

Noorul Islam College of Engineering, Kumaracoil
Department of Computer Applications
Object Oriented Programming(MC1652)
2 Mark and 16 Mark Questions & Answers

1) How do you overload all operator using friend function?

We can overload an operator for a class by using a nonmember function, which is usually a friend of the class. Since a friend function is not a member of the class, it does not have a 'this' pointer. Therefore an overload friend operator function is passed the operands explicitly. This means that a friend function that overloads a binary operator has two parameters, and a friend function that overloads a unary operator has one parameter. When overloading a binary operator using a friend function, the left operand is passed in the first parameter and the right operand is passed in the second parameter. For example
friend myclass operator +(myclass m1, myclass m2);

2) What are the restrictions on friend operator functions?

There are some restrictions that apply to friend operator functions.

(i) We may not overload the =, (), [], or ~ operators.

(ii) When overloading the increment or decrement operations, we will need to use a reference parameter.

3) Why should we pass the argument as references in friend operator function?

Since the friend functions do not have this pointers, the operand is passed by value as a parameter. So the friend functions has no way to modify the operand. By specifying the parameter to friend operator function as a reference parameter, any changes made to the parameter inside the function affect the operand that generated the call.

4) What is the error in 100 + ob (ob is all object)? How do you overcome the error?

In this case, an integer appears on the left. Since an Integer is a built-in type, no operation between an integer and an object of ob 's type is defined. Therefore the compiler will not compile this expression.

The solution to this problem is to overload addition using a friend, not a member, function. When this is done, both arguments are explicitly passed to the operator function. Therefore, to allow both object + integer and integer + object, simply overload the function twice-one version for call situation. Thus, when we overload an operator by using two friend functions, the object may appear on either the left or right side of the operator.

5) How do you overload a subscripting operator? !

A subscripting operator [] function can be used to give subscripts a meaning for class objects. The general form is:

```
type class name:: operator [ ] (int i)
{
//...
}
```

Technically, the parameter does not have to be of type int, but an operator [] () function is typically used to provide array subscripting, and as such, an integer value is generally used. For example, given an object called o, the expression

```
o [3] ;
```

translates into this call to the operator [] () function:

```
o.operator[ ] (3);
```

6) How do you overload a function call operator ()?

The overloading of operator () creates a new way to call the function. Rather, we can create an operator function that can be passed an arbitrary number of parameters. When overloading (), we can use any type of parameters and return any type of value. We can also specify default arguments.

7) How do you overload a dereferencing operator ->?

The dereferencing operator ->, also called the class member operator, can be used as a unary postfix operator. The general usage is shown here:

```
Object->element;
```

Here, object is the object that activates the call. The operator ->() function must return a pointer to an object of the class that operator ->() operates upon. The element must be some member accessible within the object.

Note: An operator ->() function must be a member of the class upon which it works.

8) How do you overload a comma operator?

The comma is a binary operator. We can make an overloaded comma perform any operation we want. If we want the overloaded comma to perform in a fashion similar to its normal operation, then our version must discard the values of all operands except the right most. The right most value becomes the result of the comma operation.

9) What are the advantages of c++ over C?

The advantages of C++ over C are

- . Classes.

- . Function overloading.

- . Operator overloading.

These features enable us to create abstract data types, inherit properties from existing data types and support polymorphism thus making C++ a truly object oriented language.

10) What are the basic data types in C++?

The basic data types in C++ are

Char, int, float, double, void, bool, wchar_t

11) What are the various data types in C++?

User-defined type

Structure, Union, Class, Enumeration

Built-in-type

Integral type -> int, char, void

Floating type -> float, double

Derived type
array, function, pointer

12) Give the structure of c++ program?

Include files

Class declaration

Class function definition Main function program

13) What is meant by access modifiers?

The access modifiers used to control the variables that how variables may be accessed or modified. These precede the type modifiers and the types names they qualify.

14) Give some new operators that are introduced in C++.

:: scope resolution

::* pointer-to-member declaration

->*pointer-to-member operator

.* pointer-to-member operator

delete memory release operator

endl line feed operator

new memory Allocation operator

setw field width operator

15) Which operation refers to Bitwise operation?

The Bitwise operation refers to testing, setting, or shifting the actual bits in a byte or word, which correspond to the char and int data types and variants. You cannot use bitwise operations on float, double, long double, void, bool, or other, more complex types.

16) What is a class?

A class is a user defined data type that links data and code. It provides common properties and relationships for a group of objects. Here the data is referred as member variable and code is referred as member functions that operates on member variables.

17) Write the output of the following code?

```
int i, j;  
i = 1;  
for (j = 0; j < 4; j ++)  
{  
i = i<<1;  
printf("shift %d ; %d\n", j, i);  
}
```

```
i =I<<2; printf("shift % d",j,i)
```

The output of the above code is

shift 0 : 2

shift 1 : 4

shift 2 : 8
shift 3 : 16
shift 4 : 64

18) Write the general form of a class?.

```
class class-name
{
    private data and functions
    access-specifier:
    data and functions
    //...
    access-speicifier:
    data and functions
} object-list;
where object-list is optional.
```

19) What is meant by access specifier in a class? What are they?

The access specifier specify the access control of data members and member functions of a class from inside a class or outside a class. They are

- (i) Private
- (ii) Public and
- (iii) Protected

20) What happen when there is no access specifier specified for data members and functions of a class?

The private access specifier has been taken when there is no access specifier for data members and member functions in a class.

21) Explain the access control of various access specifiers.

A member of a class can be private, protected, or public:

. if it is private, its name can be used only by member functions and friends of the class in which it is declared.

.if it is protected, its name can be used only by member functions and friends of the class in which it is declared and by member functions and friends of classes derived from this class.

. if it is public, its name can be used by any function.

22) What are the restrictions to class members?

There are few restrictions that apply to class members.

- i. A non-static member variable cannot have an initializer.
- ii No member can be an object of the class that is being declared.
- iii. No member can be declared as auto, extern, or register.

