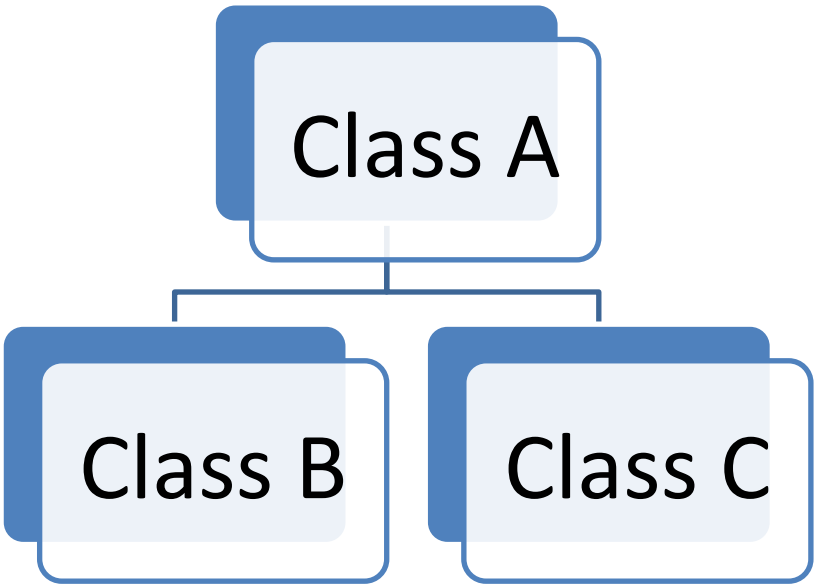
Множественное наследование

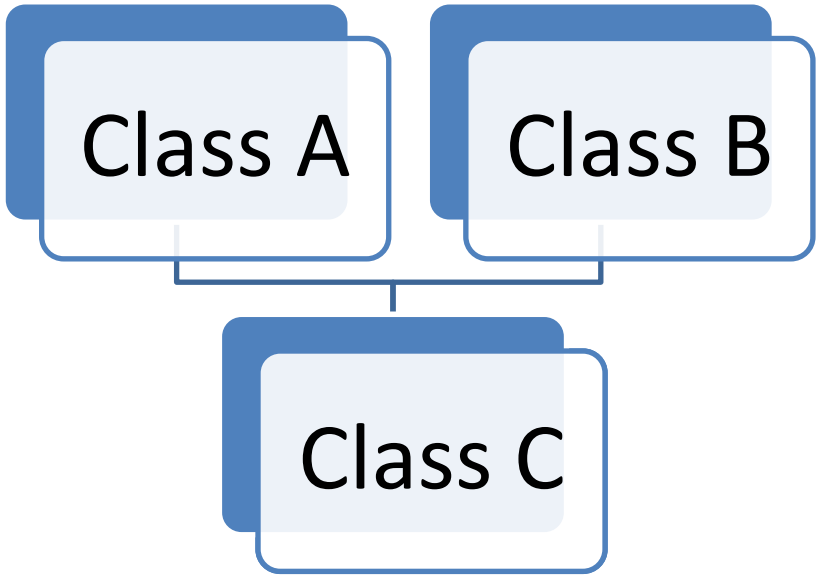
4 июля 2017 г.

Типы наследования по числу базовых классов

одиночное

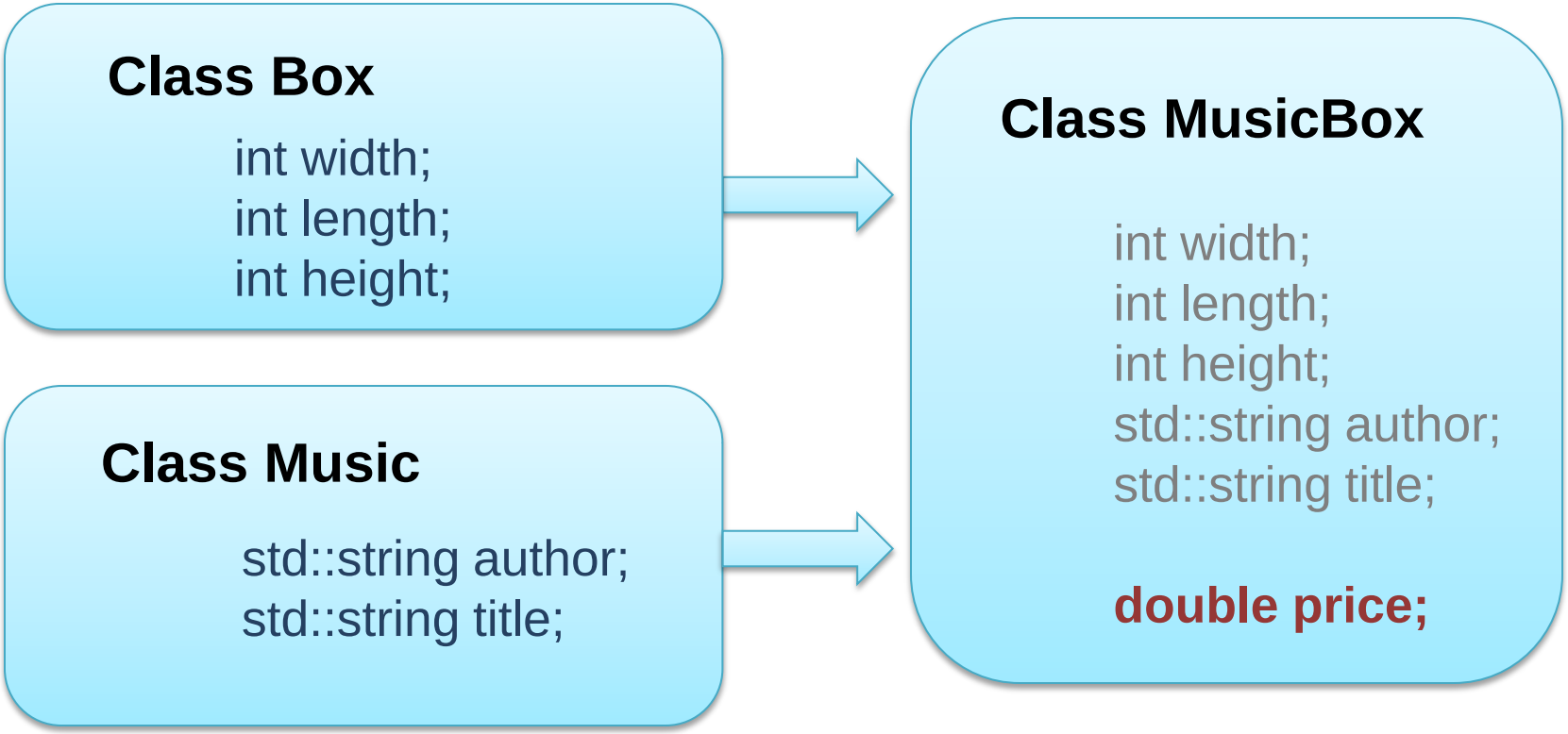


множественное



Множественное наследование

Наличие у производного класса нескольких базовых классов (наследуются поля и методы всех базовых классов)



Множественное наследование

```
class Derived : public Base1, private Base2 { };
```

- базовые классы надо объявить выше
- разные базовые классы могут иметь разные спецификаторы доступа
- базовые классы не должны повторяться
- конструкторы базовых классов вызываются в порядке объявления в списке (Base1 -> Base2), а удаляются – в обратном порядке

```
class A {
   int a;
public:
   A(int aa) : a(aa) { std::cout << "A - constr" << std::endl; }
};
class B {
   int b;
public:
   B(int bb) : b(bb) { std::cout << "B - constr" << std::endl; }
};
class C : public B, public A {
   int c;
public:
C(int aa, int bb, int cc) : A(aa), B(bb), c(cc) {    // B -> A -> C
    std::cout << "C - constr" << std::endl; }
};
void f() { C c(1, 2, 3); }
```

Проблема 1. Конфликт имен

В разных базовых классах есть члены с одинаковым именем

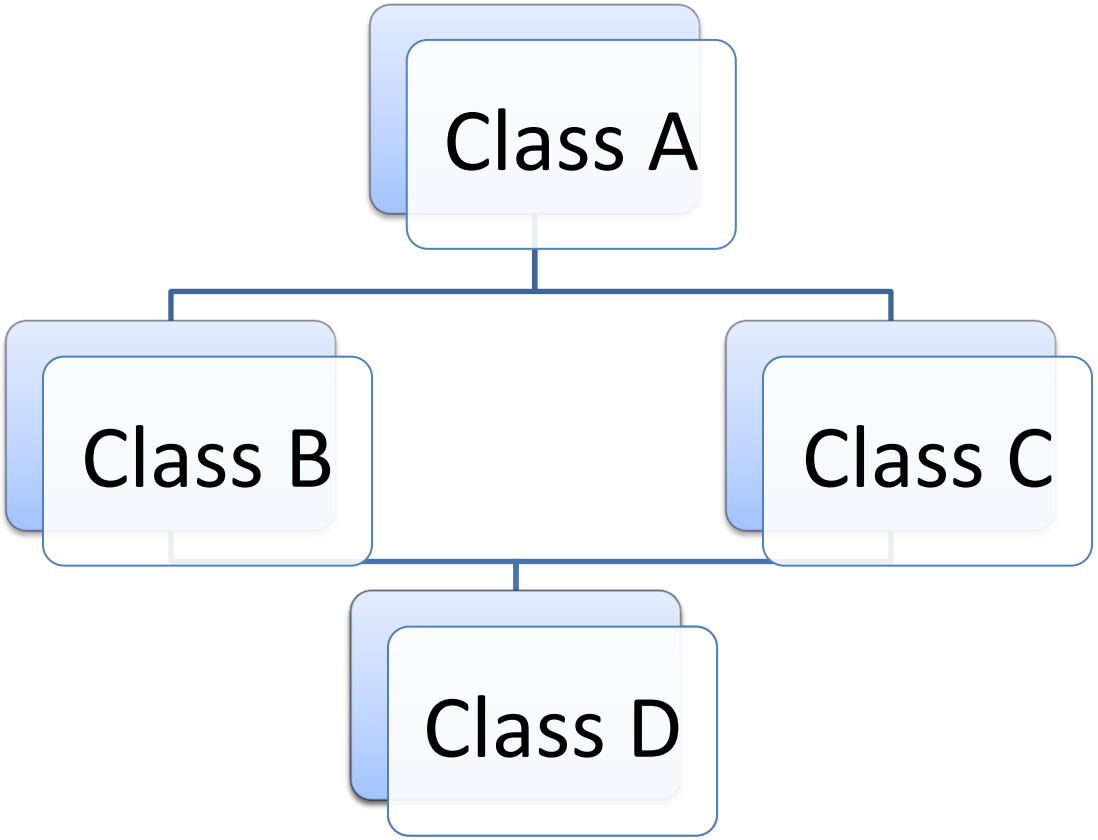
```
class A {
public:
   void f();
};
class B {
public:
   void f();
};
class C : public A, public B {
    void g() { f(); }    // Неоднозначность. Какое из двух f()?
};
```

Решение

Разрешение доступа (явное указание спецификатора базового класса)

```
class A {
public:
    void f();
};
class B {
public:
    void f();
};
class C : public A, public B {
    void g() { A::f(); }          // Или B::f()
};
```

Проблема 2. Проблема ромба

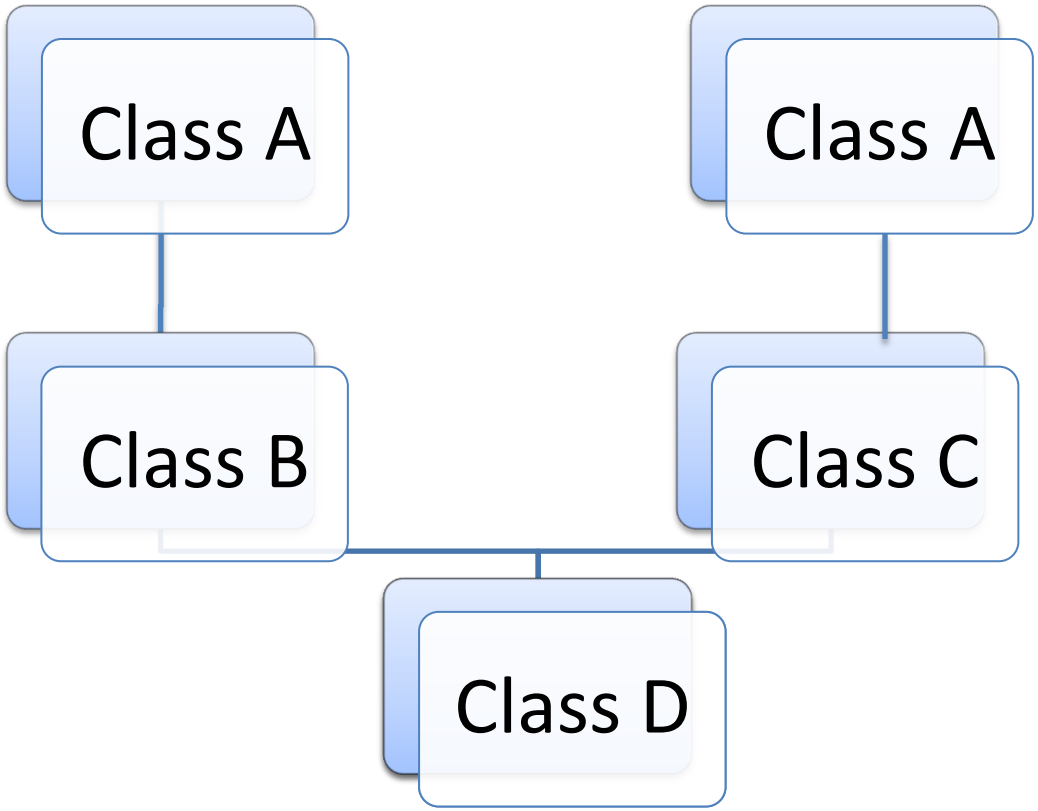


Проект иерархии классов

```
class A {
   int a;
public:
   A(int aa) : a(aa) { std::cout << "A - constr" << std::endl; }
};
class B : public A {
   int b;
public:
   B(int aa, int bb) : A(aa), b(bb) { std::cout << "B - constr" << std::endl; }
};
class C : public A {
   int c;
public:
   C(int aa, int cc) : A(aa), c(cc) { std::cout << "C - constr" << std::endl; }
};

class D : public B, public C {
   int d;
public:
   D(int aa, int bb, int cc, int dd)
      : B(aa, bb), C(aa, cc), d(dd) { std::cout << "D - constr" << std::endl; }
};
                                        void f() { D obj(1, 2, 3, 4); }
```

А вот что делает компилятор…



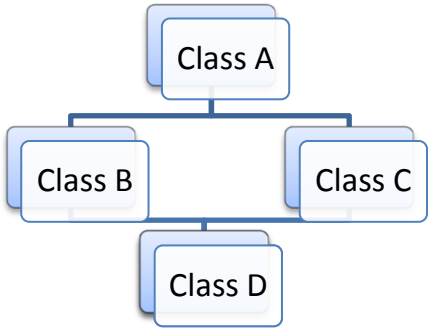

```
class D : public B, public C { … };
```

Поля класса A будут дважды включены в класс D

Решение

- используется виртуальное наследование
- при наследовании используется ключевое слово virtual (виртуальный базовый класс)
- есть одна копия общего базового класса
- в общем базовом классе должен быть конструктор по умолчанию (вызывается первым)

```
class A { };
class B : public virtual A { };
class C : public virtual A { };
class D : private B, protected C { };
```



```
class A {
   int a;
public:
   A() { std::cout << "A - default constr" << std::endl; }
   A(int aa) : a(aa) { std::cout << "A - constr" << std::endl; }
};
class B : public virtual A {
   int b;
public:
   B(int aa, int bb) : A(aa), b(bb) { std::cout << "B - constr" << std::endl; }
};
class C : public virtual A {
   int c;
public:
   C(int aa, int cc) : A(aa), c(cc) { std::cout << "C - constr" << std::endl; }
};

class D : public B, public C {
   int d;
public:
   D(int aa, int bb, int cc, int dd)
      : B(aa, bb), C(aa, cc), d(dd) { std::cout << "D - constr" << std::endl; }
};
```

Вопросы?