

# Inheritance in C++

**Fundamental Computer Programming- C++ Lab (II)**



**元智大學 資訊工程學系**

Department of Computer Science & Engineering

*Lecturer: Ho Quang Thai*

# Outline

- Introduction
- Base and Derived Classes
- Access Control and Interitance
- Type of Inheritance
- Multiple Inheritance
- Exercises

# Introduction

- One of the most important concepts in object-oriented programming.
- Allows us to **define a class in terms of another class**.
- Makes it **easier to create and maintain** an application.
- Provides an opportunity to **reuse** the code functionality and fast implementation time.

# Introduction

- When creating a class, instead of writing completely new data members and member functions, the programmer can designate that the new class should inherit the members of an existing class.
- This existing class is called the **base** class, and the new class is referred to as the **derived** class.
- The idea of inheritance implements the **is a relationship**.

# Base and Derived Classes

- A class can be derived **from more than one classes**
- It means it can **inherit data and functions from multiple base classes**.
- To define a derived class, we use a class derivation list to specify the base class(es).
- A class derivation list names one or more base classes and has the form.

```
class derived-class: access-specifier base-class
```

# Example

Base class

Derived class

OUTPUT

Total area: 35

```
#include <iostream>

using namespace std;

// Base class
class Shape {
public:
    void setWidth(int w) {
        width = w;
    }
    void setHeight(int h) {
        height = h;
    }

protected:
    int width;
    int height;
};

// Derived class
class Rectangle: public Shape {
public:
    int getArea() {
        return (width * height);
    }
};

int main(void) {
    Rectangle Rect;

    Rect.setWidth(5);
    Rect.setHeight(7);

    // Print the area of the object.
    cout << "Total area: " << Rect.getArea() << endl;

    return 0;
}
```

# Access Control and Inheritance

- A derived class can **access all the non-private members** of its base class.
- Summarize the different access types

Access	public	protected	private
Same class	yes	yes	yes
Derived classes	yes	yes	no
Outside classes	yes	no	no

# Access Control and Inheritance

- A derived class inherits all base class methods with the following **exceptions**:
  - Constructors, destructors and copy constructors of the base class.
  - Overloaded operators of the base class.
  - The friend functions of the base class.



# Type of Inheritance

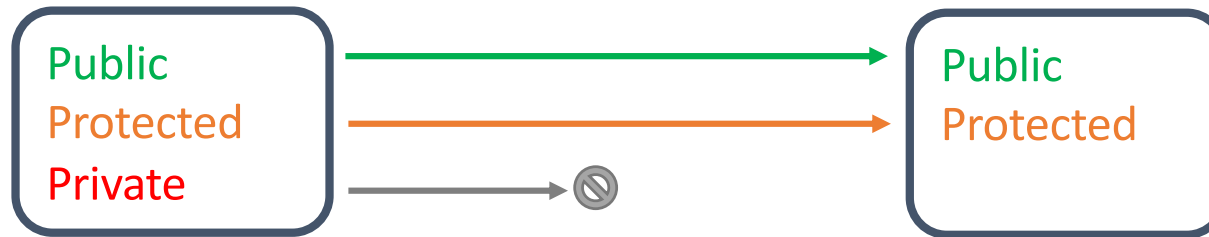
- When deriving a class from a base class, the base class may be inherited through **public**, **protected** or **private** inheritance.

```
class derived-class: access-specifier base-class
```

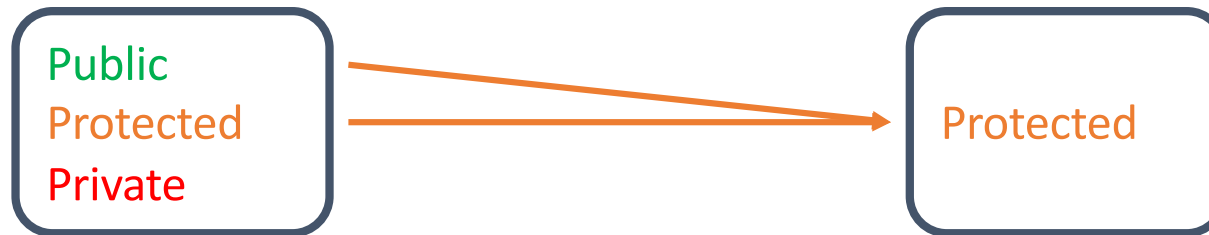
- We hardly use **protected** or **private** inheritance, but **public** inheritance is commonly used