23) How structure is differed from class in c++?

The difference between a structure and a class is that by default all members are public in a structure and private in a class.

24) What is an object?

An object is an entity that has the property of a class and has the data. It also can send and receive data from somewhere. It is also called as an instance of a class.

Example:

employee empl;

where emp 1 is an object of employee class.

25) What are instance variables?

The data objects are associated with each instance of the class, rather than with the class itself, we refer to them as instance variables.

26) What are the restrictions when we use c++ unions?

There are several restrictions that must be observed when we use C++ unions.

- (i) A union cannot inherit any other classes of any type.
- (ii) A union cannot be a base class.
- (iii) A union cannot have virtual member functions.
- (iv) No static variables can be members of a union.
- (v) A reference member cannot be used.
- (vi) A union cannot have as a member any object that overloads the = operator.
- (vii) No object can be a member of a union if the objects has an explicit constructor or ,destructor functions.,

27) Explain anonymous unions?

An anonymous unions does not include type name, and no objects of the union can be declared. Instead, an anonymous union tells the compiler that its member variables are to share the same location. However, the variables themselves are referred to directly, without the normal dot operator syntax.

28) What are the additional restrictions over anonymous unions?

- (i) The anonymous union must contain data only.
- (ii) No member functions are allowed.
- (iii) It cannot contain private or protected elements.
- (iv) Global anonymous unions must be specified as static.

29) What is meant by friend function ?

A friend function is a non member function of a class that access the 'all members of a class even private. A friend function has access to all private and protected members of the class for which it is a friend. The keyword friend is used to declare a function as a friend function.

30) What are the advantages of friend functions?

- (i) The friend functions are very useful when we are overloading certain types of operators.

(ii)The friend functions make the creation of some types I/O functions easier.

(iii)The friend functions may be desirable when two or more classes may contain members that are interrelated relative to other parts of your program.

31) What are the restrictions over friend functions?

There are two important restrictions over friend functions.

(i) A derived class does not inherit friend functions.

(ii)The friend functions may not have a storage-class specifier.

That is, they may not be declared as static or extern.

32) What are the characteristics of friend function?

. It is not in the scope of the class to which it has been declared as a friend.

. It cannot be called using the object of that class.

. Unlike member functions, it cannot access the member names directly and has to use an object name and dot membership with each member name.

.It can be declared either in the public or the private part of a class without affecting its meaning.

33) What is the difference between the interface and the implementation of a class?

The interface consists of the member function prototype and class implementation contains the definitions of the member functions.

34) What is the difference between a class member function and application function?

A class members function is part of the class, so it has access to the class's private parts but an application function is declared outside the class, and so it does not have access to the class's private parts.

35) What is friend class? What is its restriction?

A friend class is a class and all of its member functions have access to the private member defined within the other class. It only has access to names defined within the other class. It does not inherit other class. Specifically, the members of the first class do not become members of the friend class.

36) What is an inline function?

An inline function is a short function that is expanded in line at the point of each invocation rather than it is actually called. To declare a function as in line function, the inline keyword is used.

For example

```
inline int max (int a, int b)
{
return a>b? a : bi
}
```

37) When do we make inline functions? Why?

We can make a function as inline function when a function contains very less code. Since, every call of it place the entire code in it calling place, the size of the function should be very less.

38) What is advantage and disadvantage of inline functions?

The inline functions are very faster than other functions since no need to transfer the control from calling place to definition place and no need of extra actions while transfer the control.

If the body of the inline function is large, the program code become larger than its original size, since the duplication of every calling place.

39) What are the restrictions in inline functions?

- (i) The repetitive structure (loop), a switch or goto statement cannot be used.
- (ii) No static variables used.
- (iii) It cannot be recursive.

40) Differentiate an in line function and an ordinary function.

Inline function

- (i) The code is actually placed when it is invoked
- (ii) The execution is faster.
- (iii) The size should be small and simple.

Ordinary function

- (i) The control is transferred to the called function.
- (ii) It needs some extra time to transfer the control from calling place to called function and for extra actions.
- (iii) It may be larger and complex.

41) What are the two types of argument (or) parameter passing? Explain.

The C++ provides the following two techniques of parameter passing.

(i). call by value:

It copies the values of actual arguments into formal arguments of the function. In this case, the changes made to formal does not affect the actual arguments.

(ii). call-by-reference: The address of an actual argument is copied into the formal argument. Inside the function, the address is used to access the actual value. This means the changes made inside the function affect the actual argument.

42) What is constructor?

A constructor is a special function that is a member of a class and has the same name as that class. It is used to creating an instance (i.e., initializing object with values) if a class automatically. For example,

```
class stack
{
int stk[size];
int tos;
public:
```

```

stack(); //constructor
void push(int i);
int pop () ;
};
stack::stack ()
{ tos = 0;
}

```

43) What is destructor?

A destructor function is a special function that is a member of a class and has the same name as that class preceded by a dilt (~) character. It is automatically called when an object life IS came to end to destroy the previously allocated memory and for some other actions.

For example class stack

```

private:
    int stk[SIZE] ;
    int tos;
public:
    stack () { tos = 0 ;}
~ stack( ) { cout<<"\nstack destroyed";}
void push(int i) ;
int pop () ;
};

```

44) What is default constructor?

The default constructor is a constructor function, which has no arguments. If this default constructor is not defined by the user explicitly, It is created automatically by the compiler.

45) What is parameterized constructor?

A constructor function that has one or more arguments is called parameterized constructor. Typically, these arguments help initialize an object when t is created. For example

```

class myclass
{
private:
    int a, b;
public:
myclass (int i, int j.) { a =i ; b=j ;} void show () {cout<<a<< " "<<b; }
}

```

46) How many constructor can a class have?

A class can have any number of constructors, but all the constructors should be differed from each other by means of their parameter list.

47) How many destructors can a class have?

A class can have only one destructor.

