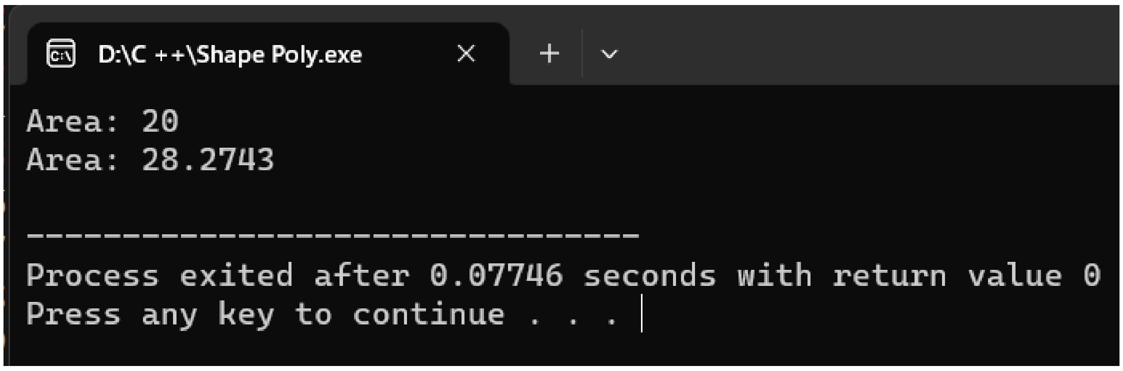
POLYMOPHISM

SHAPE

```
#include <iostream>

class Shape {
public:
    virtual double area() {
        return 0.0;
    }
};

class Rectangle : public Shape {
private:
    double length, width;

public:
    Rectangle(double l, double w) : length(l), width(w)
{}
```

```
    double area() override {
        return length * width;
    }
};

class Circle : public Shape {
private:
    double radius;

public:
    Circle(double r) : radius(r) {}

    double area() override {
        return 3.14159265359 * radius * radius;
    }
};

int main() {
    Shape* shapes[] = { new Rectangle(5, 4), new
Circle(3) };
```

```
    for (Shape* shape : shapes) {
        std::cout << "Area: " << shape->area() <<
std::endl;
    }

    return 0;
}
```



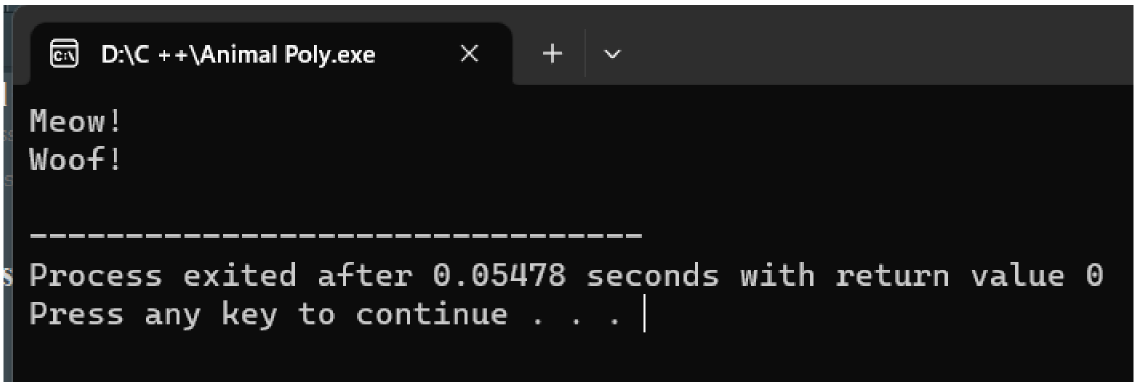
D:\C ++\Shape Poly.exe

```
Area: 20
Area: 28.2743

--------------------------------
Process exited after 0.07746 seconds with return value 0
Press any key to continue . . .
```

