Министерство науки и высшего образования Российской Федерации
Дальневосточный федеральный университет

Е.Л. Ефремов

АЛГОРИТМЫ ВЫЧИСЛЕНИЯ ЧАСТИЧНЫХ ФУНКЦИЙ

Учебно-методическое пособие

Владивосток
ИЗДАТЕЛЬСТВО ДВФУ
2021

УДК 510.51, 510.57(076)
ББК 22.12р30-я
Е92

Работа выполнена при поддержке Минобрнауки РФ,
дополнительное соглашение от 21.04.2020 № 075-02-2020-1482-1

Рецензенты:
А.А. Степанова, д-р физ.-мат. наук, доцент, профессор кафедры алгебры, геометрии и анализа ШЕН ДВФУ;
М.А. Первухин, канд. физ.-мат. наук, доцент, доцент кафедры алгебры, геометрии и анализа ШЕН ДВФУ.

Ефремов, Евгений Леонидович.

Е92 Алгоритмы вычисления частичных функций : учебно-методическое пособие / Е.Л. Ефремов. – Владивосток : Издательство Дальневосточного федерального университета, 2021. – 48 с.

ISBN 978-5-7444-5020-5.

В пособии рассматриваются основные формализации понятия «алгоритм»: машина Тьюринга, машина Поста, нормальный алгорифм. Для каждого класса алгоритмов приводится описание, подробные примеры алгоритмов, вычисляющих частичные функции, и несколько заданий для проверки усвоения материала.

Пособие может использоваться для подготовки студентов, изучающих математическую логику и теорию алгоритмов и обучающихся по направлениям «Математика», «Прикладная математика и информатика», «Программная инженерия», «Математическое обеспечение и администрирование информационных систем». Интуитивное описание и примеры каждого алгоритма подходят для изучения на факультативах и математических кружках в школе.

УДК 510.51, 510.57(076)
ББК 22.12р30-я

ISBN 978-5-7444-5020-5

Содержание

Введение 4
§ 1. Основные определения и обозначения 6
§ 2. Машины Тьюринга 8
Описание и принцип работы 8
Примеры 9
Задания 12
Определение машины Тьюринга 16
§ 3. Машины Поста 20
Описание и принцип работы 20
Примеры 21
Задания 22
§ 4. Нормальные алгорифмы Маркова 28
Описание и принцип работы 28
Примеры 29
Задания 31
Определение нормального алгорифма 34
Дополнительные задачи 37
Заключение 42
Список литературы 43
Приложение А. Машины с неограниченными регистрами 44
Приложение Б. Машины с ограниченным числом переменных 46

Введение

Слово «алгоритм» происходит от латинской формы написания имени персидского математика IX века *аль-Хорезми*. В своей книге «Китаб аль-джебр ва-ль-мукабала» («Краткая книга о восполнении и противопоставлении») он описал методы сложения, умножения и деления чисел, записанных в десятичной системе счисления (арабскими цифрами). Эти методы требовали механического выполнения строго описанных действий и приводили к правильному результату. На современном языке можно сказать, что аль-Хорезми написал алгоритмы выполнения основных арифметических операций над числами.

Интуитивно понятие алгоритма использовалось очень давно. Лейбниц (1646-1716 гг.) «мечтал» о нахождении универсального алгоритма для решения всех математических проблем. Однако строгого математического определения алгоритма долгое время не существовало. Точное определение алгоритма позволило решить вопрос Лейбница в отрицательную сторону. Алонзо Чёрч в 1936 году показал невозможность алгоритма, который по произвольному утверждению, записанному на формальном языке арифметики, отвечал бы на вопрос, будет это утверждение истинно на натуральных числах или нет. Позже оказалось, что уже исчисление предикатов (логика первого порядка) является алгоритмически неразрешимым. В последующие годы было обнаружено множество алгоритмически неразрешимых проблем в разных разделах математики.

Под *алгоритмом* понимается точное и полное предписание *исполнителю* совершить последовательность действий (*команд*), направленных на решение поставленной задачи. Как правило, интуитивного понимания бывает достаточно для того, чтобы установить, является ли данное предписание алгоритмом или нет. Вопрос о формализации понятия алгоритма существовал долго, было предложено много различных версий. В этом есть и свои плюсы, так как для решения различных задач бывает удобно использовать различные наиболее подходящие определения (можно провести аналогию с языками программирования).

Большой вклад в развитие теории алгоритмов и решение алгоритмических проблем внесли А. Тьюринг, А.А. Марков, А.И. Мальцев, П.С. Новиков, А. Чёрч, Э. Пост, С. Клини.

В пособии рассмотрены основные классы алгоритмов: машины Тьюринга, машины Поста и нормальные алгорифмы Маркова. Для каждого класса алгоритмов приведено интуитивное описание, принцип работы и примеры. Для машины Тьюринга и нормального алгорифма Маркова даны строгие определения. Для всех рассматриваемых алгоритмов приведены задания для самостоятельной работы. Первое задание простое, не требует много времени и усилий. Оно направлено на знакомство с принципом работы алгоритма. Во втором задании необходимо составить алгоритм для вычисления функции, оно требует детального знакомства с логикой алгоритма.

В конце приведён список задач творческого характера. Их решение позволит оценить универсальность изучаемых алгоритмов. Задачи этого раздела рекомендованы заинтересовавшимся в теории алгоритмов.

В приложениях приводятся описания ещё двух классов алгоритмов – машины с неограниченными регистрами и машины с ограниченным числом переменных. Возможно, они покажутся более наглядными программистам, так как используют стандартный язык написания алгоритмов. Заинтересовавшимся в теории алгоритмов читателям будет полезно познакомиться с такими классами алгоритмов и решить разобранные задачи с их помощью.

§ 1. Основные определения и обозначения

Предполагается, что читатель знаком с основами теории множеств, математической логики и теории алгоритмов, поэтому дадим лишь некоторые определения и обозначения. В случае возникновения трудностей с прочтением каких-либо записей можно обратиться, например, к [1, 2].

Под *символом* будем понимать знак, который рассматривается целым, т.е. части которого нас не интересуют. Про любые два символа можно однозначно утверждать, одинаковы они или различны. *Алфавитом* будем называть непустое множество символов.

Конечная последовательность длины $n \in \omega$[1] символов алфавита A называется *словом длины n алфавита A*. Слово, не содержащее ни одного символа ($n = 0$), называется *пустым*. На слова алфавита A можно смотреть как на формулы языка A, поэтому для записи равных (как последовательности) слов уместно обозначение $\leftrightharpoons$.

Если α, β – слова алфавита A, то через $\alpha\beta$ обозначим слово, получаемое приписыванием слова β в конец слова α. Операция приписывания называется *конкатенацией*. Нетрудно понять, что конкатенация ассоциативна, а пустое слово служит нейтральным элементом относительно неё, т.е. множество всех слов алфавита A образует полугруппу относительно конкатенации.

Слово β алфавита A называется *подсловом* слова α алфавита A, если существуют такие слова γ и δ алфавита A, что $\alpha \leftrightharpoons \gamma\beta\delta$. Может оказаться так, что $\alpha \leftrightharpoons \gamma_1\beta\delta_1$ и $\alpha \leftrightharpoons \gamma_2\beta\delta_2$, где $\gamma_1 \neq \gamma_2$. В этом случае будем говорить о различных *вхождениях* подслова β в слово α. Вхождение подслова β в слово α называется *первым*, если $\alpha \leftrightharpoons \gamma_0\beta\delta$ для некоторых слов γ_0 и δ алфавита A и γ_0 – слово наименьшей длины из всех слов γ таких, что $\alpha \leftrightharpoons \gamma\beta\delta$.

Унарной системой счисления называется система счисления, состоящая из одного символа, в которой любое натуральное число $n \in \omega$ задаётся словом из $n + 1$ символа. Для решения задач на вычисления в

[1] Через ω будем обозначать множество натуральных чисел с нулём.

унарной системе счисления будем использовать символ 1. Так, число 0 будет задаваться словом 1, число 1 – 11, число 2 – 111 и т.д.

Каждый из изучаемых алгоритмов работает над словами некоторого алфавита A. Слово, с которым начинает работать алгоритм, будем называть *начальным*. Тот факт, что в процессе выполнения алгоритма слово α за некоторое количество шагов будет преобразовано в слово β, будем обозначать как $\alpha \vdash \beta$.

Для любого из изучаемых алгоритмов $\mathfrak{A}$ может иметь место одна из следующих ситуаций:

1. Алгоритм $\mathfrak{A}$ завершился за конечное количество шагов. Слово β, в которое было преобразовано начальное слово α, будем считать *результатом* работы алгоритма. В этом случае будем писать $\mathfrak{A}(\alpha) = \beta$.

2. Алгоритм не завершается (бесконечное выполнение команд). Такую ситуацию также можно рассматривать как результат работы алгоритма и интерпретировать по-разному в зависимости от решаемой задачи. Например, если нужно составить алгоритм, который будет вычислять значение функции $f(\bar{x})$ на наборе значений $\bar{a}$, то в случае, когда $\bar{a} \notin D(f)$, можно организовать зацикливание.

Частичную функцию $f: X \to \omega$, где $X \subseteq \omega^n$, $n \in \omega$, будем называть *вычислимой*, если существует алгоритм $\mathfrak{A}$, работающий над словами $\bar{a} \in \omega^n$, такой, что

1) если $\bar{a} \in X$, то $\mathfrak{A}(\bar{a}) = f(\bar{a})$,

2) если $\bar{a} \notin X$, то $\mathfrak{A}$ не завершается.

В этом случае будем говорить, что *алгоритм* $\mathfrak{A}$ *вычисляет функцию* f.

§ 2. Машины Тьюринга

Машина Тьюринга – абстрактный универсальный исполнитель, предложенный Аланом Тьюрингом в 1936 году для уточнения понятия алгоритма.

Для знакомства с принципом работы машины Тьюринга отвлечёмся от её строгого определения. Для решения практических задач интуитивного понимания достаточно. Со строгим определением машины Тьюринга можно познакомиться в конце параграфа.

Описание и принцип работы

Машина Тьюринга состоит из бесконечной в обе стороны *ленты*, разделённой на ячейки, и *каретки*, которая управляется программой. Каждая ячейка ленты может содержать один символ или быть пустой. В каждый момент времени каретка обозревает одну ячейку ленты.

Пусть задан *внешний алфавит* $A = \{\lambda, a_1, a_2, \dots\}$ и внутренний алфавит $Q = \{q_0, q_1, q_2, \dots\}$, элементы которого будем называть *внутренними состояниями* каретки. Символ λ называется *пустым символом* и является обязательным элементом внешнего алфавита. Состояния q_0 и q_1 называются *терминальным* (*завершающим*) и *начальным* состояниями соответственно и являются обязательными внутренними состояниями каретки.

Любая *команда* для каретки является словом алфавита $A \cup Q \cup \{R, L, E\}$ и имеет один из следующих видов:

$$aRq, \qquad aLq, \qquad aEq,$$

где $a \in A$, $q \in Q$. Команда aRq означает, что каретка должна вписать в обозреваемую ячейку символ a, переместиться на одну ячейку вправо и перейти в состояние q. Аналогичным образом интерпретируются команды aLq и aEq, где каретка должна переместиться на одну ячейку влево и остаться на месте соответственно.

Программу для машины Тьюринга будем записывать в виде таблицы, в ячейках которой записаны команды для каретки (Таблица 1).

Таблица 1

Программа для машины Тьюринга

	λ	a_1	a_2	…
q_1	команда	команда	команда	…
q_2	команда	команда	команда	…
q_3	команда	команда	команда	…
…	…	…	…	…

Символ, который находится в определённый момент времени в обозреваемой кареткой ячейке, называется *считываемым*. Внутреннее состояние каретки, в котором она находится в этот момент времени, называется *текущим*. Команда, которую должна выполнить каретка, находится в таблице на пересечении столбца со считываемым символом и строки с текущим состоянием.

