河北师范大学软件学院
Software College of Hebei Normal University

C++

第十三讲 继承与派生（三）

C++备课组 丁盟

自我介绍

丁盟

QQ：2622885094

上一讲教学目标

- 掌握C++中单重继承及多重继承中构造函数的调用
- 掌握C++中单重继承及多重继承中析构函数的调用

本讲教学目标

- 了解基类和派生类间的赋值兼容规则
- 了解同名冲突及其解决方案
- 了解虚基类和虚继承的概念

1 基类和派生类间的赋值兼容规则

2 同名冲突及其解决方案

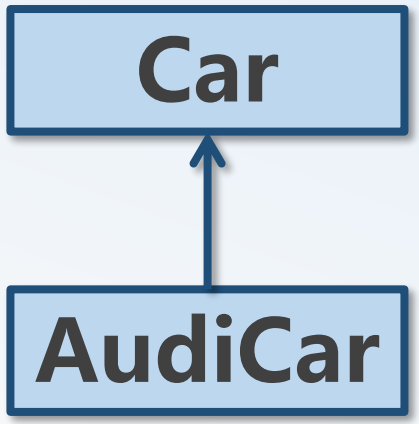
- 同名隐藏
- 多重继下同名冲突及解决方案

3 虚基类

- 虚基类和虚继承的概念
- 虚基类构造函数的调用

基类和派生类间的赋值兼容规则

- 在公有继承方式下，基类和派生类间存在赋值兼容，具体赋值规则
  - 基类对象 = 公有派生类对象(仅能访问基类部分)
  - 基类对象的指针 = 公有派生类对象的地址（同上）
  - 基类的引用 = 公有派生类对象（同上）




```
Car      carObj;
AudiCar audiObj;
carObj   = audiObj;
Car * pB    = &audiObj;
Car & refB = audiObj;
```

基类和派生类间的赋值兼容规则

```
class Car {
public:
    Car(){}
    void display() const {
      cout<<"Car" << endl;
    }
};
class AudiCar : public Car {
public:
    AudiCar():Car() {}
    void display() const {
       cout<<"AudiCar"<<endl;
    }
};
```

```
int main(void) {
//基类对象访问的一定是基类成员
    Car objCar;
    objCar.display();
//基类指针指向基类对象,只能访问基类成员
    Car * pCar = &objCar;
    pCar->display();
//基类指针指向派生类对象,只能访问基类成员
    AudiCar objDer;
    pCar = &objDer;
    pCar->display();
//基类引用作为派生类对象的别名,只能访问基类成员
    Car & obj = objDer;
    obj.display();
    return 0; }
```

基类和派生类间的赋值兼容规则

- 在公有继承方式下，赋值兼容规则的说明：
  - 基类对象只能访问基类的成员
  - 用基类的指针，无论是否指向基类对象，都只能访问基类部分的成员
  - 用基类的引用，无论是否是基类对象的别名，都只能访问基类部分的成员
  - 这是一种向上的类型转换，是安全的。

1 基类和派生类间的赋值兼容规则

2 同名冲突及其解决方案

- 同名隐藏
- 多重继下同名冲突及解决方案

3 虚基类

- 虚基类和虚继承的概念
- 虚基类构造函数的调用

同名隐藏

- 在派生类里如果存在与基类同名的成员，访问派生类里的同名成员时该如何处理？
  - 数据成员同名、成员函数同名

```
class Car {
public:
    void test() {
        cout << "Car";
    }
protected:
    int m_iVal;
};
```

```
class AudiCar : public Car {
public:
    void test() {
        cout << "AudiCar";
    }
protected:
    int m_iVal;
};
```

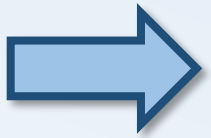
同名隐藏

❖ 同名隐藏

- 若派生类成员和基类成员重名，在派生类中使用的是派生类的同名成员，基类的同名成员自动被隐藏。
- 若要在派生类中访问基类中被隐藏同名成员（如果允许访问），可以使用基类名限定。

