**实验7：继承**

**姓名\_李飞飞\_\_班级\_\_软工2206\_学号\_\_202105710309\_\_\_**

* **请阅读此说明：实验7满分100分。做完实验后请按要求将代码和截图贴入该文档。然后将此文档、源代码文件（.hpp, .cpp）打包上传到学习通。**

**实验目的：熟悉并掌握继承机制，能够利用公有继承方式建立符合用户需求的类族。**

**实验要求：按照每个类两个文件的方式（一个头文件，一个源文件）组织工程内的代码。**

**实验内容：**

**1、请仔细观察下列类声明，并回答:**

**class A { //基类**

**public:**

**A(int v1=0,int v2=0,int v3=0):a(v1),b(v2),c(v3){ }**

**void F1(){cout<<** **"F1"<<a<<" "<<b<<" "<<c<<endl;}**

**int a;**

**protected:**

**void F2( ) {cout<<"F2"<<a<<" "<<b<<" "<<c<<endl;}**

**int b;**

**private:**

**void F3(){cout<<"F3"<<a<<" "<<b<<" "<<c<<endl;}**

**int c;**

**};**

**class B: public A{**

**public:**

**//B的构造函数缺失**

**void F4( ) {cout<<"F4"<<Ba<" "<<Bb<<" "<<Bc<<endl;}**

**int Ba;**

**protected:**

**void F5( ) {cout<<"F5"<<Ba<<" "<<Bb<<" "<<Bc<<endl;}**

**int Bb;**

**private:**

**void F6(){cout<<"F6"<<Ba<<" "<<Bb<<" "<<Bc<<endl;}**

**int Bc;**

**};**

**class C: protected B{**

**public:**

**//C的构造函数缺失**

**void F7(){cout<<"F7"<<Ba<<" "<<Bb <<endl;}**

**void F8(){cout<<"F8"<<Ca<<" "<<Cb <<endl;}**

**int Ca;**

**private:**

**int Cb;**

**};**

**//测试主函数**

**int main()**

**{**

**A Aobj1,Aobj2(1,2,3);**

**B Bobj1,Bobj2(1,2,3,4,5);**

**C Cobj1,Cobj2(1,2,3,4,5,6);**

**......**

**return 0;**

}

1. **填写表格,写出第一行标识符在第一列所展示的各个作用域的访问控制方式(public,protected,private)。（10分）**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **访问域\成员名** | **a** | **b** | **c** | **F1** | **F2** | **F3** | **Ba** | **Bb** | **F4** | **F5** | **F6** | **Ca** | **Cb** | **F7** |
| **A** | public | protected | private | public | protected | private |  |  |  |  |  |  |  |  |
| **B** | public | protected |  | public | protected |  | public | protected | public | protected | private |  |  |  |
| **C** | protected | protected |  | protected | protected |  | protected | protected | protected | protected |  | public | private | public |
| **main函数A的对象可访问（能访问V，不能X）** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **main函数B的对象可访问（能访问V，不能X）** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **main函数C的对象可访问（能访问V，不能X）** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

