

# VIRTUAL FUNCTION

1. **Create a base class called Person with a virtual function work (). Derive two classes Employee and Manager from the base class. Implement the work () function for each class**

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

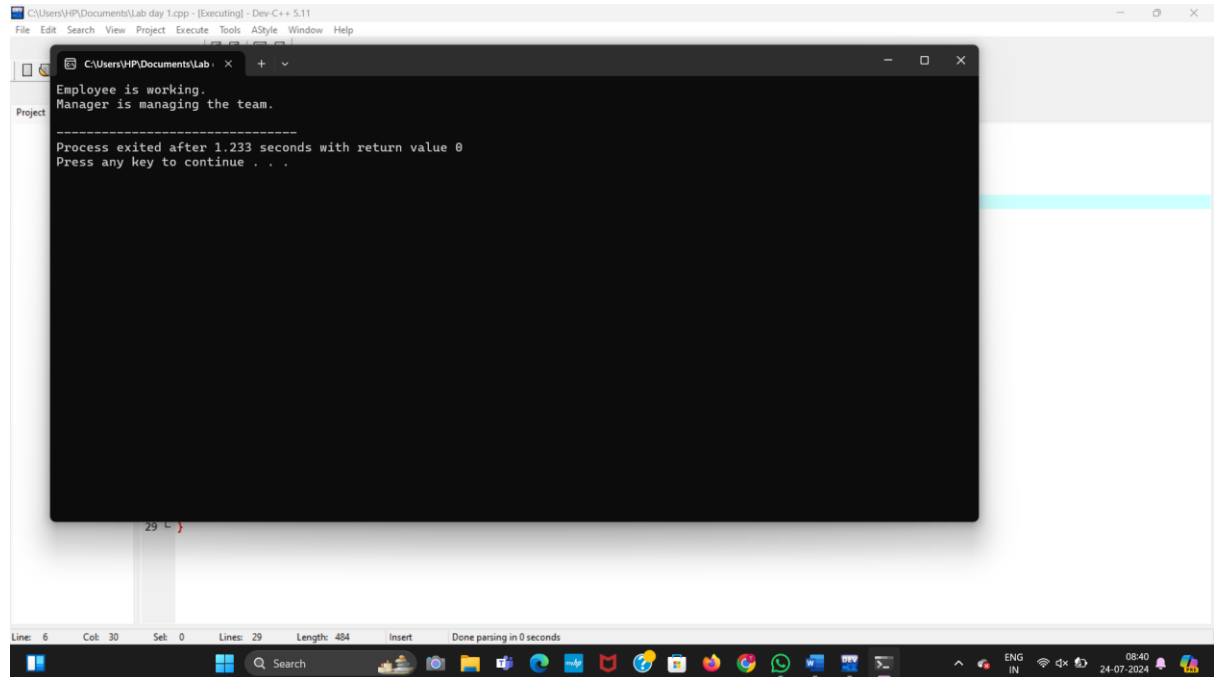
class Person {
public:
    virtual void work() = 0;
};

class Employee : public Person {
public:
    void work() override {
        cout << "Employee is working." << endl;
    }
};

class Manager : public Person {
public:
    void work() override {
        cout << "Manager is managing the team." << endl;
    }
};

int main() {
    Employee emp;
    Manager mgr;
    emp.work();
    mgr.work();
    return 0;
}
```

Output:



2. Create a base class called Animal with a virtual function eat (). Derive two classes Herbivore and Carnivore from the base class. Implement the eat function for each class.

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

```
class Animal {
public:
    virtual void eat() = 0;
};
```

```
class Herbivore : public Animal {
public:
    void eat() override {
        cout << "Herbivore is eating plants." << endl;
    }
};
```

```
class Carnivore : public Animal {
public:
    void eat() override {
        cout << "Carnivore is eating meat." << endl;
    }
};
```

```
int main() {
    Herbivore herb;
    Carnivore carn;
```

```

    herb.eat();
    carn.eat();
    return 0;
}

```

Output:

3. Create a base class called Shape with virtual functions area () and volume (). Derive two classes Sphere and Cylinder from the base class. Implement the area and volume () functions for each class

```

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

```

```

class Shape {
public:
    virtual double area() = 0; // pure virtual function
    virtual double volume() = 0; // pure virtual function
};

```

