Paper 102: Programming & Problem solving through C

Lecture-18:Unit-III Structures

Aiusha V.Hujon, Dept. of Computer Science, SAC

Introduction to structures

- A structure is a heterogeneous user defined data type
- It can contain variables of different data types
  - These are group together into a single entity
- Syntax

```
struct [<struct type name>]
{
   [<type> <variable-name> [, <variable-name>,...]];
    [<type> <variable-name> [, <variable-name>,...]];
   ...
} [<structure variables.]
```

Rules for declaration of individual members

- The individual members may be of any basic data type like
  - Int,float,char,double etc…
  - Pointers, arrays or even other structure
- All member names must be unique
  - They can be same as those declared outside the structure
- Individual members cannot be initialized inside the structure declaration

Structure declaration variables

- In the structure declaration

```
struct student
{
    int rollno;
    int subject;
    float marks;
    } student1  student2;
```

```
struct {
    int rollno;
    int subject;
    float marks;
    } student1   student2;
```

- Student is the structure tag, while student1 and student2 are variables of type student
- If other variables are not required the structure tag can be omitted

Structure declaration variables

- Using the structure tag

```
struct student
{
    int rollno;
    int subject;
    float marks;
} ;
struct student student1,student2;
```

Structure initialization

- struct student student1={10,101,88.0};

Accessing the member variables

- student1.rollno
- student2.rollno

- scanf("%d",&student1.rollno);
- printf("rollno=%d",student1.rollno);
- student1.marks=88.0;
- student1.marks +=10;
  - Marks is incremented by 10

Accessing the member variables

- struct student s[10];
  - Array of data type student
- Members are access by using

```
s[0].rollno=12;
s[1].rollno=13;

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

Structure within a structure

```
struct date
{
    int day;
    int month;
    int year;
};
struct person
{
    char personname[25];
    struct date birthday;
    float salary;
};
```

Structure within a structure

- struct person p1;
- p1.personname="John"

  P1.birthday.day=10;

  P1.birthday.month=12;

  P1.birthday.year=1990;

  P1.salary=5000.0;

  struct person p1[10];
- p1[0].personname="John"

  P1[0].birthday.day=10;

  P1[0].birthday.month=12;

  P1[0].birthday.year=1990;

  P1[0].salary=5000.0;

Creating a user defined data type

- typedef struct

```
{
   int day;
   int month;
   int year;
} date;
typedef struct person
{
   char personname[25];
   date birthday;
   float salary;
}emprec;
```

Creating a user defined data type

- emprec per;
- Structure variables can also be declared as pointers
- emprec *per;
- Structure members can be accessed as follows

  per->personname

  per->salary

Passing structures to functions

- The function prototype can be written as

  void readin(emprec record);

- An this function can be called as

  emprec rec1={"annie",10,8,75,4000.00};

  readin(rec1);

Function returning a structure

- The function defintion can be written as

```
emprec readin(emprec record)
{
   ...
   return(record);
}
```