Interpreters, Part 2

Dr. Mattox Beckman

University of Illinois at Urbana-Champaign
Department of Computer Science

Define the Types – Types.hs

```
1 data Exp = IntExp Integer
2          | IntOpExp String Exp Exp
3          | RelOpExp String Exp Exp
4          | BoolOpExp String Exp Exp
5          | BoolExp Bool
6    deriving (Show, Eq)
7
8 data Val = IntVal Integer
9          | BoolVal Bool
10   deriving (Show, Eq)
```

Eval – Bools, and, or

```
1  boolOps = [ ("&&",(&&))
2            , ("||",(||))]
3
4  liftBoolOp f (BoolVal i1) (BoolVal i2) = BoolVal (f i1 i2)
5  liftBoolOp f _            _            = BoolVal False
6
7  eval (BoolExp b) _ = BoolVal b
8
9  eval (BoolOpExp op e1 e2) env =
10   let v1 = eval e1 env
11       v2 = eval e2 env
12       Just f = lookup op boolOps
13    in liftBoolOp f v1 v2
```

Adding Comparisons

```
1  relOps = [ ("<", (<)) , ("<=", (<=)) , (">", (>))
2           , (">=", (<=)) , ("==", (<=)) , ("/=", (/=)) ]
3
4  liftRelOp f (IntVal i1) (IntVal i2) = BoolVal (f i1 i2)
5  liftRelOp f _           _           = BoolVal False
6
7  eval (RelOpExp op e1 e2) env =
8    let v1 = eval e1 env
9        v2 = eval e2 env
10       Just f = lookup op relOps
11    in liftRelOp f v1 v2
```

A Simple Let Expression

- We want to define local variables:

```
1 i4> 3 + let x = 2 + 3 in x * x end
2 IntVal 28
```

- Need two new Exp constructors.

```
1 data Exp = VarExp String
2          | LetExp String Exp Exp
3          | ...
```

Coding Eval for Variables

- For variables, we look them up in the environment.

```
1 eval (VarExp var) env =
2   case lookup var env of
3     Just val -> val
4     Nothing -> IntVal 0
```

Coding Eval for Let

```
1 eval (LetExp var e1 e2) env =
2   let v1 = eval e1 env
3    in eval e2 (insert var v1 env)
```

- The `insert var v1 env` call acts like pushing a value onto a stack!

Next Time

- You now have some interesting things for your interpreter.
- The reference implementation is in `i4`.
- We've also added a `IfExp` to the types if you want to try adding this.