1- Create a base class Vehicle with a method start to display "Vehicle started." Derive a class Car from Vehicle with a method drive to display "Car is being driven."

قم بإنشاء Vehicle من class الأساسية باستخدام طريقة البدء لعرض "بدأت المركبة"

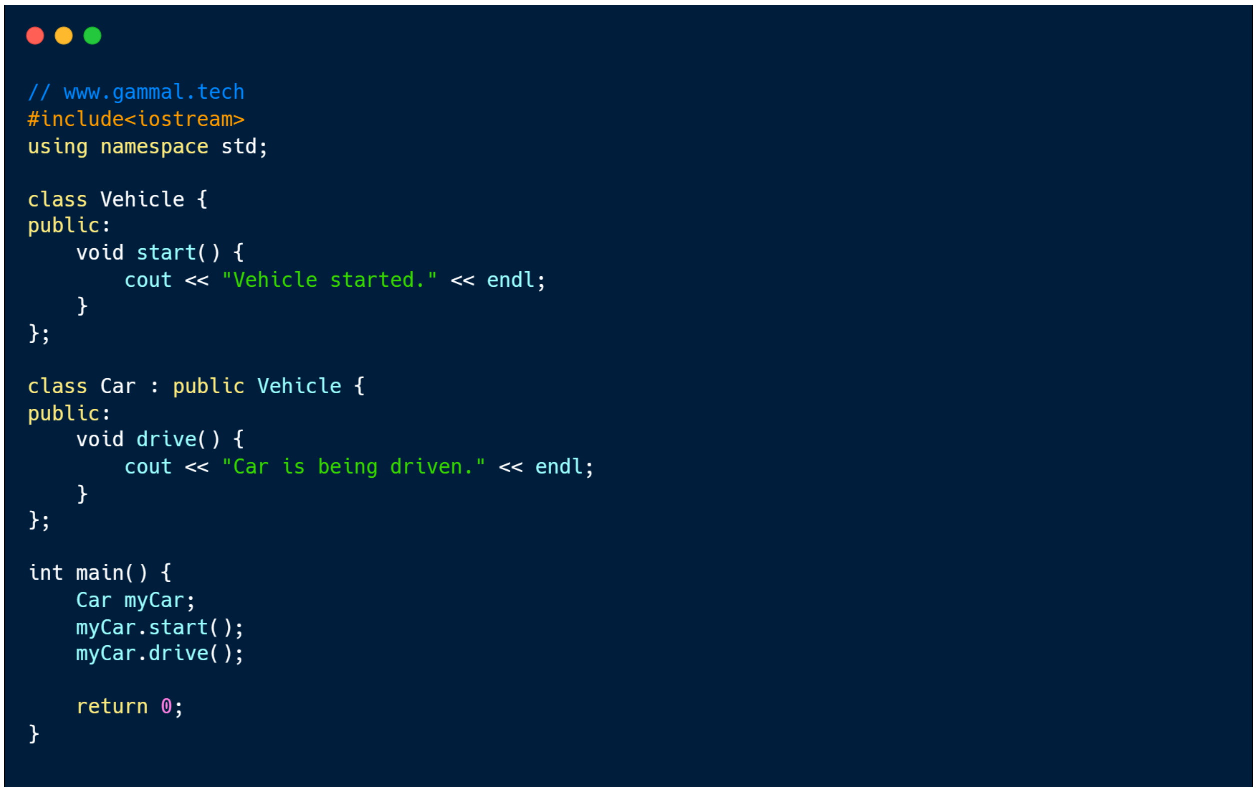
Derive a class Car from Vehicle with a method drive to display .

"يتم قيادة السيارة".

Output

```
Vehicle started.
Car is being driven.
```

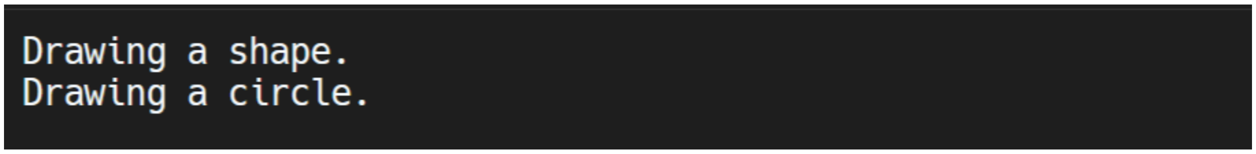
Solution



```cpp
// www.gammal.tech
#include<iostream>
using namespace std;

class Vehicle {
public:
    void start() {
        cout << "Vehicle started." << endl;
    }
};

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

int main() {
    Car myCar;
    myCar.start();
    myCar.drive();

    return 0;
}
```

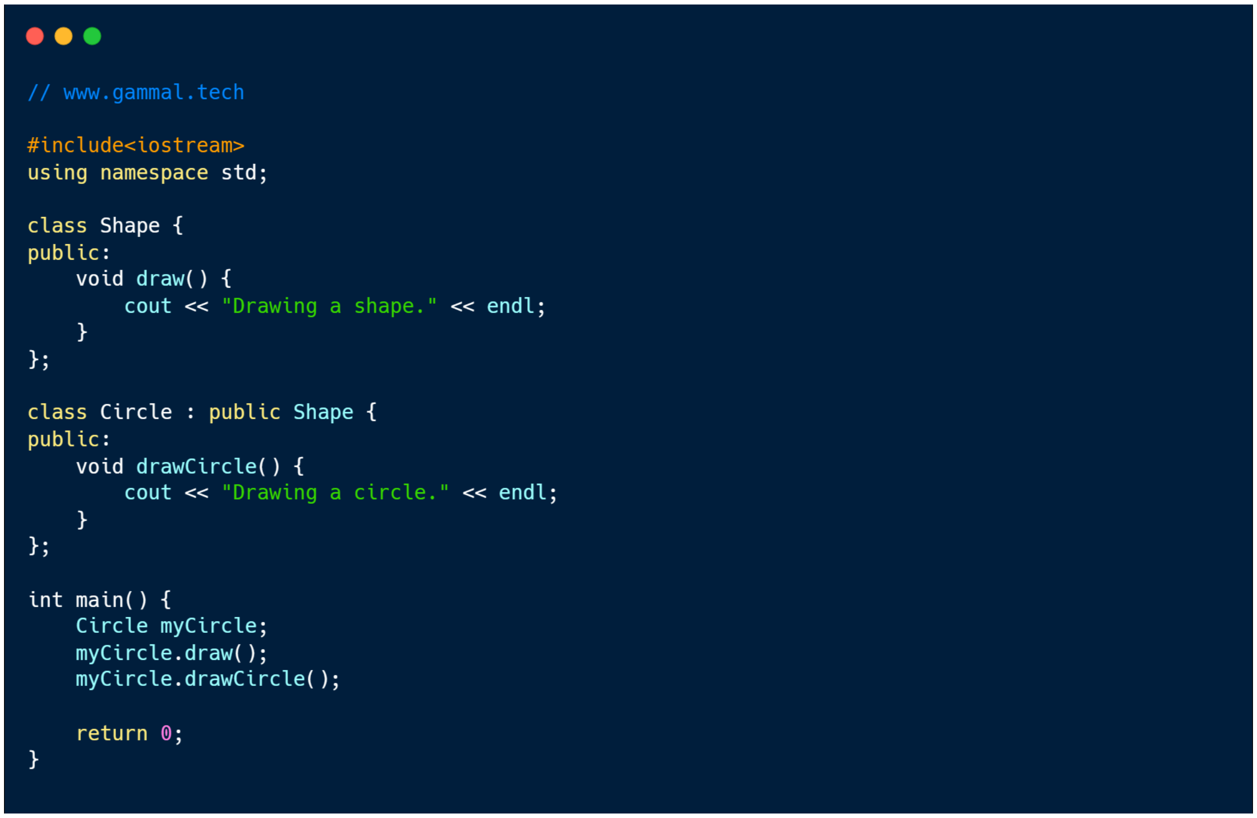
2- Create a base class Shape with a method draw to display "Drawing a shape." Derive a class Circle from Shape with a method drawCircle to display "Drawing a circle."