ANIMAL

```
#include <iostream>

class Animal {
public:
    virtual void speak() {
        std::cout << "Animal speaks." << std::endl;
    }
```

```
};

class Cat : public Animal {
public:
    void speak() override {
        std::cout << "Meow!" << std::endl;
    }
};

class Dog : public Animal {
public:
    void speak() override {
        std::cout << "Woof!" << std::endl;
    }
};

int main() {
    Animal* animals[] = { new Cat, new Dog };

    for (Animal* animal : animals) {
        animal->speak();
    }

    return 0;
}
```



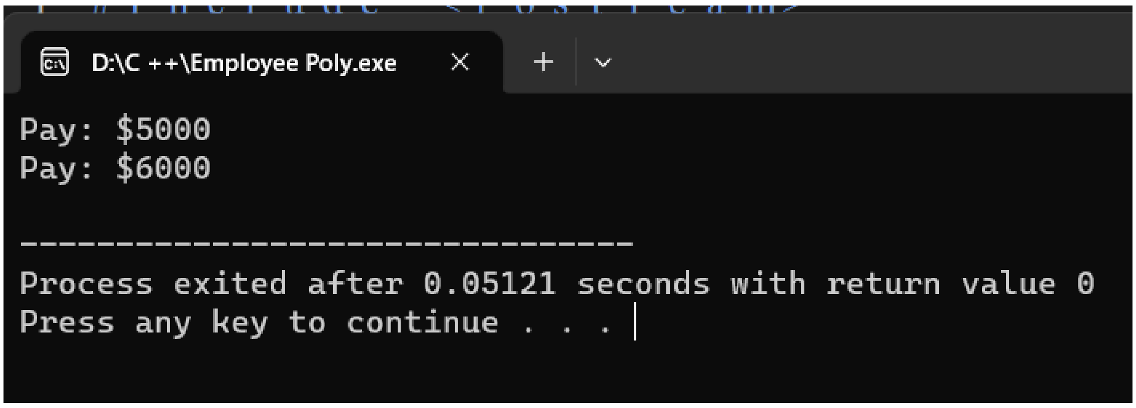
```
D:\C ++\Animal Poly.exe

Meow!
Woof!

--------------------------------
Process exited after 0.05478 seconds with return value 0
Press any key to continue . . .
```

EMPLOYEE

```
#include <iostream>

class Employee {
public:
    virtual double calculatePay() {
        return 0.0;
    }
};
```

```
class Manager : public Employee {
public:
   double calculatePay() override {
      return 5000.0;
   }
};

class Engineer : public Employee {
public:
   double calculatePay() override {
      return 6000.0;
   }
};

int main() {
   Employee* employees[] = {new Manager, new
Engineer};

   for (Employee* emp : employees) {
      std::cout << "Pay: $" << emp->calculatePay() <<
std::endl;
   }
```

```
    return 0;
}
```

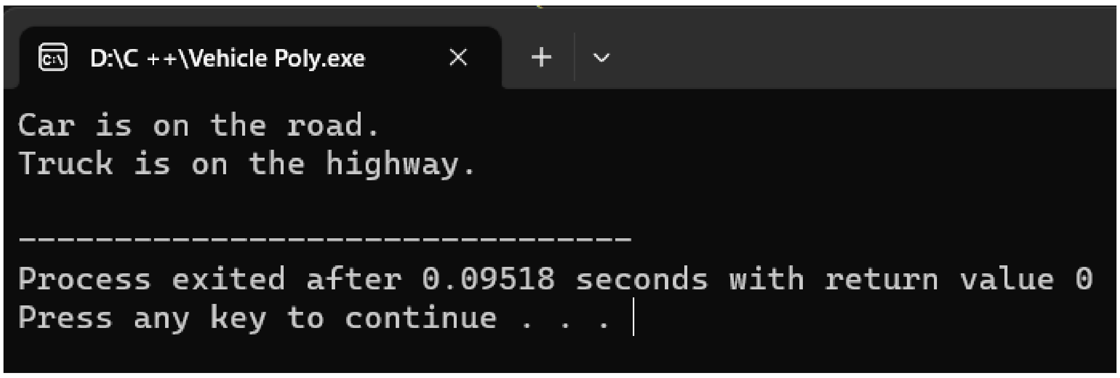


```
Pay: $5000
Pay: $6000

--------------------------------
Process exited after 0.05121 seconds with return value 0
Press any key to continue . . .
```

