# Теория алгоритмов

## Дополнительные разделы

Куценко Дмитрий Александрович

12 декабря 2010 г.

---

## Содержание



---

# 1. Рекурсивные функции

Термин *рекурсивные функции* в теории вычислимости используют для обозначения трёх множеств функций:

1) *примитивно рекурсивные функции;*

2) *общерекурсивные функции;*

3) *частично рекурсивные функции*[1)].

Последние совпадают с множеством вычислимых по Тьюрингу[2)] функций. Определения этих трёх классов сильно связаны. Они были введены Гёделем[3)] с целью формализации понятия вычислимости.

Множество частично рекурсивных функций ($\mathscr{F}_{\text{ЧРФ}}$) включает в себя множество общерекурсивных функций ($\mathscr{F}_{\text{ОРФ}}$), а множество общерекурсивных функций включает в себя множество примитивно рекурсивные функций ($\mathscr{F}_{\text{ПРФ}}$):

$$\mathscr{F}_{\text{ПРФ}} \subset \mathscr{F}_{\text{ОРФ}} \subset \mathscr{F}_{\text{ЧРФ}}.$$

Частично рекурсивные функции иногда называют просто *рекурсивными функциями.*

---

[1)]Термины «частично рекурсивная функция» и «общерекурсивная функция» прижились в силу исторических причин. По сути они являются результатом неточного перевода английских терминов partial recursive function и total recursive function, которые по смыслу более правильно переводить как «частично определённые рекурсивные функции» и «везде определённые рекурсивные функции» соответственно.

[2)]Алан Матисон Тьюринг (Alan Mathison Turing; 1912—1954) — английский математик, логик и криптограф.

[3)]Курт Фридрих Гёдель (Kurt Friedrich Gödel; 1906—1978) — австрийский логик, математик и философ.

## 1.1. Примитивно рекурсивные функции

### Определение

Функция от конечного числа аргументов называется *примитивно рекурсивной,* если её можно получить применением конечного числа операторов подстановки и примитивной рекурсии, исходя лишь из базовых примитивно рекурсивных функций.

В качестве *базовых примитивно рекурсивных функций* обычно выбирают функции следующих трёх видов:

1. *Нуль-функция* $O$ — функция, всегда возвращающая нуль, т. е.

$$O(x) = 0.$$

2. *Функция следования* $S$ одного переменного, сопоставляющая любому натуральному числу $x$ непосредственно следующее за ним натуральное число $x + 1$:

$$S(x) = x + 1.$$

3. *Функции-проекторы* $I_n^m$, где $1 \leqslant m \leqslant n$, от $n$ переменных, сопоставляющие любому упорядоченному набору $x_1, \ldots, x_n$ натуральных чисел число $x_m$ из этого набора:

$$I_n^m(x_1, \ldots, x_m, \ldots, x_n) = x_m.$$

Операторы подстановки и примитивной рекурсии определяются следующим образом:

1. *Оператор подстановки (суперпозиция функций).* Пусть $f$ — функция от $m$ переменных, а $g_1, \ldots, g_m$ — упорядоченный набор функций от $n$ переменных каждая. Тогда результатом подстановки функций $g_1, \ldots, g_m$ в функцию $f$ называется функция $h$ от $n$ переменных, сопоставляющая любому упорядоченному набору $x_1, \ldots, x_n$ натуральных чисел число

$$h(x_1, \ldots, x_n) = f\big(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n)\big).$$

   Если мы каким-либо образом умеем вычислять функции $g_1, \ldots, g_m$ и $f$, то функция $h$ может быть вычислена с помощью алгоритма, изображённого на рис. 1.

   Ясно, что если все функции $g_1, \ldots, g_m$ и $f$ всюду определены, то функция $h$ всюду определена. Функция $h$ будет не всюду определённой, если хотя бы одна из функций $g_1, \ldots, g_m$ не всюду определена, или если можно найти такие значения аргументов $a_1, \ldots, a_n$, что $b_i = g_i(a_1, \ldots, a_n)$, но $f(b_1, \ldots, b_m)$ не определено $(i = 1, \ldots, m)$.