48) What is meant by static member variable?

A variable that is part of a class, yet is not part of an object of that class, is called a static member variable. There is only one copy of static member is created instead of one copy per object.

49) What are the uses of static data members?

(i) It is used to provide access control to some shared resource used by all objects of a class.

ii) It is also used to keep track of the number of objects of a particular class type that are in existence.

50) How do you define a static data member of a class?

When you declare a static data member within a class, you are not defining it, i.e., you are not allocating storage for it. Instead you must provide a global definition for it elsewhere, outside the class.

This is done by redeclaring the static variable using the scope resolution operator to identify the class to which it belongs. This causes storage for the variable to be allocated.

51) What are the characteristics of static data member?

(i) Only one copy of a static data member exists for any number of objects for that class is created.

(ii) All static variables are initialized to zero before the first object is created.

(iii) All static variables must be defined explicitly elsewhere in outside of a class.

52) What is static member function?

A function that needs access to members of a class, yet does not need to be invoked for a particular object, is called a static member function. The static member function must also defined somewhere. The keyword static is not to be repeated in the definition of a static member.

53) What are the restrictions of the static member functions?

(i) The static member functions may only refer to other static members of the class.

(ii) The global functions and data may be accessed by a static member functions. .

(iii) A static member function does not have a this pointer.

(iv) There cannot be a static and non-static version of the same function.

(v) A static member function may not be virtual.

(vi) They cannot be declared as const or volatile.

54) When constructors and destructors are executed?

As a general rule, an object's constructor is called when the object comes into existence, and an object destructor is called when the object is destroyed.

A local object's constructor function is executed when the object's declaration statement is encountered. The destructor functions for local objects are executed, in the reverse order of the constructor functions.

Global objects have their constructor functions execute before `main()` begins execution. Global constructors are executed in order of their declaration, within the same file. Global destructors execute in reverse order after `main()` has terminated.

55) How to invoke a constructor function?

A constructor can be invoked by the following two ways.

(i) By calling the constructor explicitly

Example: `myclass mc1 = myclass (6, 5);`

(ii) By calling the constructor implicitly

Example: `myclass mc1(6, 5);`

Where `myclass` is a class which is already declared.

56) What is meant by default argument?

In the function declaration, the user may assign values to the parameters. These values may take effect when there is no value passed while calling a function. These type arguments are called default arguments. For example

`float total (int x, int y; int z = 10);` where `z` is called as default argument.

57) What are the restriction in default argument?

(i) The default values must be specified only once; and this must be first time the function is declared within the file.

(ii) Even though default arguments for the same function cannot be redefined, you can specify different default arguments for each version of an overload function.

(iii) All parameters that take default values must appear to the right of those that do not. For example. it is incorrect to define

`iputs()` like this:

```
void iputs(int indent = -1, char *str)
```

(iv) No default argument should cause harmful or destructive action.

58) What are the advantages of default arguments?

(i) They prevent us from having to provide an overloaded function that takes no parameters.

(ii) The defaulting common initial values is more convenient than specifying each time an object is declared.

(iii) It provides greater flexibility to the programmers.

59) What is the characteristic of default argument?

A default argument is type checked at the time of function declaration and evaluated at the time of the call. The default arguments may be provided for trailing arguments only.

60) What is meant by access and utility function?

An access function is a public member function of a class that return the value of one of the class's data members.

An utility function is a private member function of a class that is used only within the class to perform technical tasks.

61) State the rules of constructor.

- (i) They should be declared in the public part of the class.
- (ii) Don't use dot member operator to invoke the constructor with object name.
- (iii) They do not have return types, not even void.
- (iv) They cannot be inherited, though a derived class can call the base class constructor.
- (v) Constructor cannot be virtual.
- (vi) We cannot refer to their addresses.
- (vii) An object with a constructor cannot be used as a member of a union.
- (viii) They make implicit calls to the new and delete operators when memory allocation is required.

62) Which are the member functions created automatically by the compiler. if they are not included by the programmer in the class definition?

There are four functions created automatically by the compiler if they are not included in the class definition:

- (i) The default constructor.
- (ii) The copy constructor.
- (iii) The overloaded assignment operator.
- (iv) The destructor.

63) What is Scope resolution operator? What is its use?

A scope resolution operator links a class name with a member name in order to tell the compiler what class the member belongs to. It is mostly used in the outside of the class. It can allow access to a name in an enclosing Scope that is "hidden" by a local declaration of the same name. For example, consider this fragment:

```
int ii; // global i
void f1 {
int i // local i
i = 10 ; // local i
:: i = 20 ; // global i
}
```

64) What are the advantages of default arguments?

- (i) They prevent us from having to provide an overloaded function that takes no parameters.
- (ii) The defaulting common initial values is more convenient than specifying each time an object is declared.
- (iii) It provides greater flexibility to the programmers.

65) What is the characteristic of default argument?

A default argument is type checked at the time of function declaration and evaluated at the time of the call. The default arguments may be provided for trailing arguments only.

66) How do you pass objects to functions?

The objects are passed to functions in just the same way that any other type of variable can objects are passed to functions through the use of the standard call-by-value mechanism. This means that a copy of an object is made when it is passed to a function. However, the fact that a copy IS created means, in essence, that another object is created.

67) How constructor and destructors are called when passing objects to functions?

When a copy of an object is generated because It is passed to a function, the object' s constructor function is not called. However, when the copy of the object inside the function is destroyed, its destructor function is called. The copy is destroyed like any other local variable, when the function terminates.

68) How do you assign an object to another object?

The same type objects can be assigned by using assignment operator. This causes the data of the object on the right side to be copied into the data of the object on the left. By default, all data from one object is assigned to the other by use of bit-by-bit copy.

For example

```
class myclass
{
    int i;
    public:
    void set_i (int n) {i = n;}
    int get_i () {return i ;}
}
int main ()
{myclass ob1,ob2;
ob1.set_i(99);
ob1 = ob2 ; //assign data from ob1 to ob2
cout<<"This is ob2's i: "<< ob2.get_i();
return 0;
}
```

69) What is member operator?

