

# Элементы теории алгоритмов

# Зачем уточнять определение?

**Алгоритм** – точный набор инструкций для исполнителя, который приводит к решению задачи за конечное время.

**аль-Хорезми**: для любой математической задачи можно найти алгоритм решения, но для некоторых задач такие алгоритмы еще не найдены.

**К. Гёдель** (1931): в любой арифметике (натуральные числа, сложение, умножение) есть утверждение, которое нельзя ни доказать, ни опровергнуть (**теорема о неполноте**).



Всегда ли существует алгоритм?



Что такое алгоритм?

# Зачем уточнять определение?

*Задача:* алгоритм как математический объект.

## Теория алгоритмов (1930-е):

- доказательство алгоритмической неразрешимости задач
- анализ сложности алгоритмов
- сравнительная оценка качества алгоритмов



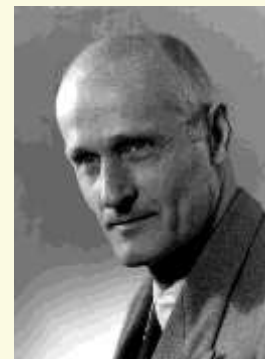
А. Тьюринг



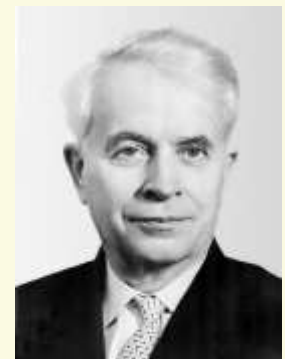
Э. Пост



А. Чёрч



С. Клини



А. Марков

# Что такое алгоритм?

Первые алгоритмы – правила арифметических действий:

- объекты – числа
- шаги – операции с однозначными числами



Что считать шагом?

Все объекты можно закодировать как символьные строки:



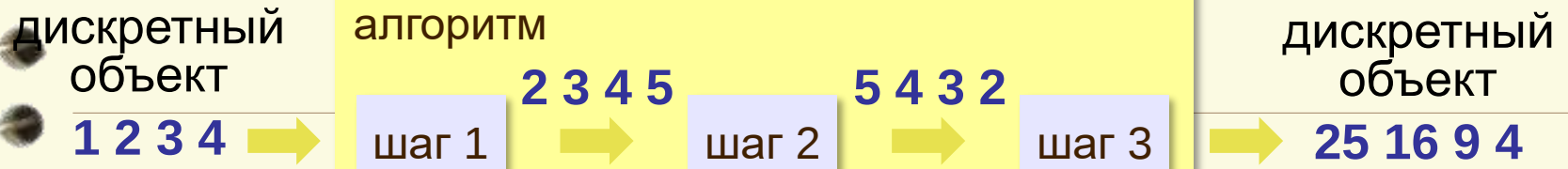
Можно рассматривать только алгоритмы обработки **строк**!

Из любого кода можно перевести в двоичный:



Можно рассматривать только алгоритмы обработки **битовых** строк!

# Как работает алгоритм?



- получает на вход дискретный объект
- в результате строит другой дискретный объект (или выдаёт сообщение об ошибке)
- обрабатывает объект по шагам
- на каждом шаге получается новый дискретный объект

# Как работает алгоритм?

ВХОДНОЕ СЛОВО

муха

алгоритм

слон

ВЫХОДНОЕ  
СЛОВО



Любой алгоритм определяет функцию!

т.е. правило преобразования входа в выход  
Функция не определена  $\Leftrightarrow$  алгоритм зацикливается  
или завершается аварийно.

ввод  $a, b$

вывод  $a * \text{sqrt}(b)$

→  $\times b < 0$

ввод  $a$

нц пока да

кц

→  $\times$  для всех  $a$

# Эквивалентные алгоритмы

Задают одну и ту же функцию:

если  $a < b$  то

$M := a$

иначе

$M := b$

все



$M := b$

если  $a < b$  то

$M := a$

все

# Универсальные исполнители

Алгоритм привязан к исполнителю  $\Rightarrow$  идея: построить универсального исполнителя.

Для любого алгоритма для любого исполнителя можно построить эквивалентный алгоритм для **универсального исполнителя.**

- если есть алгоритм для универсального исполнителя, то задача разрешима
- если доказано, что нет алгоритма для универсального исполнителя, задача неразрешима



Любой алгоритм может быть представлен как программа для универсального исполнителя!

