Lecture 13

Structure

Structure

A structure is a user defined data type. We know that arrays can be used to represent a group of data items that belong to the same type, such as int or float. However we cannot use an array if we want to represent a collection of data items of different types using a single name. A structure is a convenient tool for handling a group of logically related data items.

The syntax of structure declaration is

```
struct structure_name
{
type element 1;
type element 2;
……..
type element n;
};
```

Structure

In structure declaration the keyword struct appears first, this followed by structure name. The member of structure should be enclosed between a pair of braces and it defines one by one each ending with a semicolon. It can also be array of structure. There is an enclosing brace at the end of declaration and it end with a semicolon.

We can declare structure variables as follows
struct structure_name var1,var2,…..,var n;

Structure

For Example:
To store the names, roll number and total mark of a student you can declare 3 variables. To store this data for more than one student 3 separate arrays may be declared. Another choice is to make a structure. No memory is allocated when a structure is declared. It just defines the “form” of the structure. When a variable is made then memory is allocated. This is equivalent to saying that there's no memory for “int” , but when we declare an integer that is. int var; only then memory is allocated. The structure for the above-mentioned case will look like

```
struct student
{
int rollno;
char name[25];
float totalmark;
};
```

Structure

We can now declare structure variables stud1, stud2 as follows

struct student stud1,stud2;

Thus, the stud1 and stud2 are structure variables of type student. The above structure can hold information of 2 students.

It is possible to combine the declaration of structure combination with that of the structure variables, as shown below.

```
struct structure_name
{
type element 1;
type element 2;
……..
type element n;
}var1,var2,…,varn;
```

Structure

The following single declaration is equivalent to the two declaration presented in the previous example.

```
struct student
{
int roll no;
char name[25];
float total mark;
} stud1, stud2;
```

- The different variable types stored in a structure are called its members.
- The structure member can be accessed by using a dot (.) operator, so the dot operator is known as structure member operator.

Initializing Structure Members

Structure members can be initialized at declaration. This much the same manner as the element of an array; the initial value must appear in the order in which they will be assigned to their corresponding structure members, enclosed in braces and separated by commas .The general form is

struct stucture_name var={val1,val2,val3…..};

Initializing Structure Members

Example:

```
#include <stdio.h>
#include<conio.h>
int main()
{
struct student
{
char name;
int  rollno;
float totalmark;
};
struct student stud1={"Ashraf",1,98};
struct student stud3= {"Rahul",3,97};
struct student stud2={"Vineeth",2,99};
clrscr();
printf("STUDENTS DETAILS:\nRoll
number:%d\n\nName:%s\n\nTotel
mark:%.2f\n",stud1.rollno,stud1.name,stud1.totalmark);
printf("\nRoll number:%d\n\nName:%s\n\nTotel
mark:%.2f\n",stud2.rollno,stud2.name,
printf("\nRoll number:%d\n\nName:%s\n\nTotel
mark:%.2f\n",stud3.rollno,stud3.name,stud3.totalmark);
getch();
return 0;
}
```

Structure and Functions

- A structure can be passed as a single variable to the function.
- The structure should be defined outside the main() when they are used along with function.
- The field and member data should be same throught the program either in main() or in any function.