1. **补充完类B和类C缺失的构造函数，并将main的测试程序补充完整。要求在main中展示类A，类B，类C的所有可在main中访问的成员。（40分）**

* **补充B的构造函数：**

**B(int v1=0,int v2=0,int v3=0,int v4=0,int v5=0,int v6=0):A(v1,v2,v3),Ba(v4),Bb(v5),Bc(v6){}**

* **补充C的构造函数：**

**C(int v1=0,int v2=0,int v3=0,int v4=0,int v5=0,int v6=0,int v7=0,int v8=0):B(v1,v2,v3,v4,v5,v6),**

**Ca(v7),Cb(v8){}**

* **main函数：**

**int main(){**

**A Aobj1, Aobj2(1,2,3);**

**B Bobj1,Bobj2(1,2,3,4,5);**

**C Cobj1,Cobj2(1,2,3,4,5,6);**

**// 展示类A的成员**

**cout<<"Aobj1:\n";**

**Aobj1.F1();**

**cout<<"Aobj1.a "<<Aobj1.a<<endl;**

**cout<<"Aobj2:\n";**

**Aobj2.F1();**

**cout<<"Aobj2.a "<<Aobj2.a<<endl;**

**// 展示类B的成员**

**cout<<"Bobj1:\n";**

**Bobj1.F1();**

**Bobj1.F4();**

**cout<<"Bobj1.a "<<Bobj1.a<<endl;**

**cout<<"Bobj1.Ba "<<Bobj1.Ba<<endl;**

**cout<<"Bobj2:\n";**

**Bobj2.F1();**

**Bobj2.F4();**

**cout<<"Bobj2.a "<<Bobj2.a<<endl;**

**cout<<"Bobj2.Ba "<<Bobj2.Ba<<endl;**

**// 展示类C的成员**

**cout<<"Cobj1:\n";**

**Cobj1.F7();**

**Cobj1.F8();**

**cout<<"Cobj1.Ca: "<<Cobj1.Ca<<endl;**

**cout<<"Cobj2:\n";**

**Cobj2.F7();**

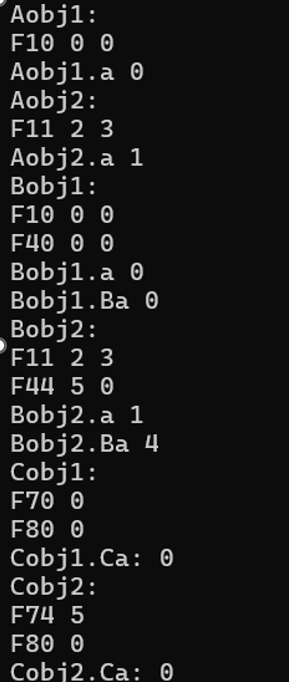
**Cobj2.F8();**

**cout<<"Cobj2.Ca: "<<Cobj2.Ca<<endl;**

**return 0;**

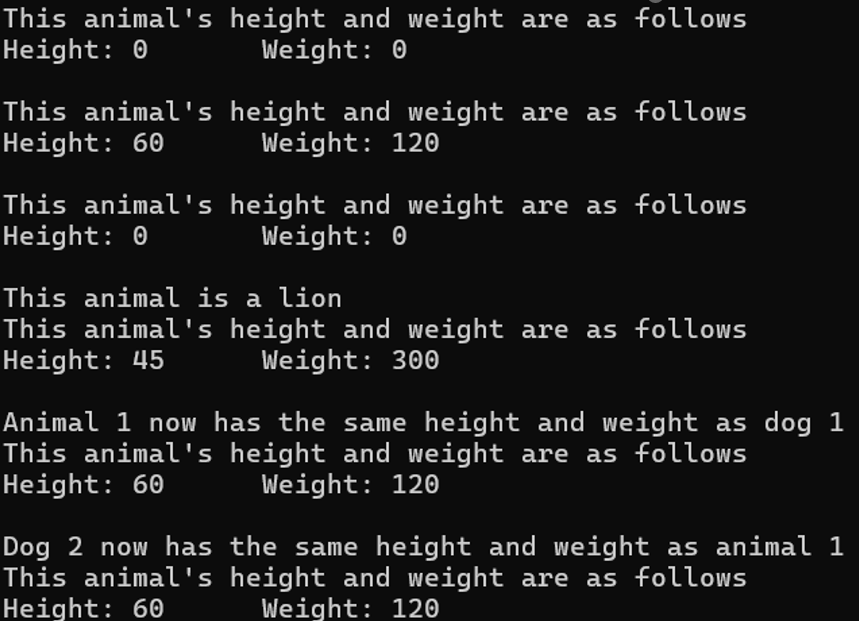
**}**

* **程序运行结果截屏：**

****

**2、代码调试: 附件中的代码在建立类族的过程中，由于编程人员的疏忽，出现了一些小问题，请帮忙修改过来。（20分）**

* **修改后的代码运行结果截屏：**

****

**3、设计交通工具类族: 开发一个名为Vehicle 的类的层次体系。创建两个类Taxi 和Truck，均以公有模式从类Vehicle 中继承而来。Taxi 类中应包含一个数据成员passenger说明其是否载客。Truck类应包含一个数据成员cargo说明其是否载货。根据题后附的测试程序输出结果 为类Vehicle添加必要的数据成员,并为所有类添加必要的函数来控制和访问类的数据。编写一段测试程序，将Vehicle对象、Truck 对象和Taxi对象打印到屏幕。（30分）**

**测试程序输出实例为：**

Vehicle

Number of doors: 2

Number of cylinders: 6

Transmission type: 3

Color: blue

Fuel level: 14.6

Taxi

Number of doors: 4

Number of cylinders: 6

Transmission type: 5

Color: yellow

Fuel level: 3.3

The taxi has no passengers.

Truck

Number of doors: 2

Number of cylinders: 16

Transmission type: 8

Color: black

Fuel level: 7.54

The truck is carrying cargo.

#include <iostream>

#include <string>

using namespace std;

class Vehicle{

protected:

    int numberOfDoors;

    int numberOfCylinders;

    int transmissionType;

    string color;

    double fuelLevel;

public:

    Vehicle(int doors = 2, int cylinders = 6, int transmission = 3, string color\_ = "blue", double fuel = 0)

        : numberOfDoors(doors), numberOfCylinders(cylinders), transmissionType(transmission), color(color\_), fuelLevel(fuel) {}

    void printInfo() const {

        cout<< "Vehicle\n"

            << "    Number of doors: " << numberOfDoors << "\n"

            << "    Number of cylinders: " << numberOfCylinders << "\n"

            << "    Transmission type: " << transmissionType << "\n"

            << "    Color: " << color << "\n"

            << "    Fuel level: " << fuelLevel << "\n";

    }

    void setColor(string newColor) { color = newColor; }

    string getColor() const { return color; }

};

class Taxi : public Vehicle {

private:

    bool passenger;

public:

    Taxi(int doors = 4, int cylinders = 6, int transmission = 5, string color\_ = "yellow", double fuel = 0, bool passenger = false)

        : Vehicle(doors, cylinders, transmission, color\_, fuel), passenger(passenger) {}

    void printInfo() const {

        Vehicle::printInfo();

        cout << "    The taxi has " << (passenger ? "" : "no ") << "passengers.\n";

    }

    void setHasPassenger(bool status) { passenger = status; }

    bool getHasPassenger() const { return passenger; }

};

class Truck : public Vehicle {

private:

    bool cargo;

public:

    Truck(int doors = 2, int cylinders = 16, int transmission = 8, string color\_ = "black", double fuel = 0, bool cargo = true)

        : Vehicle(doors, cylinders, transmission, color\_, fuel), cargo(cargo) {}

    void printInfo() const {

        Vehicle::printInfo();

        cout << "    The truck is " << (cargo ? "" : "not ") << "carrying cargo.\n";

    }

    void setIsCarryingCargo(bool status) { cargo = status; }

    bool getIsCarryingCargo() const { return cargo; }

};

int main() {

    Vehicle vehicle;

    Taxi taxi;

    Truck truck;

    vehicle.printInfo();

    cout << endl;

    taxi.printInfo();

    cout << endl;

    truck.printInfo();

    return 0;

}

效果：

