

Введение в функциональное программирование

Михаил Беляев

28 сентября 2016

Императивное программирование

- Программа — это набор *инструкций*, изменяющих *состояние* системы
- Всё остальное (процедуры, классы и т.д.) — лишь способы абстракции

```
long factorial(long i) {  
    long result = 1;  
    while(i > 1) {  
        result = result * i;  
        --i;  
    }  
    return result;  
}
```

Императивное программирование: более формально

Выполнение программы — это переход между состояниями

$$\sigma_0 \rightarrow \sigma_1 \rightarrow \sigma_2 \rightarrow \sigma_3 \rightarrow \sigma_4 \rightarrow \dots \rightarrow \sigma_S$$

Императивное программирование: языки

Как процедурные, так и объектные языки

Начиная от Fortran и Algol и заканчивая Java и C#

Функциональное программирование

- Программа — это выражение, которое можно вычислить
- Способы абстракции бывают так же разными

```
long factorial(long i) {  
    return (i <= 1)?  
        1 :  
        i * factorial(i-1);  
}
```

- Выполнение программы — это вычисление этого выражения

Начиная от первых диалектов LISP до Haskell и F#

- Функциональный стиль программирования — частный случай *декларативного* стиля

- Функциональный стиль программирования — частный случай *декларативного* стиля
- Императивная программа отвечает на вопрос «как?»
- Декларативная программа отвечает на вопрос «что?»

- Современные языки имеют широкий инструментарий для работы в стиле ФП
- Как в основном императивные, так и в основном функциональные языки пришли к некоему «гибридному» состоянию

Модно, молодёжно: Rust, Swift, Kotlin, Scala, Go, etc.

- «Старые» языки последних стандартов тоже не остаются Java 8, C++11/C++14, C# 4.X, etc.

В начале было слово?

В начале было слово?

LISP

В начале было слово?

LISP

- 1958 год
- первый функциональный язык программирования
- рекурсия, условные выражения, лямбда-функции
- динамическая типизация
- сборка мусора
- крайне простой синтаксис

Пример кода на LISP

```
(defun (fibonacci (n))  
  (if (or (= n 0) (= n 1))  
      1  
      (+  
        (fibonacci (- n 1))  
        (fibonacci (- n 2))  
      )  
    )  
  )  
)
```

Ещё немного истории

В начале была буква!

X

Алонзо Чёрч, 30е годы XX века

Алонзо Чёрч, 30е годы XX века

Два основных принципа — *аппликация* и *абстракция*

Алонзо Чёрч, 30е годы XX века

Два основных принципа — *аппликация* и *абстракция*

Абстракция через λ -терм:

$\lambda x.$ *<выражение над x >*

Алонзо Чёрч, 30е годы XX века

Два основных принципа — *аппликация* и *абстракция*

Абстракция через λ -терм:

$\lambda x. \text{ <выражение над } x\text{ >}$

Аппликация (применение функции к аргументу):

$f x$

Алонзо Чёрч, 30е годы XX века

Два основных принципа — *аппликация* и *абстракция*

Абстракция через λ -терм:

$\lambda x. \langle \text{выражение над } x \rangle$

Аппликация (применение функции к аргументу):

$f x$

Вычисление = «переписывание» термов:

$(\lambda x. y) a \rightarrow y[a/x]$

«Заменить все вхождения x внутри y на a »

- Можно ввести рекурсию через т.н. Y-комбинатор
- Можно определить структуру данных — пару
- Можно определить логические (Church booleans) и натуральные (Church numerals) значения
- Через натуральные числа можно определить целые и вещественные

- Можно ввести рекурсию через т.н. Y-комбинатор
- Можно определить структуру данных — пару
- Можно определить логические (Church booleans) и натуральные (Church numerals) значения
- Через натуральные числа можно определить целые и вещественные

На выходе имеем Тьюринг-полный язык программирования

А потом уже был LISP

- 1958 год, Стив Рассел (Steve Russell)
- LISt Processor
- Lots of Irritating Stupid Parentheses
- Lost In a Sea of Parentheses
- etc.

- Всё — это линейный список

- Всё — это линейный список
- Программа это тоже линейный список

LISP: базовый синтаксис

- Всё — это линейный список
- Программа это тоже линейный список
- Всего два служебных символа — «(» и «)».

Первый элемент списка — это функция, дальше — её аргументы

```
(* (+ 1 1) (sin (/ pi 2)))
```

- ML — Робин Милнер (Robin Milner) — 1970e
 - Вывод типов (Type inference)
 - Алгебраические типы данных
- Miranda — Дэвид Тёрнер (David Turner) — 1985
 - Call-by-name evaluation strategy (lazy evaluation)

Дано: существует важная задача X и 25 способов её решения.

