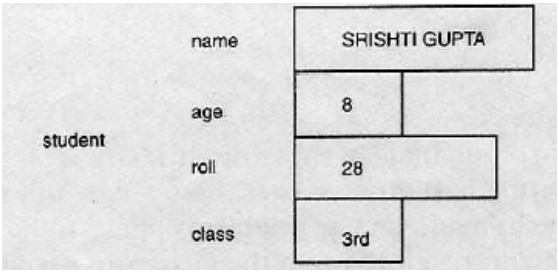
Structures-

Structure is a method of storing heterogeneous (mixed data type) elements.



Ex:

The information about a student is composed of many components like name, age, roll no., class:

The collective information about the student as shown is called a structure. It is similar to a record. The term structure can be precisely defined as: A group of related data items of arbitrary types. Each member of a structure is a variable that can be referred to through the name of the structure.

Declaring a structure

A structure is declared with the keyword struct followed by its name and a body enclosed in curly braces. The body of the structure contains the definition of its members and each member must have a name. The declaration ends by a semicolon.

Syntax

```
struct <name>
{
   member 1
   member 2
   member n
};
```

where struct is the keyboard

<name> is the name of the structure and is optional

member 1,2, ...n are the individual member declarations.

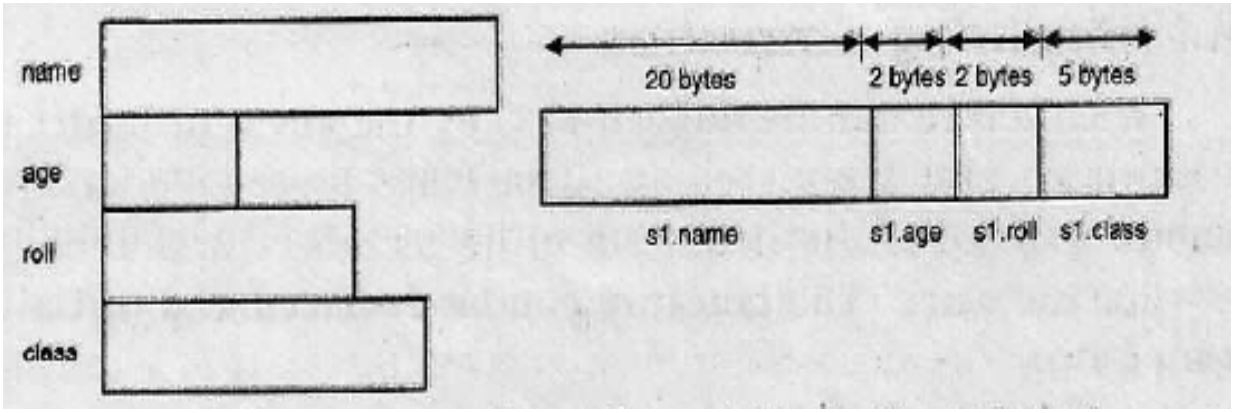
Ex:

```
struct student
{
  char name [20];
  int age;
  int roll;
  char class [5];
};
```

Student is a user defined datatype which consist of data members: name, age, roll and class. Since student is a type in itself, some variable will have to be declared of this type before data is stored into the structure variable.

struct student s1;

s1 is the variable to structure student.



One or more structure variable can be declared-

```
struct student
{
   char name [20];
   int age;
   int roll;
   char class [5];
}sl, s2;
```

s1, s2 are structure variables to student and both variable will have the same memory space.

Accessing structure elements

The structure variable sl has four members; name, age, roll and class. These members can be designated as: sl. name, s 1. age, sl. roll and s 1. class respectively. The individual member of the

structure is designated by the structure variable followed by a period and the member name. The period is called as structure member operator or **dot operator.**

The information of a student can be stored in structure variables sl is as follows.

```
sl. name = "SRISHTI GUPTA;
sl. age = 8;
sl. roll = 28;
sl. class = "3rd"
```

The variable sl can be read or written by the input/output statements as shown below:-

```
gets(sl. name);
scanf("%d %d %s", &s1. age, &s1. roll, sl. class);
printf("%s %d %d %s" , sl. name, sl. age, sl. roll, sl. class);
```

Initializing Structures

The structure can be declared and initialized as follows:

```
struct student
{
   char name [20];
   int age;
   int roll;
   char class [5];
};
struct student sI = {"SRISHTI GUPTA", 8, 28, "3rd"};
```

The values are assigned to the members of the structure in order of their appearance i.e. first value is assigned to the first member, second to second and so on.