Output



```
Drawing a shape.
Drawing a circle.
```

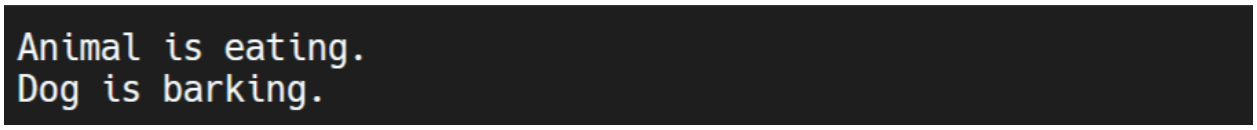
Solution



```cpp
// www.gammal.tech

#include<iostream>
using namespace std;

class Shape {
public:
    void draw() {
        cout << "Drawing a shape." << endl;
    }
};

class Circle : public Shape {
public:
    void drawCircle() {
        cout << "Drawing a circle." << endl;
    }
};

int main() {
    Circle myCircle;
    myCircle.draw();
    myCircle.drawCircle();

    return 0;
}
```

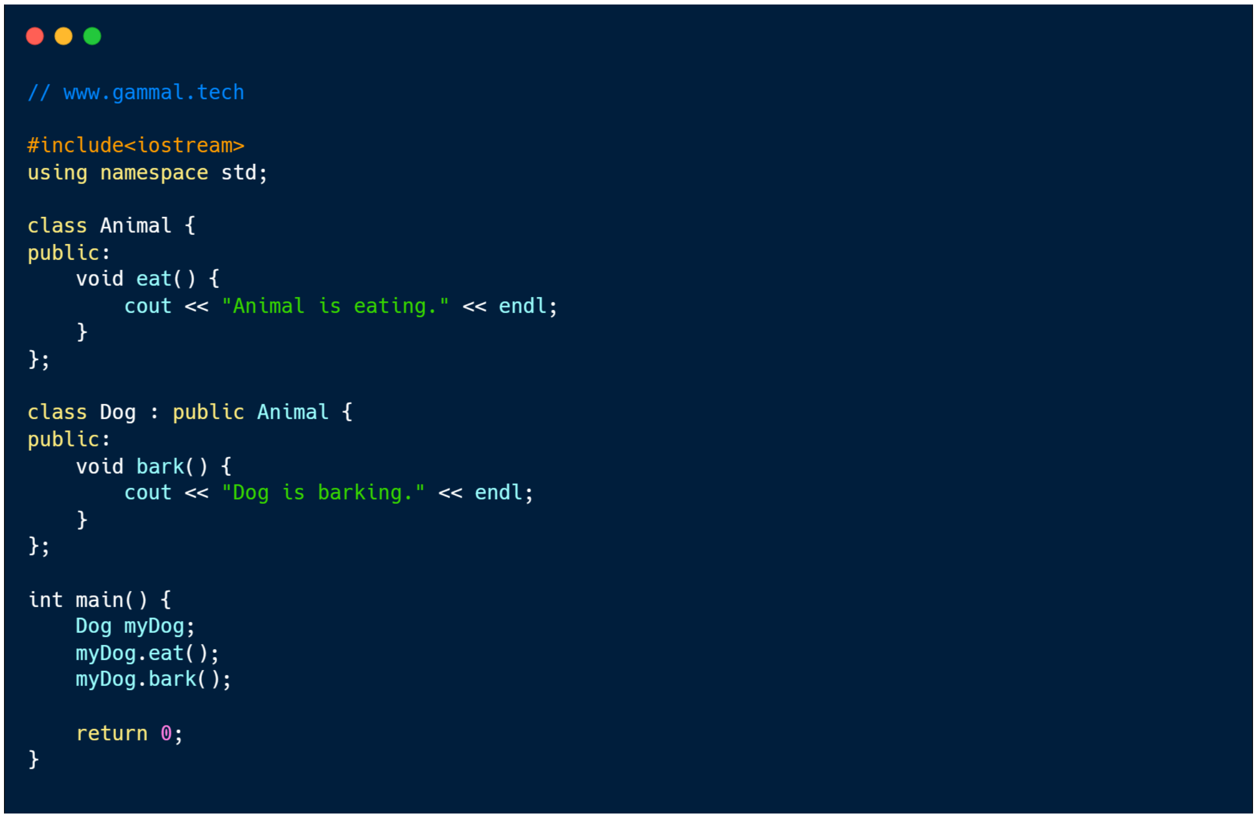
3- Create a base class Animal with a method eat to display "Animal is eating." Derive a class Dog from Animal with a method bark to display "Dog is barking."

Output



```
Animal is eating.
Dog is barking.
```

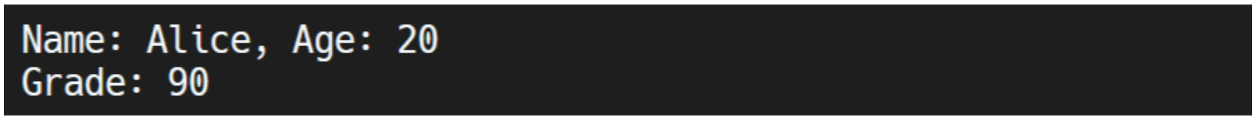
Solution



```cpp
// www.gammal.tech

#include<iostream>
using namespace std;

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

class Dog : public Animal {
public:
    void bark() {
        cout << "Dog is barking." << endl;
    }
};

int main() {
    Dog myDog;
    myDog.eat();
    myDog.bark();

    return 0;
}
```

