Билеты к экзамену по функциональному программированию

Тамарин Вячеслав

January 25, 2021

1 Сравнение функционального и императивного подходов к программированию

1.1 Императивное программирование

Вычисление (программа) описывается в терминах **инструкций**, изменяющих **состояние** вычислителя.

- Инструкции выполняются последовательно;
- Состояние изменяется инструкциями **присваивания** значений изменяемым переменным;
- Если механизм условного исполнения (if, switch);
- Инструкции можно повторять с помощью циклов (while, for);
- Типы данных описываются с оглядкой на их физическое представление в памяти.

Такой стиль иногда называют стилем фон Неймана.

```
long factorial(int n) {
        long res = 1;
        for (int i = 0; i < n; i++)
                res *= i;
        return res;
}
```

Выполнение программы — переход вычислителя из начального состояния в конечное с помощью последовательных инструкций.

Часть конечного состояния может интерпретироваться как результат вычислений.

1.2 Функциональное программирование

Функциональная программа — **выражение**, ее выполнение — вычисление (**редукция**) этого выражения.

```
factorial n = if n == 0 then 1 else n * factorial(n-1)
```

Выполнение программы — редукция выражения с помощью **подстановки** определений функций в места их „вызова” с заменой формальных параметров на фактические.

```
factorial 3
-> if 3 == 0 then 1 else 3 * factorial (3 - 1)
-> ...
-> 3 * factorial (3 - 1)
-> 3 * if (3 - 1) == 0 then 1 else (3 - 1) * factorial ((3 - 1) - 1)
-> ...
-> 3 * 2 * 1 * 1
-> 6
```

- Нет состояний — нет изменяемых переменных;
- Нет переменных — нет присваивания;
- Нет циклов, так как нет различий между итерациями – состояниями
- Последовательность не важна, поскольку выражения независимы.
- Рекурсия — замена циклов;
- Функции высших порядков;
- Сопоставление с образцом;
- Все функции — чистые.

1.3 Сильные стороны ФП

- Регулярный синтаксис, удобство анализа кода;
- Мощная типизация, при этом можно практически не использовать типы в коде за счет эффективным алгоритмам вывода типов;
- Возможность генерации программ по набору свойств;
- Эффективная доказуемость свойств программ алгебраическими методами;
- Высокоурвневые оптимизации на базе эквивалентных преобразований.

2 Основы λ-исчисления. λ-термы, свободные и связанные переменные.

λ-исчисление — формальная система, лежащая в основе ФП. Разработано Алонзо Чёрчем в 1930-х для формализации и анализа понятия вычислимости.
В λ-исчислении тремя основными понятиями являются:

- применение (аппликация, application) — задает синтаксис применения функции к ее фактическим аргументам. Применение функции F к аргументу X записывается как FX;
- лямбда-абстрацкия (abstraction) — описывает синтаксис определения функции на основе параметризованного выражения, представляющего ее тело. Абстракция по x: $\lambda x.\, F$;
- редукция (reduction) — определяет отношение вычисления, основывающегося на подстановке фактических параметров вместо формальных.

Определение 1: λ-термы

Множество **λ-термов** Λ индуктивно строится из переменных $V = \{x, y, z, \ldots\}$ с помощью применения и абстракции:

$$\begin{aligned} x \in V &\implies x \in \Lambda \\ M, N \in \Lambda &\implies (MN) \in \Lambda \\ M \in \Lambda,\ x \in V &\implies (\lambda x.\, M) \in \Lambda \end{aligned}$$

В абстрактном синтаксисе

$$\Lambda ::= V \mid (\Lambda\Lambda) \mid (\lambda V.\, \Lambda).$$

Терм может быть *переменной* (например, $x \in \Lambda$), *аппликацией* (например, $(xy) \in \Lambda$) или *лямбда-абстракцией* (например, $\lambda x.\, ((x(yx))) \in \Lambda$).

