OOP vs FP

Daniel Pecos Martínez

@danielpecos
http://danielpecos.com

Valencia.rb - 1 Abril 2014

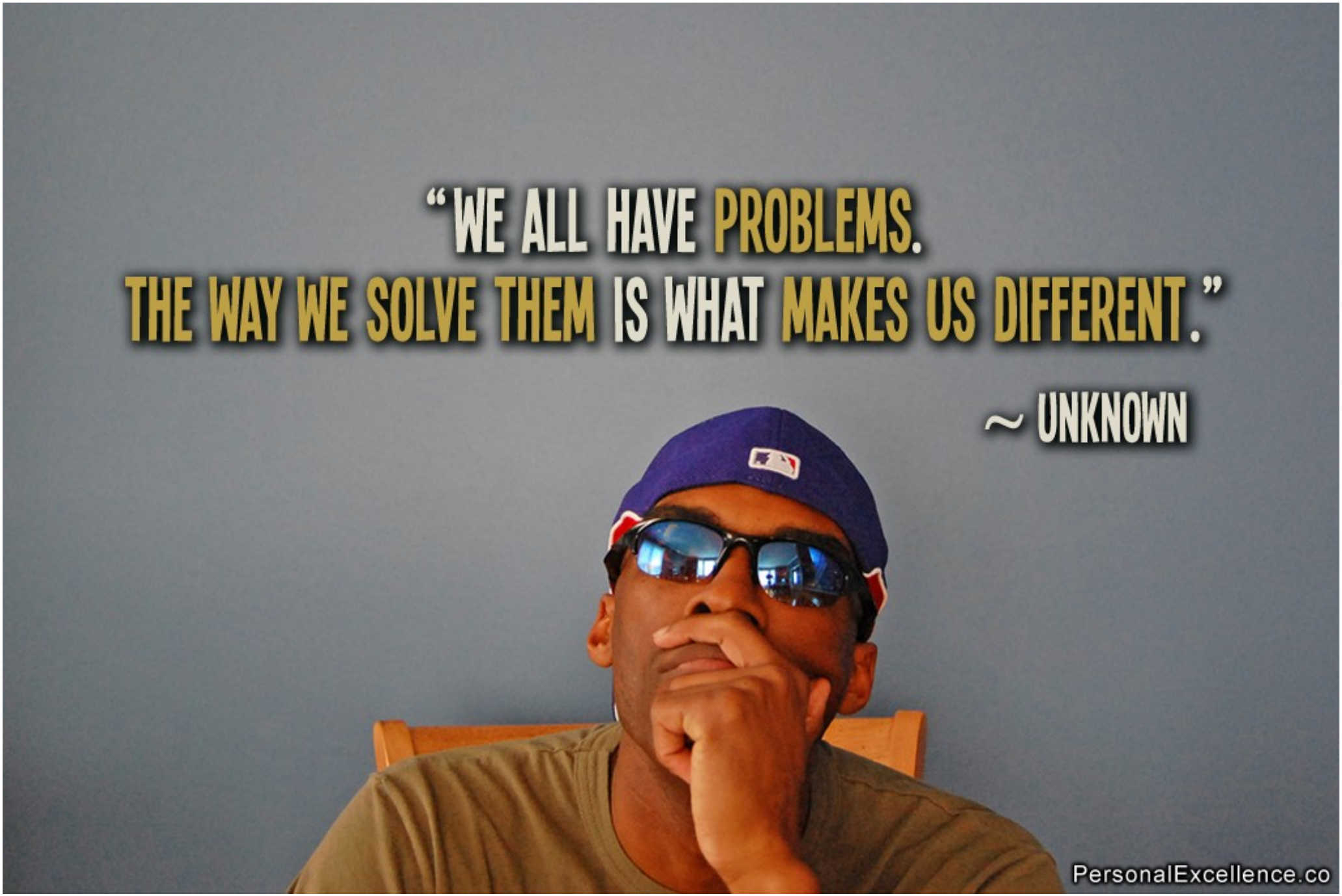
¿Qué es un paradigma?

Es un estilo fundamental de programación, una forma de construir la estructura y elementos de los programas computacionales.

Wikipedia

Es una forma de hacer las cosas. En informática, es el modo en el que se abordan la resolución de problemas.