# Type of Inheritance

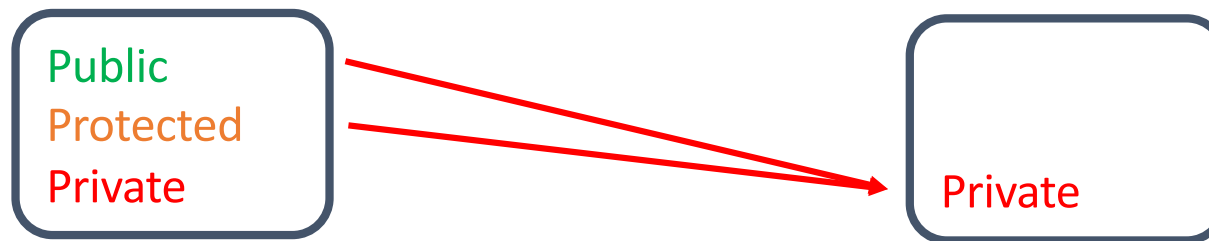
## Public Inheritance



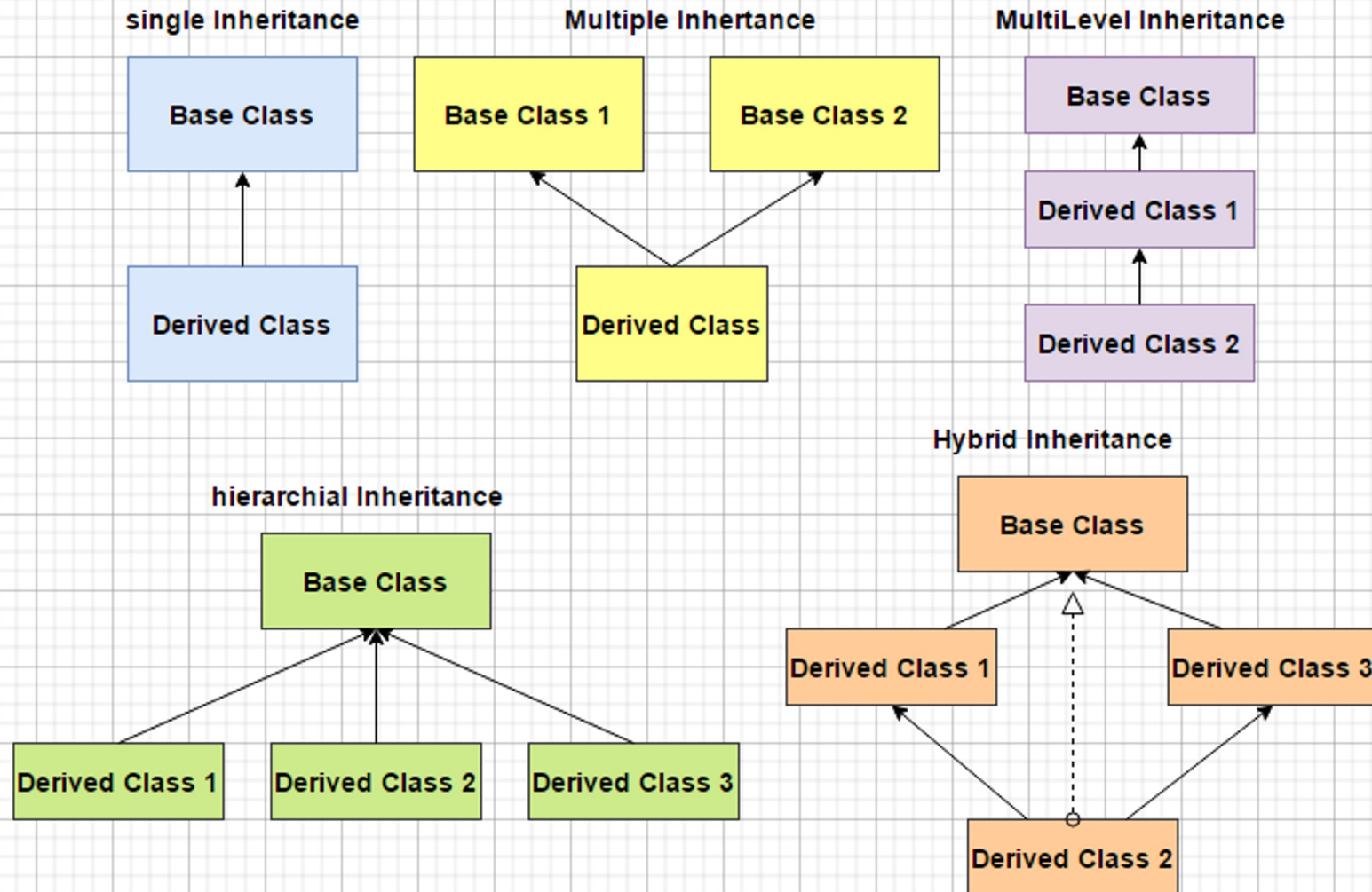
## Protected Inheritance



## Private Inheritance



# Type of Inheritance



# Multiple Inheritance

- A C++ class can inherit members from **more than one class** and here is the extended syntax

```
class derived-class: access base-A, access base-B ...
```

- Where access is one of public, protected, or private and would be given for every base class and they will be separated by comma as shown above.

# Example

Base class  
**Shape**

Base class  
**PaintCost**

Derived class

```
#include <iostream>

using namespace std;

// Base class Shape
class Shape {
public:
    void setWidth(int w) {
        width = w;
