**Лекция № 1.** Формализация понятия алгоритма. Классификация алгоритмов. Модели вычислений

**ЦЕЛЬ**: Ознакомить с понятиями: Теория алгоритмов. Формализация понятия алгоритма. классификация алгоритмов. Модели вычислений. Вычислимые функции.Числовые алгоритмы**.**

**ВОПРОСЫ**:

1. Алгоритм и его свойства. Формализация понятия алгоритма.
2. Классификация алгоритмов.
3. Основные требования к алгоритмам .
4. Введение в теорию алгоритмов.
5. Модели вычислений.
6. Эквивалентность математических моделей понятия «алгоритм».
7. Тезисы теории алгоритмов.

**ОСНОВНЫЕ ПОНЯТИЯ:** Алгоритм. Формализация понятия алгоритма. Классификация алгоритмов. Теория алгоритмов. Модели вычислений. Вычислимые функции.Числовые алгоритмы**.**

**Обоснование алгоритма.**

Термин «алгоритм» появился в математике в XII веке и происходит от ла-тинской формы написания арабского имени среднеазиатского математика IX в. Мухаммеда ибн Мусса аль-Хорезми (Мухаммеда, сына Муссы из Хорезма, ок. 783 – ок. 850 гг.) – algorithmi

Первым дошедшим до нас алгоритмом в его интуитивном понимании – конечной последовательности элементарных действий, решающих поставленную задачу, считается предложенный Евклидом в III веке до нашей эры алгоритм нахождения наибольшего общего делителя двух чисел (алгоритм Евклида). Отметим, что в течение длительного времени, вплоть до начала XX века само слово «алгоритм» употреблялось в устойчивом сочетании «алгоритм Евклида». Для описания пошагового решения других математических задач использовалось слово «метод».

Начальной точкой отсчета современной теории алгоритмов можно считать работу немецкого математика Курта Гёделя [4] (1931 год - теорема о неполноте символических логик), в которой было показано, что некоторые математические проблемы не могут быть решены алгоритмами из некоторого класса. Общность результата Геделя связана с тем, совпадает ли использованный им класс алгоритмов с классом всех (в интуитивном смысле) алгоритмов. Эта работа дала толчок к поиску и анализу различных формализаций алгоритма.

В математическом энциклопедическом словаре приведена следующая формулировка понятия алгоритма:

«АЛГОРИТМ, алгорифм – точное предписание, которое задает вычислительный процесс (называемый в этом случае алгоритмическим), начинающийся с произвольного исходного данного (из некоторой совокупности возможных для данного алгоритма исходных данных) и направленный на получение полностью определяемого этими исходными данными результата».

Алгоритм — точное и конечное описание того или иного общего метода, основанного на применении исполнимых элементарных тактов обработки. Компьютер — вычислитель, он не понимает программу, а исполняет ее. Наиболее естественный способ указать компьютеру ход исполнения про­граммы — записать ее в виде алгоритма (на алгоритмическом языке). Современное значение слова "алгоритм" во многом аналогично таким понятиям, как рецепт, процесс, метод, способ. Санкт-Петербургская (Ленинградская) школа логики исторически использует другое написание этого слова — алгорифм, например, пишут "нормальные алгорифмы Маркова".

**Формализация понятия алгоритма.**

Почему необходимо формальное определение алгоритма. После того, как в 30-е годы прошлого века была доказана алгоритмическая неразрешимость некоторых задач в разных разделах математики, возникла совершенно новая ситуация. До тех пор, пока ученые верили в возможность построения алгоритмов для всех поставленных задач, не было повода уточнять интуитивное понятие алгоритма. Доказательство существования алгоритмически неразрешимых проблем означало, что существуют классы задач, для решения которых алгоритм не просто не найден, а его не будет найдено никогда. В отличие от конкретных алгоритмов, доказательство алгоритмической неразрешимости, т.е. доказательство невозможности, в котором содержалось бы высказывание обо всех мыслимых алгоритмах, потребовало формального уточнения понятия «алгоритм». Не имея формального определения, невозможно говорить обо всех мыслимых алгоритмах.

Частичная формализация понятия алгоритма началась с попыток решения проблемы разрешения, которую сформулировал Давид Гильберт в 1928 году. Следующие этапы формализации были необходимы для определения эффективных вычислений или «эффективного метода». Среди таких формализаций — рекурсивные функции Геделя — Эрбрана —Клини (1930 – 1935 г.г.), λ-исчисление Алонзо Чёрча (1936 г.). Понятие вычислимой функции было уточнено математиками А. Черчем, К. Геделем (1936 г.). Ими был определен класс частично рекурсивных функций, имеющих строгое математическое определение.

Одним из следующих формальных определений алгоритма является определение английского математика А.Тьюринга, который в 1936 году описал схему гипотетической (абстрактной) машины, имитирующей алгоритмические процессы, которые он и назвал алгоритмом в случае их успешной реализации, т.е. алгоритм – это то, что умеет делать такая машина. А если что-то не может быть сделано машиной Тьюринга, то это уже не является алгоритмом. Таким образом, А.Тьюринг формализовал правила выполнения действий при помощи описания работы некоторой конструкции. «Всякий алгоритм может быть реализован соответствующей машиной Тьюринга». Этот тезис является формальным определением алгоритма.

Можно нестрого определить алгоритм как однозначно трактуемую процедуру решения задачи. **Алгоритм** — это заданное на некотором языке конечное предписание, задающее конечную последовательность выполнимых и точно определенных элементарных операций для решения задачи, общее для класса возможных исходных данных.

Пусть DZ — область (множество) исходных данных задачи Z, a R — множество возможных результатов, тогда мы можем говорить, что алгоритм осуществляет отображение DZ → R. Поскольку такое отображение может быть не полным, то в теории алгоритмов вводятся следующие понятия: «Алгоритм называется **частичным алгоритмом,** если мы получаем результат только для некоторых d 6 DZ и **полным алгоритмом**, если алгоритм получает правильный результат для всех d € DZ». Отсутствует одно исчерпывающе строгое определение понятия «алгоритм». Из разнообразных вариантов словесного определения алгоритма наиболее удачные, принадлежат российским ученым А. Н. Колмогорову и А. А. Маркову ([8]):

Определение 1 (Колмогоров): Алгоритм — это всякая система вычислений, выполняемых по строго определенным правилам, которая после какого-либо числа шагов заведомо приводит к решению поставленной задачи.