基类名::数据成员
基类名::成员函数名（实参）

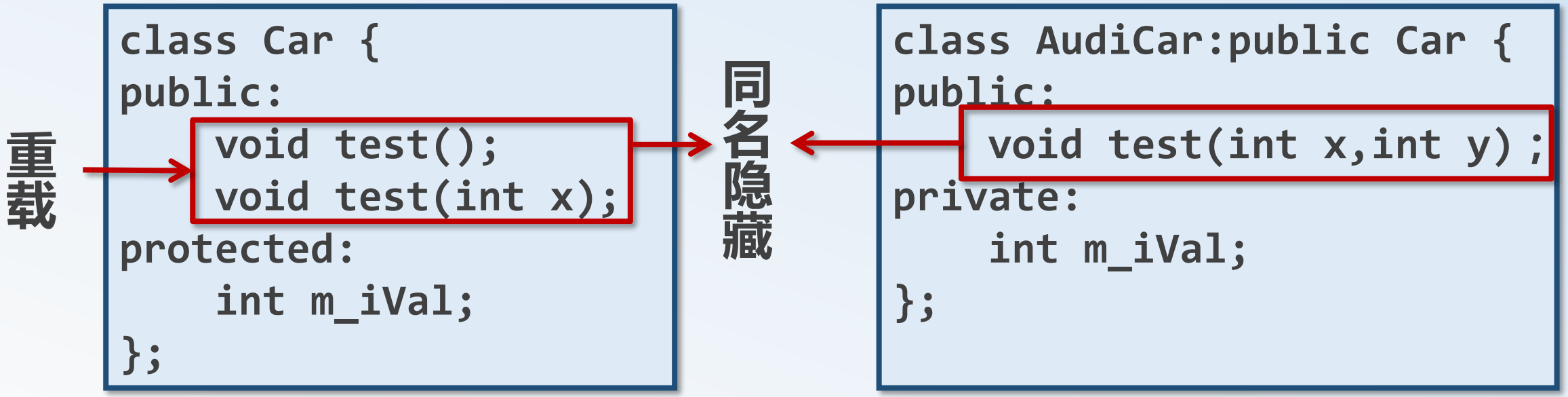
AudiCar



```
test();
m_ival;
Car::test();
Car::m_ival;
```

同名隐藏

- 如果基类和派生类中有同名函数，那么这个同名函数是函数重载吗？
  - 函数重载的条件是同一作用域，派生类和基类分别属于不同的作用域，所以派生类和基类中同名函数的不叫重载叫做同名隐藏。



1 基类和派生类间的赋值兼容规则

2 同名冲突及其解决方案

- 同名隐藏
- 多重继下同名冲突及解决方案

3 虚基类

- 虚基类和虚继承的概念
- 虚基类构造函数的调用

多重继下同名冲突及解决方案

- 什么是同名冲突？
  - 在多重继承下，在派生类的多个平行基类里有同名成员，则在派生类里有继承自基类的多个同名成员，访问派生类里同名成员时存在同名冲突

```
class Car1 {  public:
    void fun();
    int  m_iA;
};
class Car2 {  public:
    void fun();
    int  m_iA;
};
```

```
class AudiCar : public Car1,
                public Car2
{};
void fun() {
    AudiCar obj;
    obj.m_iA = 10;// 二义性
    obj.fun();    // 二义性
}
```

多重继下同名冲突及解决方案

- 多重继承下访问的二义性
  - 一派生类所继承的、来自不同基类的成员出现了同名，如何区分？用不同基类名限定同名的成员。

```
void fun()
{
    AudiCar obj;
    obj.Car1::m_iA = 10;
    obj.Car2::fun();
}
```

