QB365 Question Bank Software Study Materials

Inheritance Important 2,3 & 5 Marks Questions With Answers (Book Back and Creative)

11th Standard

Computer Science

Total Marks : 75

2 Marks

10 x 2 = 20

1) What is inheritance?

Answer : Inheritance is one of the most important features of Object Oriented Programming. In object- oriented programming, inheritance enables new class and its objects to take on the properties of the existing classes.

2) What is a base class?

Answer : A class that is used as the basis for inheritance is called a superclass or base class.

3) Why derived class is called power packed class?

Answer : The derived class is a power packed class, as it can add additional attributes and methods and thus enhance its functionality.

4) In what multilevel and multiple inheritance differ though both contains many base class?

Answer :

Multilevel Inheritance	Multiple Inheritance
In multilevel inheritance, the constructors will be executed in the order of inheritance.	If there are multiple base classes ,then it starts executing from the left most base class.

5) What is the difference between public and private visibility mode?

Answer :

Private visibility mode	Public visibility mode
When a base class is inherited with private visibility mode the public and protected members of the base class become 'private' members of the derived class.	When a base class is inherited with public visibility mode, the protected members of the base class will be inherited as protected members of the derived class and the public members of the base class will be inherited as public members of the derived class.

6) When the constructor of base class will automatically invoked?

Answer : Constructors and destructors of the base class are not inherited but during the creation of an object for derived class the constructors of base class will automatically invoked.

7) Write the main advantages of Inheritance.

Answer : The main advantage of inheritance is

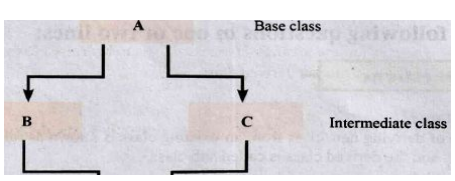
(i) It represents real world relationships well

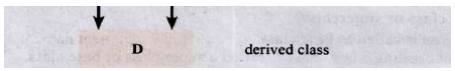
(ii) It provides reusability of code

(iii) It supports transitivity.

8) Draw the pictorial representation of Hybrid Inheritance.

Answer :





9) How many classes in multilevel inheritance?

Answer : **It has three class levels namely**

(i) Base class,

(ii) Intermediate class,

(iii) Derived class.

10) What are the disadvantages of inheritance?

Answer : Inheritance are tightly coupled that is the classes are dependent on each other. It increases the time and efforts take to jump through different levels.

3 Marks 10 x 3 = 30

11) What are the points to be noted while deriving a new class?

Answer : **The following points should be observed for defining the derived class:**

(i) The keyword class has to be used.

(ii) The name of the derived class is to be given after the keyword class.

(iii) A single colon.

(iv) The type of derivation (the visibility mode), namely private, public or protected. If no visibility mode is specified, then by default the visibility mode is considered as private.

(v) The names of all base classes (parent classes) separated by comma.

12) What is difference between the members present in the private visibility mode and the members present in the public visibility mode

Answer :

Members present in the private visibility mode	**Members present in the public visibility mode**
Can be accessed only by the class members.	Can be accessed by the outside members also.
By default the members will be in private visibility mode.	Members, to be in public visibility mode has to be specified explicitly.
When classes are inherited, private members are not inherited.	When classes are inherited, the public members are inherited as private, protected and public members of the derived class.

13) What is the difference between polymorphism and inheritance though are used for reusability of code?

Answer :

Polymorphism	**Inheritance**
Reusability of code is implemented through functions (or) methods.	Reusability of code is implemented through classes.
Polymorphism is the ability of a function to respond differently to different message.	Inheritance is the process of creating derived classes from the base class or classes.
Polymorphism is achieved through overloading.	Inheritance is achieved by various types of inheritances namely single, multiple, multilevel, hybrid and hierarchical inheritances.

14) What do you mean by overriding?

Answer : When a derived class member function has the same name as that of its base class member function, the derived class member function shadows/hides the base class's inherited function. This situation is called function overriding.

15) Write some facts about the execution of constructors and destructors in inheritance.

Answer : Some Facts About the execution of constructor in inheritance

i. Base class constructors are executed first, before the derived class constructors execution

ii. Derived class cannot inherit the base class constructor but it can call the base class constructor by using Base_class name::base_class_constructor() in derived class definition

iii. If there are multiple base classes, then its start executing from the left most base class

iv. In multilevel inheritance, the constructors will be executed in the order of inheritance The destructors are executed in the reverse order of inheritance.

16) In what situation shadowing base class function inderived class arises? How will you resolve the situation?

Answer : In case of inheritance there are situations where the member function of the base class and derived classes have the same name. If the derived class object calls the overloaded member function it leads confusion to the compiler as to which function is to be invoked. The derived class member function have higher priority than the base class member function. This shadows the member function of the base class which has the same name like the member function of the derived class. The scope resolution operator resolves this problem.

17) Can a derived class get access privilege for a private member of the base class? If yes, how?

Answer : A derived class does not get direct access privilege for a private member of its base class, however, it always can access(indirectly) the private members of its base class through an inherited function that is accessing these members.

18) What is inheritance and access control?