# Универсальные исполнители

**Алгоритм** – это программа для универсального исполнителя.

## **Модель вычислений:**

- «*процессор*» (система команд и способ их выполнения)
- «*память*» (способ хранения данных)
- *язык программирования* (способ записи программ);
- *способ ввода данных*
- *способ вывода результата*

# Универсальные исполнители



А. Тьюринг

**машина  
Тьюринга**



Э. Пост

**машина  
Поста**



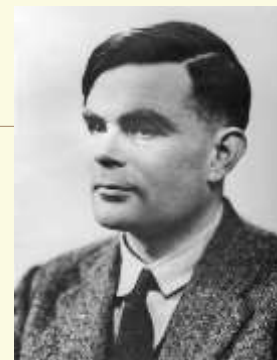
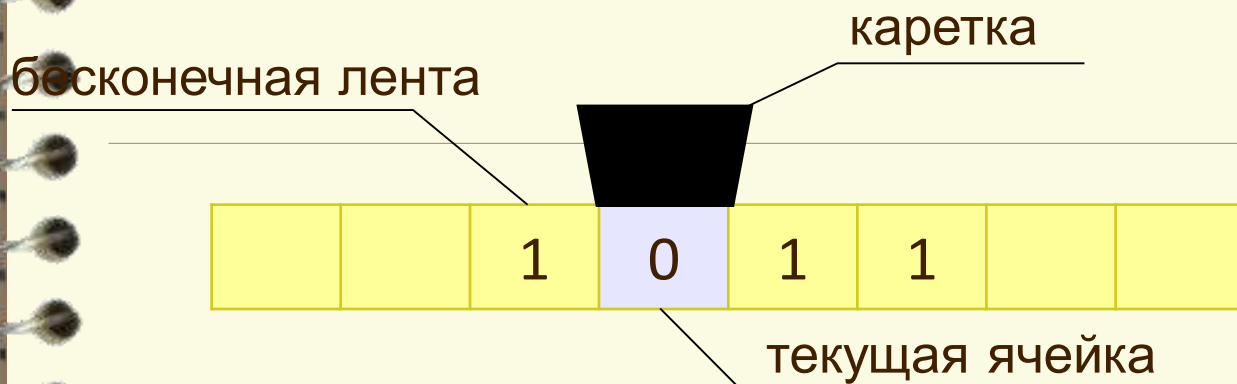
А. Марков

**нормальные  
алгоритмы  
Маркова**



Все универсальные исполнители эквивалентны!

# Машина Тьюринга



А. Тьюринг

- бесконечная лента («**память**»)
- каретка (**запись и чтение**)
- программируемый автомат («**процессор**»)

**алфавит:**  $A = \{a_1, a_2, \dots, a_N\}$

$A = \{0, 1, \square\}$

пробел

## Что такое автомат?

**Автомат** – это устройство, работающее без участия человека.

**Состояние** – промежуточная задача, которую решает автомат.

$$Q = \{q_1, q_2, \dots, q_M\}$$

начальное  
состояние

$q_0$  – остановка автомата

# Программа для машины Тьюринга

**Программа** состоит из команд:

- записать символ  $a_i$  в текущую ячейку
- переместить каретку  $\rightarrow \leftarrow$  • (не перемещать)
- перейти в состояние  $q_j$

$A = \{0, 1, \square\}$

$1 \rightarrow q_1$

- записать 1
- переместиться вправо
- перейти в состояние  $q_1$

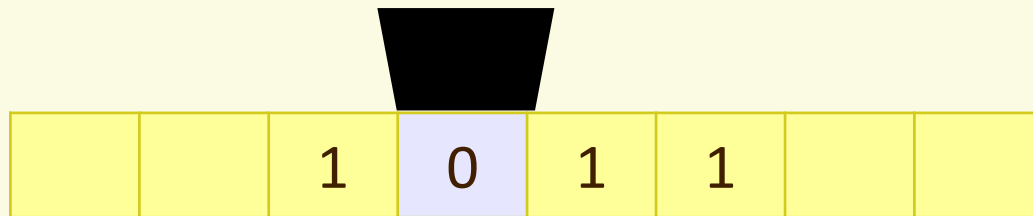
$0 \bullet q_0$

- записать 0
- не перемещать каретку
- останов ( $q_0$ )

# Программа для машины Тьюринга

**Задача.** На ленте записано число в двоичной системе счисления. Каретка находится где-то над числом. Требуется увеличить число на единицу.

