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

C++

第十一讲 继承与派生（一）

C++备课组 丁盟

自我介绍

丁盟

QQ：2622885094

上一讲教学目标

- 了解C++中嵌套类的使用
- 了解C++中局部类的使用

本讲教学目标

- 理解C++中继承的概念
- 掌握C++中如何使用继承
- 理解C++中的三种继承方式

1 为什么要使用继承

2 派生类的声明

3 继承方式

4 访问声明和using声明

为什么要使用继承

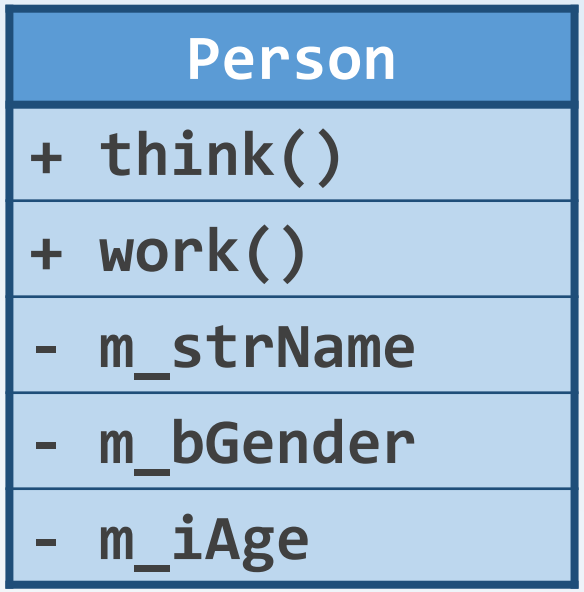
- 一个类中包含了若干数据成员和成员函数。在不同的类中，数据成员和成员函数是不相同的。但有时两个类的内容基本相同或有一部分相同。

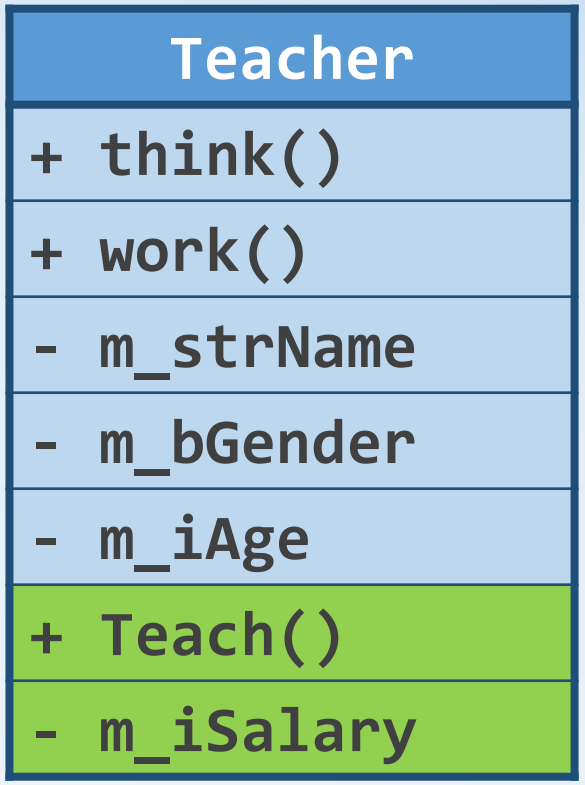
```
class Person {
public:
    void think();
    void work();
private:
    string m_strName;
    bool m_bGender;
    int m_iAge;
};
```

```
class Teacher {
public:
    void think();
    void work();
    void teach();
private:
    string m_strName;
    bool m_bGender;
    int m_iAge;
    int m_iSalary;
};
```

