

## Basic and Advance C++ Programs

- 21.** Define a class *Fahrenheit* with float temp as data member. Define another class *Celsius* with float temperature as data member. Both classes have member functions to input and print data. Write a non-member function that receives objects of both the classes and declare which one is higher than another according to their values. Also define main() to test the function. Define all member functions outside the class. (Formula for converting Celsius to Fahrenheit is  $F = (9C/5) + 32$ ). **Use the concept of friend function.**

### CODE:

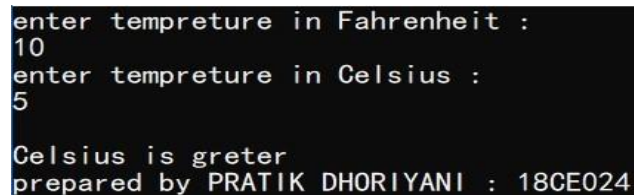
```
#include<iostream>
using namespace std;
class Celsius;
class Fahrenheit
{
    float temp;
public:
    void getdata()
    {
        cout<<"enter temperture in Fahrenheit : \n";
        cin>>temp;
    }
    friend void compare(Fahrenheit,Celsius);
};
class Celsius
{
    float temperture;
public:
    void getdata()
    {
        cout<<"enter temperture in Celsius : \n";
        cin>>temperture;
    }
    friend void compare(Fahrenheit,Celsius);
};
void compare(Fahrenheit f,Celsius c)
{
    float p;
    p=((9*c.temperture)/5)+32;
    if(f.temp==p)
    {
        cout<<"\nbooth tempertures are EQUAL";
    }
}
```

## Basic and Advance C++ Programs

```
    }
    else if(f.temp>p)
    {
        cout<<"\nFahrenheit is greter";
    }
    else if(f.temp<p)
    {
        cout<<"\nCelsius is greter";
    }
}
int main()
{
    Fahrenheit f;
    Celsius c;
    f.getdata();
    c.getdata();

    compare(f,c);
    cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";
    return 0;
}
```

### OUTPUT:

A screenshot of a terminal window showing the output of a C++ program. The text is as follows:

```
enter temperture in Fahrenheit :
10
enter temperture in Celsius :
5

Celsius is greter
prepared by PRATIK DHORIYANI : 18CE024
```

- 22.** Create a Class Date having data members: int dd, mm, yyyy. Class has one member function to input the dates and another member function which prints the dates. Write a main() function which takes two dates as input. Write a friend function swapdates() which takes two objects by reference of type Date and swaps both the dates. **Use the concept of Friend function which takes objects by reference**

### CODE:

```
#include<iostream>
using namespace std;
class date
{
```

## Basic and Advance C++ Programs

```
    int dd,mm,yy;
public:
    void getdata()
    {
        cin>>dd>>mm>>yy;
    }
    void putdata()
    {
        cout<<dd<<"/"<<mm<<"/"<<yy;
    }
    friend void swapdate(date & , date &);
};
void swapdate(date &d1 , date &d2)
{
    date temp;
    temp=d1;
    d1=d2;
    d2=temp;
}
int main()
{
    date d1,d2;
    cout<<"\nenter date , month & year for DATE 1 : \n";
    d1.getdata();
    cout<<"\nenter date , month & year for DATE 2 : \n";
    d2.getdata();

    swapdate(d1,d2);

    cout<<"\nAFTER SWAPPING : \n";
    d1.putdata();
    cout<<"\n";
    d2.putdata();
    cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";
    return 0;
}
```

## OUTPUT:

## Basic and Advance C++ Programs

```
enter date , month & year for DATE 1 :
7
8
2001

enter date , month & year for DATE 2 :
5
5
2000

AFTER SWAPPING :
5/5/2000
7/8/2001
prepared by PRATIK DHORIYANI : 18CE024
```

23. Create a class Customer having data members: name of the customer and customer number in integer and member function to get customer data. Create another class Manager having data members: name of manager and employee id in integer and member function to get managers data. Class Manager also have member function get\_cust\_data () which takes objects of class Customer as input and prints the customers details and is a friend function of class Customer . Write a main () function to test all this function. Use the concepts of **Member function of one class can be a Friend Function of another class.**

### CODE:

```
#include<iostream>
using namespace std;
class customer;
class manager
{
    char name_m[10];
    int m_id;
public:
    void getdata()
    {
        cin>>name_m>>m_id;
    }
    void putdata()
    {
        cout<<name_m<<" ----- "<<m_id;
    }
    void get_cust_data(customer);
};

class customer
{
    char name_c[10];
```

## Basic and Advance C++ Programs

```
int c_no;
public:
void getdata()
{
    cin>>name_c>>c_no;
}
friend void manager::get_cust_data(customer);
};

void manager::get_cust_data(customer c)
{
    cout<<c.name_c<<"----- "<<c.c_no<<"\n";
}

int main()
{
    class customer c1,c2;
    class manager m;
    cout<<"enter name of manager and employee id : \n";
    m.getdata();
    cout<<"enter name of the 2 customer and customer number : \n";
    c1.getdata();
    c2.getdata();
    cout<<"\ndatails of MANAGER : \n";
    m.putdata();
    cout<<"\ndatails of CUSTOMER(S) : \n";
    m.get_cust_data(c1);
    m.get_cust_data(c2);
    cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";
    return 0;
}
```

## OUTPUT:

## Basic and Advance C++ Programs

```
enter name of manager and employee id :
pratik
24
enter name of the 2 customer and customer number :
dhruv
1
pruthi l
2

details of MANAGER :
pratik-----24
details of CUSTOMER (S) :
dhruv-----1
pruthi l-----2

prepared by PRATIK DHORIYANI : 18CE024
```

24. Create a class Child having data members: name of the child and gender and a member function to get and print child data. Create another class Parent which is a friend class of child class. Class Parent have member function ReadChildData() which takes child's object by reference as input argument and Reads the childs data and DisplayChildData() which takes childs object as argument and displays childs data. Use the concepts of **Friend Class**.

### CODE:

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

class parent
{
public:
    void ReadChildData(child &);
    void DisplayChildData(child);
};

class child
{
    char name[10];
    char gender[10];
public:
    friend class parent;
};

void parent::ReadChildData(child &c1)
{
    cin>>c1.name>>c1.gender;
}
```

## Basic and Advance C++ Programs

```
void parent::DisplayChildData(child c)
{
    cout<<c.name<<"\n"<<c.gender;
}
int main()
{
    child c;
    parent p;
    cout<<"enter the name & gender of child : \n";
    p.ReadChildData(c);
    cout<<"details of child : \n";
    p.DisplayChildData(c);
    cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";
    return 0;
}
```

### OUTPUT:

A screenshot of a terminal window showing the output of the C++ program. The text is as follows:

```
enter the name & gender of child :
pratik
male
details of child :
pratik
male
prepared by PRATIK DHORIYANI : 18CE024
```

- 25.** Check the following C++ code and find if there is any error in code, give justification for the error, correct the code and write the output:

#### 1) Example of const member functions

```
#include<iostream>
using namespace std;
class sample
{
    int m, n;
    public:
    void getdata();
    void putdata() const;
};
void sample::getdata()
{
    cout<< "Enter m & n";
```

## Basic and Advance C++ Programs

```
cin>>m>>n;
}
void sample::putdata() const
{
    m=12;
    n=34;
    cout<< " m = "<<m<<"n= "<<n;
}
int main()
{
    sample s1;
    s1.getdata();
    s1.putdata();
    return 0;
}
```

### Error:

C:\Users\I Net\Desktop\.cpp|17|error: assignment of member 'sample::m' in read-only object|

### Solution Of Code:

```
#include<iostream>
using namespace std;
class sample
{
    int m, n;
public:
    void getdata();
    void putdata();
};
void sample::getdata()
{
    cout<< "Enter m & n";
    cin>>m>>n;
}
void sample::putdata()
{
    m=12;
    n=34;
    cout<< " m = "<<m<<"n= "<<n;
```



## Basic and Advance C++ Programs

```
}  
int main()  
{  
sample s1;  
s1.getdata();  
s1.putdata();  
return 0;  
}
```

### Output:

```
Enter m & n2  
3  
m = 12n= 34
```

## 2. Example of (a)Pointer to data members, (b)Pointer to member functions

### (a)

```
#include<iostream>  
using namespace std;  
class student  
{  
public: int roll_no;  
};  
int main()  
{  
// declaring pointer to data member  
int student :: *p1 = &student::roll_no;  
student s;  
student *optr = &s;  
s->*p1 = 42;  
cout<<"Roll no is "<<s->*p1<<endl;  
optr.*p1 = 45;  
cout<<"Roll no is"<<optr.*p1<<endl;  
return 0;  
}
```

### Error:

C:\Users\I Net\Desktop\.cpp|17|error: no match for 'operator->\*' (operand types are 'student' and 'int student::\*')|

C:\Users\I Net\Desktop\.cpp|18|error: no match for 'operator->\*' (operand types are 'student' and 'int student::\*')|

## Basic and Advance C++ Programs

C:\Users\I Net\Desktop\cpp\19|error: cannot apply member pointer 'p1' to 'optr', which is of non-class type 'student\*'|

C:\Users\I Net\Desktop\cpp\20|error: cannot apply member pointer 'p1' to 'optr', which is of non-class type 'student\*'|

### Solution Of Code:

```
#include<iostream>
using namespace std;
class student
{

public: int roll_no; };

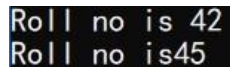
int main()

{

int student :: *p1 = &student::roll_no;
student s;
student *optr = &s;
s.*p1 = 42;
cout<<"Roll no is "<<s.*p1<<endl;
optr->*p1 = 45;
cout<<"Roll no is"<<optr->*p1<<endl;
return 0;

}
```

### Output:



```
Roll no is 42
Roll no is 45
```

(b)

```
#include<iostream>
class employee
{

public:
    void hello()
    {

        cout<<"Hi hello"<<endl;
```

## Basic and Advance C++ Programs

```
    }  
};  
int main()  
{  
    // declaring pointer to member function hello  
    void (employee::*fp)() = &employee::hello;  
    employee e;  
    employee *optr = &e;  
    (e->*fp)();  
    (optr.*fp)();  
    return 0;  
}
```

### Error:

C:\Users\I Net\Desktop\.cpp|7|error: 'cout' was not declared in this scope|

C:\Users\I Net\Desktop\.cpp|7|error: 'endl' was not declared in this scope|

C:\Users\I Net\Desktop\.cpp|16|error: no match for 'operator->\*' (operand types are 'employee' and 'void (employee::\*)()')|

C:\Users\I Net\Desktop\.cpp|17|error: cannot apply member pointer 'fp' to 'optr', which is of non-class type 'employee\*'|

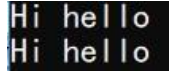
### Solution Of Code:

```
#include<iostream>  
using namespace std;  
class employee  
{  
public:  
    void hello()  
    {  
        cout<<"Hi hello"<<endl;  
    }  
};  
int main()  
{  
    //      declaring pointer to member function hello  
    void (employee::*fp)() = &employee::hello;  
    employee e;  
    employee *optr = &e;  
    (e.*fp)();  
    (optr->*fp)();  
}
```

## Basic and Advance C++ Programs

```
return 0;  
}
```

### Output:



```
Hi hello  
Hi hello
```

### 3. Example of Local Classes

```
#include<iostream>  
using namespace std;  
void testlocalclass()  
{  
    class Test  
    {  
        static int cnt;  
        public:  
        void set()  
        {cout<<"Enter Count: ";  
        cin>>cnt;  
        }  
        void get();  
    };  
    void Test:: get()  
    { cout<<"Count: = " <<cnt; }  
    Test t;  
    t.set();  
    t.get();  
}  
int main()  
{  
    testlocalclass();  
    return 0;  
}
```

### Error:

```
C:\Users\I Net\Desktop\cpp\8|error: local class 'class testlocalclass()::Test' shall not have  
static data member 'int testlocalclass()::Test::cnt' [-fpermissive]  
C:\Users\I Net\Desktop\cpp\18|error: qualified-id in declaration before '(' token|
```

## Basic and Advance C++ Programs

### Solution Of Code:

```
#include<iostream>
using namespace std;
void testlocalclass()
{
    static int cnt;
    class Test
    {
    public:
    void Set()
    {
        cout<<"Enter Count: ";
        cin>>cnt;
    }
    void get()
    {
        cout<<"Count: = " <<cnt;
    }
    };
    Test t;
    t.Set();
    t.get();
}

int main()
{
    testlocalclass();
    return 0;
}
```

### Output:

```
Enter Count: 4
Count: = 4
```

## Constructor and Destructor

**26.** Write a C++ program having class **time** with data members: hr, min and sec. Define following member functions.

- 1) getdata() to enter hour, minute and second values
- 2) putdata() to print the time in the format 11:59:59
- 3) default constructor

## Basic and Advance C++ Programs

- 4) parameterized constructor
- 5) copy constructor 6) Destructor.

Use 52 as default value for sec in parameterized constructor.

**Use the concepts of default constructor, parameterized constructor, Copy constructor, constructor with default arguments and destructor.**

### CODE:

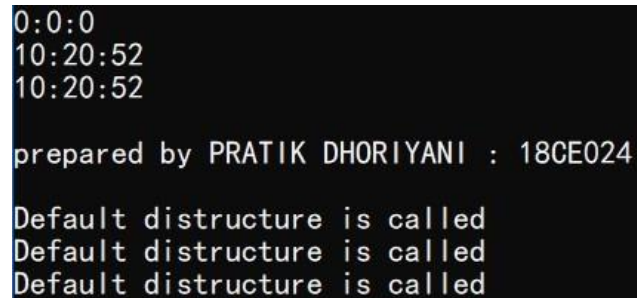
```
#include<iostream>
#include<windows.h>
using namespace std;
class time
{
    int hr;
    int mini;
    int sec;
public:
    time():hr(0),mini(0),sec(0){ }

    time(int x,int y,int z=52)
    {
        hr=x;
        mini=y;
        sec=z;
    }
    void putdata()
    {
        cout<<hr<<":"<<mini<<":"<<sec<<"\n";
    }
    time(time &t)
    {
        hr=t.hr;
        mini=t.mini;
        sec=t.sec;
    }
    ~time()
    {
        cout<<"Default distructure is called\n";
    }
}
```

## Basic and Advance C++ Programs

```
};  
int main()  
{  
    time t1,t2(10,20);  
    t1.putdata();  
    t2.putdata();  
    time t3(t2);  
    t3.putdata();  
    cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";  
    return 0;  
}
```

### OUTPUT:



A screenshot of a terminal window showing the output of a C++ program. The output consists of several lines of text: a timestamp '0:0:0', another timestamp '10:20:52', a third timestamp '10:20:52', a line 'prepared by PRATIK DHORIYANI : 18CE024', and three lines of 'Default distructure is called' (note the misspelling of 'destructor').