VEHICLE

```
#include <iostream>


class Vehicle {
public:
    virtual void drive() {
        std::cout << "Vehicle is being driven." << 
std::endl;
    }
};
```

```
class Car : public Vehicle {
public:
    void drive() override {
        std::cout << "Car is on the road." << std::endl;
    }
};

class Truck : public Vehicle {
public:
    void drive() override {
        std::cout << "Truck is on the highway." <<
std::endl;
    }
};

int main() {
    Vehicle* vehicles[] = {new Car, new Truck};

    for (Vehicle* vehicle : vehicles) {
        vehicle->drive();
    }
    return 0;
}
```



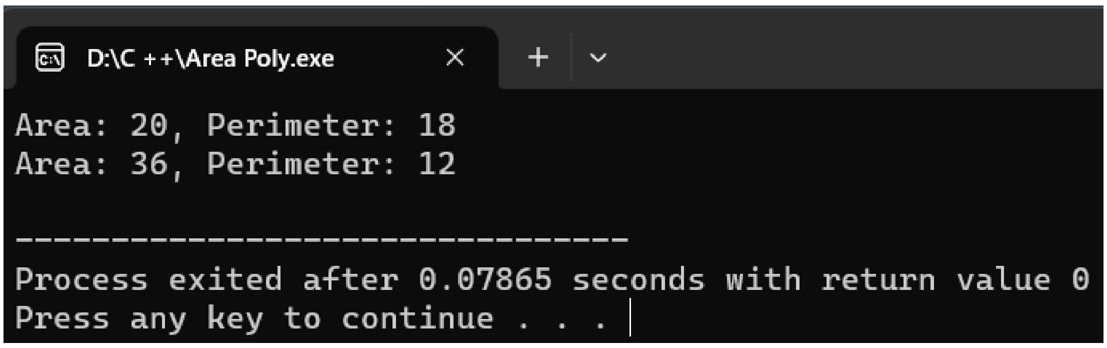
```
Car is on the road.
Truck is on the highway.

--------------------------------
Process exited after 0.09518 seconds with return value 0
Press any key to continue . . .
```

AREA

```
#include <iostream>


class Shape {
public:
    virtual double area() {
        return 0.0;
    }


    virtual double perimeter() {
        return 0.0;
    }
```

```
};

class Rectangle : public Shape {
private:
    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 {
private:
    double side1, side2, side3;

public:
    Triangle(double s1, double s2, double s3) : side1(s1),
side2(s2), side3(s3) {}

    double area() override {
        // Implement the area calculation for a triangle
(e.g., using Heron's formula)
        double s = (side1 + side2 + side3) / 2;
        return (s * (s - side1) * (s - side2) * (s - side3));
    }

    double perimeter() override {
        return side1 + side2 + side3;
    }
};

int main() {
    Shape* shapes[] = {new Rectangle(5, 4), new
Triangle(3, 4, 5)};
```

```
    for (Shape* shape : shapes) {
        std::cout << "Area: " << shape->area() << ", 
Perimeter: " << shape->perimeter() << std::endl;
    }

    return 0;
}
```

