

LAB No 17

Multilevel, hierarchical and hybrid Inheritance

If a class is derived from another derived class then it is called **multilevel inheritance**. So in C++ multilevel inheritance, a class has more than one parent class. For example, if we take animals as a base class then **mammals** are the derived class which has features of animals and then humans are the also derived class that is derived from sub-class mammals which inherit all the features of mammals.

LAB Objectives

- Construct programs using multiple, hybrid and hierarchical inheritance

Apparatus

- Computer System

Required Software

You can choose any one of the software from the list given below

- Visual Studio
- Dev C++
- Codeblocks

Lab Outcomes

By the end of this experiment, student will have basic understanding of multiple, hybrid and hierarchical inheritance and its working.

Syntax:

```
class A // base class

{
    .....
};

class B : access_specifier A // derived class

{
    .....
};
```

```
class C : access_specifier B // derived from derived class B
```

```
{ .....
```

```
};
```

Multilevel Inheritance

Multilevel inheritance is a process of deriving a class from another derived class. When one class inherits another class which is further inherited by another class, it is known as multi level inheritance in C++. Inheritance is transitive so the last derived class acquires all the members of all its base classes.

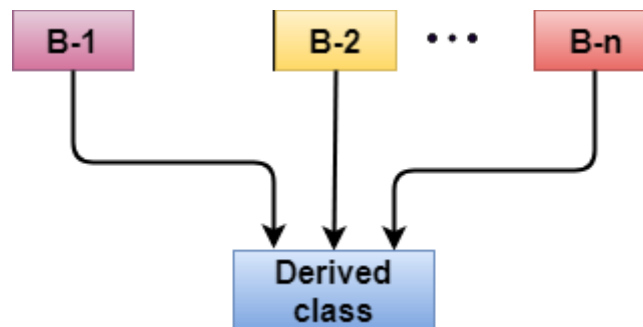


Figure 17.1 Block diagram of multilevel inheritance

Example 17.1

Compile the program and observe the output and working of multilevel inheritance concept

```
#include <iostream>
using namespace std;
class base //single base class
{
    public:
    int x;
    void getdata()
    {
        cout << "Enter value of x= "; cin >> x;
    }
};
class derive1 : public base // derived class from base class
```

```

{
    public:
    int y;
    void readdata()
    {
        cout << "\nEnter value of y= "; cin >> y;
    }
};
class derive2 : public derive1 // derived from class derive1
{
    private:
    int z;
    public:
    void indata()
    {
        cout << "\nEnter value of z= "; cin >> z;
    }
    void product()
    {
        cout << "\nProduct= " << x * y * z;
    }
};
int main()
{
    derive2 a; //object of derived class
    a.getdata();
    a.readdata();
    a.indata();
    a.product();
    return 0;
} //end of program

```

Output

```

Enter value of x= 2

Enter value of y= 3

Enter value of z= 3

Product= 18

```

Example 17.2

Compile the program in C++, and observe the output and working of multilevel inheritance concept

```

#include<iostream>
using namespace std;

```

```

class M
{
    protected:
    int m;
    public :
    void get_M(int );
};
class N
{
    protected:
    int n;
    public:
    void get_N(int);
};
class P: public M, public N
{
    public:
    void display(void);
};

void M::get_M(int x)
{
    m=x;
}
void N::get_N(int y)
{
    n=y;
}
void P::display(void)
{
    cout<<"\n\tm = "<<m<<endl;
    cout<<"\n\tn = "<<n<<endl;
    cout<<"\n\tm*n = "<<m*n<<endl;
}
int main()
{
    P p;
    p.get_M(10);
    p.get_N(20);
    p.display();
    return 0;
}

