

《面向对象程序设计》

实验报告

|  |  |
| --- | --- |
| 班 级： | **计222** |
| 学 号： | **22013166** |
| 姓 名： | **杨建鑫** |
| 指导教师： | **梁建宁** |

信息科学与工程学院

2023年11月

**实验四、类的继承与派生**

**【实验目的】**

1、理解和掌握类的继承

2、学习组合和继承两种机制在重用设计上的区别

**【实验内容】**

1. 建立基类Building, 作为楼房类，这个基类包含楼层数、房间数、楼房总面积。再建立派生类Home\_Arch, Office\_Building,分别为住宅楼类、办公室楼类。在类Home\_Arch中，包含的内容有卧室数、客厅数、卫生间数和厨房数。在类Office\_Building中，包含的内容有办公室数和会议室数。创建对象测试程序。

Code：

#include <bits/stdc++.h>

using namespace std;

class Building

{

protected:

    int floors;

    int rooms;

    float area;

public:

    Building(int *floors*, int *rooms*, float *area*)

    {

        this->floors = *floors*;

        this->rooms = *rooms*;

        this->area = *area*;

    }

    void display()

    {

        cout << "Floors: " << floors << endl;

        cout << "Rooms: " << rooms << endl;

        cout << "Area: " << area << " sq. ft." << endl;

    }

};

class Home\_Arch : public Building

{

private:

    int bedrooms;

    int livingRooms;

    int bathrooms;

    int kitchens;

public:

    Home\_Arch(int *floors*, int *rooms*, float *area*, int *bedrooms*, int *livingRooms*, int *bathrooms*, int *kitchens*)

        : Building(*floors*, *rooms*, *area*)

    {

        this->bedrooms = *bedrooms*;

        this->livingRooms = *livingRooms*;

        this->bathrooms = *bathrooms*;

        this->kitchens = *kitchens*;

    }

    void display()

    {

        Building::display();

        cout << "Bedrooms: " << bedrooms << endl;

        cout << "Living Rooms: " << livingRooms << endl;

        cout << "Bathrooms: " << bathrooms << endl;

        cout << "Kitchens: " << kitchens << endl;

    }

};

class Office\_Building : public Building

{

private:

    int offices;

    int meetingRooms;

public:

    Office\_Building(int *floors*, int *rooms*, float *area*, int *offices*, int *meetingRooms*)

        : Building(*floors*, *rooms*, *area*)

    {

        this->offices = *offices*;

        this->meetingRooms = *meetingRooms*;

    }

    void display()

    {

        Building::display();

        cout << "Offices: " << offices << endl;

        cout << "Meeting Rooms: " << meetingRooms << endl;

    }

};

int main(void)

{

    Home\_Arch home(5, 10, 3000.0, 3, 1, 4, 2);

    Office\_Building office(10, 50, 10000.0, 20, 5);

    cout << "Home Architecture Details:" << endl;

    home.display();

    cout << "\nOffice Building Details:" << endl;

    office.display();

    return 0;

}

Example：

文本

描述已自动生成

1. 利用**继承**实现Point、Circle和Cylinder类。

Code：

#include <bits/stdc++.h>

using namespace std;

const int PI = 3.14;

*// point*

class Point

{

protected:

    int x;

    int y;

public:

    Point(int *x*, int *y*) { this->x = *x*, this->y = *y*; }

    void show() { cout << "Point: (" << x << ", " << y << ")" << endl; }

    void modify(int *x*, int *y*) { this->x = *x*, this->y = *y*; }

};

*// circle*

class Circle : public Point

{

protected:

    float radius;

public:

    Circle(int *x*, int *y*, float *radius*) : Point(*x*, *y*) { this->radius = *radius*; }

    int getArea()

    {

        return PI \* radius \* radius;

    }

    void show()

    {

        Point::show();

        cout << "Circle: radius = " << radius << endl;

    }

    void modify(int *x*, int *y*, float *radius*)

    {

        Point::modify(*x*, *y*);

        this->radius = *radius*;

    }

};

*// cylinder*

class Cylinder : public Circle

{

private:

    float height;

public:

    Cylinder(int *x*, int *y*, float *radius*, float *height*) : Circle(*x*, *y*, *radius*)

    {

        this->height = *height*;

    }

    int getArea()

    {

        return 2 \* PI \* radius \* (radius + height);

    }

    void show()

    {

        Circle::show();

        cout << "Cylinder: height = " << height << endl;

    }

    void modify(int *x*, int *y*, float *radius*, float *height*)

    {

        Circle::modify(*x*, *y*, *radius*);

        this->height = *height*;

    }

};

int main()

{

    Point p(1, 4);

    p.show();

    cout << endl;

    Circle c(20, 55, 46);

    c.show();

    cout << "Area of Circle: " << c.getArea() << endl;

    cout << endl;

    Cylinder cy(100, 9, 32, 78);

    cy.show();

    cout << "Area of Cylinder: " << cy.getArea() << endl;

    return 0;

}

Example：

文本

描述已自动生成

优点和缺点分析：

1. 优点：

继承可以让派生类获得基类的属性和方法，节省代码量，便于代码管理和维护。继承可以实现代码重用，降低程序设计的复杂度。  
 组合可以让对象之间的关系更加松散，减少对象之间的依赖性。  
组合可以提高程序的灵活性，使得程序的修改更容易。

1. 缺点：

继承可能导致父类和子类之间过度紧密的耦合关系，当基类的设计变动时，会影响整个继承层次的实现，难以维护和修改。继承层次的维护需要付出更多的时间和精力，因为子类仍然需要继承父类的一些方法和属性。  
 组合需要增加代码的复杂度，因为需要在一个类中包含另一个类的实例，导致代码更加难以管理和维护。组合和继承可以导致代码的重复使用，可能会出现不必要的重复代码和过度耦合的类结构问题。