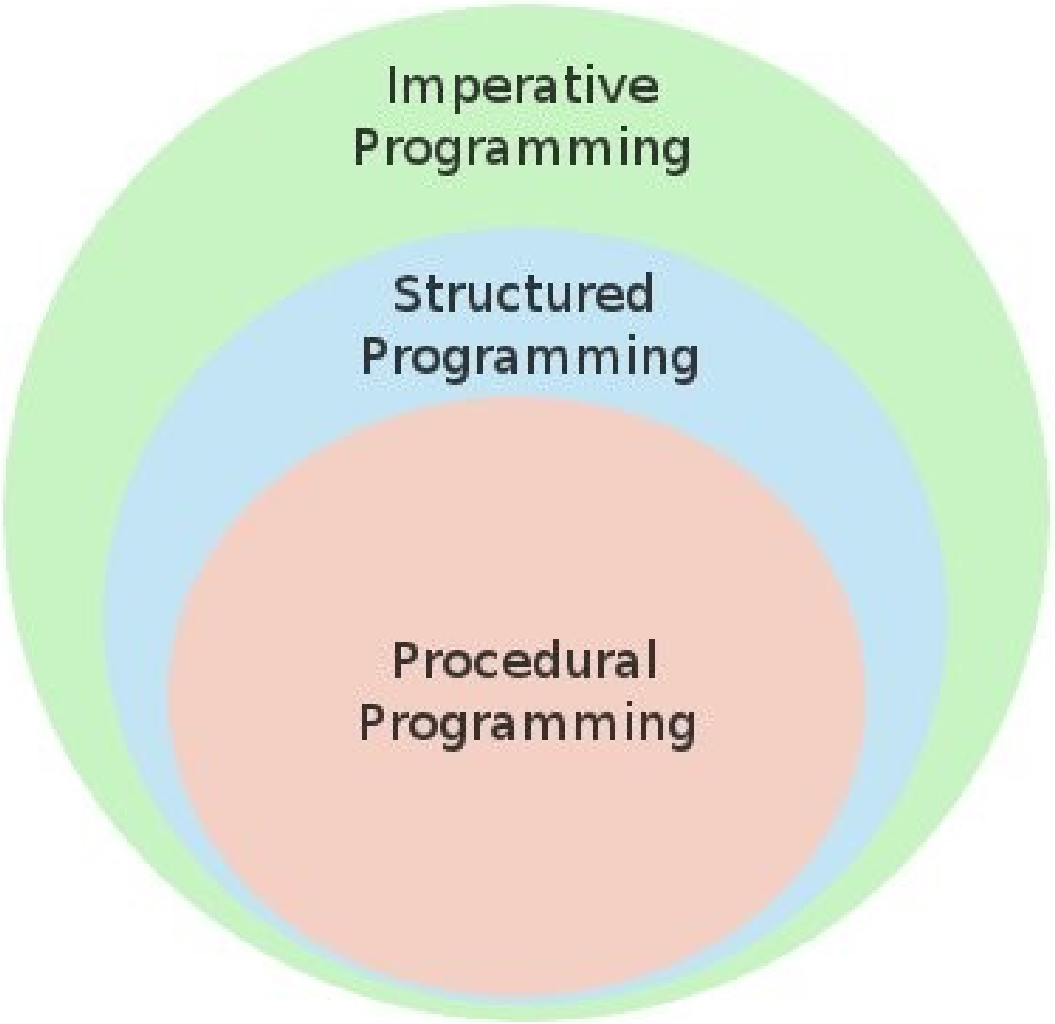
DPM (Yo mismo)


"WE ALL HAVE PROBLEMS.
THE WAY WE SOLVE THEM IS WHAT MAKES US DIFFERENT."
~ UNKNOWN
PersonalExcellence.co

http://plwords.herokuapp.com/

Programación Imperativa

- Características
  - Define las acciones a seguir para resolver el problema
  - Basado en statements que modifican el estado
  - Fuertemente influenciada por arquitectura Von Neumann
- Paradigmas que se incluyen
  - Estructurada: evita hacer uso de funciones que rompen el flujo de programa (GOTOs)
  - Procedural: descomposición del problema en subrutinas
  - Orientación a Objetos (OOP)


Imperative
Programming
Structured
Programming
Procedural
Programming

Paradigm	Description	Main characteristics	Related paradigm(s)	Examples
Imperative	Computation as statements that directly change a program state (datafields)	Direct assignments, common data structures, global variables		C, C++, Java, PHP, Python
Structured	A style of imperative programming with more logical program structure	Structograms, indentation, either no, or limited use of, goto statements	Imperative	C, C++, Java
Procedural	Derived from structured programming, based on the concept of modular programming or the procedure call	Local variables, sequence, selection, iteration, and modularization	Structured, imperative	C, C++, Lisp, PHP, Python
Object-oriented	Treats datafields as objects manipulated through pre-defined methods only	Objects, methods, message passing, information hiding, data abstraction, encapsulation, polymorphism, inheritance, serialization-marshalling		C++, C#, Eiffel, Java, PHP, Python, Ruby, Scala

