




UNITII

Classes

- Logical method to organise data and functions in a same structure.
- Also known as abstract data type (ADT).

- 
- **It's a User Defined Data-type.**
 - The Data declared in a Class are called Data-Members of the Class.
 - The Functions declared or defined in a Class are called Member-Functions of the Class.
 - The members of a Class can only be accessed through Objects of the class.

Syntax

Syntax:

```
class class_name
{
  access specifier:
    member1;
  access specifier:
    member function;

  ...
} [object_name];
```

for example

```
class a
{
  private;
  int x
  public :
    void show() {cout << a;}
};
```

Characteristics of a class

- The keyword `class` specifies abstract data type of type `class name`.
- The body of a class is enclosed with in braces and terminated by a semicolon
- The functions and variables with in the class are collectively called as members
- The members that have been declared as `private` can be accessed only from with in the class.
- Class definition is only a template and does not create any memory space for a class
- By default all the members are of type `private` .

Access Specifiers

- **Public**

- Any member declared under this specifier is Accessible from Outside the class, by using the object of the Class.

- **Syntax :-**

Class definition

```
{      public :  
        declarations or definitions  
};
```



- **Private**

- Any member declared under this specifier is Not Accessible from Outside the Class. The Private members are accessible to only the Member Functions and Friend Functions in the Class.

Syntax :-

Class definition

```
{    private :  
    declarations or definitions  
};
```

- **Protected**

Any member declared under this specifier is Not Accessible from Outside the class, by using the object of the Class.

Syntax :-

Class definition

```
{      protected :  
      declarations or definitions  
};
```


Structure	Class
Structure can be defined as a collection of dissimilar data items.	Class can be defined as combination of data items and functionality applied on that data
By default all members are public	By default all members are private
The size of the structure = size of the individual data items of a structure	The size of the structure = size of the individual data items of a class
Separate copy of data members are created for all structure variables	Separate copy of data members and only one copy of member functions are created for class
To create a structure variable use keyword struct	To create a class variable the keyword class is optional
It is Not possible to inherit structures	It is possible to inherit class
Structure are called as Passive data item	Classes are called as Active data items