2. *Оператор примитивной рекурсии.* Пусть $f$ — функция от $n$ переменных, а $g$ — функция от $n + 2$ переменных. Тогда результатом применения оператора примитивной рекурсии к паре функций $f$ и $g$ называется функция $h$ от $n + 1$ переменной вида

$$\begin{cases} h(x_1, \ldots, x_n,\ 0) = f(x_1, \ldots, x_n), \\ h(x_1, \ldots, x_n,\ y + 1) = g\big(x_1, \ldots, x_n,\ y,\ h(x_1, \ldots, x_n,\ y)\big). \end{cases}$$

   Если мы можем вычислить функции $f$ и $g$, то функция $h$ вычисляется согласно алгоритму, изображённому на рис. 2.

   Очевидно, что если функции $f$ и $g$ всюду определены, то будет всюду определена и функция $h$.

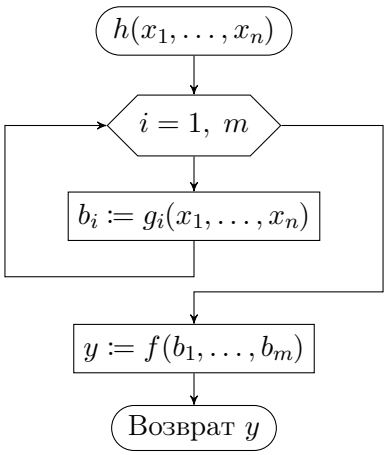


**Рис. 1.** Алгоритм вычисления функции $h$, являющейся результатом подстановки функций $g_1, \ldots, g_m$ в функцию $f$.

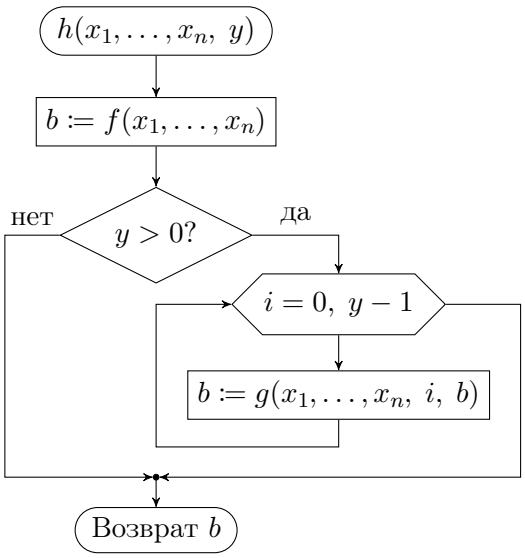


**Рис. 2.** Алгоритм вычисления функции $h$, являющейся результатом применения оператора примитивной рекурсии к паре функций $f$ и $g$.

Большинство функций из теории чисел являются примитивно рекурсивными, например, сложение, целочисленное деление, факториал, возведение в степень. Фактически, трудно разработать числовую функцию от конечного числа целочисленных аргументов, которая не является примитивно рекурсивный, хотя и такие тоже известны.

### Примеры

Укажем на ряд широко известных арифметических функций, являющихся примитивно рекурсивными.

1. *Функция сложения двух натуральных чисел*

$$\Sigma(x,\ y) = x + y$$

может быть рассмотрена в качестве примитивно рекурсивной функции двух переменных, получаемой в результате применения оператора примитивной рекурсии к функциям $I_1^1$ и $F$, вторая из которых получается подстановкой базовой функции $I_3^3$ в базовую функцию $S$:

$$\begin{cases} \Sigma(x,\ 0) = I_1^1(x), \\ \Sigma(x,\ y+1) = F\big(x,\ y,\ \Sigma(x,\ y)\big); \end{cases}$$
$$F(x,\ y,\ z) = S\big(I_3^3(x,\ y,\ z)\big).$$