OOP

- Características
  - Abstracción
  - Encapsulamiento
  - Modularidad
  - Polimorfismo
  - Herencia
- SOLID
  - Single responsibility principle
  - Open/closed principle
  - Liskov substitution principle
  - Interface segregation principle
  - Dependency inversion principle

Programación Declarativa

- Características
  - Describe la lógica de la computación sin definir el flujo de control
  - Eliminación (o reducción) de los efectos colaterales

- Paradigmas que se incluyen
  - Lógico
  - Orienta a restricciones (constraint programming)
  - Dataflow
  - Funcional (FP)

Paradigm	Description	Main characteristics	Related paradigm(s)	Examples
Declarative	Defines computation logic without defining its detailed control flow	4GLs, spreadsheets, report program generators		SQL, regular expressions, CSS
Logical	Define and query relations	Use of unifications to solve equations between symbolic expressions		Prolog
Constraint programming	Relations between variables are stated in the form of constraints	Used as complement to other paradigms. Optimizations problems are a typical use case	Functional, logical, imperative	
Dataflow	Direct graph of data flowing between operations	Numeric processing. Operations treated as chained black boxes.	Functional	
Functional	Treats computation as the evaluation of mathematical functions avoiding state and mutable data	Lambda calculus, compositionality, formula, recursion, referential transparency, no side effects		Erlang, Haskell, Lisp, Clojure, Scala, F#

FP

- Características
  - Evita estado y/o datos mutables
  - Transparencia referencial (funciones puras)
  - *First class & Higher order functions*
  - La iteración se reemplaza por recursividad
- Conceptos Clave
  - Closures
  - Monads
  - Tail recursion
  - Currying
  - Lazy Evaluation

