11. Напишите НАМ, который сортирует цифры двоичного числа так, чтобы сначала стояли все нули, а потом — все единицы.
12. Дополните приведённый в параграфе НАМ для удаления первого символа строки так, чтобы он не зацикливался на пустом слове.
13. Напишите НАМ, который умножает двоичное число на 2, добавляя 0 в конец записи числа.

§ 35
Алгоритмически неразрешимые задачи

Вычислимые и невычислимые функции

Мы уже говорили, что любой алгоритм задаёт некоторую функцию, которая для каждого входного слова, к которому применим алгоритм, однозначно задаёт результат — выходное слово. Такие функции называются вычислимыми.

Вычислимая функция — это функция, для вычисления которой существует алгоритм.

Любая вычислимая функция может задаваться разными алгоритмами (разными программами для выбранного универсального исполнителя). Например, следующие два нормальных алгорифма Маркова заменяют во входном двоичном слове все буквы «а» на нули и все буквы «б» на единицы:

а → 0	б → 1
б → 1	а → 0

Любая вычислимая функция может быть вычислена с помощью любого универсального исполнителя: машин Тьюринга и Поста, нормальных алгорифмов Маркова и др.

Рассмотрим, например, такую функцию, определённую для всех натуральных чисел:

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

Попробуем составить программу для машины Тьюринга, которая вычисляет эту функцию. Будем считать, что число записано в единичной системе счисления (в виде цепочки единиц[1]), и каретка в начальный момент стоит над самой левой единицей. Оказывается, такая программа действительно существует (рис. 5.7).

	q_1	q_2	q_3	q_4
1	$\to q_2$	$\to q_1$	$\leftarrow q_4$	□ ←
□	$\leftarrow q_3$	$\leftarrow q_4$		q_0

Рис. 5.7

Как принято, в начальный момент машина находится в состоянии q_1. Затем она движется вправо вдоль числа, поочередно переходя из состояния q_1 (пройдено чётное число единиц) в состояние q_2 (пройдено нечётное число единиц) и обратно. Таким образом, если встречен пробел и машина находится в состоянии q_2, то число нечётное и нужно просто стереть все единицы (состояние q_4). Если машина закончила просмотр в состоянии q_1, то число чётное; при этом нужно оставить одну единицу (состояние q_3) и перейти в состояние q_4 (стереть все остальные единицы). Обратите внимание, что ячейка (□, q_3) в таблице пустая – это невозможное состояние (покажите это самостоятельно).

Таким образом, рассмотренная функция вычислима, т. е. её можно вычислять с помощью машины Тьюринга, а значит, и с помощью любого универсального исполнителя. Например, нормальный алгорифм Маркова для алфавита $A = \{1\}$ выглядит так:

```
11 → ""
1 → .
→ 1.
```

В первой подстановке две соседние единицы удаляются (слово-замена здесь пустое, для ясности оно взято в кавычки, которыми можно ограничивать слова в НАМ). Это происходит до тех пор, пока не будут удалены все пары, поскольку эта подстановка стоит первой. Если остаётся одна единица, она удаляется с помощью второй подстановки, и работа программы заканчивается. Если все единицы удалены (число чётное), то с помощью третьей подстановки мы ставим одну единицу и останавливаем автомат.

[1] Пустая лента соответствует числу 0.

Существуют и **невычислимые функции**. Рассмотрим простой пример, предложенный В. А. Успенским в книге «Машина Поста». Известно, что математическая постоянная π — иррациональное число, его десятичная запись бесконечна и непериодична. Введем функцию $h(n)$, которая для любого натурального числа n равна 1, если в десятичной записи числа π есть n стоящих подряд девяток, окружённых другими цифрами, и равна нулю, если такой цепочки девяток нет. Как вычислить значение этой функции при некотором заданном n? Конечно, можно вычислять друг за другом десятичные знаки числа π (такие алгоритмы математикам известны!) и проверять, не нашлась ли в полученной последовательности цифр цепочка из n девяток. С помощью такого «наивного» алгоритма можно найти такие значения n, при которых $h(n) = 1$: обнаружив требуемую цепочку, алгоритм закончит работу. Например, анализ первых 800 знаков показывает, что $h(n) = 1$ при $n = 0, 1, 2, 6$. Но если для какого-то n функция $h(n)$ равна нулю, то «наивный» алгоритм никогда не остановится. Более того, для этой функции вообще не существует алгоритма, который при любом n останавливается и выдает значение $h(n)$ в качестве результата. Поэтому такая функция невычислима.

Когда задача алгоритмически неразрешима?

Как вы знаете, невозможно создать вечный двигатель, потому что это противоречит универсальным физическим законам сохранения. Точно так же в математике и информатике существуют задачи, для которых решение в общем виде отсутствует.

Поскольку алгоритм работает только с дискретными объектами, любая алгоритмическая задача — это функция, заданная на множестве дискретных объектов (входных слов).

Пусть, например, требуется по шахматной позиции определить, кто выигрывает при правильной игре — белые, чёрные или будет ничья. Построим функцию, соответствующую этому алгоритму. Для этого выберем способ кодирования, при котором каждая позиция может быть закодирована словом (символьной строкой) v в подходящем алфавите. Тогда приведённой задаче может соответствовать функция $f(v)$, заданная на множестве таких слов:

$$f(v) = \begin{cases} \text{'Б'}, & \text{если } v \text{ — код позиции, в которой выигрывают белые,} \\ \text{'Ч'}, & \text{если } v \text{ — код позиции, в которой выигрывают чёрные,} \\ \text{'0'}, & \text{если } v \text{ — код позиции, в которой будет ничья,} \\ \text{'?'}, & \text{если } v \text{ — ошибочный код позиции.} \end{cases}$$

Если функция, соответствующая задаче, вычислима, то задача называется **алгоритмически разрешимой** — для её вычисления можно построить алгоритм. Если определённая в задаче функция невычислима, то алгоритма для её решения не существует.