为什么要使用继承

- 在两个类Person和Teacher中的部分成员是相同的，C++有没有提供一种机制来实现代码的重用呢？





为什么要使用继承

- C++通过继承实现代码复用

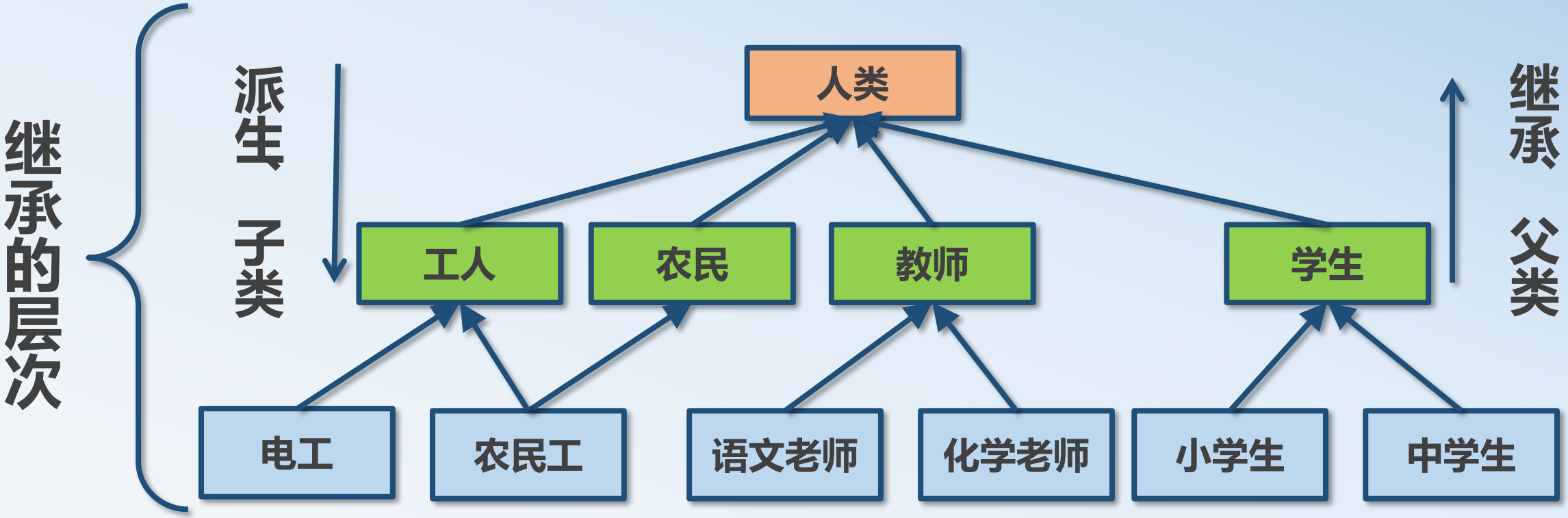
```
class Person {
public:
    void think();
    void work();
private:
    string m_strName;
    bool m_bGender;
    int m_iAge;
};
```

```
class Teacher : public Person
{
public:
    void teach();
private:
    int m_iSalary;
};
```

继承和派生的基本概念

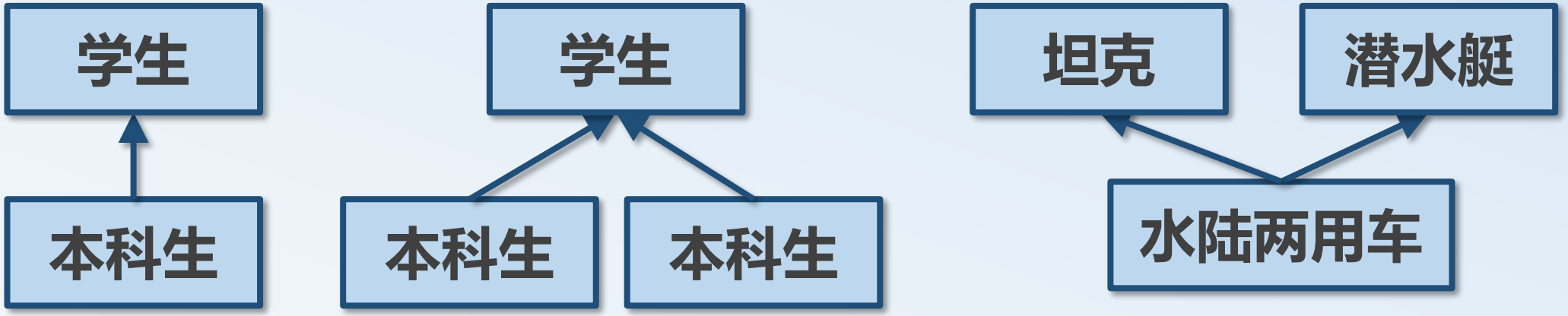
- 继承就是在一个已存在的类的基础上建立一个新的类。
  - 已存在的类称为基类(base class)或父类(father class)。
  - 新建立的类称为派生类(derived class)或子类(son class) 。
- 派生就是从已有的类(父类)产生一个新的子类。

继承和派生的基本概念



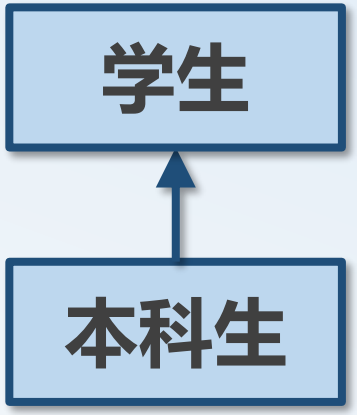
继承和派生的基本概念

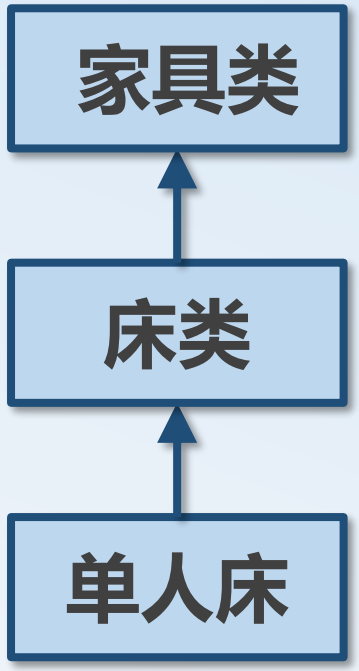
- 单重继承和多重继承
  - 单重继承:只有一个基类
  - 多重继承:有多个基类



继承和派生的基本概念

- 单级继承与多级继承
  - 单级继承:类之间的继承关系只有两层时
  - 多级继承:类之间的继承关系超过两层时





继承和派生的基本概念

- 基类与派生类的关系
  - 派生类与基类存在是一种的关系，即派生类是基类的一种（is a）
  - 派生类是基类的特化，而基类是派生类的泛化
  - 基类和派生类是相对的

1 为什么要使用继承

2 派生类的声明

3 继承方式

4 访问声明和using声明

派生类的声明

- 声明方式：
  - 单重继承声明

```
class <派生类名> :[继承方式] <基类名1>
{
    派生类新增加的成员
};
```

  - 多重继承声明

```
class <派生类名>
        :[继承方式]<基类名1> ,[继承方式]<基类名2> ,[... ...]
{
    派生类新增加的成员
};
```

派生类的声明

```
class A
{……};
class B : public A
{……};
```

单继承只有一个父类

```
class A
{……};
class B
{……};
class C :public A,public B
{……};
```

多重继承时继承方式不能被多个基类共享！

派生类的声明

- 说明
  - 有三种继承方式public,private,protected
  - 继承方式是可以省略的，如果省略则采用默认继承方式（默认继承方式取决于struct,class）
  - class声明类和struct声明类的区别

声明类的关键字	默认访问声明符	默认继承方式
struct //不推荐	public	public
class	private	private

派生类的声明

```
class Student:Undergrad
{
    int m_ival;
    void get() const;
};
```

等价

```
class Student:private Undergrad
{
private:
    int m_ival;
    void get() const;
};
```

```
struct Student:Undergrad
{
    int  m_ival;
    void get() const;
};
```

等价

```
struct Student:public Undergrad
{
public:
     int  m_ival;
     void get() const;
};
```

派生类的声明

- 派生类内只能定义派生类新增加的成员而不能定义基类成员

```
class A {     int  m_iVal; };
class B : public A {
    double  m_iVal;
};
int main(void) {
    A a;     B b;
    cout << sizeof(a) << " " << sizeof(b) << endl;
    return 0;
}
```

试图修改基类中的m_iVal从int变为double是不可能的，B中有两个m_iVal ，关于更多内容后面介绍！

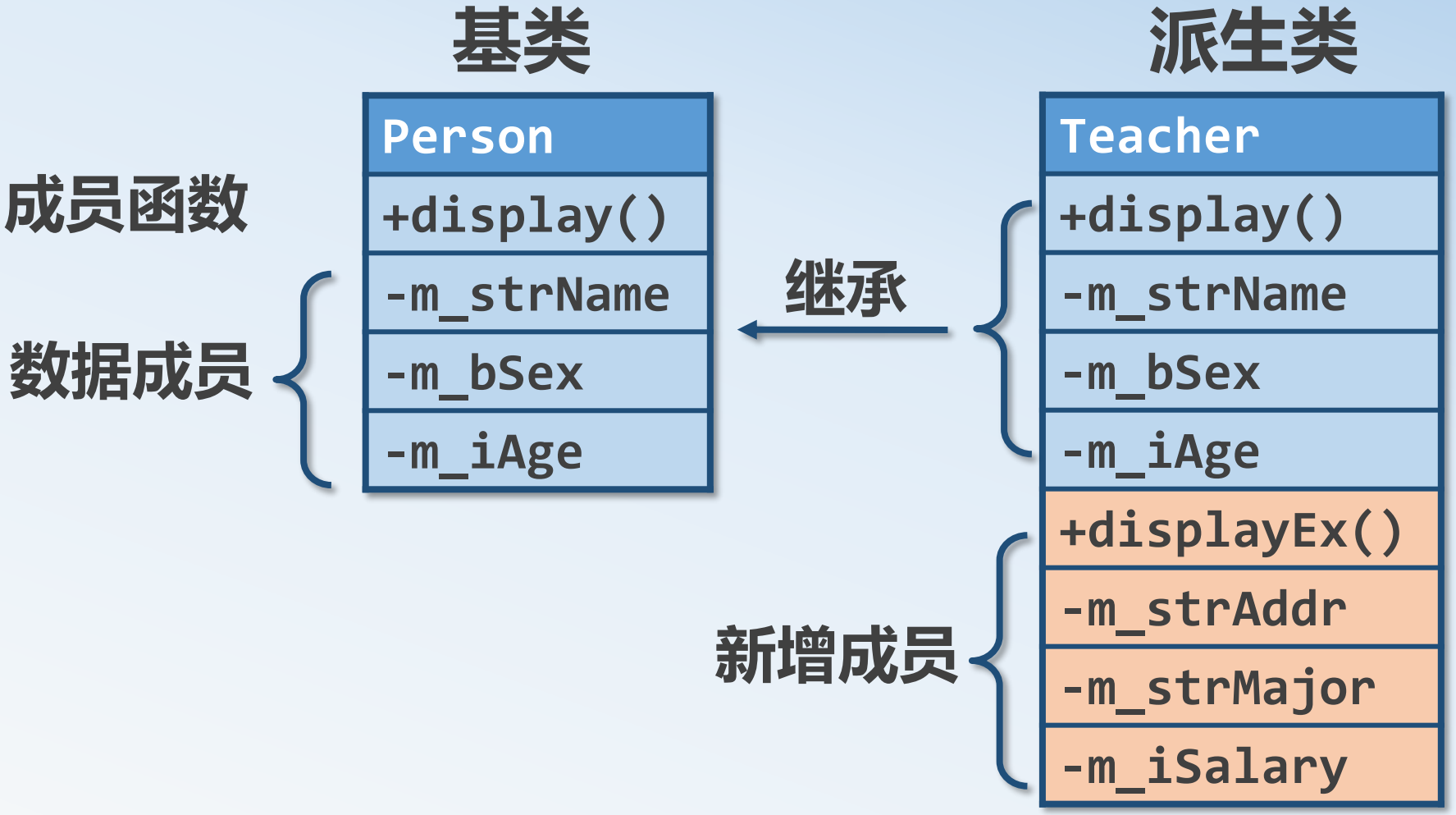
派生类的声明

```
class Person {
public:
    void display() const;
protected:
    string m_strName;
    bool m_bSex;
    int m_iAge;
};
```

```
class Teacher: public Person {
public:
    void displayEx() const;
private:
    string m_strAddr;
    string m_strMajor;
    int m_iSalary;
}
```