Определение 2. (Марков): Алгоритм — это точное предписание, определяющее вычислительный процесс, идущий от варьируемых исходных данных к искомому результату.

Определение 3. (Д. Э. Кнут). «Алгоритм – это конечный набор правил, который определяет последовательность операций для решения конкретного множества задач и обладает пятью важными чертами: конечность, определённость, ввод, вывод, эффективность»

Отметим, что различные определения алгоритма, в явной или неявной форме, постулируют следующий ряд **общих требований:**

- алгоритм должен содержать конечное количество элементарно выполнимых предписаний, т. е. удовлетворять требованию конечности записи;

- алгоритм должен выполнять конечное количество шагов при решении задачи, т.е. удовлетворять требованию конечности действий;

- алгоритм должен быть единым для всех допустимых исходных данных, т. е. удовлетворять требованию универсальности;

- алгоритм должен приводить к правильному по отношению к поставленной задаче решению, т. е. удовлетворять требованию правильности.

Неудобства словесных определений связаны с проблемой однозначной трактовки терминов. В таких определениях должен быть, хотя бы неявно, указан исполнитель действий или предписаний. Алгоритм вычисления производной для полинома фиксированной степени вполне ясен тем, кто знаком с основами математического анализа, но для прочих он может оказаться совершенно непонятным. Это рассуждение заставляет нас указать так же вычислительные возможности исполнителя, а именно уточнить какие операции для него являются «элементарными». Другие трудности связаны с тем, что алгоритм заведомо существует, но его очень трудно описать в некоторой заранее заданной форме. В связи с этим формально строгие определения понятия алгоритма связаны с введением специальных математических конструкций — формальных алгоритмических систем или моделей вычислений, каковыми являются машина Поста, машина Тьюринга, рекурсивно-вычислимые функции Черча, и постулированием тезиса об эквивалентности такого формализма и понятия «алгоритм». Несмотря на принципиально разные модели вычислений, использующиеся в теории алгоритмов для определения термина «алгоритм», интересным результатом является формулировка гипотез о эквивалентности этих формальных определений в смысле их равномощности.

Формальные свойства алгоритмов

**Дискретност**ь— алгоритм должен представлять процесс решения задачи как последовательное выполнение некоторых простых шагов. Для выполнения каждого шага алгоритма требуется конечный отрезок времени, то есть преобразование исходных данных в результат осуществляется во времени дискретно.

**Детерминированность** — определённость. В каждый момент времени следующий шаг работы однозначно определяется состоянием системы. Таким образом, алгоритм выдаёт один и тот же результат (ответ) для одних и тех же исходных данных.

**Понятность** — алгоритм для исполнителя должен включать только те команды, которые ему (исполнителю) доступны, которые входят в его систему команд.

**Завершаемость (конечность)** — при корректно заданных исходных данных алгоритм должен завершать работу и выдавать результат за конечное число шагов. Вероятностный алгоритм может и никогда не выдать результат, но вероятность этого равна 0.

**Массовость** — универсальность. Алгоритм должен быть применим к разным наборам исходных данных.

**Классификация алгоритмов.** Существует два самых больших класса алгоритмов — это алгоритмы с повторением и рекурсивные алгоритмы. В основе алгоритмов с повторением лежат циклы и условные выражения; для анализ таких алгоритмов требуется оценить число операций, выполняемых внутри цикла, и число итераций цикла. Рекурсивные алгоритмы разбивают большую задачу на фрагменты и применяются к каждому фрагменту по отдельности. Такие алгоритмы называются иногда «разделяй и властвуй», и их использование может оказаться очень эффективным. Другие виды алгоритмов: общие [комбинаторные](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B1%D0%B8%D0%BD%D0%B0%D1%82%D0%BE%D1%80%D0%B8%D0%BA%D0%B0) алгоритмы, [алгоритмы на графах](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2), алгоритмы нахождения [максимального потока](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%BA%D1%81%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D0%BE%D1%82%D0%BE%D0%BA), алгоритмы нахождения [максимального паросочетания](http://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%BE%D1%81%D0%BE%D1%87%D0%B5%D1%82%D0%B0%D0%BD%D0%B8%D0%B5), [алгоритмы поиска](http://ru.wikipedia.org/w/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA%D0%B0&action=edit&redlink=1), [алгоритмы на строках](http://ru.wikipedia.org/w/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%BD%D0%B0_%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B0%D1%85&action=edit&redlink=1), [алгоритмы поиска строки](http://ru.wikipedia.org/w/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA%D0%B0_%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B8&action=edit&redlink=1), [алгоритмы вычисления расстояния между строками](http://ru.wikipedia.org/w/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D1%8B_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D1%80%D0%B0%D1%81%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D1%8F_%D0%BC%D0%B5%D0%B6%D0%B4%D1%83_%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B0%D0%BC%D0%B8&action=edit&redlink=1) [деревья для строковых последовательностей](http://ru.wikipedia.org/w/index.php?title=%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D1%8C%D1%8F_%D0%B4%D0%BB%D1%8F_%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%BE%D0%B2%D1%8B%D1%85_%D0%BF%D0%BE%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D0%B5%D0%B9&action=edit&redlink=1), [алгоритмы сортировки](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B8), [алгоритмы слияния](http://ru.wikipedia.org/w/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D1%81%D0%BB%D0%B8%D1%8F%D0%BD%D0%B8%D1%8F&action=edit&redlink=1), [алгоритмы сжатия без потерь](http://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%82%D0%B5%D0%B3%D0%BE%D1%80%D0%B8%D1%8F:%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D1%8B_%D1%81%D0%B6%D0%B0%D1%82%D0%B8%D1%8F_%D0%B1%D0%B5%D0%B7_%D0%BF%D0%BE%D1%82%D0%B5%D1%80%D1%8C), [алгоритмы сжатия с потерями](http://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%82%D0%B5%D0%B3%D0%BE%D1%80%D0%B8%D1%8F:%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D1%8B_%D1%81%D0%B6%D0%B0%D1%82%D0%B8%D1%8F_%D1%81_%D0%BF%D0%BE%D1%82%D0%B5%D1%80%D1%8F%D0%BC%D0%B8), построение [выпуклой оболочки](http://ru.wikipedia.org/wiki/%D0%92%D1%8B%D0%BF%D1%83%D0%BA%D0%BB%D0%B0%D1%8F_%D0%BE%D0%B1%D0%BE%D0%BB%D0%BE%D1%87%D0%BA%D0%B0) [набора](http://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE) точек, [алгоритмы распределённых систем](http://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D1%91%D0%BD%D0%BD%D1%8B%D0%B5_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B), алгоритмы выделения и освобождения памяти, алгоритмы в [операционных системах](http://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), дисковые алгоритмы-планировщики, алгоритмы синхронизации процессов и т.д.