Дано: существует важная задача X и 25 способов её решения.

Лучшие умы человечества собираются вместе чтобы раз и навсегда представить один самый лучший способ.

Дано: существует важная задача X и 25 способов её решения.

Лучшие умы человечества собираются вместе чтобы раз и навсегда представить один самый лучший способ.

Долгие годы работы, кропотливый труд, радость свершения.

Дано: существует важная задача X и 25 способов её решения.

Лучшие умы человечества собираются вместе чтобы раз и навсегда представить один самый лучший способ.

Долгие годы работы, кропотливый труд, радость свершения.

Итог: существует важная задача X и **26** способов её решения.

Решение о создании языка было принято на конференции на тему функционального программирования и компьютерной архитектуры (FPCA '87) в Портленде, штат Орегон.

Haskell — 1990 — > 20 специалистов из TOP100 в области языков

- Вершина функциональной мысли на момент создания
- Чисто функциональный язык программирования
- Строгая статическая типизация
- Вывод типов
- Ленивое выполнение

- LISP продолжает развитие в виде BPCL, Racket, Clojure
- ML продолжает развитие в виде OCaml/F#

Последние годы: гибридизация

- Scala — EPFL — 2003
 - За основу взяли Java и Haskell
 - Попытка «поженить» ООП и ФП со стороны ООП
- F# — Microsoft — 2005
 - За основу взяли диалект ML под названием OCaml
 - Попытка «поженить» ООП и ФП со стороны ФП
- Swift/Rust/Kotlin и так далее
 - Все заявляют ФП как одну из основных «фишек»
 - На практике это неудачные попытки поиска золотой середины

Что же такое функциональное программирование

Что такое функция в программировании?

Что же такое функциональное программирование

Что такое функция в программировании?

Что такое функция в математике?

Что же такое функциональное программирование

Что такое функция в программировании?

Что такое функция в математике?

Чистая функция — это функция и в математическом, и в программном смысле

Что же такое функциональное программирование

Что такое функция в программировании?

Что такое функция в математике?

Чистая функция — это функция и в математическом, и в программном смысле

Можно ли написать что-то реально работающее только на чистых функциях?

Что же такое функциональное программирование

Чистая функция — это функция, не содержащая
побочных эффектов

Что же такое функциональное программирование

Чистая функция — это функция, не содержащая
побочных эффектов

Всё является функциями, причём чистыми

Что же такое функциональное программирование

Чистая функция — это функция, не содержащая *побочных эффектов*

Всё является функциями, причём чистыми

- Переменных нет, есть только константы
- Циклов нет — они бессмысленны без переменных

Побочные эффекты

- Любой ввод-вывод
- Модификация внешнего состояния (глобальные и локальные переменные)

Побочные эффекты

- Любой ввод-вывод
- Модификация внешнего состояния (глобальные и локальные переменные)

Являются ли два определения чистой функции тождественными?

Т.е. означает ли отсутствие побочных эффектов то, что для одних и тех же аргументов всегда будет выдан один и тот же результат?

Функции — полноценные значения

- Раз все значения являются функциями, то функции являются значениями
 - Функции можно передавать как параметры в другие функции
 - Функции можно создавать на лету (лямбда-абстракции) и возвращать как значения
- Функции высших порядков — функции, оперирующие функциями

Функции высших порядков: примеры

- `map` — отображение коллекции аргументов на коллекцию результатов
- `filter` — выделение элементов из коллекции по предикату

Другие ограничения чисто функционального кода

- Многие структуры данных (и, как следствие, алгоритмы) подразумевают изменяемое состояние
 - Простейший пример - массивы
 - Чтобы получить изменённый массив, нужно скопировать старый, что обычно неприемлемо
- Нужно использовать неизменяемые (immutable) структуры данных
 - Константные массивы
 - Односвязные списки
 - Деревья
 - Более сложные связные структуры
- Неизменяемость даёт много плюсов, но в среднем гораздо сложнее для понимания
 - «Дешёвая» параллельность

Вернёмся к λ -исчислению

- Две операции — abstraction & application
 - Т.е. объявление функций и их применение
- \Rightarrow Всё — это функция.

Вернёмся к λ -исчислению

- Две операции — abstraction & application
 - Т.е. объявление функций и их применение
- \Rightarrow Всё — это функция.
- На самом деле, всё — это функция с *одним аргументом*.