```

class Sphere : public Shape {
    double radius;
public:
    Sphere(double r) : radius(r) {}
    double area() override {
        return 4 * M_PI * radius * radius;
    }
    double volume() override {
        return (4.0/3.0) * M_PI * radius * radius * radius;
    }
}

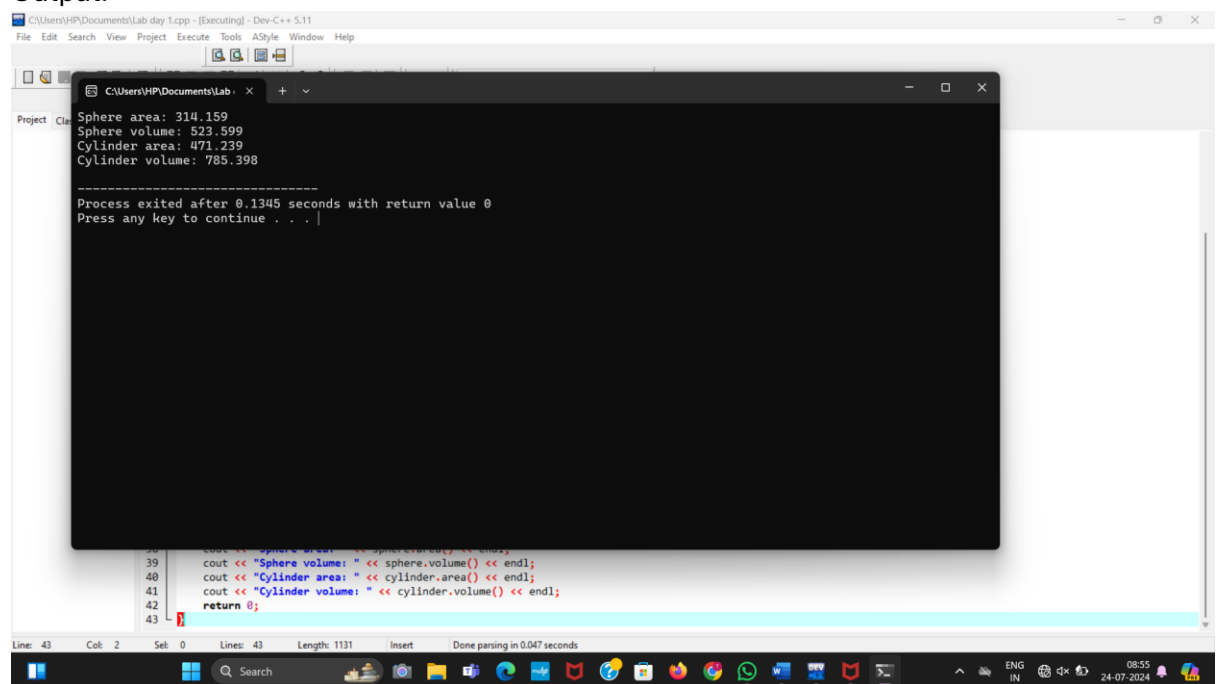
```

```
};

class Cylinder : public Shape {
    double radius, height;
public:
    Cylinder(double r, double h) : radius(r), height(h) {}
    double area() override {
        return 2 * M_PI * radius * (radius + height);
    }
    double volume() override {
        return M_PI * radius * radius * height;
    }
};

int main() {
    Sphere sphere(5);
    Cylinder cylinder(5, 10);
    cout << "Sphere area: " << sphere.area() << endl;
    cout << "Sphere volume: " << sphere.volume() << endl;
    cout << "Cylinder area: " << cylinder.area() << endl;
    cout << "Cylinder volume: " << cylinder.volume() << endl;
    return 0;
}
```

Output:



```

C:\Users\HP\Documents\Lab day 1.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help

Project: C:\Users\HP\Documents\Lab day 1.cpp
Sphere area: 314.159
Sphere volume: 523.599
Cylinder area: 471.239
Cylinder volume: 785.398

-----
Process exited after 0.1345 seconds with return value 0
Press any key to continue . . . |