Member function outside the class

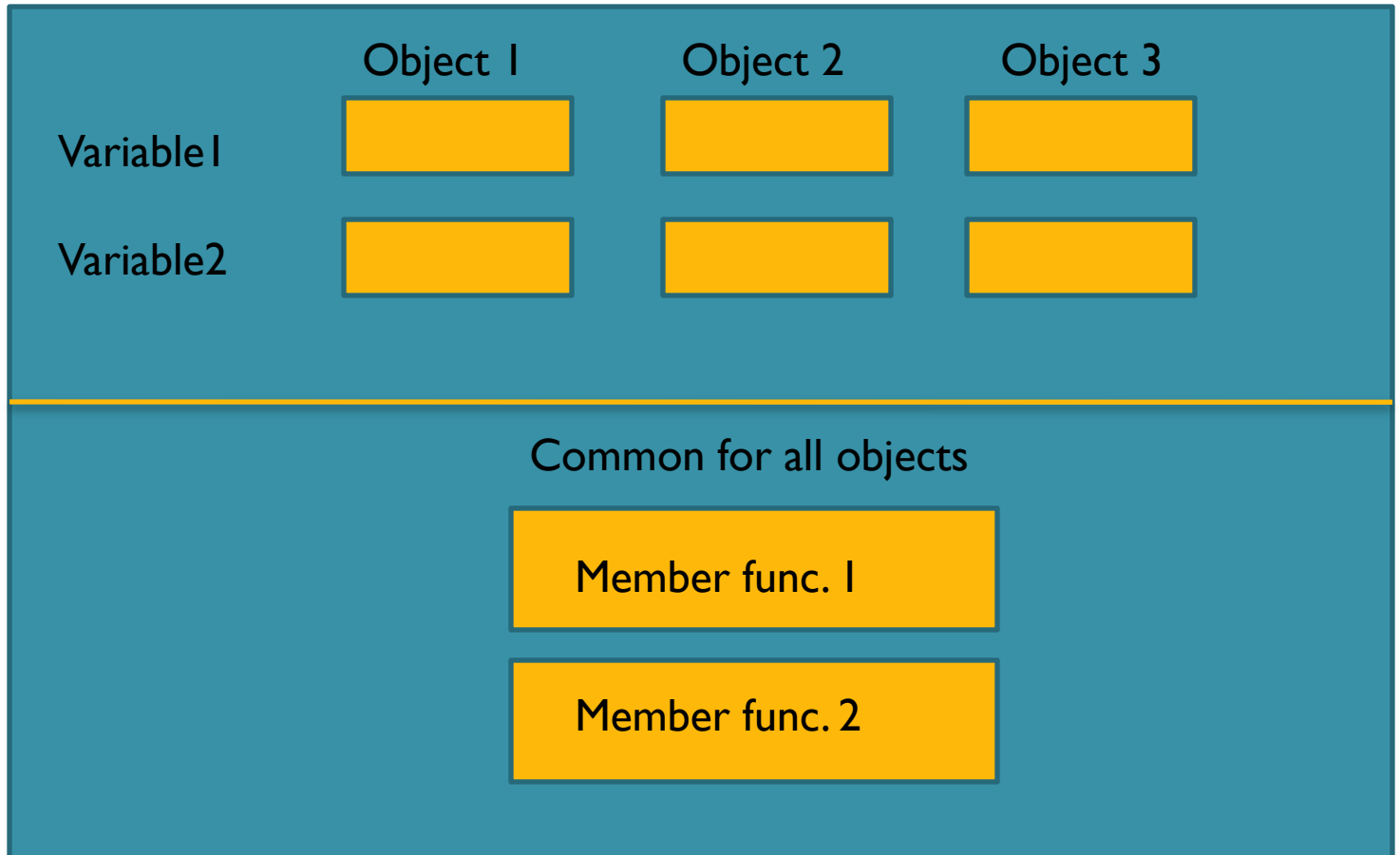
- To declare the member function of a class outside the class definition the function prototype declared within the body of a class and defined them outside the body of a class.

Objects

- *Object can be defined as an Instance of a class.*
- The process of creating objects (variables) of the class is called ***class instantiation***.
- In C++ the class variables are called as objects.
- The compiler allocates required memory for objects.

Memory allocation for objects

- For each object of a class a separate copy of data members and one copy of member functions is created.



Static Data Members

- One copy of the member, shared by all objects of the class.
- Visible only within class, but its lifetime is the entire program.
- By default initialized to 0.
- Also known as class variable.
- For Making data member static we require:
 - a) Declare with in class;
 - b) Define it out side the class

Static Member Functions

- A static member function can access only the static members of class.
- It is called using the name of the class.
- Syntax for calling static member function
`classname :: function-name;`



Array of objects

Friend Function

- Private members of a class can be accessed only by the member functions of the same class. But there are some cases where the private member needs to be accessed by members of other classes.
- This can be achieved by declaring a non-member function called ***Friend Function***

Note: Just Bcoz of friend function C++ is not fully object oriented.

Friend Functions

- Friend functions are Not member functions of any class, but they can access the private, protected and public members of the class.
- They are declared using “ **friend** “ keyword.
- They Don't belong to any specific Class.
- They can be invoked as ordinary functions, with out using any object.

- They can be defined outside the Class, here also they are defined like ordinary functions and Not like member functions.
 - The access specifiers does not effect the visibility of Friend Functions.
 - Usually a friend function receives the object as arguments.

Syntax :-

```
friend return-type function-name ( input-arguments ) ;
```

- It is not affected by the access control keywords.

- We can also declare all the member function of a class as friend function of another class using “friend class”.

Eg

```
class first
{
    .....
    friend class second;
};
class second
{
    .....
    .....
}
```