Системы автоматизированного проектирования становятся все более интеллектуальными, на передний план выходят вопросы повышения эффективности вычислительных алгоритмов.

Таким образом, **уточнение понятия алгоритма** связано с уточнением алфавита данных и формы их представления, памяти и размещения в ней данных, элементарных шагов алгоритма и механизма реализации алгоритма. Однако эти понятия сами нуждаются в уточнении. Ясно, что их словесные определения потребуют введения новых понятий, для которых в свою очередь, снова потребуются уточнения и т.д.

**Поэтому в теории алгоритмов** **принят другой подход**, основанный на конкретной алгоритмической модели, в которой все сформулированные требования выполняются очевидным образом. При этом используемые алгоритмические модели универсальны, т.е. моделируют любые другие разумные алгоритмические модели, что позволяет снять возможное возражение против такого подхода: не приводит ли жесткая фиксация алгоритмической модели к потере общности формализации алгоритма? Поэтому данные алгоритмические модели отождествляются с формальным понятием алгоритма.

В дальнейшем будут рассмотрены основные типы алгоритмических моделей, различающиеся исходными трактовками, что такое алгоритм.

Первый тип трактует алгоритм как некоторое детерминированное устройство, способное выполнять в каждый момент лишь строго фиксированное множество операций. Основной теоретической моделью такого типа является машина Тьюринга, предложенная им в 30-х годах и оказавшая существенное влияние на понимание логической природы разрабатываемых ЭВМ. Другой теоретической моделью данного типа является машина произвольного доступа ( МПД ) – введенная достаточно недавно ( в 70-х годах ) с целью моделирования реальных вычислительных машин и получения оценок сложности вычислений.

Второй тип связывает понятие алгоритма с традиционным представлением – процедурами вычисления значений числовых функций. Основной теоретической моделью этого типа являются рекурсивные функции – исторически первая формализация понятия алгоритма.

Третий тип алгоритмических моделей – это преобразования слов в произвольных алфавитах, в которых операциями являются замены кусков слов другим словом. Основной теоретической моделью этого типа являются нормальные алгоритмы Маркова.

**С алгоритмами связаны приведенные ниже области исследований:**

1. **Анализ алгоритмов.** Предмет этой области состоит в том, чтобы для заданного алгоритма определить рабочие характеристики. Например, часто желательно, чтобы алгоритм был быстрым.
2. **Теория алгоритмов.** В этой области рассматриваются вопросы существования или не существования эффективных алгоритмов вычисления определенных величин.
3. **Построение алгоритмов.** В этой области рассматриваются стандартные  
   приемы и методы, используемые при написании алгоритмов.

**Теория алгоритмов оказала существенное влияние на развитие ЭВМ и практику программирования.** В теории алгоритмов были предугаданы основные концепции, заложенные в аппаратуру и языки программирования ЭВМ. Упоминаемые выше главные алгоритмические модели математически эквивалентны; но на практике они существенно различаются сложностными эффектами, возникающими при реализации алгоритмов, и породили разные направления в программировании. Так, микропрограммирование строится на идеях машин Тьюринга, структурное программирование заимствовало свои конструкции из теории рекурсивных функций, языки символьной обработки информации (РЕФАЛ, ПРОЛОГ) берут начало от нормальных алгорифмов Маркова и систем Поста.

**Теория алгоритмов** - это наука, изучающая общие свойства и закономерности алгоритмов, разнообразные формальные модели их представления. На основе формализации понятия алгоритма возможно сравнение алгоритмов по их эффективности, проверка их эквивалентности, определение областей применимости.

К **задачам теории алгоритмов** относятся формальное доказательство алгоритмической неразрешимости задач, асимптотический анализ сложности алгоритмов, классификация алгоритмов в соответствии с классами сложности, разработка критериев сравнительной оценки качества алгоритмов и т. п.

Разработанные в 1930-х годах разнообразные формальные модели алгоритмов (Пост, Тьюринг, Черч), равно как и предложенные в 1950-х годах модели Колмогорова и Маркова, оказались эквивалентными в том смысле, что любой класс проблем, разрешимых в одной модели, разрешимы и в другой. В настоящее время полученные на основе теории алгоритмов практические рекомендации получают всё большее распространение в области проектирования и разработки программных систем .

**Современное состояние теории алгоритмов.** В настоящее время теория алгоритмов развивается, главным образом, по трем направлениям.

1. **Классическая теория алгоритмов** изучает проблемы формулировки задач в терминах [формальных языков](http://ru.wikipedia.org/wiki/Ð¤Ð¾ÑÐ¼Ð°Ð), вводит понятие задачи разрешения, проводит классификацию задач по [классам сложности](http://ru.wikipedia.org/wiki/ÐÐ) ([P](http://ru.wikipedia.org/wiki/ÐÐ), [NP](http://ru.wikipedia.org/wiki/ÐÐ) и др.).
2. **Теория асимптотического анализа алгоритмов** рассматривает методы получения асимптотических оценок ресурсоемкости или времени выполнения алгоритмов, в частности, для рекурсивных алгоритмов. Асимптотический анализ позволяет оценить рост потребности алгоритма в ресурсах (например, времени выполнения) с увеличением объема входных данных.
3. **Теория практического анализа вычислительных алгоритмов** решает задачи получения явных функции трудоёмкости, интервального анализа функций, поиска практических критериев качества алгоритмов, разработки методики выбора рациональных алгоритмов.

К теории алгоритмов тесно примыкают исследования, связанные с разработкой методов создания эффективных алгоритмов (динамическое программирование, метод ветвей и границ, метод декомпозиции, «жадные» алгоритмы, специальные структуры данных и т. д.)