38 cout << "Sphere area: " << sphere.area() << endl;
39 cout << "Sphere volume: " << sphere.volume() << endl;
40 cout << "Cylinder area: " << cylinder.area() << endl;
41 cout << "Cylinder volume: " << cylinder.volume() << endl;
42 return 0;
43
Line: 43 Col: 2 Sel: 0 Lines: 43 Length: 1131 Insert Done parsing in 0.047 seconds

```

4. Create a base class called Person with a virtual function greet(). Derive two classes Student and Teacher from the base class. implement the greet() function for each class  
#include <iostream>  
using namespace std;

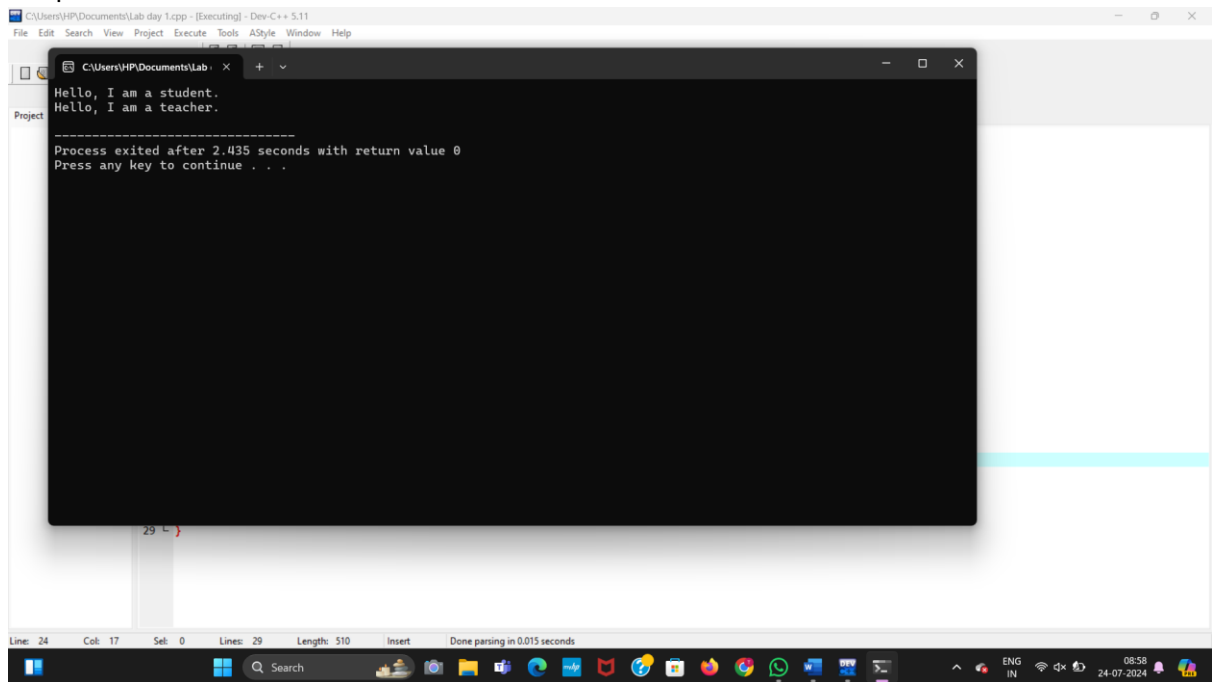
```
class Person {
public:
    virtual void greet() = 0; // pure virtual function
};
```

```
class Student : public Person {
public:
    void greet() override {
        cout << "Hello, I am a student." << endl;
    }
};
```

```
class Teacher : public Person {
public:
    void greet() override {
        cout << "Hello, I am a teacher." << endl;
    }
};
```

```
int main() {
    Student stu;
    Teacher teach;
    stu.greet();
    teach.greet();
    return 0;
}
```

Output:



The screenshot shows a Windows desktop with a Dev-C++ IDE. The IDE window is titled "C:\Users\HP\Documents\Lab day 1.cpp - [Executing] - Dev-C++ 5.11". The main window displays the output of the program, which is "Hello, I am a student." followed by "Hello, I am a teacher." on the next line. Below the output, it says "Process exited after 2.435 seconds with return value 0" and "Press any key to continue . . .". The IDE's status bar at the bottom shows "Line: 24 Col: 17 Sel: 0 Lines: 29 Length: 510 Insert Done parsing in 0.015 seconds". The Windows taskbar at the bottom shows the time as 08:58 on 24-07-2024.

