

C++ Programming

Trainer : Rohan Paramane

Email: rohan.paramane@sunbeaminfo.com



Namespace

- To prevent name conflicts/ collision / ambiguity in large projects
- to group/organize functionally equivalent / related types together.
- 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

Note :

- If we define member without namespace then it is considered as member of global namespace.
- If we want to access members of namespace frequently then we should use using directive.



Scope Resolution Operator (::)

- :: operator is used to bind a member with some class or namespace.
- It can be used to define members outside class.
- Also used to resolve ambiguity.
- It can also be used to access global members.
 - Example :- ::a =10; access global var.
- Scope resolution Operator is used to :
 - to call global functions
 - to define member functions of class outside the class
 - to access members of namespaces



Data Members and Member Functions

Data Members

- Data members of the class are generally made as private to provide the data security.
- The private members cannot be accessed outside the class.
- So these members are always accessed by the member functions.

Member Functions

- Member functions are generally declared as public members of class.
- Constructor : Initialize Object
- Destructor : De-initialize Object
- Mutators : Modifies state of the object
- Inspectors : Don't Modify state of object
- Facilitator : Provide facility like IO



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 implicitly 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.
- The invoking object is passed as implicit argument to the function.
- *this* pointer points to current object i.e. object invoking the member function.
- Thus every member function receives *this* pointer.
- Following functions do not get this pointer:
 1. Global Function
 2. Static Member function
 3. Friend Function.



Constructor

- It is a member function of a class which is used to initialize object.
- Constructor has same name as that of class and don't have any return type.
- Constructor get automatically called when object is created i.e. memory is allocated to object.
- If we don't write any constructor, compiler provides a default constructor.
- Due to following reasons, constructor is considered as special function of the class:
 1. Its name is same as class name.
 2. It doesn't have any return type.
 3. It is designed to call implicitly.
 4. In the life time of the object , it gets called only once per object and according to order of its declaration.



Types of Constructor

- Parameterless constructor
 - also called zero argument constructor or user defined default constructor
 - If we create object without passing argument then parameterless constructor gets called
 - Constructor do not take any parameter
- Parameterized constructor
 - If constructor take parameter then it is called parameterized constructor
 - If we create object, by passing argument then parameterized constructor gets called
- Default constructor
 - If we do not define constructor inside class then compiler generates default constructor for the class.
 - Compiler generated default constructor is parameterless.



Facts About Constructor

- We can not call constructor on object, pointer or reference explicitly. It is designed to call implicitly.
- We can not declare constructor static, constant, volatile or virtual. We can declare constructor only inline.
- Constructor overloading means inside a class more than one constructor is defined.
- We can have constructors with
 - No argument : initialize data member to default values
 - One or more arguments : initialize data member to values passed to it
 - Argument of type of object : initialize object by using the values of the data members of the passed object. It is called as copy constructor.



Destructor

- It is a member function of a class which is used to release the resources.
- It is considered as special function of the class
 - Its name is same as class name and always precedes with tild operator(~)
 - It doesnt have return type or doesn't take parameter.
 - It is designed to call implicitly.
- Destructor calling sequence is exactly opposite of constructor calling sequence.
- Destructor is designed to call implicitly.
- If we do not define destructor inside class then compiler generates default destructor for the class.
- Default destructor do not de allocate resources allocated by the programmer. If we want to de allocate it then we should define destructor inside class.



Other Member functions of class Setter & Getter

- **Mutator/setter :**

- If we want to modify state of object i.e value of a private data member of the class outside the class using object then we should write a mutator.
- It is recommended to start the mutator function name with set followed by data member name which will accept a single argument to change the respective single data member value.

- **Inspector/getter :**

- If we want to read the state of object i.e value of a private data member of the class outside the class using object then we should write a Inspector
- It is recommended to start the inspector function name with get followed by data member name which will return the respective single data member value.

- **Facilitator**

- Any member function of a class that deals with all the data members of class and which are used to perform business logic operations are called as facilitators



Thank You

