

SRI SAI RAM ENGINEERING COLLEGE
DEPARTMENT OF ELECTRONICS AND COMMUNICATION

2 MARKS QUESTION WITH ANSWER

DATA STRUCTURES AND OBJECT ORIENTED PROGRAMMING IN C++
SUB.CODE: EC2202

1) Give the evolution diagram of OOPS concept.

Machine language

Procedure language

Assembly language

OOPS Pg: 4

2) Draw the structure of Procedure oriented language or typical organization of Procedure oriented language.Pg.7

3) What is Procedure oriented language?

Conventional programming, using high-level language such as COBOL, FORTRAN and C are commonly known as Procedure oriented language (POP). In POP number of functions is written to accomplish the tasks such as reading, calculating and printing.

4) Give some characteristics of procedure-oriented language.

- Emphasis is on doing things (algorithms).
- Larger programs are divided into smaller programs known as functions.
- Most of the functions share global data.
- Data move openly around the system from function to function.
- Employs top-down approach in program design.

Function-1 Function-2 Function-3

Function-4 Function-5

Function-6 Function-7 Function-8

Main program

dia Pg:5

5) Write any four features of OOPS.

- Emphasis is on data rather than on procedure.
- Programs are divided into objects.
- Data is hidden and cannot be accessed by external functions.
- Follows bottom -up approach in program design.

6) What are the basic concepts of OOPS?

- Objects.
- Classes.
- Data abstraction and Encapsulation.
- Inheritance.
- Polymorphism.
- Dynamic binding.
- Message passing.

7) What are objects?

Objects are basic run-time entities in an object-oriented system. They may represent a person, a place, a bank account, a table of data or any item that the program has to

handle. Each object has the data and code to manipulate the data and these objects interact with each other.

8) What is a class?

- The entire set of data and code of an object can be made a user-defined data type with the help of a class.
- Once a class has been defined, we can create any number of objects belonging to the classes.
- Classes are user-defined data types and behave like built-in types of the programming language.

9) What is encapsulation?

Wrapping up of data and function within the structure is called as encapsulation.

10) What is data abstraction?

The insulation of data from direct access by the program is called as data hiding or information binding.

The data is not accessible to the outside world and only those functions, which are wrapped in the class, can access it.

11) What are data members and member functions?

Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight, and cost and uses functions to operate on these attributes.

The attributes are sometimes called as data members because they hold information. The functions that operate on these data are called as methods or member functions.

Eg: `int a,b; // a,b are data members`

```
Void getdata ( )
```

```
{ cin>>a; }
```

```
// member function
```

12) what is dynamic binding or late binding?

Binding refers to the linking of a procedure to the code to be executed in response to the call. Dynamic binding means that the code associated with a given procedure call is not known until the time of the call at the run-time.

13) Write the process of programming in an object-oriented language?

- Create classes that define objects and their behavior.
- Creating objects from class definition.
- Establishing communication among objects.

14) Give any four advantages of OOPS.

- The principle of data hiding helps the programmer to build secure programs that cannot be invaded by code in other parts of the program.
- It is possible to have multiple instances of an object to co-exist without any interference.
- Object oriented programming can be easily upgraded from small to large systems.
- Software complexity can be easily managed.

15) What are the features required for object-based programming Language?

- Data encapsulation.
- Data hiding and access mechanisms.
- Automatic initialization and clear up of objects.
- Operator overloading.

16) What are the features required for object oriented language?

- Data encapsulation.
- Data hiding and access mechanisms.
- Automatic initialization and clear up of objects.
- Operator overloading.
- Inheritance.
- Dynamic binding.

17) Give any four applications of OOPS

- • Real-time systems.
- • Simulation and modeling.
- • Object-oriented databases.
- • AI and expert systems.

18) Give any four applications of c++?