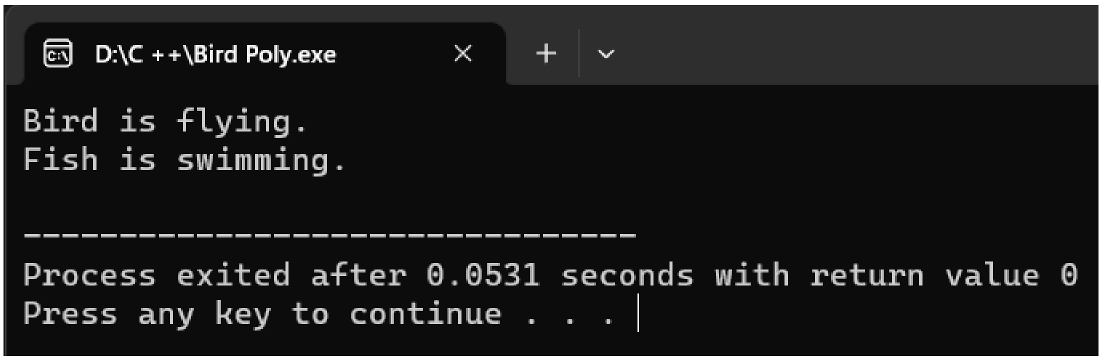


```
Area: 20, Perimeter: 18
Area: 36, Perimeter: 12

--------------------------------
Process exited after 0.07865 seconds with return value 0
Press any key to continue . . .
```

BIRD

```
#include <iostream>

class Animal {
public:
    virtual void move() {
        std::cout << "Animal is moving." << std::endl;
    }
```

```
};

class Bird : public Animal {
public:
    void move() override {
        std::cout << "Bird is flying." << std::endl;
    }
};

class Fish : public Animal {
public:
    void move() override {
        std::cout << "Fish is swimming." << std::endl;
    }
};

int main() {
    Animal* animals[] = {new Bird, new Fish};

    for (Animal* animal : animals) {
        animal->move();
    }

    return 0;
}
```

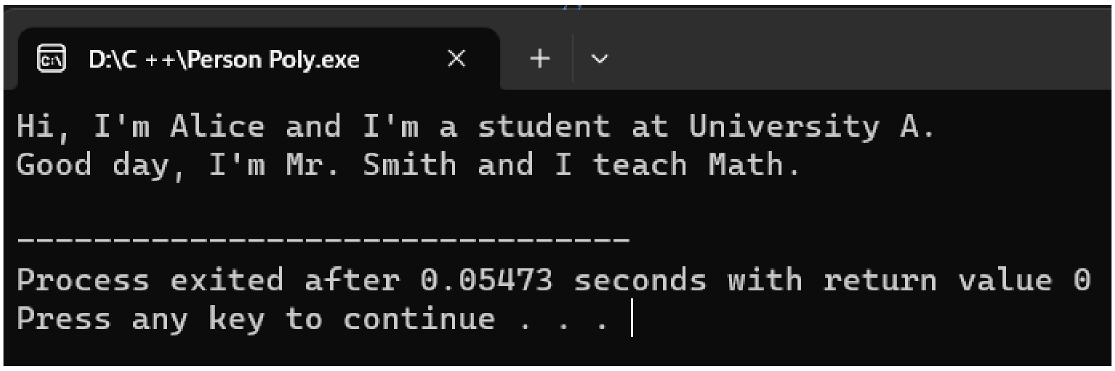


```
Bird is flying.
Fish is swimming.

--------------------------------
Process exited after 0.0531 seconds with return value 0
Press any key to continue . . .
```

PERSON

```
#include <iostream>
#include <string>

class Person {
public:
    Person(const std::string& name) : name(name) {}

    virtual void greet() {
        std::cout << "Hello, I'm " << name << "." <<
std::endl;
```

```
    }

protected:
    std::string name;
};

class Student : public Person {
public:
    Student(const std::string& name, const std::string&
school) : Person(name), school(school) {}

    void greet() override {
        std::cout << "Hi, I'm " << name << " and I'm a
student at " << school << "." << std::endl;
    }

private:
    std::string school;
};

class Teacher : public Person {
public:
    Teacher(const std::string& name, const std::string&
subject) : Person(name), subject(subject) {}

    void greet() override {
        std::cout << "Good day, I'm " << name << " and I
teach " << subject << "." << std::endl;
    }

private:
    std::string subject;
};

int main() {
    Person* people[] = {new Student("Alice",
"University A"), new Teacher("Mr. Smith", "Math")};

    for (Person* person : people) {
        person->greet();
    }

    return 0;
}
```



