DEPARTMEN OF COMPUTER SCIENCE AND ENGINEERING

CS6301 - PROGRAMMING AND DATA STRUCTURES

QUESTION BANK

UNIT II

1.What is a constructor ?

A constructor is a special member function whose task is to initialize the objects of its class. It is special because its name is the same as the class name.

The constructor is invoked whenever an object of its associated class is created. It is called constructor because it constructs the values f data members of the class.

2. How will you declare and define a constructor

Constructor declaration :

```
Class integer
 {
   int m,n;
    public :
              integer ( void )
 }
```

constructor definition

```
integer :: integer (void )
{
      m=0;
      n=0;
}
```

3. what are the characteristics of a constructor ?

- They should be declared in the public section.
- They are invoked automatically when the objects are created.
- They do not have return types, not even void and therefore, they cannot return values.
- They cannot be inherited, though a derived class can call the base class constructor.
- Constructors cannot be virtual.

4. what are the types of a constructor ?

- Default constructor
- Parameterized constructor. •

Copy constructor

5. What is a parameteized constructor ?

The constructors that can take arguments are called parameterized constructors.

```
Integer ( int x, int y )
{
        m=x;
    n=y;
}
```

6. What is a default constructor ?

A constructor that accepts no parameter is called default constructor.

```
Default ( )

{
}
```

7. What is a destructor ?

A destructor as the name implies is used to destroy the objects that have been created by a constructor. Like a constructor the destructor is a member function whose name is the same as the class name but is preceded by a tilde symbol.

```
~ integer ( )

{

}
```

8. what is Dynamic constructors

The constructors can also be used to allocate memory while creating objects. This will enable the system to allocate the right amount of memory for each object when the objects are not of the same size thus resulting in the saving of memory.

Allocation of memory to objects at the time of their construction is known as dynamic constructors. The memory is allocated with the help of the new operator.

9. What is function overloading? Give an example.

Function overloading means we can use the same function name to create functions that perform a variety of different tasks.
Eg: An overloaded add ( ) function handles different data types as shown below.

// Declarations

i. int add(int a, int b); //add function with 2 arguments of same type

ii. int add(int a, int b, int c); //add function with 3 arguments of same type
iii. double add(int p, double q); //add function with 2 arguments of different type

10. What is operator overloading?

C++ has the ability to provide the operators with a special meaning for a data type. This mechanism of giving such special meanings to an operator is known as Operator overloading. It provides a flexible option for the creation of new definitions for C++ operators.

11. List out the operators that cannot be overloaded.

_ Class member access operator (. , .*)

_ Scope resolution operator (::)

_ Size operator (sizeof) _

Conditional operator (?:)

13. What is the purpose of using operator function? Write its syntax.

To define an additional task to an operator, we must specify what it means in

relation to the class to which the operator is applied. This is done by Operator function , which describes the task. Operator functions are either member functions or friend functions. The general form is
return type classname :: operator (op-arglist)
{
function body
}
where *return type* is the type of value returned by specified operation.

Op-operator being overloaded. The *op* is preceded by a keyword operator. operator op is the function name.

14. Write at least four rules for Operator overloading.

_ Only the existing operators can be overloaded.
_ The overloaded operator must have at least one operand that is of user defined

data type.
_ The basic meaning of the operator should not be changed.
_ Overloaded operators follow the syntax rules of the original operators. They cannot be overridden.

15. How will you overload Unary & Binary operator using member functions?

When unary operators are overloaded using member functions it takes no explicit arguments and return no explicit values.

When binary operators are overloaded using member functions, it takes one explicit argument. Also the left hand side operand must be an object of the relevant class.

16. How will you overload Unary and Binary operator using Friend functions?

When unary operators are overloaded using friend function, it takes one reference argument (object of the relevant class)

When binary operators are overloaded using friend function, it takes two explicit arguments.

17. How an overloaded operator can be invoked using Friend functions?

In case of unary operators, overloaded operator can be invoked as
Operator op (x);

In case of binary operators, overloaded operator can be invoked as

Operator op (x , y)

18. List out the operators that cannot be overloaded using Friend function.

_ Assignment operator = _
Function call operator ( ) _
Subscripting operator [ ]
→_ Class member access operator

19. Explain basic to class type conversion with an example.

Conversion from basic data type to class type can be done in destination class. Using constructors does it. Constructor takes a single argument whose type is to be converted.

Eg: Converting int type to class type class
time
{

int hrs,mins;

public:
…….

```
Time ( int t) //constructor
{
hours= t/60 ; //t in minutes mins
=t % 60;
}
};
```

Constructor will be called automatically while creating objects so that this conversion is done automatically.

20. Explain class to basic type conversion with an example.

Using Type Casting operator, conversion from class to basic type conversion can be done. It is done in the source class itself.
Eg: vector : : operator double( )

```
{
double sum=0;

for(int I=0;I<size;I++)
sum=sum+v[ i ] *u[ i ] ;
return sqrt ( sum ) ;
}
```

This function converts a vector to the corresponding scalar magnitude.

21. Define inheritance ?

Inheritance is the process of creating new classes from the existing classes. The new classes are called derived classes. The existing classes are called base classes. The derived classes inherit all the properties of the base class plus its own properties. The properties of inheritance does not affect the base classes.

22. What are the types of inheritance ?

- Single inheritance •
Multiple inheritance
- Hierarchial inheritance •
Multilevel inheritance. •
Hybrid inheritance
- Multipath inheritance

23. What are the advantages of using a derived class ?

- New classes can be derived by the user from the existing classes without any modification.

- It saves time and money.
- It reduces program coding time.
- It increases the reliability of the program.

24.Define multiple inheritance

If a derived class is derived from more than one base class, then it is called multiple inheritance.

hierarchial inheritance

If two or more derived classes are derived from the same base class then it is known as hierarchial inheritance

multilevel inheritance

If a derived class is derived from another derived class then it is known as multilevel inheritance.

25. Define abstract class

A class is said to be an abstract class if it satisfies the following conditions :

- It should act as a base class
- It should not be used to create any objects

26. What is an virtual function ?

A member function whose definition can be changed during run time is called virtual function. The class which contains virtual function is called polymorphic class and it should be a base class.

Different versions for the virtual function should be present in different derived classes with same name as virtual function name

27. Define pure virtual function ?

Pure virtual function is a virtual function with no body. The general form is

Virtual void member-function-name( ) = 0

28. What are the properties of pure virtual function ?

- It has no definition in the base class.
- We cannot create object for the class containing pure virtual function.
- Pure virtual function can be called through derived class objects.
- It acts as an empty bucket which has to be filled by its derived classes.

29. What are the rules for virtual functions ?

- Virtual functions must be members of some class. •

They cannot be static members.

- They are accessed by using object pointers.
- A virtual function can be a friend of another class.
- We can have virtual destructors, but we cannot have virtual constructors.

30. Define Polymorphism

It is the property of the same object to behave differently in different contexts given the same message. A single function name can be used for various purposes and single operator is used to achieve multiple operations and the usage of either the function at the operator depends on the context in such cases.

31. Define Compile time polymorphism:

The compiler while compiling the program resolves the function call or the operator call. This is known as compile time polymorphism

32. Define Runtime polymorphism

During multiple inheritances if the appropriate member function could be selected while the program is running is known as Runtime polymorphism