OOP作业4

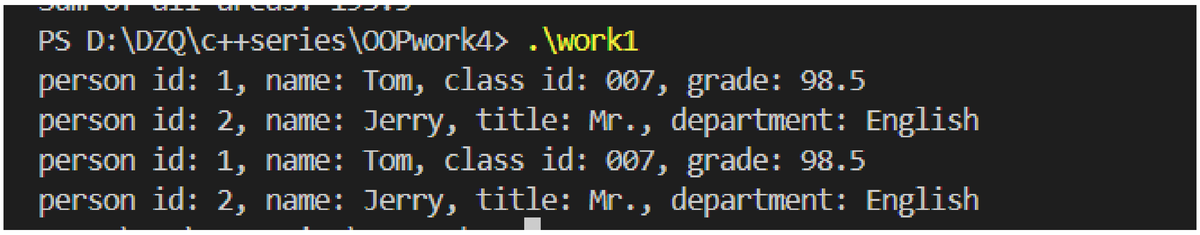
实验一

题目重述：编写一个学生和教师数据输入和显示程序。学生数据有编号、姓名、班号和成绩，教师数据有编号、姓名、职称和部门。要求将编号、姓名输入和显示设计成一个类Person，并作为学生类Student和教师类Teacher的基类。最终在主函数中进行测试。

代码：

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

// 基类Person
class Person {
    protected:
        int id;
        string name;

    public:
        Person(int id, string name) : id(id), name(name) {}

        virtual void display() {
            cout<<"person id: " << id << ", name: " << name;
        }

        virtual ~Person() {}
};

// 派生类Student
class Student : public Person {
    private:
        string classId;
        float grade;

    public:
        Student(int id, string name, string classId, float grade) : Person(id,
name), classId(classId), grade(grade) {}

        void display() override {
            Person::display();
            cout << ", class id: " + classId << ", grade: " << grade << endl;
        }
};

class Teacher : public Person {
    private:
        string title;
        string department;

    public:
        Teacher(int id, string name, string title, string department) :
Person(id, name), title(title), department(department) {}

        void display() override {
            Person::display();
            cout << ", title: " + title << ", department: " << department<< endl;
        }
};

int main() {
    Student s(1, "Tom", "007", 98.5);
    s.display();
    Teacher t(2, "Jerry", "Mr.", "English");
    t.display();

    vector<Person*> p;
    p.push_back(&s);
    p.push_back(&t);

    for (Person* person : p) {
        person->display();
    }
    return 0;
}
```

实验结果展示:



```
PS D:\DZQ\c++series\OOPwork4> .\work1
person id: 1, name: Tom, class id: 007, grade: 98.5
person id: 2, name: Jerry, title: Mr., department: English
person id: 1, name: Tom, class id: 007, grade: 98.5
person id: 2, name: Jerry, title: Mr., department: English
```

实验二

题目重述:分别定义Teacher(教师)类和Cadre(干部)类,采用多继承方式由这两个类派生出新类Teacher_Cadre(教师兼干部)。最终在主函数中进行测试。要求:

(1)在两个基类中都包含姓名、年龄、性别、地址、电话等数据成员。

(2)在Teacher类中还包含数据成员titile(职称),在Cadre类中还包含数据成员post(职务),在Teacher_Cadre类中还包含数据成员wages(工资)。

(3)对两个基类中的姓名、年龄、性别、地址、电话等数据成员用相同的名字,在引用这些数据成员时,指定作用域。

(4)在类体中声明成员函数,在类外定义成员函数。

(5)在派生类Teacher_Cadre的成员函数show中调用Teacher类中的display函数,输出姓名、年龄、性别、职称、地址、电话,然后再用cout语句输出职务与工资。

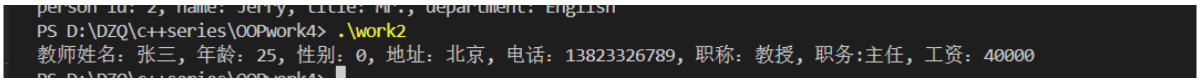
代码:

```
#include <iostream>
#include <vector>
```

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

class Teacher {
    protected:
        string name;
        int age;
        int sex; // 0代表男，1代表女
        string address;
        string phone;
        string title;

    public:
        Teacher(string name, int age, int sex, string address, string phone,
string title);
        void show();
};

class Cadre {
    protected:
        string name;
        int age;
        int sex; // 0代表男，1代表女
        string address;
        string phone;
        string post;

    public:
        Cadre(string name, int age, int sex, string address, string phone, string
post);
};

class Teacher_Cadre : public Teacher, public Cadre {
    private:
        float wages;

    public:
        Teacher_Cadre(string name, int age, int sex, string address, string
phone, string title, string post, float wages);
        void show();
};

Teacher::Teacher(string name, int age, int sex, string address, string phone,
string title) :
    name(name), age(age), sex(sex), address(address), phone(phone), title(title)
{}

void Teacher::show() {
    cout << "教师姓名: " << name << ", 年龄: " << age << ", 性别: " << sex << ", 地
址: " << address << ", 电话: " << phone << ", 职称: " << title;
}

Cadre::Cadre(string name, int age, int sex, string address, string phone, string
post) : name(name), age(age), sex(sex), address(address), phone(phone),
post(post) {}
```

```cpp
Teacher_Cadre::Teacher_Cadre(string name, int age, int sex, string address,
string phone, string title, string post, float wages) :
    Teacher(name, age, sex, address, phone, title), Cadre(name, age, sex,
address, phone, post), wages(wages) {}

void Teacher_Cadre::show() {
    Teacher::show();
    cout << ", 职务: " << Cadre::post << ", 工资: " << wages << endl;
}

int main() {
    Teacher_Cadre teacher("张三", 25, 0, "北京", "13823326789", "教授", "主任",
40000);
    teacher.show();
    return 0;
}
```

