### UNIT – 4

[Classes and Data Abstraction : Structure in C++, Class, Build- in Operations on Classes, Assignment Operator and Classes, Class Scope, Reference parameters and Class Objects (Variables), Member functions, Accessor and Mutator Functions, Constructors, default Constructor, Destructors.]

#### Classes and Data Abstraction

A class is a user defined data type. It is kind of a container or capsule or a cell, which encapsulate the set of methods, attribute and properties to provide its indented functionalities. Abstraction refers to the act of representing essential features without including the background details or explanations. A class is a logical abstraction as encapsulation also allows a class to change its internal implementation without hurting the overall functioning of the system. That idea of encapsulation is to hide how a class does its processing. Since the classes use the concept of data abstraction, they are known as Abstract Data Types (ADT). In further section class is explained in more detailed way.

#### Structure in C++

Structure is a user-defined data type that can handle data of different types which are logically related. Following is template of defining a structure in C++:

```c++
struct <structure tag> {
    member definition;
    member definition;
    ...
    member definition;
} [one or more structure variables];
```

The fields defined in place of member definition inside the curly braces (which specify the scope of the structure) are called **structure members** or **structure elements**. The keyword `struct` declares `<struct tag>` as a new data type that can hold different fields of different datatypes defined as member of structure. The structure definition is only the blueprint for the creating of variables. When a structure is created, no memory is allocated.

The structure variables can be defined at the time of structure definition (as shown in the template) or locally in main as given below:

```c++
int main{
    ......
    struct_tag [one or more structure variables]
    ......
}
```

When structure variable is defined, only then the required memory is allocated by the compiler.

The most efficient method of dealing with structure variables is to define the structure **globally** so that main and any functions in the program are aware that a new data type exists. To declare a structure globally, place it **BEFORE** `int main()`.  

The members of structure variable is accessed using a **dot** or **period (.)** operator. Following program demonstrate the structure in C++:

---

*Provided By Shipra Swati, PSTC, Vaishali, Bihar*
#include <iostream>
using namespace std;

struct Person
{
    char name[50];
    int age;
    float salary;
};

int main()
{
    Person p1;

    cout << "Enter Full name: ";
    cin.get(p1.name, 50);
    cout << "Enter age: ";
    cin >> p1.age;
    cout << "Enter salary: ";
    cin >> p1.salary;

    cout << "\nDisplaying Information." << endl;
    cout << "Name: " << p1.name << endl;
    cout << "Age: " << p1.age << endl;
    cout << "Salary: " << p1.salary << endl;
    return 0;
}

OUTPUT

Enter Full name: sovit
Enter age: 17
Enter salary: 50000

Displaying Information.
Name: sovit
Age: 17
Salary: 50000

Here a structure Person is declared which has three members name, age and salary. Note that no memory is allocated till this point.

Inside main() function, a structure variable p1 is defined. At this point the required memory is allocated. If the memory of float is 4 bytes, memory of int is 4 bytes and memory of char is 1 byte in the working system, then 58 bytes of memory is allocated for structure variable p.

Structure and Class in C++

C++ structures are very similar to a class, with the only difference being that in a class, all members are private by default. But in a C++ structure, all members are public by default.

*Task1: Students are required to find the differences between C structure and C++ structure.

Since Class is a specially introduced datatype in C++, we will concentrate on class only afterwards.

Provided By Shipra Swati, PSTC, Vaishali, Bihar
Class

A class is a way to bind the data describing an entity and its associated function together. The general form of class declaration is:

```cpp
class class-name {
    access-specifier:
    data and functions
    access-specifier:
    data and functions
    // ...
    access-specifier:
    data and functions
} object-list;
```

The class declaration is similar to the struct declaration. The body of the class is enclosed within braces and terminates with a semicolon. The variables declared inside the class are called data members and the functions are known as member functions.

Here, access-specifier which terminates by this `:` symbol, is one of these three C++ keywords:

- **public**: The public access specifier allows functions or data to be accessible to other parts of your program.

- **private**: By default, functions and data declared within a class are private to that class and may be accessed only by other members of the same class. The data hiding concept in C++ is achieved by using the keyword private.

- **Protected**: The protected access specifier is needed only when inheritance is involved.

Once a class is defined, it can be used to create variables of its type known as objects. The relation between an object and a class is the same as that of a variable and its data type. The object-list given above in general form of class declaration is optional. If present, it declares objects of the class. The objects can be created in main function also as given below:

```cpp
class-name object-list;
```

Following diagram denotes how private members of a class can be accessed in controlled manner:
Accessing class members

Class members can be accessed only by an object of that class provided the members are not specified as private. To access class members, the dot (.) operator is used. The dot operator links the name of an object with the name of a member. The general form of the dot operator is shown here:

```
object.member(data or functions)
```

A member function can call another member function directly, without using the dot operator.

Memory Allocation for class and object

Once you define class it will not allocate memory space for the data member of the class. The memory allocation for the data member of the class is performed separately each time when an object of the class is created.

Since member functions defined inside class remains same for all objects, only memory allocation of member function is performed at the time of defining the class. In other words, there is only a single copy of each member function, which is shared among all the objects.

Thus memory allocation is performed separately for different object of the same class. All the data members of each object will have separate memory space, where they can have different values for the different object. The memory allocation of class members is shown below:

Consider the first example of class:

```cpp
#include<iostream>
using namespace std;
```
class book{
    char title[50];
    float price;
public:
    void getdata(void){
        cout<<"Enter Title: ";
        cin>>title;
        cout<<"Enter Price: ";
        cin>>price;
    }
    void putdata(void){
        cout<<"Title: "<<title<<endl<<"price: "<<price<<endl;
    }
};

int main(){
    cout<<"Enter the details of book1:
";
    book1.getdata();
    cout<<"Enter the details of book2:
";
    book2.getdata();
    cout<<"Enter the details of book3:
";
    book3.getdata();

    cout<<"Information of book1:
";
    book1.putdata();
    cout<<"Information of book2:
";
    book2.putdata();
    cout<<"Information of book3:
";
    book3.putdata();
}

OUTPUT

Enter the details of book1:
Enter Title: C++
Enter Price: 200

Enter the details of book2:
Enter Title: NMCT
Enter Price: 150

Enter the details of book3:
Enter Title: BE
Enter Price: 100

Information of book1:
Title: C++
price: 200

Information of book2:
Title: NMCT
price: 150

Information of book3:
Title: BE
price: 100
the three objects, namely, book1, book2 and book3 of the class book have individual copies of the data members title and price. However, there is only one copy of the member functions getdata() and putdata() that is shared by all the three objects.