Chapter-14

STRUCTURES

➢ Introduction:

- We have seen variables of simple data types, such as float, char, and int.
- Variables of such types represent one item of information: a height, an amount, a count, and so on.
- But just as groceries are students into colleges and words into sentences, it is often convenient to organize simple variables into more complex entities.
- The C++ construction called the structure is one way to do this.
- A structure is a collection of simple variables. The variables in a structure can be of same or different types: some can be int, some can be float and so on.
- The data items in a structure are called the members of the structure.
- For C++ programmers, structures are one of the two important building blocks in the understanding of objects and classes.
- The syntax of a structure is almost identical to that of a class. A structure is a collection of data, while a class is a collection of both data and function.

➢ Defining a structure

- The process of defining a structure includes giving the structure a name called the tag and telling the compiler the name and the data type of each piece of data you want to be included in a structure.
- Syntax:

```cpp
struct structure-name
{
    datatype member-name-1;
    datatype member-name-2;
    .........................
    datatype member-name-n;
}
```

- The keyword struct introduces the structure definition.
- Next comes the structure name or tag, which is part.
- The declarations of the structures members are enclosed in braces.
- The list of all structure members is called template.
- A semicolon follows the closing braces, terminating the entire structure.
• Example: A structure definition to hold employee information.

```c
struct employee
{
    int    idno;
    char   name[15];
    char   designation[10];
    float  salary;
};
```

• This structure is named employee. It contains four members: an integer named idno, a character array to hold 15 characters for a name, another character array to hold 10 characters for designation and float named salary.

• The structure requires an area in memory, which is 31 bytes long.

• The figure given below shows the allocation of memory for the individual **elements**.

<table>
<thead>
<tr>
<th>Idno</th>
<th>Name</th>
<th>Designation</th>
<th>Float</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 bytes</td>
<td>15 bytes</td>
<td>10 bytes</td>
<td>4 bytes</td>
<td>31 bytes</td>
</tr>
</tbody>
</table>

➢ **Declaring a structure or defining a structure variable:**

• A structure can be declared (or a structure variable can be defined) as we defined a basic built-in data type.

• The general form is

```
structure-name variable;
```

• Example 1:

```c
student old_student, new_student;
```

old_student and new_student are the two structure variables declared as we declare built-in data types.

• Example 2:

```c
employee emp1, emp2;
```

emp1 and emp2 are the two structure variables declared as belong to the user-defined data type employee.

• We can combine the definition of the structure and definition of structure variables.

• Example:

```c
struct employee
{
    int    idno
    char   name[15];
    char   designation[10];
    float  salary;
} emp1, emp2;
```
Note:

- The definition of a structure does not allocate memory space. The definition just defines the blueprint for the creation of variables.
- A semicolon should follow the last brackets of the structure.
- The definition of the structure should be come either before the main function or before any other variables are declared.
- The declaration of the structure variables always comes after the definition. The declaration of the structure creates space to store the values for the variables.

Accessing the elements of a structure

- After the structure variables are defined, we should know how the member of a structure variable is accessed.
- A member of a structure is always part of a structure and it is not possible to refer the member directly.
- *The member of a structure can be accessed using* **dot operator**.
- The syntax of dot operator is as follows:

  \[
  \text{structure_variable-name.member_name}
  \]

- The structure member is written in three parts:
  - The name of the structure variable (part1);
  - The dot operator, which consists of a period (.)
  - The member name (part3).
- Note: The real name of the dot operator is **member access operator**.
- Example 1: To access the name of the member emp1 of the tag employee, we write

  \[
  \text{emp1.name;}
  \]
- Example 2: To access the salary of the member emp2 of the tag employee, we write

  \[
  \text{emp2.salary}
  \]
Sample Program: To input and display the information of a student

```c++
#include<iostream.h>
#include<iomanip.h>
#include<string.h>

void main()
{
    struct student
    {
        int regno;
        char name[15];
        char comb[4];
        float perc;
    } st;
    cout<<"Enter the register number of the student: ";
    cin>>st.regno;
    cout<<"Enter the name of the student: ";
    cin>>st.name;
    cout<<"Enter the combination of the student: ";
    cin>>st.comb;
    cout<<"Enter the percentage of the student: ";
    cin>>st.perc;
    cout<<"Register No: "<<st.regno<<endl;
    cout<<"Name: "<<st.name<<endl;
    cout<<"Combination: "<<st.comb<<endl;
    cout<<"Percentage: "<<st.perc<<endl;
    getch();
}
```

Sample run:
Enter the register number of the student: 1234
Enter the name of the student: Nalina M A
Enter the combination of the student: CEBA
Enter the percentage of the student: 90.05

Register no: 1234
Name: Nalina M A
Combination: CEBA
Percentage: 90.05

Example: Consider the assignments:

- `std.regno=12345`
- `std.name[]="empress college";`
- `std.perc=79.54;`
Initializing a structure:
- Like any other data type, it is also possible to initialize the structure members of a structure variable.
- Structures members are initialized when they are declared.
- Example: A program may contain the following initialization of the structure.
  ```c
  employee emp1 = {
      1234,
      "Lakshmi",
      "Manager",
      10500.00
  };
  ```
  - This initialization initializes idno field to 1234, the name field to "Lakshmi", the designation field to “Manager” and the salary field to 10500.00.
- A structure variable can be assigned to another structure variable.
- Example: std2 = std1;
- Here, std1 and std2 are the structure variables of the structure std.
- The value of each member of std1 is assigned to the corresponding member of std2.
- Since a large structure can have dozens of members, such an assignment statement can require the computer to do a considerable amount of work.
- Note that one structure variable can be assigned to another only when they are of the same structure type.
- If you try to assign a variable of one structure type to variable of another type, the compiler will complain.

Nested structures:
- A structure can be defined as a member of another structure. Such structure where one structure is embedded within another structure is called as a nested structure.
- During the declaration of nested structure, the definition of the embedded structure must appear before the definition of outer structure.
- Example: consider the definition of the following structure.
  ```c
  struct distance {
      int feet;
      float inches;
  };
  ```
  ```c
  struct room {
      distance length;
      distance breadth;
  };
  ```
• The structure room contains another structure distance as one of its members.
• Whenever nested structures are used, the definition of the internal structure should precede the
definition of the outer structure.
• In the above example, the definition of the structure distance precedes the definition of the
structure room.
• Example: consider the definition of structure student.

```
struct date
{
  int dayno;
  char month[10];
  int year;
};

struct student
{
  int regno;
  date doa, dob;
  int marks;
};
```

• To access the field regno of the structure std, we write `std.regno`.
• To access the day number of the date of admission of the structure std, we write `std.doa.dayno`.
• To access the year of date of birth of the student std, we write `std.dob.year`.

➢ Array of structures:
• A single variable that represents a structure creates memory space to store one set of information.
• If we want to create memory space for more instances then we may have to increase the number
of variables or use the facility called arrays.
• An array of structure is an array in which each element of the array is a structure. Thus it is a
collection of structures put together as an array.
• For example, let us consider that there are 10 students studying in a college. We have defined the
structure so that it could maintain information about students of the college. If we have to store
data of 10 students we need to use arrays rather than a single variable.
• When create an array of the structure, we are creating an array of the entire template. This can be
done as follows.

```
struct student
{
  int regno;
  char name [15];
  char comb;
  float fees;
} s[10];
```
In the declaration, s is an array of 10 elements. Each element of the array is a separate structure of type of student.

The expression:
- s[0].name will access the name of the 1st student, while
- s[9].fees access the fees paid by the last student

<table>
<thead>
<tr>
<th>regno</th>
<th>name</th>
<th>comb</th>
<th>fees</th>
</tr>
</thead>
<tbody>
<tr>
<td>s[0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s[1]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s[9]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Memory allocation for an array of structures

An array of structures can be assigned initial values just like any other array.

Remember that each element is a structure that must be assigned a corresponding set of initial values. The process of initialization is illustrated below.

Example:
```c
struct info {
    int regno;
    char name[15];
    float fees;
};
info s[ ] = {
    1, "Lavanya", 5500.00,
    2, "Manasa", 6250.00,
    3, "Sajay", 6000.00,
    4, "Sahana", 5900.00,
};
```

Practical Program 30: Write a program to input the register number, name and class of all the students in a class into a structure and output the data in a tabular manner with proper heading.

```c
#include<iostream.h>
#include<conio.h>
struct student {
    int regno;
    char name[15];
    char section[4];
};
```
void main() {
    student s[50];
    int i, j, n;
    clrscr();
    cout<<"How many students?"<<endl;
    cin>>n;
    for(i=0; i<n; i++)
    {
        cout<<"Enter the Reg No of the Student "+i+1<<":";
        cin>>s[i].regno;
        cout<<"Enter the Name of the Student "+i+1<<":";
        cin>>s[i].name;
        cout<<"Enter the Class of the Student "+i+1<<":";
        cin>>s[i].section;
    }
    cout<<"REG_NO \t NAME \t CLASS \n"<<endl;
    for(i=0; i<n; i++)
    {
        cout<<s[i].regno<<"\t"<<s[i].name<<"\t"<<s[i].section<<endl;
    }
    getch();
}

CHAPTER 14 – STRUCTURES BLUE PRINT

<table>
<thead>
<tr>
<th>VSA (1 marks)</th>
<th>SA (2 marks)</th>
<th>LA (3 Marks)</th>
<th>Essay (5 Marks)</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>01 Question</td>
<td>-</td>
<td>03 Marks</td>
</tr>
</tbody>
</table>

***************