gradeup

Prep Smart. Score Better.

OOPS Concepts And C++ Programming

ABOUT ME : NAVNEET GUPTA

- 8 years teaching experience.
- AIR 92 in GATE 2008
- Qualified UGC-NET 2012, Raj.-SET 2012, CSIR-Recruitment-Exam in 2011
- Achieved 3rd Rank in NPTEL-DBMS Course
- Achieved Silver Medal in CSIR on ERP Project in 2013
- Area of Expertise : DBMS, Programming, Algorithms, Discrete Maths, Computer Networks, Operating system

gradeup
Prep Smart. Score Better.

What is object-oriented programming?

*) Object-oriented programming combines a group of variables (properties) and functions (method) into a unit called an object. These objects are organised into classes where individual objects can be grouped together.

*) This programming style widely exists in commonly used programming languages like Java, C++ and php

gradeup
Prep Smart. Score Better.

What are the four basics of object-oriented programming?

(i) Encapsulation
(ii) Abstraction
(iii) Inheritance
(iv) Polymorphism

gradeup
Prep Smart. Score Better.

1) **Encapsulation**

a, b
show

data members.

void show()
{
cout<<a
}

*) Encapsulation in c++ is a mechanism the data (variables) and code acting on the data (i.e. methods) together as a single unit.

(*) In encapsulation, the variables of a class will be hidden from other classes and can be accessed only through the methods of their current class. Therefore it is also known as data hiding.

gradeup
Prep Smart. Score Better.

2) Abstraction

*) Abstraction is a process of hiding implementation details from the user, only the functionality will be provided to the user. In other words the user will have the information on what the object does instead of how it does it.

f1 (2+5i, 5+7i)

gradeup
Prep Smart. Score Better.

3) Inheritance

Inheritance can be defined as a process of where one (child/sub) class acquires the properties (methods & fields) of another (Parent/super).

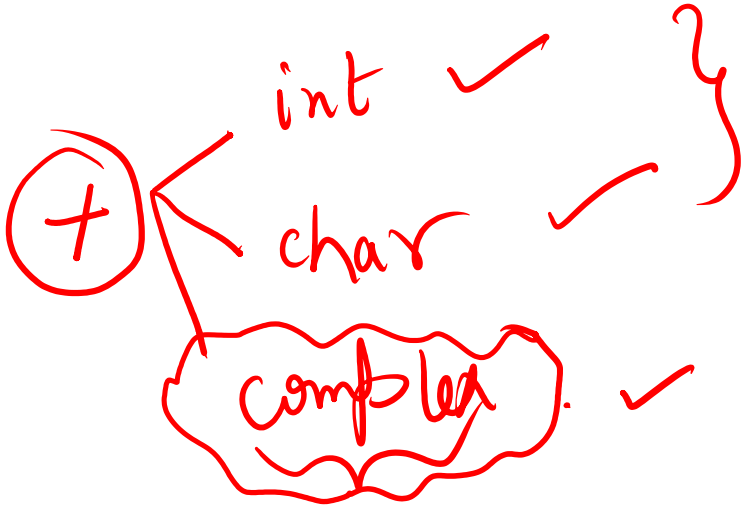
With the use of inheritance, the information is made manageable in a hierarchical order.

(*) reusability of Code.

gradeup
Prep Smart. Score Better.

4) Polymorphism

*) Polymorphism is the ability of an object to perform different actions (or exhibit different behaviors) based on the context.



gradeup
Prep Smart. Score Better.

Advantage of OOPs over Procedure-oriented programming language

1.OOPs makes development and maintenance easier where as in Procedure-oriented programming language it is not easy to manage if code grows as project size grows.

2.OOPs provide data hiding whereas in Procedure-oriented programming language a global data can be accessed from anywhere.

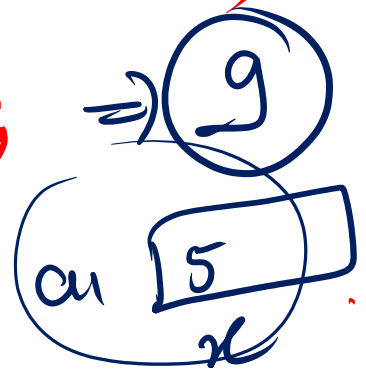
3.OOPs provide ability to simulate real-world event much more effectively. We can provide the solution of real word problem if we are using the Object-Oriented Programming language.

gradeup
Prep Smart. Score Better.