Алгоритмически неразрешимая задача — это задача, соответствующая невычислимой функции.

В 1900 г. на Международном математическом конгрессе в Париже известный математик Давид Гильберт сформулировал 23 нерешённые математические проблемы[1]. В знаменитой «десятой проблеме Гильберта» требуется найти метод, который позволяет определить, имеет ли заданное алгебраическое уравнение с целыми коэффициентами решение в целых числах. Например, уравнение

$$x^2 + y^3 + 2 = 0$$

Ю. В. Матиясевич (род. в 1947)

имеет два целочисленных решения, (5; –3) и (–5; –3). Сложность состояла в том, что требовалось найти единый метод (алгоритм), позволяющий решить задачу для *любого* такого уравнения со многими неизвестными.

В начале XX века была уверенность, что такой алгоритм есть, и поэтому его упорно искали. Однако в 1970 г. советскому математику Ю. В. Матиясевичу удалось доказать, что общего алгоритма решения этой задачи не существует.

Немецкий математик Г. В. Лейбниц в XVII веке безуспешно пытался найти метод проверки правильности любых математических утверждений. Как вы знаете, почти все математические теории основаны на использовании *аксиом* (положений, принимаемых без доказательства), из которых выводятся все остальные утверждения (*теоремы*). Задача заключалась в том, чтобы разработать алгоритм, позволяющий установить, можно ли вывести формулу Б из формулы А в рамках заданной системы аксиом

[1] Сейчас большинство из них решено полностью или частично.

(*«проблема распознавания выводимости»*). В 1936 г. американский математик А. Чёрч доказал, что эта задача *в общем виде* алгоритмически неразрешима, поэтому нельзя сформулировать универсальный алгоритм, пригодный для доказательства любой теоремы[1].

А. Чёрч (1903–1995)

Таким образом, уточнённые определения алгоритма, основанные на понятии универсальных исполнителей, сыграли в науке очень важную роль — позволили получить *отрицательные результаты*, т. е. доказать, что алгоритмов решения некоторых задач в общем виде не существует.

Для того чтобы доказать неразрешимость какой-то новой задачи, пытаются свести её к уже известным алгоритмически неразрешимым задачам. Если это удаётся, значит, и новая задача алгоритмически неразрешима[2].

Существуют также задачи, про которые неизвестно, алгоритмически разрешимы они или нет — решение не найдено, но алгоритмическая неразрешимость не доказана.

Алгоритмически неразрешимые задачи встречаются не только в математике, но и в информатике, например при разработке программ. Оказывается, невозможно написать программу для машины Тьюринга (алгоритм), которая по тексту любой программы P и её входным данным X определяет, завершается ли программа P при входе X за конечное число шагов или зацикливается. Это так называемая **проблема останова**. Её неразрешимость означает, в частности, что нельзя полностью автоматизировать тестирование любых программ, поручив это компьютеру. Однако для некоторых классов алгоритмов проблему останова решить можно. Например, линейная программа, не содержащая ветвлений и циклов, всегда завершится.

Было доказано, что алгоритмически неразрешима **проблема эквивалентности:** по двум заданным алгоритмам определить, будут ли они выдавать одинаковые результаты для любых допустимых исходных данных. Следовательно, невозможно полностью

[1] Тем не менее отдельные классы теорем можно доказывать на компьютере.

[2] Допустим, что (1) задача A неразрешима и (2) если мы можем построить алгоритм для решения задачи $Б$, то с его помощью можно построить алгоритм решения задачи A. Тогда задача $Б$ тоже неразрешима.

автоматизировать решение многих важных задач, связанных с разработкой программ, например:

- по заданному тексту программы определить, что она «делает»;
- определить, правильно ли работает программа при любых допустимых исходных данных;
- найти ошибку в программе, работающей неправильно.

Поэтому при отладке программы большую роль играет интуиция. Помогают (но не решают проблему полностью!) стандартные приёмы, позволяющие найти ошибку:

- сравнение результатов работы программы с результатами ручного счёта;
- эксперименты с программой при различных исходных данных для того, чтобы выявить закономерность появления ошибок;
- временное отключение (комментирование) частей программы и др.

Поскольку многие этапы разработки программного обеспечения в принципе невозможно представить в виде алгоритмов, программирование остаётся работой человека. Полностью поручить его компьютеру не удаётся, хотя решение некоторых задач всё же можно автоматизировать.

Вопросы и задания

1. Что такое вычислимая функция?
2. Приведите пример невычислимой функции.
3. Что такое алгоритмически неразрешимые задачи? Приведите известные вам примеры.
4. Что такое проблема останова? Каковы её следствия?
5. Что такое проблема эквивалентности?
6. Как можно доказать алгоритмическую неразрешимость новой задачи?

Задачи

1. В качестве доказательства того, что следующая функция вычислима, напишите программу для машины Поста.

$$f(n) = \begin{cases} 1, \text{ если } n \text{ — чётное;} \\ 0, \text{ если } n \text{ — нечётное.} \end{cases}$$