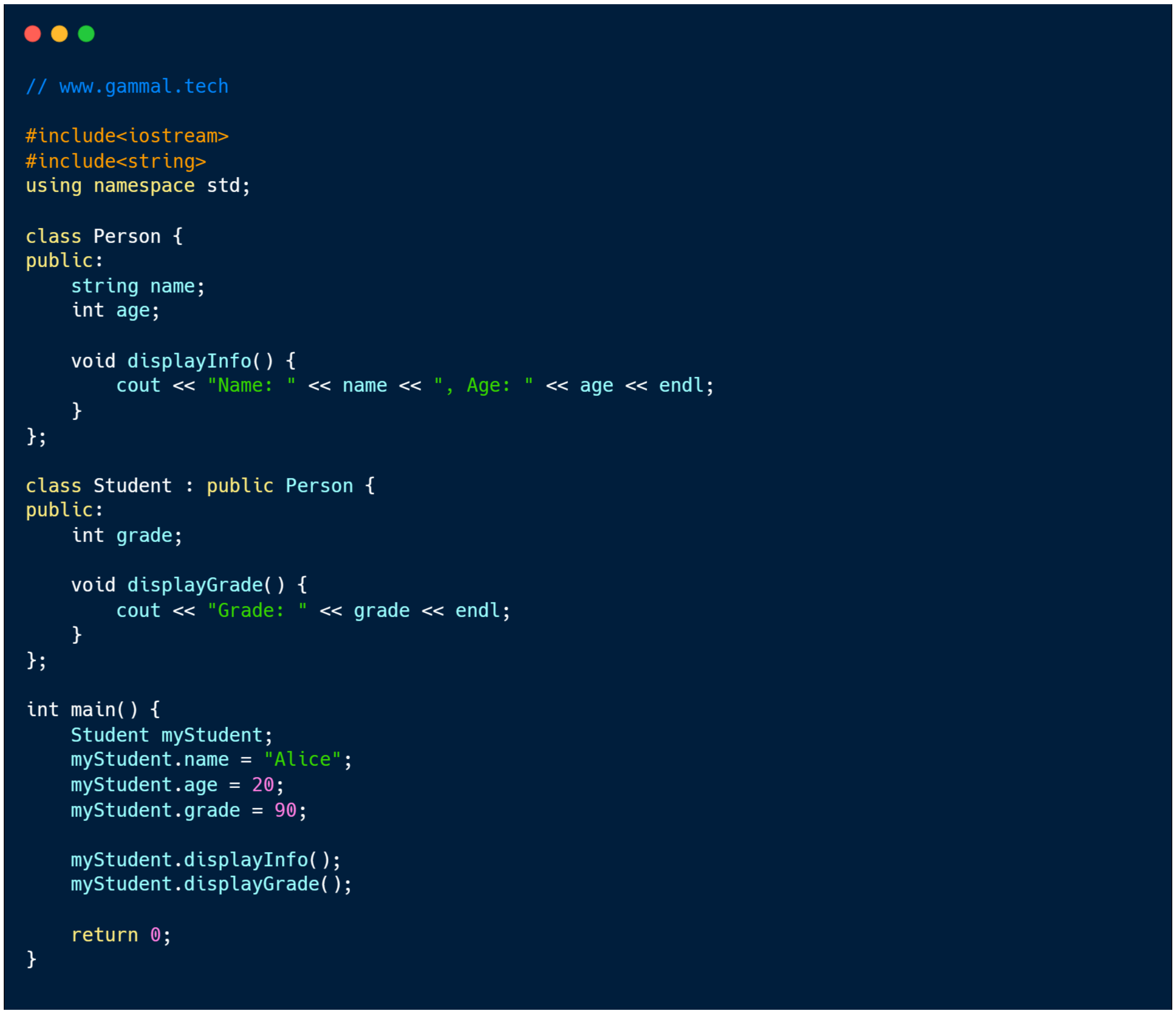
4- Create a base class Person with attributes name and age and a method displayInfo to display the person's information. Derive a class Student from Person with an additional attribute grade and a method displayGrade to display the student's grade.

Output



```
Name: Alice, Age: 20
Grade: 90
```

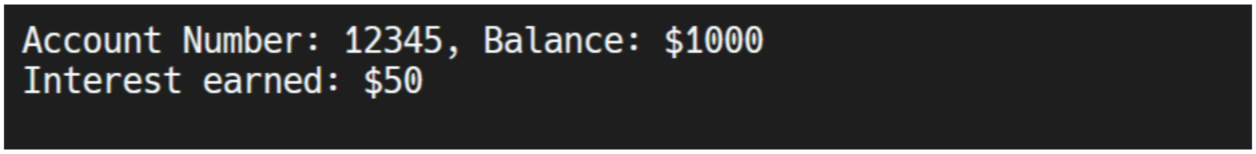
Solution



```
// www.gammal.tech

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

class Person {
public:
    string name;
    int age;

    void displayInfo() {
        cout << "Name: " << name << ", Age: " << age << endl;
    }
};

class Student : public Person {
public:
    int grade;

    void displayGrade() {
        cout << "Grade: " << grade << endl;
    }
};

int main() {
    Student myStudent;
    myStudent.name = "Alice";
    myStudent.age = 20;
    myStudent.grade = 90;

    myStudent.displayInfo();
    myStudent.displayGrade();

    return 0;
}
```

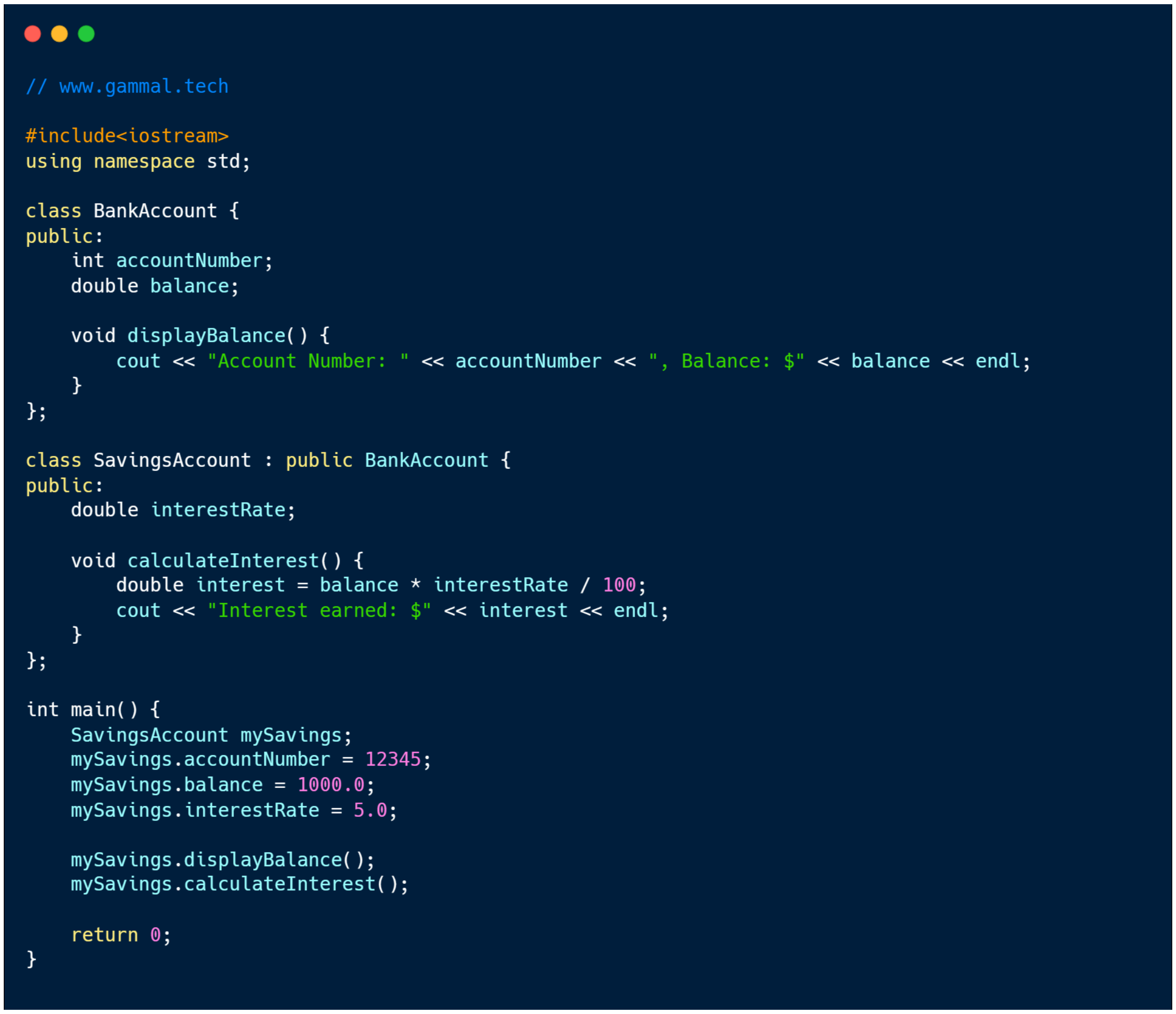
5- Create a base class BankAccount with attributes accountNumber and balance and a method displayBalance to display the account balance. Derive a class SavingsAccount from BankAccount with an additional attribute interestRate and a method calculateInterest to display the interest earned.

