UNIT -IV
Object Oriented Concepts

Structures in C++

- A STRUCTURE is a C++ data structure that is used store user- defined different data types.
- Structure is the collection of variables of different types under a single name for better visualisation of problem.
- Arrays is also collection of data but arrays can hold data of only one type whereas structure can hold data of one or more types.

```
struct number {
        int img;
        float real;
};
```

How to define a structure in C++?

When a structure is created, no memory is allocated. The structure definition is only the blueprint for the creating of variables. Its just like we declare any datatypes.

Similarly, structure definition only specifies that, what property a structure variable holds when it isdefined.

How to define a structure in C++?

The **struct** keyword is used to defines a structure followed by an identifier(name of the structure).
Whole structure is written inside the curly braces. We can declare one or more data members inside the structure
For example:

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

Here a structure person is defined which has three members: name, age and salary.

How to access members of a structure?

The members of structure variable is accessed using **dot operator.**

Suppose, you want to access age of structure

First declare the object of the srtucture.

Like person p;

p.age;
p.name;

//p.a=p.b+p.c;

EXAMPLE

```
#include <iostream>
using namespace std;
struct Person
{
        int citizenship;
        int age;
};
int main(void) {
        struct Person p;
        p.citizenship = 1;
        p.age = 27;
        cout << "Person citizenship: " <<
p.citizenship << endl;
        cout << "Person age: " << p.age << endl;
        return 0;
}
```