2. *Функция умножения двух натуральных чисел*

$$\Pi(x,\ y) = x \cdot y$$

может быть рассмотрена в качестве примитивно рекурсивной функции двух переменных, получаемой в результате применения оператора примитивной рекурсии к функциям $O$ и $G$, вторая из которых получается подстановкой базовых функций $I_3^1$ и $I_3^3$ в функцию сложения:

$$\begin{cases} \Pi(x,\ 0) = O(x), \\ \Pi(x,\ y+1) = G\big(x,\ y,\ \Pi(x,\ y)\big); \end{cases}$$
$$G(x,\ y,\ z) = \Sigma\big(I_3^1(x,\ y,\ z),\ I_3^3(x,\ y,\ z)\big).$$

3. *Функция симметрической разности (абсолютной величины разности) двух натуральных чисел*

$$\Delta(x,\ y) = |x - y|$$

может быть рассмотрена в качестве примитивно рекурсивной функции двух переменных, получаемой в результате применения следующих подстановок и примитивных рекурсий:

$$\Delta(x,\ y) = \Sigma\big(\Delta_1(x,\ y),\ \Delta_2(x,\ y)\big);$$
$$\Delta_2(x,\ y) = \Delta_1\big(I_2^2(x,\ y),\ I_2^1(x,\ y)\big);$$
$$\begin{cases} \Delta_1(x,\ 0) = I_1^1(x), \\ \Delta_1(x,\ y+1) = H\big(x,\ y,\ \Delta_1(x,\ y)\big); \end{cases}$$
$$H(x,\ y,\ z) = P\big(I_3^3(x,\ y,\ z)\big);$$
$$\begin{cases} P(0) = O(0), \\ P(y+1) = I_2^1\big(y,\ p(y)\big). \end{cases}$$

## 1.2. Частично рекурсивные функции

*Частично рекурсивные функции* — это рекурсивные функции, определённые на части множества возможных аргументов. Они определяются аналогично примитивно рекурсивным, только к двум операторам подстановки и примитивной рекурсии добавляется ещё оператор минимизации аргумента.

Пусть задана некоторая функция $f(x_1, \ldots, x_{n-1},\ y)$. Зафиксируем значения $x_1, \ldots, x_{n-1}$ и выясним, при каком $y$ функция $f(x_1, \ldots, x_{n-1},\ y) = 0$. Более сложной задачей является отыскание для функции $f(x_1, \ldots, x_{n-1},\ y)$ и фиксированных $x_1, \ldots, x_{n-1}$ *наименьшего* из тех значений $y$, при которых функция $f(x_1, \ldots, x_{n-1},\ y) = 0$. Эта задача описывается с помощью оператора минимизации аргумента.

3. *Оператор минимизации аргумента.* Пусть $f$ — функция от $n$ натуральных переменных. Тогда результатом применения оператора минимизации аргумента к функции $f$ называется функция $h$ от $n-1$ переменной, задаваемая следующим определением:

$$h(x_1, \ldots, x_{n-1}) = \min_{f(x_1,\ldots,x_n,\, y)=0} \{y\},$$

то есть функция $h$ возвращает минимальное значение последнего аргумента функции $f$, при котором значение $f$ равно 0.

Для вычисления функции $h$ можно предложить переборный алгоритм, изображённый на рис. 3.

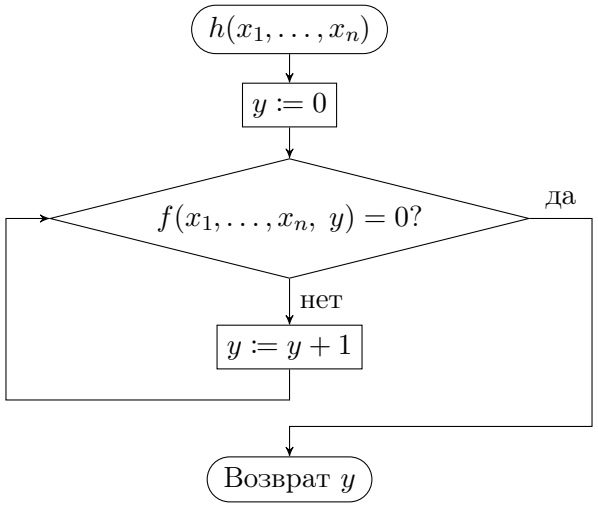


**Рис. 3.** Алгоритм вычисления функции $h$, являющейся результатом применения оператора минимизации аргумента к функции $f$.

