ОСОБЕННОСТИ ПРИМЕНЕНИЯ МНОЖЕСТВЕННОГО НАСЛЕДОВАНИЯ В C++

Г.С. Степович-Цветкова

Ивановский государственный университет

Язык программирования C++, являясь объектно-ориентированным, поддерживает возможность множественного наследования, когда класс является потомком сразу нескольких классов. С одной стороны, множественное наследование представляет собой мощный инструмент языка, позволяющий создавать комплексные типы данных, базирующиеся на различных свойствах объектов. Но с другой стороны, в процессе реализации такого варианта наследования могут возникнуть определенные трудности.

Первой возможной проблемой при работе с множественным наследованием является конфликт имен, возникающий в случае, если в базовых классах существуют поля данных или методы с одинаковым названием. Такая неоднозначная ситуация решается явным указанием того, к какому из базовых классов необходимо обратиться, с помощью операции расширения видимости.

Вторая проблема, так называемая проблема ромба, – это случай многократного включения базового класса, поскольку базовые классы одного класса в свою очередь наследуют от одного и того же косвенного базового класса. При таком описании классов объект класса D будет содержать объект базового класса A дважды.

```
class A
{
. . .
};
class B : public A
{
. . .
};
class C : public A
{
. . .
};
class D : public B, public C
{
};
```

Причина многократного включения объекта базового класса заключается в представлении классов в C++. При наследовании базовые классы и классы-потомки помещаются в памяти непосредственно друг за другом. И поэтому объект производного класса представляет собой последовательность объектов родительских классов. Таким образом, объект класса D в памяти является следующей последовательностью объектов: (A, B, A, C, D). Двойное копирование объекта класса A приводит к неоднозначности и нерациональному использова-

нию памяти компьютера, когда в действительности требуется присутствие объекта класса A лишь однажды.

Однако, возможны случаи, когда дублирование объекта базового класса оправдано. Например, в качестве класса A рассмотрим класс Пользователь, содержащий поля данных Логин и Пароль и предназначенный для создания учетных записей сотрудников некоторой организации. Потомки класса A: класс B – Сотрудник, имеющий доступ к базе данных организации; класс C – Сотрудник, имеющий доступ к почтовому серверу. Класс D наследует от классов B и C и является классом Сотрудник, имеющий доступ к базе данных и почтовому серверу. При этом одному и тому же сотруднику могут понадобиться различные идентифицирующие данные для получении доступа к базе данных и для доступа к почтовому серверу.

В случае если двойное копирование объекта косвенного базового класса все же не требуется, то существует механизм виртуального наследования, позволяющий в объект класса D поместить ровно одну копию объекта класса A. Для этого в нашем примере класс A необходимо объявить виртуальным базовым классом для классов B и C, тогда часть A объекта класса B будет совпадать с частью A объекта класса C. Механизм виртуального наследования функционирует путем добавления в классы B и C указателей на виртуальную таблицу vtable, и представление объекта класса D имеет следующий вид (vtable*, B, vtable*, C, D, A). Ценой отсутствия многозначности и единственности объекта A является увеличение размера объекта класса D на величину двух указателей.

Таким образом, работа с множественным наследованием требует особого внимания в случае возникновения конфликта имен полей данных или методов, а также в случае ромбовидного наследования.

Список литературы:

1. Виртуальное наследование. URL: http://ru.wikipedia.org/wiki/Виртуальное_наследование (дата обращения: 10.06.2013г.).

2. Труб И. О проблемах множественного наследования. URL: http://www.osp.ru/os/2001/02/179920/ (дата обращения: 10.06.2013г.).