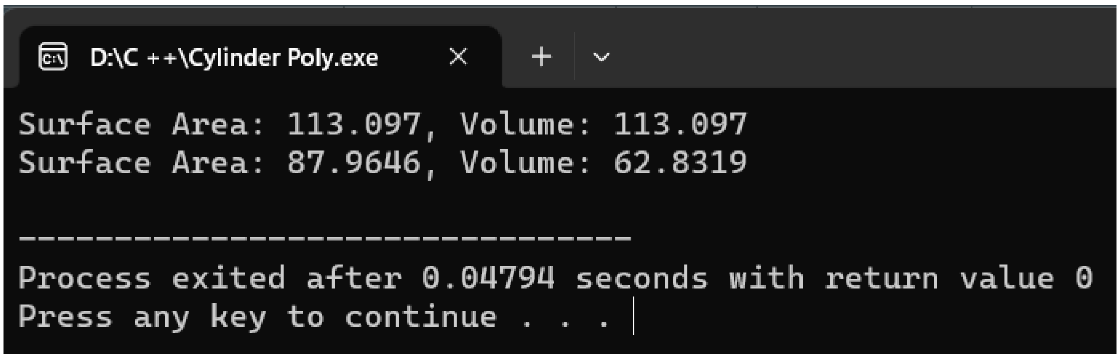
```
Hi, I'm Alice and I'm a student at University A.
Good day, I'm Mr. Smith and I teach Math.

--------------------------------
Process exited after 0.05473 seconds with return value 0
Press any key to continue . . .
```

CYLINDER

```
#include <iostream>

class Shape {
public:
    virtual double area() {
        return 0.0;
    }

    virtual double volume() {
        return 0.0;
    }
};

class Sphere : public Shape {
```

```
private:
    double radius;

public:
    Sphere(double r) : radius(r) {}

    double area() override {
        return 4 * 3.14159265359 * radius * radius;
    }

    double volume() override {
        return (4.0 / 3.0) * 3.14159265359 * radius *
radius * radius;
    }
};

class Cylinder : public Shape {
private:
    double radius;
    double height;
```

```
public:
    Cylinder(double r, double h) : radius(r), height(h) {}

    double area() override {
        return 2 * 3.14159265359 * radius * (radius +
height);
    }

    double volume() override {
        return 3.14159265359 * radius * radius * height;
    }
};

int main() {
    Shape* shapes[] = {new Sphere(3), new Cylinder(2,
5)};

    for (Shape* shape : shapes) {
        std::cout << "Surface Area: " << shape->area() <<
", Volume: " << shape->volume() << std::endl;
    }
    return 0;
}
```


D:\C ++\Cylinder Poly.exe

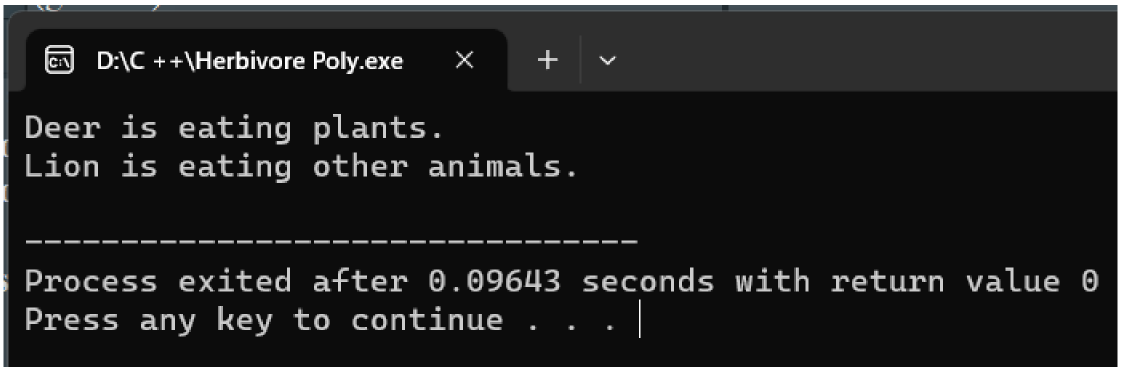
```
Surface Area: 113.097, Volume: 113.097
Surface Area: 87.9646, Volume: 62.8319

--------------------------------
Process exited after 0.04794 seconds with return value 0
Press any key to continue . . .
```