По умолчанию все ячейки ленты пусты, т.е. содержат символ λ. Подготовка к «запуску» машины Тьюринга состоит из следующих шагов:

1. В ячейки ленты записываются символы алфавита A.
2. Задаётся положение каретки.

В начальный момент времени каретка находится в состоянии q_1, после чего начинается последовательное выполнение команд.

Примеры

По умолчанию будем считать, что каретка обозревает крайнюю слева непустую ячейку, если в задаче не сказано противное. Так как мы не работаем с бесконечными словами, такая ячейка всегда существует.

Пример 1. Напишем программу для машины Тьюринга, которая вычисляет функцию $f(x) = x + 1$ в унарной системе счисления.

Число x отличается от числа $x + 1$ количеством единиц. Следовательно, достаточно дописать с какого-нибудь конца данного слова одну единицу. Так как в начальный момент времени каретка обозревает крайнюю левую единицу записанного числа x, то удобнее сдвинуться влево на одну ячейку и вписать единицу туда:

$$\underbrace{11\ldots1}_{x+1} \vdash 1\underbrace{11\ldots1}_{x+1}.$$

Таблица 2

Программа для примера 1

	λ	1
q_1	$1Eq_0$	$1Lq_1$

Пример 2. Напишем программу для машины Тьюринга, которая вычисляет функцию $f(x,y) = x + y$ в унарной системе счисления. Числа x и y записаны в указанном порядке и отделены друг от друга символом $+$.

Опишем логику работы машины. Изначально на ленте записано два слова из $x+1$ и $y+1$ единиц, разделённые $+$. Если заменить разделительный $+$ на единицу, то получится слово из $x+y+3$ единиц. Число $x+y$ записывается словом из $x+y+1$ единиц, т.е. две единицы необходимо стереть. Так как число x может быть равно 0, то в начале стирать две единицы подряд нельзя – мы можем попасть на $+$, что повлечёт выполнение других команд. Можно обойти эту ситуацию, сделав ветвление условий. Но гораздо проще стереть одну единицу в начале и одну единицу в конце (или две единицы в конце):

$$\underbrace{11\ldots1}_{x+1}+\underbrace{1\ldots11}_{y+1} \vdash \underbrace{1\ldots1}_{x}+\underbrace{1\ldots11}_{y+1} \vdash \underbrace{1\ldots1}_{x}1\underbrace{1\ldots11}_{y+1} \vdash \underbrace{1\ldots1}_{x}1\underbrace{1\ldots1}_{y}.$$

Таблица 3

Программа для примера 2

	λ	1	$+$
q_1		λRq_2	
q_2	λLq_3	$1Rq_2$	$1Rq_2$
q_3		λEq_0	

В таблице некоторые ячейки пусты. Это можно объяснить тем, что ситуации, в которых потребуется команда из этих ячеек, недостижимы.

Пример 3. Напишем программу для машины Тьюринга, которая вычисляет функцию $f(x) = 3x$ в унарной системе счисления.

Решить задачу можно следующим образом. В начальный момент времени на ленте находится слово из $x + 1$ единиц. Заменим первую единицу на $*$, оставив тем самым x единиц. Уйдём в правый конец набора единиц. Организуем цикл: на каждую записанную справа от звёздочки единицу допишем три новые единицы слева от звёздочки. В цикле будет происходить следующее:

1) удалим крайнюю справа единицу,
2) пройдём влево все оставшиеся единицы,
3) пройдём звёздочку,
4) пройдём влево все новые единицы,
5) допишем три единицы в начало,
6) вернёмся в правый конец.

Выход из цикла произойдёт тогда, когда кончатся единицы справа от звёздочки. Заменим звёздочку обратно на единицу. На ленте окажется слово из $3x + 1$ единиц, что и требовалось получить:

$$\underbrace{11\ldots11}_{x+1} \vdash *\underbrace{1\ldots11}_{x} \vdash 111*\underbrace{1\ldots1}_{x-1} \vdash$$
$$\vdash 111111*\underbrace{1\ldots1}_{x-2} \vdash \ldots \vdash \underbrace{1\ldots1}_{3x}* \vdash \underbrace{1\ldots1}_{3x}1.$$

Таблица 4

Программа для примера 3

	λ	1	$*$
q_1		$*Rq_2$	
q_2	λLq_3	$1Rq_2$	
q_3		λLq_4	$1Eq_0$
q_4	$1Lq_5$	$1Lq_4$	$*Lq_4$
q_5	$1Lq_6$		
q_6	$1Rq_7$		
q_7	λLq_3	$1Rq_7$	$*Rq_7$

Задания

Задание 1. Машина Тьюринга с внешним алфавитом $A = \{\lambda, 1\}$ и множеством внутренних состояний $Q = \{q_0, q_1, \dots, q_{13}\}$ определяется следующей программой:

Таблица 5

Программа для задания 1

	λ	1
q_1	$\lambda L q_2$	$1 E q_0$
q_2	$\lambda E q_5$	$\lambda E q_3$
q_3	$\lambda L q_4$	$1 E q_0$
q_4	$1 E q_5$	$1 L q_4$
q_5	$\lambda E q_0$	$1 L q_6$
q_6	$\lambda E q_0$	$\lambda E q_7$
q_7	$\lambda R q_8$	$1 E q_0$
q_8	$1 E q_9$	$1 R q_8$
q_9	$\lambda E q_0$	$1 L q_{10}$
q_{10}	$\lambda E q_0$	$\lambda E q_{11}$
q_{11}	$\lambda L q_{12}$	$1 E q_0$
q_{12}	$1 E q_{13}$	$1 L q_{12}$
q_{13}	$\lambda E q_0$	$1 E q_0$

Изображая на каждом такте работы машины получающуюся конфигурацию, определите, в какое слово перерабатывает машина данное слово. В начальный момент времени каретка обозревает крайнюю справа ячейку (с λ).

1. $\lambda 11 \lambda 111 \lambda 1 \lambda$
2. $\lambda 1 \lambda 11 \lambda 111 \lambda$
3. $\lambda 1 \lambda 1111 \lambda 1 \lambda$
4. $\lambda 1 \lambda 1 \lambda 1111 \lambda$
5. $\lambda 111 \lambda 1 \lambda 11111 \lambda$
6. $\lambda 111 \lambda 111 \lambda 111 \lambda$
7. $\lambda 111 \lambda 1111 \lambda$
8. $\lambda 11 \lambda 1 \lambda 11 \lambda 1 \lambda$
9. $\lambda 1 \lambda 1111 \lambda 11 \lambda$
10. $\lambda 11111 \lambda 1 \lambda$
11. $\lambda 11 \lambda 11111 \lambda$
12. $\lambda 1 \lambda 1 \lambda 1 \lambda 1 \lambda 1 \lambda$
13. $\lambda 111 \lambda 11 \lambda 1 \lambda$
14. $\lambda 11111 \lambda 1 \lambda$

15. $\lambda 111\lambda 11\lambda$
16. $\lambda 111\lambda 1\lambda 11\lambda$
17. $\lambda 11\lambda 1111\lambda$
18. $\lambda 11\lambda 11\lambda 1\lambda$
19. $\lambda 111\lambda 11\lambda 11\lambda$
20. $\lambda 11\lambda 11111\lambda 11\lambda$
21. $\lambda 111\lambda 1111\lambda 11\lambda$
22. $\lambda 1111\lambda 1111\lambda$
23. $\lambda 11\lambda 1111\lambda 1\lambda$
24. $\lambda 111\lambda 111\lambda 1111\lambda$
25. $\lambda 111\lambda 1111\lambda 111\lambda$
26. $\lambda 111\lambda 1\lambda 11\lambda$
27. $\lambda 11\lambda 1111\lambda 1\lambda$
28. $\lambda 111\lambda 111\lambda 1111\lambda$
29. $\lambda 11\lambda 1111\lambda$
30. $\lambda 111\lambda 11\lambda$
31. $\lambda 11\lambda 111\lambda 1\lambda$
32. $\lambda 1\lambda 11\lambda 111\lambda$
33. $\lambda 111\lambda 11\lambda 1\lambda$
34. $\lambda 11111\lambda 1\lambda$
35. $\lambda 11\lambda 11111\lambda$
36. $\lambda 111\lambda 1\lambda 11111\lambda$
37. $\lambda 11\lambda 1\lambda 11\lambda 1\lambda$
38. $\lambda 1\lambda 1111\lambda 11\lambda$
39. $\lambda 11111\lambda 1\lambda$
40. $\lambda 111\lambda 1111\lambda$
41. $\lambda 111\lambda 111\lambda 111\lambda$
42. $\lambda 1\lambda 1\lambda 1\lambda 1\lambda 1\lambda$
43. $\lambda 11\lambda 11\lambda 1\lambda$
44. $\lambda 1\lambda 1111\lambda 1\lambda$
45. $\lambda 1\lambda 1\lambda 11111\lambda$
46. $\lambda 11\lambda 1111\lambda$
47. $\lambda 11\lambda 11\lambda 1\lambda$
48. $\lambda 111\lambda 11\lambda 11\lambda$
49. $\lambda 11\lambda 11111\lambda 11\lambda$
50. $\lambda 111\lambda 1111\lambda 11\lambda$

Задание 2. Напишите программу для машины Тьюринга, вычисляющей функцию в унарной системе счисления. Значения $x, y \in \omega$ записаны на ленте в указанном порядке и отделены друг от друга пустой ячейкой. В начальный момент времени каретка обозревает крайнюю левую единицу числа x.

Комментарии к составлению программы:

– число x написано слева, число y – справа, между числами одна пустая ячейка,

– выражение, задающее функцию, не содержит знаков целочисленного деления или округления,

– функция имеет значение тогда и только тогда, когда после выполнения всех действий (неважно, в каком порядке) получится натуральное число,

– если функция не имеет значения на вводимом наборе чисел x, y, то машина должна зациклиться.

Для создания алгоритма используйте эмулятор[2] машины Тьюринга. На рисунке 1 показан скриншот эмулятора, разработанного К.Ю. Поляковым и свободно скачиваемого с сайта http://kpolyakov.spb.ru/. Использование эмулятора удобно для проверки работы алгоритма и поиска ошибок. В поле «Условие задачи» необходимо указать следующую информацию: фамилия, имя, номер группы, номер варианта, текст задания. Интерфейс интуитивно понятный. За более подробными инструкциями можно обратиться к справке программы.

