UNIT -IV **Object Oriented Programming with C++** **CHAPTER 14**

Classes and objects

1.Define Class in C++.

- Class is a way to **bind** the **data** and its associated **functions** together.
- It is used to create **user defined data** type.

2.Write the syntax of Declaration of a class or How to Declare a class in C++?

General Form

```
Class class_name
  {
        private:
        variable declaration
        function declaration
        protected:
        variable declaration
        function declaration
        public:
        variable declaration
        function declaration
  };
```

- A class is defined in C++ using the keyword **class** followed by the **name of the class.**
- The body of a class is enclosed within **braces** and is terminated by a **semicolon ;**
- The **class body** contains the declaration of **variables(data)** and **functions**
- The class body has three access specifiers (visibility labels) viz., **private , public and protected**

3.What is the use of Data Hiding in C++?

- The members and functions declared under **private** are not accessible by **members outside the class**, this is referred to data hiding.

4.What are the three access specifiers of class members? (or) List out the accessibility levels in C++.

- The key words **public, private**, and**protected** are called access specifiers.

The Private Members

- It can be accessed only **within** the class by **class members** and **friend functions**.
- It **cannot** be accessed from **outside** the class.
- It is a **default** access specifier.

The Protected Members

- It can be accessed from **within** the class by class members and from the members of the **inherited classes.**
- It **cannot** be accessed from **outside** the class.

The Public Members

- It can be accessed from **within** the class by class members.
- It **can** be accessed from **outside** the class by using **objects.**

5.Differentiate between private and public specifier in C++. (ref. Q4)

6.How to Define class members ?or .What are data members and member functions of a class? or What does a class comprise of? What are class (called)member? Differentiate between Data Members and Member functions.

- Class Members are classified as **Data Members** and **Member functions.**

Data members

- Data members are the **data variables** that represent the **features** or **properties** of a class.
- Data members are also called as **attributes**.
- **Separate space** is allocated for **member variables** when each object is created

Member functions

- Member functions are the **functions** that perform **specific tasks** in a class.
- Members functions are called as **methods**,
- **No separate space** is allocated for **member functions** when the objects are created.

7.What are the methods of defining methods (functions) in a class?

Two types.
(1) Inside the class definition (inline)
(2) Outside the class definition (outline)

(1) Inside the class definition (inline function)

- A member function is **defined inside** a class,
- It behaves like **inline** functions.
- These are called Inline member functions.

(2) Outside the class definition (outline function)

- A Member function **defined outside** the class.
- It behaves like **normal** function definition .
- It is called as **outline** member function or **non-inline** member function.

- **Scope resolution operator (::)** is used for this purpose.

The syntax for defining the outline member function is
return_typeclass_name::function_name (parameters)
{
function definition
}
Ex.
void add :: display()
{
}

8.Differentiate structure and class though both are user defined data type.

structure	Class
User defined data type	User defined data type
struct keyword used	**class** keyword used
public as default	**private** as default

9.What is the difference between the class and object in terms of oop?

Class	object
Class is a way to bind the data and its associated functions together.	Represents data and its associated function together into **a single unit.**
User defined data type	They are **instances** of Class (**class variable**)
Class represents a **group of similar objects**	Basically an **object** is created **from a class**.

10.What are the methods of creating class object in C++?

Objects can be created in two methods,
(1) Global object (2) Local object

Global Object

- If an object is declared **outside a**ll the function bodies or
- By**placing** their names immediately after the **closing brace of the class** declaration then it is called as Global object.
- These objects can be used by any function in the program

Ex. class xyz
{
}**a;**

Local Object

- If an object is declared **with in** a function then it is called local object.
- It cannot be accessed from **outside** the function.

Ex.
class xyz
{
};
main()
{
xyz a; }

11.How memory is allocated to the objects of a class.?

The member functions are created and placed in the memory space only when they are defined as a part of the class specification.

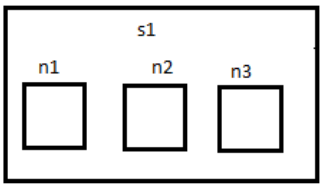
•**No separate space** is allocated for **member functions** when the objects are created.

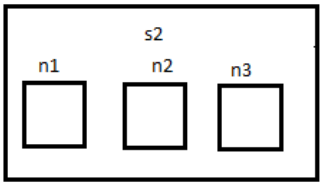
•**Separate space** is allocated for **member variables(data)**when each object is created.

• Separate memory allocations for the objects are essential because the member variables will hold different data values for different objects.

12.What is the size of the objects s1, s2?

class sum
{ int n1,n2;
public:
void add()
{int n3=10;n1=n2=10;
} } s1,s2;





S1 = 4+4+4 = 12 Bytes S2 = 4+4+4 = 12 Bytes

How are the class members accessed? Or

13.How class members are referencing in C++?