HERBIVORE

```
#include <iostream>
#include <string>

class Animal {
public:
    Animal(const std::string& name) : name(name) {}

    virtual void eat() {
        std::cout << name << " is eating." << std::endl;
    }

protected:
    std::string name;
};
```

```
class Herbivore : public Animal {
public:
   Herbivore(const std::string& name) : Animal(name)
{}

   void eat() override {
       std::cout << name << " is eating plants." <<
std::endl;
   }
};

class Carnivore : public Animal {
public:
   Carnivore(const std::string& name) : Animal(name)
{}

   void eat() override {
       std::cout << name << " is eating other animals."
<< std::endl;
   }
};
```

```
int main() {
   Animal* animals[] = {new Herbivore("Deer"), new
Carnivore("Lion")};

   for (Animal* animal : animals) {
      animal->eat();
   }

   return 0;
}
```



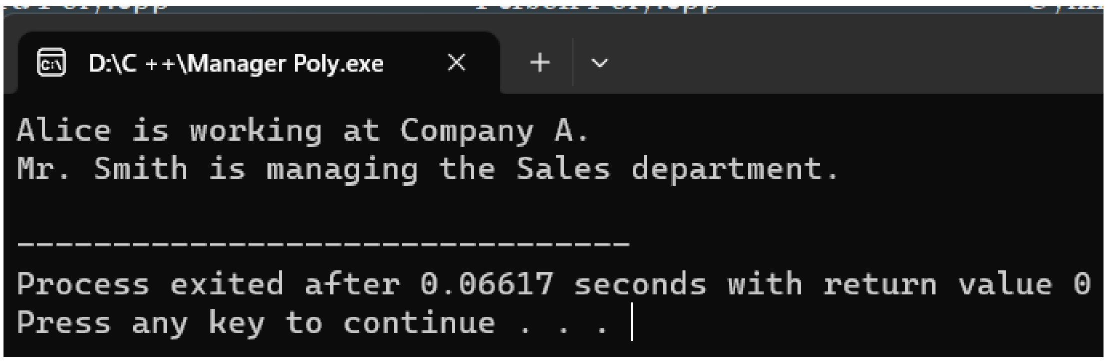
```
Deer is eating plants.
Lion is eating other animals.

--------------------------------
Process exited after 0.09643 seconds with return value 0
Press any key to continue . . .
```

MANAGER

```
#include <iostream>
#include <string>
```

```
class Person {
public:
    Person(const std::string& name) : name(name) {}

    virtual void work() {
        std::cout << name << " is working." << std::endl;
    }

protected:
    std::string name;
};

class Employee : public Person {
public:
    Employee(const std::string& name, const
std::string& company) : Person(name),
company(company) {}

    void work() override {
        std::cout << name << " is working at " <<
company << "." << std::endl;
```

```
    }

private:
    std::string company;
};

class Manager : public Person {
public:
    Manager(const std::string& name, const std::string&
department) : Person(name), department(department) {}

    void work() override {
        std::cout << name << " is managing the " <<
department << " department." << std::endl;
    }

private:
    std::string department;
};

int main() {
    Person* people[] = {new Employee("Alice",
"Company A"), new Manager("Mr. Smith", "Sales")};

    for (Person* person : people) {
        person->work();
    }

    return 0;
}
```



```
Alice is working at Company A.
Mr. Smith is managing the Sales department.

--------------------------------
Process exited after 0.06617 seconds with return value 0
Press any key to continue . . .
```