```

Example 17.3

Compile the program for finding factorial of a number and observe the output and working of multilevel inheritance concept

```

#include<iostream>
using namespace std;
//main class
class factorial
{
    public:
    unsigned long long fact=1;
    void display();
};
//child of main class
class input_factorial: public factorial
{
    public:
    int num;
    int input();
};
//child of another child class
class fact_function_factorial: public input_factorial
{
    public:
    void fact_function();
};
int input_factorial::input()
{
    cout<<"Please enter a number: ";
    cin>>num;
}

void fact_function_factorial::fact_function()
{
    for(int i=1;i<=num;i++)
    {
        fact=fact*i;
    }
}

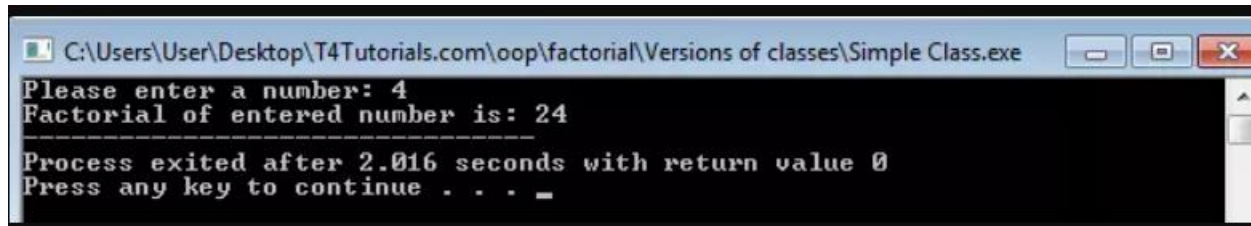
void factorial::display()
{
    cout<<"Factorial of entered number is: "<<fact;
}

int main()
{
    fact_function_factorial object;

    object.input();
    object.fact_function();
    object.display();
}

```

Output



```
C:\Users\User\Desktop\T4Tutorials.com\oop\factorial\Versions of classes\Simple Class.exe
Please enter a number: 4
Factorial of entered number is: 24
Process exited after 2.016 seconds with return value 0
Press any key to continue . . . _
```

Example 17.4

Compile the program and observe the output and working of multilevel inheritance concept

```
#include <iostream>
using namespace std;
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;
        }
};
int main()
```

```
{  
    C c;  
    c.get_a(10);  
    c.get_b(20);  
    c.display();  
  
    return 0;  
}
```

Output:

```
The value of a is : 10  
The value of b is : 20  
Addition of a and b is : 30
```

In the above example, class 'C' inherits two base classes 'A' and 'B' in a public mode.

C++ Hybrid Inheritance

Hybrid inheritance is a combination of more than one type of inheritance.

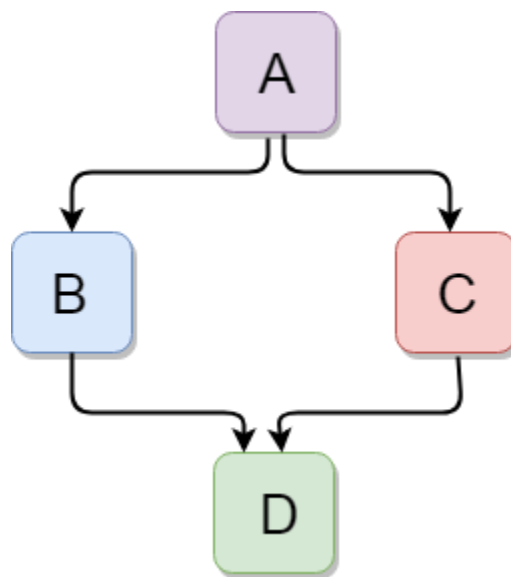


Figure 17.2 Block diagram of hybrid inheritance

Example 17.5

Compile the program and observe the output and working of multilevel inheritance concept

1. `#include <iostream>`
2. `using namespace std;`

```
3. class A
4. {
5.     protected:
6.         int a;
7.     public:
8.         void get_a()
9.         {
10.            std::cout << "Enter the value of 'a' : " << std::endl;
11.            cin>>a;
12.        }
13.};
14.
15.class B : public A
16.{
17.    protected:
18.        int b;
19.    public:
20.        void get_b()
21.        {
22.            std::cout << "Enter the value of 'b' : " << std::endl;
23.            cin>>b;
24.        }
25.};
26.class C
27.{
28.    protected:
29.        int c;
30.    public:
31.        void get_c()
32.        {
33.            std::cout << "Enter the value of c is : " << std::endl;
34.            cin>>c;
35.        }
36.};
37.
38.class D : public B, public C
39.{
```



```
40. protected:
41. int d;
42. public:
43. void mul()
44. {
45.     get_a();
46.     get_b();
47.     get_c();
48.     std::cout << "Multiplication of a,b,c is : " << a*b*c << std::endl;
49. }
50. };
51. int main()
52. {
53.     D d;
54.     d.mul();
55.     return 0;
56. }
```

Output:

```
Enter the value of 'a' :
10
Enter the value of 'b' :
20
Enter the value of c is :
30
Multiplication of a,b,c is : 6000
```

C++ Hierarchical Inheritance

Hierarchical inheritance is defined as the process of deriving more than one class from a base class.