    }
    void setHeight(int h) {
        height = h;
    }

protected:
    int width;
    int height;
};

// Base class PaintCost
class PaintCost {
public:
    int getCost(int area) {
        return area * 70;
    }
};

// Derived class
class Rectangle: public Shape, public PaintCost {
public:
    int getArea() {
        return (width * height);
    }
};

int main(void) {
    Rectangle Rect;
    int area;

    Rect.setWidth(5);
    Rect.setHeight(7);

    area = Rect.getArea();

    // Print the area of the object.
    cout << "Total area: " << Rect.getArea() << endl;

    // Print the total cost of painting
    cout << "Total paint cost: $" << Rect.getCost(area) << endl;

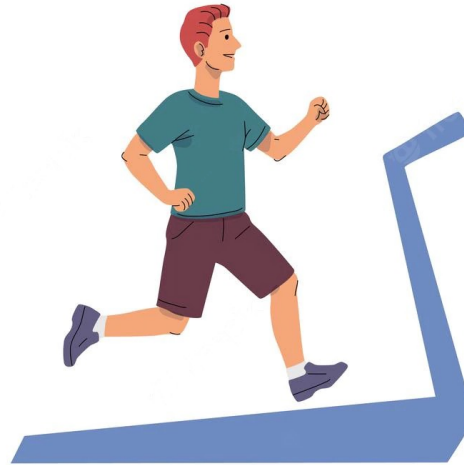
    return 0;
}
```