Интересно отметить аналогию с императивными языками программирования. Примитивно рекурсивные функции соответствуют программным функциям, в которых используется только арифметические операции, а также условный оператор и оператор цикла с фиксированным количеством шагов (цикл `for`).

Если же программист начинает использовать оператор цикла `while`, в котором число итераций заранее неизвестно, и в принципе, может быть бесконечным, то он переходит в класс частично рекурсивных функций.

Важным моментом является то, что частично рекурсивные функции для некоторых значений аргумента могут быть не определены, так как оператор минимизации аргумента не всегда корректно определён, а именно, функция $f$ может быть не равной нулю ни при каких значениях аргументов.

Собственно, оттого, что частично рекурсивные функции могут иметь корректно определённое значение лишь на части аргументов, и пошло их название.

С точки зрения программиста результатом частично рекурсивной функции может быть не только число, но и исключение (exception) или «зависание», соответствующее неопределённому значению.

### Тезис Чёрча

Тезис Чёрча[4)] формулируется следующим образом. *Каждая интуитивно вычислимая функция является частично рекурсивной.*

Этот тезис нельзя строго ни доказать, ни опровергнуть, так как он связывает нестрогое математическое понятие «интуитивно вычислимая функция» со строгим математическим понятием «частично рекурсивная функция».

Но тезис может быть опровергнут, если построить пример функции интуитивно вычислимой, но не являющейся частично рекурсивной. Пока таких примеров построено не было.

Тезис Чёрча является естественнонаучным фактом, который подтверждается опытом, накопленным математикой за весь период её развития. Тезис Чёрча является достаточным средством, чтобы придать необходимую точность формулировкам алгоритмических проблем, он делает возможным доказательство их неразрешимости.

### Примеры

1. *Частичная функция вычитания двух натуральных чисел*

   $$M(x,\ y) = x - y$$

   получается с помощью применения оператора минимизации:

   $$M(x,\ y) = \min_{f(x,\ y,\ z)=0}\{z\},$$

   где $f(x,\ y,\ z) = \Delta\Big(I_3^1(x,y,z),\ \Sigma\big(I_3^2(x,y,z),\ I_3^3(x,y,z)\big)\Big)$.

   Очевидно, что функция $M$ неприменима, когда $x < y$.

2. *Частичная функция деления двух натуральных чисел*

   $$D(x,\ y) = \frac{x}{y}$$

   получается с помощью применения оператора минимизации

   $$D(x,\ y) = \min_{g(x,\ y,\ z)=0}\{z\},$$

   где $g(x,\ y,\ z) = \Delta\Big(I_3^1(x,y,z),\ \Pi\big(I_3^2(x,y,z),\ I_3^3(x,y,z)\big)\Big)$.

   Очевидно, что функция $D$ неприменима, когда $y = 0$ или $x \bmod y \neq 0$.

## 1.3. Общерекурсивные функции

*Общерекурсивные функции* — это подмножество частично рекурсивных функций, определённых для всех значений аргументов. К сожалению, задача определения того, является ли частично рекурсивная функция с данным описанием общерекурсивной или нет, алгоритмически неразрешима.

[4)] Алонзо Чёрч (Alonzo Church; 1903—1995) — американский математик и логик, внёсший значительный вклад в основы информатики.

### Свойства

Легко понять, что любая примитивно рекурсивная функция является частично рекурсивной, так как по определению операторы для построения частично рекурсивных функций включают в себя операторы для построения примитивно рекурсивных функций.

Также понятно, что примитивно рекурсивная функция определена везде и поэтому является общерекурсивной функцией (у примитивно рекурсивной функции нет повода «зависать», так как при её построении используются операторы, определяющие везде определённые функции).