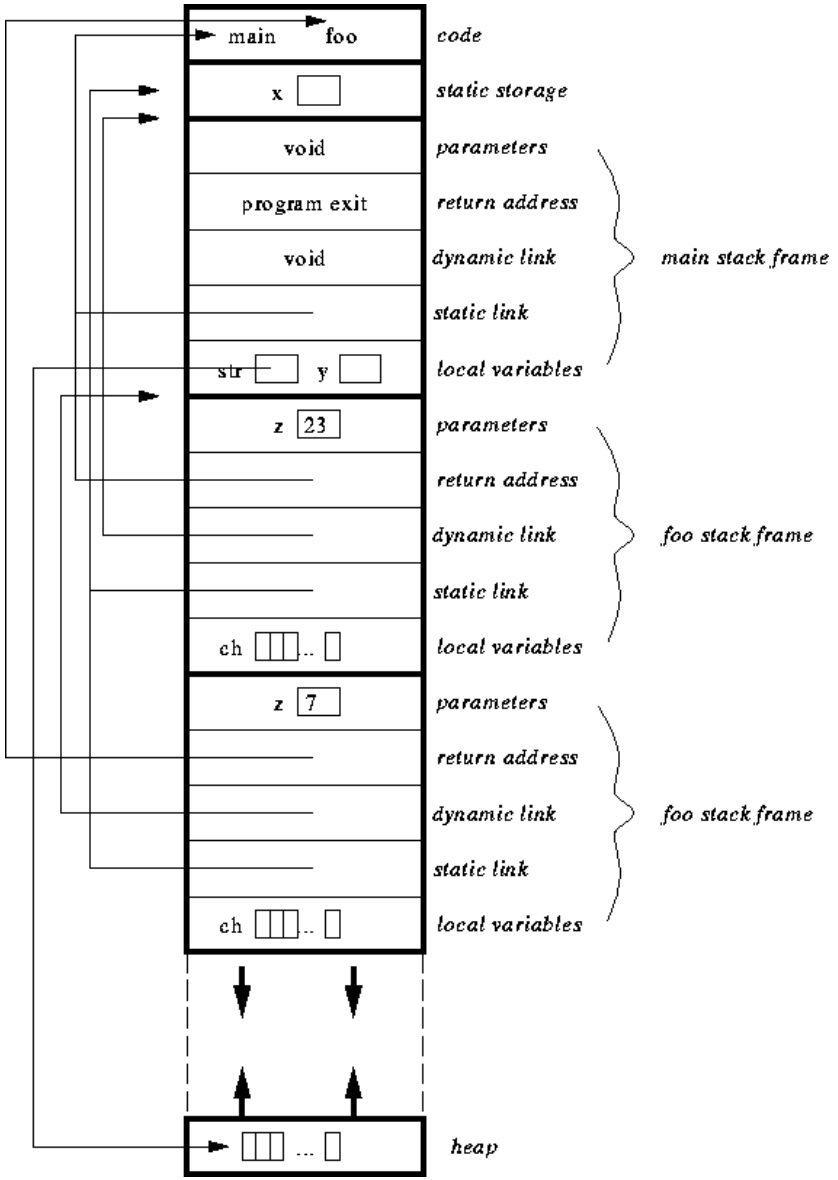
Imperative vs Functional

Characteristic	Imperative approach	Functional approach
Programmer focus	How to perform tasks (algorithms) and how to track changes in state.	What information is desired and what transformations are required.
State changes	Important.	Non-existent.
Order of execution	Important.	Low importance.
Primary flow control	Loops, conditionals, and function (method) calls.	Function calls, including recursion.
Primary manipulation unit	Instances of structures or classes.	Functions as first-class objects and data collections.

Closures

- Una closure es el binomio formado por
  - Una función junto
  - El contexto en el que se definió
- Permite acceder y modificar variables no locales a la función desde cualquier sitio en el que se invoque dicha función, aunque el contexto no sea directamente accesible
- Se utilizan para ocultar el estado
- En JS se usan intensivamente como funciones anónimas en callbacks

Tail Call Optimization



Técnica de compilación que optimiza la recursividad, cuando la llamada recursiva es la última instrucción del bloque, permitiendo la reutilización del *stack frame* para sucesivas invocaciones.

Currying

- Técnica que permite convertir una función multi-parámetro en una cadena de funciones mono-parámetro

$$f(x,y,z) = g(x)(y)(z)$$

$$mult = (x, y) \rightarrow x * y$$

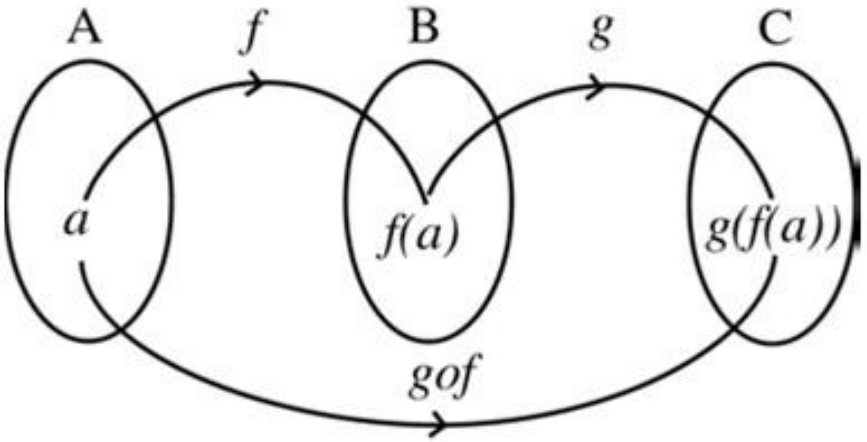
$$doubler = (x) \rightarrow mult(x, 2)$$

Lazy evaluation

- Estrategia que retrasa la evaluación de expresiones hasta que su valor es necesario
- Evita evaluación de expresiones repetidas
- Beneficios:
  - Mejora de rendimiento
  - Construcción de estructuras de datos infinitas
  - Estructuras de control como abstracciones en vez de primitivos

La joya de la corona: *function composition*

- Mecanismo para construir funciones de mayor complejidad a partir de funciones simples
- Requiere que el lenguaje soporte *higher order functions*




KEEP
CALM
AND
LET's
CODE

¿Dónde se utiliza FP?

- Cálculos matemáticos
- IA
- Entornos de alta concurrencia
- BigData - MapReduce

FP desde las trincheras

- Uso el lenguaje X en mi día a día ¿Puedo utilizar programación funcional?
- Sí, por supuesto. Pero...

No será tan natural como si lo hicieras en un lenguaje que te lo facilitase. Debes conocer los límites.

Debate

- MVC ¿OOP o FP?
- OOP vs FP
  - Modelados de dominios
  - Parallel programming
  - UI

Daniel Pecos Martínez

@danielpecos

http://danielpecos.com