```
0:0:0  
10:20:52  
10:20:52  
  
prepared by PRATIK DHORIYANI : 18CE024  
  
Default distructure is called  
Default distructure is called  
Default distructure is called
```

## Operator Overloading & Type Conversion

- 27.** Create a class Number having int num as member. The class has input and output functions. Overload unary operator (++) such that it supports  $N1=N2++$  and  $N3=++N1$  and Overload unary (-) such that it supports  $N3 = - N3$ . Also define default, parameterized and copy constructor for the class. Also explain use of nameless object in operator overloading. **Use the concept of Overloading Unary Operators. Operator overloading is also known as Compile Time Polymorphism or static binding.**

### CODE:

```
#include<iostream>  
using namespace std;  
class Number  
{
```

## Basic and Advance C++ Programs

```
public :
    int num;
    Number(int x)
    {
        num=x;
    }
    Number()
    {
        num=0;
    }

    Number operator++()
    {
        ++num;
        return *this;
    }

    Number operator++(int)
    {
        Number n;
        n.num=num++;
        return n;
    }

    Number operator-()
    {
        num=-num;
        return num;
    }
};

int main()
{
    Number N1,N2(10),N3;
    N1=N2++;
    cout<<"Value after post-fix increment : "<<N1.num<<endl;
    N3=++N1;
    cout<<"Value after pre-fix increment : "<<N3.num<<endl;
    N3=-N3;
    cout<<"Value of Negative : "<<N3.num<<endl;
    cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";
}
```



## Basic and Advance C++ Programs

```
    return 0;  
}
```

**OUTPUT:**

```
Value after post-fix increment : 10  
Value after pre-fix increment : 11  
Value of Negative : -11  
  
prepared by PRATIK DHORIYANI : 18CE024
```

### Explain use of nameless object in operator overloading :

When we want to return an object from member function of class without creating an object, for this: we just call the constructor of class and return it to calling function and there is an object to hold the reference returned by constructor.

This concept is known as **nameless temporary objects**, using this we are going to implement a C++ program for **pre-decrement operator overloading**.

- 28.** Create a class complex having data members int real , img and member function to print data. Overload Unary operator (-) using friend function such that it supports -C1 where C1 is the object of class complex. Also define default, parameterized and copy constructor for the class.

**Use the concept of Overloading Unary Operators with friend function.**

### CODE:

```
#include<iostream>  
using namespace std;  
class complex  
{  
    int real;  
    int img;  
public:  
    complex()  
    {  
        real=0;  
        img=0;  
    }  
}
```

## Basic and Advance C++ Programs

```
complex(int a,int b)
{
    real=a;
    img=b;
}
complex(complex &x)
{
    real=x.real;
    img=x.img;
}
void getdata()
{
    cout<<"enter value of REAL & IMG value of COMPLEX num : \n";
    cin>>real>>img;
}
void putdata()
{
    cout<<"\n"<<"REAL : "<<real<<" & "<<"IMG : "<<img;
}
friend void operator - (complex &);
};
void operator - (complex &c)
{
    c.real = -c.real;
    c.img = -c.img;
}
int main()
{
    complex c1;
    c1.getdata();
    -c1;
    cout<<"\nAFTER CALLING -C1 VALUE OF REAL & IMG : \N";
    c1.putdata();

    cout<<"\n\nprepared by PRATIK DHORIYANI : 18CE024\n\n";
    return 0;
}
```

### OUTPUT:

## Basic and Advance C++ Programs

```
enter value of REAL & IMG value of COMPLEX num :
2
3

AFTER CALLING -C1 VALUE OF REAL & IMG : N
REAL : -2 & IMG : -3

prepared by PRATIK DHORIYANI : 18CE024
```

29. Create a class String having character array. Class includes constructor and required member functions to get and display the object. Overload the operators  $+(s3=s1+s2)$ ,  $==(s1<s2)$ ,  $+=(s1+=s2)$  for the class. **Use the concept of Overloading Binary Operators**

### CODE:

```
#include<iostream>
#include<string.h>
using namespace std;
class string1
{
public:
    char x[30];
public:
    void getdata()
    {
        cout<<"enter the string : \n";
        cin>>x;
    }
    void putdata()
    {
        cout<<"\n"<<x;
    }
    string1()
    {
        strcpy(x,"");
    }
    string1(char temp[])
    {
        strcpy(x,temp);
    }
    string1 operator + (string1 s2)
```

## Basic and Advance C++ Programs

```
{
    string1 s3;
    strcpy(s3.x,x);
    strcat(s3.x,s2.x);
    return s3;
}

void operator < (string1 s2)
{
    if(strlen(x)>strlen(s2.x))
    {
        cout<<"s1 is greter than s2";
    }
    else if(strlen(x)<strlen(s2.x))
    {
        cout<<"s1 is less than s2";
    }
    else if(strlen(x)==strlen(s2.x))
    {
        cout<<"s1 is equal to s2";
    }
}

void operator += (string1 s2)
{
    string1 s1 = strcat(x,s2.x);
}

};

int main()
{
    char p[]="dhoriyani";
    string1 s1,s2(p),s3;
    s1.getdata();

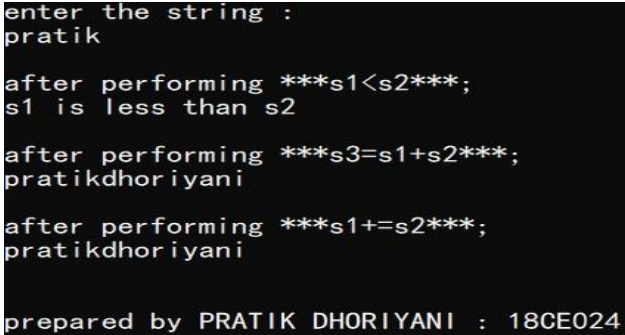
    cout<<"\nafter performing ***s1<s2***;\n";
    s1<s2;

    s3=s1+s2;
    cout<<"\n\nafter performing ***s3=s1+s2***;";
    s3.putdata();
}
```

## Basic and Advance C++ Programs

```
cout<<"\n\nafter performing ***s1+=s2***";
s1+=s2;
s1.putdata();
cout<<"\n\n";
cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";
return 0;
}
```

### OUTPUT:



```
enter the string :
pratik

after performing ***s1<s2***;
s1 is less than s2

after performing ***s3=s1+s2***;
pratikdhoriyani

after performing ***s1+=s2***;
pratikdhoriyani

prepared by PRATIK DHORIYANI : 18CE024
```

- 30.** Create a class Measure having members: meter and cm. The class has get( ) and put( ) functions. Overload operator + and – such that they support  $M1=M2+15$  and  $M3=M1 - 4.5$ . Also overload + and – such that they support  $M1=5.0+M2$  and  $M3=2.0 - M4$ . Write a main( ) to test the class.

**Use the concept of Overloading Binary Operators with friend function.**

### CODE:

#### With Nameless Object:

```
#include<iostream>
using namespace std;
class measure
{
    int meter,cm;
public:
    measure()
    {
        meter=0;
        cm=0;
    }
    measure(int m,int c)
```

## Basic and Advance C++ Programs