Output



```
Account Number: 12345, Balance: $1000
Interest earned: $50
```

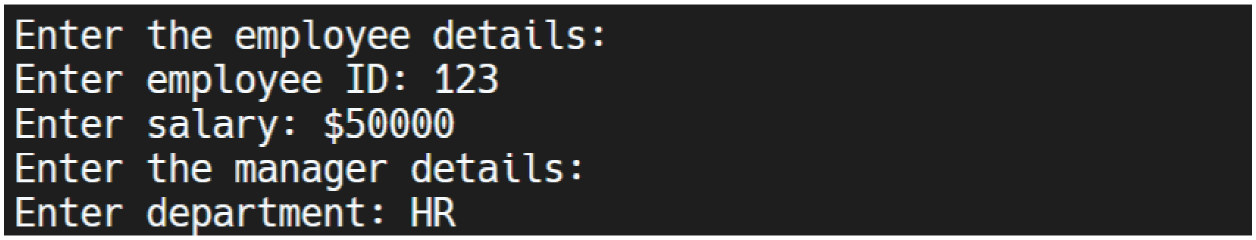
Solution



```cpp
// www.gammal.tech

#include<iostream>
using namespace std;

class BankAccount {
public:
    int accountNumber;
    double balance;

    void displayBalance() {
        cout << "Account Number: " << accountNumber << ", Balance: $" << balance << endl;
    }
};

class SavingsAccount : public BankAccount {
public:
    double interestRate;

    void calculateInterest() {
        double interest = balance * interestRate / 100;
        cout << "Interest earned: $" << interest << endl;
    }
};

int main() {
    SavingsAccount mySavings;
    mySavings.accountNumber = 12345;
    mySavings.balance = 1000.0;
    mySavings.interestRate = 5.0;

    mySavings.displayBalance();
    mySavings.calculateInterest();

    return 0;
}
```

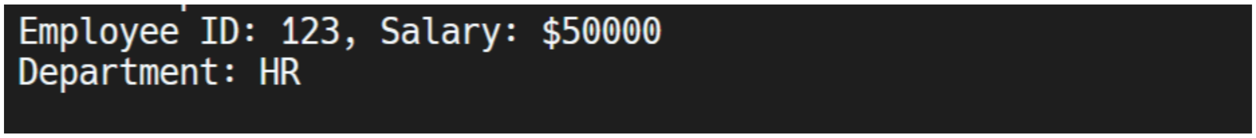
6- Create a program using inheritance. Define a base class Employee with attributes employeeId and salary and a method displayInfo to display the employee's information. Derive a class Manager from Employee with an additional attribute department and a method displayDepartment to display the manager's department. Get details from the user for both the employee and the manager.

Input



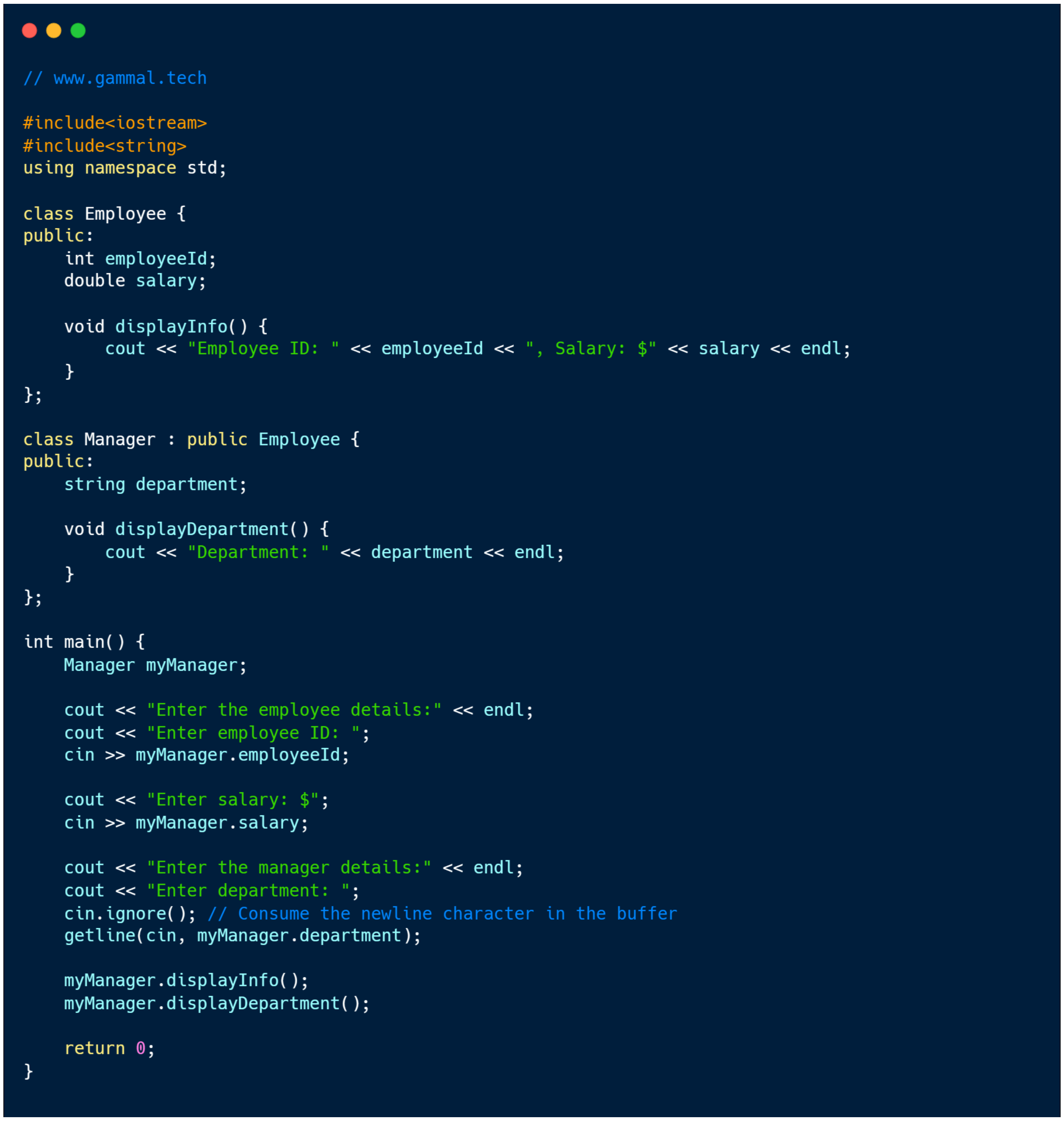
```
Enter the employee details:
Enter employee ID: 123
Enter salary: $50000
Enter the manager details:
Enter department: HR
```