```

C:\Users\HP\Documents\Lab day 1.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
C:\Users\HP\Documents\Lab day 1.cpp
Hello, I am a student.
Hello, I am a teacher.
-----
Process exited after 2.435 seconds with return value 0
Press any key to continue . . .
Line: 24 Col: 17 Sel: 0 Lines: 29 Length: 510 Insert Done parsing in 0.015 seconds
08:58 24-07-2024
```

5. Create a base class called Person with a virtual function greet(). Derive two classes Student and Teacher from the base class. implement the greet() function for each class

```
#include <iostream>
```

```
using namespace std;
```

```
class Shape {
public:
    virtual double area() = 0; // pure virtual function
    virtual double perimeter() = 0; // pure virtual function
};
```

```
class Rectangle : public Shape {
    double length, width;
public:
    Rectangle(double l, double w) : length(l), width(w) {}
    double area() override {
        return length * width;
    }
    double perimeter() override {
        return 2 * (length + width);
    }
};
```

```
class Triangle : public Shape {
    double a, b, c; // sides of the triangle
public:
    Triangle(double x, double y, double z) : a(x), b(y), c(z) {}
    double area() override {
        double s = (a + b + c) / 2;
        return sqrt(s * (s - a) * (s - b) * (s - c));
    }
    double perimeter() override {
        return a + b + c;
    }
};
```

```
int main() {
    Rectangle rect(4, 5);
    Triangle tri(3, 4, 5);
    cout << "Rectangle area: " << rect.area() << endl;
    cout << "Rectangle perimeter: " << rect.perimeter() << endl;
    cout << "Triangle area: " << tri.area() << endl;
    cout << "Triangle perimeter: " << tri.perimeter() << endl;
    return 0;
}
```

6. Create a base class called Shape with virtual functions area( ) and perimeter(). Derive two classes Rectangle and Triangle from the base class. Implement the area ( ) and perimeter ( ) functions for each class.

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

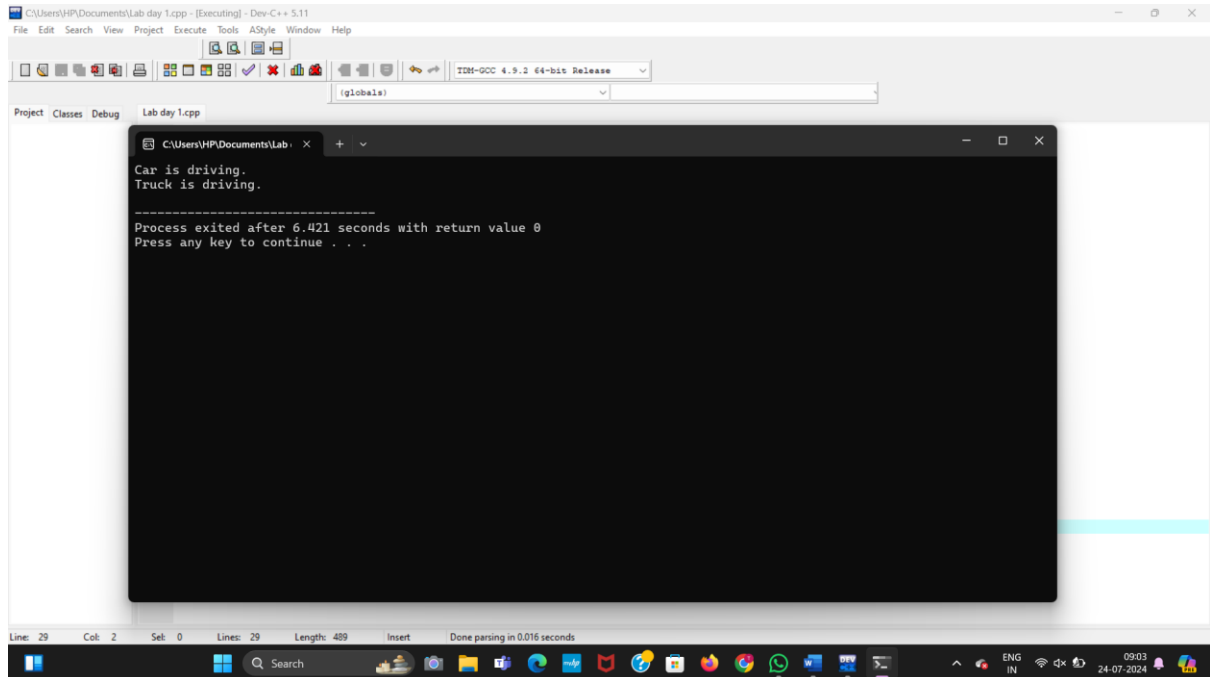
class Vehicle {
public:
    virtual void drive() = 0; // pure virtual function
};

class Car : public Vehicle {
public:
    void drive() override {
        cout << "Car is driving." << endl;
    }
};

class Truck : public Vehicle {
public:
    void drive() override {
        cout << "Truck is driving." << endl;
    }
};

int main() {
    Car car;
    Truck truck;
    car.drive();
    truck.drive();
    return 0;
}
```