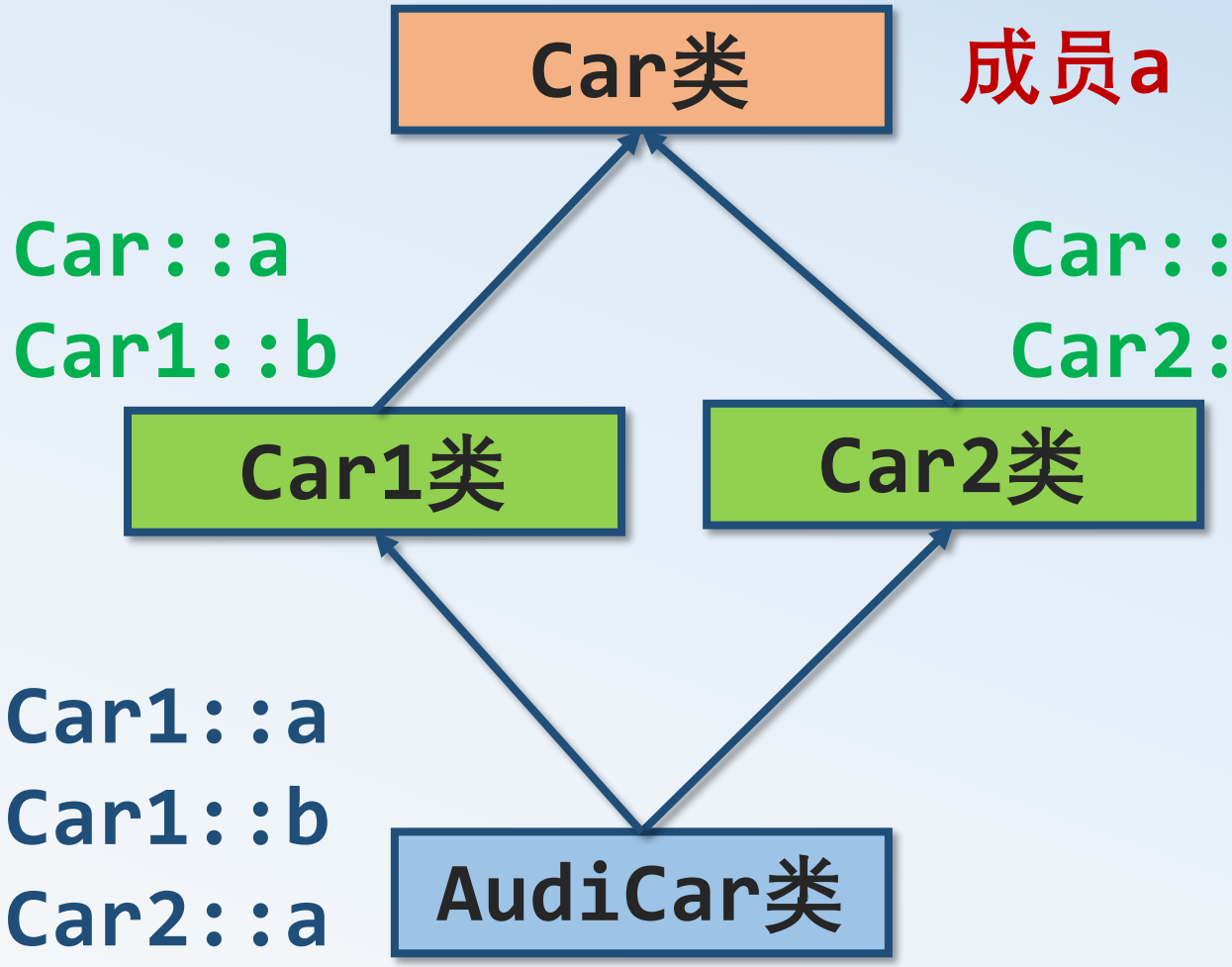
多重继下同名冲突及解决方案

- 当派生类从多个基类派生，而这些基类又从同一个基类派生，则在访问此共同基类中的成员时，将产生二义性(菱形问题)。

```
class Car{
public: int a;
};
class Car1 : public Car{
public: int b;
};
class Car2 : public Car{
public: int c;
};
```

```
class AudiCar :public Car1,
                public Car2
{};
void fun() {
    AudiCar obj;
    // 访问的二意性
    //obj.a = 100;
    //obj.Car::a = 100;
}
```