- The members of a class are referenced (accessed) by using the **object of the class** followed by the **dot (membership) operator** and the **name of the member.**

The general syntax for calling the member function is:
Object_name .function_name(actual parameter);
Ex. **stud.execute();**

14.Define Array of objects.

- An array which contains the class type of element is called array of objects.
- It is declared and defined in the same way as any other type of array.

Class **stu**
{,…. ……
} **a[3];**

Here,a[0],a[1] , a[2] are the array of object of the class stu.

15.What is nesting of member functions in C++?

- A member function can call another member function of the**same class** directly **without** using the **dot** operator.
- This is called as nesting of member functions.

16.What are the ways of objects can be passed as arguments?

Objects can also be passed in both ways

(1) Pass By Value
(2) Pass By Reference

17.Explain how the objects can be passed in pass by value method.

- When an **object is passed** as argument by **value** ,**new object** is created in **formal** parameter.
- The objects from actual parameter **copied** in to formal parameter.
- **Any changes** made to the object inside the **formal parameter ,do not affect** the actual object.

```
#include <iostream>
using namespace std;
class sample
{
private:
intnum;
public:
void pass(sample obj)
{
 obj1.num=100;
cout<<"\n\n Changed value of object1 "<<obj1.num;
}
void print( )
 {
cout<<num;
 }
};
int main()
{
sample s1,s2;
s2.pass(s1);
s1.print();
return 0;
}
```

18.Explain how the objects can be passed in pass by reference method.

- When an **object is passed** as argument by **reference ,object** in **formal** parameter **alias** the object of actual parameter.
- The formal parameter works directly on the actual object.
- **Any changes** made to the object inside the **formal parameter ,affect** the actual object.

```
#include <iostream>
using namespace std;
class sample
{
private:
intnum;
public:
void pass(sample &obj)
{
 obj1.num=100;
cout<<"\n\n Changed value of object1 "<<obj1.num;
}
void print( )
 {
cout<<num;
 }
};
int main()
{
sample s1,s2;
s2.pass(s1);
s1.print();
return 0;
}
```

19.What is nested class?

- When **one** class become the member of another class then it is called Nested class.

20.What are the ways the classes can be nested?

Classes can be nested in two ways.
1. Defining a class within another class
2. Declaring an object of a class as a member to another class

Defining a class with in another

- When a class is declared with in **another class**,
- The **inner** class is called as **Nested cla**ss .
- The outer class is known as **Enclosing** class.
- Nested class can be defined in **private** as well as in the **public** section of the Enclosing class.

```
#include<iostream>
using namespace std;
class enclose
{
private:
int x;
        class nest
         {
        private :
        int y;
        public:
        void prn()
        {
          y=3;
        cout<<y;
        }
        };
        nest  n;
public:
void square()
 {
n.prn();
 x=2;
cout<< x * x;
 }
};
```

```
int main()
{
enclose e;
e.square();
}
```

In the above program

- The inner class **nest** is defined inside the outer class **enclose**.
- nest is accessed by enclose by creating an object of nest

Declaring an object of a class as a member to another class

- Whenever an object of a class is declared as a member of another class it is known as a **container class.**
- In the container-ship the object of one class is declared in another class.

```
#include<iostream>
using namespace std;
class outer
{
int data;
public:
void get();
};
class inner
{
int value;
outer ot;
public:
voidgetdata();
};
Intmain()
{
Inner  x;
}
```

In the above program

- class**outer** and **inner** are defined separately.
- But both the classes are**connected** by the object '**ot**' which is a member of class inner.

21.What is Container class?

- Whenever an **object of a class** is declared as a member of **another class** it is known as a **container class.**

22.Write the example how will you dynamically initialize objects?

- When the **initial** values are provided **during runtime** is called dynamic initialization.

Example

```
#include<iostream>
using namespace std;
class add
{
public:
void add(int a, int b)
{
cout<<a+b;
}
};
int main( )
{
Int x,y;
cin>>x>>y;
add(x,y);  // dynamic initialization
}
```

Introduction to Constructors& Destructors

23.What is a constructor?

- Classes include special member functions called as constructors.
- The name of the constructor must be same as that of the class
- Used to **allocate memory space** and **initialize** the **data member** of the class object
- It has **no return type**
- Constructor can **have parameter** list
- Constructor **can be overloaded**
- They **cannot be inherited** but a derived class can call the base class constructor

Syntax:

```
Class class_name
{
public:
class_name()
{
}
};
```

24.What are the functions of constructors?

1) To **allocate memory space** to the object .
2) To **initialize** the **data member** of the class object

25.What are the Characteristics of Constructors? Or What are the rules of constructors?

