Ahorcado

Programación funcional imperativa

Juan Pedro Villa Isaza

Stack Builders

24 de abril de 2015

Stack Builders

Haskell es:

- Funcional
- *Puro*
- ...

Haskell es:

- Funcional
- *Puro*
- ...
- *Interesante*

Haskell es:

- Funcional
- *Puro*
- ...
- *Interesante*
- ¿Inútil?

- ```
  reverse :: [a] -> [a]
  reverse = ...
  ```

- ```
  > reverse "haskell"
  "lleksah"
  ```

Stack Builders

```
main :: IO ()
main = putStrLn (reverse "haskell")
```

```
> main
lleksah
```

Stack Builders

```haskell
main :: IO ()
main = do
  line <- getLine
  putStrLn (reverse line)
```

```
> main
haskell
lleksah
```

Stack Builders

```
type IO a = Mundo -> (a,Mundo)
```

Stack Builders

Haskell es el mejor lenguaje de programación imperativa del mundo.

—Simon Peyton Jones

Stack Builders

Ahorcado

- Distinción clara entre:
  - Acciones y funciones
  - Código impuro y código impuro
  - Programación imperativa y programación no imperativa
- Razonamiento ecuacional
- Refactoring
- Menos dolores de cabeza

Haskell es el mejor lenguaje de programación imperativa del mundo.

—Simon Peyton Jones

Stack Builders

https://github.com/stackbuilders/hangman

- Con base en las respuestas de más de 368000 personas:
  - “El aprendizaje de este lenguaje cambió significativamente la manera en que uso otros lenguajes”.

1. Haskell
2. Scheme
3. Coq
4. Common Lisp
5. Erlang

43. Fortran
44. R
45. AWK
46. Visual Basic
47. Cobol

`http://hammerprinciple.com/therighttool`

- Con base en las respuestas de más de 368000 personas:
  - "El aprendizaje de este lenguaje mejoró mi habilidad como programador".

1. Haskell
2. Standard ML
3. Scheme
4. Coq
5. Common Lisp

45. Shell
46. Matlab
47. PHP
48. Visual Basic
49. Cobol

`http://hammerprinciple.com/therighttool`

Stack Builders

- Con base en las respuestas de más de 368000 personas:
  - "Recomendaría a la mayoría de programadores aprender este lenguaje, sin importar si tienen o no una necesidad específica de hacerlo".

1. Haskell
2. Scheme
3. Coq
4. Clojure
5. Erlang

44. ActionScript
45. Pascal
46. Fortran
47. Visual Basic
48. Cobol

`http://hammerprinciple.com/therighttool`

```haskell
reverse :: [a] -> [a]
reverse =  foldl (flip (:)) []
```

```haskell
reverse :: [a] -> [a]
reverse l = rev l []
  where
    rev []     a = a
    rev (x:xs) a = rev xs (x:a)
```

Stack Builders