Page No.
Date

Assignment - 14

Q.1 What is mean by class variable and instance variable of class ?

- There are two types of characteristics in a class.

1. Static characteristics
2. Non-static characteristics

1. **static characteristics** - are called as class variable.
   - when we create an object memory for instance variable is allocated.
   - for eg. - we declared no in Demo class.
   - we can initialize it & access with class:: scope resolution operator outside the class.

Example :

```
class Demo{
static int no;
};
```

Datat. class_name :: variable = value;

memory for class variable is allocated irrespective of No of object creation.

2. **Non-static** - These are Instance variables.

for eg → class Demo{
int no;
};

Q.2. What happens if we remove & operator in case of copy constructor?

—> • & operator used in copy constructor to give reference of other function to creates copy of that object.

• If we don't use & operator while creating a copy constructor we unable to create copy of other function.

• & compiler will generate syntax error.

Q.3. What is mean by default argument?

—> • A default argument is a value provided in a function declaration that is automatically assigned by a compiler if the calling function doesn't provide a value for the argument. In case of any value is passed, the default value is overwritten.

• Defaut value must be write at the end of function defination.

eg. # include < iostream >
using namespace std;

```
int sum ( int x, int y, int z=0, int w=0)
{
    return (x+y+z+w);
}

int main()
{
    cout << sum(10, 15) << endl ;    // 25

    cout << sum(10, 15, 5) << endl ; // 30

    cout << sum(10, 15, 5, 50 << endl ; //80

    return 0;
}

// 25
30
80
```

- We can call above sum() by giving 2 arguments or 3 arguments or 4 arguments.
- z and w is a default arguments.

Q.4. What is difference between static and non-static characteristics of class?

—>

• Non-static - also called as local variables.

- variable defined within a block or method or constructor is called local variables.

• static - Also called as global variables.

- when a variable is declared as a static, then a single copy of the variable is created and shared among the all objects at a class level.

Non-static -

• when we create object of a class memory for non-static characteristics gets allocated for each object seperately.

- Non-static characteristics called instance variables.

• static - Irrespective of object creation memory for static characteristics gets allocated.

- memory for static variables gets allocated only once.

• static - static characteristics of class are called as class variables.

- **Non-static** - when we create object memory for non-static characteristics gets allocated for each object seperately.

- Inside the constructor we initialize non-static characteristics.

- static - static characteristics should be initialize outside the class using scope resolution operator ::

- we can access static characteristics without creating object with the help of class name & scope resolution operator.

Q.5. Can we call member function using this pointer from constructor?

→ - Yes we can call member function using this pointer from a constructor.

Example - lets take a example

```
# include < iostream>
using namespace std;

class person {
public: string name;
        person ( string name)
    {
        this -> name = name;
    }
```

```
void Display ()
{
    cout << " My name is " << name << endl;
}
};
int main(){
    person person ("omkar");
    person.Display();        // My name is omkar
}
```

6. What is lifetime of static characteristics of class ?

→ The life time of static characteristics is Throughout the program.

1. Life Duration - Throughout program.
2. shared across the instances
3. Access using class name & :: operator

7. Explain concept of parameterized constructor - with default argument's.

--> • We can use concept of default argument in case of constructor.

• Example :

```
class Demo
{   public:
    Demo ( int a = 10 , int b = 20 )
    {
        // parameterized constructor
        with default value /
        Argument
    }
};
```

• In above class Demo() is accept default accept a = 10 , b = 10 if object is created (defaut).

• a & b are initialize in parameterized constructor so it becomes parameterized constructor with default value.

Q.8 can we create the private static characteristics of class ? explain with example.

--

- Yes, we can create the static characteristics of a class but the created variable cannot be accessed outside the created class.

- static can only be accessed within the class.

- Example -

```
class Demo
{
    private:
        static int i;
};
int main()
{
    Demo :: i = 10; // can't allocated
}
```

Q.9 can we access private non-static characteristics of class from static method? Explain with example.

--> • No, we cant access the private non-static characteristics of class from static method, because static method is used only to access the static characteristics but non-static characteristics method can call both i.e static as well as non-static variables.

• Example :

```
class Demo
{
        private :
        int i ,j :

        static void fun ():
        {
          cout << " value of :" << sizeof (i) << endl;
          cout << " value of : " << sizeof (j) << endl;
        }
};
```

Q.10 How to initialize the static characteristics of class.

—> • We can initialize of static characteristics of class using class name & scope resolution operator (::) outside the class.

• But, need to declare the static variable inside the class

• Example :

```
class Demo{
    public :
        static int no;

    Demo(){
        //
    }
};
// Datatype class_name :: variable name = value;
int Demo :: no = 100;

int main()
{
    cout << " value of no : " << Demo :: no << "\n";
}
```