Определение 2: Подтермы

Множество **подтермов** терма Q определяется индуктивно:

$$\begin{aligned} \mathtt{subterms}(x) &= \{x\} \\ \mathtt{subterms}(MN) &= \{MN\} \cup \mathtt{subterms}(M) \cup \mathtt{subterms}(N) \\ \mathtt{subterms}(\lambda x.\, M) &= \{\lambda x.\, M\} \cup \mathtt{subterms}(M) \end{aligned}$$

2.1 Соглашения

Общеприняты следующие соглашения для термов:

- Внешние скобки опускаются
- Применение ассоциативно *влево*:
$$FXYZ \equiv (((FX)Y)Z).$$
- Абстракция ассоциативна *вправо*:
$$\lambda xyz.\, M \equiv (\lambda x.\, (\lambda y.\, (\lambda z.\, M))).$$
- Тело абстракции простирается вправо насколько это возможно:
$$\lambda x.\, MNK \equiv \lambda x.\, (MNK).$$

Определение 3: Свободные переменные

Множество **свободных переменных** в терме Q ($FV(Q)$) определяется следующим образом:

$$\begin{aligned} FV(x) &= \{x\} \\ FV(MN) &= FV(M) \cup FV(N) \\ FV(\lambda x.\, M &= FV(M) \setminus \{x\} \end{aligned}$$

Определение 4: Связвнные переменные

Множество **связанных переменных** в терме Q ($BV(Q)$) определяется следующим образом:

$$\begin{aligned} BV(x) &= \varnothing \\ BV(MN) &= BV(M) \cup BV(N) \\ BV(\lambda x.\, M &= BV(M) \cup \{x\} \end{aligned}$$

2.2 Комбинаторы

Определение 5: Комбинатор

M — **замкнутый λ-терм (комбинатор)**, если $FV(M) = \varnothing$.
Множество замкнутых λ-термов обозначается Λ^0.

1. $I = \lambda x.\, x$
2. $\omega = \lambda x.\, xx$
3. $\Omega = \omega\omega = (\lambda x.\, xx)\,(\lambda x.\, xx)$
4. $K = \lambda xy.\, x$
5. $K_* = \lambda xy.\, y$
6. $C = \lambda fxy.\, fyx$
7. $B = \lambda fgx.\, f(gx)$
8. $S = \lambda fgx.\, fx(gx)$

3 Подстановка λ-терма. Лемма подстановки.

Определение 6: Подстановка

Подстановка терма N вместо *свободных* вхождений переменной x в терм M ($[x \mapsto N]M$) задается следующими правилами:

$$\begin{array}{lcll}
[x \mapsto N]x & = & N & \\
[x \mapsto N]y & = & y & \\
[x \mapsto N]PQ & = & ([x \mapsto N]P)\,([x \mapsto N]Q) & \\
[x \mapsto N]\lambda x.\, P & = & \lambda x.\, P & \\
[x \mapsto N]\lambda y.\, P & = & \lambda y.\, [x \mapsto N]P, & \text{если } y \notin FV(N) \\
[x \mapsto N]\lambda y.\, P & = & \lambda x.\, [y \mapsto z]P\,([y \mapsto z]P)\,, & \text{если } y \in FV(N)
\end{array}$$

Предполагается, что x и y различны, а z — „свежая” переменная, то есть $z \notin FV(P) \cup FV(N)$.

Пример 1 (Пример подстановки).

$$[x \mapsto uv]\,((\lambda x.\, (\lambda x.\, xz)x)\, x) = (\lambda x.\, (\lambda x.\, xz)x)\,(uv).$$

Лемма 1 (подстановки). Пусть $M, N, L \in \Lambda$, $x \not\equiv y$ и $x \notin FV(L)$. Тогда

$$[y \mapsto L][x \mapsto N]M = [x \mapsto [y \mapsto L]N][y \mapsto L]M.$$

4 α- и β-конверсии. η-конверсия и экстенсиональная эквивалентность.

4.1 α-эквивалентность

Позволяет менять переменную на „свежую”. Главная аксиома α-преобразования:

$$\lambda x.\, M =_\alpha \lambda y.\, [x \mapsto y]M, \text{ если } y \notin FV(M). \qquad \text{(правило } \alpha\text{)}$$

Аналогично β-эквивалентности можно определить совместимость.

4.2 β-эквивалентность

Основная схема аксиом: для любых $M, N \in \Lambda$:

$$(\lambda x.\, M)N =_\beta [x \mapsto N]M. \qquad \text{(правило } \beta\text{)}$$

Чтобы сделать $=_\beta$ отношением эквивалентности добавим логические аксиомы и правила:

$$\begin{array}{lcl}
 & & M =_\beta M \\
M =_\beta N & \Rightarrow & M =_\beta N \\
M =_\beta N,\ N =_\beta L & \Rightarrow & M =_\beta L
\end{array}$$

Также определим правила совместимости:

$$\begin{array}{l}
M =_\beta N \Rightarrow ZM =_\beta ZN \\
M =_\beta N \Rightarrow MZ =_\beta NZ \\
M =_\beta N \Rightarrow \lambda x.\, M =_\beta \lambda x.\, N \qquad \text{(правило } \xi\text{)}
\end{array}$$

Если $M =_\beta N$ доказуемо в λ-исчислении, пишут

$$\lambda \vdash M =_\beta N.$$

4.3 η-эквивалентность

Схема аксиом η-преобразования:

$$\lambda x.\, Mx =_\eta M, \text{ если } x \notin FV(M). \qquad \text{(правило } \eta\text{)}$$

Аналогично можем определить правила и совместимость, как для β.

