0].name="Rahul"…..
s[1].rollno=2;
s[2].rollno=3;
```

// c program to read roll no and marks in 3 subjects of n students. Print all data along with average marks.

```
struct students
{   int rollno;
    int m1;
    int m2;
    int m3;
    float avg;
};
main()
{
    int i,n;
    struct students s[25];
    printf("\n Enter number of students :");
    scanf("%d",&n);
```

```
for(i=0;i<n;i++)
{
    printf("\n Enter data for student %d",i+1);
    printf("\n Enter roll number and marks ");
    scanf("%d%d%d%d",
          &s[i].rollno,&s[i].m1,&s[i].m2,&s[i].m3);
    s[i].avg=(s[i].m1+s[i].m2+s[i].m3)/3.0;
}
printf("\n Data entered :");
printf("\n Roll no   M1 M2 M3  Average");
for(i=0;i<n;i++)
{
    printf("\n %d       %d %d %d  %f",
           s[i].rollno,s[i].m1,s[i].m2,s[i].m3,s[i].avg);
}
}
```

Conclusion

- Structure combines variables of different types
- Initialization can't be done inside
- Array of structures can be used to access multiple structure variables
- Assigning structure variables are possible with assignment operator.
- Comparing structure variables has to be done with individual members.