Output



```
Employee ID: 123, Salary: $50000
Department: HR
```

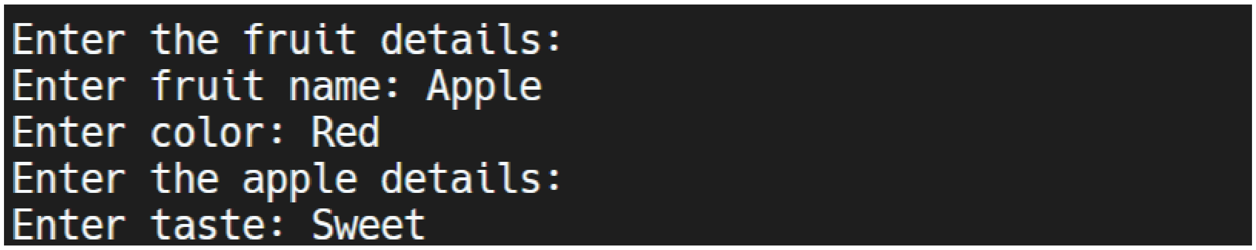
Solution



```cpp
// www.gammal.tech

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

class Employee {
public:
    int employeeId;
    double salary;

    void displayInfo() {
        cout << "Employee ID: " << employeeId << ", Salary: $" << salary << endl;
    }
};

class Manager : public Employee {
public:
    string department;

    void displayDepartment() {
        cout << "Department: " << department << endl;
    }
};

int main() {
    Manager myManager;

    cout << "Enter the employee details:" << endl;
    cout << "Enter employee ID: ";
    cin >> myManager.employeeId;

    cout << "Enter salary: $";
    cin >> myManager.salary;

    cout << "Enter the manager details:" << endl;
    cout << "Enter department: ";
    cin.ignore(); // Consume the newline character in the buffer
    getline(cin, myManager.department);

    myManager.displayInfo();
    myManager.displayDepartment();

    return 0;
}
```

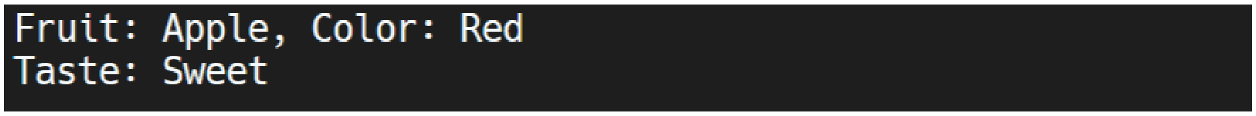
7- Create a program demonstrating inheritance. Define a base class Fruit with attributes name and color and a method displayInfo to display the fruit's information. Derive a class Apple from Fruit with an additional attribute taste and a method displayTaste to display the taste of the apple. Get details from the user for both the fruit and the apple.

Input



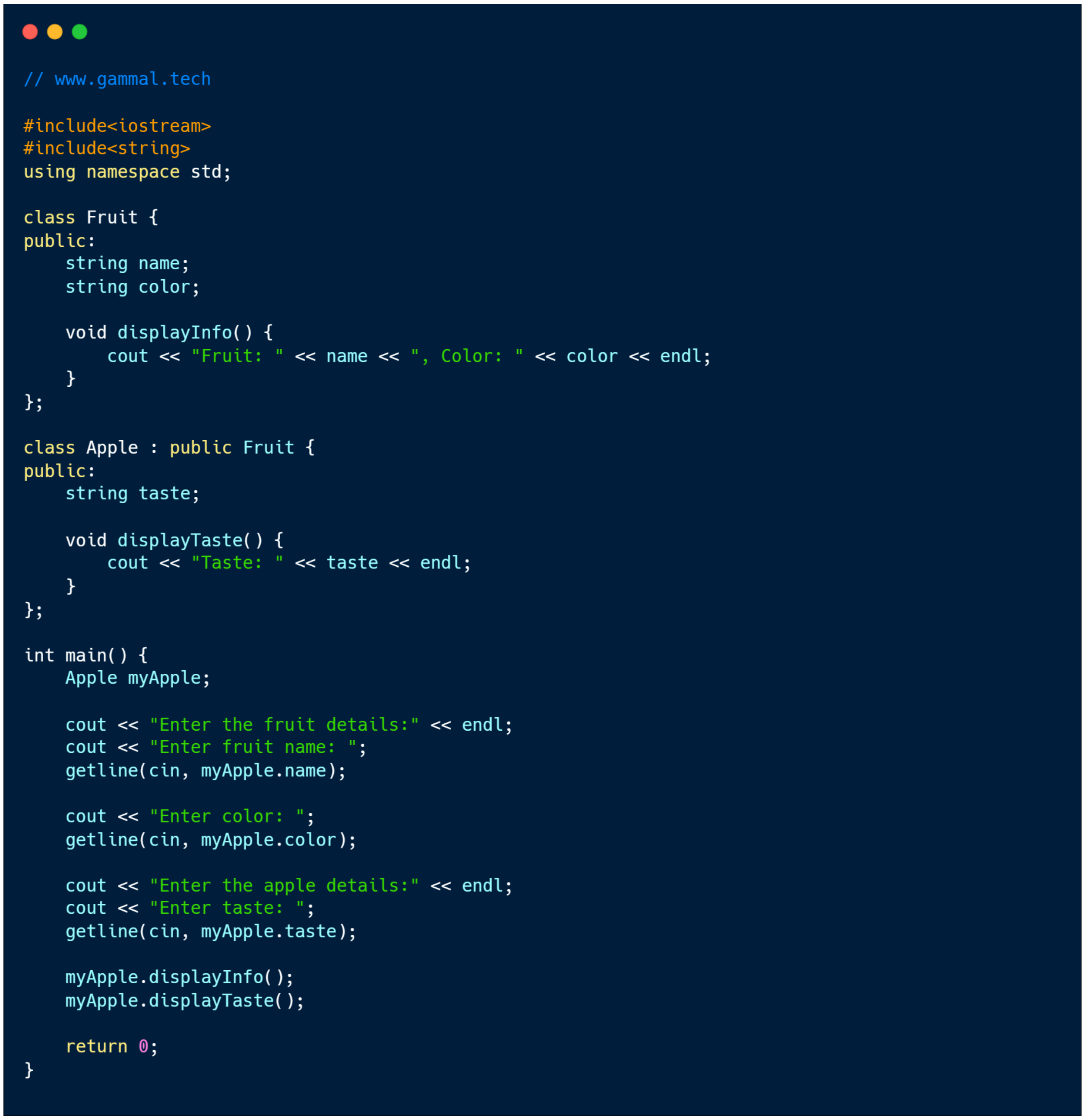
```
Enter the fruit details:
Enter fruit name: Apple
Enter color: Red
Enter the apple details:
Enter taste: Sweet
```

