Ĉu pure funkciaj programlingvoj finvenkos?

Richard Hable

KAEST 2018

La plezuroj de programkreado

- Krei konceptojn
- Difini arkitekturon de programsistemoj
- Utiligi diferencajn programlingvojn
- Vidi sukcesan aplikon de propre kreitaj programoj

  SED
- Dependas de sistemoj, metodoj, iloj
  - Limigita aparataro (memoro, rapideco)
  - Komplikaj operaciumoj, bibliotekoj
  - Komplikaj, malsekuraj programlingvoj

Supraĵa historio de programlingvoj

1940-aj	Estiĝis principa arkitekturo de komputiloj: procesoro + memoro (kun komandoj kaj datumoj)	`LOAD  123`
		`ADD   124`
		`STORE 125`

1950-aj	Aŭtomataj tradukiloj de formuloj (FORTRAN)	`A = B + C`
1960-aj 1970-aj	Ordonemaj, algoritmaj programlingvoj (Pascal, C)	`if x > y then a := b + c` `else a := b - c`
1980-aj	Modernaj funkciaj programlingvoj (Miranda, Haskell, ML)	`a := if x > y then b + c` `     else b - c`
1990-aj 2000-aj	Objektema programado (C++, Java)	`a.add (b)`
2010-aj	Funkciaj elementoj en ordonemaj programlingvoj (C#, Java 8)	`a.map (y -> x + y)`

Eŭklida algoritmo: plej granda komuna divizoro

- Pseŭdokodo: funkcie

```
FUNKCIO pgkd (a, b)
    SE b = 0
        REDONI a
    ALIE
        REDONI pgkd (b, a % b)
```

- Pseŭdokodo: ordoneme

```
FUNKCIO pgkd (a, b)
    DUM b ≠ 0
        t := b
        b := a % b
        a := t
    REDONI t
```

(agordigita de https://eo.wikipedia.org/wiki/E%C5%ADklida_algoritmo)

Eŭklida algoritmo: plej granda komuna divizoro

- Pseŭdokodo: funkcie

```
FUNKCIO pgkd (a, b)
    SE b = 0
        REDONI a
    ALIE
        REDONI pgkd (b, a % b)
```

- Pseŭdokodo: ordoneme

```
FUNKCIO pgkd (a, b)
    DUM b ≠ 0
        t := b
        b := a % b
        a := t
    REDONI a
```

(agordigita de https://eo.wikipedia.org/wiki/E%C5%ADklida_algoritmo)

Filtro-funkcio

en la pure funkcia programlingvo Haskell

```
filtro p [] = []
filtro p (x:xs) | p x       = x : filtro p xs
                | otherwise = filtro p xs
```

(agordita de http://en.wikibooks.org/wiki/Yet_Another_Haskell_Tutorial)

Signifas kion?

```
filtro p [] = []
filtro p (x:xs) | p x       = x : filtro p xs
                | otherwise = filtro p xs
```

filtro (< funkcio p >, < malplena listo >) =
        < malplena listo >

filtro (< funkcio p >, < listo kun minimume unu elemento >) =
        SE p (< unua elemento >) :
                < unua elemento > + filtro (< funkcio p >, < resto de listo >)
        ALIE:
                filtro (< funkcio p >, < resto de listo >)

Projekto: nova pure funkcia programlingvo "Trankvila"

- Celoj
  - Plej eble simpla, tamen kapabla
  - Facile legebla
  - Simila al kutimaj (ordonemaj) programlingvoj

- Rimedoj
  - Konvencia sintakso
  - Datumtipoj simile al objektemaj programlingvoj
  - Neniu struktura komparo kaj elpreno (*x:xs*)
  - Malpli da inferenco de datumtipoj
  - Neniu parta apliko de funkcioj ("*currying")*
  - Neniuj sennomaj vicoj da elementoj ("tuples")
  - Neniu speciala sintakso por monadoj

La filtrofunkcio en Trankvila

- Haskell

```
filtro p [] = []
filtro p (x:xs) | p x       = x : filtro p xs
                | otherwise = filtro p xs
```

- Trankvila

Predikato [T] = FUNCTION (value: T): Boolean;

(:List) filtro (:Predikato [T]): List [T] := LIST ();

(nl: Nonempty List) ! filtro (:Predikato [T]): List [T] :=
    IF predikato (nl: first) THEN nl: first -> nl: rest: filter (predikato)
    ELSE nl: rest: filtro (predikato);

LIST (1, 2, 3): filtro (?(nombro) nombro % 2 = 0) = LIST (2)

Ekzemplo: Saluton, Mondo!

```
TRANKVILA MODULE Saluton (v1.0) BY Acme;

LET
        saluto* := "Saluton, Mondo!" + new line;
END.
```

(agordigita de Trankvila Modules/Acme/Hello.trankvila)

Ekzemplo: Fibonaĉi-nombroj

```
-- Senlima listo kun maldiligenta (laŭbezona) taksado

fibonaĉi nombroj (unua, dua: Integer): List [Integer] :=
        unua -> fibonaĉi nombroj (dua, unua + dua);

fibonaĉi nombroj 1 ĝis 50 := fibonaĉi nombroj (0, 1): take (50);
```

(agordigita de Trankvila Modules/Trankvila/Classic Examples.trankvila)

Realigstato de Trankvila

- Tradukilo bazita sur Ĝavo
- Programada cirkaŭaĵo bazita sur Eklipso
- Simplaj bibliotekaj kaj ekzemplaj moduloj
- Ankoraŭ sufiĉe da cimoj!

Planoj

- Detala priskribo de la programlingvo
- Plibonigo de la tradukilo
- Kreado de praktika bibliotekaj moduloj
- Interalie subteno de enigo kaj eligo

http://trankvila.org

Diskutu!

Demandoj?

Rimarkoj?

Ideoj?

Plendoj?

Richard.Hable@trankvila.org

http://trankvila.org

Quine
(= programo kiu skribas sian propran fontkodon)

```
quine* :=
	LET
		comma := LIST (CHARACTER (44));
		lines := LIST [String] (
			"quine* :=",
			"	LET",
			"		comma := LIST (CHARACTER (44));",
			"		lines := LIST [String] (",
			"		);",
			"	IN",
			"		(",
			"			lines: take (4): map [String] (?(line) tab + line + new line)",
			"			+ lines: take (12): map [String] (?(line) tab + tab + tab + tab + quote + line + quote + comma + new line)",
			"			+ LIST (tab + tab + tab + tab + quote + lines: get (12) + quote + new line)",
			"			+ lines: drop (4): map [String] (?(line) tab + line + new line)",
			"		)",
			"		: fold left [String] (LIST (), ?(left, right) left + right);"
		);
	IN
		(
			lines: take (4): map [String] (?(line) tab + line + new line)
			+ lines: take (12): map [String] (?(line) tab + tab + tab + tab + quote + line + quote + comma + new line)
			+ LIST (tab + tab + tab + tab + quote + lines: get (12) + quote + new line)
			+ lines: drop (4): map [String] (?(line) tab + line + new line)
		)
		: fold left [String] (LIST (), ?(left, right) left + right);
```

La sama en Haskell...

```
ap(++)show"ap(++)show"
```

http://rosettacode.org/wiki/Quine#Haskell