УДК 004.4:006.92 EDN YUYXIQ

Проблема репрезентации времени в языках программирования: философско-методологические аспекты

© С.А. Соловьев

ООО «ТЦР», Москва, 127287, Россия

Проведен междисциплинарный анализ эволюции методов структурирования темпоральности человеком. Показано влияние исторических событий на инженерную практику конструирования языков программирования. Выявлены трудности построения формализуемой и алгоритмизируемой модели времени, коренящиеся в различных этапах развития инструментария его измерения. Проанализированы ограничения в устройстве временных моделей, вызванные влиянием исторического развития календарных систем, шкал времени и концепции часовых зон. Учтены проблемы, связанные с календарными реформами, нерегулярностью астрономических циклов, постоянным уточнением временных шкал и изменением метрологических оснований, подвижностью границ и корректировкой правил административных часовых зон. Показано, что социокультурные факторы — региональные традиции, социальная инерция — вносят значимый вклад в проблему репрезентации времени в языках программирования наравне с физическими факторами. Рассмотрены фундаментальные методологические ограничения существующих инженерных решений, таких как Unix time и Time Zone Database. Продемонстрировано, как компромиссы и допущения в моделях времени порождают ошибки при высокоточной синхронизации и долговременном хранении данных. Новизна исследования заключается в системном сопоставлении культурно-исторической трансформации подходов к измерению времени с конкретными архитектурными вызовами при проектировании языков программирования и инженерных стандартов. Сделан вывод о невозможности построения универсальной темпоральной модели в силу фундаментального конфликта между идеальной природой математических моделей времени и непредсказуемостью социальной и физической реальности. Показано, что задача синхронизации шкалы времени с социальными и биологическими циклами на Земле неизбежно приводит к ее нерегулярности и требует постоянной ручной корректировки, исключая возможность полной алгоритмизации. Отмечено, что проблема предела формализации времени в контексте создания языков программирования нуждается в дальнейшем философско-методологическом анализе.

Ключевые слова: *репрезентация времени, временные значения, языки программирования, темпорология, темпоральность, календари, шкалы времени, часовые пояса, часовые зоны*

Время — одна из фундаментальных категорий философии и науки, обладающая одновременно очевидной и парадоксальной природой. На протяжении столетий мыслители пытались постичь его сущность.

Аристотель в «Физике» дает детальный анализ сразу нескольких аспектов проблемы времени, но при этом о его природе философ пишет с определенной долей неуверенности: «Время, скорее всего, представляется каким-то движением и изменением» [1, с. 101, 103]. В период научной революции время обрело очерченные рамки абсолютной сущности, лежащей в основе классической ньютоновой механики. В начале XX в. человечество обогатилось знанием контринтуитивных релятивистских эффектов, вытекающих из теории относительности Эйнштейна. Но, несмотря на теоретические успехи, фундаментальная природа времени по сей день остается нерешенной проблемой.

Задача измерения времени имеет большое практическое значение. Календарная дата и время суток во многом определяют жизненный ритм людей, лежат в основе экономических процессов, юридических отношений, религиозных праздников и т. п. На протяжении веков инструментарий для измерения времени совершенствовался, но, несмотря на значительные успехи, человечество все еще далеко от создания универсальных измерительных процедур. В попытках структурирования темпоральных аспектов своего бытия человечество проходит трудности, коренящиеся в сложной физической и метафизической природе времени, а также в социально-историческом контексте развития науки.

В конце XX в. стало очевидно, что «область человеческой деятельности, связанная с построением компьютерных “баз знаний”, “искусственного интеллекта”, в отличие от многих естественных наук обязательно требует алгоритмизируемой конструкции реального времени» [2, с. 14]. Сегодня компьютерные системы и научное оборудование выдвигают гораздо более строгие требования к структуре операций со временем на новом уровне точности и глобальности [3, с. 294].