Output



```
Fruit: Apple, Color: Red
Taste: Sweet
```

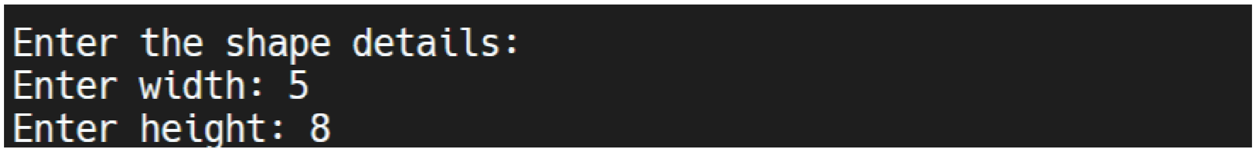
Solution



```cpp
// www.gammal.tech

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

class Fruit {
public:
    string name;
    string color;

    void displayInfo() {
        cout << "Fruit: " << name << ", Color: " << color << endl;
    }
};

class Apple : public Fruit {
public:
    string taste;

    void displayTaste() {
        cout << "Taste: " << taste << endl;
    }
};

int main() {
    Apple myApple;

    cout << "Enter the fruit details:" << endl;
    cout << "Enter fruit name: ";
    getline(cin, myApple.name);

    cout << "Enter color: ";
    getline(cin, myApple.color);

    cout << "Enter the apple details:" << endl;
    cout << "Enter taste: ";
    getline(cin, myApple.taste);

    myApple.displayInfo();
    myApple.displayTaste();

    return 0;
}
```

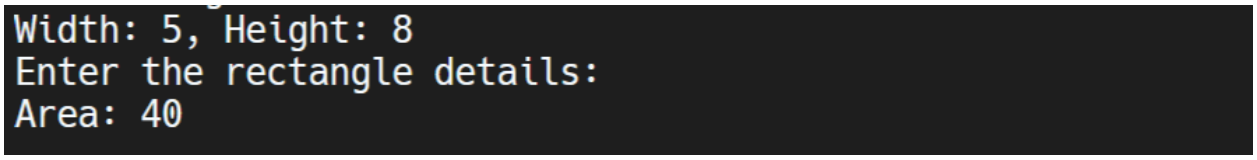
8- Create a program using inheritance. Define a base class Shape with attributes width and height and a method displayDimensions to display the shape's dimensions. Derive a class Rectangle from Shape with an additional attribute area and a method calculateArea to calculate and display the area of the rectangle. Get details from the user for both the shape and the rectangle.

Input



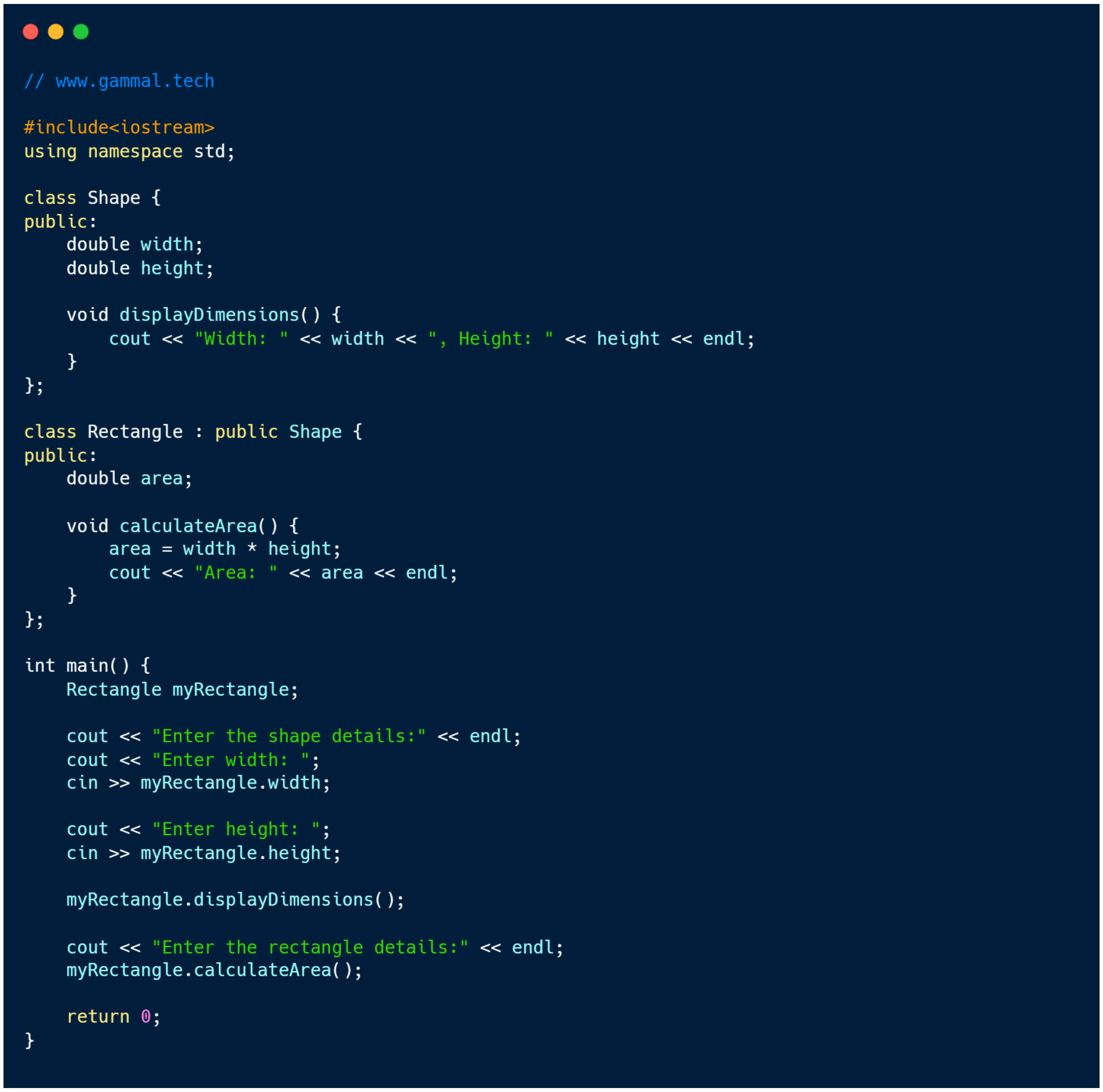
```
Enter the shape details:
Enter width: 5
Enter height: 8
```