Обобщая исследования в различных разделах теории алгоритмов, можно выделить следующие **основные задачи и направления развития, характерные для современной теории алгоритмов:**

- формализация понятия «алгоритм» и исследование формальных алгоритмических систем (моделей вычислений);

- доказательство алгоритмической неразрешимости задач;

- формальное доказательство правильности и эквивалентности алгоритмов;

- классификации задач, определение и исследование сложностных классов;

- доказательство теоретических нижних оценок сложности задач;

- получение методов разработки эффективных алгоритмов;

- асимптотический анализ сложности итерационных алгоритмов;

- исследование и анализ рекурсивных алгоритмов;

- получение явных функций трудоемкости алгоритмов;

- разработка классификаций алгоритмов;

- исследование емкостной (по ресурсу памяти) сложности задач и алгоритмов;

- разработка критериев сравнительной оценки ресурсной эффективности алгоритмов и методов их сравнительного анализа.

Полученные в теории алгоритмов результаты находят сегодня достаточно широкое практическое применение, в рамках которого можно выделить два следующих аспекта:

**Теоретический аспект:** при исследовании некоторой задачи результаты теории алгоритмов позволяют ответить на вопрос - является ли эта задача в принципе алгоритмически разрешимой. Для алгоритмически неразрешимых задач возможно их сведение к задаче останова машины Тьюринга. В случае алгоритмической разрешимости задачи следующим важным теоретическим вопросом является вопрос о принадлежности этой задачи к классу NP-полных задач. При утвердительном ответе можно говорить о существенных временных затратах для получения точного решения этой задачи для больших размерностей исходных данных, иными словами — об отсутствии быстрого точного алгоритма ее решения.

**Практический аспект:** методы и методики теории алгоритмов, в основном разделов асимптотического и практического анализа, позволяют осуществить:

- рациональный выбор из известного множества алгоритмов решения данной задачи, учитывающий особенности их применения в разрабатываемой программной системе. Например, в условиях ограничений на время выполнения программной реализации алгоритма, и в условиях критерия минимального объема дополнительно используемой алгоритмом памяти, выбор, скорее всего, будет сделан в пользу различных алгоритмов;

- получение временных оценок решения сложных задач на основе функции трудоемкости;

- получение достоверных оценок невозможности решения некоторой задачи за определенное время. Такого рода обратные по временной эффективности задачи оказываются сегодня также востребованы, например, для криптографических методов;

- разработку и совершенствование эффективных алгоритмов решения практически значимых задач в области обработки информации.

**Модели вычислений:**

1. **Машина Тьюринга**
2. **Лямбда-исчисление** — рассматривается пара — λ-выражение и его аргумент, — а вычислением считается применение, или апплицирование первого члена пары ко второму. Это позволяет отделить функцию и то, к чему она применяется. В более общем случае вычислением считаются цепочки, начинающиеся с исходного λ-выражения, за которым следут конечная последовательность λ-выражений, каждое из которых получается из предыдущего применением β-редукции, то есть правила подстановки.
3. **Комбинаторная логика** — трактовка вычисления сходна с λ-исчислением, но имеются и важные отличия (например, комбинатор неподвижной точки Y имеет нормальную форму в комбинаторной логике, а в λ-исчислении — не имеет). Комбинаторная логика была первоначально разработана для изучения природы парадоксов и для построения концептуально ясных оснований математики, причем представление о переменной исключалось вовсе, что помогало прояснить роль и место переменных в математике.

**Вычислимые функции**

**Вычислимые функции** — это множество функций вида, http://upload.wikimedia.org/math/5/b/9/5b9a207130d8ca06d676488f2e8cb66c.png которые могут быть реализованы на машине Тьюринга. Задачу вычисления функции f называют **алгоритмически разрешимой** или **алгоритмически неразрешимой**, в зависимости от того, возможно ли написать алгоритм, вычисляющий эту функцию.

В качестве множества N обычно рассматривается множество B* — множество слов в двоичном алфавите B=\{0,1\}, с оговоркой, что результатом вычисления может быть не только слово, но и специальное значение «неопределённость», соответствующее случаю, когда алгоритм «зависает». Таким образом, можно дать следующее определение N:

N = B^* \cup \{ undef \}, где B=\{0,\;1\}, а undef — специальный элемент, означающий неопределённость. Роль множества N может играть множество натуральных чисел, к которому добавлен элемент undef, и тогда вычислимые функции — это некоторое подмножество натуральнозначных функций натурального аргумента. Удобно считать, что в качестве N могут выступать различные счётные множества — множество натуральных чисел, множество рациональных чисел, множество слов в каком-либо конечном алфавите и др. Важно, чтобы существовал некоторый формальный язык в конечном алфавите описания элементов множества N и чтобы задача распознавания корректных описаний была вычислима. Например, для описания натуральных чисел удобно использовать двоичную систему счисления — язык описания натуральных чисел в алфавите B.

В данном определении вместо исполнителя машин Тьюринга можно взять один из [Тьюринг-полных](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%BD%D0%BE%D1%82%D0%B0_%D0%BF%D0%BE_%D0%A2%D1%8C%D1%8E%D1%80%D0%B8%D0%BD%D0%B3%D1%83) исполнителей. Грубо говоря, «эталонным исполнителем» может быть некоторый абстрактный компьютер, подобный используемым персональным компьютерам, но с потенциально бесконечной памятью и особенностями архитектуры, позволяющими использовать эту бесконечную память.

Важно отметить, что множество программ для этого исполнителя (например, множество машин Тьюринга при фиксированном алфавите входных и выходных данных) [счётно](http://ru.wikipedia.org/wiki/%D0%A1%D1%87%D1%91%D1%82%D0%BD%D0%BE%D0%B5_%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE). Поэтому множество вычислимых функций не более чем счётно, в то время как множество функций вида http://upload.wikimedia.org/math/5/b/9/5b9a207130d8ca06d676488f2e8cb66c.png несчётно, если N счётно. Значит, есть невычислимые функции, причём их [мощность](http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D1%89%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B0) больше, чем мощность вычислимых функций. Примером невычислимой функции (алгоритмически неразрешимой проблемы) может быть [функция определения останова](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B1%D0%BB%D0%B5%D0%BC%D0%B0_%D0%BE%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%B0) — функция, которая получает на вход описание некоторой машины Тьюринга и вход для неё, а возвращает 0 или 1 в зависимости от того, остановится данная машина на данном входе или нет.