实验结果展示:



```
person id: 2, name: Jerry, title: Mr., department: English
PS D:\DZQ\c++series\OOPwork4> .\work2
教师姓名: 张三, 年龄: 25, 性别: 0, 地址: 北京, 电话: 13823326789, 职称: 教授, 职务:主任, 工资: 40000
PS D:\DZQ\c++series\OOPwork4>
```

实验三

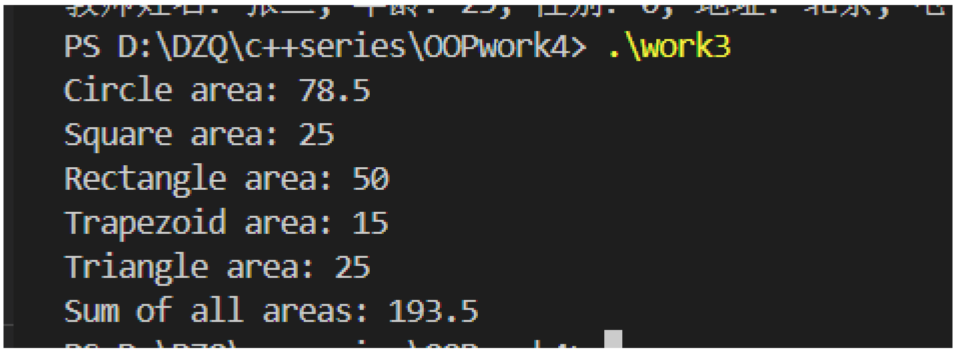
题目重述: 写一个程序，定义抽象基类Shape，由它派生出5个派生类: Circle，Square，Rectangle，Trapezoid，Triangle。用虚函数分别计算几种图形面积，并求它们的和。要求使用基类指针数组，使它的每一个元素指向一个派生类对象。最终在主函数中进行测试。

代码:

```cpp
#include <iostream>
#include <vector>
#include <cmath>
using namespace std;
#define PI 3.14

class Shape {
    protected:
        float area;

    public:
        ~Shape() {}
        virtual float getArea() {return 0;};
};

class Circle : public Shape {
    private:
        float radius;
    public:
        Circle(float r) : radius(r) {}
        float getArea() override {
            area = PI * pow(radius, 2);
            cout << "Circle area: " << area << endl;
            return area;
        };
};
```

```
class Square : public Shape {
    private:
        float a;
    public:
        Square(float a) : a(a) {}
        float getArea() override {
            area = a * a;
            cout << "Square area: " << area << endl;
            return area;
        };
};

class Rectangle : public Shape {
    private:
        float a;
        float b;
    public:
        Rectangle(float a, float b) : a(a), b(b) {}
        float getArea() override {
            area = a * b;
            cout << "Rectangle area: " << area << endl;
            return area;
        };
};

class Trapezoid : public Shape {
    private:
        float a;
        float b;
        float c;
    public:
        Trapezoid(float a, float b, float c) : a(a), b(b), c(c) {}
        float getArea() override {
            area = ((a + b)  * c) / 2.0;
            cout << "Trapezoid area: " << area << endl;
            return area;
        };
};

class Triangle : public Shape {
    private:
        float a;
        float b;
    public:
        Triangle(float a, float b) : a(a), b(b) {}
        float getArea() override {
            area = (a * b) / 2.0;
            cout << "Triangle area: " << area << endl;
            return area;
        };
};

int main () {
    Shape* shapes[5];
    shapes[0] = new Circle(5);
    shapes[1] = new Square(5);
    shapes[2] = new Rectangle(5, 10);
    shapes[3] = new Trapezoid(5, 10, 2);
    shapes[4] = new Triangle(5, 10);
    float sum = 0;

    for (int i = 0; i < 5; ++i) {
        sum += shapes[i]->getArea();
    }

    cout << "Sum of all areas: " << sum << endl;

}
```

实验结果展示:



```
PS D:\DZQ\c++series\OOPwork4> .\work3
Circle area: 78.5
Square area: 25
Rectangle area: 50
Trapezoid area: 15
Triangle area: 25
Sum of all areas: 193.5
```