**Sunbeam Infotech**

**DAC : CPP - Day 2**

* class and object
* Scope resolution operator
* this pointer
* Namespace
* Constructor
* Types of Constructor
* Array of objects

==============================================

**\* Class and Object**

**Class:**

-Class is collection of data member and member function.

-Class represents set/group of such objects which is having common structure and common

behavior.

-class is logical entity.

- Class contain (Nested Type[enum,class,structure,union],Data Members,Member Function]

**Object :**

- Instance of class is object

-An entity, which get space inside memory is called object.

-Object is used to access data members and member function of the class.

-Process of creating object from a class is called instantiation.

**- Object has**

1. State

- Value stored inside object is called state of the object.

- Value of data member represent state of the object.

2. Behavior

- Set of operation that we perform on object is called behavior of an

object.

- Member function of class represent behavior of the object.

3. Identity

- Value of any data member, which is used to identify object uniquly

is called its identity.

- If state of object is same the its address can be considered as its

identity.

-Member function do not get space inside object.

-If we create object of the class then only data members get space inside object. Hence size of object is depends on size of all the data members declared inside class.

-Data members get space once per object according to the order of data member declaration.

-Structure of the object is depends on data members declared inside class.

-Member function do not get space per object rather it gets space on code segment and all the objects of same class share single copy of it.

-Member function's of the class defines behavior of the object.

**\* Data Members**

-Data member is also called as field, attribute, property etc.

**\*Member Functions**

-A function implemented inside class scope is called member function

-Member function is also called as method, operation, behavior or message.

**\* Scope Resoultion Operator**

**\*this Pointer**

-To process state of the object we should call member function on object. Hence we must define member function inside class.

-If we call member function on object then compiler implicitly pass address of that object as a argument to the function implicitly.

-To store address of object compiler implitly declare one pointer as a parameter inside member function. Such parameter is called **this pointer.**

-this is a keyword. "this" pointer is a constant pointer.

-this is used to store address of current object or calling object.

**This** pointer is used to represent the address of an object inside a member function.

**For example**, consider an object ***obj*** calling one of its member function say **fn()** as ***obj.fn()*.** Then, **this** pointer will hold the address of object *obj* inside the member function fn(). The **this** pointer acts as an [implicit argument](http://stackoverflow.com/questions/3057859/whats-the-difference-between-explicit-and-implicit-assignment-in-c) to all the member functions.

-Following functions do not get this pointer:

1. Global Function

2. Static Member function

3. Friend Function.

**Uses of this Keyword:**

**1) return object :**

return \* this;

inside a member function will return the object that calls the function

### **2) Distinguish Data Members**

### **this->a=a;**

### **this->b=b;**

**To understand ‘this’ pointer, it is important to know how objects look at functions and data members of a class.**

1. **Each object gets its own copy of the data member.**
2. **All-access the same function definition as present in the code segment.**

- Meaning each object gets its own copy of data members and all objects share a single copy of member functions.

**// returning this pointer from function : return \*this;**

**\* Namespace**

- to prevent name conflicts/ collision / ambiguity in large projects

- to group/orgaize functionally equivalent / related types toghter.

If we want to access value of global variable then we should use scope resolution operator ( ::)

-We can not instantiate namespace.

-It is designed to avoid name ambiguity and grouping related types.

-If we want to define namespace then we should use **namespace keyword.**

-we can not define namespace inside function/class.

-If name of the namespaces are same then name of members must be different.

We can not define main function inside namespace.

Namespace can contain:

1. Variable

2. Function

3. Types[ structure/union/class]

4. Enum

5. Nested Namespace

\* **Constructor**

* + It is a member function of a class which is used to initialize object.
  + Constructor is automatically called when object(instance of class) is created
  + Due to following reasons, constructor is considered as special function of the class:
    - Its name is same as class name
    - It doesn't have any return type.
    - It is designed to call implicitly.
    - In the life time of the object is gets called only once
  + We can not declare constructor static, constant, volatile or virtual. We can declare constructor inline.

**\* Types of Constructor:**

* + - **Default constructor**
      * If we do not define constructor inside class then compiler generates default constructor for the class.
      * Compiler generated default constructor is parameterless.
    - **Parameterless constructor** 
      * also called zero argument constructor or user defined default constructor
      * If we create object without passing argument then parameterless constructor gets called
    - **Parameterized constructor**
      * If we create object, by passing argument then paramterized constructor gets called
      * It is possible to pass arguments to constructors.
      * Typically, these arguments help initialize an object when it is created.
      * To create a parameterized constructor, simply add parameters to it the way you would to any other function. When you define the constructor’s body, use the parameters to initialize the object.

**\* Constructor Overloading**

**\* Array of objects**

- The array of type class contains the objects of the class as its individual elements.

- Thus, an array of a class type is also known as an array of objects.

- An array of objects is declared in the same way as an array of any built-in data type.

The syntax for declaring an array of objects is

class\_name array\_name [size] ;

**Example on array of objects**