Понимание, что часть функций вида http://upload.wikimedia.org/math/2/1/5/21547812009b75aa081ad2255d8db8ee.png вычислима, а часть нет, появилось ещё до появления первых компьютеров. [Теория вычислимости](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D0%B8) берёт своё начало от диссертации [Тьюринга](http://ru.wikipedia.org/wiki/%D0%A2%D1%8C%D1%8E%D1%80%D0%B8%D0%BD%D0%B3) ([1936](http://ru.wikipedia.org/wiki/1936)), в которой он ввёл понятие абстрактной вычислительной машины, и функций, вычислимых на ней. По мере развития [теории вычислимости](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D0%B8) было сформулировано несколько определений, которые, как оказалось впоследствии, определяют одно и то же множество функций — множество вычислимых функций:

* функции, реализуемые на [машинах Тьюринга](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%B0_%D0%A2%D1%8C%D1%8E%D1%80%D0%B8%D0%BD%D0%B3%D0%B0) ([Алан Тьюринг](http://ru.wikipedia.org/wiki/%D0%A2%D1%8C%D1%8E%D1%80%D0%B8%D0%BD%D0%B3))
* функции, реализуемые на [нормальных алгорифмах Маркова](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%84%D0%BC_%D0%9C%D0%B0%D1%80%D0%BA%D0%BE%D0%B2%D0%B0) ([А. А. Марков](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%80%D0%BA%D0%BE%D0%B2,_%D0%90%D0%BD%D0%B4%D1%80%D0%B5%D0%B9_%D0%90%D0%BD%D0%B4%D1%80%D0%B5%D0%B5%D0%B2%D0%B8%D1%87_(%D0%BC%D0%BB%D0%B0%D0%B4%D1%88%D0%B8%D0%B9)))
* функции, реализуемые на [машине Поста](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%B0_%D0%9F%D0%BE%D1%81%D1%82%D0%B0)
* [частично рекурсивные функции](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BA%D1%83%D1%80%D1%81%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D0%B8)) ([Курт Гёдель](http://ru.wikipedia.org/wiki/%D0%93%D1%91%D0%B4%D0%B5%D0%BB%D1%8C,_%D0%9A%D1%83%D1%80%D1%82" \o "Гёдель, Курт), [Стивен Клини](http://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%BD%D0%B8,_%D0%A1%D1%82%D0%B8%D0%B2%D0%B5%D0%BD_%D0%9A%D0%BE%D1%83%D0%BB))
* функции, реализуемые на [регистровой машине](http://ru.wikipedia.org/w/index.php?title=%D0%A0%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0&action=edit&redlink=1)

### Абстрактные исполнители и формальные системы вычислений:

* [Нормальный алгорифм Маркова](http://ru.wikipedia.org/wiki/ÐÐ¾ÑÐ¼Ð°Ð) ([продукционное программирование](http://ru.wikipedia.org/w/index.php?title=%D0%9F%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1))
* [Машина Поста](http://ru.wikipedia.org/wiki/ÐÐ°ÑÐ¸Ð½Ð°_ÐÐ¾ÑÑÐ°) ([автоматное программирование](http://ru.wikipedia.org/wiki/ÐÐ²ÑÐ¾Ð¼Ð°ÑÐ½Ð¾Ðµ_Ð¿ÑÐ¾Ð³ÑÐ°Ð¼Ð¼Ð¸ÑÐ¾Ð²Ð°Ð½Ð¸Ðµ))
* [Рекурсивная функция (теория вычислимости)](http://ru.wikipedia.org/wiki/Ð ÐµÐºÑÑÑÐ¸Ð²Ð½Ð°Ñ_ÑÑÐ½ÐºÑÐ¸Ñ_(ÑÐµÐ¾ÑÐ¸Ñ_Ð²ÑÑÐ¸ÑÐ)
* [Лямбда-исчисление](http://ru.wikipedia.org/wiki/ÐÑÐ¼Ð±Ð´Ð°-Ð¸ÑÑÐ¸ÑÐ) ([функциональное программирование](http://ru.wikipedia.org/wiki/Ð¤ÑÐ½ÐºÑÐ¸Ð¾Ð½Ð°Ð))
* [Brainfuck](http://ru.wikipedia.org/wiki/Brainfuck) ([императивное программирование](http://ru.wikipedia.org/wiki/ÐÐ¼Ð¿ÐµÑÐ°ÑÐ¸Ð²Ð½Ð¾Ðµ_Ð¿ÑÐ¾Ð³ÑÐ°Ð¼Ð¼Ð¸ÑÐ¾Ð²Ð°Ð½Ð¸Ðµ))

