

# Szybko. Szybciej. Haskell.

darkestkhan

20 kwietnia 2010

# Słów kilka o historii programowania funkcjonalnego:

- Alonzo Church(1903-1995) i  $\lambda$  calculus(1930)

# Słów kilka o historii programowania funkcjonalnego:

- Alonzo Church(1903-1995) i  $\lambda$  calculus(1930)
- $\lambda x.x^2$

# Słów kilka o historii programowania funkcjonalnego:

- Alonzo Church(1903-1995) i  $\lambda$  calculus(1930)
- $\lambda x.x^2$
- Haskell Curry(1900-1982) i combinatory logic(Moses Schofinkel, 1924)

# Podstawowe elementy programowania funkcjonalnego:

- funkcja jako podstawowy budulec

# Podstawowe elementy programowania funkcjonalnego:

- funkcja jako podstawowy budulec
- lista podstawową strukturą danych

# Podstawowe elementy programowania funkcjonalnego:

- funkcja jako podstawowy budulec
- lista podstawową strukturą danych
- zachłanne czy leniwe wartościowanie—odwieczny konflikt

# Podstawowe elementy programowania funkcjonalnego:

- funkcja jako podstawowy budulec
- lista podstawową strukturą danych
- zachłanne czy leniwe wartościowanie—odwieczny konflikt
- garbage collector—wymysł programistów funkcyjnych

# Podstawowe elementy programowania funkcjonalnego:

- funkcja jako podstawowy budulec
- lista podstawową strukturą danych
- zachłanne czy leniwe wartościowanie—odwieczny konflikt
- garbage collector—wymysł programistów funkcyjnych
- bezimienna funkcja lambda

# Podstawowe elementy programowania funkcjonalnego:

- funkcja jako podstawowy budulec
- lista podstawową strukturą danych
- zachłanne czy leniwe wartościowanie—odwieczny konflikt
- garbage collector—wymysł programistów funkcyjnych
- bezimienna funkcja lambda
- składanie funkcji i funkcje wyższego rzędu



# Haskell

Freedom  
from  
state

# Główny gość programu - Haskell:



- jest libre :)

# Haskell - czyli programowanie czysto funkcyjnego:

- jest libre :)
- interpreter - Glasgow Haskell Compiler Interpreter

# Haskell - czyli programowanie czysto funkcyjnego:

- jest libre :)
- interpreter - Glasgow Haskell Compiler Interpreter
- kompilator - Glasgow Haskell Compiler

# Haskell - czyli programowanie czysto funkcyjnego:

- jest libre :)
- interpreter - Glasgow Haskell Compiler Interpreter
- kompilator - Glasgow Haskell Compiler
- jestem leniwy i nie będę zachłannie wartościował

# Haskell - czyli programowanie czysto funkcyjnego:

- jest libre :)
- interpreter - Glasgow Haskell Compiler Interpreter
- kompilator - Glasgow Haskell Compiler
- jestem leniwy i nie będę zachłannie wartościował
- hej, to jest typ `Int` a nie `Integer`, więc nie policzę

# Haskell - czyli programowanie czysto funkcyjnego:

- jest libre :)
- interpreter - Glasgow Haskell Compiler Interpreter
- kompilator - Glasgow Haskell Compiler
- jestem leniwy i nie będę zachłannie wartościował
- hej, to jest typ `Int` a nie `Integer`, więc nie policzę
- jest szybki...

# Haskell - czyli programowanie czysto funkcyjnego:

- jest libre :)
- interpreter - Glasgow Haskell Compiler Interpreter
- kompilator - Glasgow Haskell Compiler
- jestem leniwy i nie będę zachłannie wartościował
- hej, to jest typ `Int` a nie `Integer`, więc nie policzę
- jest szybki...
- a nawet jeszcze szybszy

# Haskell - czyli programowanie czysto funkcyjnego:

- jest libre :)
- interpreter - Glasgow Haskell Compiler Interpreter
- kompilator - Glasgow Haskell Compiler
- jestem leniwy i nie będę zachłannie wartościował
- hej, to jest typ `Int` a nie `Integer`, więc nie policzę
- jest szybki...
- a nawet jeszcze szybszy
- niezawodny –nigdy się nie myli

# Monady:

Monada jest trójką  $(M, \text{return}, \gg=)$  składająca się z konstruktora typów  $M$  i dwóch polimorficznych funkcji:

- $\text{return} :: a \rightarrow Ma$
- $(\gg=) :: Ma \rightarrow (a \rightarrow Mb) \rightarrow Mb$

Muszą spełniać następujące trzy prawa:

- $m \gg= \text{return} = m$
- $\text{return} x \gg= f = fx$
- $(m \gg= f) \gg= g = m \gg= (\lambda x. fx \gg= g)$

Gdzie  $\gg=$  oznacza przypisanie

# Z czego się uczyć?

- ze wszystkich książek traktujących o programowaniu funkcyjnym

# Z czego się uczyć?

- ze wszystkich książek traktujących o programowaniu funkcyjnym
- IRC (freenode)

# Z czego się uczyć?

- ze wszystkich książek traktujących o programowaniu funkcyjnym
- IRC (freenode)
- wikipedia

# Z czego się uczyć?

- ze wszystkich książek traktujących o programowaniu funkcyjnym
- IRC (freenode)
- wikipedia
- i ze spisu literatury :)

- [www.haskell.org](http://www.haskell.org)
- “School of Expression”
- “The Haskell Road to Logic, Maths and Programming”
- “Haskell - The Craft of functional programming”
- “Learn You a Haskell for Great Good!”
- “Real World Haskell”
- “Programming Haskell”
- [www.lambda-the-ultimate.org](http://www.lambda-the-ultimate.org)