Write a program to enter data of a student using structure whose elements are name with 20 characters, age and roll no. of integer type and class with 5 characters. Also print the same.

```
#include <stdio.h>
#include <conio.h>
void main( )
{
  struct student
  {
    char name [20];
    int age;
    int roll;
    char class [5];
  }
  struct student s1;
  clrscr( );
  printf (" \n Enter data of a student ");
  gets (s1. name);
  scanf ("%d %d %s", &s1. age, &s1. roll, sl. class);
  printf (" \n First student name, age, roll no, class ... \n")
```

```
   printf ("%s \t %d \t %d \t %s", sl. name, sl. age, sl. roll,
sl. class);
   getch ( );
}
```

Write a program which reads a record of a student and computes the percentage obtained by him. It also assign the grade on the following criteria

Percentage marks	Grade
> = 80	A
> = 60 < 80	B
> = 50 <60	C
< 50	D

The structure of a student record is given below:-

name	roll no.	sub 1	sub 2	sub 3	sub 4	percentage

/* Percentage of marks and grade of a student */

```
#include <stdio.h>
#include <conio.h>
void main( )
{
    struct student
    {
      char name[201;
      int roll;
      int subl, sub2, sub3, sub4;
      float per;
     }sl;
   clrscr ( );
   printf (" \n Enter the record of the student");
   printf ("\n Name:");
   gets (s 1. name);
   printf (" \n Roll No.:");
   scanf ("%d", &sl .roll);
   printf (" \n Subject 1:");
   scanf ("%d", &sl.subl);
   printf (" \n Subject 2:");
   scanf ("%d", &sl.sub2);
   printf (" \n Subject 3:");
   scanf ("%d", &sl.sub3);
   printf (" \n Subject 4:");
   scanf ("%d", &sl.sub4);
    s1. per = (s1 . sub1 + s1 . sub2 + s1 .sub3 + s1 . sub4)/
4.0;
  if (sl.per> = 80)
```

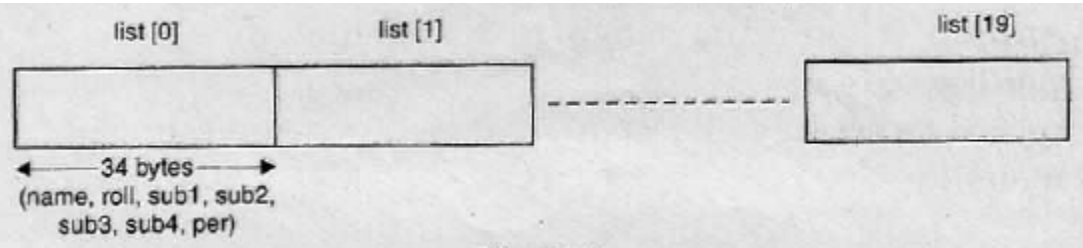
```
    printf (" \n A Grade with %f per", sl.per);
  else if (s1 .per> = 60)
     printf (" \n B Grade with %f per ", sl.per);
  else if (sl.per> 50)
     printf (" \n C Grade with%f per", sl. per);
  else
     printf ("\n D Grade with %f per", sl.per);
  getch( );
}
```

Array of structures

An array of structures can be declared by specifying the size of the array in structure

Ex: A 20 element array of type student called list can be defined as shown below:-

```
struct student
{
    char name[20];      /* 20bytes*/
    int roll;           /* 2bytes */
    int sub 1, sub2, sub3, sub4;  /* 8 bytes */
    float per;          /*4 bytes */
} struct student list[20];    /* declaration of an array of structures */
```



