Subject Name: **Object Oriented Programming Using C++**

Subject Code: **BCA-301 N**

Subject Topic: **Multiple Inheritance and Ambiguity in Multiple Inheritance**

Abhishek Dwivedi

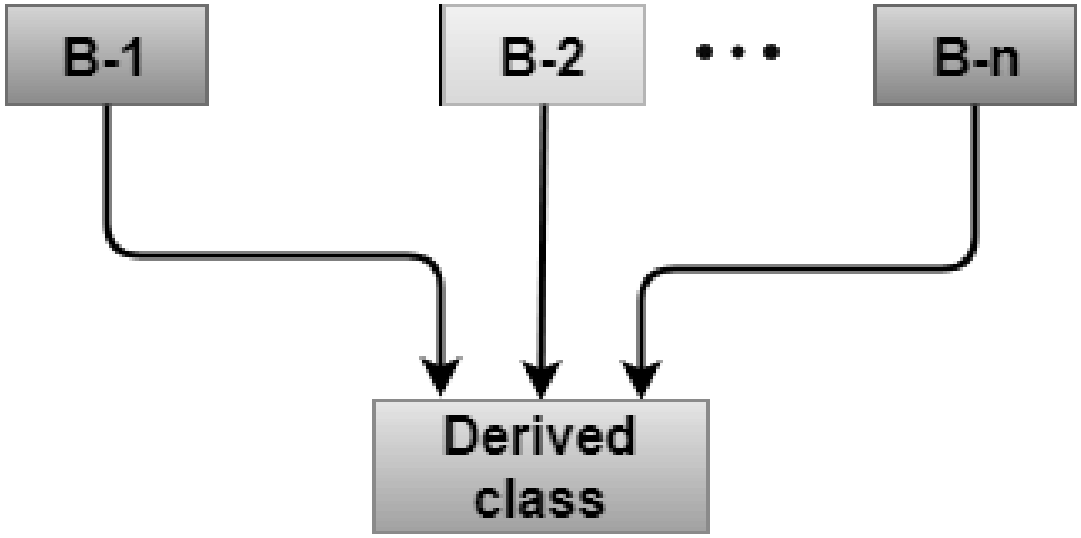
Assistant Professor

Department of Computer Application

UIET, CSJM University, Kanpur

Multiple inheritance

- In this type of inheritance a single derived class may inherit from two or more than two base classes.



Syntax of the Derived class

- class subclass_name : access_mode base_class1, access_mode base_class2, ...

```
{
   //body of subclass
};
class A
{
   protected:
    int a;
   public:
   void get_a(int n)
   {
       a = n;
   }
};
```

```
class B
{
    protected:
    int b;
    public:
    void get_b(int n)
    {
        b = n;
    }
};
```

```
class C : public A, public B
{
   public:
    void display()
    {
        cout << "The value of a is : " <<a<< endl;
        cout << "The value of b is : " <<b<< endl;
        cout<<"Addition of a and b is : "<<a+b;
    }
};
```

```
void main()
{
   C c;
   c.get_a(10);
   c.get_b(20);
   c.display();

    getch();
}
```

Ambiguity in Multiple Inheritance

- Ambiguity can be occurred in using the multiple inheritance when a function with the same name occurs in more than one base class. An example:

```
class A
{
   public:
   void display()
   {
      cout << "Class A" << endl;
   }
};
class B
{
   public:
   void display()
   {
      cout << "Class B" << endl;
   }
};
```

Example

```
class C : public A, public B
{
    Public:
    void view()
    {
        display();
    }
};
void main()
{
    C c;
    c.view();
    getch();
}
```

Result and Resolution

- error: reference to 'display' is ambiguous display();
- There are 2 ways to avoid this ambiguity:

1. Use scope resolution operator
2. Use virtual base class

Use scope resolution operator

- The above issue can be resolved by using the class resolution operator with the function. In the above example, the derived class code can be rewritten as:

```
class C : public A, public B
{
    public:
    void view()
    {
        A :: display();      // Calling the display() function of class A.
        B :: display();      // Calling the display() function of class B.

    }
};
```

Ambiguity in Single Inheritance in C++

```
class staff
{
    private:
        char name[50];
        int code;
    public:
        void getdata();
        void display();
};
class typist: public staff
{
    private:
        int speed;
    public:
        void getdata();
        void display();
};
```

```
void staff::getdata()
{
     cout<<"Name:";
     gets(name);
    cout<<"Code:";
    cin>>code;
}
void staff::display()
{
    cout<<"Name:"<<name<<endl;
    cout<<"Code:"<<code<<endl;
}
```

```
void typist::getdata()
{
   cout<<"Speed:";
   cin>>speed;
}
 void typist::display()
{
   cout<<"Speed:"<<speed<<endl;
}
```

```
void main()
{
   typist  t;
   cout<<"Enter data"<<endl;
   t.staff::getdata();
   t.getdata();
   cout<<"Display data"<<endl;
   t.staff::display();
   t.display();
   getch();
}
```

References:

- www.studytonight.com
- www.tutorialpoint.com
- www.geeksforgeeks.org
- “Object oriented programming in C++” Robert Lafore
- “Object oriented programming with C++”, E.Balagurusamy