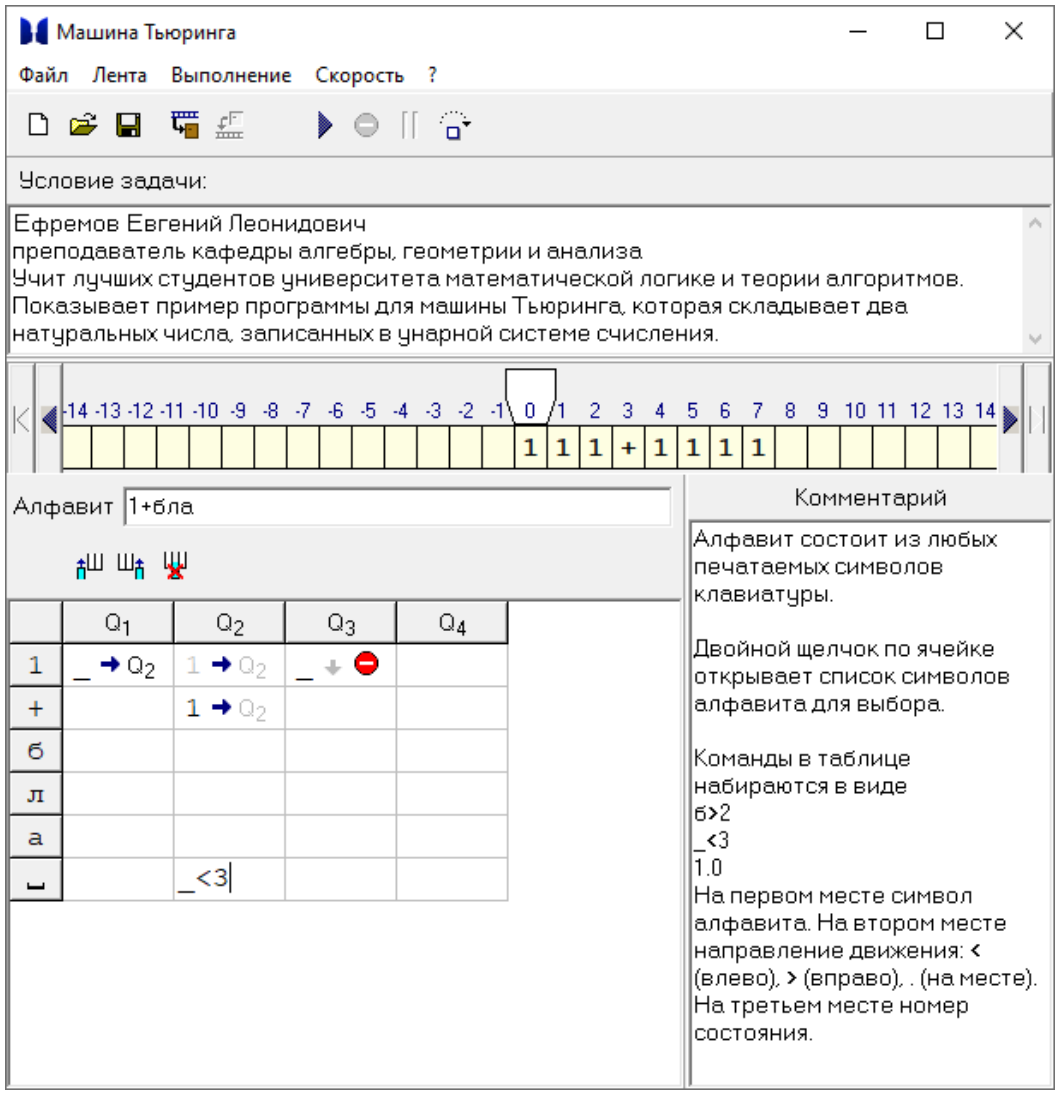


Рис. 1. Пример работы на эмуляторе машины Тьюринга

[2] *Эмулятор* – это приложение на ПК, имитирующее машину и реализующее работу алгоритма.

1. $\frac{4-2y+3x}{2}$.
2. $\frac{2-3x+4y}{4}$.
3. $\frac{3-4y+3x}{2}$.
4. $\frac{2-4x+4y}{3}$.
5. $\frac{2-4y+2x}{3}$.
6. $\frac{4-2x+4y}{3}$.
7. $\frac{3-5y+2x}{2}$.
8. $\frac{4-3y+3x}{2}$.
9. $\frac{2-2x+3y}{4}$.
10. $\frac{4-2y+4x}{3}$.
11. $\frac{3-2x+3y}{4}$.
12. $\frac{3-5x+2y}{4}$.
13. $\frac{2-3y+3x}{4}$.
14. $\frac{2-4x+3y}{3}$.
15. $\frac{3-4x+2y}{3}$.
16. $\frac{2-2y+4x}{3}$.
17. $\frac{4-4x+2y}{3}$.
18. $\frac{3-4x+3y}{4}$.
19. $\frac{2-4y+3x}{4}$.
20. $\frac{4-3x+4y}{2}$.
21. $\frac{2x-3y-4}{4}$.
22. $\frac{2-3x+2y}{4}$.
23. $\frac{4-2y+2x}{3}$.
24. $\frac{3-3y+2x}{4}$.
25. $\frac{4-4y+3x}{2}$.
26. $\frac{2-3y+2x}{4}$.
27. $\frac{2-3y+4x}{3}$.
28. $\frac{3-2y+4x}{3}$.
29. $\frac{4-4y+2x}{3}$.
30. $\frac{3-3y+4x}{4}$.
31. $\frac{3x-4y-2}{3}$.
32. $\frac{2-2x+4y}{3}$.
33. $\frac{4-4x+3y}{2}$.
34. $\frac{2-4y+4x}{3}$.
35. $\frac{3-2x+5y}{4}$.
36. $\frac{3-4x+5y}{2}$.
37. $\frac{3-3x+4y}{2}$.
38. $\frac{4-2x+3y}{2}$.
39. $\frac{4-3y+4x}{2}$.
40. $\frac{3-2x+4y}{2}$.
41. $\frac{3-3x+2y}{4}$.
42. $\frac{2-2y+3x}{4}$.
43. $\frac{4-3x+2y}{2}$.
44. $\frac{3-4y+5x}{2}$.
45. $\frac{4-2x+2y}{3}$.
46. $\frac{2-3x+3y}{4}$.
47. $\frac{4-3y+2x}{2}$.
48. $\frac{4-3x+3y}{2}$.
49. $\frac{3-2y+3x}{4}$.
50. $\frac{2-4x+2y}{3}$.

Определение машины Тьюринга

Пусть A и Q – два алфавита,

$P = \{\langle x_i, y_i, u_i, v_i\rangle \mid i \leq m\} \subseteq Q \times A \times Q \times (A \cup \{L, R\})$.

Шестёрка $\mathfrak{M} = \langle A, a_0, Q, q_0, q_1, P\rangle$ называется *машиной Тьюринга*, если выполняются следующие условия:

1) $|A| < \omega$,
2) $|Q| < \omega$,
3) $A \cap Q = \emptyset$,
4) $L, R \notin A \cup Q$,
5) $a_0 \in A$,
6) $q_0, q_1 \in Q$,
7) если $\langle x, y, u, v\rangle, \langle x, y, u', v'\rangle \in P$, то $u = u'$ и $v = v'$,
8) если $\langle x, y, u, v\rangle \in P$, то $x \neq q_0$.

Алфавит A называется *внешним*, алфавит Q – *внутренним алфавитом машины Тьюринга* $\mathfrak{M}$. Множество четвёрок P называется *программой машины* $\mathfrak{M}$, элементы программы называются *командами*.

Слово α алфавита $A \cup Q$ называется *машинным словом в* $\langle A, Q\rangle$, если α является словом алфавита $A \cup \{q\}$ для некоторого $q \in Q$, причём q входит в α ровно один раз. Через α^{a_0} обозначим слово $a_0 \alpha a_0$. Если $\alpha \leftrightharpoons a\beta b$ для некоторых $a, b \in A \cup Q$, то через α_{a_0} обозначим слово

- β, если $a = a_0 = b$;
- βb, если $a = a_0 \neq b$;
- $a\beta$, если $a \neq a_0 = b$;
- $a\beta b$, если $a \neq a_0 \neq b$.

Пусть α и β – машинные слова в $\langle A, Q\rangle$ и $q \in Q$ входит в α. Будем говорить, что *машина Тьюринга* $\mathfrak{M}$ *переводит слово* α *в слово* β (обозначение: $\alpha \overset{\mathfrak{M}}{\rightarrow} \beta$), если выполняются следующие условия:

1) если $\alpha^{a_0} \leftrightharpoons \alpha_1 q a \alpha_2$ и $\langle q, a, r, b\rangle \in P$, где $b \in A$, то $\beta \leftrightharpoons (\alpha_1 r b \alpha_2)_{a_0}$,
2) если $\alpha^{a_0} \leftrightharpoons \alpha_1 b q a \alpha_2$ и $\langle q, a, r, L\rangle \in P$, то $\beta \leftrightharpoons (\alpha_1 r b a \alpha_2)_{a_0}$,
3) если $\alpha^{a_0} \leftrightharpoons \alpha_1 q a \alpha_2$ и $\langle q, a, r, R\rangle \in P$, то $\beta \leftrightharpoons (\alpha_1 a r \alpha_2)_{a_0}$.

Если машина Тьюринга $\mathfrak{M}$ не переводит машинное слово α ни в какое слово, то α называется *тупиковым машинным словом для* $\mathfrak{M}$.

Пусть α и β – слова алфавита $A \setminus \{a_0\}$. Будем говорить, что *машина Тьюринга* $\mathfrak{M}$ *преобразует слово* α *в слово* β, если существует последовательность $\gamma_0, \gamma_1, \dots, \gamma_n$ машинных слов в $\langle A, Q\rangle$ такая, что

1) $\gamma_0 \leftrightarrows q_1\alpha$,

2) $\gamma_i \overset{\mathfrak{M}}{\to} \gamma_{i+1}$ для всех $i < n$,

3) γ_n – тупиковое машинное слово для $\mathfrak{M}$,

4) β получается из γ_n заменой всех вхождений символов q_0 и a_0 на пустое слово.

Машина Тьюринга $\mathfrak{M}$ может преобразовывать слово α только в одно слово. Если машина $\mathfrak{M}$ не преобразовывает α ни в какое слово, то будем говорить, что $\mathfrak{M}$ *не применима к* α или что $\mathfrak{M}(\alpha)$ *не определено*. Это возможно в следующих случаях:

– существует бесконечная последовательность $\gamma_0, \gamma_1, \dots, \gamma_n, \dots$ $(n \in \omega)$ машинных слов, удовлетворяющих условиям 1) и 2) определения;

– существует конечная последовательность $\gamma_0, \gamma_1, \dots, \gamma_n$ машинных слов, удовлетворяющих условиям 1), 2) и 3), где γ_n не содержит q_0.

Составим машины Тьюринга для примеров 1 и 2. Через 1^x обозначим слово, состоящее из $x + 1$ единиц $(x \in \omega)$, т.е. натуральное число x, записанное в унарной системе счисления.

Пример 1. Составим машину Тьюринга $\mathfrak{M}_{+1}$, которая преобразует слово 1^x в слово 1^{x+1}.

$$A = \{a_0, 1\}, Q = \{q_0, q_1\},$$
$$P = \{\langle q_1, 1, q_1, L\rangle, \langle q_1, a_0, q_0, 1\rangle\}.$$

Машина Тьюринга $\mathfrak{M}_{+1} = \langle A, a_0, Q, q_0, q_1, P\rangle$ составлена. Покажем, например, что $\mathfrak{M}_{+1}(1^2) = 1^3$. Запишем «процесс вычисления», оставив комментарии читателю:

$$\boldsymbol{\alpha \leftrightarrows 111}$$
$$\gamma_0 \leftrightarrows \boldsymbol{q_1 111}$$
$$\gamma_0{}^{a_0} \leftrightarrows a_0 q_1 111 a_0$$

$$\langle q_1, 1, q_1, L\rangle \in P$$
$$\gamma_1 \leftrightarrows (q_1 a_0 111 a_0)_{a_0} \leftrightarrows \boldsymbol{q_1 a_0 111}$$
$$\gamma_1{}^{a_0} \leftrightarrows a_0 q_1 a_0 111 a_0$$
$$\langle q_1, a_0, q_0, 1\rangle \in P$$
$$\gamma_2 \leftrightarrows (a_0 q_0 1111 a_0)_{a_0} \leftrightarrows \boldsymbol{q_0 1111}$$
$$\gamma_2{}^{a_0} \leftrightarrows a_0 q_0 1111 a_0$$
$$\forall q \in Q\ \forall a \in A\ (\langle q_0, 1, q, a\rangle \notin P)$$
γ_2 – тупиковое машинное слово для $\mathfrak{M}_{+1}$
$$\boldsymbol{\beta \leftrightarrows 1111}$$

Пример 2. Составим машину Тьюринга $\mathfrak{M}_+$, которая преобразует слово $1^x + 1^y$ в слово 1^{x+y}.

$$A = \{a_0, 1, +\}, Q = \{q_0, q_1, q_2, q_3\},$$
$$P = \{\langle q_1, 1, q_1, a_0\rangle, \langle q_1, a_0, q_2, R\rangle, \langle q_2, 1, q_2, R\rangle,$$
$$\langle q_2, +, q_2, 1\rangle, \langle q_2, a_0, q_3, L\rangle, \langle q_3, 1, q_0, a_0\rangle\}.$$

Машина Тьюринга $\mathfrak{M}_+ = \langle A, a_0, Q, q_0, q_1, P\rangle$ составлена. Покажем, например, что $\mathfrak{M}_+(1^1 + 1^2) = 1^3$. Запишем «процесс вычисления», оставив комментарии читателю:

$$\boldsymbol{\alpha \leftrightarrows 11 + 111}$$
$$\gamma_0 \leftrightarrows \boldsymbol{q_1 11 + 111}$$
$$\gamma_0{}^{a_0} \leftrightarrows a_0 q_1 11 + 111 a_0$$
$$\langle q_1, 1, q_1, a_0\rangle \in P$$
$$\gamma_1 \leftrightarrows (a_0 q_1 a_0 1 + 111 a_0)_{a_0} \leftrightarrows \boldsymbol{q_1 a_0 1 + 111}$$
$$\gamma_1{}^{a_0} \leftrightarrows a_0 q_1 a_0 1 + 111 a_0$$
$$\langle q_1, a_0, q_2, R\rangle \in P$$
$$\gamma_2 \leftrightarrows (a_0 a_0 q_2 1 + 111 a_0)_{a_0} \leftrightarrows \boldsymbol{a_0 q_2 1 + 111}$$
$$\gamma_2{}^{a_0} \leftrightarrows a_0 a_0 q_2 1 + 111 a_0$$
$$\langle q_2, 1, q_2, R\rangle \in P$$
$$\gamma_3 \leftrightarrows (a_0 a_0 1 q_2 + 111 a_0)_{a_0} \leftrightarrows \boldsymbol{a_0 1 q_2 + 111}$$
$$\gamma_3{}^{a_0} \leftrightarrows a_0 a_0 1 q_2 + 111 a_0$$
$$\langle q_2, +, q_2, 1\rangle \in P$$
$$\gamma_4 \leftrightarrows (a_0 a_0 1 q_2 1111 a_0)_{a_0} \leftrightarrows \boldsymbol{a_0 1 q_2 1111}$$
$$\gamma_4{}^{a_0} \leftrightarrows a_0 a_0 1 q_2 1111 a_0$$
$$\langle q_2, 1, q_2, R\rangle \in P$$

$$\gamma_5 \leftrightharpoons (a_0 a_0 11 q_2 111 a_0)_{a_0} \leftrightharpoons \boldsymbol{a_0 11 q_2 111}$$
$${\gamma_5}^{a_0} \leftrightharpoons a_0 a_0 11 q_2 111 a_0$$
$$\langle q_2, 1, q_2, R\rangle \in P$$
$$\gamma_6 \leftrightharpoons (a_0 a_0 111 q_2 11 a_0)_{a_0} \leftrightharpoons \boldsymbol{a_0 111 q_2 11}$$
$${\gamma_6}^{a_0} \leftrightharpoons a_0 a_0 111 q_2 11 a_0$$
$$\langle q_2, 1, q_2, R\rangle \in P$$
$$\gamma_7 \leftrightharpoons (a_0 a_0 1111 q_2 1 a_0)_{a_0} \leftrightharpoons \boldsymbol{a_0 1111 q_2 1}$$
$${\gamma_7}^{a_0} \leftrightharpoons a_0 a_0 1111 q_2 1 a_0$$
$$\langle q_2, 1, q_2, R\rangle \in P$$
$$\gamma_8 \leftrightharpoons (a_0 a_0 11111 q_2 a_0)_{a_0} \leftrightharpoons \boldsymbol{a_0 11111 q_2}$$
$${\gamma_8}^{a_0} \leftrightharpoons a_0 a_0 11111 q_2 a_0$$
$$\langle q_2, a_0, q_3, L\rangle \in P$$
$$\gamma_9 \leftrightharpoons (a_0 a_0 1111 q_3 1 a_0)_{a_0} \leftrightharpoons \boldsymbol{a_0 1111 q_3 1}$$
$${\gamma_9}^{a_0} \leftrightharpoons a_0 a_0 1111 q_3 1 a_0$$
$$\langle q_3, 1, q_0, a_0\rangle \in P$$
$$\gamma_{10} \leftrightharpoons (a_0 a_0 1111 q_0 a_0 a_0)_{a_0} \leftrightharpoons \boldsymbol{a_0 1111 q_0 a_0}$$
$${\gamma_{10}}^{a_0} \leftrightharpoons a_0 a_0 1111 q_0 a_0 a_0$$
$$\forall q \in Q\ \forall a \in A\ (\langle q_0, a_0, q, a\rangle \notin P)$$
γ_{10} – тупиковое машинное слово для $\mathfrak{M}_+$
$$\boldsymbol{\beta \leftrightharpoons 1111}$$

Упражнение 1. Составьте машину Тьюринга $\mathfrak{M}_{3\times}$, которая преобразует слово 1^x в слово 1^{3x}.

Упражнение 2. Сравните описание машины Тьюринга, приведённое в начале параграфа, и её строгое математическое определение. Сравните составленные алгоритмы для соответствующих примеров.

Тезис Чёрча-Тьюринга. Любая вычислимая частичная функция f вычислима по Тьюрингу, т.е. существует машина Тьюринга, вычисляющая f.

В силу определения вычислимой частичной функции верно и обратное утверждение. Таким образом, *класс всех вычислимых частичных функций совпадает с классом всех частичных функций, вычислимых по Тьюрингу*.

§ 3. Машины Поста

Машина Поста – абстрактная вычислительная машина, созданная в 1936 году Эмилем Постом для уточнения понятия алгоритма.

Описание и принцип работы

Машина Поста состоит из бесконечной в обе стороны *ленты*, разделённой на ячейки, и *каретки*, которая управляется программой. Каждая ячейка ленты находится в одном из двух состояний: содержит метку или пустая. В каждый момент времени каретка обозревает одну ячейку ленты.

Программа для машины Поста представляет собой конечную пронумерованную последовательность команд, каждая из которых имеет один из следующих видов:

$\rightarrow n$ – сдвинуться вправо на одну ячейку и перейти к выполнению команды с номером n,

$\leftarrow n$ – сдвинуться влево на одну ячейку и перейти к выполнению команды с номером n,

$\vee n$ – поставить метку в обозреваемую ячейку и перейти к выполнению команды с номером n,

$\times n$ – очистить обозреваемую ячейку и перейти к выполнению команды с номером n,

$? \ n, m$ – команда проверки наличия метки: если метки в обозреваемой ячейке нет, то перейти к выполнению команды с номером n; в противном случае перейти к выполнению команды с номером m,

! – команда завершения выполнения алгоритма.

По умолчанию все ячейки ленты пусты. Подготовка к запуску машины Поста состоит из следующих шагов:

1. В ячейки ленты помещаются метки.
2. Задаётся положение каретки.

Для машины Поста есть две недопустимые ситуации:

– требуется выполнить команду $\vee n$, если в обозреваемой ячейке уже есть метка,

– требуется выполнить команду $\times n$, если в обозреваемой ячейке нет метки.

Составитель программы должен заранее позаботиться об исключении таких ситуаций.

Примеры

Так как машина Поста позволяет работать с числами только в унарной системе счисления (хотя в различных модификациях возможно двоичное кодирование), то при решении всех задач на вычисления по умолчанию будем полагать, что речь идёт про числа из ω, записываемые в унарной системе счисления метками. Также будем считать, что каретка в начальный момент времени обозревает крайнюю слева ячейку с меткой.

Решим те же задачи, что мы решали в предыдущем параграфе. Так как мы работали в унарной системе счисления, то можно сохранить логику решения.

Пример 1. Напишем программу для машины Поста, которая вычисляет функцию $f(x) = x + 1$.

1. $\leftarrow$ 2
2. $\vee$ 3
3. !

Пример 2. Напишем программу для машины Поста, которая вычисляет функцию $f(x, y) = x + y$. Числа x и y записаны на ленте в указанном порядке и отделены друг от друга пустой ячейкой.

1. $\times$ 2
2. $\rightarrow$ 3
3. ? 4, 2
4. $\vee$ 5
5. $\rightarrow$ 6
6. ? 7, 5
7. $\leftarrow$ 8
8. $\times$ 9
9. !

Первой командой убирается крайняя слева метка. Командами 2–3 организован проход числа x. Четвёртой командой ставится метка в разделительную ячейку. Командами 5–6 осуществляется проход числа y. Командами 7–8 убирается последняя метка.

Пример 3. Напишем программу для машины Поста, которая вычисляет функцию $f(x) = 3x$.

1. × 2
2. → 3
3. ? 4, 2
4. ← 5
5. ? 21, 6
6. × 7
7. ← 8
8. ? 9, 7
9. ← 10
10. ? 11, 9
11. ∨ 12
12. ← 13
13. ∨ 14
14. ← 15
15. ∨ 16
16. → 17
17. ? 18,16
18. → 19
19. ? 20, 18
20. ← 5
21. ∨ 22
22. !

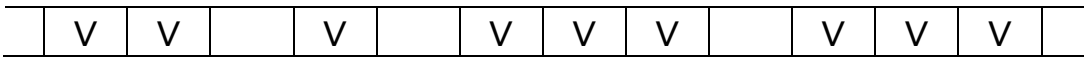
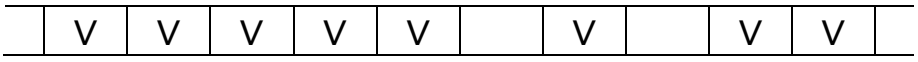
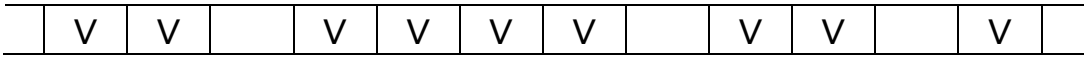
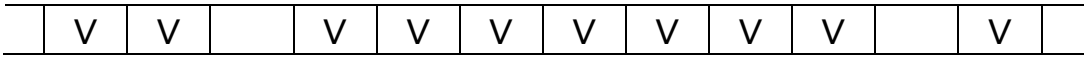
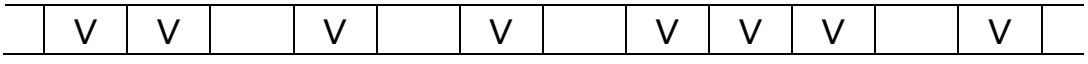
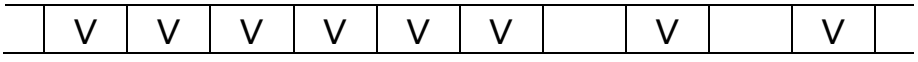
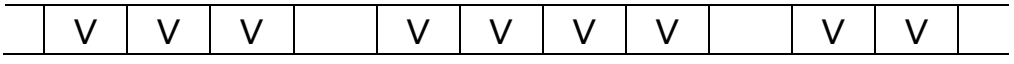
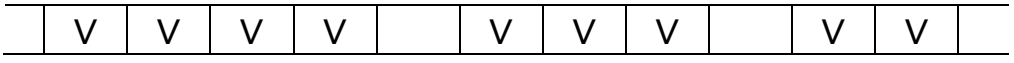
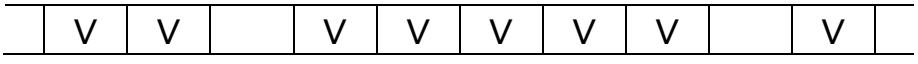
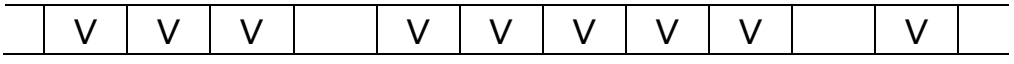
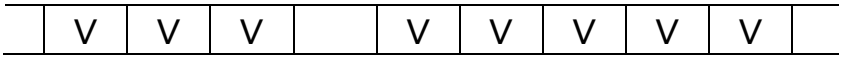
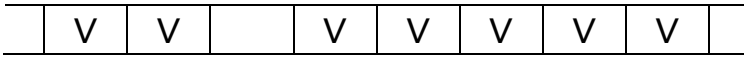
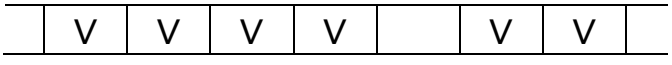
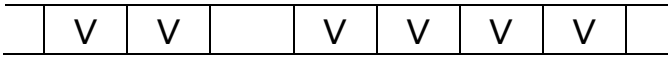
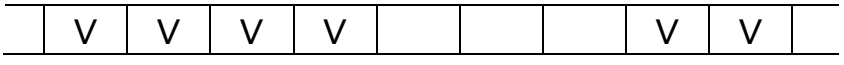
Командами 1–4 убирается крайняя слева метка и осуществляется проход в правый конец числа. Цикл, внутри которого будут дописываться новые три метки в счёт старой одной, начинается с команды 5. Командой 6 удаляется крайняя справа метка. Командами 7–10 осуществляется проход старых меток, разделительной ячейки и новых меток. Командами 11–15 добавляются новые три метки слева. Командами 16–19 осуществляется проход старых и новых меток и разделительной ячейки в правую сторону. Командой 20 цикл замыкается. Командой 21 ставится дополнительная метка.

Задания

Задание 1. Машина Поста определяется следующей программой:

1. ? 4, 2
2. × 3
3. → 9
4. ∨ 5
5. → 6
6. ? 7, 6
7. ∨ 8
8. ← 9
9. ? 11, 10
10. → 1
11. !

Изображая на каждом такте работы машины получающуюся конфигурацию, определите конечное состояние ленты. В начальный момент времени машина обозревает крайнюю левую ячейку с меткой.

1. 
2. 
3. 
4. 
5. 
6. 
7. 
8. 
9. 
10. 
11. 
12. 
13. 
14. 
15. 
16. 
17. 

18. | | V | V | V | | V | V | V | V | |

19. | | V | V | V | V | V | | V | V | |

20. | | V | V | V | V | V | | V | |

21. | | V | V | V | | V | | V | V | V | |

22. | | V | V | V | | V | V | | V | |

23. | | V | V | V | V | | V | V | V | |

24. | | V | V | V | | V | V | |

25. | | V | V | | V | | V | | V | V | V | |

26. | | V | V | V | | V | | V | V | |

27. | | V | V | V | | | | V | V | V | |

28. | | V | V | | V | V | | V | V | | V | |

29. | | V | V | | V | V | V | V | V | | V | |

30. | | V | V | V | V | V | V | |

31. | | V | V | | V | | V | | V | | V | |

32. | | V | V | V | V | | V | | V | |

33. | | V | V | V | | V | V | | V | V | |

34. | | V | V | V | V | | V | | V | V | |

35. | | V | V | | V | V | V | | V | |

36. | | V | V | V | | V | V | V | | V | |

37. | | V | V | V | | V | V | V | |

38. | | V | V | | V | | V | V | V | V | |

39. | | V | V | | V | V | V | V | V | V | V | |

40. | | V | V | V | V | V | V | | V | V | |

41. | | V | V | | V | V | V | V | V | V | |

42. | | V | V | V | V | | | V | V | | V | V | |

43. | | V | V | V | | V | V | V | V | V | V | |

44. | | V | V | V | V | V | V | V | | V | V | |

45. | | V | V | V | V | V | V | V | | V | |

46. | | V | V | | V | V | V | | V | V | V | |

47. | | V | V | V | V | V | | V | V | | V | |

48. | | V | V | V | V | V | V | | V | V | |

49. | | V | V | V | V | V | | | V | V | |

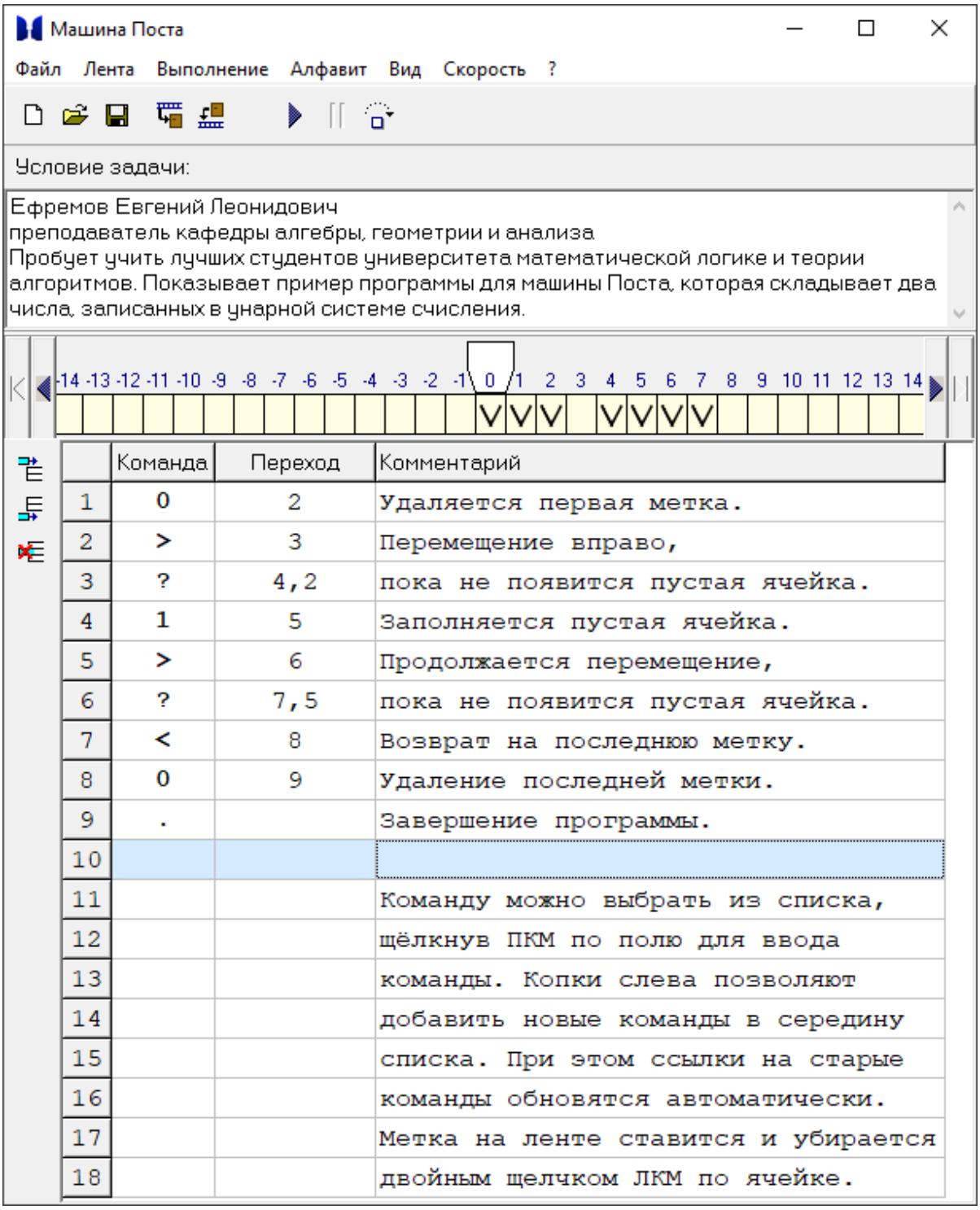
50. | | V | V | V | V | V | V | V | V | |

Задание 2. Напишите набор команд для машины Поста, вычисляющей функцию в унарной системе счисления. Значения $x, y \in \omega$ записаны на ленте в указанном порядке и отделены друг от друга пустой ячейкой. В начальный момент времени каретка обозревает крайнюю левую метку числа x.

При составлении программы следует иметь в виду комментарии, приводимые в задании 2, §1.

Для создания алгоритма используйте эмулятор машины Поста. На рисунке 2 показан скриншот эмулятора, разработанного К.Ю. Поляко-

вым и свободно скачиваемого с сайта http://kpolyakov.spb.ru/. Использование эмулятора удобно для проверки работы алгоритма и поиска ошибок. В поле «Условие задачи» необходимо указать следующую информацию: фамилия, имя, номер группы, номер варианта, текст задания. Интерфейс интуитивно понятный. За более подробными инструкциями можно обратиться к справке программы.



	Команда	Переход	Комментарий
1	0	2	Удаляется первая метка.
2	>	3	Перемещение вправо,
3	?	4,2	пока не появится пустая ячейка.
4	1	5	Заполняется пустая ячейка.
5	>	6	Продолжается перемещение,
6	?	7,5	пока не появится пустая ячейка.
7	<	8	Возврат на последнюю метку.
8	0	9	Удаление последней метки.
9	.		Завершение программы.
10			
11			Команду можно выбрать из списка,
12			щёлкнув ПКМ по полю для ввода
13			команды. Копки слева позволяют
14			добавить новые команды в середину
15			списка. При этом ссылки на старые
16			команды обновятся автоматически.
17			Метка на ленте ставится и убирается
18			двойным щелчком ЛКМ по ячейке.

Рис. 2. Пример работы на эмуляторе машины Поста

1. $\frac{4-3x+4y}{2}$.
2. $\frac{2x-3y-4}{4}$.
3. $\frac{2-3x+2y}{4}$.
4. $\frac{4-2y+2x}{3}$.
5. $\frac{3-3y+2x}{4}$.
6. $\frac{4-4y+3x}{2}$.
7. $\frac{2-3y+2x}{4}$.
8. $\frac{2-3y+4x}{3}$.
9. $\frac{3-2y+4x}{3}$.
10. $\frac{4-4y+2x}{3}$.
11. $\frac{3-3y+4x}{4}$.
12. $\frac{3x-4y-2}{3}$.
13. $\frac{2-2x+4y}{3}$.
14. $\frac{4-4x+3y}{2}$.
15. $\frac{2-4y+4x}{3}$.
16. $\frac{3-2x+5y}{4}$.
17. $\frac{3-4x+5y}{2}$.
18. $\frac{3-3x+4y}{2}$.
19. $\frac{4-2x+3y}{2}$.
20. $\frac{4-3y+4x}{2}$.
21. $\frac{3-2x+4y}{2}$.
22. $\frac{3-3x+2y}{4}$.
23. $\frac{2-2y+3x}{4}$.
24. $\frac{4-3x+2y}{2}$.
25. $\frac{3-4y+5x}{2}$.
26. $\frac{4-2x+2y}{3}$.
27. $\frac{2-3x+3y}{4}$.
28. $\frac{4-3y+2x}{2}$.
29. $\frac{4-3x+3y}{2}$.
30. $\frac{3-2y+3x}{4}$.
31. $\frac{2-4x+2y}{3}$.
32. $\frac{4-2y+3x}{2}$.
33. $\frac{2-3x+4y}{4}$.
34. $\frac{3-4y+3x}{2}$.
35. $\frac{2-4x+4y}{3}$.
36. $\frac{2-4y+2x}{3}$.
37. $\frac{4-2x+4y}{3}$.
38. $\frac{3-5y+2x}{2}$.
39. $\frac{4-3y+3x}{2}$.
40. $\frac{2-2x+3y}{4}$.
41. $\frac{4-2y+4x}{3}$.
42. $\frac{3-2x+3y}{4}$.
43. $\frac{3-5x+2y}{4}$.
44. $\frac{2-3y+3x}{4}$.
45. $\frac{2-4x+3y}{3}$.
46. $\frac{3-4x+2y}{3}$.
47. $\frac{2-2y+4x}{3}$.
48. $\frac{4-4x+2y}{3}$.
49. $\frac{3-4x+3y}{4}$.
50. $\frac{2-4y+3x}{4}$.