```
{
    meter=m;
    cm=c;
}
void get()
{
    cout<<"\nenter values of METER & CM : \n";
    cin>>meter>>cm;
}
void put()
{
    cout<<"\n"<<"METER : "<<meter<<"\n"<<"CM : "<<cm<<"\n";
}
measure operator + (int x)
{
    return (measure(meter+x , cm+x));
}
measure operator - (double x)
{
    return (measure(meter-x , cm-x));
}
friend measure operator + (double x,measure m);
friend measure operator - (double x,measure m);
};
measure operator + (double x,measure m)
{
    return (measure(x+m.meter , x+m.cm));
}
measure operator - (double x,measure m)
{
    return (measure(x-m.meter , x-m.cm));
}
int main()
{
    measure m1,m2,m3,m4;
    cout<<"enter values for OBJECT M2 : ";
    m2.get();
    cout<<"enter values for OBJECT M4 : ";
    m4.get();
    cout<<"\nafter performing M1=M2+15 : ";
```

## Basic and Advance C++ Programs

```
m1=m2+15;
cout<<"\nvalue of M1 object : ";
m1.put();
cout<<"\nafter performing M3=M1-4.5 : ";
m3=m1-4.5;
cout<<"\nvalue of M3 object : ";
m3.put();
cout<<"\nafter performing M1=5.0+M2 : ";
m1=5.0 + m2;
cout<<"\nvalue of M1 object : ";
m1.put();
cout<<"\nafter performing M3=2.0-M4 : ";
m3=2.0 - m4;
cout<<"\nvalue of M3 object : ";
m3.put();
cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";
return 0;

}
```

### Without Nameless Object:

```
#include<iostream>
using namespace std;
class measure
{
    int meter,cm;
public:
    void get()
    {
        cout<<"\nenter values of METER & CM : \n";
        cin>>meter>>cm;
    }
    void put()
    {
        cout<<"\n"<<"METER : "<<meter<<"\n"<<"CM : "<<cm<<"\n";
    }
    measure operator + (int x)
    {
        measure t1;
        t1.meter=meter + x;
```

## Basic and Advance C++ Programs

```
t1.cm=cm + x;
return t1;
}
measure operator - (double x)
{
    measure t2;
    t2.meter=meter - x;
    t2.cm=cm - x;
    return t2;
}
friend measure operator + (double x,measure m);
friend measure operator - (double x,measure m);
};
measure operator + (double x,measure m)
{
    measure t3;
    t3.meter=x + m.meter;
    t3.cm=x + m.cm;
    return t3;
}
measure operator - (double x,measure m)
{
    measure t3;
    t3.meter=x - m.meter;
    t3.cm=x - m.cm;
    return t3;
}
int main()
{
    measure m1,m2,m3,m4;
    cout<<"enter values for OBJECT M2 :";
    m2.get();
    cout<<"enter values for OBJECT M4 :";
    m4.get();
    cout<<"\nafter performing M1=M2+15 : ";
    m1=m2+15;
    cout<<"\nvalue of M1 object : ";
    m1.put();
    cout<<"\nafter performing M3=M1-4.5 : ";
    m3=m1-4.5;
```



## Basic and Advance C++ Programs

```
    cout<<"\nvalue of M3 object : ";
    m3.put();
    cout<<"\nafter performing M1=5.0+M2 : ";
    m1=5.0 + m2;
    cout<<"\nvalue of M1 object : ";
    m1.put();
    cout<<"\nafter performing M3=2.0-M4 : ";
    m3=2.0 - m4;
    cout<<"\nvalue of M3 object : ";
    m3.put();
    cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";
    return 0;

}
```

### OUTPUT:

```
enter values for OBJECT M2 :
enter values of METER & CM :
5
10
enter values for OBJECT M4 :
enter values of METER & CM :
5
50

after performing M1=M2+15 :
value of M1 object :
METER : 20
CM : 25

after performing M3=M1-4.5 :
value of M3 object :
METER : 15
CM : 20

after performing M1=5.0+M2 :
value of M1 object :
METER : 10
CM : 15

after performing M3=2.0-M4 :
value of M3 object :
METER : -3
CM : -48

prepared by PRATIK DHORIYANI : 18CE024
```

## Basic and Advance C++ Programs

- 31.** Create a class Celsius with float. Define appropriate member functions such that it support the statements: C1=30.5F; float temperature; temperature=C2;  
Use the concept of **Type conversion from basic type to class type and class type to basic type.**

### CODE:

```
#include<iostream>
using namespace std;
class celsius
{
    float temp;
public:
    celsius(float x)    //ONE ARGUMENT CONSTRUCTURE
    {
        temp=x;
    }
    operator float()    //OVERLOAD CASTING OPERATOR FUNCTION
    {
        return temp;
    }
    void put()
    {
        cout<<temp;
    }
};
int main()
{
    celsius c1=30.5f,c2(40.5);
    float temperture;
    c1.put();
    temperture=c2;
    cout<<"\n"<<temperture;
    cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";
    return 0;
}
```

### OUTPUT:

## Basic and Advance C++ Programs

```
30.5  
40.5  
prepared by PRATIK DHORIYANI : 18CE024
```

- 32.** Create classes Celsius and Fahrenheit with float. Define appropriate member functions such that they support the statements in main( ): Celsius C1, C2=5.0; Fahrenheit F1, F2; F1=C2; C1=F2;

Use the concepts of **Type conversion from class type to class type**.

**Write this Program in two ways.**

**Define appropriate member function in class Celsius.**

**Define appropriate member function in class Fahrenheit.**

### CODE(1):

```
#include<iostream>  
using namespace std;  
class celsius;  
class Fahrenheit  
{  
    float temp;  
public:  
    Fahrenheit()  
    {  
        temp=0;  
    }  
    Fahrenheit(float x)  
    {  
        temp=x;  
    }  
    void display()  
    {  
        cout<<temp;  
    }  
    Fahrenheit(celsius c);  
};  
class celsius  
{  
    float t;
```

## Basic and Advance C++ Programs

```
public:
    celsius()
    {
        t=0;
    }
    celsius(float x)
    {
        t=x;
    }
    void put()
    {
        cout<<t;
    }
    float getcel()
    {
        return t;
    }
};
Fahrenheit::Fahrenheit(celsius c)
{
    temp=((9)*c.getcel())/5 + 32;
}
int main()
{
    celsius c1,c2=5.0;
    Fahrenheit f1,f2;
    f1=c2;
    c2.put();
    f1.display();
    cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";
}
```

### CODE(2):

```
#include<iostream>
using namespace std;
class Fahrenheit
{
    float temp;
public:
    Fahrenheit()
```

## Basic and Advance C++ Programs

```
{
    temp=0;
}
Fahrenheit(float x)
{
    temp=x;
}
void display()
{
    cout<<temp;
}
float getfah()
{
    return temp;
}
};
class celsius
{
    float t;
public:
    celsius()
    {
        t=0;
    }
    celsius(float x)
    {
        t=x;
    }
    void put()
    {
        cout<<t;
    }
    celsius(Fahrenheit f)
    {
        t=(f.getfah() - 32)*(float(5)/9);
    }
};

int main()
{
```

## Basic and Advance C++ Programs

```
celsius c1,c2;
Fahrenheit f1,f2=41.0;
c1=f2;
c1.put();
f2.display();
cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";
}
```

### CODE(3):

```
#include<iostream>
using namespace std;
class celsius;
class Fahrenheit
{
    float temp;
public:
    Fahrenheit()
    {
        temp=0;
    }
    void display()
    {
        cout<<temp;
    }
    Fahrenheit (float a)
    {
        temp=a;
    }
};
class celsius
{
    float t;
public:
    celsius()
    {
        t=0;
    }
    celsius(float x)
    {
        t=x;
    }
};
```

## Basic and Advance C++ Programs