Output:



7. Create a base class called Vehicle with a virtual function drive(). Derive two classes Car and Truck from the base class. Implement the drive() function for each class.

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

```
class Employee {
public:
    virtual double calculatePay() = 0; // pure virtual function
};
```

```
class Manager : public Employee {
    double salary;
public:
    Manager(double s) : salary(s) {}
    double calculatePay() override {
        return salary;
    }
};
```

```
class Engineer : public Employee {
    double hourlyRate;
    int hoursWorked;
public:
    Engineer(double rate, int hours) : hourlyRate(rate), hoursWorked(hours) {}
    double calculatePay() override {
        return hourlyRate * hoursWorked;
    }
};
```



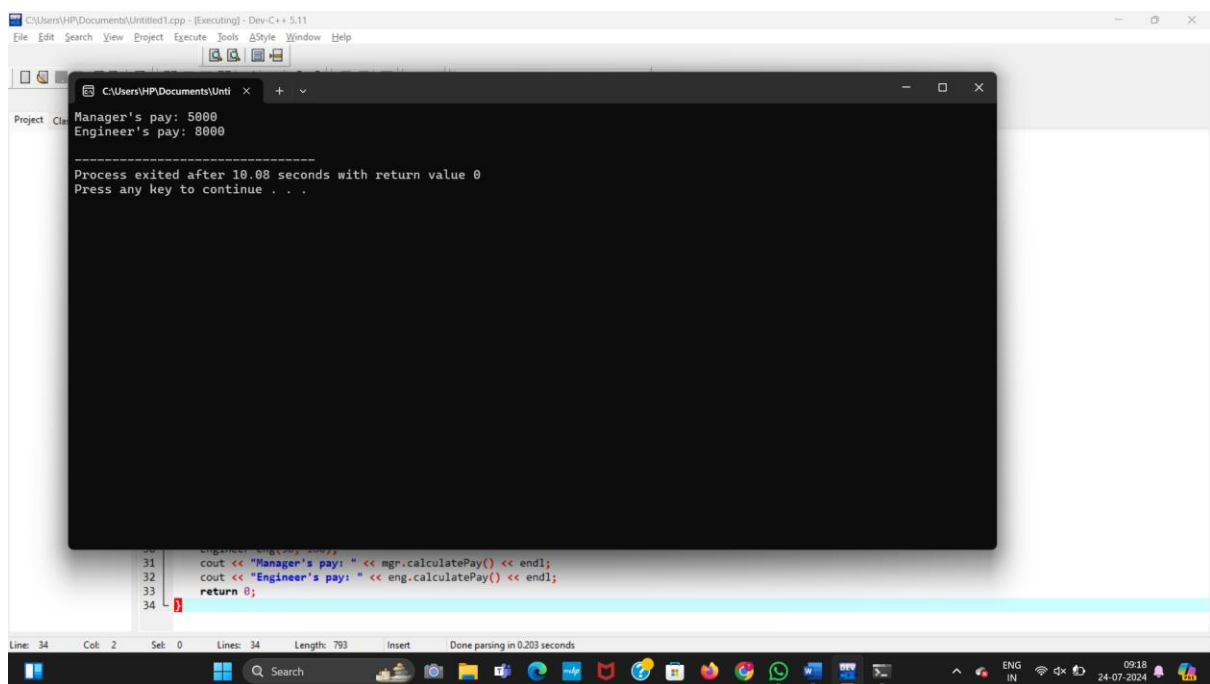
```

    }
};

int main() {
    Manager mgr(5000);
    Engineer eng(50, 160);
    cout << "Manager's pay: " << mgr.calculatePay() << endl;
    cout << "Engineer's pay: " << eng.calculatePay() << endl;
    return 0;
}

```

Output:



The screenshot shows a Windows desktop with a Dev-C++ IDE. The IDE window displays a C++ program that has been executed. The output window shows the following text:

```

Manager's pay: 5000
Engineer's pay: 8000
-----
Process exited after 10.08 seconds with return value 0
Press any key to continue . . .