Целью настоящей работы является философско-методологический анализ влияния исторической эволюции систем измерения времени на проблему репрезентации временны́х значений в языках программирования. Исследование направлено на выявление фундаментальных ограничений, препятствующих созданию универсальной и алгоритмизируемой системы работы со временем. Результаты анализа могут быть полезны при изучении и конструировании языков программирования, создании библиотек исходного кода для обработки времени. Широкий круг разработчиков может применить материалы статьи для снижения числа ошибок в программном коде за счет построения более точной ментальной модели предметной области.

Актуальность работы обусловлена нарастающей компьютеризацией всех сфер жизни, распространенными заблуждениями инженеров-практиков о времени и подходах к его измерению, увеличением цены ошибки при написании программ. Сбои цифровых сервисов

могут привести к остановке экономической деятельности миллионов человек, а ошибка в системе управления АЭС — к ядерной катастрофе. Примечательно также, что в настоящий момент ведется разработка Temporal — нового стандарта обработки дат и времени в ECMAScript, который претендует на звание наиболее полного и точного среди всех языков программирования в области работы с календарными системами и часовыми зонами. Несмотря на активность рабочей группы, создание стандарта продолжается семь лет, что свидетельствует о высокой сложности предметной области и подчеркивает актуальность настоящей статьи.

Новизна исследования заключается в рассмотрении проблемы репрезентации времени в языках программирования с междисциплинарной позиции, сочетающей элементы философской рефлексии и опыт инженерной практики. В работе делается акцент не столько на описании исторического процесса, сколько на методологическом анализе и выявлении фундаментальных ограничений алгоритмизации времени. Рассмотрено влияние культурных и социальных факторов на эволюцию систем измерения и современные вычислительные модели времени. Работа вписывается в контекст темпорологии как формирующейся области знания, исследующей фундаментальную природу времени и его представления в различных дисциплинах.

Календарные системы. Для того чтобы понять историко-философский контекст проблемы репрезентации временны́х значений в языках программирования, следует проанализировать эволюцию методов измерения времени. Заметим, что под измерением времени могут подразумеваться разные процедуры, в зависимости от того, какая используется шкала. Для измерения продолжительности временны́х интервалов (например, с помощью секундомера или таймера) точка отсчета может выбираться произвольно. Также существуют универсальные шкалы, для которых характерно именование уникальными или регулярными именами промежутков (или моментов) времени начиная с общепринятой фиксированной точки отсчета в прошлом: например, к таким системам относятся календари или время суток. В настоящей работе в основном будет рассмотрен второй подход.

Начать анализ следует с одного из древнейших методов структурирования темпоральности — *календаря*. Сегодня под календарем подразумевается система счисления больших (в масштабе человеческой жизни) промежутков времени, основанная на периодичности движения небесных тел. Первые календари появились в связи с необходимостью предсказания регулярных событий: разливов рек, сезонов дождей в целях ведения сельского хозяйства. Не менее важной задачей для древнего человека было определение начала религиозных праздников.

С развитием денежных отношений у календаря появилась еще одна важная функция: в Древнем Риме должники платили проценты в дни календ, первых чисел месяца, откуда и происходит современное слово «календарь».

Григорианский календарь, принятый в настоящее время в России и большинстве других стран, восходит к древнеримскому. Из-за ограниченности сведений, дошедших до наших дней, ученые спорят о том, как именно происходила эволюция римского календаря. Согласно преданиям, первый календарь, состоявший из 10 лунных месяцев, начиная с марта, римлянам дал первый правитель и основатель города — Ромул [4, с. 293]. Его преемник Нума Помпилий провел реформу, увеличив количество месяцев и приведя календарь в соответствие с солнечным годом. При этом календарная система оставалась сложной и зависела от человеческого фактора: вставку 13-го месяца мерцедония определял верховный понтифик, который мог использовать свои полномочия в целях политического давления. По некоторым свидетельствам, расхождение календаря с наблюдаемыми временами года могло достигать 4 месяцев.