- Since c++ allows us to create hierarchy-related objects, we can build special object-oriented libraries, which can be used later by many programmers.
- C++ are easily maintainable and expandable.
- C part of C++ gives the language the ability to get close to the machine-level details.
- It is expected that C++ will replace C as a general-purpose language in the near future.

19) What are tokens?

The smallest individual units in a program are known as tokens. C++ has the following tokens,

- Keyword
- Identifiers
- Constants
- Strings
- Operator

20) What are keywords?

The keywords implement specific C++ language features. They are explicitly reserved identifiers and cannot be used as names for the program variables or other user defined program elements.

Eg: go to, If, struct, else, union etc.

21) Rules for naming the identifiers in C++.

- Only alphabetic characters, digits and underscore are permitted.
- The name cannot start with a digit.
- The upper case and lower case letters are distinct.
- A declared keyword cannot be used as a variable name.

22) What are the operators available in C++?

All operators in C are also used in C++. In addition to insertion operator << and extraction operator >> the other new operators in C++ are,

- : Scope resolution operator
- :: * Pointer-to-member declarator
- >* Pointer-to-member operator
- * Pointer-to-member operator
- delete Memory release operator
- endl Line feed operator
- new Memory allocation operator
- setw Field width operator

23) What is a scope resolution operator?

Scope resolution operator is used to uncover the hidden variables. It also allows access to global version of variables.

Eg:

```
#include<iostream. h>
int m=10; // global variable m
void main ( )
{
int m=20; // local variable m
cout<<"m="<<m<<"\n";
cout<<": : m="<<: : m<<"\n";
}
```

output:

20

10 (: : m access global m)

Scope resolution operator is used to define the function outside the class.

Syntax:

Return type <class name> : : <function name>

Eg:

```
Void x : : getdata()
```

24) What are free store operators (or) Memory management operators?

New and Delete operators are called as free store operators since they allocate the memory dynamically.

New operator can be used to create objects of any data type.

Pointer-variable = new data type;

Initialization of the memory using new operator can be done. This can be done as,

Pointer-variable = new data-type(value)

Delete operator is used to release the memory space for reuse. The general form of its use is

Delete pointer-variable;

25) What are manipulators?

setw, endl are known as manipulators.

Manipulators are operators that are used to format the display. The endl manipulator when used in an output statement causes a linefeed to be inserted and its effect is similar to that of the newline character "\n".

Eg: Cout<<setw(5)<<sum<<endl;

26) What do you mean by enumerated datatype?

An enumerated datatype is another user-defined datatype, which provides a way for attaching names to numbers, thereby increasing comprehensibility of the code. The syntax of an enum statement is similar to that of the struct statesmen.

Eg:

```
enum shape{ circle, square, triangle}
enum color{ red, blue, green, yellow}
```

27) What are symbolic constants?

There are two ways for creating symbolic constants in C++:

- Using the qualifier constant.

- Defining a set of integer constants using enum keyword.

The program in any way cannot modify the value declared as constant in c++.

Eg:

```
Const int size =10;
```

```
Char name [size];
```

28)What do you mean by dynamic initialization of variables?

C++ permits initialization of the variables at run-time. This is referred to as dynamic initialization of variables.

In C++ ,a variable can be initialized at run-time using expressions at the place of declaration as,

```
.....
```

```
.....
```

```
int n =strlen(string);
```

```
.....
```

```
float area=3.14*rad*rad;
```

Thus declaration and initialization is done simultaneously at the place where the variable is used for the first time.

29) What are reference variable?

A reference variable provides an alias(alternative name) for a previously defined variable.

For example , if make the variable a reference to the variable , then sum and total can be used interchangeably to represent that variable.

Syntax :

```
Data-type &reference-name = variable-name
```

Eg:

```
float total = 100;
```

```
float sum = total;
```

30)What is member-dereferencing operator?

C++ permits to access the class members through pointers. It provides three pointer-to-member operators for this purpose,

:* To declare a pointer to a member of a class.

* To access a member using object name and a pointer to the member

->* To access a member using a pointer to the object and a pointer to that member.