```

The code editor shows the following code:

```

31 cout << "Manager's pay: " << mgr.calculatePay() << endl;
32 cout << "Engineer's pay: " << eng.calculatePay() << endl;
33 return 0;
34

```

8. Create a base class called Employee with a virtual function calculate Pay(). Derive two classes Manager and Engineer from the base class. Implement the calculatePay () function for each class.

```

#include <iostream>
using namespace std;

class Animal {
public:
    virtual void speak() = 0; // pure virtual function
};

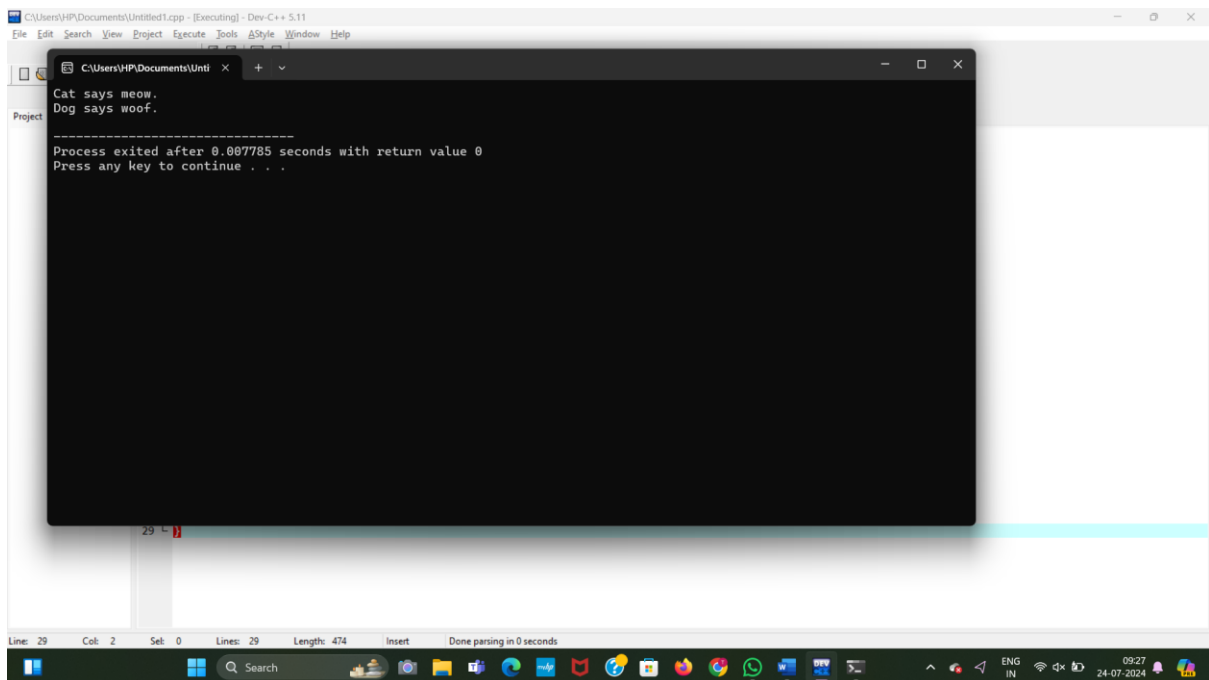
```

```
class Cat : public Animal {
public:
    void speak() override {
        cout << "Cat says meow." << endl;
    }
};

class Dog : public Animal {
public:
    void speak() override {
        cout << "Dog says woof." << endl;
    }
};

int main() {
    Cat cat;
    Dog dog;
    cat.speak();
    dog.speak();
    return 0;
}
```

Output:



```
C:\Users\HP\Documents\Untitled1.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help

C:\Users\HP\Documents\Untitled1.cpp
Cat says meow.
Dog says woof.

-----
Process exited after 0.007785 seconds with return value 0
Press any key to continue . . .

Line: 29 Col: 2 Sel: 0 Lines: 29 Length: 474 Insert Done parsing in 0 seconds
```

9. Create a base class called Animal with a virtual function speak(). Derive two classes Cat and Dog from the base class. Implement the speak() function for each class.