多重继下同名冲突及解决方案



声明:

```
AudiCar obj;
```

则以下访问有二义性：

```
obj.a
obj.Car::a
```

多重继下同名冲突及解决方案

- 解决方案
  - 这样的程序存在两个问题：
    - ✓ 访问上的二义性
    - ✓ 产生过多的冗余成员，浪费内存
  - 解决方法： 虚基类

1 基类和派生类间的赋值兼容规则

2 同名冲突及其解决方案

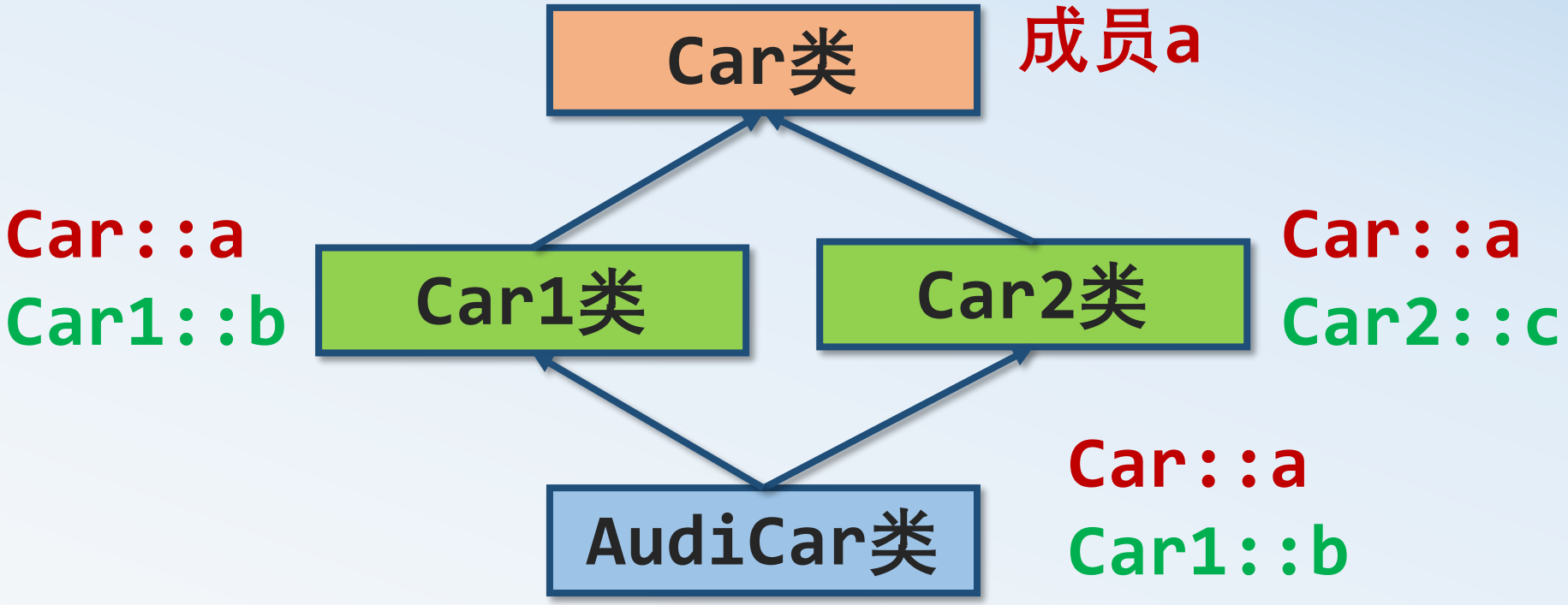
- 同名隐藏
- 多重继下同名冲突及解决方案

3 虚基类

- 虚基类和虚继承的概念
- 虚基类构造函数的调用

虚基类和虚继承的概念

- 作用：解决菱形问题，使基类成员在所有派生类中只存在一份拷贝



虚基类和虚继承的概念

- 虚基类：在虚继承体系中通过virtual继承而来的基类
- 虚继承：在继承中包含了virtual关键字的继承关系

格式：

```
class 派生类名 : virtual 继承方式 基类名
{...};
```

推荐

```
class 派生类名 : 继承方式 virtual 基类名
{...};
```

```
class x1 : virtual public x
{ …… };

class x2 : public virtual x
{ …… };
```

1 基类和派生类间的赋值兼容规则

2 同名冲突及其解决方案

- 同名隐藏
- 多重继下同名冲突及解决方案

3 虚基类

- 虚基类和虚继承的概念
- 虚基类构造函数的调用

虚基类构造函数的调用

- 调用原则：
  - 虚基类的直接或间接派生类构造函数中通常包含对该虚基类构造函数的直接调用，但如果表示调用虚基类默认构造函数则可省略
  - 虚基类中的数据成员只能被初始化一次，只在最后一层派生类的构造函数中调用虚基类的构造函数，虚基类直接派生类对虚基类构造函数的调用均被忽略

