计算机科学技术学院实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **课程名称** | 面向对象程序设计 | | | **学 号** |  |
| **实验项目** | 实验七 | | | **姓 名** |  |
| **学 时** | 2 | **项目性质** | 综合型 | **班 级** | 2405111B |
| **指导教师** | 蒋振刚 | **实验地点** | 实训楼424 | **日 期** | 2025年4月28日 |
| 1. **实验目的和要求**   第7个实验：虚拟继承  目的：验证虚拟继承在多基类继承中的作用  要求：合理设计一个至少三层的类族，通过虚拟继承，解决基类成员访问冲突的问题。 | | | | | |
| 1. **实验环境**   系统：fedora 42  环境：g++ | | | | | |
| 1. **实验内容与过程**   **#include <iostream>**  **#include <string>**  **using namespace std;**  **// 基类：People**  **class People {**  **protected:**  **string name;**  **int age;**  **public:**  **People(const string& n = "Unknown", int a = 0) : name(n), age(a) {**  **cout << "People constructor called. Name: " << name << ", Age: " << age << endl;**  **}**    **virtual ~People() {**  **cout << "People destructor called. Name: " << name << endl;**  **}**    **void showBasicInfo() const {**  **cout << "Name: " << name << ", Age: " << age << endl;**  **}**  **};**  **// 第二层：两个派生类通过虚拟继承People**  **class Student : virtual public People {**  **protected:**  **string studentID;**  **float gpa;**  **public:**  **Student(const string& n = "Unknown", int a = 0,**  **const string& sid = "00000", float g = 0.0)**  **: People(n, a), studentID(sid), gpa(g) {**  **cout << "Student constructor called. ID: " << studentID << ", GPA: " << gpa << endl;**  **}**    **~Student() {**  **cout << "Student destructor called. ID: " << studentID << endl;**  **}**    **void showStudentInfo() const {**  **cout << "Student ID: " << studentID << ", GPA: " << gpa << endl;**  **}**  **};**  **class Employee : virtual public People {**  **protected:**  **string employeeID;**  **double salary;**  **public:**  **Employee(const string& n = "Unknown", int a = 0,**  **const string& eid = "E0000", double s = 0.0)**  **: People(n, a), employeeID(eid), salary(s) {**  **cout << "Employee constructor called. ID: " << employeeID << ", Salary: " << salary << endl;**  **}**    **~Employee() {**  **cout << "Employee destructor called. ID: " << employeeID << endl;**  **}**    **void showEmployeeInfo() const {**  **cout << "Employee ID: " << employeeID << ", Salary: " << salary << endl;**  **}**  **};**  **// 第三层：多重继承自Student和Employee**  **class TeachingAssistant : public Student, public Employee {**  **private:**  **string department;**  **int workHours;**  **public:**  **TeachingAssistant(const string& n = "TA", int a = 22,**  **const string& sid = "S1234", float g = 3.5,**  **const string& eid = "E5678", double s = 2000,**  **const string& dept = "Computer Science", int hours = 20)**  **: People(n, a),  // 必须显式调用虚基类的构造函数**  **Student(n, a, sid, g),**  **Employee(n, a, eid, s),**  **department(dept),**  **workHours(hours) {**  **cout << "TeachingAssistant constructor called. Department: " << department**  **<< ", Work Hours: " << workHours << endl;**  **}**    **~TeachingAssistant() {**  **cout << "TeachingAssistant destructor called. Department: " << department << endl;**  **}**    **void showTAInfo() const {**  **// 通过虚拟继承，可以直接访问People的成员，不会产生歧义**  **showBasicInfo();**  **showStudentInfo();**  **showEmployeeInfo();**  **cout << "Department: " << department << ", Work Hours: " << workHours << " hrs/week" << endl;**  **}**  **};**  **int main() {**  **cout << "Creating a TeachingAssistant object:" << endl;**  **TeachingAssistant ta("Zhang San", 24, "S2023001", 3.8, "E2023045", 3000, "Computer Science", 15);**    **cout << "\nTeaching Assistant information:" << endl;**  **ta.showTAInfo();**    **// 验证没有歧义**  **cout << "\nDirectly accessing People members from TeachingAssistant:" << endl;**  **ta.showBasicInfo(); // 如果不使用虚拟继承，这一行会导致编译错误，因为存在歧义**    **return 0;**  **}** | | | | | |
| 1. **实验结果与分析**   **本实验通过三层类族验证了虚拟继承在多基类继承中的作用。当Student和Employee使用虚拟继承People时，TeachingAssistant多重继承二者后仅保留一个People子对象，彻底解决了成员访问的二义性问题。实验输出显示，创建TA对象时People构造函数仅调用一次，证明虚基类被正确共享。TA可直接调用showBasicInfo()，未触发编译错误，说明基类成员访问路径已合并。析构顺序严格按照反向构造原则执行，验证了对象结构的完整性。虚拟继承通过合并重复基类，确保派生类中仅存在单一公共基类实例，有效避免了传统多重继承导致的"菱形继承"问题。显式调用虚基类构造函数的设计，则强化了初始化控制的精确性。本实验证实虚拟继承是解决多继承冲突的关键机制。** | | | | | |
| 1. **实验心得**   通过本次虚拟继承实验，我深入理解了其在解决多继承成员冲突中的关键作用。当**Student**和**Employee**通过虚拟继承共享**People**基类时，**TeachingAssistant**多重继承后仅保留一个**People**实例，彻底避免了传统菱形继承中的二义性问题。实验结果验证：构造**TeachingAssistant**时**People**构造函数仅调用一次，且能直接访问**showBasicInfo()**，证明虚基类成员路径已合并。此外，显式调用虚基类构造函数的设计让我意识到初始化顺序的严谨性，而析构顺序的反向执行则体现了对象生命周期的完整性。实验还让我反思多继承的复杂性：尽管虚拟继承优化了内存和对象模型，但过度使用可能增加设计耦合。未来开发中，需权衡继承与组合的适用场景，优先通过合理类设计平衡复用性与可维护性。本次实验巩固了我对C++多继承机制的理解，为复杂系统设计奠定了基础。 | | | | | |
| 1. **教师评语** | | | | | |
| 1. **实验成绩**   教师签名： 蒋振刚 批阅日期： 2025 年 4 月 28日 | | | | | |

注：项目性质为 演示型、验证型、设计型、综合型和创新型。