프로그래밍의 원리 2012 가을
실습 9
OCaml 시작하기

서울대학교 프로그래밍 연구실
강동옥,최민아

2012년 11월 15일

- 다음과 같이 실습/숙제하시면 편리합니다.
  1. 에디터를 사용해서 코드를 작성합니다.
  2. 코드가 있는 디렉토리에서 ocaml을 실행시킵니다. (대화형 실행기)
  3. 작성한 코드를 아래와 같이 불러옵니다. 코드에 오류가 없다면 코드에서 정의한 내용을 모두 읽어 올 것이고, 오류가 있으면 메시지가 뜰 것입니다.

     ```
     #use "FILENAME";;
     #use "ex1.ml";;
     ```

  4. 혹은 다음처럼 한번에 실행할 수도 있습니다.

     ```
     ocaml -init [FILE NAME]
     ledit ocaml -init [FILE NAME]
     ```

  5. 데이터를 넣어 테스트해 봅시다. 대화형 실행기는 바로 결과를 얻을 수 있어서 편리합니다.
  6. ctrl+D를 누르거나, #quit;;를 입력하면 대화형 실행기가 종료됩니다.

연습문제

1. 이제 처음으로 돌아가서, 이전 실습에서 Scheme으로 작성했던 프로그램을 OCaml로 똑같이 작성해 봅시다. 먼저 아래와 같이 `sign` 함수를 만들어 봅시다. 이 함수는 정수 하나를 인자로 받아 그 수가 양수이면 1, 음수이면 -1, 아니면 0을 돌려 줍니다. `sign.ml`로 저장합시다.

   ```
   let sign x = ...
   ```

   이제 대화형 실행기를 켜서 아래와 같이 코드를 불러오면, 자동으로 타입이 유추됩니다.

   ```
   # #use "sign.ml";;
   val sign : int -> int = <fun>
   ```

   함수가 제대로 동작하는지 확인해 봅시다.

```
# sign 0;;
- : int = 0
# sign 10;;
- : int = 1
# sign (-99);;
- : int = -1
```

2. fact 함수를 만들어 봅시다. 이 함수는 정수를 인자로 받아서 팩토리얼 값을 돌려줍니다. 음수인 경우는 0을 돌려 줍니다.

```
# fact 3;;
- : int = 6
# fact 5;;
- : int = 120
```

3. reverse 함수를 만들어 봅시다. 이 함수는 리스트를 인자로 받아 원소들을 역순으로 나열한 리스트를 돌려 줍니다.

```
> let x = reverse [1; 2; 3; 4];;
val x: int list = [4; 3; 2; 1]
```

4. filter 함수를 만들어 봅시다. 이 함수는 리스트와 함수를 인자로 받아 리스트의 원소중에 함수의 결과값을 참으로 하는 입력들만 남긴 리스트를 돌려줍니다.

```
> let x = filter [1; 2; 3; 4] (fun x -> (x mod 2) = 0) ;;
val x: int list = [2; 4]
```

5. tree_to_list를 만들어 봅시다. 이 함수는 나무의 잎새에 달려있는 모든 원소들을 가져와서 리스트로 돌려줍니다. type constructor를 사용하여 나무 타입을 정의해보고, tree_to_list함수를 만들어 봅시다.

```
> let tr = Node [Leaf 3; Node [Leaf 4; Leaf 5]; Leaf 7] in
  tree_to_list tr;;
- : int list = [3; 4; 5; 7]
```

6. add라는 함수를 정의해봅시다. add는 다음과 같은 4가지 종류의 값에 대해 연산을 수행합니다.

   - 정수 덧셈 연산.
   - 실수 덧셈 연산.
   - 리스트 접합(list concatenation) 연산.
   - 문자열 접합(string concatenation) 연산.

   각각의 연산에 대해 알맞는 연산자는 +, +., @, ^가 있습니다. 아래에 수업시간에 배운 type function을 이용하여 add가 받는 값의 타입을 정의해 놓았습니다. add를 만들어 보세요.

```
val add : 'a value -> 'a value -> 'a value

type 'a value = INT of int
   | FLOAT of float
   | LIST of 'a list
   | STRING of string

# add (INT 3) (INT 4);;
- : 'a value = INT 7
# add (FLOAT 4.) (FLOAT 3.2);;
- : 'a value = FLOAT 7.2
# add (LIST ["a";"b"]) (LIST ["c"]);;
- : string value = LIST ["a"; "b"; "c"]
# add (STRING "ab") (STRING "c");;
- : 'a value = STRING "abc"
# add (INT 3) (FLOAT 2.);;
Exception: Type_Not_Matched.
# add (LIST ["a";"b"]) (LIST [1;2]);;
Error: This expression has type int but an expression was expected of type
         string
```