31)what is function prototype ?

The function prototype describes function interface to the compiler by giving details such as number ,type of arguments and type of return values

Function prototype is a declaration statement in the calling program and is of the following

```
Type function_name(argument list); Eg float volume(int x,float y);
```

32)what is an inline function ?

An inline function is a function that is expanded in line when it is invoked. That is compiler replaces the function call with the corresponding function code.

The inline functions are defined as Inline function-header

```
{
function body
}
```

33) Write some situations where inline expansion may not work

- for functions returning values, if loop, a switch, or a goto exists
- for functions not returning values, if a return statement exists
- if function contain static variables
- if inline functions are recursive

34) what is a default argument ?

Default arguments assign a default value to the parameter, which does not have matching argument in the function call. Default values are specified when the function is declared.

Eg : float amount(float principle,int period,float rate=0.15)

Function call is

```
Value=amount(5000,7);
```

Here it takes principle=5000& period=7

And default value for rate=0.15

```
Value=amount(5000,7,0.34)
```

Passes an explicit value of 0.34 to rate

We must add default value from right to left

35) What are constant arguments ?

Keyword is const. The qualifier const tells the compiler that the function should not modify the argument. The compiler will generate an error when this condition is violated. This type of declaration is significant only when we pass arguments by reference or pointers

```
eg: int strlen( const char *p);
```

36) How the class is specified?

Generally class specification has two parts

- class declaration

It describes the type and scope of its member

- class function definition

It describes how the class functions are implemented

The general form is

```
Class class_name
```

```
{
```

```
private:
```

```
variable declarations;
```

```
function declaration;
```

```
public:
```

```
variable declaration;
```

```
function declaration;
```

```
};
```

37) How to create an object ?

Once the class has been declared, we can create variables of that type by using the classname

Eg: classname x; //memory for x is created

38) How to access a class member?

object-name. function-name (actual arguments)

```
eg:x.getdata(100,75.5);
```

39) How the member functions are defined ?

Member functions can be defined in two ways

- outside the class definition

Member function can be defined by using scope resolution operator::

General format is

Return type class_name::function-name(argument declaration)

```
{  
}
```

- Inside the class definition

This method of defining member function is to replace the function declaration by the actual function definition inside the class. It is treated as inline function

Eg: class item

```
{  
int a,b ;  
void getdata(int x,int y)  
{  
a=x;  
b=y;  
};  
}
```

40) What is static data member?

Static variable are normally used to maintain values common to the entire class.

Feature:

- It is initialized to zero when the first object is created. No other initialization is permitted
- only one copy of that member is created for the entire class and is shared by all the objects
- It is only visible within the class, but its life time is the entire class type and scope of each static member variable must be defined outside the class
- It is stored separately rather than objects

Eg: static int count//count is initialized to zero when an object is created.

int classname::count;//definition of static data member

41) What is static member function?

A member function that is declared as static has the following properties

- A static function can have access to only other static member declared in the same class
- A static member function can be called using the classname as follows

classname ::function_name;

42) How the objects are used as function argument?

This can be done in two ways

- A copy of the entire object is passed to the argument
- only address of the objects is transferred to the function

43) what is called pass by reference?

In this method address of an object is passed, the called function works directly on the actual arguments.

44) Define const member

If a member function does not alter any data in the class, then we may declare it as const member function as

Void mul(int ,int)const;

45) Define pointers to member

It is possible to take the address of a member of a class and assign it to a pointer. The address of a member can be obtained by applying the operator & to a “fully qualified” class member name. A class member pointer can be declared using the operator ::* with the class name.

Eg: class A

```
{  
int m;  
public:  
void show( );  
};
```

pointer to member m is defined as

```
int A::*ip=&A::m;
```

A::*->pointer to member of A class

&A::m->address of the m member of A class

46) When the dereferencing operator ->* is used?

It is used to access a member when we use pointer to both the object and the member.

47) When the dereferencing operator .* is used?

It is used to access a member when the object itself is used as pointers.

48) Define local classes.

Classes can be defined and used inside a function or a block. such classes are called local classes. It can use global variables and static variables declared inside the function but cannot use automatic local variables.

Eg;

```
void test(int a)
```

```
{  
.....  
}
```

```
class student
```

```
{  
.....  
};
```

```
student s1(a);
```

```
}
```

49) Define 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 same as class name. The constructor is invoked whenever an object of its associated class is created. It is called constructor because it constructs the values of data members of the class

Eg:

```
integer Class
```

```
{
```

```
.....
```

```
public:
```

```
integer( );//constructo r
```

```
.....
```

}

50) Define default constructor

The constructor with no arguments is called default constructor

Eg:

Class integer

{

int m,n;

Public:

Integer();

.....

};

integer::integer()//default constructor

{

m=0;n=0;

}

the statement

integer a;

invokes the default constructor

51) Define parameterized constructor

constructor with arguments is called parameterized constructor

Eg;

Class integer

{ int m,n;

public:

integer(int x,int y)

{ m=x;n=y;

}

To invoke parameterized constructor we must pass the initial values as arguments to the constructor function when an object is declared. This is done in two ways

1.By calling the constructor explicitly

eg: integer int1=integer(10,10);

2.By calling the constructor implicitly

eg: Integer int1(10,10);

52) Define default argument constructor

The constructor with default arguments are called default argument constructor

Eg:

Complex(float real,float imag=0);

The default value of the argument imag is 0

The statement complex a(6.0)

assign real=6.0 and imag=0

the statement

complex a(2.3,9.0)

assign real=2.3 and imag=9.0

53) What is the ambiguity between default constructor and default argument constructor ?

The default argument constructor can be called with either one argument or no

arguments. When called with no arguments, it becomes a default constructor. When both these forms are used in a class, it cause ambiguity for a statement such as A a; the ambiguity is whether to call A::A () or A::A (int i=0)

54) Define copy constructor

A copy constructor is used to declare and initialize an object from another object. It takes a reference to an object of the same class as an argument

Eg: integer i2 (i1);

would define the object i2 at the same time initialize it to the values of i1.

Another form of this statement is

Eg: integer i2=i1;

The process of initializing through a copy constructor is known as copy initialization .

55) Define dynamic constructor

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

Eg:

Class string

```
{
char *name;
int length;
public:
string( )
{
length=0;
name=new char[ length +1];
}
void main( )
{
string name1(“Louis”),name3(Lagrange);
}
```

56) Define const object

We can create constant object by using const keyword before object declaration.

Eg: Const matrix x(m,n);

57) Define destructor

It is used to destroy the objects that have been created by constructor. Destructor name is same as class name preceded by tilde symbol (~)

Eg;

~integer ()

```
{
}
```

A destructor never takes any arguments nor it does it return any value. The compiler upon exit from the program will invoke it. new Whenever operator is used to allocate memory in the constructor, we should use delete to free that memory.

58) Define multiple constructors (constructor overloading).

The class that has different types of constructor is called multiple constructors

Eg:

```

#include<iostream. h>
#include<conio.h>
class integer
{
int m,n;
public:
integer( ) //default constructor
{
m=0;n=0;
}
integer(int a,int b) //parameterized constructor
{
m=a; n=b;
}
integer(&i) //copy constructor
{
m=i. m;
n=i.n;
}
void main()
{
integer i1; //invokes default constructor
integer i2(45,67); //invokes parameterized constructor
integer i3(i2); //invokes copy constructor
}

```

59) Write some special characteristics of 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, and they cannot return values
- They cannot be inherited, though a derived class can call the base class
- They can have default arguments
- Constructors cannot be virtual function

60) How the objects are initialized dynamically?