Смысл этой эквивалентности в том, что поведение данных двух термов одинаково; для произвольного N будет верно

$$(\lambda x.\, M)N =_\beta MN.$$

Принцип экстенсиональности Две функции считаются **экстенсионально эквивалентными**, если они дают одинаковый результат при одинаковом входе:

$$\forall N\colon FN =_\beta GN.$$

Выберем $u \notin FV(F) \cup FV(G)$, теперь

$$Fy =_\beta Gy \Longrightarrow \lambda y.\, Fy =_\beta \lambda y.\, Gy \Longrightarrow F =_{\beta\eta} G$$

5 Кодирование булевых значений, кортежей в чистом бестиповом λ-исчислении.

5.1 Булевы значения

$$\begin{aligned} \texttt{tru} &\equiv \lambda tf.\, t \\ \texttt{fls} &\equiv \lambda tf.\, f \end{aligned}$$

5.2 Булевы операции

$$\begin{aligned} \texttt{if} &\equiv \lambda bxy.\, bxy \\ \texttt{not} &\equiv \lambda b.\, b\,\texttt{fls}\,\texttt{tru} \\ \texttt{and} &\equiv \lambda xy.\, xy\,\texttt{fls} \\ \texttt{or} &\equiv \lambda xy.\, x\,\texttt{tru}\,y \end{aligned}$$

5.3 Кортежи

Пара и стандартные операции:

$$\begin{aligned} \texttt{pair} &\equiv \lambda xyf.\, fxy \\ \texttt{fst} &\equiv \lambda p.\, p\,\texttt{tru} \\ \texttt{snd} &\equiv \lambda p.\, p\,\texttt{fls} \end{aligned}$$

6 Кодирование чисел Чёрча в чистом бестиповом λ-исчислении.

$$\begin{aligned} 0 &\equiv \lambda sz.\, z \\ 1 &\equiv \lambda sz.\, sz \\ 2 &\equiv \lambda sz.\, s(sz) \\ 3 &\equiv \lambda sz.\, s(s(sz)) \\ 4 &\equiv \lambda sz.\, s(s(s(sz))) \\ &\dots \end{aligned}$$

Определим $F^n(X)$, где $n \in \mathbb{N}$, $F, Z \in \Lambda$:

$$\begin{aligned} F^0(X) &\equiv X \\ F^{n+1}(X) &\equiv F(F^n(X)) \end{aligned}$$

Теперь число Чёрча принимает следующий вид

$$n \equiv \lambda\, sz.\ s^n(z).$$

Функция проверки на ноль:

$$\mathtt{iszero} \equiv \lambda\, n.\ n(\lambda\, x.\ \mathtt{fls})\,\mathtt{tru}.$$

Переход к следующему числу:

$$\mathtt{succ} \equiv \lambda\, nsz.\ s(nsz).$$

Сложение:

$$\mathtt{plus} \equiv \lambda\, mnsz.\ ms(nsz).$$

Умножение:

$$\begin{aligned}\mathtt{mult} &\equiv \lambda\, mn.\ m(\mathtt{plus}\, n)0\\ \mathtt{mult} &\equiv \lambda\, mnsz.\ m(ns)x\end{aligned}$$

7 Теорема о неподвижной точке. Y-комбинатор.

7.1 Решение уравнений на термы

Например, хотим найти F такой, что $\forall M, N, L\colon \lambda \vdash FMNL = ML(NL)$.

$$\begin{aligned}FMNL &= ML(NL)\\ FMNL &= (\lambda\, l.\ Ml(Nl))L\\ FMN &= \lambda\, l.\ Ml(Nl)\\ FM &= \lambda\, n.\ \lambda\, l.\ Ml(nl)\\ F &= \lambda\, mnl.\ ml(nl)\end{aligned}$$

А что делать, если уравнение рекурсивное? Например, $FM = MF$.

