Create a program that simulates a zoo with various animals. Each animal should have a common method called "speak" that makes a sound specific to the animal type.

Objective:

Utilize runtime polymorphism to achieve the following:

Define an abstract base class named Animal with a method speak that doesn't have an implementation (declare it abstract).

Create subclasses for different animals like Lion, Elephant, etc., inheriting from Animal.

Override the speak method in each subclass to define the specific sound of the animal (e.g., Lion roars, Elephant trumpets).

In the main program, create an array of Animal references. Populate this array with objects of different animal subclasses.

Loop through the animal array and call the speak method on each reference. Since the references are of the base class type, runtime polymorphism will determine the actual subclass and invoke the appropriate overridden speak method.

This exercise will demonstrate runtime polymorphism by:

Highlighting the separation between declared type (reference variable type) and actual type (object type).

Showing how the method call is resolved at runtime based on the actual object.

Code:

#include <iostream>

using namespace std;

class Animal {

public:

virtual void speak() = 0;

virtual ~Animal() {}

};

class Lion : public Animal {

public:

void speak() override {

cout << "Roar!" << endl;

}

};

class Elephant : public Animal {

public:

void speak() override {

cout << "Trumpet!" << endl;

}

};

class Monkey : public Animal {

public:

void speak() override {

cout << "Ooh ooh ah ah!" << endl;

}

};

int main() {

Animal\* animals[3];

animals[0] = new Lion();

animals[1] = new Elephant();

animals[2] = new Monkey();

for (int i = 0; i < 3; i++) {

animals[i]->speak();

}

return 0;

}

Output:

