09장 자료 추상화와 객체지향

9.8 연습문제

9.19 클래스 D가 클래스 A, B와 C를 상속하고 이 셋은 공통의 조상을 가지고 있지 않다고 가정하자. D의 자료 멤버와 vtable를 어떻게 메모리에 배치하는지를 보여라. 또한 D로의 참조를 A, B, C 객체로의 참조로 변환시키는 방법을 보여라.

9.20 ©(심화학습에 있는) 예 9.55에서 설명한 person_interface와 list_node_interface 클래스를 살펴보라. student를 person_interface와 list_node_interface에서 유도한다면 아래 메소드 호출에서 어떤 일이 발생할지를 설명하라.

```
student s;
person *p = &s;
...
p.debug_print();
```

일어나는 메소드 검색과 계산된 관점을 설명하기 위해 student 객체의 표현의 그림을 사용하기 바랄 수도 있다. 공유 상속 없이 그림 9.8과 유사한 구현을 가정할 수 있다.

9.21 ©(심화학습에 있는) 예 9.56의 상속 트리가 주어졌을 때 클래스 student_prof의 객체에 대한 표현을 보여라. ©(심화학습에 있는) 그림 9.7, 9.8과 9.9에서 도움을 받을 수 있을 것이다.

9.22 ©(심화학습에 있는) 그림 9.7의 메모리 배치와 다음 선언이 주어졌을 때

```
student& sr;
gp_list_node& nr;
```

다음과 같은 대입

```
nr = sr;
```

에 대해 생성해야 하는 코드를 보여라(함정: `nil` 포인터를 고려해야 함).

9.23 혼합 상속을 사용하는 자바와 같은 언어로 작성된 다음 코드를 살펴보자.

```
class A {
   private int a;
   ...
}
class B extends A implements Runnable {
   private int b;
   ...
}
...
Runnable r = new B();
```

(a) `B` 객체에 대한 구현의 도면을 그리고 항목(자료 멤버)과 vtable을 보여라.

(b) `r.run()`(호출자 쪽만)에 대한 호출 순서를 (의사 어셈블리어로) 보여라. `r`을 레지스터 `r1`에 불러왔고 `Runnable` 인터페이스가 정의하는 메소드임을 가정할 수 있다. 필요한 만큼의 레지스터를 사용할 수 있다. `r1`을 보존할 필요가 없다. `this` 매개변수를 레지스터 `r2`에 넣어 전달해야 함을 가정할 수 있다.

9.24 표준 C++은 클래스에 대해 "멤버로의 포인터" 방법을 제공한다.

```
class C {
public:
   int a;
   int b;
} c;
int C::*pm = &C::a;
   // pm은 (임의의) C 객체의 멤버 a을 가리킴
...
c->*pm = 3;          // c.a에 3을 대입한다.
```

가상 메소드를 포함해 서브루틴 멤버(메소드)에 대해서도 멤버로의 포인터를 허용한다. C++ 양식의 다중 상속에서 가상 메소드에 대한 포인터를 어떻게 구현할 것인가?

9.25 다중 상속을 사용하는 한 언어의 vtable 항목에서 (메소드 주소, `this` 보정) 순서쌍을 사용하는 대신 그 항목을 간단한 포인터로서 남겨둘 수 있지만 `this`를 인라인으로 갱신하는 코드를 가리키게 할 수도 있다. 이 기법 하에서 실행될 명령어를 보여라. (심화학습에 있는) 예 9.53에서 보여지는 코드보다 더 빠를지 혹은 더 느릴지에 영향을 주는 요소는 무엇인가? 어떤 기법이 공간을 덜 차지하는가? (코드 크기와 자료 구조 크기 모두를 계산하는 것을 기억하고 모든 호출 지점마다 어떤 명령어를 중복시켜야 함을 고려하라)

실행 가능한 코드를 자료 구조로 대체하는 것을 더 선호한다면 vtable 자체가 실행 가능한 코드로 구성된 구현을 고려하라. 이 코드가 어떻게 생겼는지를 보이고 시간과 공간 상의 오버헤드에 대한 관계도 논의하라.

9.26 에펠에서 공유 상속은 그 예외가 아닌 기본이다. 개명된 특징만이 중복될 수 있다. 결과적으로 어떤 클래스에서 유도된 클래스가 그 멤버를 중복 상속했는지 혹은 공유 상속했는지를 볼 수 있는 시점을 말하는 것은 가능하지 않다. 중복해 동작할 것인지 혹은 공유해 동작할 기본 클래스에서 상속된 멤버를 검색 위한 정규 방법을 설명하라(힌트: 7.4.3절에서 설명했던 동적 형태의 배열을 포함하는 레코드에 대해 도프 벡터의 사용을 고려하라. 좀 더 자세한 사항에 대해서는 빌헬름과 마우러의 컴파일러 문서[WM95, Sec. 5.3]를 참고하라).

9.27 (심화학습에 있는) 그림 9.9에서 `A`에 선언됐지만 `B`, `C`, `D` 객체 관점에서 호출하는 가상 메소드에 대한 호출을 생각하라. `D`/`B`로의 vtable의 `A` 부분 사본과 그 vtable의 `C` 부분을 (적절하게 조정된 `this` 보정을 사용해) 덧붙임으로써 간접 접근의 한 수준을 피할 수 있다. 대안 구현에 대한 호출 순서를 제시하라. 최악의 경우에 그 vtable은 n개의 조상을 가진 클래스에 대해 얼마나 더 큰 공간을 차지하는가?

9.28 루비에서 인터페이스(혼합)은 메소드 서명(이름과 매개변수 목록)뿐만 아니라 메소드 코드를 제공할 수 있다(그것은 자료 멤버를 제공하지 않지만 다중 상속일 것이다). 이 특징은 자바에서 이해가 되겠는가? 설명하라.

9.29 (심화학습에 있는) 9.6.1절의 끝에 제시된 유클리드 알고리즘의 스몰토크 구현을 고려하라. `4 gcd: 6`의 계산에 포함된 메시지를 추적하라.