To call parameterized constructor we should pass values to the object i.e., for the constructor `integer(int a,int b)` it is invoked by `integer a(10,18)` this value can be get during run time. i.e., for above constructor

```

int p,q;
cin>>p>>q;
integer a(p,q);

```

61) Define Inline Function?

Inline function is defined as a function definition such that each call to the function is in effect, replaced by the statements that define the function. It is expanded in line when it is invoked. The general form is

```

inline function-header
{

```

```
function body
}
```

62) Explain return by reference with an example.

A function can also return a reference. Consider the following function

```
int & max( int &x , int &y)
{ if(x>y)
return x;
else
return y;
}
```

Since the return type of max () is int & the function returns reference to x or y (and not the values). Then a function call such as max (a , b) will yield a reference to either a or b depending on their values.

The statement

```
max ( a , b) = -1;
```

is legal and assigns -1 to a if it is larger, otherwise -1 to b.

63) What are Friend functions? Write the syntax

A function that has access to the private member of the class but is not itself a member of the class is called friend functions.

The general form is

```
friend data_type function_name( );
```

Friend function is preceded by the keyword 'friend'.

64) Write some properties of friend functions.

- Friend function is not in the scope of the class to which it has been declared as friend. Hence it cannot be called using the object of that class.
- Usually it has object as arguments.
- It can be declared either in the public or private part of a class.
- It cannot access member names directly. It has to use an object name and dot membership operator with each member name. eg: (A . x)

65) What are virtual functions?

A function qualified by the 'virtual' keyword is called virtual function. When a virtual function is called through a pointer, class of the object pointed to determine which function definition will be used.

66) Write some of the basic rules for virtual functions

- Virtual functions must be member of some class.
- They cannot be static members and they are accessed by using object pointers
- Virtual function in a base class must be defined.
- Prototypes of base class version of a virtual function and all the derived class versions must be identical.
- If a virtual function is defined in the base class, it need not be redefined in the derived class.

67) What are pure virtual functions? Write the syntax.

A pure virtual function is a function declared in a base class that has no definition relative to the base class. In such cases, the compiler requires each derived class to either define the function or redeclare it as a pure virtual function. A class containing pure virtual functions cannot be used to declare any object of its own. It is also known as "donothing"

function.

The “do-nothing” function is defined as follows:

```
virtual void display () =0;
```

68) What is polymorphism? What are its types?

Polymorphism is the ability to take more than one form. An operation may exhibit different behaviors in different. The behavior depends upon the type of data used.

Polymorphism is of two types. They are

- Function overloading
- Operator overloading

69) 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
```

```
//Function calls
```

```
add ( 3 , 4); //uses prototype ( i. )
```

```
add ( 3, 4, 5); //uses prototype ( ii. )
```

```
add ( 3 , 10.0); //uses prototype ( iii. )
```

70) 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.

71) List out the operators that cannot be overloaded.

- Class member access operator (. , .*)
- Scope resolution operator (::)
- Size operator (sizeof)
- Conditional operator (?:)

72) 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.

73) 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.

74) 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.

75) 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.

76) How an overloaded operator can be invoked using member functions?

In case of Unary operators, overloaded operator can be invoked as op object_name or object_name op

In case of binary operators, it would be invoked as Object . Operator op(y) where op is the overloaded operator and y is the argument.

77) 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)

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

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

79) What is meant by casting operator and write the general form of overloaded casting operator.

A casting operator is a function that satisfies the following conditions

- It must be a class member.
- It must not specify a return type.
- It must not have any arguments.

The general form of overloaded casting operator is
operator type name ()

```
{
..... // function statements
}
```

It is also known as conversion function.

80) 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.

81) 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.

82) Explain one class to another class conversion with an example.

Conversion from one class type to another is the combination of class to basic and basic to class type conversion. Here constructor is used in destination class and casting operator function is used in source class.

Eg: objX = objY

objX is the object of class X and objY is an object of class Y. The class Y type data is converted into class X type data and the converted value is assigned to the obj X. Here class Y is the source class and class X is the destination class.