7.2 Теоремы о неподвижной точке

Теорема 1. Для любого λ-терма F существует неподвижная точка:

$$\forall F \in \Lambda\ \exists X \in \Lambda\colon \lambda \vdash FX = X.$$

Proof. Пусть $W \equiv \lambda\, x.\ F(xx)$ и $X \equiv WW$. Тогда

$$X \equiv WW \equiv (\lambda\, x.\ F(xx))W = F(WW) \equiv FX.$$

□

Теорема 2. Существует комбинатор неподвижной точки **Y** такой, что

$$\forall F\colon F(\mathbf{Y}F) = \mathbf{Y}F.$$

Proof. Пусть $\mathbf{Y} = \lambda\, f.\ (\lambda\, x.\ f(xx))(\lambda\, x.\ f(xx))$. Тогда

$$\mathbf{Y}F \equiv (\lambda\, x.\ F(xx))(\lambda\, x.\ F(xx)) = F(\underbrace{(\lambda\, x.\ F(xx))(\lambda\, x.\ F(xx))}_{\mathbf{Y}F}) \equiv F(\mathbf{Y}F).$$

□

7.3 Рекурсия

Y-комбинатор позволяет ввести рекурсию в λ-исчисление. Например, можно задать факториал рекурсивно:

$$\mathtt{fac} = \lambda n.\ \mathtt{if}(\mathtt{iszero}\, n)1(\mathtt{mult}\, n(\mathtt{fac}(\mathtt{pred}\, n))).$$

Можно переписать:

$$\mathtt{fac} = \underbrace{(\lambda f n.\ \mathtt{if}(\mathtt{iszero}\, n)1(\mathtt{mult}\, n(f(\mathtt{pred}\, n))))}_{\mathtt{fac}'}\, \mathtt{fac}\,.$$

Тогда $\mathtt{fac}$ — неподвижная точка для $\mathtt{fac}'$, поэтому $\mathtt{fac} = \mathsf{Y}\,\mathtt{fac}'$.

8 Редексы. Одношаговая и многошаговая редукция. Нормальная форма. Редукционные графы.

Если мы хотим доказать равенство, то эффективным способом будет сокращение всех редексов:

$$\begin{aligned}
\mathsf{KI} &\equiv (\lambda xy.\ x)\,(\lambda z.\ z) = \lambda yz.\ z \\
\mathsf{IIK}_* &\equiv (\lambda x.\ x)\mathsf{IK}_* = \mathsf{IK}_* = (\lambda x.\ x)(\lambda yz.\ z) = \lambda yz.\ z
\end{aligned}$$

Но как доказать неравенство?

8.1 Редукция

Определение 7: Совместимое отношение

Бинарное отношение $\mathcal{R}$ над Λ называют **совместимым** (с операциями λ-исчисления), если для всех $M, N, Z \in \Lambda$:

$$\begin{aligned}
M\ \mathcal{R}\ N \quad \Rightarrow \qquad (ZM)\ &\mathcal{R}\ (ZN), \\
(MZ)\ &\mathcal{R}\ (NZ), \\
(\lambda x.\ M)\ &\mathcal{R}\ (\lambda x.\ N)
\end{aligned}$$

Определение 8

Совместимое отношение эквивалентности называют **отношением конгруэнтоности** над Λ.
Совместимое, рефлексивное и транзитивное отношение называют **отношением редукции** над Λ.

Определение 9

Бинарное отношение **β-редукции за один шаг** $\to_\beta$ над Λ:

$$\begin{aligned}
(\lambda x.\ M)N \quad &\longrightarrow_\beta \quad [x \mapsto N]M \\
M \longrightarrow_\beta N \quad &\Longrightarrow \quad ZM \longrightarrow_\beta ZN \\
M \longrightarrow_\beta N \quad &\Longrightarrow \quad MZ \longrightarrow_\beta NZ \\
M \longrightarrow_\beta N \quad &\Longrightarrow \quad \lambda x.\ M \longrightarrow_\beta \lambda x.\ N
\end{aligned}$$

Это правило и совместимость.

Определение 10: β-редукция

Бинарное отношение **β-редукции** $\twoheadrightarrow_\beta$ над Λ определяется индуктивно:

$$\begin{aligned}
M &\twoheadrightarrow_\beta M && \text{(refl)} \\
M \longrightarrow_\beta N \ \Rightarrow\ M &\twoheadrightarrow_\beta N && \text{(ini)} \\
M \twoheadrightarrow_\beta N,\ N \twoheadrightarrow_\beta L \ \Rightarrow\ M &\twoheadrightarrow_\beta L && \text{(trans)}
\end{aligned}$$

Это отношение — транзитивное, рефлексивное замыкание $\longrightarrow_\beta$, поэтому является отношением редукции.