OUTPUT

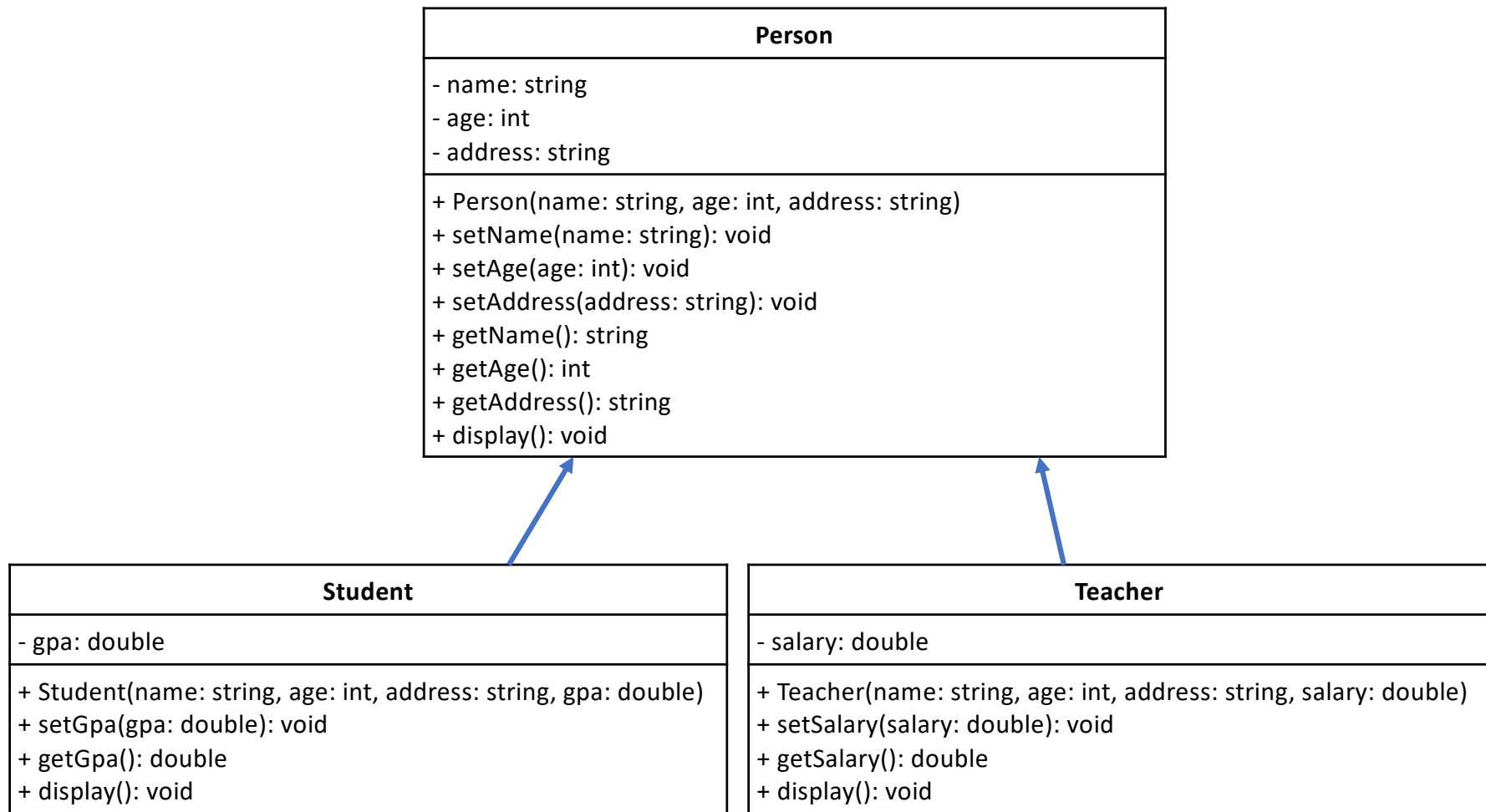
Total area: 35

Total paint cost: \$2450

# Exercises



# Exercise 1



# Exercise 1

Suggested main function:

```
#include <iostream>
#include "Person.cpp"
#include "Student.cpp"
#include "Teacher.cpp"

int main() {
    Student s("Lin Jia-Hao", 23, "52 Lide Street", 9.0);
    s.display();

    Teacher t("Chen Zhu-Wei", 35, "12 ShongShan Road", 42000);
    t.display();

    return 0;
}
```