Answer : When you declare a derived class, a visibility mode can precede each base class in the base list of the derived class. This does not alter the access attributes of the individual members of a base class, but allows the derived class to access the members of a base class with restriction. Classes can be derived using any of the three visibility mode:

1. In a public base class, public and protected members of the base class remain public and protected members of the derived class
2. In a protected base class, public and protected members of the base class are protected members of the derived class
3. In a private base class, public and protected members of the base class become private members of the derived class.

In all these cases, private members of the base class remain private and cannot be used by the derived class. However it can be indirectly accessed by the derived class using the public or protected member function of the base class since they have the access privilege for the private members of the base class.

19) What are access modifiers?

Answer : Access modifiers determine the scope of the method or variables that can be accessed from other various objects or classes. There are three types of access modifiers, they are

(i) Private

(ii) Protected

(iii) public

20) What is method or function over-riding?

Answer : When a derived class member function has the same name as that of its base class member function, the derived class member function shadow hides the base class inherited function. This situation is called function overriding and this can be resolved by giving the base class name followed by : : and the member function name.

5 Marks 5 x 5 = 25

21) Consider the following c++ code and answer the questions

```
class Personal
{
int Class,Rno;
char Section;
protected:
char Name[20];
public:
personal();
void pentry();
voidPdisplay();
};
```

```
class Marks:private Personal
{
float M{5};
protected:
char Grade[5];
public:
Marks();
void M entry();
void M display();
};
class Result:public Marks
{
float Total,Agg;
public:
char FinalGrade, Commence[20];
Result();
void R calculate();
void R display();
};
```

(i) Which type of Inheritance is shown in the program?

Answer : Multiple Inheritance

22) Consider the following c++ code and answer the questions

```
class Personal
{
int Class,Rno;
char Section;
protected:
char Name[20];
public:
personal();
void pentry();
void Pdisplay(); };
class Marks:private Personal
{ float M{5};
protected:
char Grade[5];
public:
Marks();
void Mentry();
void Mdisplay(); };
class Result:public Marks
{
float Total,Agg;
public:
char FinalGrade, Commence[20];
Result();
void Rcalculate();
void Rdisplay();
};
```

(iii) Give the sequence of Constructor/Destructor Invocation when object of class author is created.

Answer : branch(); // constructor of branch class

publisher(); // constructor of publisher class .

author (): // constructor of author class

-author(); // destructor of author class

-publisher (); // destructor of publisher class

-branch(): // destructor of branch class

23) Consider the following c++ code and answer the questions

```
class Personal
{
int Class,Rno;
char Section;
protected:
char Name[20];
public:
personal();
void pentry();
void Pdisplay(); };
class Marks:private Personal
{ float M{5};
protected:
char Grade[5];
public:
Marks();
void Mentry();
void Mdisplay(); };
class Result:public Marks
{
float Total,Agg;
public:
char FinalGrade, Commence[20];
Result();
void Rcalculate();
void Rdisplay();
};
```

(v) Give number of bytes to be occupied by the object of the following class:

(a) Personal

(b) Marks

(c) Result

Answer :

Publisher	**Branch**	**Author**
Data member Bytes char pname [15] 15 char hoffice [15] 15 char address [25] 25 double turnover 8 char phone[3][10] 30 Total 93	Data member Bytes char beity[15] 15 char address[25] 25 int no_of_emp 4 char bphone[2][10] 20 Total 64	Data member Bytes char pname [15] 15 char hoffice [15] 15 char address [25] 25 double turnover 8 char phone[3][10] 30 char beity[15] 15 char address[25] 25 int no_of_emp 4 char bphone[2][10] 20 int aut code 4
		char aname[20] 20 float income 4 Total 185

24) Write the output of the following program.

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

```
class Container {
public:
  // Constructor definition
  Container(double 1 = 2.C, double b = 2.0, double
  h= 2.0) { -
  cout< < "Constructor called." < < endl;
  length = 1;
  breadth = b;
  height = h;
}
double Volume ( ) {
return length * breadth * height;
}
int compare(Container container)
{
return this- > Volume( ) > Container.volume ( ):
}
private:
  double length; // Length of a Container
  double breadth; // Breadth of a Container
  double height; // Height of a Container
} ;
int main(void) {
  Container Container1(3.3, 1.2, 1.5); II Declare
  Container!
  Container Container2(8.5, 6.0, 2.0); II Declare
  Container2
  if(Container 1.compare(Container2)) {
  cout < < "Container2 is smaller than Container1"
  < } else {
  cout < < "Container2 is equal to or larger than
  Container1' n < }
return 0
}
```

Answer : Output:

Constructor called.

Constructor called

Container 2 is equal to or larger than Container 1

25) Explain this pointer.

Answer : 'this' pointer is a constant pointer that holds the memory address of the current object. .It identifies the currently calling object.It is useful when the argument variable name in the member function and the data member name are same.

```
#include< iostream >
using namespace std;
class T
{
public:
int x;
Void foo( )
{
x = 6;            // same as this- > x = 6;
this- > x=5;  // explicit use of this- >
cout < < endl < < x < <  " "< < this- > x;
}
void foo(int x) //parameter x shadows the member with the same name
{
this- > x = x;    // unqualified x refers to the parameter. 'this- > ' required for disambiguation
cout < < endl < < x < < " "< < this - > x;
}};
int main ( )
{
T tl, t2;
t1.foo();
t2.foo();
}
```

Output

```
5 5
5 5
---------------------------------------
Process exited after 0.1 seconds with return value 0
Press any key to continue ....
```