PAGE No.
DATE / /

Assignment - 14

Q.] What is meant by class variable & instance variable of class?

Ans In object oriented Programming & in order to achieve encapsulation we create a class

- A class consists of two important things characteristics of class & behaviours of class
- Characteristics of class are divid into two parts as static & non-static characteristics.

- Static characteristics of a class are called as class variables.
- Non-static characteristics of a class are called as instance variables.

- Static Memory for the static characteristics, gets allocated irrespective of object creation.
- Memory for static variables get allocated only once.

- Memory for non-static characteristics gets allocated for object seperately, when object of that class is created.
- Hence it is called instance variable of class.

Q2] What happens if we remove & operator in case of copy constructor?

Ans

1] The & operator inside the copy constructor is ~~constru~~ considered as a reference operator.

2] It refers to the object passed in the parameter.

3] If & operator is remove, it may lead to recursive calls.

Q3] What is ~~as~~ meant by default ~~g~~ argument?

Ans

1] There are two types of arguments in a function as a regular compulsory argument & a default ~~option~~ argument.

2] Default is a type of argument which is optional

3] If we skip the default argument parameter, while calling the function then its default value gets considered.

Example

```
~~class~~ #include <iostream.h>
  using ~~namespace~~ std;

float ~~fpercentage~~ calculate (float fmarks, float fout_of = 100.0)
{
    float fpercentage = ((fmarks / ~~100.0~~ fout_of) * 100);
    return fpercentage;
}
int main()
{
  ~~int~~ float fvalue = 0.0f;
```

```
float fans = 0.0f;

fans = calculate (86, 100);
cout << "The percentage is : " << fans << endl;

fans = calculate (86);   // 100.0 gets considered by default
cout << "The percentage is : " << fans << endl;

fans = calculate (320.0, 420.0)

Cout << "The percentage is : " << fans << endl;

return 0;
}
```

- In the above application, the calculate function accepts two parameters as marks & out of.
- The marks parameter is compulsory & the outof parameter is optional.
- If we skip the outof parameter, then its default value gets considered as 100.0.
- One more rule is that, default arguments should be end of function arguments; else (i.e it should be the last argument) else compiler will genenerate an error.

Q4] What is the difference between static & non-static characteristics of a class?

Ans

1] Class consits of two things as characteristics & behaviours

2] characteristics is of two types as static & non-static characteristics

3] Memory for non-static characteristics gets allocated when object of the class is created, & for each object seperately.

4] Memory for static characteristics gets allocated irrespective of object creation.
Memory for static characteristics gets allocated only once.

5] Non-static characteristics are called instance variables.

6] static characteristics are called class variables

7] Non-static characteristics are initialised inside class.

8] static characteristics are initialised outside class.

PAGE No.
DATE / /

Q5] Can we call member function using this pointer from constructor?

Ans

Q6] What is lifetime of static characteristics of class?

Ans Lifetime of static characteristics of class is for program.

Q7] Explain the concept of Parameterized Constructor with default arguments.

Ans

1] We can use the concept of default arguments in case of constructor.

2] If we create a parameterized constructor which uses default argument, then that type of constructor is called as parameterized constructor with default argument

Example:

```
Class Demo {
  public:
    int i,j;
    int inum1 = 0, inum2 = 0;

    Demo (int iValue1 = 10, int iValue2 = 20)
    {
      inum1 = iValue1;
```

```
    inum2 = iValue2;
  }
  void display()
  {
   cout<<" Value of inum1 is "<<inum1<<endl;
   cout<<" Value of inum2 is "<<inum2<<endl;
  };

int main()
{
   Demo obj1;
   void obj1.display();          // inum1 = 10, inum2 = 20

   Demo obj2(90);
   obj2.display();               // inum1 = 90, inum2 = 20

   Demo obj3(9000, 10000);
   Obj3.display();               // inum1 = 9000, inum2 = 10000

   return 0;
}
```

Q10] How to initialize static characteristics of class?

Ans 1] static characteristics are initialized outside the class using scope resolution operator

2] Static characteristics can also be accessed without creating object & using class name & scope resolution.

Example:

PAGE No.
DATE / /

```
class Demo
{
    Public :
        int iNum1 = 0 , iNum2 = 0 ;
        static int iNum3 ;

        Demo ( int iValue1 = 10, int iValue2 = 20)
        {
            iNum1 = iValue1 ;
            iNum2 = iValue2 ;
        }
};

int Demo :: iNum3 = 35 ;    // Initialising Static Variable.

int main ()
{
    cout << " Value of static variable is " << Demo :: iNum3

    Demo obj1 (10, 200);

    return 0 ;

}
```

PAGE No.
DATE / /

Q8] Can we create private static characteristics of class? Explain with example.

Ans

PAGE No.
DATE / /

~~Q 9]~~ ~~Can~~

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