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

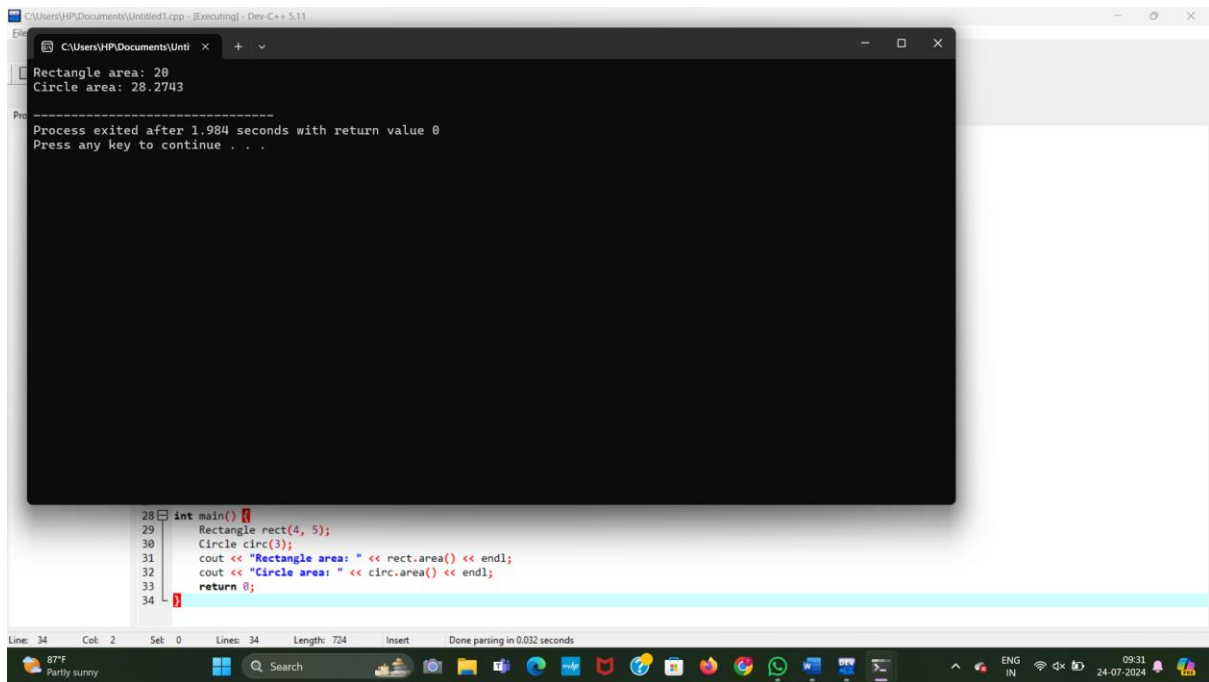
class Shape {
public:
    virtual double area() = 0; // pure virtual function
};

class Rectangle : public Shape {
    double length, width;
public:
    Rectangle(double l, double w) : length(l), width(w) {}
    double area() override {
        return length * width;
    }
};

class Circle : public Shape {
    double radius;
public:
    Circle(double r) : radius(r) {}
    double area() override {
        return M_PI * radius * radius;
    }
};

int main() {
    Rectangle rect(4, 5);
    Circle circ(3);
    cout << "Rectangle area: " << rect.area() << endl;
    cout << "Circle area: " << circ.area() << endl;
    return 0;
}
```

Output:



```
int main() {
    Rectangle rect(4, 5);
    Circle circ(3);
    cout << "Rectangle area: " << rect.area() << endl;
    cout << "Circle area: " << circ.area() << endl;
    return 0;
}
```

10 Create a base class called Shape with a virtual function area(). Derive two classes Rectangle and Circle from the base class. Implement the area() function for each class

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