An operator (.) that is used to access a member of a class directly in conjunction with object is known as member operator. It is also called as dot operator.

70) What is arrow operator? (or) How can you access the members when pointer to objects?

The arrow operator (->) is used to access the member of objects when a pointer to a structure, union or class is used. For example

```
class c1
{
int i;
public:
c1 (int j) {i = j;}
int get_i () {return i ;}
};
```

```

int main ()
{
c1 ob (88), *p;
p = &ob;
cout<<"The value is = "<< p->get_i();
    //use -> to call get_i().
return 0;
}

```

71) What is an object pointer?

A pointer which points to an object created by a class is called as object pointer, For example

```
myclass *p;
```

where myclass is a class name and p is an object pointer. It is also called as dereferencing operator.

72) What is this pointer?

When a non-static member function is called, it is automatically passed an implicit argument that is a pointer to the invoking object. This pointer is called 'this' pointer. However, this is not an ordinary variable; it is not possible to take the address of 'this' or to assign to 'this'.

73) What are the restrictions on 'this' pointer?

- (i) The friend functions are not members of a class and are not passed a 'this' pointer.
- (ii) The static member functions do not have a 'this' pointer.
- (iii) It is not possible to take the address of 'this' or to assign to 'this'.

75) When (. *) operator and (->*) are used in case of pointer-to member?

When we are accessing a member of an object by using an object or a reference, we must use the. * operator. However, if you are using a pointer to the object, we need to use the->* operator.

76) What is a dangling pointer?

A dangling pointer is a pointer that has not been initialized. It is dangerous because it could be pointing to unallocated memory, or inaccessible memory.

77) What is static binding and dynamic binding?

Static binding is when memory is allocated at the compile time.

For example an array declaration.

```
float x [20] ;
```

Dynamic binding is when memory, is allocated at the run time.

This can be achieved by using 'new' operator.

For example

```
float *x;
```

```
x = new float [20]
```

76. What is meant by reference?

A reference is an alternative name for an object. It is essentially an implicit pointer. The main use of references is for specifying arguments and return values for functions in general and for overloaded operators in particular. It may be used as a stand alone reference. The notation and a means reference to a. For example void. f ()

```
int i = 1; int &r = i;
int x = r; // x= 1
r = 2; // i = 2
}
```

77. How the reference parameters treated in the function?

The reference parameters are treated as same as pointer. It also follow the call by reference mechanism. Inside the function, the reference parameter is used directly without the need to apply the *operator. The changes made in the reference parameter with * operator does not point to some new location, but the value is changed.

78. What is tilde advantage in passing objects as references?

When passing parameters by reference, the change to the object inside the function affect the calling object. This may produce side effects. The main advantage is passing all but the smallest objects by reference is faster than passing them by value.

78. What is independent reference?

We can declare a reference that is simply as variable. This type of reference is called an independent reference. All independent references must be initialized when they are created.

Which are the member functions created automatically by the compiler. if they are not included by the programmer in the class definition?

There are four functions created automatically by the compiler if they are not included in the class definition:

- (i) The default constructor.
- (ii) The copy constructor.
- (iii) The overloaded assignment operator.
- (iv) The destructor.

79. What is Scope resolution operator? What is its use?

A scope resolution operator links a class name with a member name in order to tell the compiler what class the member belongs to. It is mostly used in the outside of the class. It can allow access to a name in an enclosing Scope that is "hidden" by a local declaration of the same name. For example, consider this fragment:

```
int ii; // global i
void f1 {
int i // local i
i = 10 ; // local i
:: i = 20 ; // global i
}
```

80. What is destructor?

A destructor function is a special function that is a member of a class and has the same name as that class preceded by a dilt (~) character. It is automatically called when an object life IS came to end to destroy the previously allocated memory and for some other actions.

For example class stack

```
private:
    int stk[SIZE] ;
    int tos;
public:
    stack () { tos = 0 ;}
~ stack() {cout<<"\nstack destroyed";}
void push(int i) ;
int pop () ;
};
```

81. What is default constructor?

The default constructor is a constructor function, which has no arguments. If this default constructor is not defined by the user explicitly, It is created automatically by the compiler.

82. What is parameterized constructor?

A constructor function that has one or more arguments is called parameterized constructor. Typically, these arguments help initialize an object when t is created. For example

```
class myclass
{
private:
    int a, b;
public:
myclass (int i, int j.) {a =i ; b= j ;} void show () {cout<<a<< " "<<b; }
```

83. How many constructor can a class have?

A class can have any number of constructors, but all the constructors should be differed from each other by means of their parameter list.

84. How many destructors can a class have?

A class can have only one destructor.

85. How do you overload a function call operator ()?

The overloading of operator () creates a new way to call the function. Rather, we can create an operator function that can be passed an arbitrary number of parameters. When overloading (),we can use any type of parameters and return any type of value.We can also specify default arguments.

86. How do you overload a dereferencing operator ->?

The dereferencing operator ->, also called the class member operator, can be used as a unary postfix operator. The general usage is shown here:

Object->element;

Here, object is the object that activates the call. The operator ->() function must return a pointer to an object of the class that operator ->() operates upon. The element must be some member accessible within the object.

Note: An operator ->() function must be a member of the class upon which it works.

87. How do you overload a comma operator?

The comma is a binary operator. We can make an overloaded comma perform any operation we want. If we want the overloaded comma to perform in a fashion similar to its normal operation, then our version must discard the values of all operands except the right most. The right most value becomes the result of the comma operation.

88. What are the advantages of c++ over C?

The advantages of C++ over C are

. Classes.

. Function overloading.

. Operator overloading.

These features enable us to create abstract data types, inherit properties from existing data types and support polymorphism thus making C++ a truly object oriented language.