OUTPUT

Name: Lin Jia-Hao

Age: 23

Address: 52 Lide Street

GPA: 9.0

Name: Chen Zhu-Wei

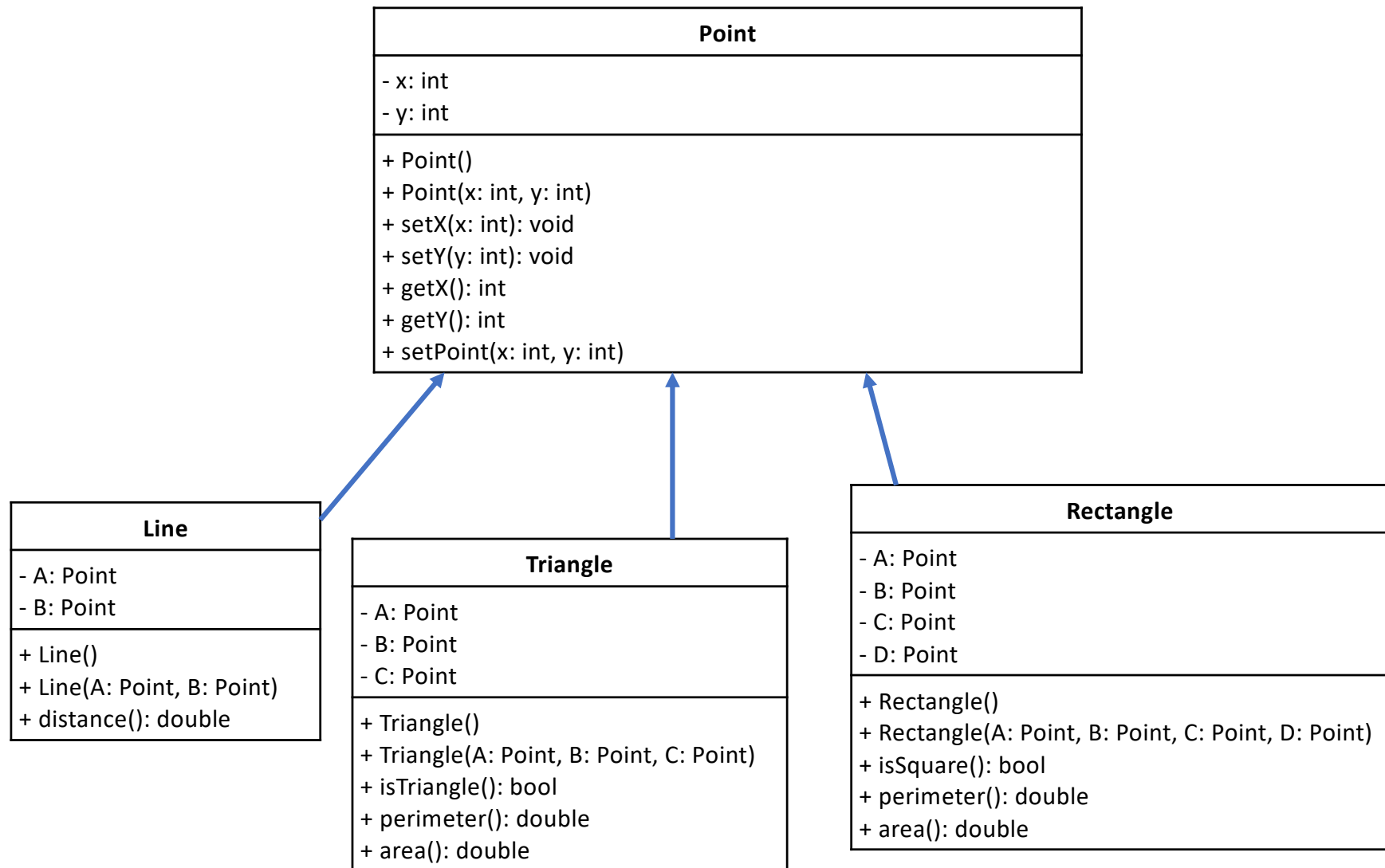
Age: 35

Address: 12 ShongShan Road

Salary: 42,000 TWD



# Exercise 2



# Exercise 2

Suggested main function:

```
#include <iostream>
#include "Point.cpp"
#include "Line.cpp"
#include "Triangle.cpp"
#include "Rectangle.cpp"

int main() {
    Point A(3,4);
    Point B(5,6);
    Point C(3,2);
    Point D(1,4);

    Line line(A,B);
    cout << "Distance AB: " << line.distance() << endl;

    Triangle tri(A,B,C);
    if (tri.isTriangle()) {
        cout << "ABC is a triangle" endl;
        cout << "Perimeter: " << tri.perimeter() << endl;
        cout << "Area: " << tri.area() << endl;
    }
    else {
        cout << "ABC is not a triangle" << endl;
    }

    Rectangle rec(A,B,C,D);
    cout << "Perimeter: " << rec.perimeter() << endl;
    cout << "Area: " << rec.area() << endl;

    return 0;
}
```



Finding **perimeter** and **area** formulas by yourself!!

# Questions & Answers