**实验7：继承**

**姓名：陈王子**

**班级：大数据分析2101班**

**学号：202103150503**

* **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 |
| A | public | protected | private | public | protected | private | - | - |
| B | public | protected | private | public | protected | private | public | protected |
| C | protected | protected | private | protected | protected | private | public | protected |
| main函数 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| F4 | F5 | F6 | Ca | Cb | F7 |
| - | - | - | - | - |  |
| public | protected | private | - | - |  |
| public | protected | private | protected | private | public |
| N/A | N/A | N/A | N/A | N/A | N/A |

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：**

class B: public A{

public:

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

    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;

};

* **补充C的构造函数：**

class C : protected B {

public:

    C(int v1 = 0, int v2 = 0, int v3 = 0, int v4 = 0, int v5 = 0, int v6 = 0) : B(v1, v2, v3, v4, v5), Ca(v6), Cb(0) {}

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

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

    int Ca;

protected:

    int Cb;

};

* **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);

    // 访问类A的成员

    Aobj1.a = 10;

    Aobj1.F1();

    // 访问类B的成员

    Bobj2.a = 20;

    Bobj2.Ba = 30;

    Bobj2.F4();

    // 访问类C的成员

    Cobj2.Ca = 60;

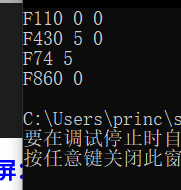
    Cobj2.F7();

    Cobj2.F8();

    return 0;

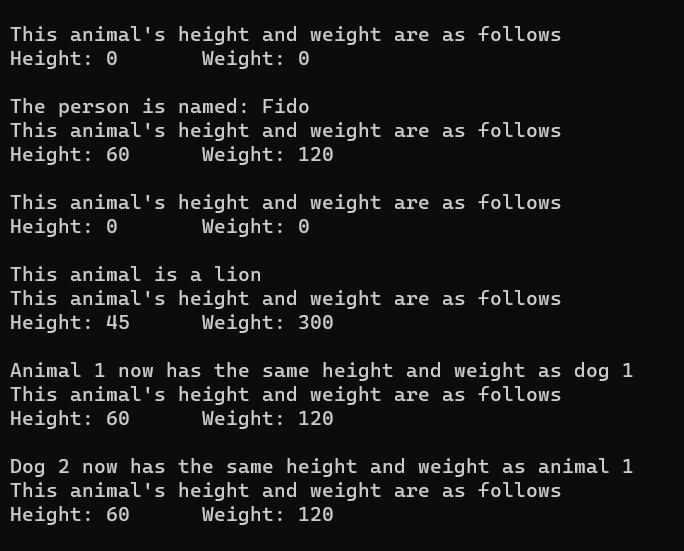
}

* **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）**

#include <iostream>

#include <string>

using namespace std;

class Vehicle {

public:

    int doors;

    int cylinders;

    int transmissionType;

    string color;

    double fuelLevel;

    Vehicle(int \_doors, int \_cylinders, int \_transmissionType, string \_color, double \_fuelLevel) {

        doors = \_doors;

        cylinders = \_cylinders;

        transmissionType = \_transmissionType;

        color = \_color;

        fuelLevel = \_fuelLevel;

    }

    virtual void print() {

        cout << "Vehicle" << endl;

        cout << "\tNumber of doors: " << doors << endl;

        cout << "\tNumber of cylinders: " << cylinders << endl;

        cout << "\tTransmission type: " << transmissionType << endl;

        cout << "\tColor: " << color << endl;

        cout << "\tFuel level: " << fuelLevel << endl;

    }

};

class Taxi : public Vehicle {

public:

    bool passenger;

    Taxi(int \_doors, int \_cylinders, int \_transmissionType, string \_color, double \_fuelLevel, bool \_passenger) : Vehicle(\_doors, \_cylinders, \_transmissionType, \_color, \_fuelLevel) {

        passenger = \_passenger;

    }

    void print() override {

        cout << "Taxi" << endl;

        cout << "\tNumber of doors: " << doors << endl;

        cout << "\tNumber of cylinders: " << cylinders << endl;

        cout << "\tTransmission type: " << transmissionType << endl;

        cout << "\tColor: " << color << endl;

        cout << "\tFuel level: " << fuelLevel << endl;

        if (passenger) {

            cout << "\tThe taxi has passengers." << endl;

        }

        else {

            cout << "\tThe taxi has no passengers." << endl;

        }

    }

};

class Truck : public Vehicle {

public:

    bool cargo;

    Truck(int \_doors, int \_cylinders, int \_transmissionType, string \_color, double \_fuelLevel, bool \_cargo) : Vehicle(\_doors, \_cylinders, \_transmissionType, \_color, \_fuelLevel) {

        cargo = \_cargo;

    }

    void print() override {

        cout << "Truck" << endl;

        cout << "\tNumber of doors: " << doors << endl;

        cout << "\tNumber of cylinders: " << cylinders << endl;

        cout << "\tTransmission type: " << transmissionType << endl;

        cout << "\tColor: " << color << endl;

        cout << "\tFuel level: " << fuelLevel << endl;

        if (cargo) {

            cout << "\tThe truck is carrying cargo." << endl;

        }

        else {

            cout << "\tThe truck is not carrying cargo." << endl;

        }

    }

};

int main() {

    Vehicle v(2, 6, 3, "blue", 14.6);

    v.print();

    Taxi t(4, 6, 5, "yellow", 3.3, false);

    t.print();

    Truck tr(2, 16, 8, "black", 7.54, true);

    tr.print();

    return 0;

}