- 通过单继承Person类，我们实现了一个新类Teacher，那么Teacher类的结构又是怎样的呢？都有那几部分组成呢

派生类的声明


基类
派生类
Person
+display()
-m_strName
-m_bSex
-m_iAge
Teacher
+display()
-m_strName
-m_bSex
-m_iAge
+displayEx()
-m_strAddr
-m_strMajor
-m_iSalary
成员函数
数据成员
继承
新增成员

派生类的声明

```
class Person {
public:
    Person( string aName="", bool aSex=true, int aAge=0 )
        :m_strName(aName), m_bSex(aSex), m_iAge(aAge){}
    void display() const {
        cout << m_strName << endl;
        string sex;
        if( m_bSex )  sex = "male";
        else  sex = "female";
        cout << sex << endl;
        cout << m_iAge << endl;
}
private:
    string m_strName;
    bool m_bSex;
    int m_iAge;
};
```

派生类的声明

```
class Teacher: public Person {
public:
    Teacher(string aName, int aSex,       int aAge,
            string aAddr, string aMajor, int aSalary)
        :Person(aName,aSex,aAge), m_strAddr(aAddr),
         m_strMajor(aMajor),      m_iSalary(aSalary)
    {}
    void displayEx() const{
        display();
        cout << m_strAddr << endl;
        cout << m_strMajor << endl;
        cout << m_iSalary << endl;
    }
protected:
    string m_strAddr;
    string m_strMajor;
    int m_iSalary;
};
```

派生类的声明

- 继承过程
  - 吸收基类成员
    - 派生类把基类全部的成员(不包括构造函数和析构函数)接收过来。
  - 调整基类成员
    - 调整基类public和protected成员的访问权限
    - 隐藏同名基类成员
  - 增加子类成员
    - 这部分内容是很重要的，是派生类对基类功能的扩展。要根据需要仔细考虑应当增加哪些成员。在声明派生类时，一般还应当自己定义派生类的构造函数和析构函数，因为构造函数和析构函数是不能从基类继承的。

1 为什么要使用继承

2 派生类的声明

3 继承方式

4 访问声明和using声明

继承方式

- 继承方式
  - 公有继承(public)
  - 保护继承(protected)
  - 私有继承(private)
- 不同继承方式的影响主要体现在：
  - 派生类对基类成员的访问控制
  - 派生类对象对基类成员的访问控制

继承方式

- 类成员的访问属性

访问声明符	本类成员函数	类作用域外	友元
private	Y	N	Y
protected	Y	N	Y
public	Y	Y	Y

继承方式 – 公有继承

- 如果以公有方式继承则基类中的public和protected在派生类中保持不变，私有成员在派生类中为不可访问(inaccessible)

基类成员访问控制	公有继承（public）	
	派生类成员访问属性	类外访问属性
public	public	public
protected	protected	inaccessible
private	inaccessible	inaccessible

