ВикипедиЯ

Виртуальное наследование

Материал из Википедии — свободной энциклопедии

Про наследование виртуальных методов, см виртуальный метод.

Виртуа́льное насле́дование (англ. *virtual inheritance*) в языке программирования C++ — один из вариантов наследования, который нужен для решения некоторых проблем, порождаемых наличием возможности множественного наследования (особенно «ромбовидного наследования»), путём разрешения неоднозначности того, методы которого из суперклассов (непосредственных классов-предков) необходимо использовать. Оно применяется в тех случаях, когда множественное наследование вместо предполагаемой полной композиции свойств классов-предков приводит к ограничению доступных наследуемых свойств вследствие неоднозначности. Базовый класс, наследуемый множественно, определяется виртуальным с помощью ключевого слова `virtual`.

Содержание

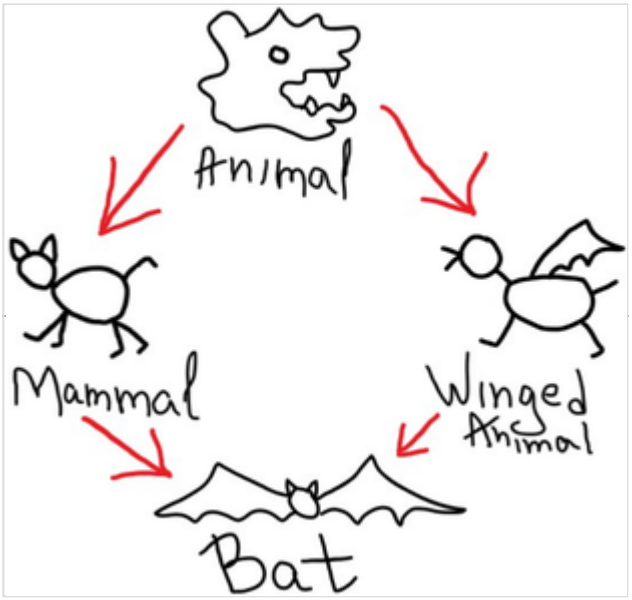
Суть проблемы

Рассмотрим следующую иерархию классов:

```
class Animal
{
 public:
  virtual void eat();  // Метод определяется для данного класса
 ...
};

class Mammal : public Animal
{
 public:
  Color getHairColor();
 ...
};

class WingedAnimal : public Animal
{
 public:
  void flap();
 ...
};

// A bat is a winged mammal
```

```
class Bat : public Mammal, public WingedAnimal {};   //<--- обратите внимание, что метод eat() не переопределен в Bat

Bat bat;
```

Для вышеприведенного кода вызов `bat.eat()` является неоднозначным. Он может относиться как к `Bat::WingedAnimal::Animal::eat()` так и к `Bat::Mammal::Animal::eat()`. У каждого промежуточного наследника (`WingedAnimal`, `Mammal`) метод `eat()` может быть переопределен (это не меняет сущность проблемы с точки зрения языка). Проблема в том, что семантика традиционного множественного наследования не соответствует моделируемой им реальности. В некотором смысле, сущность `Animal` единственна по сути; `Bat` — это `Mammal` и `WingedAnimal`, но свойство животности (`Animal`ness) летучей мыши (`Bat`), оно же свойство животности млекопитающего (`Mammal`) и оно же свойство животности `WingedAnimal` — по сути это одно и то же свойство.



Такая ситуация обычно именуется «ромбовидным наследованием» и представляет собой проблему, которую призвано решить виртуальное наследование.

Представление класса

Прежде чем продолжить, полезным будет рассмотреть, как классы представляются в C++. В частности, при наследовании классы предка и наследника просто помещаются в памяти друг за другом. Таким образом объект класса Bat это на самом деле последовательность объектов классов (Animal,Mammal,Animal,WingedAnimal,Bat), размещенных последовательно в памяти, при этом Animal повторяется дважды, что и приводит к неоднозначности.

Решение

Мы можем переопределить наши классы следующим образом:

```
class Animal
{
 public:
  virtual void eat();
 ...
};

// Two classes virtually inheriting Animal:
class Mammal : public virtual Animal                // <--- обратите внимание на ключевое слово virtual
{
 public:
  Color getHairColor();
 ...
};

class WingedAnimal : public virtual Animal          // <--- обратите внимание на ключевое слово virtual
{
 public:
  void flap();
 ...
};
```

```
// A bat is still a winged mammal
class Bat : public Mammal, public WingedAnimal {};
```

Теперь, часть `Animal` объекта класса `Bat::WingedAnimal` *та же самая*, что и часть `Animal`, которая используется в `Bat::Mammal`, и можно сказать, что `Bat` имеет в своем представлении только одну часть `Animal` и вызов `Bat::eat()` становится однозначным.

Виртуальное наследование реализуется через добавление указателей в классы `Mammal` и `WingedAnimal`. Таким образом, `Bat` представляется, как (ptr, Mammal, ptr, WingedAnimal, Bat, Animal). *ptr содержит информацию о смещении в памяти между началом `Mammal` и его `Animal`. Это необходимо, потому что для указателя Animal* p;, который может ссылаться на Animal, на Mammal, на WingedAnimal, смещение в памяти между началом объекта и его Animal части неизвестно на этапе компиляции, а выясняется только во время выполнения.

Пример

Чтобы понять суть виртуального наследования без лишнего "шума", следует рассмотреть следующий пример:

```
#include <iostream>

class A {
public:
    virtual int foo() {
        return 1;
    }
};

class B : public virtual A {};

class C : public virtual A {};

class D : public B, public C {};

int main () {
    D d;
    std::cout << d.foo();
    return 0;
}
```

Если убрать ключевое слово *virtual*, то метод *foo()* не может быть определён однозначно и в результате не будет доступен как объект класса *D* и код не скомпилируется.

См. также

- Объектно-ориентированное программирование
- Наследование

Литература

- *Подбельский В. В.* Глава 10.2 Множественное наследование и виртуальные базовые классы // Язык Си++ / рец. Дадаев Ю. Г.. — 4. — М.: Финансы и статистика, 2003. — С. 336-359. — 560 с. — ISBN 5-279-02204-7, УДК 004.438Си(075.8) ББК 32.973.26-018 1я173.