```
    }
    void put()
    {
        cout<<t;
    }
    operator Fahrenheit()
    {
        Fahrenheit f;
        f=(9*t)/5 + 32;
        return f;

        //also we can write ***return (9*t)/5 + 32;***
    }
};
int main()
{
    celsius c1,c2=5.0;
    Fahrenheit f1,f2;
    f1=c2;
    c2.put();
    f1.display();
    cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";
}
```

### CODE(4):

```
#include<iostream>
using namespace std;
class Fahrenheit;
class celsius
{
    float t;
public:
    celsius()
    {
        t=0;
    }
    celsius(float x)
    {
        t=x;
    }
}
```

## Basic and Advance C++ Programs

```
void put()
{
    cout<<t;
}
};
class Fahrenheit
{
    float temp;
public:
    Fahrenheit()
    {
        temp=0;
    }
    void display()
    {
        cout<<temp;
    }
    Fahrenheit (float a)
    {
        temp=a;
    }
    operator celsius()
    {
        celsius c;
        c=(temp-32)*(float(5)/9);
        return c;

        //also we can write ***(temp-32)*(float(5)/9);***
    }
};
int main()
{
    celsius c1,c2;
    Fahrenheit f1,f2=41.0;
    c1=f2;
    c1.put();
    f2.display();
    cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";
}
```



## Basic and Advance C++ Programs

### OUTPUT:

```
541
prepared by PRATIK DHORIYANI : 18CE024
```

### Inheritance

- 34.** Define a Base Class Vegetable having data member Color and member function getdata() which takes color as an input and putdata() which print the color as an output. Vegetable Class has one subclass named Tomato having data members weight and size and member function gtdata() which takes weight and size as an input and ptdata() which prints weight and size as output. Write a C++ Program which inherits the data of Vegetable class in Tomato class using **Single Inheritance**.

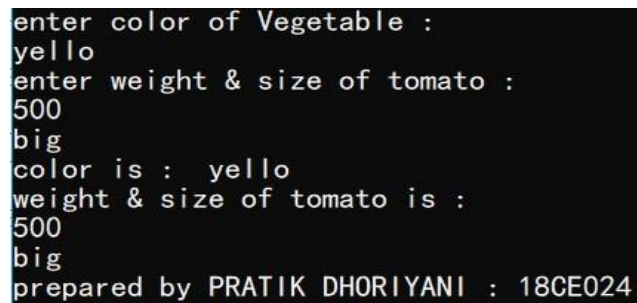
### CODE:

```
#include<iostream>
using namespace std;
class Vegetable
{
public:
    char color[10];
    void getdata()
    {
        cout<<"enter color of Vegetable : \n";
        cin>>color;
    }
    void putdata()
    {
        cout<<"color is : "<<color;
    }
};
class Tomato : public Vegetable
{
    int w;
    string s;
public:
    void gtdata()
    {
        cout<<"enter weight & size of tomato : \n";
        cin>>w>>s;
    }
};
```

## Basic and Advance C++ Programs

```
    }  
    void ptdata()  
    {  
        cout<<"\nweight & size of tomato is : \n"<<w<<"\n"<<s;  
    }  
};  
int main()  
{  
    Tomato o;  
    o.getdata ();  
    o.gtdata();  
    o.putdata();  
    o.ptdata();  
    cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";  
    return 0;  
}
```

### OUTPUT:



```
enter color of Vegetable :  
yello  
enter weight & size of tomato :  
500  
big  
color is : yello  
weight & size of tomato is :  
500  
big  
prepared by PRATIK DHORIYANI : 18CE024
```

- 35.** Write a program to create a class **Medicine** which stores type of medicine, name of company, date of manufacturing. Class **Tablet** is inherited from Medicine. Tablet class has name of tablet, quantity per pack, price of one tablet as members. Class **Syrup** is also inherited from Medicine and it has quantity per bottle, dosage unit as members. Both the classes contain necessary member functions for input and output data. Write a main( ) that enter data for tablet and syrup, also display the data. **Use the concepts of Hierarchical Inheritance.**

### CODE:

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

## Basic and Advance C++ Programs

```
{
public:
    char type[10];
    char comp[10];
    char date[10];
};
class Tablet : public Medicine
{
public:
    char tab[10];
    int qun;
    int price;
    void get()
    {
        cout<<"enter type of medicine, name of company, date of manufacturing : \n";
        cin>>type>>comp>>date;
        cout<<"enter name of tablet , quantity per pack , price of one tablet : \n";
        cin>>tab>>qun>>price;
    }
    void put()
    {
        cout<<"values of type of medicine, name of company, date of manufacturing : \n";
        cout<<type<<"\n"<<comp<<"\n"<<date<<"\n";
        cout<<"values of name of tablet , quantity per pack , price of one tablet : \n";
        cout<<tab<<"\n"<<qun<<"\n"<<price<<"\n";
    }
};
class Syrup : public Medicine
{
public:
    int qun;
    int dosage;
    void in()
    {
        cout<<"\nenter type of medicine, name of company, date of manufacturing : \n";
        cin>>type>>comp>>date;
        cout<<"enter quantity per bottle, dosage unit : \n";
        cin>>qun>>dosage;
    }
    void out()
```

## Basic and Advance C++ Programs

```
{
    cout<<"values of type of medicine, name of company, date of manufacturing : \n";
    cout<<type<<"\n"<<comp<<"\n"<<date<<"\n";
    cout<<"values of quantity per bottle, dosage unit : \n";
    cout<<qun<<"\n"<<dosage<<"\n";
}
};
int main()
{
    Tablet t;
    Syrup s;
    t.get();
    t.put();
    s.in();
    s.out();
    cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";
    return 0;
}
```

### OUTPUT:

```
enter type of medicine, name of company, date of manufacturing :
painkiller
pratikmed
07/08/2018
enter name of tablet , quantity per pack , price of one tablet :
dmed
5
10
values of type of medicine, name of company, date of manufacturing :
painkillerpratikmed
pratikmed
07/08/2018dmed
values of name of tablet , quantity per pack , price of one tablet :
dmed
5
10

enter type of medicine, name of company, date of manufacturing :
vomit
pratikmed
09/08/2018
enter quantity per bottle, dosage unit :
500
1/2
values of type of medicine, name of company, date of manufacturing :
vomit
pratikmed
09/08/2018
values of quantity per bottle, dosage unit :
500
1
prepared by PRATIK DHORIYANI : 18CE024
```

## Basic and Advance C++ Programs

- 36.** Create a class shape having data member shape\_name and member function to get and print shape\_name. Derive a Class Circle which is inherited publicly from class shape and having data members radius of a circle and member function to get and print radius of a circle. Derive a Class Area which is inherited publicly from Class Circle and having data members area\_of\_circle and member function display () which displays area of a circle. Use object of class Area in main () function and get and display all the information. **Use the concepts of Multilevel Inheritance.**

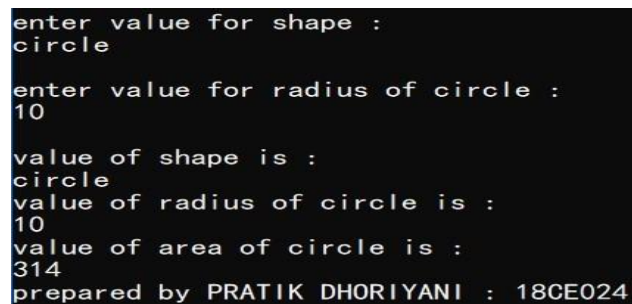
### CODE:

```
#include<iostream>
using namespace std;
class shape
{
public:
    char shape_name[10];
    void get()
    {
        cout<<"\nenter value for shape : \n";
        cin>>shape_name;
    }
    void print()
    {
        cout<<"\nvalue of shape is : \n";
        cout<<shape_name;
    }
};
class circle : public shape
{
public:
    int rad;
    void in()
    {
        cout<<"\nenter value for radius of circle : \n";
        cin>>rad;
    }
    void out()
    {
        cout<<"\nvalue of radius of circle is : \n";
```

## Basic and Advance C++ Programs

```
        cout<<rad;
    }
};
class area : public circle
{
public:
    int area;
    void display()
    {
        cout<<"\nvalue of area of circle is : \n";
        cout<<3.14*rad*rad;
    }
};
int main()
{
    area o;
    o.get();
    o.in();
    o.print();
    o.out();
    o.display();
    cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";
    return 0;
}
```

### OUTPUT:

A screenshot of a terminal window showing the output of a C++ program. The text is as follows:

```
enter value for shape :
circle

enter value for radius of circle :
10

value of shape is :
circle
value of radius of circle is :
10
value of area of circle is :
314
prepared by PRATIK DHORIYANI : 18CE024
```

- 37.** Define a class Hospital having rollno and name as data members and member function to get and print data. Derive a class Ward from class Hospital having data members: ward number and member function to get and print data. Derive another class Room from Hospital having data member bed number and nature of illness and member function to get and print data. Derive class Patient from Class Ward and Class Room. In main () declare 5

## Basic and Advance C++ Programs

object of Class Patient and get and display all the information. **Use the concept of Virtual Base Class and Hybrid Inheritance.**

### CODE:

```
#include<iostream>
using namespace std;
class hospital
{
public:
    int rollno;
    char name[10];
    void get()
    {
        cout<<"\nenter value for roll no & name of hospital : \n";
        cin>>rollno>>name;
    }
    void put()
    {
        cout<<"\nvalue for roll no & name of hospital : \n";
        cout<<rollno<<"\n"<<name;
    }
};
class ward : virtual public hospital
{
public:
    int wardno;
    void in()
    {
        cout<<"enter value for ward no : \n";
        cin>>wardno;
    }
    void out()
    {
        cout<<"\nvalue of ward no : \n";
        cout<<wardno;
    }
};
```

## Basic and Advance C++ Programs

```
class room : virtual public hospital
{
public:
    int bedno;
    char nature[10];
    void getdata()
    {
        cout<<"enter value for room no & nature of illness : \n";
        cin>>bedno>>nature;
    }
    void putdata()
    {
        cout<<"\nvalue of room no & nature of illness : \n";
        cout<<bedno<<"\n"<<nature;
    }
};

class patient : public ward , public room
{

};

int main()
{
    patient o[5];
    for(int i=0;i<2;i++)
    {
        cout<<"\n\n\n***** ENTER DETAILS FOR "<<(i+1)<<" PATIENT *****\n";
        o[i].get();
        o[i].in();
        o[i].getdata();
        o[i].put();
        o[i].out();
        o[i].putdata();
    }
    cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";
    return 0;
}
```



## Basic and Advance C++ Programs

### OUTPUT:

```
***** ENTER DETAILS FOR 1 PATIENT *****
enter value for roll no & name of hospital :
1
civil
enter value for ward no :
100
enter value for room no & nature of illness :
500
injured

value for roll no & name of hospital :
1
civil
value of ward no :
100
value of room no & nature of illness :
500
injured

***** ENTER DETAILS FOR 2 PATIENT *****
enter value for roll no & name of hospital :
2
civil
enter value for ward no :
101
enter value for room no & nature of illness :
501
accident

value for roll no & name of hospital :
2
civil
value of ward no :
101
value of room no & nature of illness :
501
accident
prepared by PRATIK DHORIYANI : 18CE024
```

38. Create a Class alpha having data member: int x and one argument constructor which initializes the value of x. It also has member function which displays the value of x. Create another class beta which contains data member: float y and one argument constructor which initializes the value of y. It also has member function which displays the value of y. Create a Class Gamma which publicly inherits from class alpha and class beta and has two data members: int m, n and a constructor which passes argument to the base class constructor as well as initializes its own data members. Class Gamma also has member function to print the values of m and n. Write main function which creates object of class Gamma which passes values of base class constructor as well as derived class constructor. **Use the concept of Multiple Inheritance and Constructor in Derived Class.**

### CODE:

```
#include<iostream>
```

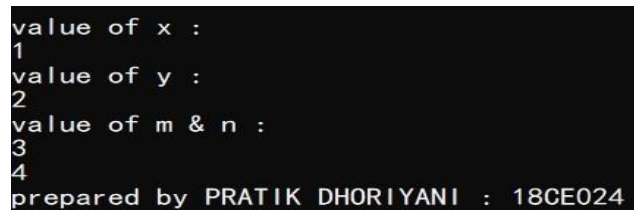
## Basic and Advance C++ Programs

```
using namespace std;
class alpha
{
public:
    int x;
    alpha(int temp)
    {
        x=temp;
    }
    void put()
    {
        cout<<"\nvalue of x : \n"<<x;
    }
};
class beta
{
public:
    float y;
    beta(int temp)
    {
        y=temp;
    }
    void putdata()
    {
        cout<<"\nvalue of y : \n"<<y;
    }
};
class gamma : public alpha , public beta
{
public:
    int m,n;
    gamma(int a,int b,int t1,int t2):alpha(a),beta(b)
    {
        m=t1;
        n=t2;
    }
    void print()
    {
        cout<<"\nvalue of m & n : \n"<<m<<"\n"<<n;
    }
}
```

## Basic and Advance C++ Programs

```
};  
int main()  
{  
    gamma o(1,2,3,4);  
    o.put();  
    o.putdata();  
    o.print();  
    cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";  
    return 0;  
}
```

### OUTPUT:



A screenshot of a terminal window showing the output of a C++ program. The output is as follows:  
value of x :  
1  
value of y :  
2  
value of m & n :  
3  
4  
prepared by PRATIK DHORIYANI : 18CE024

**39.** Create one application in a group of 3 person which implement all type of inheritance

### CODE:

```
#include<iostream>  
using namespace std;  
class type_industry  
{  
protected:  
    char t_name[20];  
public:  
    void get_t()  
    {  
        cout<<"\nenter type of industry : \n";  
        cin>>t_name;  
    }  
    void put_t()  
    {  
        cout<<"\ntype of industry : \n"<<t_name;
```

## Basic and Advance C++ Programs

```
    }  
};  
class about_industry:public type_industry  
{  
protected:  
    char n_name[20];  
    char state[20];  
    char city[20];  
public:  
    void get_a()  
    {  
        cout<<"\nenter the name of the industry : \n";  
        cin>>n_name;  
        cout<<"\nenter the name of the state : \n";  
        cin>>state;  
        cout<<"\nenter the name of the city : \n";  
        cin>>city;  
    }  
    void put_a()  
    {  
        cout<<"\nthe name of the industry : \n";  
        cout<<n_name;  
        cout<<"\nthe name of the state : \n";  
        cout<<state;  
        cout<<"\nthe name of the city : \n";  
        cout<<city;  
    }  
};  
class work_time:public about_industry  
{  
protected:  
    int work_p_day;  
    int work_p_week;  
public:  
    void get_wt()  
    {  
        cout<<"\nenter working hour of industry per day : \n";  
        cin>>work_p_day;  
        cout<<"\nenter working day of week of industry : \n";  
        cin>>work_p_week;
```

## Basic and Advance C++ Programs