Write a program to calculate the percentage of marks of n students of above structure using array of structures.

```
#include <stdio.h>
#include <conio.h>
void main( )
{
  struct student
  {
    char name [20];
    int roll;
    int sub 1, sub2, sub3, sub4;
    float per;
  }  list[20];
  int i, n;
  clrcsr( );
  printf ("\n Enter number of students");
  scanf ("%d", & n)
  printf (" \n Enter data of %d students", n);
  for (i = 0; i<n; i++)
  {
          printf (" \n Enter name, roll no, marks in four subjects of %d student \n", i+1);
     fflush(stdin);
     gets (list[i]. name);
     scanf ("%d%d%d%d%d", & list [i].roll, &list[i] subl, &list[i].sub2, & list[i]. sub3,
```

```
      &list[i].sub4);
list[i].per=(list[i].subl+list[i].sub2+list[i].sub3+list[i].sub4
)/ 4.0;
    }
    printf (" \n Print the data of % d students", n);
    for (i = 0; i< n; i++)
                printf (" \n(%s \t %d \t %f\n", list[i].name,
list[i].roll, list[i].per);
  getch();
}
```

Passing struct to function

- A structure can be passed to any function from main function or from any sub function.
- Structure definition will be available within the function only.
- It won't be available to other functions unless it is passed to those functions by value or by address(reference).
- Else, we have to declare structure variable as global variable. That means, structure variable should be declared outside the main function. So, this structure will be visible to all the functions in a C program.

It can be done in below 3 ways.

1. Passing structure to a function by value
2. Passing structure to a function by address(reference)

3. No need to pass a structure – Declare structure variable as global

PASSING STRUCTURE TO FUNCTION BY VALUE:

In this, the whole structure is passed to another function by value. It means the whole structure is passed to another function with all members and their values. So, this structure can be accessed from called function. This concept is very useful while writing very big programs.

```
#include <stdio.h>
#include <string.h>
struct student
{
        int id;
        char name[20];
        float percentage;
};
void func(struct student record);   // prototype
int main()
{
        struct student record;   //int a
        record.id=1;   //scanf("%d", &record.id);
        strcpy(record.name, "Raju"); scanf("%s", record.name)
       record.percentage = 86.5; scanf("%f", record.percentage)

        func(record);   //function calling
        return 0;
```

```
}

void func(struct student record) //body function callled
{
        printf(" Id is: %d \n", record.id);
        printf(" Name is: %s \n", record.name);
        printf(" Percentage is: %f \n", record.percentage);
}
```

OUTPUT

```
Id is: 1
Name is: Raju
Percentage is: 86.500000
```

PASSING STRUCTURE TO FUNCTION BY ADDRESS:

In this, the whole structure is passed to another function by address. It means only the address of the structure is passed to another function. The whole structure is not passed to another function with all members and their values. So, this structure can be accessed from called function by its address.

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

struct student
{
```

```
        int id;
        char name[20];
        float percentage;
};

void func(struct student *record);

int main()
{
        struct student record;

        record.id=1;
        strcpy(record.name, "Raju");
        record.percentage = 86.5;

        func(&record);
        return 0;
}

void func(struct student *record)
{
        printf(" Id is: %d \n", record->id);
        printf(" Name is: %s \n", record->name);
        printf(" Percentage is: %f \n", record->percentage);
}
```

DECLARE A STRUCTURE VARIABLE AS GLOBAL:

Structure variables also can be declared as global variables. So, When a structure variable is declared as global, then it is visible to all the functions in a program. In this scenario, we don't need to pass the structure to any function separately.

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

struct student
{
        int id;
        char name[20];
        float percentage;
};
struct student record; // Global declaration of structure

void structure_demo();

int main()
{
        record.id=1;
        strcpy(record.name, "Raju");
        record.percentage = 86.5;

        structure_demo();
        return 0;
}
```

```
void structure_demo()
{
        printf(" Id is: %d \n", record.id);
        printf(" Name is: %s \n", record.name);
        printf(" Percentage is: %f \n", record.percentage);
}
```