- 说明：

inaccessible成员不能被子类成员函数直接访问，但可通过基类成员函数间接访问

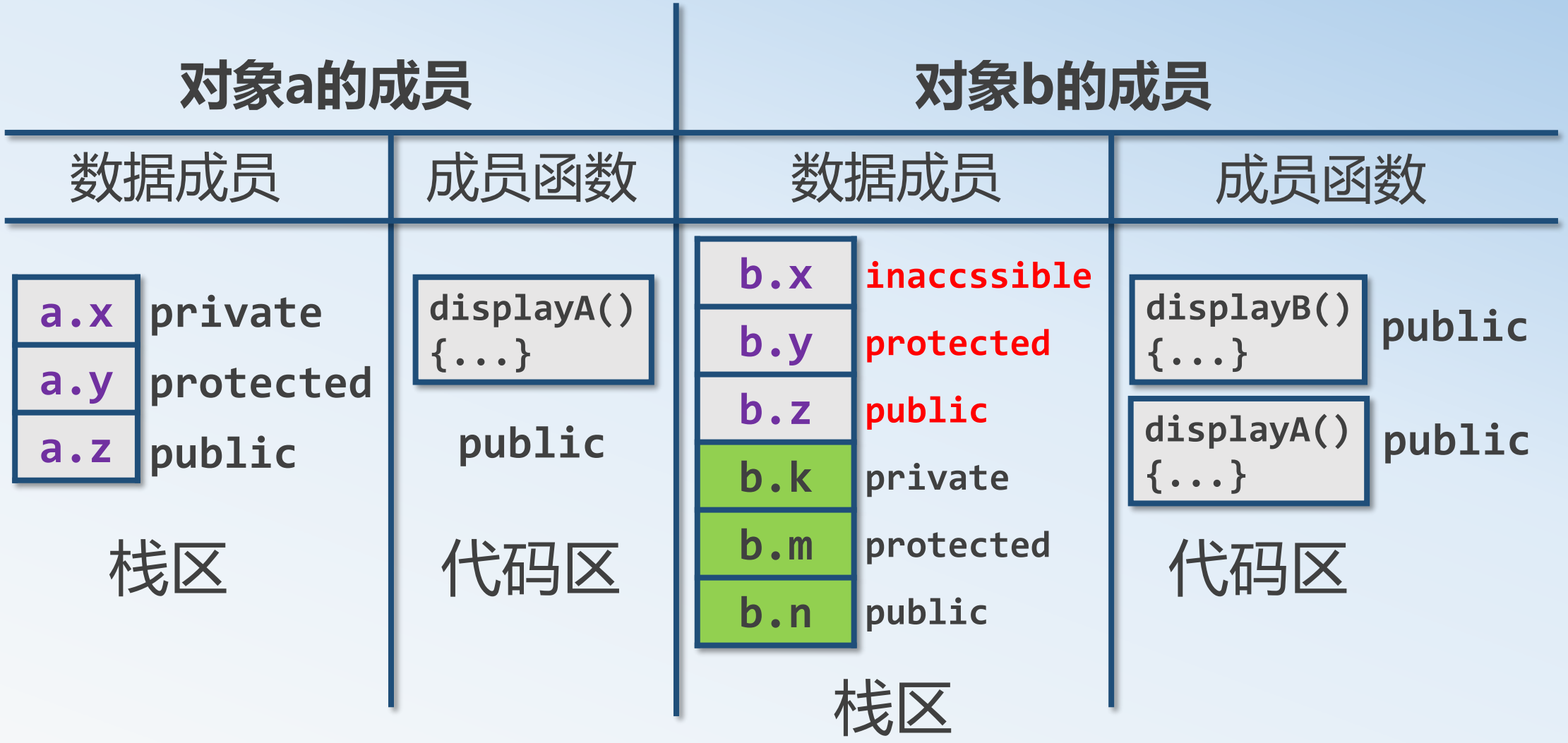
继承方式 – 公有继承

```
class A {
private:
    int x;
protected:
    int y;
public:
    int z;
    void displayA() {
        cout << x
             << y
             << z
             << endl;
    }
};
```

```
class B:public A {
private:
    int k;
protected:
    int m;
public:
    int n;
    void displayB() {
        cout << k
             << m
             << n
             <<endl;
      displayA();
    }
};
```

```
int main(void) {
    A  a;
    a.displayA();
    B  b;
    b.displayB();
    cout << b.x
         << b.y
         << b.z
         << endl;
    cout << b.k
         << b.m
         << b.n
         << endl;
}
```

继承方式 – 公有继承


对象a的成员
对象b的成员
数据成员
成员函数
数据成员
成员函数
a.x
private
a.y
protected
a.z
public
displayA()
{...}
public
b.x
inaccssible
b.y
protected
b.z
public
b.k
private
b.m
protected
b.n
public
displayB()
{...}
public
displayA()
{...}
public
栈区
代码区
栈区
代码区

继承方式 – 公有继承

- 公有继承的特点：

  public继承保持了基类中protected和public属性不变，最大限度的保持了基类的原态，在实际开发中常常使用这种继承。

继承方式 – 保护继承

- Protected访问控制符限定基类成员只能被派生类成员访问，类外不能访问保护成员。

基类成员访问控制	保护继承（protected）	
	派生类成员访问属性	类外访问属性
public	protected	inaccessible
protected	protected	inaccessible
private	inaccessible	inaccessible