Для того чтобы решить накопившиеся проблемы, Гай Юлий Цезарь в 45 г. до н. э. провел реформу календаря, сделав его арифметическим, т.е. основанным на вычислительных правилах, а не результатах актуальных астрономических наблюдений. Реформа вводила строгие принципы определения продолжительности года, составляющей в среднем 365,25 солнечных суток: каждый четвертый год длился ровно на один день дольше остальных. При этом стоит отметить, что в Древнем Риме вместо привычной сегодня системы сквозной нумерации лет (например, от сотворения мира или Рождества Христова) годы обычно назывались по именам избранных консулов. Во многих других культурах древности также была распространенной традиция отсчитывать годы заново от начала правления каждого следующего монарха. Еще до появления компьютеров и необходимости алгоритмической обработки такая нумерация затрудняла ретроспективный анализ и ограничивала возможность выражения дат в будущем.

В первом тысячелетии в Европе широко распространилась практика отсчета лет от Рождества Христова. Годы до этого события стали нумероваться в обратном порядке. Такая система лучше поддается алгоритмизации и компьютерному моделированию, но имеет свои проблемы. Отсутствие нулевого года приводит к тому, что последовательность названий лет не получается удобно проецировать на шкалу целых чисел, что затрудняет арифметические вычисления при написании программ. Также возникает сложность с группировкой лет по векам. Века нашей эры заканчиваются годом, кратным 100, а века до нашей эры таким годом начинаются. В зависимости от положения

века (наша эра или до нее) выражения «первая половина» или «третья четверть» трактуются по-разному с точки зрения числовой характеристики границ выделяемого ими диапазона.

Реформы Цезаря в I в. до н. э. упростили структуру римского календаря, но не избавили его от проблем полностью. Правило вставки дополнительного дня недостаточно точно приближало реальную продолжительность тропического года, из-за чего за полтора тысячелетия применения календаря накопилось смещение относительно естественных сезонов (весеннего равноденствия) на 10 суток. Это приводило к проблемам с определением дат религиозных праздников, в частности Пасхи, поэтому назревала необходимость в новой реформе. В 1582 г. Папа Григорий XIII ввел новый календарь, получивший впоследствии название «григорианский». Он был основан на юлианском и предусматривал усовершенствованное правило вставки дополнительных дней: годы, кратные 100, за исключением делящихся на 400, объявлялись невисокосными. Это позволило уменьшить среднюю продолжительность года до 365,2425 суток, что более точно соответствует тропическому году и приводит к ошибке на 1 день лишь за 3300 лет. Для компенсации уже накопившейся разницы были пропущены 10 календарных дней, и после четверга 4 октября наступила пятница 15 октября. При этом деление года на 12 месяцев различной нерегулярной продолжительности, унаследованное от юлианского календаря, осталось неизменным. Такое деление усложняет алгоритмы обработки временны́х значений, исключая возможность определения номера месяца по дню года путем деления по модулю.

Несмотря на имеющиеся проблемы, григорианский календарь стал международным стандартом и широко используется сегодня как в повседневном общении, так и в компьютерных языках. Тем не менее, его распространение и признание в разных странах заняло несколько столетий. Примечателен пример Швеции, в которой было принято решение постепенного перехода с юлианского на григорианский календарь за счет отмены високосных дней с 1700 по 1740 г. Но вскоре после начала реформы по ошибке снова стало выполняться правило юлианского календаря, поэтому Швеция какое-то время жила по собственному шведскому календарю, но в 1712 г. ввиду непрактичности такого подхода вернулась к старому стилю за счет добавления в феврале двух дней. Таким образом, в Швеции было 30 февраля 1712 г. Подобные нестандартные ситуации встречаются и в других странах, приводя к трудностям при автоматической обработке исторических документов XVI–XX вв. Для корректной интерпретации письменной даты необходимо учитывать метаинформацию: место происхождения документа, язык текста и даже религиозную идентичность автора. При моделировании дат в языках программирования для простоты

и однозначности часто применяют пролептический григорианский календарь, который предполагает распространение правил григорианского календаря как в будущее, так и в прошлое — на время, когда он еще не был создан. Такая система проще в реализации, но имеет ограниченную практическую ценность при моделировании исторических источников и событий, поскольку в этих предметных областях обычно используются оригинальные даты по юлианскому или особому местному календарю.