**алфавит:**  $A = \{0, 1, \square\}$



**состояния:**  $q_1$  — поиск правого конца слова  
подзадачи  $q_2$  — увеличение числа на 1

# Программа для машины Тьюринга

## $q_1$ : ПОИСК КОНЦА СЛОВА

- если 0, то  $\rightarrow$
- если 1, то  $\rightarrow$
- если  $\square$ , то ...?  $\leftarrow$  и **переход в  $q_2$**

ТОЛЬКО  
изменения!

|           | $q_1$            |
|-----------|------------------|
| 0         | $\rightarrow$    |
| 1         | $\rightarrow$    |
| $\square$ | $\leftarrow q_2$ |

## $q_2$ : увеличение числа на 1

- если 0, то записать 1 и стоп ( $q_0$ )
- если 1, то записать 0 и  $\leftarrow$
- если  $\square$ , то записать 1 и стоп ( $q_0$ )

|           | $q_2$          |
|-----------|----------------|
| 0         | 1 • $q_0$      |
| 1         | 0 $\leftarrow$ |
| $\square$ | 1 • $q_0$      |



Как объединить две программы?

# Программа для машины Тьюринга

|           | $q_1$            |
|-----------|------------------|
| 0         | $\rightarrow$    |
| 1         | $\rightarrow$    |
| $\square$ | $\leftarrow q_2$ |

|           | $q_2$               |
|-----------|---------------------|
| 0         | $1 \rightarrow q_1$ |
| 1         | $0 \rightarrow q_1$ |
| $\square$ | $1 \bullet q_0$     |



Связь подзадач через ячейку ( $\square$ ,  $q_1$ )!

Если алгоритмы А и Б можно запрограммировать на машине Тьюринга, то и любую их комбинацию тоже можно запрограммировать.

**Тезис Чёрча-Тьюринга:** Любой алгоритм (в интуитивном смысле этого слова) может быть представлен как программа для машины Тьюринга.

# Программа для машины Тьюринга

начальное  
состояние

новая  
метка

переход

новое  
состояние

(0,  $q_1$ , 0,  $\rightarrow$ ,  $q_1$ )

(1,  $q_1$ , 1,  $\rightarrow$ ,  $q_1$ )

( $\square$ ,  $q_1$ ,  $\square$ ,  $\leftarrow$ ,  $q_2$ )

(0,  $q_2$ , 1,  $\bullet$ ,  $q_0$ )

(1,  $q_2$ , 0,  $\leftarrow$ ,  $q_2$ )

( $\square$ ,  $q_2$ , 1,  $\bullet$ ,  $q_0$ )

|           | $q_1$               |
|-----------|---------------------|
| 0         | $0 \rightarrow q_1$ |
| 1         |                     |
| $\square$ |                     |

|           | $q_2$              |
|-----------|--------------------|
| 0         | $1 \bullet q_0$    |
| 1         | $0 \leftarrow q_2$ |
| $\square$ | $1 \bullet q_0$    |

# Программы для машины Тьюринга

|           | $q_1$             |
|-----------|-------------------|
| 0         | $\leftarrow$      |
| 1         | $\leftarrow$      |
| $\square$ | $\rightarrow q_0$ |

|           | $q_1$             |
|-----------|-------------------|
| 0         | $\rightarrow q_0$ |
| 1         | $\rightarrow q_0$ |
| $\square$ | $\leftarrow$      |

|           | $q_1$        | $q_2$                |
|-----------|--------------|----------------------|
| 0         | $q_2$        | $\square \leftarrow$ |
| 1         | $q_2$        | $\square \leftarrow$ |
| $\square$ | $\leftarrow$ | $q_0$                |

? Что делает программа?

? Когда заиклиивается?

# Программы для машины Тьюринга

---

Задача 1. Уменьшить двоичное число на 1.

Задача 2. Увеличить на единицу число, записанное в десятичной системе счисления.

Задача 3. Уменьшить на единицу число, записанное в десятичной системе счисления.

Задача 4. Сложить два числа в двоичной системе, разделенные на ленте знаком «+».

Задача 5. Сложить два числа в десятичной системе, разделенные на ленте знаком «+».