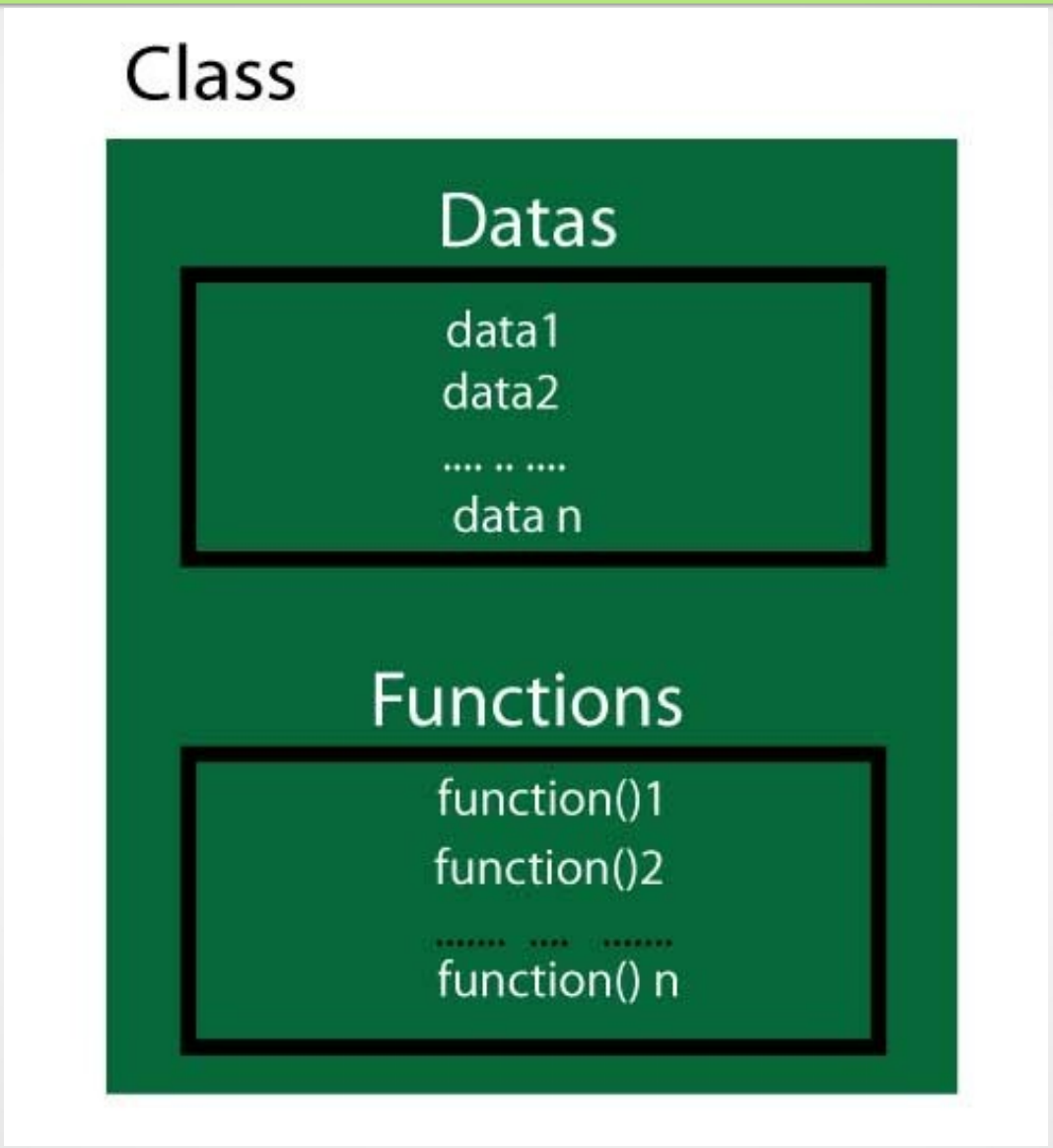
Structure vs Class

Structures	Class
Members are public by default	Members are private by default
It is value type	It is reference type
It is stored on stack	It is stored on heap
It does not support Inheritance	It supports Inheritance
It has only parameterized constructor	It has non parameterized consturctor, default constructor and destructor
Member variables are not initialized directly	Member variables can be initialized directly
Objects can be created without using new keyword	Objects can be created using new keyword

Classes in C++

- A class is the collection of related data and function under a single name.
- A C++ program can have any number of classes. When related data and functions are kept under a class.
- It helps to visualize the complex problem efficiently and effectively.

Classes in C++


Class
Datas
data1
data2
....
data n
Functions
function()1
function()2
....
function() n

Classes in C++

- A Class is a blueprint for objects

- When a class is defined, no memory is allocated. You can imagine like a datatype.
    - int var;
- The above code specifies var is a variable of type integer;
- int is used for specifying variable var is of integer type. Similarly, class are also just the specification for objects and object bears the property of that class.

Defining the Class in C++

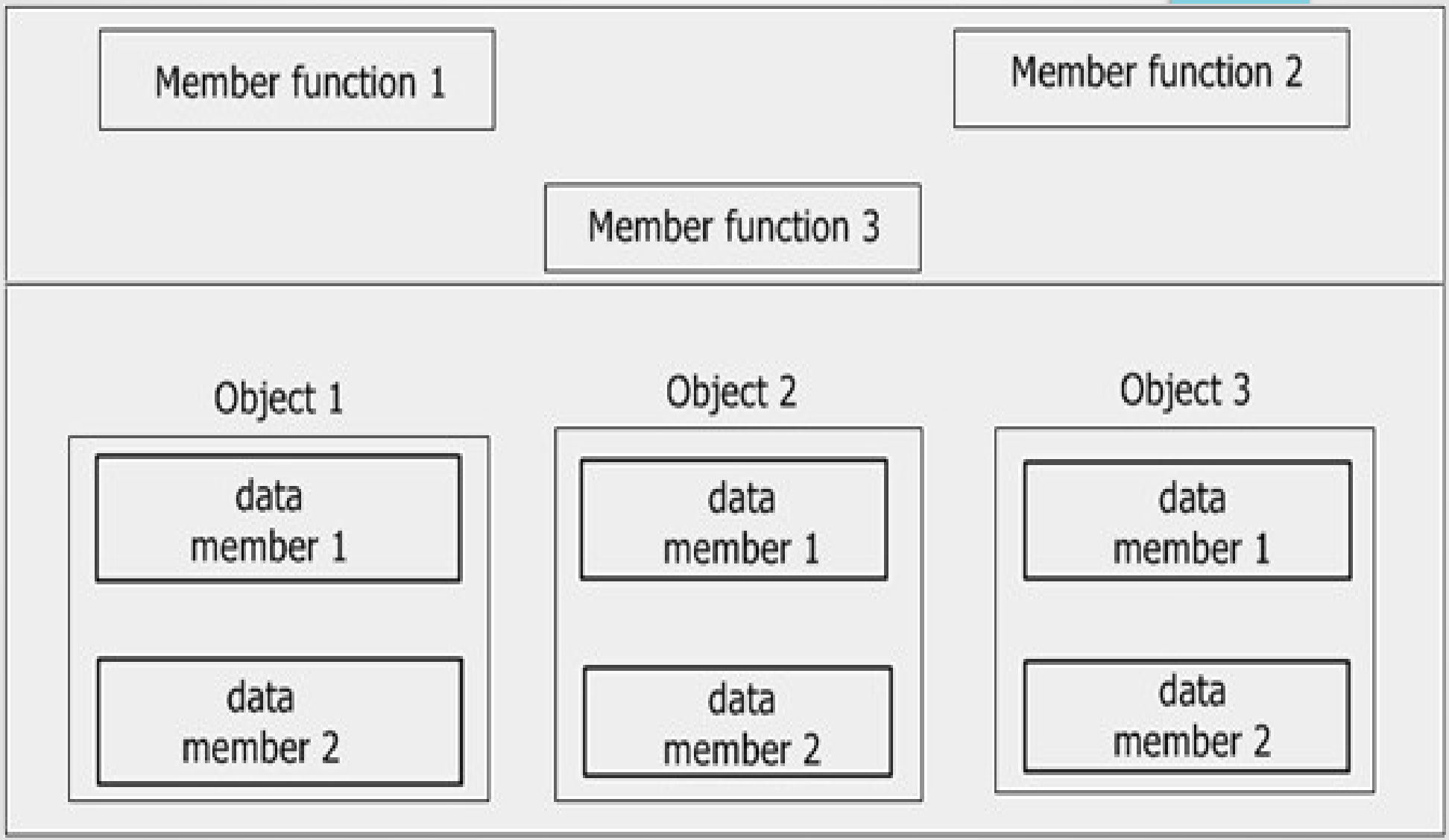
class is defined inside curly brackets an terminated by semicolon at the end in similar way as structure.

```
class class_name
{
// some data
// some functions
};
```