虚基类构造函数的调用

```
class Car {
public:
     Car();
     ~Car();
};
class Car1:virtual public Car {
public:
    Car1();
    ~Car1();
};
class Car2:virtual public Car {
public:
    Car2();
    ~Car2();
};
```

```
class AudiCar:public Car1,
                public Car2 {
public:
    AudiCar ;
    ~AudiCar();
};
int main(void)
{
    AudiCar obj;
    return 0;
}
```

```
Car 构造
Car1 构造
Car2 构造
AudiCar 构造

AudiCar 析构
Car2 析构
Car1 析构
Car 析构
```

虚基类构造函数的调用

- 如果主函数中定义的是Car1或Car2的对象而不是AudiCar的对象，则Car虚基类的特性不会显现！

```
int main(void)
{
    Car1 objCar1;
    Car2 objCar2;

    return 0;
}
```

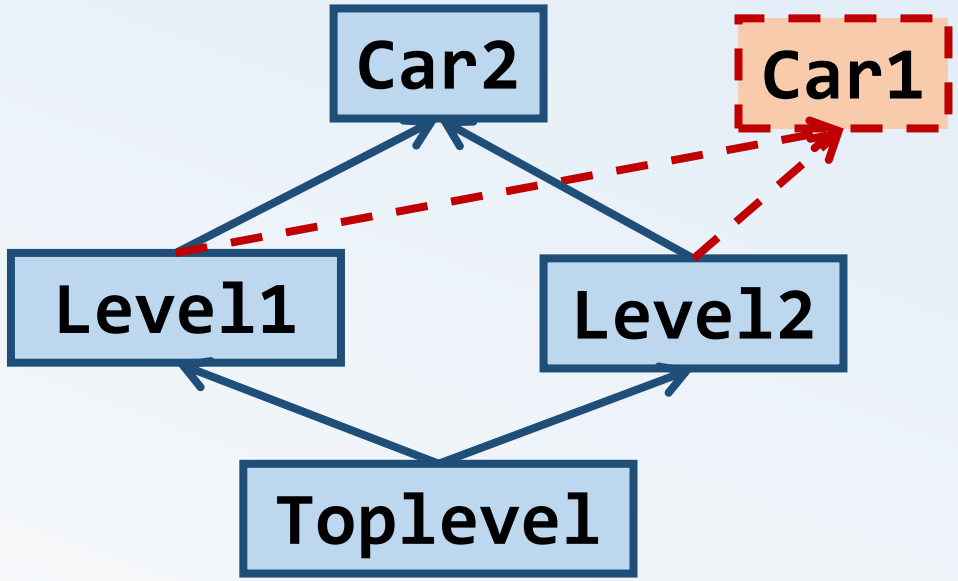
```
Car 构造
Car1 构造
Car 构造
Car2 构造

Car2 析构
Car 析构
Car1 析构
Car 析构
```

虚基类构造函数的调用

- 虚基类与普通基类的调用顺序

在同一层次中若同时包含对虚基类和非虚基类构造函数的调用，则一定先调用虚基类构造函数，再调用普通基类构造函数




```
class Car1
class Car2
class Level1
class Car2
class Level2
class TopLevel
```

虚基类构造函数的调用

```
class Car1 {
public:
    Car1() {
        cout << "class Car1"
             << endl;
    }
};
class Car2 {
public:
    Car2() {
        cout << "class Car2"
             << endl;
    }
};
```

```
class Level1 : public Car2,
               virtual public Car1 {
public:
    Level1() {
        cout << "class Level1"
             << endl;
    }
};
class Level2 : public Car2,
               virtual public Car1 {
public:
    Level2() {
        cout << "class Level2"
             << endl;
    }
};
```