Output



```
Width: 5, Height: 8
Enter the rectangle details:
Area: 40
```

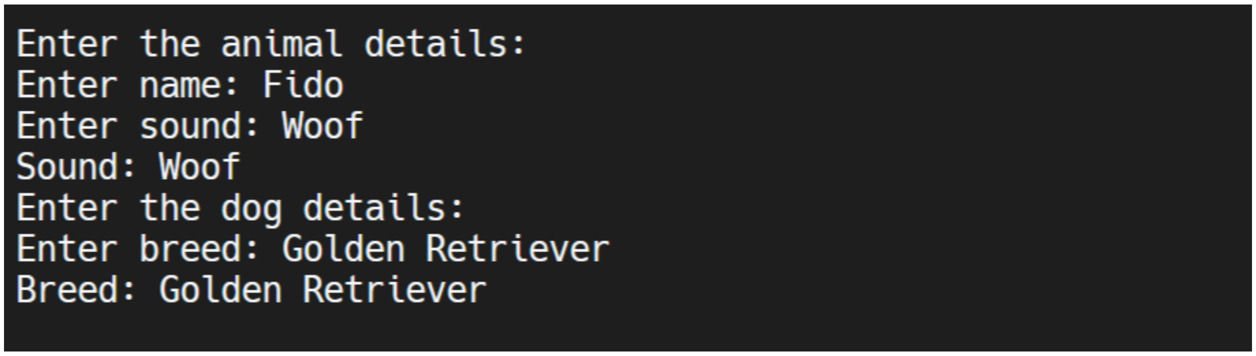
Solution



```cpp
// www.gammal.tech

#include<iostream>
using namespace std;

class Shape {
public:
    double width;
    double height;

    void displayDimensions() {
        cout << "Width: " << width << ", Height: " << height << endl;
    }
};

class Rectangle : public Shape {
public:
    double area;

    void calculateArea() {
        area = width * height;
        cout << "Area: " << area << endl;
    }
};

int main() {
    Rectangle myRectangle;

    cout << "Enter the shape details:" << endl;
    cout << "Enter width: ";
    cin >> myRectangle.width;

    cout << "Enter height: ";
    cin >> myRectangle.height;

    myRectangle.displayDimensions();

    cout << "Enter the rectangle details:" << endl;
    myRectangle.calculateArea();

    return 0;
}
```

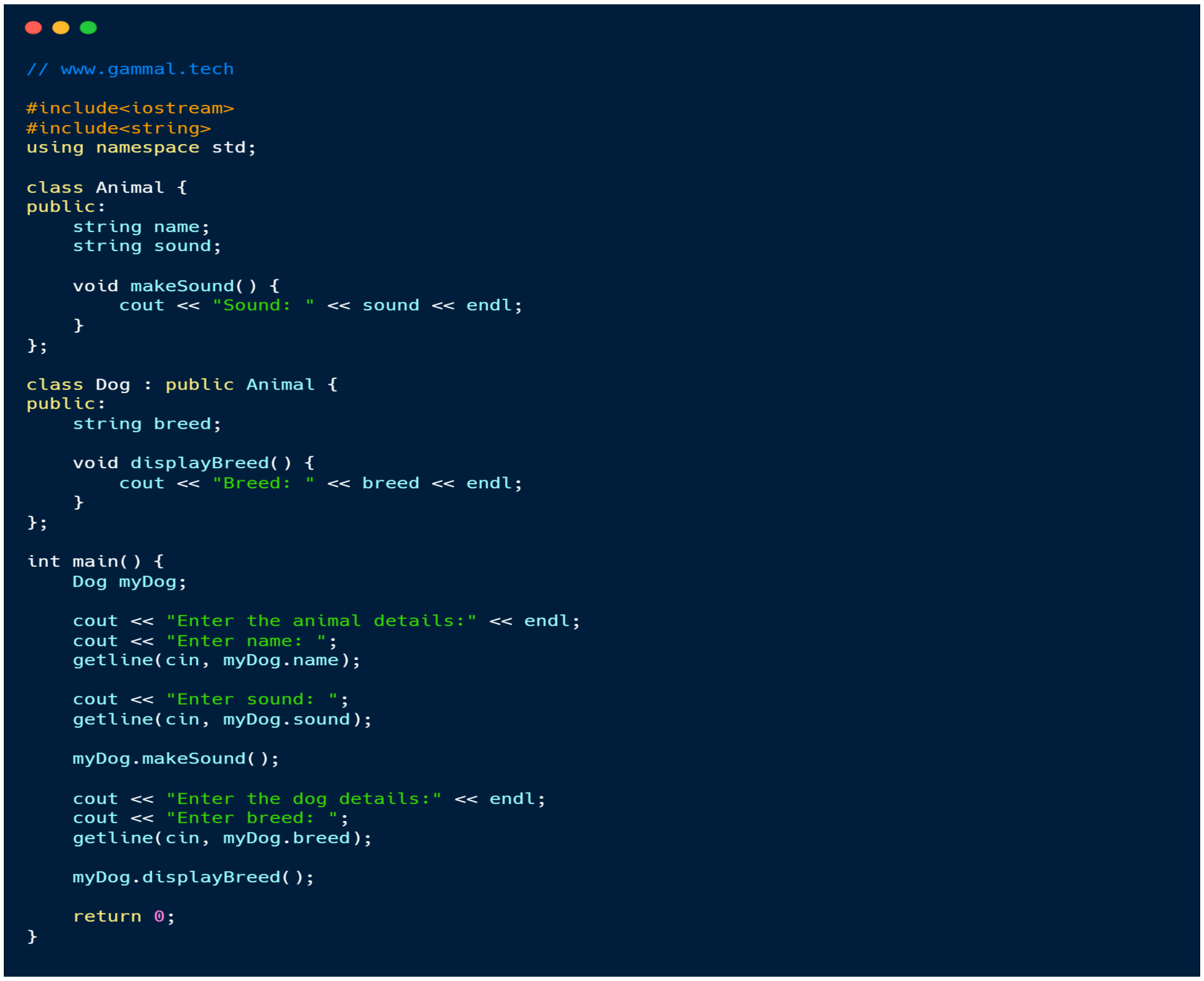
9- create a program demonstrating inheritance. Define a base class Animal with attributes name and sound and a method makeSound to display the animal's sound. Derive a class Dog from Animal with an additional attribute breed and a method displayBreed to display the dog's breed. Get details from the user for both the animal and the dog.

Input & Output



```
Enter the animal details:
Enter name: Fido
Enter sound: Woof
Sound: Woof
Enter the dog details:
Enter breed: Golden Retriever
Breed: Golden Retriever
```