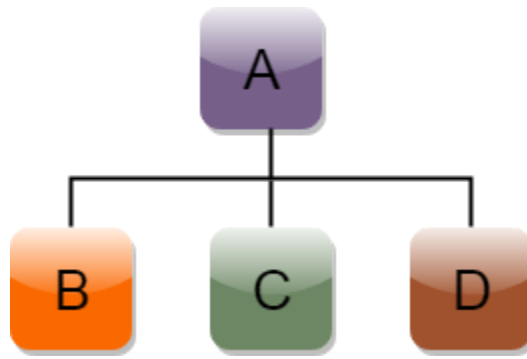


Figure 17.3 Block diagram of hierarchical inheritance

Syntax

```
class A
{
    // body of the class A.
}
class B : public A
{
    // body of class B.
}
class C : public A
{
    // body of class C.
}
class D : public A
{
    // body of class D.
}
```

Example 17.6

Compile the program and observe the output and working of multilevel inheritance concept

```
#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;
}

```

Output:

```

Enter the length and breadth of a rectangle:
23
20
Area of the rectangle is : 460
Enter the base and height of the triangle:
2
5
Area of the triangle is : 5

```

Example 17.7

Compile the program and observe the output and working of multilevel inheritance concept

```

// header files
#include<iostream>
using namespace std;
//base class level 0
class student
{
protected:
int roll_no;
public:
void get_roll_no(int no)
{
roll_no=no;
}
void display()
{
cout<<"\nRoll number of the student is :- "<<roll_no;
}
};

```

```

//derived class at level 1 which is base class of derived class at level 2
class test: public student
{
protected:
int math, science;
public:
void get_marks (int no1,int no2)
{
math = no1;
science = no2;
}
void display()
{
cout<<"\nMarks in math :- "<<math;
cout<<"\nMarks in science :- "<<science<<"\n";
}
};
//derived class at level 2
class result: public test
{
int average;
public:
void display()
{
average = (math + science)/2;
}
//class to display member function of base class at level 0
student::display();
//class to display member function of intermediate base class at level 1
test::display();
cout<<"\nAverage marks = "<<average;
}
};
int main()
{
// object of derived class at level 2
result obj1;
//call to member functions
obj1.get_roll_no(12);
obj1.get_marks(75, 85);
obj1.display();
return 0;
}

```

The definition of the member function also includes the function call statements to the member functions of base classes **student** and **test**. When you call the member function display of derived class from the main (), indirect call is made to the display () of both the **student** and **test** classes which are nested into the display () of the derived class result.

Output

```

Roll number of the student is :- 12
Marks in math :- 75

```

Marks in science :- 85 Average marks = 80
--

Example 17.8

To demonstrate the calling of constructor's in case of multilevel inheritance

```
// header files
#include<iostream>
using namespace std;

// definition of base class at level 0
class base0
{ public:
base0 ( )
{
cout<<"\nConstructor function of base class at level 0";
}
};

// definition of intermediate base class at level 1
class base1
{
private:
public:
base1 ( )
{
cout<<"\nConstructor function of base class at level 1";
}
};

// definition of derived class at level 2
class derived2 : public base0 , public base 1
{ private:
public:
derived2 ( )
{
cout<<"\nConstructor function of derived class at level 2";
}
};

int main ()
{
// object of derived class at level 2
derived2 obj1;
return 0;
}
```

Output

Constructor function of base class at level 0

Constructor function of base class at level 1 Constructor function of derived class at level 2

Nested Class

Nesting of classes is an interesting concept. Sometimes it is taken as an alternative to inheritance, but in actual it is not an alternative to inheritance. You cannot get the benefits of the protected access specifier in case of nested classes.

What is a nested class?

When you declare an object of 1 class as data member of other class, first class becomes nested class of second class. You can then use the object of nested class to access public members of nested class. Second class is neither friend class of the first class nor it inherits the features of the first class. It is a simple concept where in you create the object of 1 class as member of other class.

