上机实验报告 4

姓名：王越洋　　学号：22009200894

1. 实验一

1.1题目

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。

1.2 代码

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

// Person 类基类
class Person {
protected:
    string id;
    string name;
public:
    void input() {
        cout << "请输入编号: ";
        cin >> id;
        cout << "请输入姓名: ";
        cin >> name;
    }
    void display() const {
        cout << "编号: " << id << ", 姓名: " << name << endl;
    }
};

// Student 类继承自 Person
class Student : public Person {
private:
    string classNo;
    float grade;
public:
    void input() {
        Person::input();
        cout << "请输入班号: ";
        cin >> classNo;
```

```
        cout << "请输入成绩: ";
        cin >> grade;
    }
    void display() const {
        Person::display();
        cout << "班号: " << classNo << ", 成绩: " << grade << endl;
    }
};

// Teacher 类继承自 Person
class Teacher : public Person {
private:
    string title;
    string department;
public:
    void input() {
        Person::input();
        cout << "请输入职称: ";
        cin >> title;
        cout << "请输入部门: ";
        cin >> department;
    }
    void display() const {
        Person::display();
        cout << "职称: " << title << ", 部门: " << department << endl;
    }
};

int main() {
    Student student;
    Teacher teacher;

    cout << "输入学生信息:" << endl;
    student.input();

    cout << "输入教师信息:" << endl;
    teacher.input();

    cout << "\n学生信息:" << endl;
    student.display();

    cout << "教师信息:" << endl;
    teacher.display();

```

```
76.     return 0;
77.}
```

1.3 分析

（1）Person 类：

Person 是一个基类，包含了两个成员变量：id（编号）和 name（姓名）。这两个成员是所有继承者（Student 和 Teacher）的共同属性。

input() 函数：该函数用于输入 id 和 name，即学生或教师的基本信息。

display() 函数：该函数用于显示 id 和 name 信息。

（2）Student 类：

Student 继承自 Person 类，表示一个学生对象。除了继承自 Person 类的属性（id 和 name），它还包含两个独有的属性：classNo（班号）、grade（成绩）

input() 函数：通过调用基类 Person 的 input() 函数获取学生的基本信息，然后再输入学生的班号和成绩。

display() 函数：调用基类 Person 的 display() 函数显示学生的基本信息，然后显示学生的班号和成绩。

（3）Teacher 类：

Teacher 也继承自 Person 类，表示一个教师对象。除了继承 Person 类的属性，它还有两个独有的属性：title（职称）、department（部门）

input() 函数：通过调用基类 Person 的 input() 函数输入教师的基本信息，然后再输入教师的职称和部门。

display() 函数：调用基类 Person 的 display() 函数显示教师的基本信息，并显示职称和部门信息。

（4）main 函数：

创建了 Student 和 Teacher 类型的对象。

对学生和教师信息进行了输入和显示：

先输入学生信息，然后输入教师信息。

接着分别调用 Student 和 Teacher 类的 display() 函数显示其信息。

2. 实验二

2.1题目

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。

2.2代码