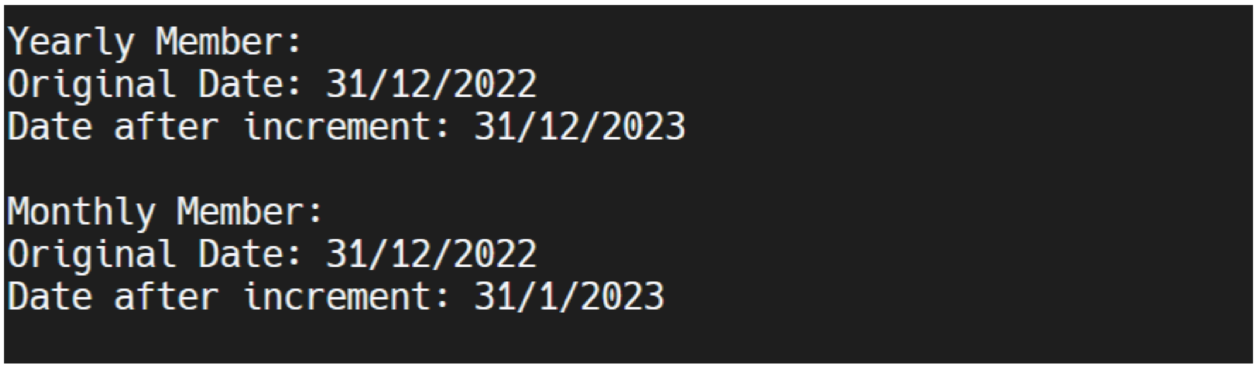
Solution



```cpp
// www.gammal.tech

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

class Animal {
public:
    string name;
    string sound;

    void makeSound() {
        cout << "Sound: " << sound << endl;
    }
};

class Dog : public Animal {
public:
    string breed;

    void displayBreed() {
        cout << "Breed: " << breed << endl;
    }
};

int main() {
    Dog myDog;

    cout << "Enter the animal details:" << endl;
    cout << "Enter name: ";
    getline(cin, myDog.name);

    cout << "Enter sound: ";
    getline(cin, myDog.sound);

    myDog.makeSound();

    cout << "Enter the dog details:" << endl;
    cout << "Enter breed: ";
    getline(cin, myDog.breed);

    myDog.displayBreed();

    return 0;
}
```

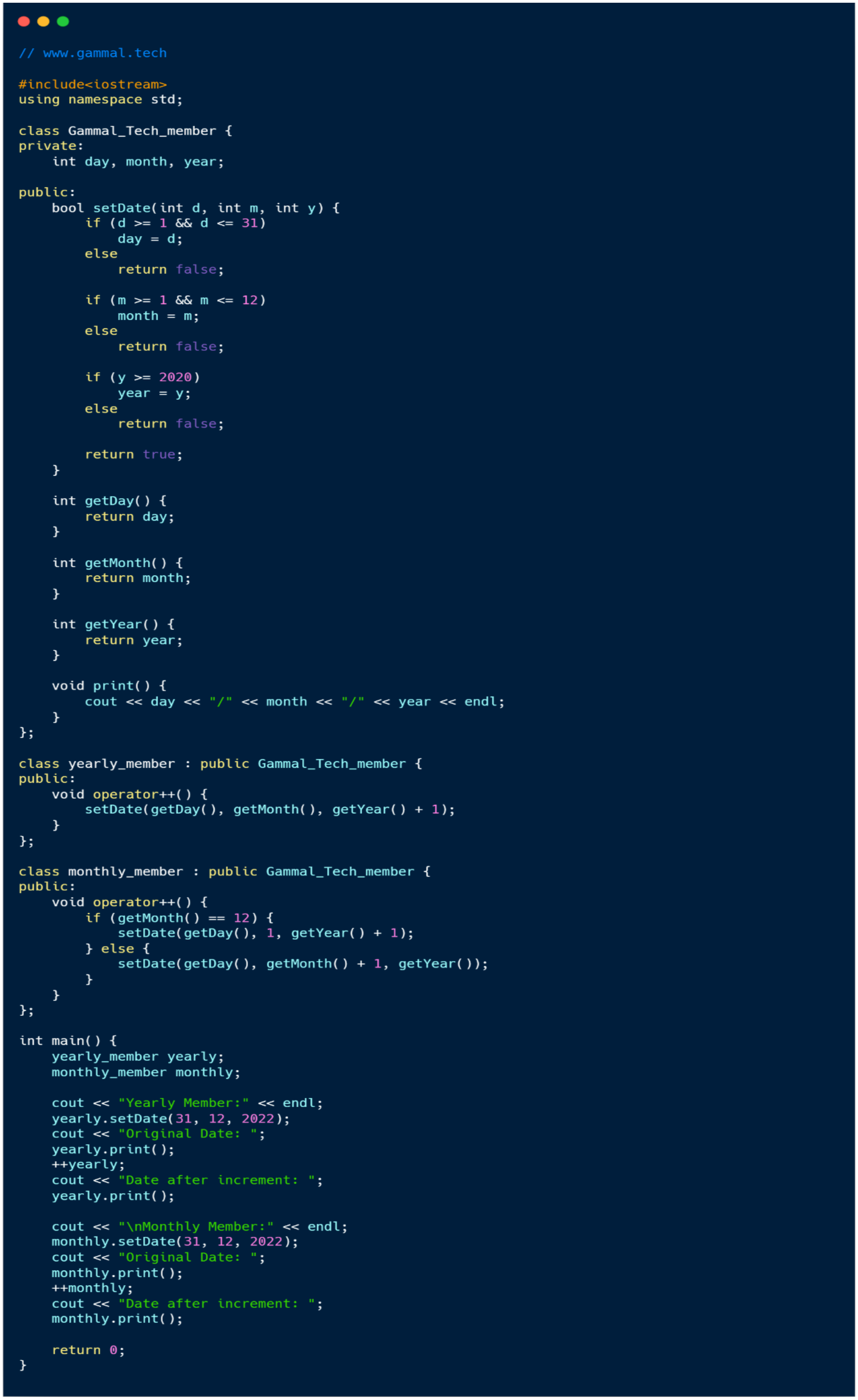
10- Create a program defines a class Gammal_Tech_member representing a generic member with the ability to set and print a date. It also includes two derived classes, yearly_member and monthly_member, representing members with yearly and monthly subscriptions. The derived classes inherit from the base class Gammal_Tech_member and override the operator++ to implement the increment behavior.

Output



```
Yearly Member:
Original Date: 31/12/2022
Date after increment: 31/12/2023

Monthly Member:
Original Date: 31/12/2022
Date after increment: 31/1/2023
```

Solution



```cpp
// www.gammal.tech

#include<iostream>
using namespace std;

class Gammal_Tech_member {
private:
    int day, month, year;

public:
    bool setDate(int d, int m, int y) {
        if (d >= 1 && d <= 31)
            day = d;
        else
            return false;

        if (m >= 1 && m <= 12)
            month = m;
        else
            return false;

        if (y >= 2020)
            year = y;
        else
            return false;

        return true;
    }

    int getDay() {
        return day;
    }

    int getMonth() {
        return month;
    }

    int getYear() {
        return year;
    }

    void print() {
        cout << day << "/" << month << "/" << year << endl;
    }
};

class yearly_member : public Gammal_Tech_member {
public:
    void operator++() {
        setDate(getDay(), getMonth(), getYear() + 1);
    }
};

class monthly_member : public Gammal_Tech_member {
public:
    void operator++() {
        if (getMonth() == 12) {
            setDate(getDay(), 1, getYear() + 1);
        } else {
            setDate(getDay(), getMonth() + 1, getYear());
        }
    }
};

int main() {
    yearly_member yearly;
    monthly_member monthly;

    cout << "Yearly Member:" << endl;
    yearly.setDate(31, 12, 2022);
    cout << "Original Date: ";
    yearly.print();
    ++yearly;
    cout << "Date after increment: ";
    yearly.print();

    cout << "\nMonthly Member:" << endl;
    monthly.setDate(31, 12, 2022);
    cout << "Original Date: ";
    monthly.print();
    ++monthly;
    cout << "Date after increment: ";
    monthly.print();

    return 0;
}
```