Example of Class in C++

```
class temp
{
private:
int data1;
float data2;
public:
void func1()
{    data1=2;  }
 float func2()
{}
};
```

C++ Objects


Member function 1
Member function 2
Member function 3
Object 1
data member 1
data member 2
Object 2
data member 1
data member 2
Object 3
data member 1
data member 2

C++ Objects

When class is defined, only specification for the object is defined. Object has same relationship to class as variable has with the data type. Objects can be defined in similary way as structure is defined.
Syntax to Define Object in C++

class_name variable name;

For the above defined class temp, objects for that class can be defined as:
temp obj1,obj2;

Here, two objects(obj1 and obj2) of temp class are defined.

Accessing Data Members and Member functions

Data members and member functions can be accessed in similar way the member of structure is accessed using member operator(.). For the class and object defined above, func1() for object obj2 can be called using code:

obj2.func1();

Similary, the data member can be accessed as:

object_name.data_memeber;

onject_name.data tupe; p1.a

Access Specifiers

Access modifiers are used to implement an important aspect of OOPS known as **Data Hiding**. The access restriction to the class members is specified by the labeled public, private, and protected sections within the class body. **The keywords public, private, and protected are called access specifiers.** Access specifiers in C++ class defines the access control rules.

C++ has 3 new keywords introduced:

public
private
protected

Public keyword

1. Public:

All the class members declared under the public specifier will be available to everyone.

The data members and member functions declared as public can be accessed by other classes and functions too.

The public members of a class can be accessed from anywhere in the program using the direct member access operator (.) with the object of that class.

Example:

```
class Circle{   public:  double radius;
                         double  compute_area() { return 3.14*radius*radius; }
};
 int main()
{
    Circle obj;
    obj.radius = 5.5;
    cout << "Radius is: " << obj.radius << "\n";
    cout << "Area is: " << obj.compute_area();
    return 0;
}
```

Private keyword

2. Private:

The class members declared as private can be accessed only by the member functions inside the class.

They are not allowed to be accessed directly by any object or function outside the class

Only the member functions or the friend functions are allowed to access the private data members of a class.

```
class Circle
{      private:   double radius;
       public:
        double  compute_area()
        {   // member function can access private
            // data member radius
            return 3.14*radius*radius;
        }};
 int main()
{      Circle obj;
         // trying to access private data member
    // directly outside the class
    obj.radius = 1.5;
    cout << "Area is:" << obj.compute_area();
    return 0;}
```

Protected keyword

3. Protected:

Protected access modifier is similar to private access modifier in the sense that it can't be accessed outside of it's class unless with the help of friend class, the difference is that the class members declared as Protected can be accessed by any subclass(derived class) of that class as well.

```
class Parent
{    protected:
    int id_protected;
};
 // sub class or derived class from public base class
class Child : public Parent
{
    public:    void setId(int id)
                {
        // Child class is able to access the inherited
        // protected data members of base class
                    id_protected = id;
    }
         void displayId()
    {cout << "id_protected is: " << id_protected << endl;    }
};
```

Characteristics of Access Specifiers

The accessibility of access specifiers in classes during inheritance is shown in Table.

Access Specifiers	Private	Protected	Public
Access in same class	√	√	√
Access in derived class	x	√	√
Access outside the class	x	x	√

Characteristics of Access Specifiers

Characteristics of access specifiers (private,public and protected)

- Private section of a class can have both data members and member functions, usually data members are made private for data security.
- It is not mandatory that private section has to declared first in the class and then the public section.
- If no member access specifier is specified then by default the members are private for the class.
- There may be any number of private, public or protected section in a class declaration.
- Protected specifier is used for declaring the class members which can be accessed by its own class and its derived class.