Série d'exercices #$7\frac{1}{2}$

IFT-2035

November 9, 2023

Programmation impérative en langage fonctionnel

Re-voici le code d'un évaluateur en Haskell:

```
type Var = String

-- Expressions du code source en forme ASA.
data Exp = Enum Int          -- Une constante
         | Evar Var          -- Une variable
         | Elambda Var Exp   -- Une fonction
         | Ecall Exp Exp     -- Un appel de fonction

data Val = Vnum Int            -- Un nombre entier
         | Vfun (Val → Val)    -- Une fonction

elookup ((x1,v1):env) x =
    if x == x1 then v1 else elookup env x

env0 = [("+", Vfun (λ(Vnum x1) → Vfun (λ(Vnum x2) → Vnum (x1 + x2))))]

eval env (Enum n) = Vnum n
eval env (Evar x) = elookup env x
eval env (Elambda arg body) =
    Vfun (λactual → eval ((arg,actual):env) body)
eval env (Ecall fun actual) =
    case eval env fun of
      Vfun f → f (eval env actual)
```

Changer le code en utilisant le monade `IO` pour y ajouter une fonction primitive *print_hello* dans l'environnement initial qui imprime le mémorable message "`Hello:`" suivi de son argument numérique (qui est aussi renvoyé comme valeur de retour). Pour cela, il utilisera *putStrLn* :: `String -> IO ()`.