### Примеры

Довольно сложно доказать существование и привести пример общерекурсивной функции, не являющейся примитивно рекурсивной. Идея состоит в том, чтобы построить такую вычислимую функцию, которая растёт быстрее любой примитивно рекурсивной и поэтому примитивно рекурсивной функцией не является.

*Функция Аккермана*[5] $A(m,\ n)$ — простой пример вычислимой функции, которая не является примитивно рекурсивной. Она принимает два неотрицательных целых числа в качестве параметров и возвращает натуральное число. Эта функция растёт очень быстро, например, число $A(4,\ 4)$ настолько велико, что количество цифр в порядке этого числа многократно превосходит количество атомов в наблюдаемой части вселенной.

Функция Аккермана определяется рекурсивно для неотрицательных целых чисел $m$ и $n$ следующим образом:

$$A(m,\ n) = \begin{cases} n+1, & \text{если } m = 0; \\ A(m-1,\ 1), & \text{если } m > 0 \text{ и } n = 0; \\ A\big(m-1,\ A(m,\ n-1)\big), & \text{если } m > 0 \text{ и } n > 0. \end{cases}$$

Может показаться не очевидным, что рекурсия всегда заканчивается. Это следует из того, что при рекурсивном вызове или уменьшается значение $m$, или значение $m$ сохраняется, но уменьшается значение $n$. Это означает, что каждый раз пара $(m,\ n)$ уменьшается с точки зрения лексикографического порядка, значит, значение $m$ в итоге достигнет нуля: для одного значения $m$ существует конечное число возможных значений $n$ (так как первый вызов с данным $m$ был произведён с каким-то определённым значением $n$, а в дальнейшем, если значение $m$ сохраняется, значение $n$ может только уменьшаться), а количество возможных значений $m$, в свою очередь, тоже конечно. Однако, при уменьшении $m$ значение, на которое увеличивается $n$, не ограничено и обычно очень велико.

Например, вычислим значение $A(1,\ 2)$:

$$\begin{aligned} A(1,\ 2) &= A\big(0,\ A(1,\ 1)\big) = \\ &= A\Big(0,\ A\big(0,\ A(1,\ 0)\big)\Big) = \\ &= A\Big(0,\ A\big(0,\ A(0,\ 1)\big)\Big) = \\ &= A\big(0,\ A(0,\ 2)\big) = \\ &= A(0,\ 3) = \\ &= 4. \end{aligned}$$

В табл. 1 приведены значения $A(m,\ n)$ для $m = 0, \ldots, 4$.

[5] Вильгельм Фридрих Á к к е р м а н (Wilhelm Friedrich Ackermann; 1896—1962) — немецкий математик.

*Таблица 1*

**Значения функции Аккермана $A(m, n)$**

| $m$ | $A(m,\ 0)$ | $A(m,\ 1)$ | $A(m,\ 2)$ | $A(m,\ 3)$ | $A(m,\ 4)$ | $A(m,\ n)$ |
|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | $n+1$ |
| 1 | 2 | 3 | 4 | 5 | 6 | $2+(n+3)-3$ |
| 2 | 3 | 5 | 7 | 9 | 11 | $2\cdot(n+3)-3$ |
| 3 | 5 | 13 | 29 | 61 | 125 | $2^{n+3}-3$ |
| 4 | 13 | 65 533 | $2^{65536}-3$ | $2^{2^{65536}}-3$ | $2^{2^{2^{65536}}}-3$ | $\underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{n+3}-3$ |

# 2. Нормальные алгоритмы Маркова

*Нормальный алгоритм Маркова* — один из стандартизованных вариантов представления об алгоритме. Понятие нормального алгоритма введено А. А. Марковым[6)] в конце 1940-х годов.