C++ Operators Overloading Example

```
class A
{
  int x;
  public:
    A(int i)
    {
      x=i;
    }
    void operator + (A);
    void display ();
};
```

```
void A :: operator + (A a1)
{
  int y;
  y = a1.x + x;
  cout << y;
}
main ()
{
  A a1(5);
  A a2(4);
  a1 + a2;
}
```



gradeup
Prep Smart. Score Better.

C++ Operators Overloading Example

```
class test
{
  int num;
public:
  test(): num(8) {}
  void operator ++()
  {
    num = num + 5;
  }
  void print()
  {
    cout << num;
  }
};
```

```
main()
{
  test tt;
  ++tt;
  tt.print();
}
```

tt → 8 13 (num)

o/p

13

gradeup
Prep Smart. Score Better.

C++ Function Overriding

*) If derived class defines same function as defined in its base class, it is known as function overriding in c++. It is used to achieve runtime polymorphism.

gradeup
Prep Smart. Score Better.

Function Overriding Example

```
class animal
{
  public:
    void eat ()
    {
      cout << "Eat Me!";
    }
};
```

```
class Dog : public animal
{
  public:
    void eat ()
    {
      cout << "Eat you!";
    }
};
```

```
main ()
{
  Dog d;
  d.eat ();
}
```

o/p :

Eat you!

gradeup
Prep Smart. Score Better.

C++ virtual function

*) It is a member function in the base class that you redefine in a derived class. It is declared using the keyword virtual.

*) There is necessity to use the single pointer to refer to all objects of different classes. So, we create a pointer to the base class that refers to all derived classes. But when base class pointer contains the address of derived class object, always execute base class function.

gradeup
Prep Smart. Score Better.

C++ virtual function (Rules)

*) Virtual function must be member of some class.

*) Virtual function cannot be static members.

*) These are accessed through object pointer.

*) They can be a friend of another class. ✓

*) A virtual function has to be defined in the base class even though it is not used.

*) function signature (prototype) in all the classes should be same

*) Virtual construct (No), virtual destructor (can be) ✓

gradeup
Prep Smart. Score Better.

Consider the situation when we don't use the virtual keyword:

```
class A
{ int x = 5;
  public :
    void display ()
    {
      cout << x;
    }
};
```

```
class B : public A
{
  int y = 10;
  public:
    void display ()
    {
      cout << y;
    }
};
```

```
main()
{
  A *a1;
  B b1;
  a1 = &b1;
  a1 -> display ();
}
```

gradeup
Prep Smart. Score Better.

C++ virtual function Example

```
class A
{public:
    virtual void display()
        {cout<<"I am in A";
        }
};
```

o/p

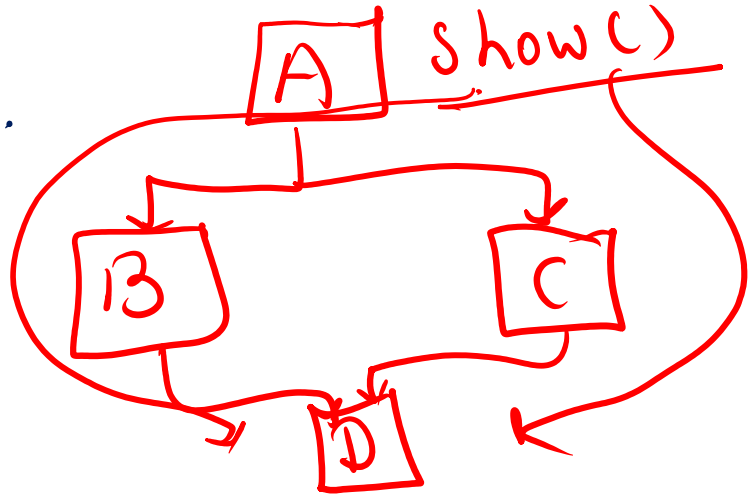
I am in B

```
class B:public A
{
public:
    void display()
    {
        cout<<"I m in B";
    }
};
main()
{
    A *a1;
    B b1;
    a1=&b1;
    a1->display();}
```

gradeup
Prep Smart. Score Better.