89. How to invoke a constructor function?

A constructor can be invoked by the following two ways.

(i) By calling the constructor explicitly

Example: myclass mc1 = myclass (6, 5);

(ii) By calling the constructor implicitly

Example: myclass mc1(6, 5);

Where myclass is a class which is already declared.

90. What is meant by default argument?

In the function declaration, the user may assign values to the parameters. These values may take effect when there is no value passed while calling a function. These type arguments are called default arguments. For example

float total (int x, int y; int z = 10); where z is called as default argument.

91. What are the restriction in default argument?

(i) The default values must be specified only once; and this must be first time the function is declared within the file.

(ii) Even though default arguments for the same function cannot be redefined, you can specify different default arguments for each version of an overload function.

(iii) All parameters that take default values must appear to the right of those that do not.

For example. it is incorrect to define

iputs() like this:

```
void iputs(int indent = -1, char *str)
```

(iv) No default argument should cause harmful or destructive action.

92 What are the advantages of default arguments?

(i) They prevent us from having to provide an overloaded function that takes no parameters.

(ii) The defaulting common initial values is more convenient than specifying each time an object is declared.

(iii) It provides greater flexibility to the programmers.

93. What is the characteristic of default argument?

A default argument is type checked at the time of function declaration and evaluated at the time of the call. The default arguments may be provided for trailing arguments only.

94. What is meant by access and utility function?

An access function is a public member function of a class that return the value of one of the class's data members.

An utility function is a private member function of a class that is used only within the class to perform technical tasks.

95. What is meant by access modifiers?

The access modifiers used to control the variables that how variables may be accessed or modified. These precede the type modifiers and the types names they qualify.

96. Give some new operators that are introduced in C++.

:: scope resolution
::* pointer-to-member declaration
->*pointer-to-member operator
.* pointer-to-member operator
delete memory release operator
endl line feed operator
new memory Allocation operator
setw field width operator

97. Which operation refers to Bitwise operation?

The Bitwise operation refers to testing, setting, or shifting the actual bits in a byte or word, which correspond to the char and int data types and variants. You cannot use bitwise operations on float, double, long double, void, bool, or other, more complex types.

98. What is a class?

A class is a user defined data type that links data and code. It provides common properties and relationships for a group of objects. Here the data is referred as member variable and code is referred as member functions that operates on member variables.

99. Write the output of the following code?

```
int i, j;  
i = 1;  
for (j = 0; j < 4; j ++)  
{
```

```
i = i<<1;
printf("shift %d ; %d\n", j, i);
}
```

```
i =I<<2; printf("shift %d",j,i)
```

The output of the above code is

```
shift 0 : 2
shift 1 : 4
shift 2 : 8
shift 3 : 16
shift 4 : 64
```

100. Write the general form of a class?.

```
class class-name
```

```
{
    private data and functions
    access-specifier:
    data and functions
    //...
    access-speicifier:
    data and functions
```

```
} object-list;
```

where object-list is optional.

(16 mark question & answers)

1. Define a class to represent a BANK ACCOUNT . Include the following members.

Name of the depositor , Account number, Type of account, balance amount in the account.

Provide the member function to perform the following tasks.

To assign initial values

To deposit an amount

To withdraw an amount after checking the balance

```
#include<iostream.h>
```

```
#include<conio.h>
```

```
class bank
```

```
{
char nam[30],adr[30]; float bal;
public: int acno;
void get()
{ cout<<"\nEnter the Data :\nAcno: "; cin>>acno;
cout<<"\nName & adr \n"; cin>>nam>>adr;
cout<<"\nEnter the Balance : "; cin>>bal;
if(bal<500){
cout<<"Minimum Balance 500/- needed";
} }
}
```

```
void put();
```

```
void deposit()
```

```

    { float b; cout<<"\nEnter amount to Deposit ";
      cin>>b; bal+=b;
    }
    void withdraw();
};
void bank::put()
{ cout<<"\nName "<<nam<<"\nAddress "<<adr;
  cout<<"\nAccount No. "<<acno<<"\nBalance "<<bal;
}
void bank::withdraw()
{ float b;
  cout<<"\nEnter amount to Withdraw "; cin>>b;
  if((bal-b)<500) cout<<"\nCannot Withdraw\n";
  else bal-=b;
}
void main()
{
bank a[50]; clrscr(); int n,i,op,id;
cout<<"\nEnter no of Customer "; cin>>n;
for(i=0;i<n;i++) a[i].get();
while(1)
{ cout<<"\n1.Deposit\n2.Withdraw\n3.balance\noption ";
  cin>>op;
  switch(op)
  {
case 1:
  cout<<"\nEnter the Account No : "; cin>>id;
  for(i=0;i<n;i++) if(a[i].acno==id) a[i].deposit();
  break;
case 2:
  cout<<"\nEnter the Account No : "; cin>>id;
  for(i=0;i<n;i++) if(a[i].acno==id) a[i].withdraw();
  break;
case 3:
  cout<<"\nEnter the Account No : "; cin>>id;
  for(i=0;i<n;i++)
    if(a[i].acno==id) a[i].put();
  break;
default: return;
}   getch();
}}

```

2. Write a c++ program to perform multiplication between an integer and complex number object using friend operator function.

```

#include<iostream.h>
#include<conio.h>
class complex
{
int real,img;
public:
friend complex operator*(int,complex);
friend istream& operator>>(istream&,complex&);
friend ostream& operator<<(ostream&,complex&);
complex operator*(int n)
{
complex a;
a.real=real*n;
a.img=img*n;
return a;
}};
istream& operator>>(istream&In,complex&c)
{
cout<<"\nEnter the real img part of complex\n";
In>>c.real>>c.img;
return In;
}
ostream& operator<<(ostream&Out,complex&c)
{
Out<<"\nComplex:\t"<<c.real;
if(c.img>0) Out<<" +";
Out<<c.img<<"i\n";
return Out;
}
complex operator*(int n,complex c)
{
complex a;
a.real=c.real*n;
a.img=c.img*n;
return a;
}
void main()
{
clrscr();
complex a,b,c; int n,m;
cin>>a;
cout<<"\nEnter the value to multiply with complex\t";
cin>>n;
cout<<"\nComplex * Integer\n"; c=a*n;
cout<<c; getch();
cin>>b;

