**Structures**

* A structure is a collection of simple variables. The variables of a structure can be of different data types: int, char, float and so on.
* The data items or variables declared in the structure are called the members of the structures and the structure name is referred to as its tag.
* The keyword struct tells the compiler that a structure is being defined.
* General form of a structure definition is:

struct tag

{

type variable\_name1

type variable\_name2

type variable\_name3

.

.

} variable1, variable2,….;

For eg:

struct time

{

int hh;

int mm;

int ss;

};

Here, the identifier time is the structure tag.

* The definition of the structure doesn’t occupy any memory as no structure variable has been defined. Only the form of data has been defined by defining a structure.

Therefore the complete declaration is:

struct time

{

int hh;

int mm;

int ss;

};

time arrival\_time;

Now the structure arrival has its members as hh, mm and ss.

* The two separate statements can be joined as:

struct time

{

int hh;

int mm;

int ss;

} arrival\_time;

* More than one structure variable can be defined while defining structure.
* The structure tag can be eliminated when there is only one variable for structure is needed.
* Once a structure variable has been defined its members can be accessed using the dot operator. The structure variable name followed by a period (.) and the element name refer to that individual structure element. The syntax is:

< structure name>.<element name>

For eg: arrival\_time.ss=39;

* The structure members are treated just like normal variable.

To print element we write: cout<<arrival\_time.ss;

And to get value from user we write: cin>>arrival\_time.ss;

* The elements of structures can be initialized either using separate assignment operator or jointly using the notation similar to that user for initializing arrays.

For example:

struct student

{

int r\_no;

char name[20];

int standard;

float marks;

char grade;

}stud1,stud2;

* + Separately

stud1.r\_no=2;

stud1.name=”rahul”;

stud1.standard=12;

stud1.marks=67;

stud1.grade=’A’;

* + Jointly

student stud1={2,”rahul”,12,67,’A’};

* One structure can be assigned to another only if both are of the same type.

stud2=stud1;

For eg:

struct first

{

int x;

};

struct second

{

int x;

};

first v1,v2;

second v3;

cin>>v1.x;

v2=v1; //correct assignment

v3=v1; //error type mismatch

.

* A structure element may b complex or simple. The simple elements of a structure are any of the fundamental data types of c++. En elements of a structure may be an array or a structure itself. Ans is calles as a complex structure.

For eg:

struct distance

{

float feet;

float inches;

};

struct volume

{

distance length;

distance width;

distance height;

};

volume vol;

* Difference between arrays and structures:
  + Structures are user defined data types. Where arrays are derived data types.
  + Array is a collection of analogous elements while structure assembles dissimilar elements under one roof.
* There may be structures contained in an array or an array can be elements of a structure.

For eg:

struct item

{

Int itemno;

Char desc [20];

Float cost;

};

item it[50];

A structure can also be passed as an argument to a function. A Structure can be declare in two ways: globally and locally.

A structure can be passed to a function by value or by reference.

Returning structure is also the same as returning other data type.