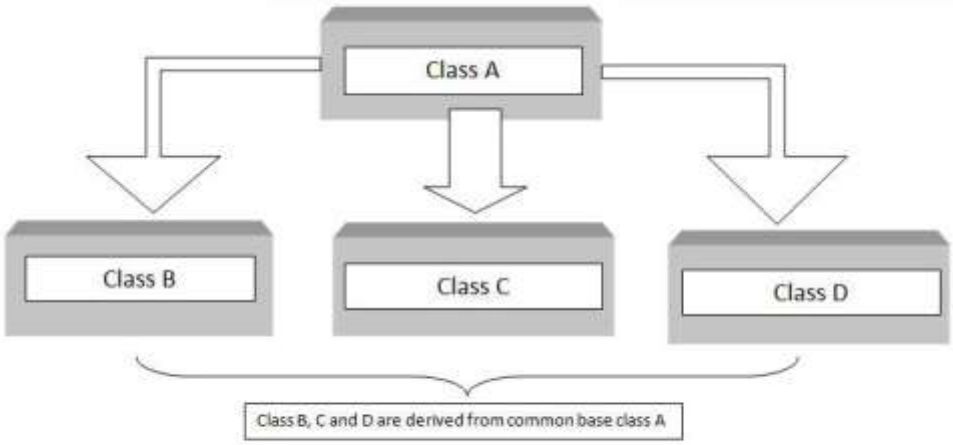
C++ Hierarchical Inheritance Block Diagram




```
      //C++ Hierarchical Inheritance Example
// hierarchial inheritance.cpp
#include <iostream>
using namespace std;

class A
{
   public:
      int x, y;
      void getdata()
      {
         cout << "\nEnter value of x and y:\n"; cin >> x >> y;
      }
};
class B : public A
```

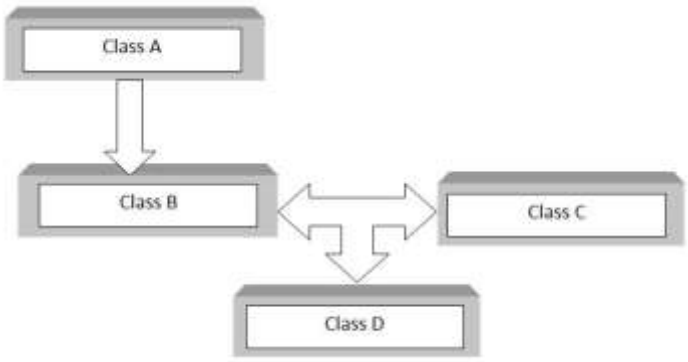
```
{
   public:
      void product()
      {
         cout << "\nProduct= " << x * y;
      }
};
class C : public A
{
   public:
      void sum()
      {
      cout << "\nSum= " << x + y;
      }
};
int main()
{
   B obj1;
   C obj2;
   obj1.getdata();
   obj1.product();
   obj2.getdata();
   obj2.sum();
   return 0;
}
```

```
#include <iostream>
using namespace std;
class Shape               // Declaration of base class.
{
   public:
   int a;
   int b;
   void get_data(int n,int m)
   {
      a= n;
      b = m;
   }
};
class Rectangle : public Shape  // inheriting Shape class
{
   public:
   int rect_area()
   {
      int result = a*b;
      return result;
   }
};
class Triangle : public Shape    // inheriting Shape class
{
   public:
```

```
        int triangle_area()
        {
            float result = 0.5*a*b;
            return result;
        }
    };
    int main()
    {
        Rectangle r;
        Triangle t;
        int length,breadth,base,height;
        std::cout << "Enter the length and breadth of a rectangle: " <<
std::endl;
        cin>>length>>breadth;
        r.get_data(length,breadth);
        int m = r.rect_area();
        std::cout << "Area of the rectangle is : " <<m<< std::endl;
        std::cout << "Enter the base and height of the triangle: " << std::endl;
        cin>>base>>height;
        t.get_data(base,height);
        float n = t.triangle_area();
        std::cout <<"Area of the triangle is : "  << n<<std::endl;
        return 0;
    }
```

C++ Hybrid Inheritance

C++ Hybrid Inheritance Block Diagram




```
//hybrid inheritance.cpp
#include <iostream>
using namespace std;
class A
{
public:
int x;
};
class B : public A
{
public:
B()
{
x = 10;
}
};
class C
```

```
{
public:
int y;
C()
{
y = 4;
}
};
class D : public B, public C
{
public:
void sum()
{
cout << "Sum= " << x + y;
}
};

int main()
{
D obj1;
obj1.sum();
return 0;
}
```

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

```
{
   protected:
   int a;
   public:
   void get_a()
   {
      std::cout << "Enter the value of 'a' : " << std::endl;
      cin>>a;
   }
};

class B : public A
{
   protected:
   int b;
   public:
   void get_b()
   {
       std::cout << "Enter the value of 'b' : " << std::endl;
      cin>>b;
   }
};
class C
{
   protected:
   int c;
```

```
   public:
   void get_c()
   {
      std::cout << "Enter the value of c is : " << std::endl;
      cin>>c;
   }
};

class D : public B, public C
{
   protected:
   int d;
   public:
   void mul()
   {
      get_a();
      get_b();
      get_c();
      std::cout << "Multiplication of a,b,c is : " <<a*b*c<< std::endl;
   }
};
int main()
{
   D d;
   d.mul();
   return 0;
```

```
}
```