```

```

cout<<"\nEnter the value to multiply with complex\t";
cin>>m;
cout<<"\nInteger * Complex\n"; c=m*b;
cout<<c;
getch();
}

```

3. Write functions in c++ using function overloading to compute the area of a square, circle, triangle, rectangle

```

#include<iostream.h>
#include<conio.h>
class shape
{
public:
void area(float val,float v1,float v2)
{
cout<<"\nBreadth :"<<v1<<"\nHeight :"<<v2;
cout<<"\nArea of Triangle "<<v1*v2*val;
}
void area(float v1,float v2)
{
cout<<"\nlength :"<<v1<<"\nHeight :"<<v2;
cout<<"\nArea of Rectangle "<<v1*v2;
}
void area(int v1,float v2)
{
cout<<"\nArea of Circle "<<v1*v2*v1;
}
void area(float v1)
{
cout<<"\nArea of Square "<<v1*v1;
}};
void main()
{
shape a;
float x,y; int z; clrscr();
cout<<"\nEnter the Initial value for Triangle\n";
cin>>x>>y;
a.area(.5,x,y);
cout<<"\nEnter the Initial value for Rectangle\n";
cin>>x>>y;
a.area(x,y);
cout<<"\nEnter the Initial value for Circle\n";
cin>>z;
a.area(z,3.14);
}

```

```

cout<<"\nEnter the Initial value for Square\n";
cin>>x;
a.area(x);
getch();
getch();
}

```

4. Define a class String to represent the value of the string and its length. Allocate memory dynamically for the string. Overload the following operators as
+ for concatenation

[] extract the characters at specified index

() length of the string

Write a c++ program to test this class

```

#include<iostream.h>
#include<conio.h>
#include<string.h>
class string
{
char *x;
int len;
public:
string(){}
string(char a[])
{
len=strlen(a);
x=new char[len+1];
x=a;
}
friend ostream& operator<<(ostream&,string&);
friend string operator+(string,string);
char operator[](int n) { return x[n-1]; }
int operator()() { return len; }
};
string operator+(string a,string b)
{
string c;
c.len=a.len+b.len;
c.x=new char[c.len]; cout<<"\n";
c.x=a.x;
for(int i=0;i<b.len;i++)
*(c.x+i+a.len)=*(b.x+i);
*(c.x+c.len)='\0';
return c;
}
ostream& operator<<(ostream&Out,string&s)
{

```

```

Out<<s.x;
return Out;
}
void main()
{
clrscr();
char a[20],b[20]; int i;
cout<<"\nEnter the String "; cin>>a;
string s(a);
cout<<"\nEnter the String "; cin>>b;
string t(b);
string z; z=s+t; cout<<z;
cout<<"\nEnter the Index "; cin>>i;
cout<<"\nCharacter at Index "<<i<<" in string1 is "<<s[i];
cout<<"\nLength of String "<<z<<" is "<<z();
getch();
}

```

5.Create class called shopping list . The shopping list includes details such as the item code number and price of item. Provide operations such as

Adding an item to the list

Deleting an item from the list

Printing the total value

```
#include<iostream.h>
```

```
#include<conio.h>
```

```
class shop
```

```

{
float price;
public:
int code;
void get(int cod)
{
code=cod;
cout<<"\nEnter the Price "; cin>>price;
}
void put(int n=0)
{
cout<<"Code "<<code<<" Price : "<<price<<"$";
if(n>0) cout<<"Total Price : "<<code*price;
}
};
void main()
{

```

```

int n=0,op,cod,i,c,flg; shop s[50];
while(1)
{
clrscr();
cout<<"1.New\n2.Delete\n3.Total Price\t";
cin>>op;
switch(op)
{
case 1:
l:
cout<<"\nEnter the Code "; cin>>cod;
for(i=0;i<n;i++)
if(s[i].code==cod)
{
cout<<"\nDuplicate Error - Retry"; goto l;
}
s[n++].get(cod);
break;
case 2:
if(n==0) { cout<<"No item"; break; }
cout<<"\nEnter the code ";
cin>>cod;
for(i=0,flg=1;i<n;i++)
if(s[i].code==cod) { s[i]=s[--n]; flg=0; }
if(flg) cout<<"\nData not found";
break;
case 3:
if(n==0) { cout<<"No item"; break; }
cout<<"\nEnter the code ";
cin>>cod;
for(i=0,flg=1;i<n;i++)
if(s[i].code==cod)
{
cout<<"\nEnter the No. Of item "; cin>>c;
s[i].put(c); flg=0;
}
if(flg) cout<<"\nData not found";
break;
default : return;
}
getch();
}
}

```

6. Design a base class called shape . Use this class to store two double type values that could be used to compute the area of figures. Derive two specific classes triangle and

rectangle from the base class shape. Add to the base class a member function getdata() to initialize base class members and another member function display area() to compute and display the area of figures

```
#include<iostream.h>
#include<conio.h>
class shape
{
protected:
float v1,v2;
public:
void getdata()
{
cout<<"\nEnter the Initial value for shape\n";
cin>>v1>>v2;
}
};
class triangle:public shape
{
public:
void area()
{
cout<<"\nBreadth :"<<v1;
cout<<"\nHeight :"<<v2;
cout<<"\nArea of Triangle "<<v1*v2*.5;
}};
class rectangle:public shape
{
public:
void area()
{
cout<<"\nlength :"<<v1;
cout<<"\nHeight :"<<v2;
cout<<"\nArea of Rectangle "<<v1*v2;
}};
void main()
{
triangle t;
rectangle r;
clrscr();
cout<<"\nTriangle : ";
t.getdata();
t.area();
getch();
clrscr();
cout<<"\nRectangle : ";
```

```

r.getdata();
r.area();
getch();
}