```
}
void put_wt()
{
    cout<<"\nworking hour of industry per day : \n";
    cout<<work_p_day;
    cout<<"\nworking day of week of industry : \n";
    cout<<work_p_week;
}
};
class manager:virtual public work_time
{
protected:
int sal_m_p_m;
public:
void get_m()
{
    cout<<"\nenter salary of manager per month : \n";
    cin>>sal_m_p_m;
}
void put_m()
{
    cout<<"\nsalary of manager per month : \n";
    cout<<sal_m_p_m;
}
};
class worker:virtual public work_time
{
protected:
int sal_w_p_m;
public:
void get_w()
{
    cout<<"\nenter salary of worker per month : \n";
    cin>>sal_w_p_m;
}
void put_w()
{
    cout<<"\nsalary of worker per month : \n";
    cout<<sal_w_p_m;
}
}
```

## Basic and Advance C++ Programs

```
};
class watchman:public virtual work_time
{
protected:
int sal_wm_p_m;
public:
void get_wm()
{
cout<<"\nenter salary of watchman per month : \n";
cin>>sal_wm_p_m;
}
void put_wm()
{
cout<<"\nsalary of watchman per month : \n";
cout<<sal_wm_p_m;
}
};
class salary:public manager,public worker,public watchman
{
public:
void put_s()
{
cout<<"\nsalary of manager per year : "<<(sal_m_p_m*12)<<endl;
cout<<"\nsalary of worker per year : "<<(sal_w_p_m*12)<<endl;
cout<<"\nsalary of watchman per year : "<<(sal_wm_p_m*12)<<endl;
}
};
class display:public salary
{
public:
void display1()
{
get_t();
cout<<"\n*****\n";
get_a();
cout<<"\n*****\n";
get_wt();
cout<<"\n*****\n";
get_m();
cout<<"\n*****\n";
}
```

## Basic and Advance C++ Programs

```
get_w();
cout<<"\n*****\n";
get_wm();
cout<<"\n\n#####\n\n";
put_t();
cout<<"\n*****\n";
put_a();
cout<<"\n*****\n";
put_wt();
cout<<"\n*****\n";
put_m();
cout<<"\n*****\n";
put_w();
cout<<"\n*****\n";
put_wm();
cout<<"\n*****\n";
put_s();
}
};
int main()
{
    display o;
    o.display1();
    cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";
    return 0;
}
```

**OUTPUT:**

## Basic and Advance C++ Programs

```
enter type of industry :
diamond

*****

enter the name of the industry :
harikrushna foundation

enter the name of the state :

enter the name of the city :
surat

*****

enter working hour of industry per day :
8

enter working day of week of industry :
6

*****

enter salary of manager per month :
500000

*****

enter salary of worker per month :
50000

*****

enter salary of watchman per month :
25000

#####

type of industry :
diamond
*****

the name of the industry :
harikrushna
the name of the state :
foundation
the name of the city :
surat
*****

working hour of industry per day :
8
working day of week of industry :
6
*****

salary of manager per month :
500000
*****

salary of worker per month :
50000
*****

salary of watchman per month :
25000
*****

salary of manager per year : 6000000
salary of worker per year : 600000
salary of watchman per year : 300000
prepared by PRATIK DHORIYANI : 18CE024
```



## Basic and Advance C++ Programs

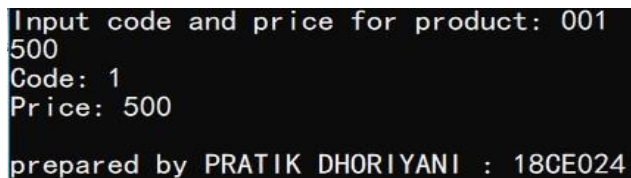
### Pointers and Virtual Functions

**40.** What is the output of the following code:

**(a) Pointer to Objects**

```
#include<iostream>
using namespace std;
class product
{
    int code;
    float price;
public:
    void getdata(int a, float b)
    {
        code=a;
        price=b;
    }
    void show()
    {
        cout<<"Code: "<<code<<endl;
        cout<<"Price: "<<price<<endl;
    }
};
int main()
{
    product * p = new product;
    product *d = p;
    int x,i;
    float y;
    cout<<"Input code and price for product: ";
    cin>>x>>y;
    p->getdata(x,y);
    d->show();
    cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";
    return 0;
}
```

**OUTPUT:**

A screenshot of a terminal window showing the output of the C++ program. The text is as follows:

```
Input code and price for product: 001
500
Code: 1
Price: 500

prepared by PRATIK DHORIYANI : 18CE024
```

**(b) This pointer**

```
#include<iostream>
using namespace std;
class student
{
```

## Basic and Advance C++ Programs

```
int roll_no;
float age;
public:
    student(int r, float a)
    {
        roll_no = r;
        age = a;
    }
    student & greater (student & x)
    {
        if(x.age>=age)
            return x;
        else
            return *this;
    }
    void display()
    {
        cout<<"Roll No "<<roll_no<<endl;
        cout<<"Age "<<age<<endl;
    }
};

int main()
{
    student s1 (23,18),s2 (30,20),s3 (45,16);
    student s = s1.greater(s3);
    cout<<"Elder Person is : "<<endl;
    s.display();
    cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";
    return 0;
}
```

### OUTPUT:

A screenshot of a terminal window with a black background and white text. The output shows the result of the C++ program execution. It displays the roll number and age of the elder person, followed by the author's name and ID.

```
Elder Person is :
Roll No 23
Age 18

prepared by PRATIK DHORIYANI : 18CE024
```

### Pointer To Derived Object:

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

class BC
{
public:
    int b;
    void show()
```

## Basic and Advance C++ Programs

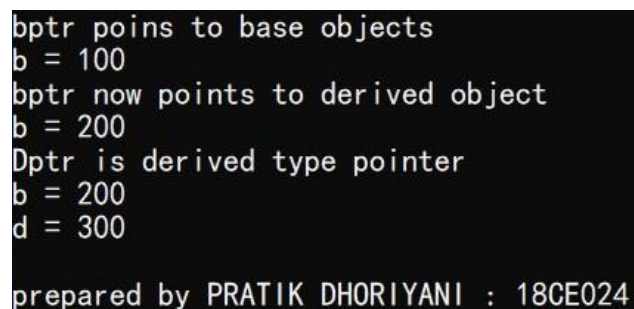
```
{
    cout<<"b = "<<b<<endl;
}
};

class DC : public BC
{
    public:
    int d;
    void show()
    {
        cout<<"b = "<<b<<endl;
        cout<<"d = "<<d<<endl;
    }
};

int main()
{
    BC *bptr;
    BC base;
    bptr = &base;
    bptr->b = 100;
    cout<<"bptr poins to base objects"<<endl;
    bptr->show();
    DC derived;
    bptr = &derived;
    bptr->b = 200;
    /*bptr->b = 300;*/ // wont work
    cout<<"bptr now points to derived object"<<endl;
    bptr->show();
    DC *dptr;
    dptr=&derived;
    dptr->d=300;
    cout<<"Dptr is derived type pointer"<<endl;
    dptr->show();
    cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";

    return 0;
}
```