§ 4. Нормальные алгорифмы Маркова

Теория нормальных алгорифмов была разработана советским математиком А.А. Марковым в конце 40-х годов XX века. Марков считал, что слово «алгоритм» правильно произносить как «алгорифм», и сам использовал эту версию слова во время чтения лекций и написания научных работ. Со временем термин «нормальный алгорифм» стал устоявшимся.

Как и в случае с машинами Тьюринга, для начала дадим интуитивное описание и принцип работы алгоритмов, а потом сформулируем их строгое определение.

Описание и принцип работы

Пусть A – алфавит, не содержащий символов . и → (эти символы используются для записи команд программы). Пустое слово будем обозначать через λ.

Последовательность выражений

$$P_1 \to (\,.\,)Q_1$$
$$P_2 \to (\,.\,)Q_2$$
$$\dots$$
$$P_n \to (\,.\,)Q_n$$

где P_i, Q_j $(i, j \le n)$ – слова алфавита A и $P_i \neq P_j$ для всех $i \neq j$, называется *нормальным алгорифмом в алфавите* A. Каждое выражение в этой последовательности называется *марковской подстановкой*. Запись $(\,.\,)$ используется для того, чтобы показать, что наличие символа . необязательно.

Пусть на входе дано некоторое слово $\mathcal{B}$ в алфавите A. Если $i \le n$ – наименьший номер такой, что P_i является подсловом слова $\mathcal{B}$, то первое вхождение слова P_i заменяется на слово Q_i. Если в -ой подстановке при этом содержится символ . (т.е. она имеет вид $P_i \to .\,Q_i$), то выполнение алгорифма заканчивается. В противном случае все действия повторяются с преобразованным словом. Если среди слов $P_1, \dots, P_n$ нет ни одного подслова слова $\mathcal{B}$, то выполнение алгорифма заканчивается.

Примеры

Пример 1. Составим алгорифм, который добавит в начало любого слова произвольного алфавита слово «матлогика»:

$$\lambda \to .\,\text{матлогика}$$

Алгорифм состоит только из одной команды. Какое бы слово не поступало на вход, сначала осуществляется поиск λ. Очевидно, пустое слово является подсловом любого слова, и первое его вхождение – перед первым символом этого слова. Поэтому в начало поступающего на вход слова добавится слово «матлогика». Так как подстановка содержит, то выполнение алгорифма на этом закончится.

Пример 2. Составим алгорифм, который добавит в конец любого слова алфавита $A = \{a, b, c\}$ слово «матлогика»:

$$* a \to a *$$
$$* b \to b *$$
$$* c \to c *$$
$$* \to .\,\text{матлогика}$$
$$\lambda \to *$$

Так как поиск подслова в начальном слове осуществляется слева направо, то мы добавляем в начало входного слова символ $*$, прогоняем его в конец и заменяем $*$ на слово «матлогика»:

$$baba \vdash * baba \vdash b * aba \vdash ba * ba \vdash bab * a \vdash baba * \vdash baba\text{матлогика}$$

На этом примере заметим несколько важных моментов, которые нужно учитывать при составлении алгорифмов:

1. Добавление нового символа $\lambda \to *$ должно быть последней подстановкой. Если эту подстановку написать в начале или середине алгорифма, то все последующие подстановки никогда не будут выполнены: пустое слово является подсловом любого слова.

2. Замена дополнительных символов на новые слова должна осуществляться после того, как будут описаны какие-то действия с этими символами. Если после подстановки $* \to .\,\text{матлогика}$ поместить подстановку $* a \to a *$, то вторая никогда не выполнится, так как слово $*$ является подсловом слова $* a$.

3. Фактически составитель алгорифма не ограничен в алфавите A. Для составления подстановок можно (а порой и нужно) использо-

вать новые символы. В этом случае говорят, что задан *нормальный алгорифм над алфавитом* A.

На основе первых двух замечаний можно сделать вывод, что подстановки лучше записывать с конца. Конечно, не для всякой задачи есть необходимость составлять алгорифм с конца, но во многих случаях это бывает полезным.

Решим те же задачи, что мы решали в предыдущих параграфах.

Пример 3. Составим нормальный алгорифм, который вычисляет функцию $f(x) = x + 1$:

$$\lambda \to .1$$

Комментарии излишни.

Пример 4. Составим нормальный алгорифм, который вычисляет функцию $f(x, y) = x + y$. Числа x и y заданы в указанном порядке и отделены друг от друга символом $+$.

$$+1 \to .\lambda$$

Комментарии всё ещё излишни.

Пример 5. Составим нормальный алгорифм, который вычисляет функцию $f(x) = 3x$:

$$* 1 \to 111 *$$
$$11 * \to .\lambda$$
$$\lambda \to *$$

Пожалуй, дадим некоторые комментарии. Логика алгорифма проста: добавить новый символ в начало числа x (последняя подстановка $\lambda \to *$) и прогнать его в конец, заменяя одну единицу на три новые (первая подстановка $* 1 \to 111 *$). Когда $*$ окажется в конце, левее неё будет $3x + 3$ единицы, что на две больше нужного. Поэтому уберём лишние две единицы с конца вместе с ненужной звёздочкой (вторая подстановка $11 * \to .\lambda$). Обратите внимание: подстановку $11 * \to .\lambda$ мы поместили до подстановки $\lambda \to *$ (иначе до неё исполнитель никогда не добрался бы) и после подстановки $* 1 \to 111 *$ (иначе в момент, когда слева от $*$ будет две единицы, выполнение алгорифма сразу же закончится).

Задания

Задание 1. Пусть задан следующий нормальный алгорифм:

$$ab \to a * b$$
$$ba \to b * b$$
$$a * \to ba$$
$$* \to \lambda$$
$$b \to .b$$
$$\lambda \to *$$

Изображая последовательно каждое преобразование, выясните, какое слово будет выведено из слова

1. $baabaaba$
2. $abaaaaaa$
3. $aabbaaaa$
4. $babbaaba$
5. $abbaaaaa$
6. $bbbaaaba$
7. $aaabbaaa$
8. $baabbaba$
9. $abaabaaa$
10. $aabbbaaa$
11. $babbbaba$
12. $abbabbaa$
13. $aaabbbab$
14. $baabbbbb$
15. $abaabbab$
16. $bbaabbbb$
17. $aabbbbab$
18. $abbabaaa$
19. $bbbababa$
20. $aaabbaab$
21. $baabbabb$
22. $abaabaab$
23. $bbaababb$
24. $aabbbaab$
25. $aaabaaab$
26. $abaaaaab$
27. $bbaaaabb$
28. $abbabaab$
29. $bbbababb$
30. $baababba$
31. $abaaabaa$
32. $aabbabab$
33. $babbabbb$
34. $abbaabab$
35. $bbbaabbb$
36. $baabbbba$
37. $abaabbaa$
38. $bbaabbba$
39. $aabbbbaa$
40. $babbbbba$
41. $bbaaabba$
42. $aabbabaa$
43. $babbabba$
44. $bbbaabba$
45. $aaababab$
46. $baababbb$
47. $abaaabab$
48. $bbaaabbb$
49. $aabbaaab$
50. $abbaaaab$

Задание 2. Составьте нормальный алгорифм для вычисления функции в унарной системе счисления. Значения $x, y \in \omega$ записаны в указанном порядке и отделены друг от друга символом $*$.

При составлении программы стоит иметь в виду комментарии, приводимые в задании 2, §1.

Для создания алгоритма используйте эмулятор машины, реализующей нормальные алгорифмы. На рисунке 3 показан скриншот эмулятора, разработанного К.Ю. Поляковым и свободно скачиваемого с сайта http://kpolyakov.spb.ru/. Использование эмулятора удобно для проверки работы алгоритма и поиска ошибок. В поле «Условие задачи» необходимо указать следующую информацию: фамилия, имя, номер группы, номер варианта, текст задания. Интерфейс интуитивно понятный. За более подробными инструкциями можно обратиться к справке программы.