Несмотря на доминирование григорианского календаря, в мире продолжают использоваться другие системы, в том числе не основанные на юлианской. Например, в Иране официальным календарем является солнечная хиджра, а в Израиле наравне с григорианским применяется еврейский лунно-солнечный календарь. В религиозных и культурных целях (как правило, для определения праздников) по всему миру используются десятки различных календарей. Все это приводит к экономическим трудностям, правовым коллизиям, а также значительно усложняет компьютерное моделирование в предметных областях, связанных с обработкой дат. Принципиальные трудности вызывает неалгоритмический характер многих традиционных систем летоисчисления: в мусульманском календаре смена суток происходит в момент захода солнца, а не наступления полуночи, как в григорианской системе. Началом месяца считается день, когда серп Луны впервые становится видимым в вечерних сумерках после новолуния. Хотя вычисление даты новолуния возможно с высокой точностью, предсказать реальную видимость полумесяца сложнее, так как она зависит от погодных условий, характеристик атмосферы и местоположения наблюдателя. Отсутствие единых правил смены месяцев, не привязанных к астрономическим наблюдениям, приводит к тому, что указание дня, месяца и года в мусульманском календаре не всегда однозначно идентифицирует конкретный день, поэтому для устранения неопределенности между разными региональными традициями дополнительно может указываться день недели.

Историческая эволюция календарных систем породила значительные трудности в задаче алгоритмизации и репрезентации дат в языках программирования. Наличие лунных, солнечных и лунно-солнечных календарей, различия в правилах смены суток, месяцев и лет, зависимость от фактических астрономических наблюдений создают фундаментальные сложности при автоматической обработке временны́х значений. Постепенный переход с юлианского календаря на григорианский, локальные реформы, а также использование пролептических систем в вычислительных моделях приводят к неоднозначности в интерпретации дат в исторических документах. Арифметику дат усложняет отсутствие нулевого года, различная продолжительность месяцев

и лет, сквозная нумерация дней недели, пересекающая границы лет. При этом особенности календарных систем нельзя сводить к задаче форматированного ввода/вывода, поскольку они могут быть частью предметной области.

Время суток. Отслеживание времени суток, наряду с календарными системами, является одним из древнейших методов упорядочения человеческой деятельности. Исторически системы отсчета времени суток создавались путем введения дольных единиц для солнечных суток. Корни современной системы с делением суток на 24 часа по 60 минут по 60 секунд уходят в шумерскую математическую традицию, где применялась шестидесятеричная система счисления. Используемые сегодня названия происходят от выражений на латыни: минута — от лат. *pars minuta prima*, а секунда — от лат. pars minuta secunda. Также существует редкоупотребимая 60-я доля секунды, называющаяся «терция», от лат. *pars minuta tertia*. Сегодня распространение получила десятичная система счисления, в компьютерных науках также популярны системы с основанием степени двойки. Различный вес разрядов в записи времени — деление с коэффициентами 24 и 60 — выбивается из привычной системы и приводит к трудностям при проектировании алгоритмов. Во время Великой французской революции декретом конвента от 5 октября 1793 г. была установлена система деления суток на 10 часов по 100 минут по 100 секунд для удобства счета и научных исследований, но такая система не прижилась в силу социальной инерционности и была отменена в 1795 г. Тем не менее, недостатки существующей системы привели к тому, что для удобства вычислений в астрономии время суток нередко может обозначаться в долях суток, выраженных десятичными дробями.

