

Pointfrip Dokumentation

2022-05-06

"FP zwingt den Programmierer, so daß die konstruierten Funktionen immer pure sind."

Designziele

Infix-Combinator-Mechanismus

mit Objekt-Orientierung durch Arrays und Index-Typen (Objekt-Klassen)

Vokabular baut auf einem System der Emergenz auf + Spezialisierung

FP als schachtelbare Sprache - strenge Form der strukturierten Programmierung

keine Gotos nur Calls, Tailrecursion unterstützung

Function-Level ,Type-Level

Orientierung an APLs Rechts-vor-Links Regel

Trennung in Pure-Functions und Monaden-Effekte

Pattern Matching für Dict

kein Scope, weil nur Instanzenvariablen

(Out-of-the-Box)

Precompiler + FP Interpreter

Typen-orientierter Automat (higher-Order)

Infix-Combinator-Mechanismus (infix meta combination)

Erweiterbarkeit der Sprache

Erweiterbarkeit nach den Ideen von „Growing a Language“ von Guy Steel

ist das FP-Projekt mit einem Objekt-Mechanismus ausgestattet, so dass man

Objekte und Klassen definieren und erben kann und auf vier verschiedenen Weisen ansprechen kann. Damit man

- complex-Klassen,
 - fraction-Klassen,
 - interval-Klassen,
 - matrix-Klassen
- etc, hinzufügen kann.

Außerdem besteht für die FP-Sprache die Möglichkeit der Meta-Programmierung, um den Sprachumfang (anders als bei Java) in der Richtung zu erweitern, dass man eigene Schleifen und Kontrollstrukturen hinzufügen kann.

Letzte Änderungen:

- hex-Funktion implementiert,
- pointersize-Funktion implementiert,
- kosmetische Arbeiten am Quelltext,
- reference.rtf/.tmdx erweitert

// (CC0)