Определение 11: Конвертируемость

Бинарное отношение **конвертируемости** $=_\beta$ над Λ определяется индуктивно:

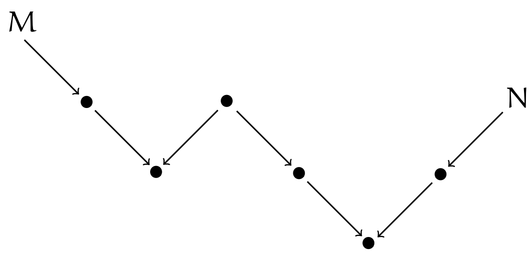
$$M \twoheadrightarrow_\beta N \Rightarrow M =_\beta N \tag{ini}$$
$$M =_\beta N \Rightarrow N =_\beta M \tag{sym}$$
$$M =_\beta N,\ N =_\beta L \Rightarrow M =_\beta L \tag{trans}$$

Это отношение является отношением конгруэнтности.

Утверждение 1. $M =_\beta N \iff \lambda \vdash M = N$.

Proof. Индукция по количеству шагов, расписываем по определению. □

Два терма M, N связаны отношением $=_\beta$, если есть связывающая цепочка стрелок $\longrightarrow_\beta$:



Редукционный граф для терма $M \in \Lambda$ (нотация $G_\beta(M)$) — ориентированный мультиграф с вершинами в $\{N \mid M \twoheadrightarrow_\beta N\}$, ребро проводится между N и L, если $N \longrightarrow_\beta L$.

Отношение β-эквивалентности не является разрешимым в общем случае (то есть для пары термов не существует универсального алгоритма, проверяющего эквивалентность).

8.2 Нормальная форма

Это то, что лежит в самом низу на картинке.

Определение 12

λ-терм M *находится* в **β-нормальной форме** (β-NF), если в нем нет подтермов, являющихся β-редексами.
λ-терм M *имеет* β-нормальную форму, если для некоторого N выполняется $M =_\beta N$ и N находится в β-NF.

Пример 2.

- $\lambda xy.\, x(\lambda z.\, zx)y$ находится в β-NF
- $(\lambda x.\, xx)y$ не находится в β-NF, но имеет в качестве β-NFтерм yy
- Ω не имеет нормальной формы

9 Теорема Чёрча-Россера и ее следствия.

Теорема 3 (Черч, Россер). Если $M \twoheadrightarrow_\beta N$, $M \twoheadrightarrow_\beta K$, то существует такой L, что $N \twoheadrightarrow_\beta L$ и $K \twoheadrightarrow_\beta L$.

Иначе говоря, β-редукция обладает *свойством ромба* или *конфлюентностью*:

$$\begin{tikzcd}
 & M \arrow[dl, two heads] \arrow[dr, two heads] & \\
N \arrow[dr, dotted, two heads] & & K \arrow[dl, dotted, two heads] \\
 & L &
\end{tikzcd}$$

Следствие 1 (о существовании общего редукта). Если $M =_\beta N$, то существует L такой, что $M \twoheadrightarrow_\beta L$ и $N \twoheadrightarrow_\beta L$.

Proof. Индукция по генерации $=_\beta$. Разберем три случая вывода $M =_\beta N$:

$M \twoheadrightarrow_\beta N$. Возьмем $L = N$.

$N =_\beta M$. По предположению индукции есть общий β-редукт L_1 для N и M. Можем взять $L = L_1$.

$M =_\beta N'$, $N' =_\beta N$. Тогда

$$\begin{tikzcd}
M \arrow[dr, two heads] & \text{(по ПИ)} & N' \arrow[dl, two heads] \arrow[dr, two heads] & \text{(по ПИ)} & N \arrow[dl, two heads] \\
 & L_1 \arrow[dr, dotted, two heads] & \text{(CR)} & L_2 \arrow[dl, dotted, two heads] & \\
 & & L & &
\end{tikzcd}$$

$\square$

Следствие 2 (о редуцируемости к β-NF). Если M имеет N в качестве β-NF, то $M \twoheadrightarrow_\beta N$.

Следствие 3 (о единственности β-NF). λ-терм имеет не более одной β-NF.

10 Стратегии редукции. Теорема о нормализации. Механизмы вызова в функциональных языках.