```
3. #include <iostream>
```

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

// Teacher 类
class Teacher {
protected:
    string name;
    int age;
    string gender;
    string address;
    string phone;
    string title;  // 职称
public:
    void input();  // 在类体内声明函数
    void display() const;  // 在类体内声明函数
};

// Cadre 类
class Cadre {
protected:
    string name;
    int age;
    string gender;
    string address;
    string phone;
    string post;  // 职务
public:
    void input();  // 在类体内声明函数
    void display() const;  // 在类体内声明函数
};

// Teacher_Cadre 类，继承自 Teacher 和 Cadre
class Teacher_Cadre : public Teacher, public Cadre {
private:
    double wages;  // 工资
public:
    void input();  // 在类体内声明函数
    void show() const;  // 在类体内声明函数
};

// Teacher 类成员函数定义
void Teacher::input() {
    cout << "请输入姓名: ";
    cin >> name;
```

```
48.     cout << "请输入年龄: ";
49.     cin >> age;
50.     cout << "请输入性别: ";
51.     cin >> gender;
52.     cout << "请输入地址: ";
53.     cin >> address;
54.     cout << "请输入电话: ";
55.     cin >> phone;
56.     cout << "请输入职称: ";
57.     cin >> title;
58.}
59.
60.void Teacher::display() const {
61.     cout << "姓名: " << name << ", 年龄: " << age << ", 性
   别: " << gender
62.          << ", 地址: " << address << ", 电话: " << phone << ", 职
   称: " << title << endl;
63.}
64.
65.// Cadre 类成员函数定义
66.void Cadre::input() {
67.     cout << "请输入姓名: ";
68.     cin >> name;
69.     cout << "请输入年龄: ";
70.     cin >> age;
71.     cout << "请输入性别: ";
72.     cin >> gender;
73.     cout << "请输入地址: ";
74.     cin >> address;
75.     cout << "请输入电话: ";
76.     cin >> phone;
77.     cout << "请输入职务: ";
78.     cin >> post;
79.}
80.
81.void Cadre::display() const {
82.     cout << "姓名: " << name << ", 年龄: " << age << ", 性
   别: " << gender
83.          << ", 地址: " << address << ", 电话: " << phone << ", 职
   务: " << post << endl;
84.}
85.
86.// Teacher_Cadre 类成员函数定义
87.void Teacher_Cadre::input() {
```

```
    Teacher::input();  // 调用 Teacher 类的输入函数
    Cadre::input();    // 调用 Cadre 类的输入函数
    cout << "请输入工资: ";
    cin >> wages;
}

void Teacher_Cadre::show() const {
    Teacher::display();  // 显示 Teacher 类的信息
    Cadre::display();    // 显示 Cadre 类的信息
    cout << "工资: " << wages << endl;
}

int main() {
    Teacher_Cadre teacher_cadre;
    teacher_cadre.input();
    cout << "\n 教师兼干部信息:" << endl;
    teacher_cadre.show();

    return 0;
}
```

2.3 分析

（1）Teacher 类：

Teacher 类包含了教师的基本信息（姓名、年龄、性别、地址、电话）以及特有的职称信息（title）。

它提供了 input() 函数，用于输入教师的个人信息和职称；以及 display() 函数，用于输出这些信息。

（2）Cadre 类：

Cadre 类用于表示干部，包含干部的基本信息（姓名、年龄、性别、地址、电话）以及职务信息（post）。

类似于 Teacher 类，Cadre 类也提供了 input() 和 display() 函数，分别用于输入和输出干部的个人信息和职务。

（3）Teacher_Cadre 类：

Teacher_Cadre 类是一个多继承类，继承自 Teacher 和 Cadre，表示一个既是教师又是干部的对象。

Teacher_Cadre 类包含额外的成员变量 wages（工资），用于表示这个人物的工资。

它提供了 input() 函数，首先调用 Teacher 类和 Cadre 类的 input() 函数来输入这两个角色的共有信息，然后输入工资。

show() 函数调用 Teacher 和 Cadre 的 display() 函数，输出所有的信息，并在最后显示工资。

（4）多继承的应用：

通过多继承，Teacher_Cadre 类能够同时访问 Teacher 和 Cadre 类中的数据成员和成员函数，完成了两个类特性的组合。派生类 Teacher_Cadre 通过调

用两个基类的 input() 和 display() 函数，避免了重复代码，并且能够轻松扩展额外的功能（如工资）。

（5）main 函数：

在 main 函数中，首先创建了一个 Teacher_Cadre 对象，然后调用其 input() 函数输入数据，接着调用 show() 函数输出包含教师、干部和工资等信息的完整数据。

