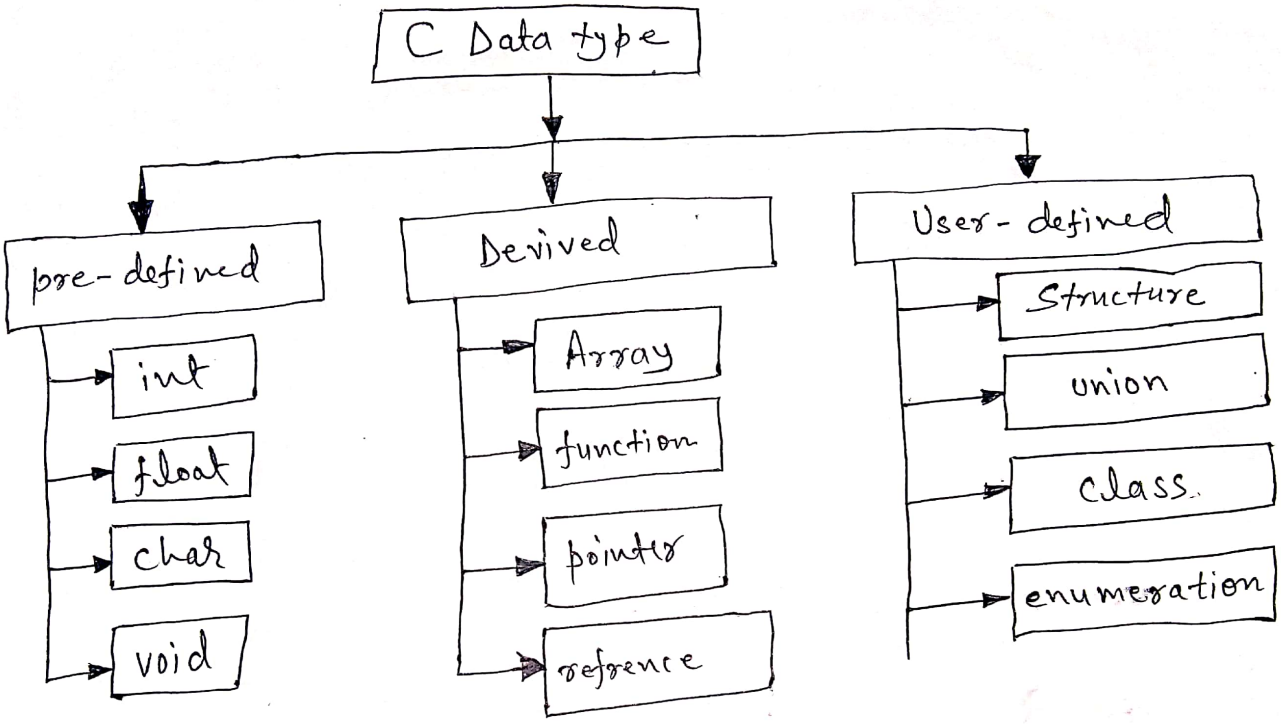


Prof. Suresh K Shukla

Earlier, we have seen how ordinary variable (like x, y...) can hold one piece of data and how Arrays can hold a number of pieces of data/information of the same data type. These two data types can handle a great variety of situation.

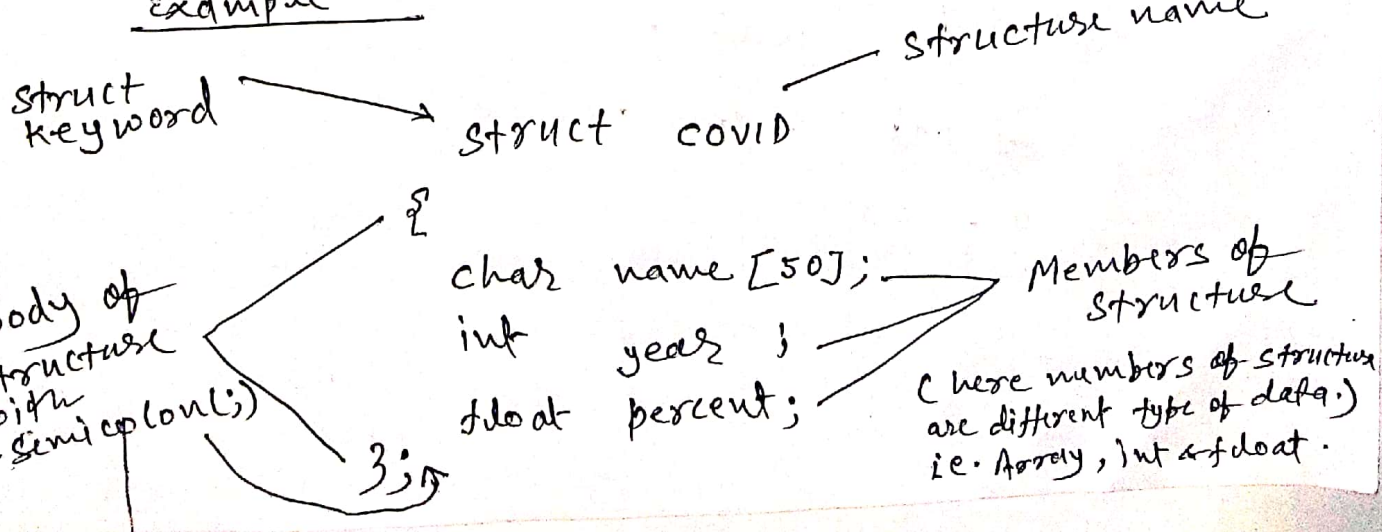
But quite often we deal with entities that are collection of dissimilar data types.