### Описание

Нормальные алгоритмы являются *вербальными,* то есть предназначенными для применения к словам в различных алфавитах. Определение всякого нормального алгоритма состоит из двух частей: определения *алфавита* алгоритма (к словам в котором алгоритм будет применяться) и определения его *схемы. Схемой* нормального алгоритма называется конечный упорядоченный набор т. н. *формул подстановки,* каждая из которых может быть *простой* или *заключительной. Простыми формулами подстановки* называются слова вида $L \to D$, где $L$ и $D$ — два произвольных слова в алфавите алгоритма (называемые, соответственно, левой и правой частями формулы подстановки). Аналогично, *заключительными формулами подстановки* называются слова вида $L \to\cdot D$, где $L$ и $D$ — два произвольных слова в алфавите алгоритма. При этом предполагается, что вспомогательные буквы $\to$ и $\to\cdot$ не принадлежат алфавиту алгоритма (в противном случае на исполняемую ими роль разделителя левой и правой частей следует избрать другие две буквы).

Примером схемы нормального алгоритма в пятибуквенном алфавите $\{\mathtt{|}, \mathtt{*}, \mathtt{a}, \mathtt{b}, \mathtt{c}\}$ может служить схема

$$\begin{cases} \mathtt{|b} & \to & \mathtt{ba|} \\ \mathtt{ab} & \to & \mathtt{ba} \\ \mathtt{b} & \to & \\ \mathtt{*|} & \to & \mathtt{b*} \\ \mathtt{*} & \to & \mathtt{c} \\ \mathtt{|c} & \to & \mathtt{c} \\ \mathtt{ac} & \to & \mathtt{c|} \\ \mathtt{c} & \to\cdot & \end{cases}$$

Процесс применения нормального алгоритма к произвольному слову $V$ в алфавите этого алгоритма представляет собой дискретную последовательность элементарных ша-

[6)] Андрей Андреевич М а́ р к о в (младший) (1903—1979) — советский математик, сын известного русского математика Андрея Андреевича Маркова (старшего) (1856—1922).

гов, состоящих в следующем. Пусть $V'$ — слово, полученное на предыдущем шаге работы алгоритма (или исходное слово $V$, если текущий шаг является первым).

1. Если среди формул подстановки нет такой, левая часть которой входила бы в $V'$, то работа алгоритма считается завершённой, и результатом этой работы считается слово $V'$.

2. Иначе, среди формул подстановки, левая часть которых входит в $V'$, выбирается самая верхняя:

   а) если эта формула подстановки имеет вид $L \to \cdot D$, то из всех возможных представлений слова $V'$ в виде $RLS$ выбирается такое, при котором $R$ — самое короткое, после чего работа алгоритма считается завершённой с результатом $RDS$;

   б) если же эта формула подстановки имеет вид $L \to D$, то из всех возможных представлений слова $V'$ в виде $RLS$ выбирается такое, при котором $R$ — самое короткое, после чего слово $RDS$ считается результатом текущего шага, подлежащим дальнейшей переработке на следующем шаге.

Например, в ходе процесса применения алгоритма с указанной выше схемой к слову `|*||` последовательно возникают слова `|b*|`, `ba|*|`, `a|*|`, `a|b*`, `aba|*`, `baa|*`, `aa|*`, `aa|c`, `aac`, `ac|` и `c||`, после чего алгоритм завершает работу с результатом `||`.

Любой нормальный алгоритм эквивалентен некоторой машине Тьюринга, и наоборот — любая машина Тьюринга эквивалентна некоторому нормальному алгоритму. Вариант тезиса Чёрча—Тьюринга, сформулированный применительно к нормальным алгоритмам, принято называть «принципом нормализации».

### Примеры

Описанный ниже алгоритм преобразует числа, записанные в бинарной системе счисления, в числа, записанные в унарной (в последней вместо единиц будем использовать чёрточки «`|`»). Иными словами, если на входе было число $N$ (записанное в бинарной системе счисления), то на выходе получается строка из $N$ чёрточек.

Например, `101` преобразуется в `|||||`.

Схема:

$$\begin{cases} \texttt{|0} & \to & \texttt{0||} \\ \texttt{1} & \to & \texttt{0|} \\ \texttt{0} & \to & \end{cases}$$

Исходная строка: `101`

Выполнение:

1) `0|01` 2) `00||1` 3) `00||0|` 4) `00|0|||`
5) `000|||||` 6) `00|||||` 7) `0|||||` 8) `|||||`