Handling Parametrized Constructors In Inheritance

If the constructor in Base class is parametrized then we must call that constructor from the constructor of derived class explicitly . Otherwise the compiler will generate syntax error

```
#include <iostream>
using namespace std;
class A
{
public:
   A(int i)
   {
      cout<<"In constructor of base class A"<<endl;
   }
   ~A()
   {
      cout<<"In destructor of base class A"<<endl;
   }
};
```

This is compulsory!

```
class B:public A
{
public:
   B():A(25) // Explicit inline call of Constructor
   {
      cout<<"In constructor of derived class B"<<endl;
   }
   ~B()
   {
      cout<<"In destructor of derived class B"<<endl;
   }
};
int main()
{
   B obj;
   return 0;
}
```

```
#include <iostream>
using namespace std;
class Num
{
   protected:
      int a,b;
   public:
      Num(int i,int j)
      {
         a=i;
         b=j;
      }
      void show()
      {
         cout<<"a="<<a<<",b="<<b<<endl;
      }
};
```

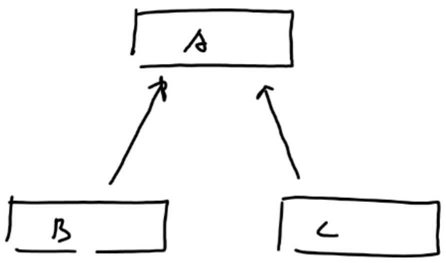
```
#include <iostream>
using namespace std;
class Num
{
    protected:
        int a,b;
    public:
        Num(int i,int j)
        {
            a=i;
            b=j;
        }
        void show()
        {
            cout<<"a="<<a<<",b="<<b<<endl;
        }
};
```

```
class AddNum: public Num
{
    int c;
public:
    AddNum(int x,int y):Num(x,y)
    {

    }
    void add()
    {
        c=a+b;
    }
    void show()
    {
        Num::show();
        cout<<"Sum is "<<c<<endl;
    }

};
int main()
{
    AddNum obj1(10,20);
    obj1.add();
    obj1.show();
    AddNum obj2(30,40);
    obj2.add();
    obj2.show();
    return 0;
}
```

Hierarchial Inheritance



```
class Figure
{

  protected:
             int dim1,dim2;
public:
          void get()
          {
              cout<<"enter dimensions:";
              cin>>dim1>>dim2;
          }
          void show()
          {
            cout<<dim1<<","<<dim2<<endl;
          }

};
```

```
class Rectangle: public Figure
{
   public:
              void area()
            {
               cout<<"Rect area="<<dim1*dim2;
             }
};
class Triangle:public Figure
{
public:
              void area()
            {
               cout<<"Tri area="<<0.5*dim1*dim2;
             }

};
```

```
int main()
{

   Rectangle R;
   R.get();
   R.show();
   R.area();

   Triangle T;
   T.get();
   T.show();
   T.area();
   return 0;
}
```

Hybrid Inh

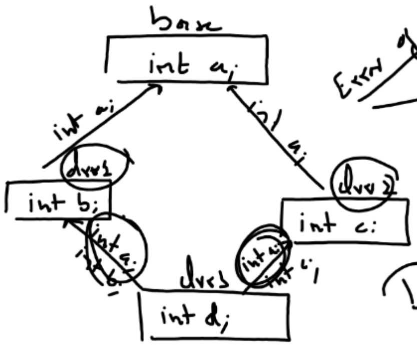
```
class base
{
   public:
              int a;
};

class drv1:public base
{
   public:
              int b;
};
class drv2:public base
{
   public:
              int c;
};
```

```
class drv3:public drv1,public drv2
{
   public:
              int d;
};
```

```
int main()
{
   drv3 obj;
   obj.a=10;
   obj.b=20;
   obj.c=30;
   obj.d=obj.a+obj.b+obj.c;
   cout<<"Sum is "<<obj.d;
   return 0;
}
```

Error of ambiguity



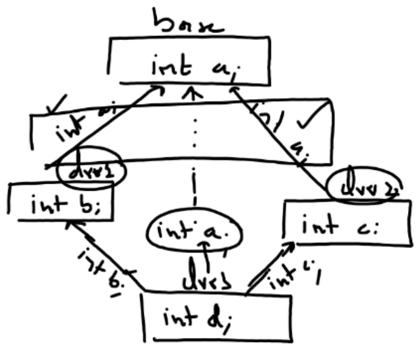
Diamond Problem

Hybrid Inh / Virtual Inh

```
class base
{
  public:
            int a;
};

class drv1:virtual public
base
{
  public:
            int b;
};
class drv2: virtual public
base
{
  public:
            int c;
};
```

```
class drv3:public drv1,public drv2
{
  public:
            int d;
};
```



```
int main()
{
  drv3 obj;
  obj.a=10;
  obj.b=20;
  obj.c=30;
  obj.d=obj.a+obj.b+obj.c;
  cout<<"Sum is "<<obj.d;
  return 0;
}
```

$\uparrow$ 60