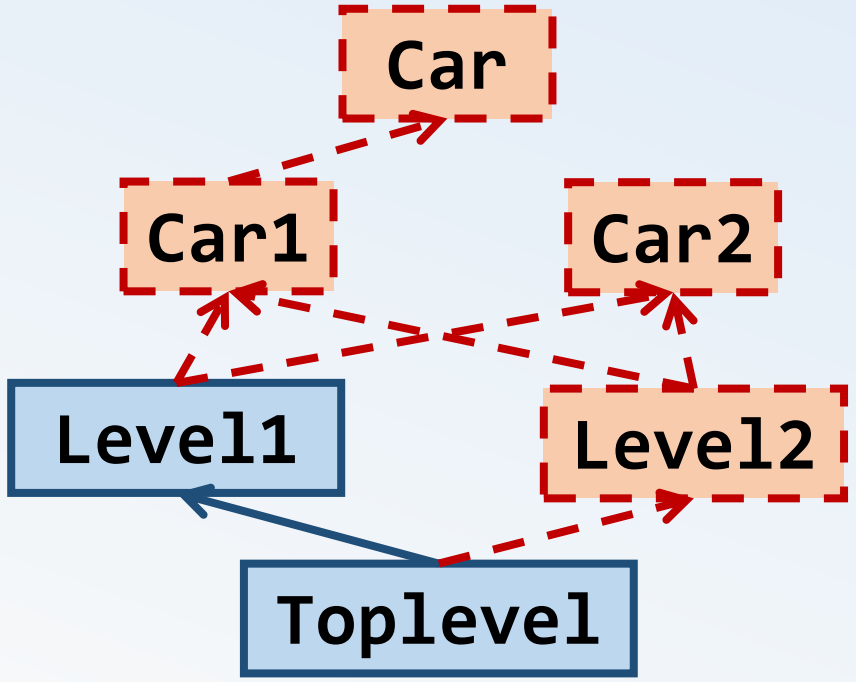
虚基类构造函数的调用

```
class Toplevel : public Level1,
                 virtual public Level2 {
public:
    Toplevel() {
        cout << "class Toplevel" << endl;
    }
};
int main(void) {
    Toplevel obj;
    return 0;
}
```

```
class Car1
class Car2
class Level2
class Car2
class Level1
class TopLevel
```