## OUTPUT:

A screenshot of a terminal window showing the output of the C++ program. The text is as follows:  
bptr poins to base objects  
b = 100  
bptr now points to derived object  
b = 200  
Dptr is derived type pointer  
b = 200  
d = 300  
prepared by PRATIK DHORIYANI : 18CE024

## Basic and Advance C++ Programs

- 41.** Create a class Media that stores the title (a string) and price (float). Class Media has two argument constructor which initializes data members of class Media. Also declare a virtual function display () in Class Media. From the class Media derive two classes: Class book, which contains data member page count (int): and Class tape, which contains data member playing time in minutes (float). Both Class book and Class tape should have a constructor which initializes base class constructor as well as its own data members and display ( ) function which displays book details and tape details respectively. Write a main ( ) to test book and tape classes by creating instances of them, asking the user to fill data and displaying them. **Use the concept of Virtual function and Constructor in Derived Class. Virtual function is also known as Runtime Polymorphism or Dynamic Binding.**

### CODE:

```
#include<iostream>
#include<string>
using namespace std;
class Media
{
    string title;
    int price;
public:
    Media(string a,int b) : title(a),price(b) {}
    virtual void display(){ }
};

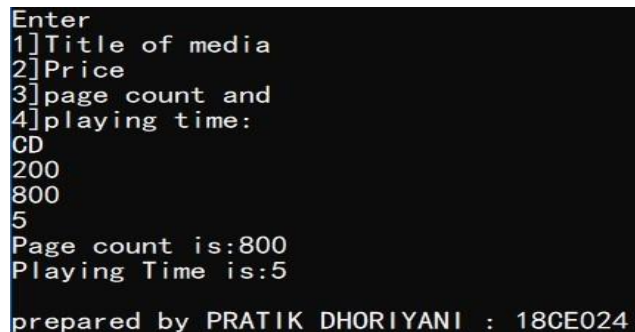
class Book : public Media
{
    int page_count;
public:
    Book(string a,int b,int c) : Media(a,b)
    {
        page_count=c;
    }
    void display()
    {
        cout<<"Page count is:"<<page_count<<endl;
    }
};

class Tape : public Media
```

## Basic and Advance C++ Programs

```
{
    float time;
public:
    Tape(string a,int b,float d):Media(a,b)
    {
        time=d;
    }
    void display()
    {
        cout<<"Playing Time is:"<<time<<endl;
    }
};
int main()
{
    string s;
    int price,pc;
    float time;
    cout<<"Enter \n1]Title of media \n2]Price \n3]page count and \n4]playing time:\n";
    cin>>s>>price>>pc>>time;
    Book b1(s,price,pc);
    Tape t1(s,price,time);
    b1.display();
    t1.display();
    cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";
    return 0;
}
```

### OUTPUT:

A screenshot of a terminal window showing the output of a C++ program. The text is as follows:

```
Enter
1]Title of media
2]Price
3]page count and
4]playing time:
CD
200
800
5
Page count is:800
Playing Time is:5
prepared by PRATIK DHORIYANI : 18CE024
```

42. Create a Abstract class vehicle having average as data and pure virtual function getdata() and putdata(). Derive class car and truck from class vehicle having data members: fuel

## Basic and Advance C++ Programs

type (petrol, diesel, CNG) and no of wheels respectively. Write a main ( ) that enters the data of two cars and a truck and display the details of them. **Use the concept of Abstract Base class and Pure Virtual functions.**

### CODE:

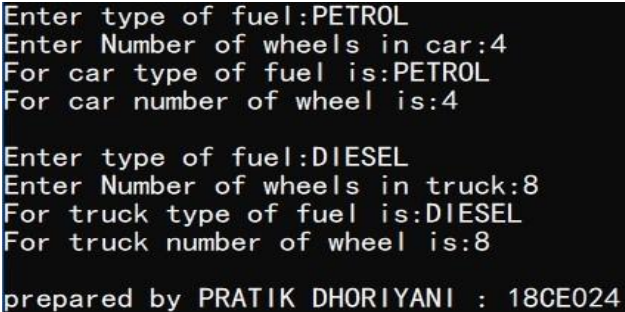
```
#include<iostream>
#include<string>
using namespace std;
class Vehical
{
    float avg;
public:
    virtual void getdata() = 0;
    virtual void putdata() = 0;
};
class Car : public Vehical
{
    string fuel_type;
    int no_of_wheel;
public:
    void getdata()
    {
        cout<<"Enter type of fuel:";
        cin>>fuel_type;
        cout<<"Enter Number of wheels in car:";
        cin>>no_of_wheel;
    }
    void putdata()
    {
        cout<<"For car type of fuel is:"<<fuel_type<<endl;
        cout<<"For car number of wheel is:"<<no_of_wheel<<endl;
    }
};
class Truck : public Vehical
{
    string fuel_type;
    int no_of_wheel;
public:
    void putdata()
```

## Basic and Advance C++ Programs

```
{
    cout<<"For truck type of fuel is:"<<fuel_type<<endl;
    cout<<"For truck number of wheel is:"<<no_of_wheel<<endl;

}
void getdata()
{
    cout<<"Enter type of fuel:";
    cin>>fuel_type;
    cout<<"Enter Number of wheels in truck:";
    cin>>no_of_wheel;
}
};
int main()
{
    Car c1;
    Truck t1;
    c1.getdata();
    c1.putdata();
    cout<<endl;
    t1.getdata();
    t1.putdata();
    cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";
    return 0;
}
```

### OUTPUT:



```
Enter type of fuel:PETROL
Enter Number of wheels in car:4
For car type of fuel is:PETROL
For car number of wheel is:4

Enter type of fuel:DIESEL
Enter Number of wheels in truck:8
For truck type of fuel is:DIESEL
For truck number of wheel is:8

prepared by PRATIK DHORIYANI : 18CE024
```

## File Handling

- 43.** Write a program that creates a text file that contains ABC...Z. A program should print the file in reverse order on the screen. i.e. ZYX...BA. Use concept of **Opening the file using**

## Basic and Advance C++ Programs

**constructor and open () function. Use all error handling functions like eof() , fail() , bad() , good() and functions for manipulation of file pointer like seekg() and tellg().**

### CODE:

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
    fstream f1("t1.txt",ios::out|ios::in);

    if(f1.bad())
    {
        cout<<"File is encrypted so ,Can not open.."<<endl;
    }
    else if(f1.fail())
    {
        cout<<"File does not exit"<<endl;
    }
    else if(f1.good())
    {
        cout<<"File successfully opened"<<endl;
        f1<<"ABCDEFGHJKLMNOPQRSTUVWXYZ";
    }
    if(f1.eof())
    {
        cout<<"File has reached end"<<endl;
    }
    int length;
    f1.seekg(0,ios::end);
    length=f1.tellg();
    char ch;
    for(int i=1;i<=length;i++)
    {
        f1.seekg(-1,ios::cur);
        f1>>ch;
        cout<<ch;
        f1.seekg(-1,ios::cur);
    }
    cout<<"\nprepared by PRATIK DHORIYANI : 18CE024\n\n";
}
```



## Basic and Advance C++ Programs

```
    return 0;  
}
```

### OUTPUT:

```
File successfully opened  
ZYXWVUTSRQPONMLKJIHGFEDCBA  
prepared by PRATIK DHORIYANI : 18CE024
```



t1 - Notepad

File Edit Format View Help

**ABCDEFGHIJKLMNOPQRSTUVWXYZ**