**К числовым алгоритмам относятся:** [Алгоритмы факторизации](http://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%82%D0%B5%D0%B3%D0%BE%D1%80%D0%B8%D1%8F:%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D1%8B_%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8).[Тесты простоты](http://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%82%D0%B5%D0%B3%D0%BE%D1%80%D0%B8%D1%8F:%D0%A2%D0%B5%D1%81%D1%82%D1%8B_%D0%BF%D1%80%D0%BE%D1%81%D1%82%D0%BE%D1%82%D1%8B). [Ρ-метод Полларда дискретного логарифмирования](http://ru.wikipedia.org/wiki/%CE%A1-%D0%BC%D0%B5%D1%82%D0%BE%D0%B4_%D0%9F%D0%BE%D0%BB%D0%BB%D0%B0%D1%80%D0%B4%D0%B0_%D0%B4%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BB%D0%BE%D0%B3%D0%B0%D1%80%D0%B8%D1%84%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F).[Алгоритм COS](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_COS" \o "Алгоритм COS). [Алгоритм Адлемана](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%90%D0%B4%D0%BB%D0%B5%D0%BC%D0%B0%D0%BD%D0%B0).[Алгоритм быстрого возведения в степень](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%B1%D1%8B%D1%81%D1%82%D1%80%D0%BE%D0%B3%D0%BE_%D0%B2%D0%BE%D0%B7%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2_%D1%81%D1%82%D0%B5%D0%BF%D0%B5%D0%BD%D1%8C" \o "Алгоритм быстрого возведения в степень).[Алгоритм Гаусса вычисления даты Пасхи](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%93%D0%B0%D1%83%D1%81%D1%81%D0%B0_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B4%D0%B0%D1%82%D1%8B_%D0%9F%D0%B0%D1%81%D1%85%D0%B8" \o "Алгоритм Гаусса вычисления даты Пасхи).[Алгоритм Евклида](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%95%D0%B2%D0%BA%D0%BB%D0%B8%D0%B4%D0%B0" \o "Алгоритм Евклида).[Алгоритм Монтгомери](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%9C%D0%BE%D0%BD%D1%82%D0%B3%D0%BE%D0%BC%D0%B5%D1%80%D0%B8" \o "Алгоритм Монтгомери).[Алгоритм Полига — Хеллмана](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%9F%D0%BE%D0%BB%D0%B8%D0%B3%D0%B0_%E2%80%94_%D0%A5%D0%B5%D0%BB%D0%BB%D0%BC%D0%B0%D0%BD%D0%B0" \o "Алгоритм Полига — Хеллмана).[Алгоритм Фюрера](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%A4%D1%8E%D1%80%D0%B5%D1%80%D0%B0" \o "Алгоритм Фюрера).[Алгоритм Шенкса](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%A8%D0%B5%D0%BD%D0%BA%D1%81%D0%B0" \o "Алгоритм Шенкса).[Бинарный алгоритм вычисления НОД](http://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%BD%D0%B0%D1%80%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%9D%D0%9E%D0%94" \o "Бинарный алгоритм вычисления НОД).[Метод квадратичных форм Шенкса](http://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%BA%D0%B2%D0%B0%D0%B4%D1%80%D0%B0%D1%82%D0%B8%D1%87%D0%BD%D1%8B%D1%85_%D1%84%D0%BE%D1%80%D0%BC_%D0%A8%D0%B5%D0%BD%D0%BA%D1%81%D0%B0" \o "Метод квадратичных форм Шенкса).[Односторонняя функция с потайным входом](http://ru.wikipedia.org/wiki/%D0%9E%D0%B4%D0%BD%D0%BE%D1%81%D1%82%D0%BE%D1%80%D0%BE%D0%BD%D0%BD%D1%8F%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_%D1%81_%D0%BF%D0%BE%D1%82%D0%B0%D0%B9%D0%BD%D1%8B%D0%BC_%D0%B2%D1%85%D0%BE%D0%B4%D0%BE%D0%BC" \o "Односторонняя функция с потайным входом).[Признаки делимости](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%B7%D0%BD%D0%B0%D0%BA%D0%B8_%D0%B4%D0%B5%D0%BB%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D0%B8" \o "Признаки делимости).[Решето Аткина](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%88%D0%B5%D1%82%D0%BE_%D0%90%D1%82%D0%BA%D0%B8%D0%BD%D0%B0" \o "Решето Аткина).[Решето Сундарама](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%88%D0%B5%D1%82%D0%BE_%D0%A1%D1%83%D0%BD%D0%B4%D0%B0%D1%80%D0%B0%D0%BC%D0%B0" \o "Решето Сундарама).[Решето Эратосфена](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%88%D0%B5%D1%82%D0%BE_%D0%AD%D1%80%D0%B0%D1%82%D0%BE%D1%81%D1%84%D0%B5%D0%BD%D0%B0" \o "Решето Эратосфена).[Тест Миллера (теория чисел)](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82_%D0%9C%D0%B8%D0%BB%D0%BB%D0%B5%D1%80%D0%B0_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D1%87%D0%B8%D1%81%D0%B5%D0%BB)" \o "Тест Миллера (теория чисел)).[Тест простоты](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82_%D0%BF%D1%80%D0%BE%D1%81%D1%82%D0%BE%D1%82%D1%8B" \o "Тест простоты) [Люка](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82_%D0%BF%D1%80%D0%BE%D1%81%D1%82%D0%BE%D1%82%D1%8B_%D0%9B%D1%8E%D0%BA%D0%B0).[Тест Ферма](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82_%D0%A4%D0%B5%D1%80%D0%BC%D0%B0" \o "Тест Ферма)

**Арифметические алгоритмы:**  [Решето Эратосфена](http://informatics.mccme.ru/moodle/mod/resource/view.php?id=825).  [Функция Эйлера](http://informatics.mccme.ru/moodle/mod/resource/view.php?id=828).  [Бинарное возведение в степень за О(log N)](http://informatics.mccme.ru/moodle/mod/resource/view.php?id=858).  [Обратный элемент в поле по модулю](http://informatics.mccme.ru/moodle/mod/resource/view.php?id=862).  [Дискретное логарифмирование](http://informatics.mccme.ru/moodle/mod/resource/view.php?id=865).  [Линейное модулярное уравнения с одним неизвестным](http://informatics.mccme.ru/moodle/mod/resource/view.php?id=867).  [Китайская теорема об остатках. Алгоритм Гарнера](http://informatics.mccme.ru/moodle/mod/resource/view.php?id=868).  [Нахождение степени делителя факториала](http://informatics.mccme.ru/moodle/mod/resource/view.php?id=869).  [Вычисления факториала по модулю](http://informatics.mccme.ru/moodle/mod/resource/view.php?id=871).  [Первообразный корень.](http://informatics.mccme.ru/moodle/mod/resource/view.php?id=874)   [Дискретное извлечение корня](http://informatics.mccme.ru/moodle/mod/resource/view.php?id=875)

**Алгоритмы для работы с системами счисления:**  [Целые числа](http://informatics.mccme.ru/moodle/mod/statements/view.php?id=595). [Дроби](http://informatics.mccme.ru/moodle/mod/statements/view.php?id=594):  [Троичная сбалансированная система счисления](http://informatics.mccme.ru/moodle/mod/resource/view.php?id=870).