3. 实验三

3.1 题目

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。

3.2 代码

```
#include <iostream>
#include <cmath>
using namespace std;

// 抽象基类 Shape
class Shape {
public:
    virtual double area() const = 0;  // 纯虚函数
    virtual ~Shape() {}  // 虚析构函数
};

// 圆形类
class Circle : public Shape {
private:
    double radius;
public:
    Circle(double r) : radius(r) {}
    double area() const  {
        return M_PI * radius * radius;
    }
};

// 正方形类
class Square : public Shape {
private:
    double side;
public:
    Square(double s) : side(s) {}
    double area() const  {
```

```
30.         return side * side;
31.     }
32. };
33. 
34. // 矩形类
35. class Rectangle : public Shape {
36. private:
37.     double width, height;
38. public:
39.     Rectangle(double w, double h) : width(w), height(h) {}
40.     double area() const  {
41.         return width * height;
42.     }
43. };
44. 
45. // 梯形类
46. class Trapezoid : public Shape {
47. private:
48.     double a, b, height;
49. public:
50.     Trapezoid(double x, double y, double h) : a(x), b(y), height(h) {
    }
51.     double area() const  {
52.         return 0.5 * (a + b) * height;
53.     }
54. };
55. 
56. // 三角形类
57. class Triangle : public Shape {
58. private:
59.     double base, height;
60. public:
61.     Triangle(double b, double h) : base(b), height(h) {}
62.     double area() const  {
63.         return 0.5 * base * height;
64.     }
65. };
66. 
67. int main() {
68.     Shape* shapes[5];
69.     shapes[0] = new Circle(5.0);
70.     shapes[1] = new Square(4.0);
71.     shapes[2] = new Rectangle(6.0, 8.0);
72.     shapes[3] = new Trapezoid(3.0, 5.0, 4.0);
```

```
    shapes[4] = new Triangle(6.0, 4.0);

    double total_area = 0;
    for (int i = 0; i < 5; ++i) {
        total_area += shapes[i]->area();
    }

    cout << "所有图形的总面积是: " << total_area << endl;

    for (int i = 0; i < 5; ++i) {
        delete shapes[i];
    }

    return 0;
}
```

3.3 分析

（1）Shape 抽象基类：

Shape 类是一个抽象类，它定义了一个纯虚函数 area()，这个函数在派生类中必须被重写，用于计算不同图形的面积。

Shape 类还定义了一个虚析构函数，确保在删除派生类对象时能够正确调用派生类的析构函数，避免内存泄漏。

（2）图形类（Circle, Square, Rectangle, Trapezoid, Triangle）：

每个图形类继承自 Shape 类，并重写 area() 函数来计算其对应图形的面积。

Circle 类：包含 radius 属性，通过 M_PI * radius * radius 公式计算圆的面积。

Square 类：包含 side 属性，通过 side * side 公式计算正方形的面积。Rectangle 类：包含 width 和 height 属性，通过 width * height 公式计算矩形的面积。

Trapezoid 类：包含底边 a、上边 b 和高 height 属性，通过 0.5 * (a + b) * height 公式计算梯形的面积。

Triangle 类：包含底边 base 和高 height 属性，通过 0.5 * base * height 公式计算三角形的面积。

（3）main 函数：

在 main 函数中，我们创建了一个 Shape* 类型的指针数组，每个数组元素指向一个不同类型的图形对象（如圆形、正方形、矩形、梯形和三角形）。

通过基类指针数组调用 area() 函数，利用多态机制动态计算每个图形的面积，并累加所有图形的面积，最终输出总面积。

程序的核心在于利用基类指针数组，充分展示了多态的特性，派生类对象的具体 area() 实现通过基类指针调用。

（4）多态的使用：

使用基类指针数组存储派生类对象，利用多态的特性在运行时决定调用哪个类的 area() 函数。通过这种方式，程序能够处理不同类型的图形对象，而不需

要显式地知道它们的类型。

这样设计使得程序具备很好的扩展性。未来如果需要增加更多的图形类型，只需要添加新的派生类，并实现 area() 函数，无需修改原有代码。