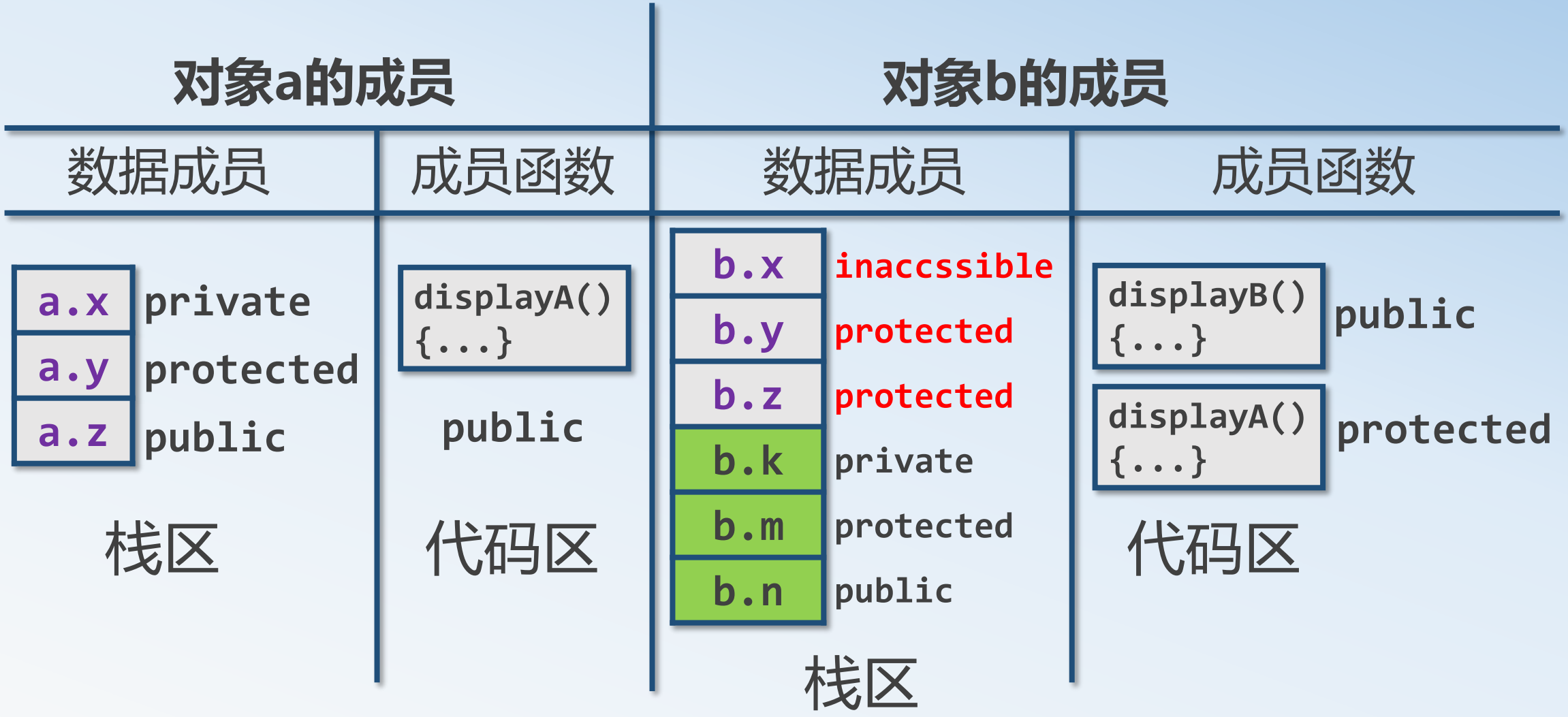
继承方式 – 保护继承

```
class A {
private:
    int x;
protected:
    int y;
public:
    int z;
    void displayA() {
        cout << x
             << y
             << z
             << endl;
    }
};
```

```
class B:protected A {
private:
    int k;
protected:
    int m;
public:
    int n;
    void displayB() {
        cout << k
             << m
             << n
             <<endl;
      displayA();
    }
};
```

```
int main(void) {
    B  b;
    //b.displayA();
    // protected
    b.displayB( );
    cout << b.x
         << b.y
         << b.z
         << endl;
    cout << b.k
         << b.m
         << b.n
         << endl;
}
```

继承方式 – 保护继承


对象a的成员
对象b的成员
数据成员
成员函数
数据成员
成员函数
a.x
private
a.y
protected
a.z
public
displayA()
{...}
public
b.x
inaccssible
b.y
protected
b.z
protected
b.k
private
b.m
protected
b.n
public
displayB()
{...}
public
displayA()
{...}
protected
栈区
代码区
栈区
代码区

继承方式 – 保护继承

- 保护继承的特点：

protected继承使基类中protected和public属性都变为protected，如果只是希望继承基类而不希望在派生类外访问,这时推荐protected继承。

继承方式 – 私有继承

- 如果采用私有继承，则基类的private成员在派生类中为inaccessble,基类的pubic和protected成员在派生类中均被继承且均为private

基类成员访问控制	私有继承（private）	
	派生类成员访问属性	类外访问属性
public	private	inaccessible
protected	private	inaccessible
private	inaccessible	inaccessible