# Элементы теории алгоритмов

# Вычислимые функции

ВХОДНОЕ СЛОВО

муха

алгоритм

слон

ВЫХОДНОЕ  
СЛОВО



Любой алгоритм определяет функцию!

т.е. правило преобразования входа в выход

**Вычислимая функция** – это функция, для вычисления которой существует алгоритм.

может задаваться разными алгоритмами:

$a \rightarrow 0$

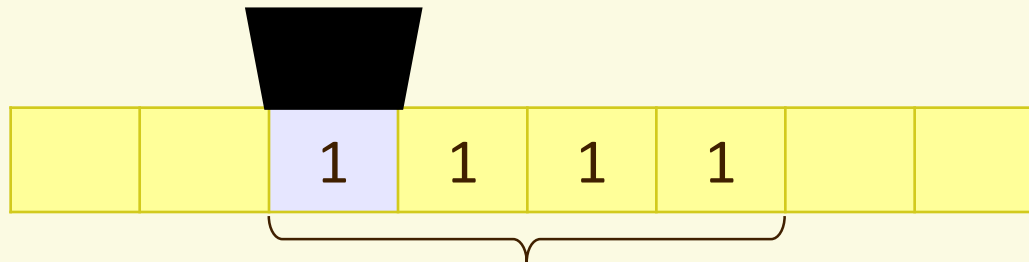
$b \rightarrow 1$

$b \rightarrow 1$

$a \rightarrow 0$

# Вычислимые функции

$$f(n) = \begin{cases} 1, & \text{если } n - \text{чётное} \\ 0, & \text{если } n - \text{нечётное} \end{cases}$$



в унарной системе счисления

|           | $q_1$          | $q_2$           | $q_3$        | $q_4$        |
|-----------|----------------|-----------------|--------------|--------------|
| 1         | $\rightarrow$  | $\rightarrow q$ | $\leftarrow$ | $\square$    |
|           | $q_2$          | 1               | $q_4$        | $\leftarrow$ |
| $\square$ | $\leftarrow q$ | $\leftarrow q$  |              | $q_0$        |
|           | 3              | 4               |              |              |

$q_1$  – чётное число единиц

$q_2$  – нечётное число единиц

$q_3$  – оставить 1 единицу

$q_4$  – стереть все единицы



Почему пусто?

# Вычислимые функции

$$f(n) = \begin{cases} 1, & \text{если } n - \text{чётное} \\ 0, & \text{если } n - \text{нечётное} \end{cases}$$

11 → ""  
1 → .  
→ 1.



Как написать НАМ?

**Невычислимая функция** (В.А. Успенский):

$$h(n) = \begin{cases} 1, & \text{если в записи числа } \pi \text{ есть } n \text{ стоящих подряд} \\ & \text{девяток в окружении других цифр} \\ 0, & \text{если такой цепочки нет} \end{cases}$$

перебор 800 знаков:

$$h(n) = 1 \text{ для } n = 1, 2, 6.$$



Если  $h(n)=0$ , перебор не остановится!

# Алгоритмически неразрешимые задачи

**Алгоритмически неразрешимая задача** – это задача, соответствующая невычислимой функции.

⇒ общего решения задачи нет, его бесполезно искать!

**10-я проблема Гильберта** (1900): найти метод, который позволяет определить, имеет ли заданное алгебраическое уравнение с целыми коэффициентами решение в целых числах.



Ю.В. Матиясевич

$$x^2 + y^3 + 2 = 0 \quad \Rightarrow (5; -3) \text{ и } (-5; -3)$$



1970: общего алгоритма **нет!**

# Алгоритмически неразрешимые задачи

**Г.В. Лейбниц, XVII в.:** разработать алгоритм, позволяющий установить, можно ли вывести формулу Б из формулы А в рамках заданной системы аксиом («**проблема распознавания выводимости**»).



1936: в общем виде задача **неразрешима!**



удалось получить отрицательные результаты



А. Чёрч

# Алгоритмически неразрешимые задачи

**Проблема останова:** по тексту любой программы  $P$  и ее входным данным  $X$  определяет, завершается ли программа  $P$  при входе  $X$  за конечное число шагов или зацикливается.

**Проблема эквивалентности:** по двум заданным алгоритмам определить, будут ли они выдавать одинаковые результаты для любых допустимых исходных данных.



Невозможно полностью автоматизировать отладку программ!