

# Введение в функциональное программирование

Михаил Беяев

22 сентября 2015

- 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;  
}
```

Выполнение программы — это переход между состояниями

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

Как процедурные, так и объектные языки

Начиная от Fortran и Algol и заканчивая Java и C#

- 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?

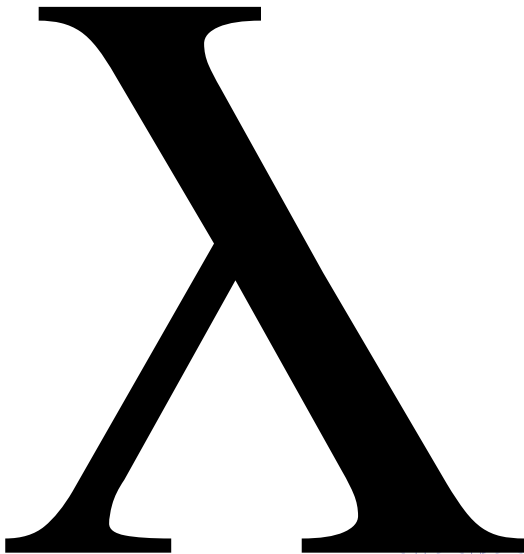
LISP

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

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

## Ещё немного истории

В начале была буква!



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  $\lambda$ -терм:

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



Алонзо Чёрч, 30е годы XX века

Два основных принципа — *аппликация* и *абстракция*

Абстракция через  $\lambda$ -терм:

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

Аппликация (применение функции к аргументу):

$f x$

Алонзо Чёрч, 30е годы XX века

Два основных принципа — *аппликация* и *абстракция*

Абстракция через  $\lambda$ -терм:

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

Аппликация (применение функции к аргументу):

$f x$

Вычисление = «переписывание» термов:

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

Это означает «заменить все вхождения  $x$  внутри  $y$  на  $a$ »

- Можно ввести рекурсию через т.н. Y-комбинатор
- Можно определить простую 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 простую 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

- 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) — 1970е
  - 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



- 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

# 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



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

- 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?

- 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

- `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
  - Персистентные структуры данных — хранение истории с экономией памяти
  - Более чёткая и формальная структура кода

Утверждение «всё — это функция» можно усилить:

«всё — это функция с числом параметров, равным 0 или 1»

Утверждение «всё — это функция» можно усилить:

«всё — это функция с числом параметров, равным 0 или 1»

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

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

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



$$\text{pair} = \lambda x. \lambda y. \lambda f. f x y$$
$$\text{fst} = \lambda p. p (\lambda x. \lambda y. x)$$
$$\text{snd} = \lambda p. p (\lambda x. \lambda y. y)$$
$$\text{oneAndTwo} = \text{pair } 1 \ 2$$
$$\text{fst oneAndTwo} \Rightarrow 1$$
$$\text{snd oneAndTwo} \Rightarrow 2$$

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

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

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

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

Это соответствует линейному однонаправленному связному списку в С