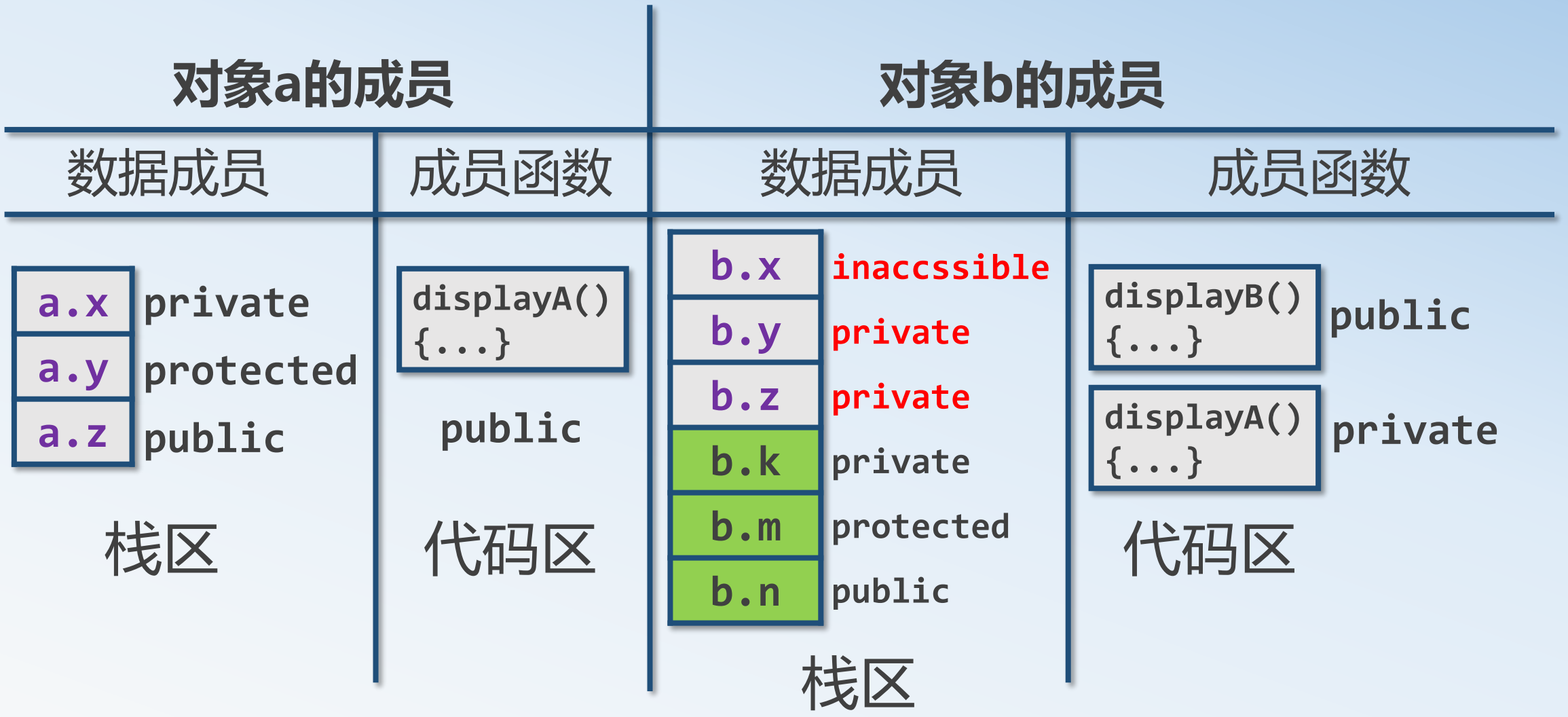
继承方式 – 私有继承

```
class A {
private:
    int x;
protected:
    int y;
public:
    int z;
    void displayA() {
        cout << x
             << y
             << z
             << endl;
    }
};
```

```
class B:private A {
private:
    int k;
protected:
    int m;
public:
    int n;
    void displayB() {
      cout << k << m
        << n << endl;
      cout << x << y
        << z << endl;
    }
};
```

```
int main(void) {
    B  b;
    //b.displayA();
    // private
    b.displayB( );
    cout << b.x
         << b.y
         << b.z
         << endl;
    cout << b.k
         << b.m
         << b.n
         << endl;
}
```

继承方式 – 私有继承


对象a的成员
对象b的成员
数据成员
成员函数
数据成员
成员函数
a.x
private
a.y
protected
a.z
public
displayA()
{...}
public
b.x
inaccssible
b.y
private
b.z
private
b.k
private
b.m
protected
b.n
public
displayB()
{...}
public
displayA()
{...}
private
栈区
代码区
栈区
代码区

继承方式 – 私有继承

- 私有继承的特点：

  private继承使基类中protected和public属性都变为private，实际上私有继承很少使用！

继承方式 – 私有继承和组合

- 组合表示的是“has -a”关系

```
class Engine
{
};
class Car
{
private:
    Engine  m_Engine; // 组合
};
```

- Car包含了一个Engine子对象，表示Car有一个引擎部件。

继承方式 – 私有继承和组合

- 私有继承无法表示”is a”关系，私有继承和组合类似，通过继承在Car里包含了所有Engine的成员，相当于包含了以Engine匿名对象。

```
class Engine
{
};
class Car : private Engine
{
};
```

继承方式 – 私有继承和组合

- 私有继承和组合的区别
  - 组合方式，在Car里，只能通过m_Engine对象访问Engine的成员，私有继承可以访问Engine的public和protected成员
  - 在私有继承方式下会发生同名隐藏或覆盖，组合方式不会出现类似问题。

继承方式(总结)

- 三种继承方式的总结：

① 公有继承方式下，基类的protected和public成员在派生类中保持原访问属性

② 在保护继承方式下，基类的protected和public成员在派生类中的访问属性均为protected

继承方式(总结)

③ 在私有继承方式下，基类的protected和public成员在派生类中均为private，下面以表的形式再总结一遍

④ 基类的private成员无论何种继承方式在派生类中均不可直接访问（inaccessible）

⑤ 派生类无法直接访问inaccessible成员，只能通过继承来的基类成员函数间接访问！

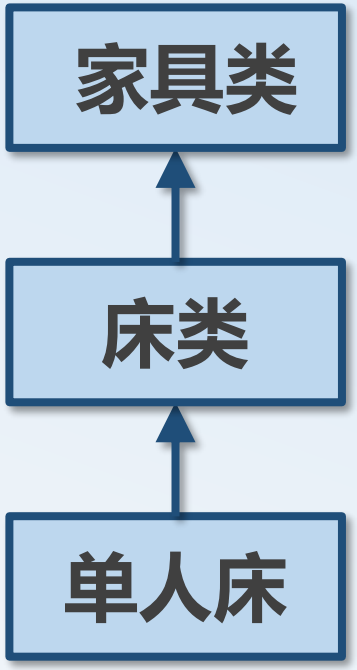
继承方式(总结)