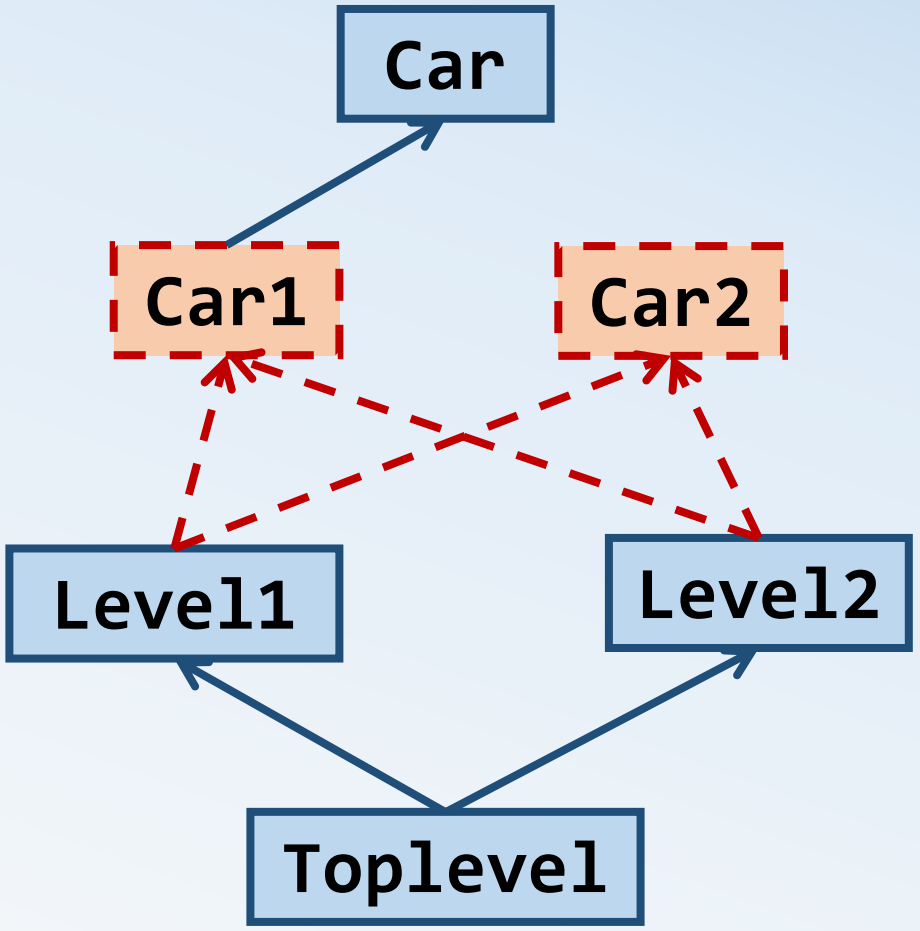
虚基类构造函数的调用

- 若有多个虚基类，仍然遵循自上而下先左后右的原则，与调用非虚基类的构造函数顺序一样
- 若虚基类是其他类的派生类，仍然先调用虚基类的基类构造函数再调用虚基类构造函数



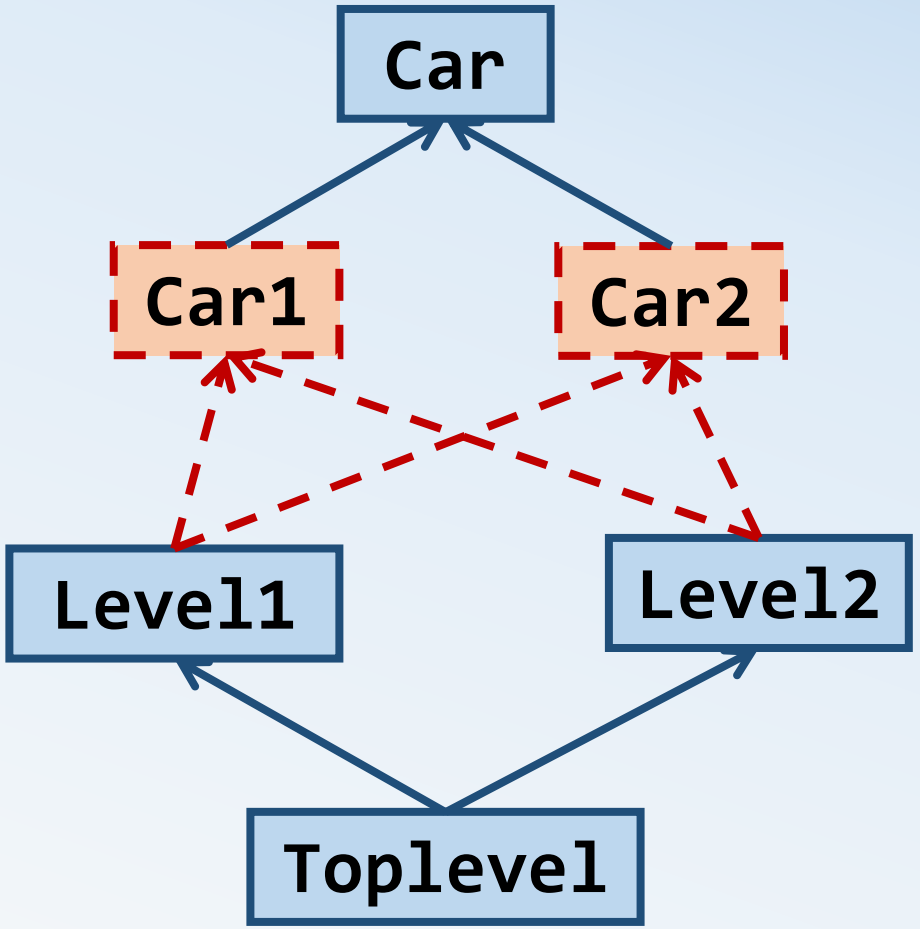

```
class Car
class Car1
class Car2
class Level2
class Level1
class TopLevel
```

虚基类构造函数的调用




```
class Car
class Car1
class Car2
class Level1
class Level2
class TopLevel
```

虚基类构造函数的调用




```
class Car
class Car1
class Car
class Car2
class Level1
class Level2
class TopLevel
```

本讲教学目标

- 了解基类和派生类间的赋值兼容规则
- 了解同名冲突及其解决方案
- 了解虚基类和虚继承的概念


THANKS