08장 서브루틴과 제어 추상화

【8.3.2】 이름에 의한 호출

이름에 의한 호출은 6.6.2절에 설명한 정규 순서 인자 값 계산을 구현한다. 호출자의 참조 환경에서 이름에 의한 호출의 매개변수를 사용할 때마다 다시 계산한다. 그 효과는 호출된 루틴을 호출의 시점에서 텍스트적으로 확장하는 것처럼 형식 매개변수의 모든 출현을 (복잡한 수식일 수도 있는) 실제 매개변수로 대체하는 것이다. 매크로 매개변수와 관련된 보통의 문제를 피하기 위해 문법적으로 유효한 경우에는 언제나 대체될 매개변수 주위의 괄호를 포함하고 어떤 형식 매개변수 이름이나 같은 이름을 쓰는 지역 식별자의 "적절히 조직적인 변경"하게 해 의미가 충돌하지 않게 "확장"을 정의한다[NBB+63, p. 12]. 이름에 의한 호출은 알골 60에서 기본이다. 값에 의한 호출은 대안으로 이용할 수 있다. 시뮬라에서 값에 의한 호출은 기본이다. 이름에 의한 호출은 대안이다.

예 8.64
얀센의 장치

이름에 의한 호출을 구현하기 위해 알골 60은 호출자의 참조 환경에서 실제 매개변수를 계산하는 숨겨진 서브루틴을 전달한다. 숨겨진 루틴은 보통 **덩어리**라 부른다.[1] 대부분의 경우 덩어리는 사소한 것이다. 예를 들어 실제 매개변수가 변수 이름이면 그 덩어리는 단지 메모리로부터 변수를 읽어온다. 하지만 어떤 경우에는 덩어리를 정교화할 수 있다. 가장 유명한 것은 욘 얀센의 이름을 딴 얀센의 장치로 알려진 것에서 일어나는 것일 것이다[Rut69]. 그 생각은 서브루틴에 조립한 수식과 수식에서 사용한 한 개 이상의 변수를 전달하는 것이다. 그러고 나서 각 변수의 값을 변경함으로써 호출된 루틴은 신중하고 조직적으로 조립한 수식의 값을 변경한다. 예를 들어 합계 루틴을 작성하기 위해 이 장치를 사용할 수 있다.

주1. 일반적으로 덩어리는 수식의 계산을 유예하기 위해 사용하는 인자가 없는 프로시저다. 덩어리의 다른 예를 예 6.94의 delay 방법과 연습문제 10.13의 promise 생성자에서 볼 수 있다.

```
real procedure sum(expr, i, low, high);
    value low, high;
       comment low와 high를 값으로 전달함;
       comment expr과 i를 이름으로 전달함;
    real expr;
    integer i, low, high;
begin
    real rtn;
    rtn := 0;
    for i := low step 1 until high do
       rtn := rtn + expr;
       comment expr의 값은 i의 값에 따라 다름;
    sum := rtn
end sum
```

이제 다음과 같은 합을 계산하기 위해

$$y = \sum_{1 \le x \le 10} 3x^2 - 5x + 2$$

단순히 아래와 같이 말할 수도 있다.

```
y := sum(3*x*x - 5*x + 2, x, 1, 10);
```

레이블 매개변수

알골 60과 알골 68 모두 레이블을 매개변수로 전달할 수 있게 한다. 호출된 루틴이 특정 레이블로의 goto를 수행하면 제어는 보통 그 지역 문맥을 빠져나가면서 서브루틴 호출 스택을 풀어낼 필요가 있을 것이다. 풀기 연산은 레이블의 위치에 따라 다르다. 각 간섭하는 유효 범위에 대해 goto는 저장된 레지스터를 복구하고, 스택 프레임을 해제하고, 일반적으로 에필로그 코드가 처리하는 다른 연산 모두를 수행한다. 레이블 매개변수를 구현하기 위해 알골 구현은 일반적으로 주어진 레이블에 대해 적절한 연산을 수행하는 덩어리를 전달한다. 레이블의 목적지가 일반적으로 주변 유효 범위에 있어야 하고 거기에서 정적 유효 범위 지정 규칙 하에서 호출자에게 보여야 함을 기억하자.

서브루틴이 일반적인 연산을 수행하는 것을 피하고 지역적 문맥에서 처리할 수 없는 조건인 예외적인 조건을 처리하기 위해 보통 레이블 매개변수를 사용한다. 복귀하는 대신 문제(예를 들면 잘못된 입력)를 만나는 루틴은 레이블이 교정적인 연산을 수행하거나 적절한 오류 메시지를 출력하는 코드를 참조한다는 가정에서 레이블 매개변수로의 goto를 수행할 수 있다. 좀 더 최신 언어에서 레이블 매개변수는 8.5절에서 논의한 좀 더 구조적인 예외 처리 방법으로 대체한다.

설계와 구현

이름에 의한 호출

실질적으로 알골 60과 시뮬라 프로그램에서 이름에 의한 호출의 대부분을 단지 서브루틴이 실제 매개변수의 값을 변경하기 위해 사용한다. 두 언어 모두 참조에 의한 호출을 제공하지 않는다. 불행히도 이름에 의한 호출은 참조에 의한 호출보다 훨씬 더 비싸다. 그것이 (단순한 간접적 접근과 대조적으로) 모든 형식 매개변수의 사용에 대해 덩어리의 호출을 요구하기 때문이다. 또한 이름에 의한 호출은 주변 유효 범위에서의 변수 변경이 무심코 형식 매개변수의 값을 바꿀 때 프로그램 버그를 잡기 힘들게 하는 경향이 있다(참조에 의한 호출은 3.3.4절에서 논했던 것처럼 이 문제의 좀 더 관대한 형태로부터도 고통을 받는다). 얀슨의 장치와 같은 고의적인 교묘함은 상대적으로 드물고 다른 언어에서 형식 서브루틴의 사용을 통해 그것을 모방할 수 있다. 알골 68에서는 참조의 의한 호출을 지지해 이름에 의한 호출을 버린다.

설계와 구현

필요에 의한 호출

미란다와 하스켈과 같은 함수형 언어는 일반적으로 정규 순서 값 계산의 암기 구현을 사용해 매개변수를 전달한다. 이 게으른 구현을 때때로 필요에 의한 호출이라 한다. 암기는 그것이 처음 필요한 때에 매개변수의 값을 계산하고 기록하고 기록된 값을 그 이후로 사용한다. 부수효과가 없을 때 필요에 의한 호출은 이름에 의한 호출과 구분할 수 없다. 그것은 반복된 값 계산의 비용을 피하지만 부수효과를 정말로 갖는 언어에서 얀슨의 장치와 같은 기법의 사용을 배제한다. 명령형 언어 사이에서 필요에 의한 호출은 스크립팅 언어 R에서 나타난다. 거기서 그것을 사실상 불필요한 복잡한 인자를 단 한번이라도 계산하는 비용을 피하기 위해 사용한다.

✓ 확인문제

60. 이름에 의한 호출은 무엇인가? 그것을 처음 제공한 언어는 무엇인가? 그 언어의 후속 버전이 그것을 사용하지 않는 이유는 무엇인가?

61. 필요에 의한 호출은 무엇인가? 이름에 의한 호출과 어떻게 다른가?

62. 이름에 의한 호출 매개변수를 사용한 서브루틴은 매크로와 어떻게 다른가?

63. 덩어리는 무엇인가? 무엇을 위해 그것을 사용하는가?

64. 얀슨의 장치는 무엇인가?