**Алгоритмы для работы с битовыми операциями:**   [Перебор всех подмасок данной маски](http://informatics.mccme.ru/moodle/mod/resource/view.php?id=872)

**Алгоритмы для работы с "длинной" арифметикой**  [( "длинные" числа)](http://informatics.mccme.ru/moodle/mod/resource/view.php?id=1725)

**Более сложные арифметические алгоритмы:**   [алгоритм Карацубы и быстрое преобразование Фурье (](http://informatics.mccme.ru/moodle/mod/book/view.php?id=430) [Умножение двух полиномов или длинных чисел с помощью Быстрого преобразования Фурье за O (N log N)](http://informatics.mccme.ru/moodle/mod/resource/view.php?id=878). [Реализация БПФ в поле вычетов по модулю p](http://informatics.mccme.ru/moodle/mod/book/view.php?id=464). [Тест BPSW на простоту чисел за O (log N)](http://informatics.mccme.ru/moodle/mod/resource/view.php?id=876). [Эффективные алгоритмы факторизации: Полларда p-1, Полларда p, Бента, Полларда Монте-Карло, Ферма](http://informatics.mccme.ru/moodle/mod/resource/view.php?id=877).

**ЭКВИВАЛЕНТНОСТЬ МАТЕМАТИЧЕСКИХ МОДЕЛЕЙ ПОНЯТИЯ «АЛГОРИТМ», ТЕЗИСЫ ТЕОРИИ АЛГОРИТМОВ. ( подробно см. литературу)**

***Совпадение класса правильно-вычислимых функций по Тьюрингу и класса частично рекурсивных функций***

**Лемма 1** [Клини,1957,с.330]**.**  Все простейшие функции правильно-вычислимы по Тью­рингу.

**Лемма 2** [Клини,1957,с.331; Эббинхауз,Якобс,Ман,Хермес,1972,с.67]**.**  Операция подста­новки сохраняет правильную вычислимость по Тьюрингу.

**Лемма 3** [Клини,1957,с.332]**.**  Операция минимизации сохраняет правильную вычисли­мость по Тьюрингу.

**Лемма 4** [Клини,1957,с.331]**.**  Операция примитивной рекурсии сохраняет правильную вычислимость по Тьюрингу.

**Предложение 1** [Матросов,1989,с.63-64]**.**  Всякая частично рекурсивная функция является правильно-вычислимой по Тьюрингу.

**Предложение 2** [Клини,1957; Матросов,1989,с.64]**.**  Всякая правильно-вычислимая фун­кция по Тьюрингу является частично рекурсивной функцией.

***Совпадение класса нормально-вычислимых функций и класса функций, вычислимых по Тьюрингу.*** Утверждение, состоящее в том, что класс нормально-вычислимых функций сов­падает с классом функций, вычислимых по Тьюрингу, доказано в[Мендельсон,1984,с.256-261].

***Совпадение класса нормально-вычислимых функций и класса частично-рекурсивных функций.*** Утверждение, состоящее в том, что класс нормально-вычислимых функций совпада­ет с классом частично-рекурсивных функций, доказано в [Мендельсон,1984,с.242-248].

***Совпадение класса частично рекурсивных функций и класса функций, вычислимых по Эрбрану-Гёделю***

**Предложение**.

**(1)** [Мендельсон,1984,с.263]Всякая частично рекурсивная функция вычислима по Эрбра­ну-Гёделю.

**(2)** (С.Клини,1936) Всякая функция, вычислимая по Эрбрану-Гёделю, является частично рекурсивной функцией.

Таким образом, способы уточнения понятия "*интуитивная вычислимость*" (в терминах машин Тьюринга, нормальных алгорифмов***,*** рекурсивных функций и функций,вычислимых по Эрбрану-Гёделю) по существу эквивалентны. Совпадение всех упомянутых классов интуитив­но воспринимается как ещё один факт, говорящий в пользу различных тезисов теории алгорит­мов.

**ТЕЗИСЫ ТЕОРИИ АЛГОРИТМОВ**

В математике объект *тезис* это не *теорема*, ибо тезис не имеет доказательства; это не *ги­потеза*, ибо тезис и не может быть доказан; тезис - это не *аксиома*, которую мы вольны прини­мать или не принимать. Все это так из-за то­го, что тезис не является точным математическим утверждением.

**1.** ***Тезис Чёрча-Клини.*** *Тезис Чёрча-Клини* - это *естественнонаучный факт*, подтвер­ждаемый опытом, накопленным в математике за всю её историю. Иногда (см. [Успенский,1960, с.157]) тезис Чёрча-Клини называют *основной гипотезой теории вычислимых функций*. Он не может быть доказан средствами математики, т.к. он утверждает тождественность понятия "*ин­туитивная вычислимость*", не являющегося математическим, с математическим понятием "*частичная рекурсивность*".

***Тезис Чёрча-Клини*** (1936)**. (1)** Каждая интуитивно вычислимая функция является час­тично рекурсивной функцией, т.е. KВФÌKЧРФ.**(2)** (По [Успенский,1960,с.157-158]) Понятие "*час­тично рекурсивная функция*" является точным математическим аналогом понятия "*интуитивно вычислимой функции*". Только после формулирования (и *принятия*!) тезиса Чёрча-Клини при­обретают точный смысл и могут быть доказаны утверждения такого типа, как "*не существует вычислимой функции такой, что...*", "*для всякой вычислимой функции*..." и т.п.

**2. *Принцип нормализации Маркова*** (*тезис Маркова*).

**(1) Всякий алгоритм в алфавите å эквивалентен относительно å некоторому норма­льному алгорифму над å.**

**(2) (По [Игошин,1991,с.238-239]) Алгоритм для нахождения значений словарной функции существует тогда и только тогда, когда функция нормально-вычислима по Мар­кову.**

Этот принцип носит внематематический характер, т.к. он выдвинут на основании матема­тического и практического опыта человечества. Поэтому его нельзя доказать, а можно лишь *приводить доводы в его пользу*:

(1) главным доводом является тот факт, что все до сих пор рассмотренные с этой точки зрения алгоритмы оказались нормализуемыми, т.е. эквивалентными нормальным алгоритмам;

(2) все построенные до сих пор уточнения понятия "*алгоритм*" привели к типам алгорит­мов, для которых удалось доказать нормализуемость;

(3) различные способы комбинирования алгорифмов (способы построения новых алгорит­мов из данных) ведут от нормализуемых алгорифмов к нормализуемым.

После принятия принципа нормализации Маркова *доказательство невычислимости неко­торой функции*, т.е. несуществования алгоритма, вычисляющего её значения, сводится к дока­зательству невычислимости по Маркову этой функции, т.е. к доказательству факта несущество­вания нормального алгорифма Маркова.