Virtual base class in C++

Virtual base classes are used in virtual inheritance in a way of preventing multiple instances of a given class appearing in an inheritance hierarchy when using multiple inheritance.



gradeup
Prep Smart. Score Better.

o/p error

Example: To show the need of Virtual Base Class in C++

```
class A
{
public : void show()
    {
      cout << "Hello from A";
    }
};
class B : public A
{ };
class C : public A
{ };
class D : public B,
          public C
{ };
main()
{ D obj;
  obj.show();
}
```

gradeup
Prep Smart. Score Better.

How to resolve this issue?

```
class A
{ public : void show()
           { cout << "Hello";
           }
};

class B : virtual public A
{ };

class C : public virtual A
{ };
class D : public virtual B,
          public virtual C
{ };
main()
{ D obj;
  obj.show();
}
```

o/p Hello ✓

gradeup
Prep Smart. Score Better.

What is virtual inheritance?

a) C++ technique to avoid multiple copies of the base class into children/derived class

b) C++ technique to avoid multiple inheritances of classes

c) C++ technique to enhance multiple inheritance

d) C++ technique to ensure that a private member of the base class can be accessed somehow

gradeup
Prep Smart. Score Better.

a) “Constructor called” five times and then “Destructor called” five times
b) “Constructor called” five times and then “Destructor called” once
c) Error
d) Segmentation fault

```
#include <iostream>
using namespace std;
class A{
public:
        A(){
                cout<<"Constructor called\n";
           }
        ~A(){
                cout<<"Destructor called\n";
            }
};
int main(int argc, char const *argv[])
{
        A *a = new A[5];
        delete a;
        return 0;
}
```

gradeup
Prep Smart. Score Better.

a) “Constructor called” five times and then “Destructor called” five times
b) “Constructor called” five times and then “Destructor called” once
c) Error
d) Segmentation fault

```
#include <iostream>
using namespace std;
class A{
public:
        A(){
                cout<<"Constructor called\n";
            }
        ~A(){
                cout<<"Destructor called\n";
            }
};
int main(int argc, char const *argv[])
{
        A *a = new A[5];
        delete[] a;
        return 0;
}
```

gradeup
Prep Smart. Score Better.

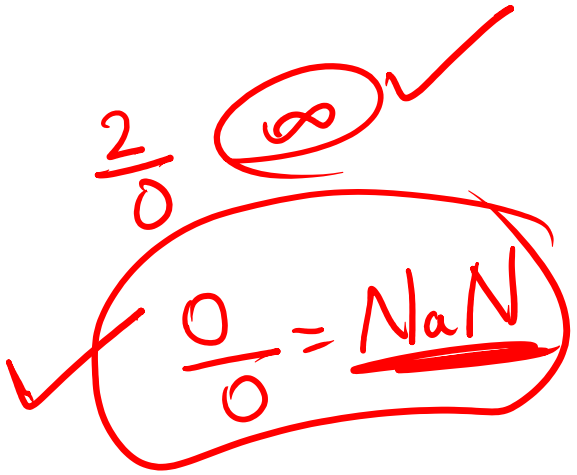
Which of the following is not considered as an error in JavaScript?

a) Syntax error

b) Missing of semicolons

c) Division by zero

d) Missing of Bracket



gradeup
Prep Smart. Score Better.

The statement a===b refers to _____

a) Both a and b are equal in value, type and reference address

b) Both a and b are equal in value

c) Both a and b are equal in value and type

d) There is no such statement

a === b

↳ strict comparison operator in javascript which check value of a, b and also datatypes of (a & b)

gradeup
Prep Smart. Score Better.

What will be the output of the following JavaScript code?

```
function equalto()
{
    int num=10;
    if(num===”10”)
        return true;
    else
        return false;
}
```

a) true
b) false
c) runtime error
d) compilation error

gradeup
Prep Smart. Score Better.

What will be the output of the following JavaScript code?

```
int a=1;
if(a!=null)
    return 1;
else
    return 0;
```

a) 1
b) 0
c) runtime error
d) compiler error