**13 简答题1**

**什么是面向对象编程？C++中如何实现面向对象编程？**

答：面向对象编程是一种程序设计范式，它以对象为中心，通过封装、继承和多态等概念来组织代码。这种编程范式的主要目标是提高代码的可重用性、可扩展性和可维护性。面向对象三个最基本的特性就是封装、继承与多态。C++认为万事万物都皆为对象，对象上有其属性和行为。如人可以作为对象，属性有姓名、年龄、身高、体重…，行为有走、跑、跳、吃饭、唱歌等。车也可以作为对象，属性有轮胎、方向盘、车灯…,行为有载人、放音乐、放空调等。具有相同性质的对象，我们可以抽象称为类，人属于人类，车属于车类。

关键的面向对象编程概念包括：

封装： 将数据和操作数据的方法封装在一个单元内。这个单元就是类，类的实例则是对象。封装可以隐藏对象的内部实现，只暴露必要的接口。在C++中，封装就是把数据和方法打包到一个类里面。

如设计一个圆类，将半径这一属性封装，用成员函数实现获取周长和面积方法：

//需求：设计一个圆类

#define PI 3.14

class round

{

private:

//属性

int r;//半径

public: //访问权限

//获取到圆的周长

double calculate ()

{

return 2 \* PI \* r;

}

//获取圆的面积

double area ()

{

return PI \* r\* r ;

}

};

继承： 允许一个类（子类/派生类）继承另一个类（父类/基类）的属性和方法。这便于代码的重用，同时可以建立类之间的层次关系。具体来说，继承就是基于现有的一个类去定义一个新类，这样有助于重用代码，更好的组织代码。

C++中继承的语法：class 子类 : 继承方式 父类

继承方式一共有三种：公共继承，保护继承，私有继承

如设计一个页面类

//公共页面

class BasePage

{

public:

void header()

{

cout << "首页、公开课、登录、注册...（公共头部）" << endl;

}

void footer()

{

cout << "帮助中心、交流合作、站内地图...(公共底部)" << endl;

}

void left()

{

cout << "Java,Python,C++...(公共分类列表)" << endl;

}

};

//Java页面

class Java : public BasePage

{

public:

void content()

{

cout << "JAVA学科视频" << endl;

}

};

//Python页面

class Python : public BasePage

{

public:

void content()

{

cout << "Python学科视频" << endl;

}

};

//C++页面

class CPP : public BasePage

{

public:

void content()

{

cout << "C++学科视频" << endl;

}

};

多态： 允许使用相同的接口来处理不同类型的对象，提高代码的灵活性和可扩展性。

多态分为静态多态和动态多态。

静态多态: 函数重载和运算符重载属于静态多态，复用函数名。

动态多态: 派生类和虚函数实现运行时多态。

静态多态和动态多态区别：

静态多态的函数地址早绑定 - 编译阶段确定函数地址。

动态多态的函数地址晚绑定 - 运行阶段确定函数地址。

如使用多态来设计制作饮品类：

//抽象制作饮品

class AbstractDrinking {

public:

//烧水

virtual void Boil() = 0;

//冲泡

virtual void Brew() = 0;

//倒入杯中

virtual void PourInCup() = 0;

//加入辅料

virtual void PutSomething() = 0;

//规定流程

void MakeDrink() {

Boil();

Brew();

PourInCup();

PutSomething();

}

};

//制作咖啡

class Coffee : public AbstractDrinking {

public:

//烧水

virtual void Boil() {

cout << "煮农夫山泉!" << endl;

}

//冲泡

virtual void Brew() {

cout << "冲泡咖啡!" << endl;

}

//倒入杯中

virtual void PourInCup() {

cout << "将咖啡倒入杯中!" << endl;

}

//加入辅料

virtual void PutSomething() {

cout << "加入牛奶!" << endl;

}

};

//制作茶水

class Tea : public AbstractDrinking {

public:

//烧水

virtual void Boil() {

cout << "煮自来水!" << endl;

}

//冲泡

virtual void Brew() {

cout << "冲泡茶叶!" << endl;

}

//倒入杯中

virtual void PourInCup() {

cout << "将茶水倒入杯中!" << endl;

}

//加入辅料

virtual void PutSomething() {

cout << "加入枸杞!" << endl;

}

};