```

7. Write a program in c++ to implement set with overloaded operators +, -, <, > for the operations of union, intersection, subset and superset of two sets, respectively.

```

#include<iostream.h>
#include<conio.h>
class set
{
int x[50],n;
public:
friend set operator+(set,set);
friend set operator-(set,set);
friend int operator<(set,set);
friend int operator>(set,set);
friend istream& operator>>(istream&,set&);
friend ostream& operator<<(ostream&,set&);
};
istream& operator>>(istream&In,set&s)
{
cout<<"\nEnter the No. Of element in Set\t";
In>>s.n;
cout<<"\nEnter the Set Values \n";
for(int i=0;i<s.n;i++) In>>s.x[i];
return In;
}
ostream& operator<<(ostream&Out,set&s)
{
if(s.n==0) Out<<"\nNo Element\n";
else
{
Out<<"\nSet of Element "<<s.n<<"\nElements : ";
for(int i=0;i<s.n;i++) Out<<"\t"<<s.x[i];
}
return Out;
}
set operator+(set a,set b)
{
set c; int i,j,k,flg;
for(i=0;i<a.n;i++) c.x[i]=a.x[i];
c.n=a.n;
for(i=0;i<b.n;i++)
{

```

```

    for(j=0,flg=1;j<a.n;j++) if(a.x[j]==b.x[i]) flg=0;
    if(flg==1) c.x[c.n++]=b.x[i];
}
return c;
}
set operator-(set a,set b)
{
    set c; c.n=0; int i,j;
    for(i=0;i<a.n;i++)
    for(j=0;j<b.n;j++)
    if(a.x[i]==b.x[j])
    c.x[c.n++]=a.x[i];
    return c;
}
int operator<(set b,set a)
{
    int i,j,flg;
    for(i=0;i<b.n;i++)
    {
        flg=1;
        for(j=0;j<a.n;j++) if(b.x[i]==a.x[j]) flg=0;
        if(flg) return 0;
    }
    return 1;
}

int operator>(set a,set b)
{
    int i,j,flg;
    for(i=0;i<b.n;i++)
    {
        flg=1;
        for(j=0;j<a.n;j++) if(b.x[i]==a.x[j]) flg=0;
        if(flg) return 0;
    }
    return 1;
}
void main()
{
    clrscr();
    set p,q,r;
    cin>>p>>q;
    cout<<"\n\nUnion";    r=p+q; cout<<r;
    cout<<"\n\nIntersection"; r=p-q; cout<<r;
    if(p>q) cout<<"\nSet2 is Sub Set of Set1\n";
    else cout<<"\nSet2 is not Sub Set of Set1\n";
}

```

```

if(p<q) cout<<"\nSet2 is Super Set of Set1\n";
else cout<<"\nSet2 is not Super Set of Set1\n";
getch();
}

```

8. Write an object oriented program in c++ to list all the palindromes in a given text.

```

#include<iostream.h>
#include<stdio.h>
#include<conio.h>
void main()
{
char b[50],str[10];
int i=0,j=0;
int Palin(char[]);
clrscr();
cout<<"\nEnter the line of Text & Exit by .(dot)\n";
while((b[i++]=getchar())!='. ');
b[i-1]='\0';
cout<<"\nPalindromes are as follows \n\t";
for(i=0;b[i]!='\0';i++)
{
if(b[i]==' '\b[i]=='\n'){
str[j]='\0'; j=0;
if(Palin(str))
cout<<str<<"\t";
}
else str[j++]=b[i];
}
getch();
}
int Palin(char a[])
{
int i=0,l=0;
while(a[i++]!='\0') ;
i-=2;
for(l=0;l<i;l++,i--)
if(a[i]!=a[l]) return 0;
return 1;
}

```

9. Write a c++ program to display the given number in words for example. If the input is 123 and the output is ONE TWO THREE.

```

#include<iostream.h>
#include<conio.h>

```

```

class NumInWord
{
int n;
public:
void get()
{
cout<<"\nEnter the Number : ";
cin>>n;
}
void Word();
};
void NumInWord::Word()
{
int a[20],i=0,m;
char
x[10][15]={"Zero","One","Two","Three","Four","Five","Six","Seven","Eight","Nine"};
cout<<"\nNumber "<<n<<" In Words Is \n";
for(;n>0;a[i++]=n%10,n/=10);
for(i--;i>=0;i--) cout<<"\t"<<x[a[i]];
}
void main()
{
clrscr();
NumInWord num;
num.get();
num.Word();
getch();
}

```

10. Write a c++ program to overload new operator

```

#include<iostream.h>
#include<conio.h>
#include<malloc.h>
class area
{
int b,h;
public:
area(int i,int j) { cout<<"\nConstructor "; b=i; h=j; }
void put()
{
cout<<"\n\n\nBreadth : "<<b<<"\tHeight : "<<h;
cout<<"\nArea of Triangle "<<(.5*b*h);
}
void* operator new(size_t t)
{

```

```

void *d;
d=malloc(t);
cout<<"\nInside New op Overload Fn.";
return d;
}
void operator delete(void* d)
{
cout<<"\n\nDelete Operator : deleting Object ";
free(d);
}
};
void main()
{
clrscr();
int x,y;
cout<<"\nEnter the Breadth & Height\n";
cin>>x>>y;
area *d=new area(x,y);
(*d).put();
delete d;
getch();
}

```

11. Write a c++ program to add two matrices using overloaded + operator

```

#include<iostream.h>
#include<conio.h>
class matrix
{
int x[50][50],r,c;
public:
friend matrix operator+(matrix,matrix);
friend istream& operator>>(istream&,matrix&);
friend ostream& operator<<(ostream&,matrix&);
};
istream& operator>>(istream&In,matrix &c)
{
int i,j;
cout<<"\nEnter the Row & Col of Matrix\t";
In>>c.r>>c.c;
cout<<"Enter the values\n";
for(i=0;i<c.r;i++)
for(j=0;j<c.c;j++)
In>>c.x[i][j];
return In;
}
ostream& operator<<(ostream&Out,matrix&c)