- The name of the constructor must be same as that of the class
- Used to **allocate memory space** and **initialize** the **data member** of the class object
- The constructor is **executed automatically** when the **object is created**
- **No return** type for constructor
- A constructor can **have parameter** list
- The constructor **can be overloaded**
- They **cannot be inherited** but a derived class can call the base class constructor
- Default constructor is **public** member function

26.Define constructor overloading. Types of constructor.

- Function overloading can be applied for constructors, are called **constructor overloading** .

Three types of constructors, they are
1.Non-Parameterized constructor or default constructor
2.Parameterized constructor
3.copy constructor

27.Define Non-Parameterized constructor or default constructor.

- A constructor that accepts **no parameter** is called default constructor.
- In the absence of user defined constructor, the compiler automatically **generates** default **constructor** when the **object is created**
- A default constructor implicitly as an **inline public member**.

Advantage :

- Default constructors are very useful to create objects without having specific initial value.
- It is also used to create array of objects.

Ex.

```
class add
{
        public:
        add();
        {
        }
};
int main()
{
add x;
}
```

28.Define Parameterized Constructors

- A constructor with arguments is called parameterized constructor .
- This type of constructor helps to **create objects** with **different initial values.**
- This is achieved by **passing parameters** to the **function**.
- Parameterized constructor can **havedefault arguments**.

Ex.

```
class add
{
public:
        add(int a)
        {
        }
};
int main()
{
add x(5);
}
```

29.What are the ways to create an object using parameterized constructor?

There are two ways to create an object using parameterized constructor 1)Implicit call 2) Explicit call

Implicit call

- In this method ,the parameterized constructor is invoked automatically whenever an object is created.

For example **simple s1(10,20);**

- In this for creating the object **s1** parameterized constructor is automatically invoked.

Explicit call

- In this method , the object can be created and initialized
  For example **simple s1=simple(10,20);**
- An explicit call to constructor creates temporary instance(object).

30.Explain copy constructor with an example.

- Copy Constructor is a type of constructor which is used to create a copy of an already existing object of a class type.
- While defining copy constructor the argument (object) should be passed only by **reference** not by **value** method.

Ex, **simple (simple&x)** - **simple**- class name.

Calling copy constructors

A copy constructor is called

1.When an object is passed as a parameter to any of the member functions
Example:**void simple::putdata(simple x);**
2.When a member function returns an object
Example:**simple getdata()** { }
3.When an object is passed by reference to an instance of its own class
Example:**simples1, s2(s1);** // s2(s1) calls copy constructor

31.How constructors are executed (Order of constructor invocation) ?

- The constructors are executed in the order of the object declared.

For example
Test t1; Test t2;
The order of constructor execution is first for t1 and then for t2.
Sample s1,s2,s3 ; //The order of construction is s1 then s2 and finally s3

- But if a class containing an object of another class as its member then that class constructor is executed first.

32.Define Distructors.

- Classes include special member functions called as destructor .
- The destructor has the same name as the class tag prefixed by the **~ (**tilde)
- The destructor is executed automatically when the control reaches the **end of class scope**
- Destructor function **removes the memory** of an object

Syntax:

```
class  simple
{
Public:
~simple()
{

}
};
```

33.What are the rules or Characteristics of Destructors?

- The destructor has the same name as the class tag prefixed by the **~ (**tilde)
- The destructor is executed automatically when the control reaches the **end of class scope**
- Destructor function **removes the memory** of an object
- It has **no return type**
- Destructor **cannot have parameter** list
- Destructor **canno**t be **overloaded**
- They **cannot** be **inherited**

34.Write down the importance of Destructor.

- The purpose of the destructor is to free the resources.
- A destructor function **removes** the memory of an object which was allocated by the constructor at the time of creating a object.

35. Why it is considered as a good practice to define a constructor though compiler can automatically generate a constructor ?

- Default constructors are automatically generate a constructor
- It simply allocates memory for the object.
- It is also used to create array of objects.

36.Mention the differences between constructor and destructor.

Constructor	Destructor
The name of the constructor must be same as that of the class	The destructor has the same name as the class tag prefixed by the **~ (**tilde)
The constructor is **executed automatically** when the **object is created**	The destructor is executed automatically when the control reaches the **end of class scope**
Used to **allocatememory space** and **initialize** the **data member** of the class object	Destructor function **removes the memory** of an object
It has **no return type**	It has **no return type**
Constructor can **have parameter** list	Destructor can not**have parameter** list
Constructor **can be overloaded**	Destructor **canno**t be **overloaded**
They **cannot be inherited** but a derived class can call the base class constructor	They **cannot** be **inherited**
Default constructor is **public** member function	Destructor is **public** member function

class **class_name**
{
public:
class_name()
{
}
};
	class **simple**
	{
	Public:
	~simple()
	{
	
	}
	};

****