Example 17.9

Compile the program and observe the output by implementing the concept of nesting class

```
// header files
#include<iostream>
using namespace std;

// definition of class to be nested
class student
{
private:
int roll_no, class_no;
char name[20];
public:
void input ( )
{
cout<<"Enter the students name :- ";
cin>>name;
cout<<"Enter the students class :- ";
cin>>class_no;
cout<<"Enter the students roll number:- ";
cin>>roll_no;
}
void display ( )
{
cout<<"\nStudents name :- "<<name;
cout<<"\nStudents class:- "<<class_no;
cout<<"\nStudents Roll number:- "<<roll_no;
}
```

```

};

// definition of class that will implement nesting of class
class marks
{
private:
int math, science, english, avg;

//object of class student
student s1;
public:
void input ( )
{

//call to member function of class student
s1.input();
cout<<"Enter marks for math :- ";
cin>>math;
cout<<"Enter marks for science :- ";
cin>>science;
cout<<"Enter marks for english :- ";
cin>>english;
}
void compute_avg()
{
avg = (math + science + english)/3;
}
void display ( )
{

//call to member function of class student
s1.display();
cout<<"\nAverage marks are :- "<<avg;
}
};

int main ()
{
// object of class marks

marks obj1;

// call to member functions
obj1.input( );
obj1.compute_avg();
obj1.display();
return 0;
}

```

Summary of inheritance

Single Inheritance: a single child class derived from a single base class

■ **Hierarchical Inheritance:** multiple child classes derived from a single base class

■ **Multi-level Inheritance:** a single derived class inherited from another derived class

■ **Multiple Inheritance:** a single derived class inherited from more than one base classes

■ **Hybrid Inheritance:** any legal combination of more than one type of inheritance

LAB Review Questions

Question No. 1(CLO3, P2)

Compile all the example programs from this lab and verify the output by your lab instructor

Question No. 2 (CLO4, P4)

Construct a program in C++, to display student marks sheet using multiple inheritance concept.

Make 2 classes (student, marks)

The member functions of class students included

- Getdata→Enter the student(name, roll number, semester, city)
- Put data→student(name, roll number, semester, city)

The member functions of class marks included

- Getdata→enter the marks in (sub1, sub2, sub3)
- Putdata→show the marks(sub1, sub2, sub3)
- Calculate→(total percentage of marks)

Using do while loop and switch statement in main function print the following menu and the output should be like this

Output

welcome in the student information system

1.input data

2.output data

3.Calculate percentage

4.exit

Enter the choice :1

Enter the student name: kashif

Enter the student roll no: 1

Enter the Semester no: 1

Enter the city name: lahore

enter the marks1: 80

enter the marks2: 90

enter the marks3: 70

1.input data

2.output data

3.Calculate percentage

4.exit

Enter the choice :: 2

the student name: kashif

the student roll no: 1

Semester No. 1

City: lahore

marks1: 80

marks2: 90

marks3: 70

1.input data

2.output data

3.Calculate percentage

4.exit

Enter the choice :: 3

total percentage :: 80

1.input data

2.output data

3.Calculate percentage

4.exit

Enter the choice :: 4

Question No. 3 (CLO4, P4)

Construct a program in C++ to enter the student details using hierarchical inheritance and observe.
The desired output of the program is mention below

Program Requirements:

Make 3 classes (student, arts, science)

Class student → base class

Class arts → derive from base class student

Class science → derive from base class student

Output

Entering details of the arts student

Enter the first name: John

Enter the second name: Max

Enter the age: 19

Enter the roll_no: 123

Enter the subject1 of the arts student: A

Enter the subject2 of the arts student: B

Enter the subject3 of the arts student: C

Displaying the details of the arts student

First Name = John
Last Name = Max
Age = 19
Roll Number = 123
Subject1 of the arts student = A
Subject2 of the arts student = B
Subject3 of the arts student = C

Entering details of the science student

Enter the first name: kashif
Enter the second name: ali
Enter the age: 20
Enter the roll_no: 211
Enter the subject1 of the science student: E
Enter the subject2 of the science student: F
Enter the subject3 of the science student: G

Displaying the details of the science student

First Name = kashif
Last Name = ali
Age = 20
Roll Number = 211
Subject1 of the science student = E
Subject2 of the science student = F
Subject3 of the science student = G

LAB Evaluation

Student Name:		Instructor Name	Syed M Hamedoon
Registration No:		Assessment	CLO3,P2 / CLO4, P4
Date:		Total Marks	

- **Note:** Lab rubrics page must be attached at the end of the lab