**3.** ***Тезис Тьюринга***

**(1)** [Колмогоров,Драгалин,1984,с.64] Всякий словарный, дискретный, массовый, детерми­нированный, замкнутый алгоритм может быть задан в виде машины Тьюринга.

**(2)** (По [Гладкий,1998,с.297]) Всякая интуитивно вычислимая функция является вычисли­мой по Тьюрингу (*тезис Чёрча-Тьюринга*).

Рассмотрим два вопроса, возникающих в связи с формулировкой тезиса Тьюринга.

(а) *Каково значение этого тезиса для теории алгоритмов*?

Тезис Тьюринга уточняет понятие "*алгоритм*" посредством специального, но точного ма­тематического понятия "*машина Тьюринга*". Этот тезис утверждает, что всякий раз, когда кон­кретное предписание признано алгоритмом (в интуитивном смысле), оказывается возможным его реализовать в виде машины Тьюринга.

Таким образом, теория алгоритмов объявляет в качестве объекта своего исследования ма­шины Тьюринга. Становятся осмысленными постановки таких вопросов, как вопрос о возмож­ности или невозможности разрешающего алгоритма для задачи того или другого типа. Именно теперь это следует понимать как вопрос о существовании машины Тьюринга, обладающей нуж­ным свойством. Например, пусть удалось доказать, что некоторая конкретная функция f не вы­числима по Тьюрингу. Тезис Тьюринга указывает, что бесполезно искать алгоритм (в инту­итивном смысле слова), вычисляющий функцию f, ибо функция f и интуитивно невычислима. Таким образом, открывается возможность математическими средствами обнаружить невычис­лимость функций.

(б) *В чем заключается обоснование этого тезиса*?

Конечно, не может быть и речи о том, чтобы доказать этот тезис подобно тому, как обыч­но доказываются в математике теоремы. Действительно, формулировка тезиса не имеет харак­тера теоремы, поскольку она представляет собою утверждение о понятии "*алгоритм*", которое не является точным математическим понятием, а следовательно, не может быть объектом стро­гих математических рассуждений.

Уверенность в справедливости тезиса основана главным образом на опыте. Все известные алгоритмы, которые были придуманы в течении многих тысячелетий истории математики, мо­гут быть заданы посредством машины Тьюринга. Содержание тезиса имеет и вполне отчетли­вый характер прогноза на будущие времена: всякий раз, когда в будущем какое-либо предписа­ние будет признано алгоритмом, то независимо от того, в какой форме и какими средствами это предписание будет первоначально выражено, его можно будет задать также в виде функциона­льной схемы машины Тьюринга. В этом смысле тезис можно сравнить с физическим законом (например, с законом сохранения энергии). Именно огромный фактический опыт в прошлом признается достаточным основанием для подобных прогнозов.

Более того, когда в науке возникла острая потребность в выработке точного понятия "*ал­горитм*"***,*** многие математики предприняли исследования с целью выявить некоторую стандарт­ную форму задания алгоритмов, достаточно точную для того, чтобы она могла стать объектом математических исследований и достаточно общую для того, чтобы всем мыслимым алгорит­мам можно было придавать такую форму.

Были предложены следующие способы уточнения этого понятия [Катленд,1983,с.56-57]:

(1) l-*определимые функции* (А.Чёрч, С.К.Клини, 1933-1936);

(2) *общерекурсивные функции*, определённые с помощью исчисления рекурсивных уравнений (Ж.Эрбран, К.Гёдель, С.К.Клини, 1934-1936);

(3) m-*рекурсивные функции* и *частично* *рекурсивные функции* (К.Гёдель, С.К.Клини, 1936);

(4) *машины Тьюринга* (А.Тьюринг, Э.Пост, 1936);

(5) *функции, определяемые каноническими системами Поста* (Э.Пост, 1943);

(6) *нормальные алгоритмы* *Маркова* (А.А.Марков, 1950). (7) *МНР- вычислимые функции* (Дж.Шепердсон, Х.Стерджис, 1963).

Между этими подходами имеются большие различия; каждый из них имеет свои преиму­щества для соответствующего описания вычислимости.

Следующий замечательный результат получен усилиями многих исследователей.

**Предложение** (*основное*) [Катленд,1983,с.57].Каждое из вышеупомянутых уточнений по­нятия "*вычислимость*" приводит к одному и тому же классу вычислимых функций.

Этот факт нельзя считать случайным; он является дополнительным доводом в пользу те­зиса Тьюринга.

Таким образом, перечисленные доводы позволяют принять тезис Тьюринга в качестве естественно-научной гипотезы, утверждающей, что неформальное, интуитивное понятие "*алго­ритм*" и математическое понятие "*алгорифм в виде машины Тьюринга*" равно-объёмны.

Таким образом, оказывается, что принятие тезиса Тьюринга равносильно принятию *тези­са Чёрча-Клини* (для частично рекурсивных функций) или *принципа нормализации* (для нормальных алгорифмов).

Любой из упомянутых выше тезисов является основанием для вывода о неразрешимости данной алгоритмической проблемы после того, как строго доказано, что эта проблема не может быть решена в рамках того или иного уточнения понятия "*алгоритм*".

Заметим, что в отличие от других тезисов тезис Тьюринга выглядит более "убедитель­ным". Действительно, производя вычисле­ния согласно избранному плану, человек-математик работает сходным с машиной Тьюринга образом: рассматривая какое-либо место в своих запи­сях и находясь в определённом "умонастроении", он делает необходимые изменения в написан­ном, проникается новым "умонастроением" и переходит к рассмотрению дальнейших записей (то обстоятельство, что при этом он совершает более сложные, чем машина Тьюринга, шаги, представляется не слишком принципиальным).

**ВЫВОДЫ:**

Существует два самых больших класса алгоритмов — это алгоритмы с повторением и рекурсивные алгоритмы. С алгоритмами связаны приведенные ниже области исследований: Анализ алгоритмов, теория алгоритмов, построение алгоритмов. Теория алгоритмов оказала существенное влияние на развитие ЭВМ и практику программирования.. На основе формализации понятия алгоритма возможно сравнение алгоритмов по их эффективности, проверка их эквивалентности, определение областей применимости. Модели вычислений: Машина Тьюринга, Лямбда-исчисление, Комбинаторная логика.