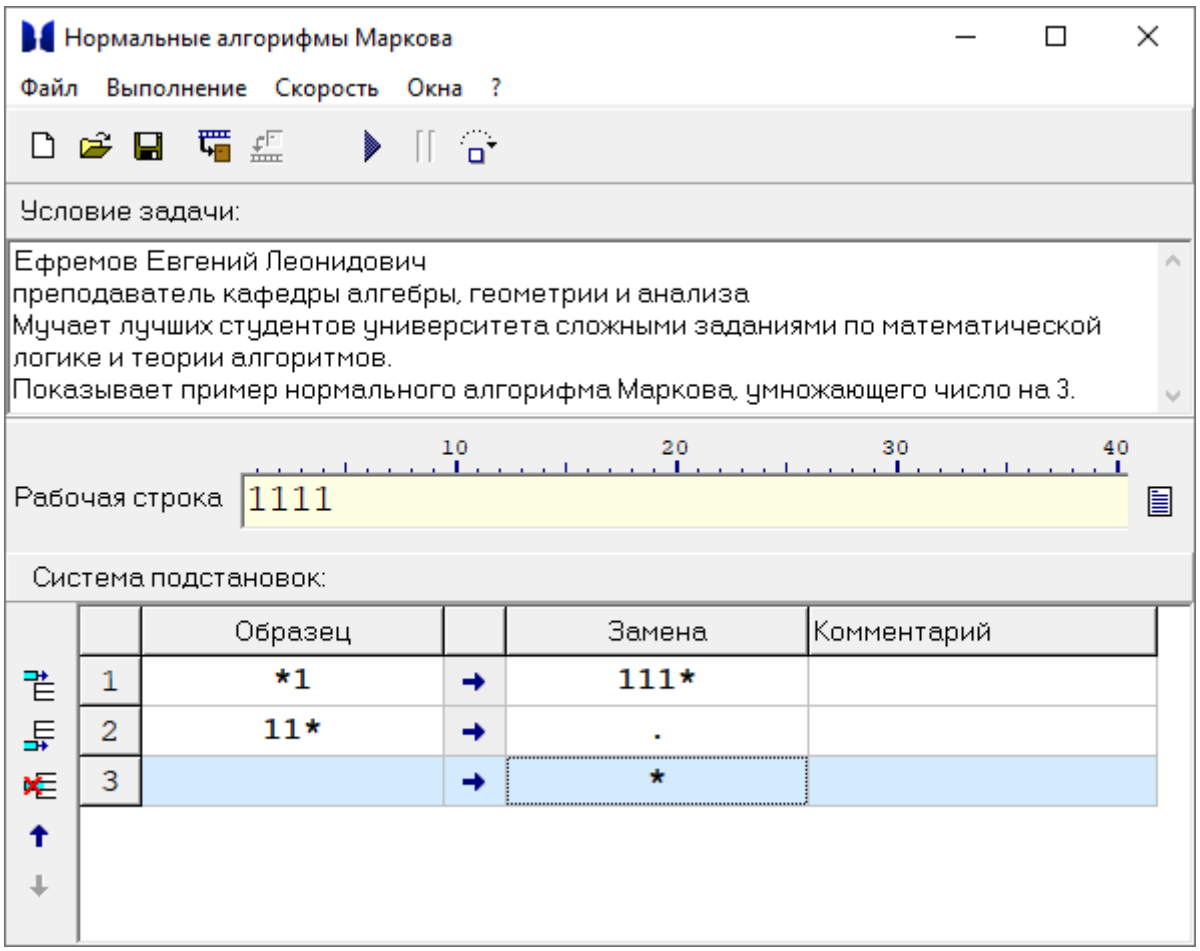


Рис. 3. Пример работы на эмуляторе нормальных алгорифмов Маркова

1. $\frac{3-4x+5y}{2}$.
2. $\frac{3-3x+4y}{2}$.
3. $\frac{4-2x+3y}{2}$.
4. $\frac{4-3y+4x}{2}$.
5. $\frac{3-2x+4y}{2}$.
6. $\frac{3-3x+2y}{4}$.
7. $\frac{2-2y+3x}{4}$.
8. $\frac{4-3x+2y}{2}$.
9. $\frac{3-4y+5x}{2}$.
10. $\frac{4-2x+2y}{3}$.
11. $\frac{2-3x+3y}{4}$.
12. $\frac{4-3y+2x}{2}$.
13. $\frac{4-3x+3y}{2}$.
14. $\frac{3-2y+3x}{4}$.
15. $\frac{2-4x+2y}{3}$.
16. $\frac{4-2y+3x}{2}$.
17. $\frac{2-3x+4y}{4}$.
18. $\frac{3-4y+3x}{2}$.
19. $\frac{2-4x+4y}{3}$.
20. $\frac{2-4y+2x}{3}$.
21. $\frac{4-2x+4y}{3}$.
22. $\frac{3-5y+2x}{2}$.
23. $\frac{4-3y+3x}{2}$.
24. $\frac{2-2x+3y}{4}$.
25. $\frac{4-2y+4x}{3}$.
26. $\frac{3-2x+3y}{4}$.
27. $\frac{3-5x+2y}{4}$.
28. $\frac{2-3y+3x}{4}$.
29. $\frac{2-4x+3y}{3}$.
30. $\frac{3-4x+2y}{3}$.
31. $\frac{2-2y+4x}{3}$.
32. $\frac{4-4x+2y}{3}$.
33. $\frac{3-4x+3y}{4}$.
34. $\frac{2-4y+3x}{4}$.
35. $\frac{4-3x+4y}{2}$.
36. $\frac{2x-3y-4}{4}$.
37. $\frac{2-3x+2y}{4}$.
38. $\frac{4-2y+2x}{3}$.
39. $\frac{3-3y+2x}{4}$.
40. $\frac{4-4y+3x}{2}$.
41. $\frac{2-3y+2x}{4}$.
42. $\frac{2-3y+4x}{3}$.
43. $\frac{3-2y+4x}{3}$.
44. $\frac{4-4y+2x}{3}$.
45. $\frac{3-3y+4x}{4}$.
46. $\frac{3x-4y-2}{3}$.
47. $\frac{2-2x+4y}{3}$.
48. $\frac{4-4x+3y}{2}$.
49. $\frac{2-4y+4x}{3}$.
50. $\frac{3-2x+5y}{4}$.

Определение нормального алгорифма

Пусть A – алфавит, S – множество всех слов алфавита A,

$P = \langle\langle \alpha_1, \beta_1, \varepsilon_1\rangle, \langle \alpha_2, \beta_2, \varepsilon_2\rangle, \ldots, \langle \alpha_n, \beta_n, \varepsilon_n\rangle\rangle \in (S \times S \times \{0,1\})^n$.

Пара $\mathfrak{N} = \langle A, P\rangle$ называется *нормальным алгорифмом в алфавите* A, если $\alpha_i \neq \alpha_j$ для любых $i, j \leq n$. Набор троек P называется *схемой алгоритма* $\mathfrak{N}$.

Пусть α – слово алфавита A. Если ни одно из слов $\alpha_1, \alpha_2, \ldots, \alpha_n$ не является подсловом слова α, то будем говорить, что α *не поддаётся* $\mathfrak{N}$.

Если i_0 – наименьшее число, для которого α_{i_0} является подсловом слова α, и β – результат замены первого вхождения α_{i_0} в α на слово β_{i_0}, то будем говорить, что $\mathfrak{N}$ *просто переводит* α *в* β, если $\varepsilon_{i_0} = 0$, и $\mathfrak{N}$ *заключительно переводит* α *в* β, если $\varepsilon_{i_0} = 1$.

Будем говорить, что *нормальный алгорифм* $\mathfrak{N}$ *преобразует слово* α *в слово* β, если существует последовательность $\gamma_0, \gamma_1, \ldots, \gamma_n$ слов алфавита A такая, что

1) $\gamma_0 \leftrightarrows \alpha$,
2) если $n = 0$, то α не поддаётся $\mathfrak{N}$,
3) $\mathfrak{N}$ просто переводит γ_i в γ_{i+1} для всех $i < n - 1$,
4) если $n > 0$, то имеет место только один из следующих случаев:
a) $\mathfrak{N}$ заключительно переводит γ_{n-1} в γ_n;
b) $\mathfrak{N}$ просто переводит γ_{n-1} в γ_n и γ_n не поддаётся $\mathfrak{N}$,
5) $\gamma_n \leftrightarrows \beta$.

Нормальный алгорифм $\mathfrak{N}$ может преобразовывать слово α только в одно слово. Если $\mathfrak{N}$ не преобразовывает α ни в какое слово, то будем говорить, что $\mathfrak{N}$ *не применим к* α или что $\mathfrak{N}(\alpha)$ *не определено*. Это возможно только в том случае, когда существует бесконечная последовательность $\gamma_0, \gamma_1, \ldots, \gamma_n, \ldots$ $(n \in \omega)$ слов, удовлетворяющих условиям 1) и 3) определения.

Составим нормальные алгорифмы для примеров 3–5. Будем использовать слово 1^x для записи числа $x \in \omega$ в унарной системе счисления. Через λ будем обозначать пустое слово.

Пример 1. Составим нормальный алгорифм $\mathfrak{N}_{+1}$, который преобразует слово 1^x в слово 1^{x+1}.

$$A = \{1\},$$
$$P_1 = \langle \lambda, 1, 1 \rangle,$$
$$P = \langle P_1 \rangle.$$

Нормальный алгорифм $\mathfrak{N}_{+1} = \langle A, P \rangle$ составлен. Покажем, например, что $\mathfrak{N}_{+1}(1^2) = 1^3$. Как обычно, запишем «процесс вычисления», оставив комментарии читателю:

$$\boldsymbol{\alpha} \leftrightarrows \mathbf{111}$$
$$\boldsymbol{\gamma_0} \leftrightarrows \mathbf{111}$$
$$P_1 = \langle \lambda, 1, 1 \rangle$$
$$\boldsymbol{\gamma_1} \leftrightarrows \mathbf{1111}$$
$$\boldsymbol{\beta} \leftrightarrows \mathbf{1111}$$

Пример 2. Составим нормальный алгорифм $\mathfrak{N}_{+}$, который преобразует слово $1^x + 1^y$ в слово 1^{x+y}.

$$A = \{1, +\},$$
$$P_1 = \langle +1, \lambda, 1 \rangle,$$
$$P = \langle P_1 \rangle.$$

Нормальный алгорифм $\mathfrak{N}_{+} = \langle A, P \rangle$ составлен. Покажем, например, что $\mathfrak{N}_{+}(1^1 + 1^2) = 1^3$:

$$\boldsymbol{\alpha} \leftrightarrows \mathbf{11 + 111}$$
$$\boldsymbol{\gamma_0} \leftrightarrows \mathbf{11 + 111}$$
$$P_1 = \langle +1, \lambda, 1 \rangle$$
$$\boldsymbol{\gamma_1} \leftrightarrows \mathbf{1111}$$
$$\boldsymbol{\beta} \leftrightarrows \mathbf{1111}$$

Пример 3. Составим нормальный алгорифм $\mathfrak{N}_{3\times}$, который преобразует слово 1^x в слово 1^{3x}.

$$A = \{1,*\},$$
$$P_1 = \langle * 1, 111 *, 0 \rangle,$$
$$P_2 = \langle 11 *, \lambda, 1 \rangle,$$
$$P_3 = \langle \lambda, *, 0 \rangle,$$
$$P = \langle P_1, P_2, P_3 \rangle.$$

Нормальный алгорифм $\mathfrak{N}_{3\times} = \langle A, P \rangle$ составлен. Покажем, например, что $\mathfrak{N}_{3\times}(1^2) = 1^6$:

$$\boldsymbol{\alpha \leftrightharpoons 111}$$
$$\boldsymbol{\gamma_0 \leftrightharpoons 111}$$
$$P_3 = \langle \lambda, *, 0 \rangle$$
$$\boldsymbol{\gamma_1 \leftrightharpoons * 111}$$
$$P_1 = \langle * 1, 111 *, 0 \rangle$$
$$\boldsymbol{\gamma_2 \leftrightharpoons 111 * 11}$$
$$P_1 = \langle * 1, 111 *, 0 \rangle$$
$$\boldsymbol{\gamma_3 \leftrightharpoons 111111 * 1}$$
$$P_1 = \langle * 1, 111 *, 0 \rangle$$
$$\boldsymbol{\gamma_4 \leftrightharpoons 11111111 *}$$
$$P_2 = \langle 11 *, \lambda, 1 \rangle$$
$$\boldsymbol{\gamma_5 \leftrightharpoons 1111111}$$
$$\boldsymbol{\beta \leftrightharpoons 1111111}$$

Упражнение 1. Составьте нормальный алгорифм, который добавит в конец любого слова алфавита $A = \{a, b, c\}$ слово «матлогика».

Упражнение 2. Сравните описание нормального алгорифма Маркова, приведённое в начале параграфа, и его строгое математическое определение. Сравните составленные алгоритмы для соответствующих примеров.

Сформулируем основное положение об «универсальности» нормальных алгорифмов.

Принцип нормализации. Любой алгоритм $\mathfrak{A}$ над конечным алфавитом A эквивалентен относительно A некоторому нормальному алгорифму $\mathfrak{N}$ над A, т.е. для любого слова α алфавита A выполняются следующие условия:

1) если $\mathfrak{A}$ применим к α, то и $\mathfrak{N}$ применим к α, причём $\mathfrak{A}(\alpha) = \mathfrak{N}(\alpha)$,

2) если $\mathfrak{N}$ применим к α, то и $\mathfrak{A}$ применим к α, причём $\mathfrak{A}(\alpha) = \mathfrak{N}(\alpha)$.

Если сузить принцип нормализации к частичным функциям, то получим следующее утверждение.

Принцип нормализации для частичных функций. Любая вычислимая частичная функция f нормально вычислима, т.е. существует нормальный алгорифм, вычисляющий f.

Таким образом, *класс всех вычислимых частичных функций совпадает с классом всех нормально вычислимых частичных функций*.

Дополнительные задачи

Задачи к § 2

1. Выяснить, является ли число, записанное в унарной системе счисления, чётным. Если да, то оставить 1. Если нет – 0.

2. Определить, делится ли записанное в десятичной системе счисления число на 5. Если делится, оставить слово ДА. В противном случае оставить слово НЕТ.

3. Алгоритм, возвращающий 1, если количество символов в последовательности из $*$ чётно, и 0 в противном случае. Данная последовательность из $*$ должна сохраниться.

4. Копирование символов EM. Количество копий задано числом в десятичной системе счисления.

5. На ленте записано слово латинского алфавита. Посчитать количество символов a.

6. На ленте через символ $*$ написано n наборов единиц. Оставить -ый набор. Число m записано набором единиц справа от последнего набора через пробел.

7. Найти целую часть числа $\frac{2x}{3}$. Число x записано в унарной системе счисления.

8. Алгоритм, определяющий, что в данном слове алфавита $\{0, 1\}$ за каждой 1 непосредственно следует 0. Если да, то выдать ответ 1, иначе 0.

9. Определить, равны ли два натуральных числа, записанных в десятичной системе счисления через символ $*$.

10. Если во введённом слове алфавита $\{a, b\}$ символов a больше, чем символов b, то выдать ответ a. Если символов a меньше, чем символов b, то выдать ответ b. Иначе в качестве ответа выдать пустое слово.

11. На ленте записано слово алфавита $\{a, b\}$. Собрать все символы a в левой части слова, все символы b – в правой.