Рассмотрим три варианта терма:

- Переменная v, редукция завершена, ничего интересного
- Абстракция $\lambda x.\ M$, просто редуцируем M
- Аппликация MN. Разбираем аппликацию влево (в M) до не-аппликации. Пока получаем аппликацию, разделяем дальше. Два варианта остановки:
  - Переменная. Тогда мы получили такой вид
    $$(\ldots((vN_1)N_2)\ldots N_k).$$
    Теперь нам нужно просто проредуцировать каждый N_i, общая структура от этого не изменится.
  - Абстракция. Получили:
    $$(\ldots(((\lambda x.\ M')N_1)N_2)\ldots N_k).$$
    Будем работать дальше. Здесь есть две стратегии:
    * **Нормальная стратегия**: сразу сокращаем редекс $(\lambda x.\ M)N_1$.

* **Аппликативная стратегия**: сначала редуцируем отдельно все N_i слева направо до нормальной формы N_j', а уже потом сокращаем редекс $(\lambda x.\, M)N_1'$.
Также можно редуцировать только N_1 и сразу сократить.

Разбор терма можно представить в виде дерева, где узлы @ задают аппликацию, а узлы λ — абстракцию.

Теорема 4. Лямбда-терм может иметь одну из двух форм:

$$\lambda \vec{x}.\, y\vec{N} \equiv \lambda x_1 x_2 \dots x_n.\, yN_1, N_2, \dots N_k, \quad n, k \geqslant 0 \qquad \text{(HNF)}$$

$$\lambda \vec{x}.\, (\lambda z.\, M)\vec{N} \equiv \lambda x_1 x_2 \dots x_n.\, (\lambda z.\, M)N_1, N_2, \dots N_k, \quad n \geqslant 0, k > 0$$

Определение 13

Головная нормальная форма — форма следующего вида:

$$\lambda \vec{x}.\, y\vec{N} \equiv \lambda x_1 x_2 \dots x_n.\, yN_1, N_2, \dots N_k, \quad n, k \geqslant 0. \qquad \text{(HNF)}$$

Слабая головная нормальная форма (WHNF) — это HFN или лямбда-абстракция (то есть не редекс на верхнем уровне).
Переменная y называется **головной переменной**. Редекс $(\lambda z.M)N_1$ называется **головным редексом**.

Переменная y может совпадать с одним из x_i.

В Haskell вычисления как раз форсируются до слабой нормальной формы.

Теорема 5 (о нормализации)**.** Если терм M имеет нормальную форму, то последовательное сокращение самого левого внешнего редекса приводит к этой нормальной форме.

То есть, если HNF есть, то нормальная стратегия гарантировано приводит к ней.

10.1 Механизмы вызова в ФЯ

Аппликативная стратегия более эффективная, хоть и не всегда приводит к HNF, поэтому применяется во многих языках программирования.

Пусть N — очень большой терм, который вычисляется сутки. Если мы запустим нормальную стратегию на

$$(\lambda x.\, Fx(Gx)x)N,$$

получим $FN(GN)N$ и N придется в дальнейших редукциях сократить трижды. Но для такого нормальная стратегия не вычислит N ни разу:

$$(\lambda xy.\, y)N \longrightarrow_\beta \lambda y.\, y.$$

Аппликативная стратегия будет вычислять N только один раз в каждом из примеров.

В „ленивых” языках программирования (Haskell, Clean) используется похожая на нормальную стратегия, а для решения проблем с эффективностью используют механизм *разделения*: вместо непосредственной подстановки терма подставляют указатель на терм, который хранится в памяти как *отложенное вычисление*.

Механизм вызова в функциональных языках:

- *вызов по значению* — аппликативый порядок редукций до WHNF
- *вызов по имени* — нормальный порядок редукций до WHNT
- *вызов по необходимости* — „вызов по имени” с разделением.

11 Функция предшествования для чисел Черча. Комбинатор примитивной рекурсии.

11.1 Функция предшествования

Введем вспомогательные функции:

$$\begin{aligned}
\mathtt{zp} &\equiv \mathtt{pair}\,0 0 \\
\mathtt{sp} &\equiv \lambda\, p.\, \mathtt{pair}(\mathtt{snd}\, p)(\mathtt{succ}(\mathtt{snd}\, p)))
\end{aligned}$$

Вторая функция позволяет, приняв пару $(,j)$, вернуть пару $(j, j+1)$:

$$\mathtt{sp}(\mathtt{pair}\, ij) = \mathtt{pair}\, j(j+1).$$