**Structures:**

It is a Collection of values/members of different data types.

It is a user defined data type by which you can declare variable of its type (Structure variable).

**Syntax:**

struct struct\_name

{

data\_type var1,var2,...;

data\_type var1,var2,...;

}struct\_var1,struct\_var2,....;

Eg: struct Student

{

int roll\_no;

char name[24];

float height,percentage;

int total,marks[6];

};

**Structure variable declaration:**

1. one way of declaring structure variable is,

struct struct\_name st\_var1,st\_var2,...;

2. We can declare a structure variable at the time structure definition.

Example:

struct Student

{

.......

......

}std1,std2,st3,.....;

**Initializing Structure Members:**

Structure members can be initialized at declaration. This is similar to the initialization of arrays; the initial values are simply listed inside a pair of braces, with each value separated by a comma.

struct Student std1 = {10, "Harish" , 165.6, 68.9, 476, {67,57,89,47,55,74}};

**Member accessing using dot operator:**

In order to access the members in structure we use a dot (.) operator with the structure variable.

Syntax:

struct\_var.member = value;

printf("format",struct\_var.member);

Eg:

std1.name = "Sateesh";

std2.rool\_no = 23;

**/\* program to define a structure and access members using dot operator \*/**

struct student

{

int rollno;

char name[24];

float average;

};

void main()

{

struct student std1,std2;

clrscr();

std1.rollno = 123;

std2.rollno = 234;

printf("Enter name of the two students:\n");

gets(std1.name);

gets(std2.name);

printf("Enter averages of the two students:\n");

scanf("%f%f",&std1.average,&std2.aver age);

printf("The Details of the two students:\n");

printf("Student 1:\nrollno: %d\nname: %s\naverage:%f",std1.rollno,std1.name, std1.average);

printf("\n\nStudent 2:\nrollno: %d \n name: %s\naverage: %f",std2.rollno, std2.name,std2.average);

getch();

}

**Arrays to Structure:**

The structure variable may be an array, which can be used to declare array of structures.

Syntax:

struct student std[25];

std[0].name = "raga";

/\* program to use array of structures \*/

#include<stdio.h>

#include<conio.h>

struct student

{

int rno;

char name[15];

};

struct dob

{

int day, month, year;

};

void main()

{

struct student s[10];

struct dob d[10];

int i;

clrscr();

/\* reading 5 student information \*/

printf("Enter student info: \nroll name dob in (date/mon/year)");

for(i=0; i<5; i++)

{

scanf("%d", &s[i].rno);

gets(s[i].name);

scanf("%d/%d/%d", &d[i].day, &d[i].month, &d[i].year);

}

/\* printing the student info you entered \*/

printf("Enter which student info you want(from 1 to 5): ");

scanf("%d", &i);

i-= 1;

printf("The Student info for %d is\n");

printf("Rollno: %d\nName: %s\nD.O.B: %d/%d/%d", s[i].rno, s[i].name,d[i].day, d[i].month, d[i].year);

getch();

}

**Pointer to a Structure:**

Here we assign a pointer to a structure in order to store the address of the structure variable.

Syntax:

struct student std[20],std3,\*ssp;

ssp = &std3;

Accessing members using pointer to a structure:

Syntax:

ssp->name = "something";

**/\* program to define a structure and pointer to it then access members using arrow(->) operator \*/**

struct student

{

int rollno;

char name[24];

float average;

};

void main()

{

struct student std1,std2,\*stdp;

clrscr();

/\* assigning the structure address to a structure pointer \*/

stdp = &std1;

std1.rollno = 123;

std2.rollno = 234;

printf("Enter name of the two students: \n");

gets(std1.name);

gets(std2.name);

printf("Enter averages of the two students:\n");

scanf("%f%f",&std1.average,&std2.aver age);

stdp->rollno = 1;

printf("The Details of the two students:\n");

printf("Student 1:\nrollno: %d\nname: %s\naverage: %f",stdp->rollno,stdp-> name,stdp->average);

printf("\n\nStudent 2:\nrollno: %d\n name: %s\naverage: %f",std2.rollno, std2.name,std2.average);

getch();

}

**Structures inside a structure:**

We can declare a structure inside another structure.

Syntax:

struct Student

{

int no;

struct DateOfBirth

{

int day,month,year;

}dob;

}std;

Accessing a member which is inside another structure:

std.dob.day = 24;

std.dob.month = 5;

std.dob.year = 1996;

**/\* program to understand the structures inside another structure \*/**

void main()

{

struct Student

{

int rollno;

char \*name;

struct DateofBirth

{

int day, month, year;

}DOB;

};

struct Student std[10];

int i, nos;

clrscr();

/\* Reading no. of students \*/

printf("Enter no. of Students(<10): ");

scanf("%d", &nos);

/\* Reading the students info \*/

printf("Enter %d students information: \n", nos);

for( i=0; i<nos; i++)

{

printf("\nEnter Student %d info: \n", i);

printf("Roll no: ");

scanf("%d", &std[i].rollno);

printf("Name: ");

flushall();

gets(std[i].name);

printf("D.O.B(d/m/y): ");

scanf("%d/%d/%d", &std[i]. DOB.day, &std[i].DOB.month, &std[i].DOB.year);

}

/\* Print a particular student information \*/

printf("Enter a student no from 1 to %d: ", nos);

scanf("%d", &nos);

printf("Student %d information:\n", nos--);

printf("Roll no: %d\nName: %s\nD.O.B: %d/%d/%d",

std[nos].rollno, std[nos].name,

std[nos].DOB.day, std[nos].DOB.month, std[nos].DOB.year);

getch();

}

**Introduction to Unions:**

Unions are also C variables whose syntax looks similar to structures, but act in a completely different manner. A union is a variable that can take on different data types in different situations. The union syntax is:

union tag\_name

{

type1 member1;

type2 member2;

…

};

For example, the following code declares a union data type called intfloat and a union variable called Useit:

union intfloat

{

float f;

int i;

};

union intfloat Useit;

**Unions and Memory:**

• Once a union variable has been declared, the amount of memory reserved is just enough to be able to represent the largest member (Unlike a structure where its memory is reserved for all members).

• In the previous example, 4 bytes are set aside for the variable intfloat since a float will take up 4 bytes and an int only 2 (on some machines).

• Data actually stored in a union’s memory can be the data associated with any of its members. But only one member of a union can contain valid data at a given point in the program.

• It is the user’s responsibility to keep track of which type of data has most recently been stored in the union variable.

/\*program to understand the Union concept \*/

void main()

{

union intfloat

{

int i;

float f;

};

union intfloat ifl;

ifl.i = 45; /\* Integer value assigned to intfloat structure \*/

printf("intfloat.i = %d, intfloat.f = %f\n",ifl.i, ifl.f);

ifl.f = 45.0687; /\* Float value assigned to intfloat structure \*/

printf("intfloat.i = %d, intfloat.f = %f\n",ifl.i, ifl.f);

getch();

}