Вернёмся к λ -исчислению

- Две операции — abstraction & application
 - Т.е. объявление функций и их применение
- \Rightarrow Всё — это функция.
- На самом деле, всё — это функция с *одним аргументом*.
- Можно ли хоть что-то запрограммировать в таких условиях???

Позволяет писать функции с более чем одним параметром

Позволяет писать функции с более чем одним параметром

Функция с 2 параметрами \rightarrow Функция с 1 параметром, возвращающая функцию с 1 параметром

Изобретено Хаскеллом Карри (Haskell Curry), в честь которого и названо

$$\lambda x y. f x y === \lambda x. \lambda y. f x y$$

Принцип замыкания

Позволяет хранить данные в функциях

Принцип замыкания

Позволяет хранить данные в функциях

При объявлении функция *захватывает* все переменные, имеющиеся выше её объявления

Те из них, которые реально используются, называются *замыканием* (closure) этой функции

$$a = \dots$$
$$b = \lambda x. \lambda f. f(a\ x)$$

В функции *b* теперь всегда в некоем виде хранится функция *a*

$$pair = \lambda x. \lambda y. \lambda f. f\ x\ y$$
$$fst = \lambda p. p\ (\lambda x. \lambda y. x)$$
$$snd = \lambda p. p\ (\lambda x. \lambda y. y)$$

Пары Чёрча

$pair = \lambda x. \lambda y. \lambda f. f\ x\ y$

$fst = \lambda p. p\ (\lambda x. \lambda y. x)$

$snd = \lambda p. p\ (\lambda x. \lambda y. y)$

$oneAndTwo = pair\ \mathbf{a}\ \mathbf{b}$

$fst\ oneAndTwo \Rightarrow \mathbf{a}$

$snd\ oneAndTwo \Rightarrow \mathbf{b}$

Список — это либо пустой список (`nil`), либо пара из элемента и списка

$[1,2,3,4,5] \rightarrow (1, (2, (3, (4, (5, \text{nil}))))))$

Список — это либо пустой список (`nil`), либо пара из элемента и списка

$[1,2,3,4,5] \rightarrow (1, (2, (3, (4, (5, \text{nil}))))))$

Это соответствует линейному однонаправленному связному списку в С

Позволяют определить значения в рамках логики 1го порядка

Булеаны Чёрча

Позволяют определить значения в рамках логики 1го порядка

$$true = \lambda x. \lambda y. x$$

$$false = \lambda x. \lambda y. y$$

$$if = \lambda p. \lambda x. \lambda y. p x y$$

$$not = \lambda p. if p false true$$

Булеаны Чёрча

Позволяют определить значения в рамках логики 1го порядка

$$true = \lambda x. \lambda y. x$$

$$false = \lambda x. \lambda y. y$$

$$if = \lambda p. \lambda x. \lambda y. p x y$$

$$not = \lambda p. if p false true$$

$$if true \mathbf{a} \mathbf{b} \rightarrow \mathbf{a}$$

$$if false \mathbf{a} \mathbf{b} \rightarrow \mathbf{b}$$

Можно определить и прочие логические операции.

Позволяют определить целые числа

Позволяют определить целые числа

$$0 = \lambda f. \lambda x. x$$

$$1 = \lambda f. \lambda x. f x$$

$$2 = \lambda f. \lambda x. f (f x)$$

$$3 = \lambda f. \lambda x. f (f (f x))$$

и т.д.

Простейшее рекурсивное выражение —
омега-комбинатор

$$M = \lambda x. x x$$

$$\omega = M M$$

Простейшее рекурсивное выражение —
омега-комбинатор

$$M = \lambda x. x x$$

$$\omega = M M$$

Абсолютно бесполезен в реальном применении

Y-комбинатор

$$Y = \lambda f. (\lambda x. f(x x)) (\lambda x. f(x x))$$

$$Y \mathbf{x} === \mathbf{x} (Y \mathbf{x})$$

Y-комбинатор

$$Y = \lambda f. (\lambda x. f(x x)) (\lambda x. f(x x))$$

$$Y \mathbf{x} === \mathbf{x} (Y \mathbf{x})$$

Вместе с условным оператором позволяет реализовать полноценную рекурсию

λ -исчисление может использоваться как полноценный, Тьюринг-полный язык программирования. И это за десятки лет до появления программируемых компьютеров и до самого Тьюринга!

На практике интересно в основном математикам и исследователям в области языков.

В следующей лекции мы перейдём к настоящим языкам функционального программирования.