To solve this problem, C provides a special data type — the "STRUCTURE".

Structure :— A structure is a user defined data type in C language which allows us to combine data of different types together.

" Structure is a group of variables of different data types represented by a single name.

example



P-2 Example-2

```
struct student
{
 char name[25];
 int age;
 char branch[10];
 char gender;
};
```

Example-3

```
struct lnjpit
{
 char Add[30];
 int id;
 char branch[10];
 int year;
 float result;
};
```

→ New things, so read carefully

The memory is allocated when the variable of structure tag type is created. To define a variable of structure type, we use the structure tag followed by the variable name. like

```
book X;
```

Here X is a structure variable of book type. Therefore X will have title, author, publication, year and price. Now suppose we want to represent a ~~date~~ date in day/month/year ~~formate~~, say 10/04/2020. One can see that ~~date~~ day month and year all are integer types of values, but combination of these three will make a single entity ie date. So we must declare a structure for date rather than an array of three elements. We can define the variable along with the structure definition. For example;

(1)

```
struct date
{
int day;
int month;
int year;
} adate, bdate;
```

Structure variable

(2)

```
struct
{
char title[20];
char author[10];
char publication[20];
int year;
float price;
} B;
```

In the definition given in case ①, date is a structure tag and adate and bdate are the variable (instances) of data type.

→ Since two structure variables have been declared, the memory space has been reserved for these two variables.

→ Some time structure tag is skipped when only one variable is declared. ie in case ②.

The Dot operator;

The structure variable cannot read or write the values in the in the single comond. ie

```
printf("%d", X);
```

this statement produces the error since a single statement cannot write the value for all members of X.

Each member of a structure is accessed by using components; a variable name followed by a dot, and then member name. Once the structure variable is defined, its member can be accessed by using dot (.) operator. ie

① adate.month = 11;
② x.price = 750.78;
③ x.year = 2020;

Cont.....

The format for the dot operator is;

Syntax

```
structure_variable . member_name;
```

Initialization of a structure

book X = {"C", "Suwesh Shukla", "WILEY", 2020, "250.19"};

here X is variable of book type whose members are initialized as

X. title = "C"
X. author = "Suwesh Shukla"
X. publication = "WILEY"
X. year = 2020
X. price = 250.19

Q Write a program to intialization of value in structure.

```
struct student
  {
   char name[20];
   int roll_no;
   float marks;
   char gender;
  long int phone_no;
};
```

cont....

P-6

```
void main()
{
 struct student st1 = {"Ram", 4, 79.8, 'M', 7542958330};
printf("Name\t RollNO\t Marks\t Gender\t PhoneNo.");
printf("\n %s\t %d\t %f\t %c\t dd", st1.name,
        st1.rollno, st1.marks, st1.gender, st1.phone_no);
 getch();
}
```

---

8 Create a structure named student that has [redacted] roll and marks as member. Assume appropriate types and size of member. w.a.p. using structure to read and display the data entered by the user.

```
struct student
 {
  [redacted]
  int roll;
  float marks;
 };
void main()
{
 struct student s;
```

P.T.O.

```
printf("\n Enter roll : \t");
 scanf ("%d", &s.roll);
printf("\n Enter marks: \t");
 scanf("%f", &s.mark);
printf("\n Roll \t Mark \n");
printf("\n----------\n");
printf("\n%d \t %f", s.roll, s.mark);
 getch();
}
```