12. На ленте записаны два слова алфавита $\{a, b\}$, разделенные $*$. Определить, встречается ли во втором слове первая буква первого слова. Если да, то выдать ответ 1, иначе – 0.

13. На ленте записано слово алфавита $\{a, b, c, d, (,)\}$. Определить правильность расстановки скобок в этом слове. Если расстановка правильная, оставить 1, иначе – 0.

14. Найти разность двух чисел. Числа записаны в двоичной системе счисления и отделены символом $*$.

15. Удаление символов E в последовательности из символов E и M, после чего все символы M будут расположены рядом.

16. Поиск максимальных по длине последовательностей единиц среди трёх записанных. Последовательности отделены символом $*$. Остальные последовательности должны быть стёрты.

17. На ленте записана последовательность единиц. Вывести число единиц в десятичной системе счисления.

18. Двоично-восьмеричный дешифратор.

19. Умножение двух чисел в унарной системе счисления.

20. Поиск частного от деления большего из двух указанных натуральных чисел, записанных в унарной системе счисления, на меньшее.

21. Изменение порядка букв в слове алфавита $\{a, b, c, d, e, f\}$ на обратный.

22. Модуль разности чисел в десятичной системе счисления. Числа разделены символом $*$.

23. На ленте в произвольном порядке подряд записаны 4 буквы латинского алфавита. Упорядочить последовательность по алфавиту.

24. НОД двух натуральных чисел, записанных в унарной системе счисления через символ $*$.

Задачи к § 3

Под *массивом* будем понимать последовательность подряд идущих меток, ограниченную пустыми ячейками. Под *числом* n будем понимать массив, содержащий $n + 1$ метку.

1. Определить, делится ли записанное на ленте число на 5. Если делится, оставить на ленте две метки, записанные через пустую ячейку. В противном случае лента должна остаться пустой.

2. Найти частное от деления записанного на ленте числа на 3.

3. Если количество меток в массиве кратно четырём, то стереть метки в этом массиве через одну. В противном случае стереть метки через две.

4. На ленте через пустую ячейку записано несколько чисел. Стереть все нечётные числа.

5. На ленте расположено несколько массивов, отделённых друг от друга пустыми ячейками. Определить количество массивов.

6. На ленте через пустую ячейку записаны два числа. Определить большее из них.

7. На ленте через пустую ячейку записаны два числа. Определить меньшее из них.

8. На ленте записан массив из нечётного количества меток. Стереть середину массива.

9. На ленте где-то есть метка. Найти её.

10. На ленте записан массив из чётного количества меток. Вписать в середину этого массива пустую ячейку (разделить на две половины).

11. На ленте записаны несколько меток, расположенных друг от друга на расстоянии одной пустой ячейки. Склеить их в один массив.

12. Дан массив меток. Раздвинуть массив так, чтобы после каждой метки была пустая ячейка. Первая метка массива должна остаться на своём месте.

13. На ленте через пустую ячейку записаны два числа. Поменять их местами.

14. Дан массив меток. Каретка располагается где-то над массивом, но не над крайними метками. Стереть все метки, кроме крайних, и поставить каретку в исходное положение.

Задачи к §4

1. Удвоить слово алфавита $A = \{a, b, c, d\}$.

2. В слове алфавита $A = \{a, b, c\}$ перенести все буквы a в конец слова, а буквы b – в начало.

3. В слове алфавита $A = \{a, b, c, d, e, \dots, x, y, z\}$ удвоить все согласные буквы.

4. Поменять местами первый и последний символы слова в алфавите $A = \{0, 1, 2, 3, 4, 5\}$.

5. В слове алфавита $A = \{a, b, c, d, e, \dots, x, y, z\}$ удвоить все буквы, стоящие на чётных местах в исходном слове.

6. Если в слове алфавита $A = \{a, b\}$ совпадают первый и последние символы, удалить их. В противном случае слово не менять.

7. В слове алфавита $A = \{a, b, c, d, e, f\}$ все символы a заменить на f, а все f – на af.

8. В слове алфавита $A = \{a, b, c, d, x, y, z\}$ заменить все вхождения последовательности xa на последовательность xax.

9. Циклический сдвиг символов слова алфавита $A = \{a, b, c, d, e, f\}$ вправо.

10. Упорядочить слово алфавита $A = \{a, b, c, d\}$.

11. Перевернуть слово в алфавите $A = \{a, b, c, d\}$.

12. Определить, является ли слово алфавита $A = \{a, b\}$ палиндромом.

13. Даны два числа в унарной системе счисления, записанные через $*$. Вместо звёздочки поставить $>$, $<$ или $=$.

14. Вычисление значения функции $f(x) = x \bmod 3$ в унарной системе счисления.

15. Считая слово P записью числа в унарной системе счисления, получить запись этого числа в троичной системе.

16. Перевод числа из двоичной системы счисления в восьмеричную.

17. Умножение на 3 числа в троичной системе счисления.

18. Считая непустое слово P записью числа в четверичной системе, найти остаток от деления этого числа на 4.

19. Определение системы счисления (найти возможное минимальное основание), в которой записано натуральное число. Предполагается, что основание не может превышать 16.

20. Считая непустое слово P записью двоичного числа, определить, является ли это число степенью 2 (1, 2, 4, ...). Ответ: слово 1, если является, или слово 0 в противном случае.

21. Определить, сбалансировано ли слово алфавита $A = \{(\ ,\ )\}$ по скобкам.

Заключение

Из тезиса Чёрча-Тьюринга и принципа нормализации следует эквивалентность понятий «машина Тьюринга» и «нормальный алгорифм».

Машины Поста не были столь подробно рассмотрены. Нетрудно понять, что понятие «машина Поста» эквивалентно понятию «машина Тьюринга». Как легко заметить, абсолютно любую машину Поста можно переписать на языке машин Тьюринга, задав внутренний алфавит $A = \{a_0, 1\}$. Обратное следует из того факта, что абсолютно любой конечный алфавит можно закодировать двоичным кодом.

Таким образом, любая задача, алгоритмически разрешимая относительно одного из рассматриваемых алгоритмов, алгоритмически разрешима относительно любого из них.

Существует много различных формализаций понятия «алгоритм», с помощью которых можно вычислять частичные функции. Читателю наверняка будут интересны машины с неограниченными регистрами и машины с ограниченным числом переменных, рассмотренные в приложениях. Также заинтересованным читателям рекомендуется изучить λ-исчисление, которое было разработано Чёрчем для формализации и анализа понятия «вычислимость».

Список литературы

1. Ершов Ю.Л., Палютин Е.А. Математическая логика. – М.: Физматлит, 2011. – 356 с.

2. Мальцев А.И. Алгоритмы и рекурсивные функции. – М.: Наука, 1986. – 368 с.

3. Верещагин Н.К., Шень А. Лекции по математической логике и теории алгоритмов. Часть 3. Вычислимые функции. – М.: МЦНМО, 2012. – 160 с.

4. Степанова А.А., Плешкова Т.Ю., Гусев Е.Г. Математическая логика и теория алгоритмов: практикум. – Владивосток: Изд-во ВГУЭС, 2010. – 48 с.

Приложение А
Машины с неограниченными регистрами

Машина с неограниченными регистрами (МНР) состоит из *ленты*, содержащей счётное число *регистров* (которые будем обозначать через R_1, R_2, R_3, …), и *программы*, состоящей из пронумерованного конечного списка *команд*. Каждый регистр R_n в любой момент времени содержит некоторое число $r_n \in \omega$.

Команды бывают следующих видов:

$Z(n)$ – обнулить R_n,

$S(n)$ – увеличить r_n на единицу,

$T(m,n)$ – заменить содержимое R_n числом r_m,

$J(m,n,q)$ – сравнить содержимое регистров R_m и R_n: если $r_m = r_n$, то перейти к выполнению команды с номером q.

Начальное состояние МНР – все регистры содержат число 0. Выполнение программы начинается с первой команды. Команды выполняются последовательно. МНР прекращает работу, если были выполнены все команды программы или если число q в команде $J(m,n,q)$ больше номера последней команды программы.

Пример. Следующая МНР вычисляет функцию $f(x,y) = x - y$:

1. $J(1,2,6)$
2. $S(2)$
3. $S(3)$
4. $J(1,2,6)$
5. $J(1,1,2)$
6. $T(3,1)$

Рассмотрим работу МНР при начальных значениях $x = 5$ и $y = 2$:

команда	R_1	R_2	R_3	R_4	R_5	…
	5	**2**	0	0	0	
$J(1,2,6)$	5	2	0	0	0	
$S(2)$	5	3	0	0	0	
$S(3)$	5	3	1	0	0	

$J(1,2,6)$	5	3	1	0	0	
$J(1,1,2)$	5	3	1	0	0	
$S(2)$	5	4	1	0	0	
$S(3)$	5	4	2	0	0	
$J(1,2,6)$	5	4	2	0	0	
$J(1,1,2)$	5	4	2	0	0	
$S(2)$	5	5	2	0	0	
$S(3)$	5	5	3	0	0	
$J(1,2,6)$	5	5	3	0	0	
$T(3,1)$	**3**	5	3	0	0	

Тезис. Класс всех вычислимых частичных функций совпадает с классом всех МНР-вычислимых функций.

Приложение Б
Машины с ограниченным числом переменных

Машина с ограниченным числом переменных (МОП) состоит из конечного набора *переменных* (которые будем обозначать через a, b, c, …, z), и *программы*, состоящей из пронумерованного конечного списка *команд*.

Каждая переменная может принимать любое значение из ω.

Команды бывают следующих видов:

$a := 0$ – присвоить переменной a значение 0,

$a := b$ – присвоить переменной a значение переменной b,

$a := b + 1$ – присвоить переменной a значение $b + 1$,

$a := b - 1$ – присвоить переменной a значение $b \dot{-} 1$,

goto n – перейти к выполнению команды с номером n,

if $a = 0$ then goto n else goto m – если $a = 0$, то перейти к выполнению команды с номером n, в противном случае – к команде с номером m,

stop – прекратить работу.

Начальное состояние МОП – все переменные принимают значение 0. Выполнение программы начинается с первой команды. Команды выполняются последовательно. МОП прекращает работу, если были выполнены все команды программы или если выполнена команда stop.

Пример. Следующая МОП вычисляет функцию $f(x, y) = x - y$:

1. if $b = 0$ then goto 5 else goto 2
2. $a := a - 1$
3. $b := b - 1$
4. goto 1
5. stop

Рассмотрим работу МОП при начальных значениях $x = 5$ и $y = 2$:

команда	a	b	c	d	…
	5	**2**	0	0	
if $b = 0$ then goto 5 else goto 2	5	2	0	0	
$a := a - 1$	4	2	0	0	
$b := b - 1$	4	1	0	0	
goto 1	4	1	0	0	
if $b = 0$ then goto 5 else goto 2	4	1	0	0	
$a := a - 1$	3	1	0	0	
$b := b - 1$	3	0	0	0	
goto 1	3	0	0	0	
if $b = 0$ then goto 5 else goto 2	3	0	0	0	
stop	**3**	0	0	0	

Тезис. Класс всех вычислимых частичных функций совпадает с классом всех МОП-вычислимых функций.

Учебное издание

Ефремов Евгений Леонидович

Алгоритмы вычисления частичных функций

Учебно-методическое пособие

Подписано в печать 09.04.2021 г.
Формат 60×84 / 16. Усл. печ. л. 2,79.
Тираж 100 экз. (1-й завод 1–32). Заказ xxx.

Дальневосточный федеральный университет
690922, Приморский край, г. Владивосток, о. Русский, п. Аякс, 10.

Отпечатано в Дальневосточном федеральном университете
690922, Приморский край, г. Владивосток, о. Русский, п. Аякс, 10.
(Типография Издательства ДВФУ,
690091, г. Владивосток, ул. Пушкинская, 10)