派生类成员		派生类内访问		类外访问
		派生类成员函数	基类成员函数	派生类对象 对象指针 对象引用
新增成员	public	√	×	√
	protected	√	×	×
	private	√	×	×
基类成员	public	√	√	√
	protected	√	√	×
	private	√	√	×
	inaccessble	×	√	×

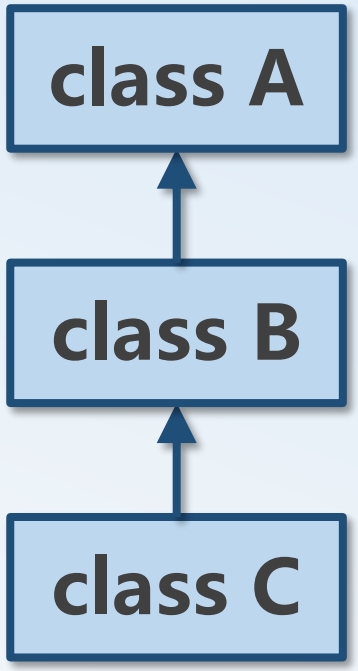
继承方式(总结)

[思考]

多级继承下成员的访问属性是怎样的呢？



继承方式(总结)



继承方式	class A的成员	class B	class C
public	private	inaccessible	inaccessible
	protected	protected	protected
	public	public	public
protected	private	inaccessible	inaccessible
	protected	protected	protected
	public	protected	protected
private	private	inaccessible	inaccessible
	protected	private	inaccessible
	public	private	inaccessible

单级继承下私有继承和保护继承的区别无法体现

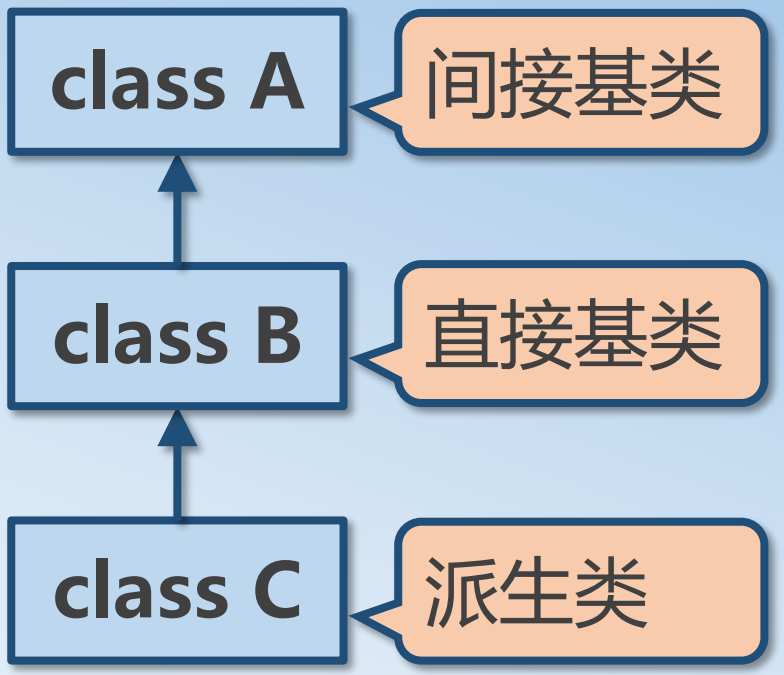
多级继承下私有继承和保护继承的区别显而易见

继承方式(总结)

[注意] 多级继承下，当前类仅会继承直接基类的成员，不会继承间接基类的成员。

```
class A {
private: int a;
};
class B : public A {
protectd: int b;
};
class C : public B {
public:
    void display()
    {cout <<a <<b <<c << endl; }
private: int c;
};
```

C不会从A继承任何成员！



类	成员	属性
A	a	private
B	A::a	inaccessible
	b	protected
C	A::a	inaccessible
	B::b	protected
	c	private

1 为什么要使用继承

2 派生类的声明

3 继承方式

4 访问声明和using声明

访问声明和using声明

- 格式：

```
using 基类名::成员名
基类名::成员名
```

- 含义：将继承来的非inaccessible成员变为任意属性！
- 位置：在派生类中任意访问属性中。

访问声明和using声明

```
class A {
private:    int x
protected:  void test();
public :    int y; int z;
};
class B : private A {
private:    using A::x; // 仅针对非inaccessible
protected:  using A::test(); // 应该是A::test
            using A::y  // Right! 改变属性
public:     using A::z  // Right  保持属性
};
```

访问声明和using声明

[注意]

- 格式中只能是成员名不包括成员的其他部分
- 仅针对非inaccessible成员
- 既能只能保持也能改变属性
- 对基类的重载函数均有效 (慎重)
- 本质上说就是基类作用域中名字的引用

本讲教学目标

- 理解C++中继承的概念
- 掌握C++中如何使用继承
- 理解C++中的三种继承方式


THANKS