上机实验报告4

姓名：王越洋 学号：22009200894

1. **实验一**
   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**

1. #include <iostream>
2. #include <string>
3. using namespace std;
4. *// Person 类基类*
5. class Person {
6. protected:
7. string id;
8. string name;
9. public:
10. void input() {
11. cout << "请输入编号: ";
12. cin >> id;
13. cout << "请输入姓名: ";
14. cin >> name;
15. }
16. void display() const {
17. cout << "编号: " << id << ", 姓名: " << name << endl;
18. }
19. };
20. *// Student 类继承自 Person*
21. class Student : public Person {
22. private:
23. string classNo;
24. float grade;
25. public:
26. void input() {
27. Person::input();
28. cout << "请输入班号: ";
29. cin >> classNo;
30. cout << "请输入成绩: ";
31. cin >> grade;
32. }
33. void display() const {
34. Person::display();
35. cout << "班号: " << classNo << ", 成绩: " << grade << endl;
36. }
37. };
38. *// Teacher 类继承自 Person*
39. class Teacher : public Person {
40. private:
41. string title;
42. string department;
43. public:
44. void input() {
45. Person::input();
46. cout << "请输入职称: ";
47. cin >> title;
48. cout << "请输入部门: ";
49. cin >> department;
50. }
51. void display() const {
52. Person::display();
53. cout << "职称: " << title << ", 部门: " << department << endl;
54. }
55. };
56. int main() {
57. Student student;
58. Teacher teacher;
60. cout << "输入学生信息:" << endl;
61. student.input();
63. cout << "输入教师信息:" << endl;
64. teacher.input();
66. cout << "\n学生信息:" << endl;
67. student.display();
69. cout << "教师信息:" << endl;
70. teacher.display();
71. return 0;
72. }

**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。

1. **实验二**
   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。

* 1. **代码**

1. #include <iostream>
2. #include <string>
3. using namespace std;
4. *// Teacher 类*
5. class Teacher {
6. protected:
7. string name;
8. int age;
9. string gender;
10. string address;
11. string phone;
12. string title;  *// 职称*
13. public:
14. void input();  *// 在类体内声明函数*
15. void display() const;  *// 在类体内声明函数*
16. };
17. *// Cadre 类*
18. class Cadre {
19. protected:
20. string name;
21. int age;
22. string gender;
23. string address;
24. string phone;
25. string post;  *// 职务*
26. public:
27. void input();  *// 在类体内声明函数*
28. void display() const;  *// 在类体内声明函数*
29. };
30. *// Teacher\_Cadre 类，继承自 Teacher 和 Cadre*
31. class Teacher\_Cadre : public Teacher, public Cadre {
32. private:
33. double wages;  *// 工资*
34. public:
35. void input();  *// 在类体内声明函数*
36. void show() const;  *// 在类体内声明函数*
37. };
38. *// Teacher 类成员函数定义*
39. void Teacher::input() {
40. cout << "请输入姓名: ";
41. cin >> name;
42. cout << "请输入年龄: ";
43. cin >> age;
44. cout << "请输入性别: ";
45. cin >> gender;
46. cout << "请输入地址: ";
47. cin >> address;
48. cout << "请输入电话: ";
49. cin >> phone;
50. cout << "请输入职称: ";
51. cin >> title;
52. }
53. void Teacher::display() const {
54. cout << "姓名: " << name << ", 年龄: " << age << ", 性别: " << gender
55. << ", 地址: " << address << ", 电话: " << phone << ", 职称: " << title << endl;
56. }
57. *// Cadre 类成员函数定义*
58. void Cadre::input() {
59. cout << "请输入姓名: ";
60. cin >> name;
61. cout << "请输入年龄: ";
62. cin >> age;
63. cout << "请输入性别: ";
64. cin >> gender;
65. cout << "请输入地址: ";
66. cin >> address;
67. cout << "请输入电话: ";
68. cin >> phone;
69. cout << "请输入职务: ";
70. cin >> post;
71. }
72. void Cadre::display() const {
73. cout << "姓名: " << name << ", 年龄: " << age << ", 性别: " << gender
74. << ", 地址: " << address << ", 电话: " << phone << ", 职务: " << post << endl;
75. }
76. *// Teacher\_Cadre 类成员函数定义*
77. void Teacher\_Cadre::input() {
78. Teacher::input();  *// 调用 Teacher 类的输入函数*
79. Cadre::input();    *// 调用 Cadre 类的输入函数*
80. cout << "请输入工资: ";
81. cin >> wages;
82. }
83. void Teacher\_Cadre::show() const {
84. Teacher::display();  *// 显示 Teacher 类的信息*
85. Cadre::display();    *// 显示 Cadre 类的信息*
86. cout << "工资: " << wages << endl;
87. }
88. int main() {
89. Teacher\_Cadre teacher\_cadre;
90. teacher\_cadre.input();
91. cout << "\n教师兼干部信息:" << endl;
92. teacher\_cadre.show();
94. return 0;
95. }

**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**

1. #include <iostream>
2. #include <cmath>
3. using namespace std;
4. *// 抽象基类 Shape*
5. class Shape {
6. public:
7. virtual double area() const = 0;  *// 纯虚函数*
8. virtual ~Shape() {}  *// 虚析构函数*
9. };
10. *// 圆形类*
11. class Circle : public Shape {
12. private:
13. double radius;
14. public:
15. Circle(double r) : radius(r) {}
16. double area() const  {
17. return M\_PI \* radius \* radius;
18. }
19. };
20. *// 正方形类*
21. class Square : public Shape {
22. private:
23. double side;
24. public:
25. Square(double s) : side(s) {}
26. double area() const  {
27. return side \* side;
28. }
29. };
30. *// 矩形类*
31. class Rectangle : public Shape {
32. private:
33. double width, height;
34. public:
35. Rectangle(double w, double h) : width(w), height(h) {}
36. double area() const  {
37. return width \* height;
38. }
39. };
40. *// 梯形类*
41. class Trapezoid : public Shape {
42. private:
43. double a, b, height;
44. public:
45. Trapezoid(double x, double y, double h) : a(x), b(y), height(h) {}
46. double area() const  {
47. return 0.5 \* (a + b) \* height;
48. }
49. };
50. *// 三角形类*
51. class Triangle : public Shape {
52. private:
53. double base, height;
54. public:
55. Triangle(double b, double h) : base(b), height(h) {}
56. double area() const  {
57. return 0.5 \* base \* height;
58. }
59. };
60. int main() {
61. Shape\* shapes[5];
62. shapes[0] = new Circle(5.0);
63. shapes[1] = new Square(4.0);
64. shapes[2] = new Rectangle(6.0, 8.0);
65. shapes[3] = new Trapezoid(3.0, 5.0, 4.0);
66. shapes[4] = new Triangle(6.0, 4.0);
68. double total\_area = 0;
69. for (int i = 0; i < 5; ++i) {
70. total\_area += shapes[i]->area();
71. }
72. cout << "所有图形的总面积是: " << total\_area << endl;
73. for (int i = 0; i < 5; ++i) {
74. delete shapes[i];
75. }
76. return 0;
77. }

**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。