```
class Shape {
public:
    virtual double area() = 0; // pure virtual function
};
```

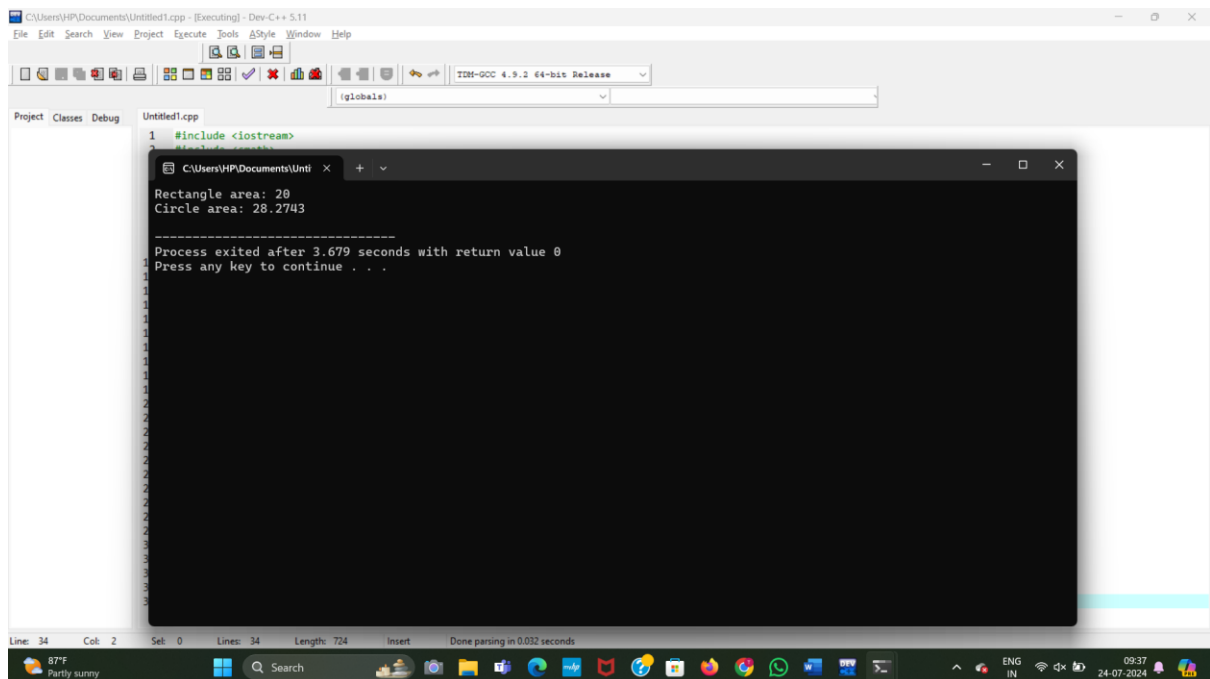
```
class Rectangle : public Shape {
    double length, width;
public:
    Rectangle(double l, double w) : length(l), width(w) {}
    double area() override {
        return length * width;
    }
};
```

```
class Circle : public Shape {
    double radius;
public:
    Circle(double r) : radius(r) {}
    double area() override {
        return M_PI * radius * radius;
    }
};
```

```
};

int main() {
    Rectangle rect(4, 5);
    Circle circ(3);
    cout << "Rectangle area: " << rect.area() << endl;
    cout << "Circle area: " << circ.area() << endl;
    return 0;
}
```

Output:



The screenshot shows a C++ IDE window titled "C:\Users\HP\Documents\Untitled1.cpp - [Executing] - Dev-C++ 5.11". The IDE has a menu bar (File, Edit, Search, View, Project, Execute, Tools, Style, Window, Help) and a toolbar. The main editor area shows the source code for "Untitled1.cpp" with line numbers 1 through 34. The code is as follows:

```
1 #include <iostream>
2 using namespace std;
3
4 struct Rectangle {
5     int length;
6     int breadth;
7
8     Rectangle(int l, int b) {
9         length = l;
10        breadth = b;
11    }
12
13    int area() {
14        return length * breadth;
15    }
16 };
17
18 struct Circle {
19     int radius;
20
21     Circle(int r) {
22         radius = r;
23     }
24
25     double area() {
26         return 3.14 * radius * radius;
27     }
28 };
29
30 int main() {
31     Rectangle rect(4, 5);
32     Circle circ(3);
33     cout << "Rectangle area: " << rect.area() << endl;
34     cout << "Circle area: " << circ.area() << endl;
35     return 0;
36 }
```

Below the editor, a black console window displays the output of the program:

```
Rectangle area: 20
Circle area: 28.2743

-----
Process exited after 3.679 seconds with return value 0
Press any key to continue . . .
```

The IDE status bar at the bottom shows "Line: 34, Col: 2, Sel: 0, Lines: 34, Length: 724, Insert, Done parsing in 0.032 seconds". The Windows taskbar at the very bottom shows the date and time as "24.07.2024 09:37" and the system language as "ENG IN".