Солнечные часы долгое время оставались основным инструментом измерения времени. Они помогали определить момент суток по фактическому положению Солнца на небе. Длительность солнечных суток (период между двумя последовательными верхними кульминациями Солнца) не является инвариантом и меняется в течение года. Таким образом, продолжительность часа и других дольных единиц при подобном определении варьировалась в зависимости от сезона, что затрудняло создание единых стандартов измерения. Чтобы устранить эту вариативность, с распространением механических часов в качестве основы измерения времени стало использоваться понятие средних солнечных суток. Такие сутки задаются усредненной продолжительностью солнечных суток в течение года. Зависимость отклонения среднего солнечного времени от истинного солнечного в течение года фиксирует так называемое уравнение времени.

Однако этот подход тоже оказался несовершенным. Со времен Джона Флемстида (1646–1719) считалось, что дневное вращение Земли равномерно. Но в конце XIX — начале XX в. с увеличением

точности астрономических наблюдений было обнаружено, что период вращения Земли вокруг своей оси (т. е. длина звездных суток) испытывает колебания на коротких временны́х интервалах, а в целом медленно увеличивается (из-за приливного торможения). Доказательства этого факта были суммированы де Ситтером в 1927 г. Поэтому в 1952 г. в целях удобства научных измерений была предложена шкала эфемеридного времени (ET), в которой продолжительность секунды была привязана к длительности 1900-го тропического года. Примечательно, что его продолжительность была вычислена математически, а не измерена эмпирически. Это определение секунды, задающее равномерную шкалу, где длительность секунды является инвариантом, вошло в СИ с ее появлением в 1960 г. Неизбежным недостатком равномерной эфемеридной шкалы стало ее постепенное расхождение с фактически наблюдаемым неравномерным средним солнечным временем, что делало ее неудобной для бытовых нужд. Еще один недостаток заключался в технической сложности измерения времени в соответствии с такой шкалой, поскольку определение секунды было привязано не к процедуре измерения, а к абстрактному эталону.

Решением части проблем стало развитие технологии атомных часов и появление шкалы международного атомного времени (TAI). Масштаб системы TAI принят равным масштабу эфемеридного времени (ET), т. е. атомные часы физически воспроизводят шкалу ET. При этом нуль-пункт был сдвинут на 32,184 с, чтобы компенсировать разницу со средним солнечным временем, накопившуюся с 1900 г. В 1967 г. определение секунды, основанное на неизменных физических свойствах атомов, стало стандартом СИ. Несмотря на преимущества TAI, примечательно, что в отличие от ET она неудобна для астрономических измерений, поскольку по определению зависит от фактических наблюдений за ходом часов на Земле, подверженных сезонным гравитационным возмущениям в силу эллиптичности земной орбиты и влияния других небесных тел. Для атомного времени, как и для ET, осталась актуальной проблема расхождения времени суток с естественными суточными циклами Земли и фазой Солнца на небе.

Для того чтобы сохранить преимущества однородной шкалы международного атомного времени, но при этом решить проблему синхронизации с истинным солнечным временем, в 1972 г. был принят стандарт всемирного координированного времени (UTC). Идея состояла в том, чтобы сохранить секунду постоянной продолжительности на основе фундаментальных свойств атомов, но сделать переменной продолжительность суток. При расхождении международного атомного времени (TAI) с уточненной шкалой среднего солнечного времени по Гринвичу (UT1) на 0,9 с Международная служба вращения Земли принимала бы решение о введении дополнительной

секунды в сутках для компенсации разницы. Таким образом, все секунды имели бы продолжительность, равную секунде TAI, но при этом некоторые дни содержали бы на 1 с больше — 86 401 с. Нуль-пункт UTC был смещен относительно TAI на 10 с, чтобы компенсировать разницу абсолютной шкалы и истинного Солнца, накопившуюся с 1952 по 1972 г. Именно UTC лежит в основе измерения времени суток по всему миру. Всего с 1972 по 2025 г. было добавлено 27 дополнительных секунд. Вставка дополнительных секунд ad hoc делает шкалу UTC принципиально неалгоритмизируемой. Таким образом, при конструировании модуля дат языка программирования стоит выбор: либо игнорировать дополнительные секунды, жертвуя точностью вычислений и соответствием гражданскому времени, либо предусматривать базу данных дополнительных секунд как неотъемлемую часть среды выполнения. Некоторые создатели языков идут по другому пути: вообще не включают модуль обработки дат в спецификацию языка, оставляя заботу о его реализации на авторов библиотек программного кода.