```

```

{
int i,j;
Out<<"\nMatrix:\n";
for(i=0;i<c.r;i++,Out<<endl)
for(j=0;j<c.c;j++)
Out<<"\t"<<c.x[i][j];
return Out;
}
matrix operator+(matrix a,matrix b)
{
matrix c;
if(a.r!=b.r||a.c!=b.c)
cout<<"\nData Error - Invalid Data ";
else
{
c.r=a.r; c.c=a.c;
int i,j;
for(i=0;i<a.r;i++)
for(j=0;j<a.c;j++)
c.x[i][j]=a.x[i][j]+b.x[i][j];
}
return c;
}
void main()
{
clrscr();
matrix a,b,c;
cin>>a>>b;
c=a+b;
cout<<a<<b;
cout<<c;
getch();
}

```

12.Design an object oriented system for student information system

```

#include<iostream.h>
#include<conio.h>
class stud
{
char nam[50],adr[50];
int rno;
public:
void get()
{
cout<<"\nEnter the name :"; cin>>nam;
cout<<"\n Address "; cin>>adr;
}
}

```

```

    cout<<"\nRoll No "; cin>>rno;
}
void put()
{
    cout<<"\nName : "<<nam<<"\nAdrs : "<<adr;
    cout<<"\nRoll : "<<rno;
}
};
class imca:public stud
{
    float m1,m2,m3;
public:
    void getmark()
    {
        get();
        cout<<"\nEnter the Mark "; cin>>m1>>m2>>m3;
    }
    float total() { return m1+m2+m3; }
    void putmark()
    {
        put();
        cout<<"Marks : "<<m1<<"\t"<<m2<<"\t"<<m3;
    }
};
class Date
{
    int dd,mm,yy;
public:
    void getdate()
    {
        cout<<"\nEnter Date : "; cin>>dd>>mm>>yy;
    }
    void putdate()
    {
        cout<<dd<<" : "<<mm<<" : "<<yy<<"\n";
    }
};
class STUDENT:public imca,public Date
{
    float tot;
public:
    void getdata()
    {
        getmark();
        cout<<"\nDate of Birth : "; getdate();
    }
}

```

```

void putdata()
{
    putmark(); putdate();
    tot=total();
    cout<<"\nTotal : "<<tot;
}
};
void main()
{
    STUDENT s;
    clrscr();
    s.getdata();
    s.putdata();
    getch();
}

```

13. Analyse the reason for popularity of Object Oriented programming .Compare OOP with POP.

It supports the Object oriented concepts such as classes ,Objects, inheritance, polymorphism, templates etc...give explanation for all the above.
Compare OOP and POP(study notes).

14. What do you mean by Operator overloading ? What are the rules for overloading operators?

It allows to extend the functionality of an existing operator to operate on user defined data type also.

Arithmetic, Bitwise, Logical, Relational, Shift ,Unary, Subscript, Function operators, new, delete are the overloadable operators.

Operators that cannot be overloaded are conditional operator(?:), Member access operator(.), Sizeof operator, Scope resolution operator(::)

Explain the rules by giving examples.

15. Discuss the need for Virtual Destructor with example.

Destructor can be virtual ,but constructor cannot be virtual. When a derived object pointed to by the base class pointer is deleted, destructor of the derived class as well as destructor of all its base classes are invoked. Explain it with example.

16. Can we Overload the subscript operator? Justify your answer.

Yes, Explain subscript operator overloading with example.

17. What is a constructor ? What are the various type of constructors available in C++.

Explain their use by giving an example for each.

Constructor Definition

Constructor with default arguments, Constructor with parameters, Copy constructors, Overloaded constructors. Explain with examples.

18. Explain in detail the types of inheritance with necessary examples for each.

Single Inheritance

Multiple Inheritance

Hybrid Inheritance

Multilevel Inheritance

Multipath Inheritance
Hierarchical Inheritance

--Explain each with example

19. Discuss modular programming and generic programming.

Define Modular Programming and generic programming.

A set of related procedures with the data they manipulate is often called a module. Decide which modules you want; partition the program so that data is hidden within modules. This paradigm is also known as the data hiding principle.

Decide which algorithms you want; parameterize them so that they work for a variety of suitable types and data structures.

Explain each with examples

Write down the difference between both.

20. Distinguish between early and late binding with examples? Discuss each in detail

Define Late Binding.

It is called dynamic binding, which means the code associated with a given procedure call is not known until the time of the call at runtime.

Write a program to implement Late Binding (eg. Virtual Functions)

Define Early Binding.

Events that occur at compile time is called early binding, also called static binding.

Write a program to implement Late Binding (eg: operator overloading)

Differentiate both.

21. Can we overload the new and delete operators? Justify your answer.

Yes, Explain new and delete operator overloading with example.

22. Explain STL with examples?

The STL is a complex piece of software engineering that uses some C++'s most sophisticated features. It provides general purpose, template classes and functions that implement many popular and commonly used algorithms and data structures like vectors, stacks, queues and lists. It also defines various routines that access them. Explain List, Map, Graph, Vector, Graph etc with examples.

23. Explain the kinds of classes with examples for each.

Explain the following.

*Concrete types

*Abstract types

*Node class

- *Action class
- *Interface class
- *Handles
- *Application framework

24.Explain the relationship between classes.

- *Inheritance relationship
- *Containment relationship
- *Use relationship
- *Programmed in relationship
- *Relationship within classes.

25.Explain Object Oriented design fundamentals

Explain Development process-

Analysis

Design

Implementation,

Testing...

Design steps-

Find classes,

Specify Operations,

Specify dependency

Specify Interface.