6장 데이터 타입(4)

2020. 6. 11

순천향대학교 컴퓨터공학과

목차

타입 검사

- 피연산자와 연산자의 개념을 일반화하여 부프로그램과 배정문을 포함
  - 부프로그램은 피연산자가 매개변수인 연산자
  - 배정은 좌측 변수와 우측 식이 피연산자인 이항 연산자
- 타입 검사(type checking)는 연산자에 대한 피연산자들의 타입이 호환가능한지를 확인하는 행위
- 호환가능 타입(compatible type) 이란 연산자에 대해서 적법하거나, 또는 적법한 타입으로 컴파일러에 의해서 묵시적으로 변환되는 것이 언어 규칙에서 허용되는 타입
- 타입의 묵시적 자동변환을 타입 강제변환(coercion)이라 한다.
- 타입 오류(type error)는 연산자를 부적절한 타입의 피연산자에 적용할 때 발생

타입 검사

- 모든 타입 바인딩이 정적이면, 거의 모든 타입 검사는 정적일 수 있다.

- 타입 바인딩이 동적이면, 타입 검사는 동적이어야 하는데, 이를 동적 타입 검사(dynamic type checking)이라 한다.

강 타입

- 프로그래밍 언어는 타입 오류가 항상 탐지되면 강 타입 (strongly typed languages)이다.
  - 모든 피연산자의 타입이 컴파일 시간 또는 실행 시간에 결정
  - 타입 오류를 초래하는 변수의 모든 잘못된 사용 탐지

언어 예제

- Ada: 거의 강타입 언어
  - 프로그래머가 타입 검사 중단 요청 가능
  - Unchecked_Conversion으로 일시적 타입 검사 중단 가능
- Java, C#: Ada와 유사
  - 타입 캐스팅에 의한 오류 가능성
  - 타입 오류가 탐지 되지 않은 묵시적 방법은 없다
- C, C++:
  - 공용체로 인하여 강타입 언어가 아님

강 타입 (2)

- 타입 강제변환은 강 타입을 약화시킴

```
int a, b;
float d;
…
a = a+d; // a + b의 작성을 의도했다면?
```

- 다음 언어에서 타입 강제 변환의 정도는? (배정문, 산술식)
  - Ada: 강제 변환이 거의 없음
  - Java, C#
  - C, C++

타입 동등(type equivalence)

- 배열과 레코드의 구조화된 타입에 대해서는 타입 동등(type equivalence) 개념 적용
  - 식에서 한 타입의 피연산자가 타입 강제 변환 없이 다른 타입의 피연산자로 대체될 수 있으면, 이 두 개의 타입은 동등하다.
  - 타입 강제 변환이 없는 타입 호환성의 엄격한 형식

- 타입 동등을 정의하는 2가지 접근 방법
  - 이름 타입 동등(name type equivalence)
  - 구조 타입 동등(structure type equivalence)

이름 타입 동등

- 두 변수가 동일한 선언문에 속하거나, 같은 타입 이름으로 선언되면, 이러한 두 변수는 이름 타입 동등(name type equivalence)하다.

- 구현하기 쉬우나 매우 제약적

```
type idxtype 1..100; // 새로운 타입 생성
count: integer;
Idx: idxtype; // idx와 count는 타입 동등한가?
```

  - 정수의 부분범위 타입이 정수와 동등하지 않다.
  - 구조화된 타입(사용자 정의 타입)이 부프로그램 매개변수로 전달될 때, 그러한 타입은 전역적으로 단지 한번 정의되어야 한다.

구조 타입 동등

- 두 변수의 타입이 동일한 구조를 가지면 구조 타입 동등(structure type equivalence)하다.

- 더 유연하지만 구현하기 어렵다
  - 두 타입의 전체 구조를 비교하는 것이 간단하지 않다.
    - 두 레코드 타입이 구조적으로 동일하나 다른 필드 이름을 가지면
    - 두 배열이 첨자만이 다른 경우 (e.g. [1..10], [0..9])
    - 두 열거 타입에서 리터럴 철자만 다른 경우
  - 동일한 구조를 갖는 타입들을 구별할 수 없다.

```
type celsius = real;
type fahrenheit = real; // celsius와 fahrenheit는 타입 동등?
```

언어의 예제: C

- struct, union, enum에는 이름 타입 동등 적용
- 다른 비 스칼라 타입에는 구조 타입 동등 적용
  - 배열 타입은 동일한 원소 타입을 가지면 동등

- typedef는 새로운 타입을 생성하지 않음에 유의
  - 기존 타입에 대한 새로운 이름을 정의
  - typedef로 정의된 타입은 그 부모 타입과 동등

- 예외 사항: 구조체, 열거, 공용체가 다른 파일에 정의될 경우에는 구조 타입 동등 적용

언어의 예제(2)

- C++
  - C와 유사
  - 다른 파일에 정의된 구조체, 공용체에 대한 예외사항이 없음
- Java / C#