Следует отметить, что, несмотря на многовековой тренд замедления суточного вращения Земли, в течение второго десятилетия XXI в. оно ускорялось. Впрочем, UTC предусматривает возможность не только вставки дополнительной секунды, но и пропуска. На практике такого ни разу не случалось, но, согласно UTC, сутки могут длиться 86 399 с.

Практический опыт демонстрирует, что добавление дополнительных секунд сопряжено с возникновением серьезных технических трудностей. Навигационная система ГЛОНАСС, работающая по стандарту UTC, была сутки недоступна из-за программного сбоя, вызванного корректировкой внутреннего времени после добавления дополнительной секунды в 1997 г. Предполагая возможные проблемы, инженеры при проектировании систем спутниковой навигации GPS и BeiDou отказались от учета дополнительных секунд, создав собственные шкалы с масштабом секунд международного атомного времени с фиксированным смещением нуль-пункта. Сложности при проектировании систем, учитывающих дополнительные секунды, привели к предложению полной отмены дополнительных секунд Международным астрономическим союзом в 2009 г. и активной международной дискуссии вокруг этого вопроса. В ноябре 2022 г. на очередной Генеральной конференции по мерам и весам было предложено отказаться от дополнительных секунд к 2035 г., но четко зафиксированные решения пока не были приняты.

Во многом в силу неудобств, связанных с неалгоритмизируемой конструкцией UTC, в индустрии компьютерных технологий большее распространение получила шкала Unix time (время Unix или время POSIX). Отметку времени Unix можно определить как число секунд,

прошедших с 00:00 1 января 1970 г. по UTC без учета дополнительных секунд. Обработка дополнительных секунд UTC зависит от конкретной реализации, но в большинстве случаев сводится к игнорированию: в момент вставки дополнительной секунды время Unix повторяет прошедшую ранее секунду, чтобы сохранять синхронизацию времени суток с UTC, но при этом поддерживать фиксированную продолжительность суток, номинально равную 86 400 с. Это приводит к неоднозначности проекции времени Unix в UTC. Существуют альтернативные реализации, предполагающие инкрементирование счетчика времени и во время вставки дополнительной секунды, что задает шкалу, аналогичную TAI, но со смещенным относительно нее нуль-пунктом.

Время Unix лежит в основе моделей времени во многих популярных языках программирования. Например, стандартный объект Date в JavaScript основан на Unix-метках с точностью до миллисекунд, а будущий стандарт Temporal API — с точностью до наносекунд. Игнорирование дополнительных секунд приводит к неточностям при вычислении продолжительности временны́х интервалов и темпоральной арифметике встроенными средствами языка. Помимо этого опора на время Unix ставит множество нетривиальных вопросов перед проектировщиками языков. Например: как правильно моделировать временны́е значения до 1970 г.? Если использовать отрицательные числа для событий в далеком прошлом, на какие конкретно моменты времени они указывают, учитывая отсутствие данных измерений атомных часов за тот период? Как обрабатывать дополнительные секунды при необходимости получить миллисекундную и более высокую точность: останавливать счетчики времени или делать шаг на секунду назад? Как обрабатывать пропуск секунды, если он когда-либо случится в UTC?

Ключевая методологическая проблема, возникающая в задаче репрезентации понятия «время суток» в языках программирования, сводится к неустранимому принципиальному противоречию между потребностью в создании равномерной и алгоритмизируемой шкалы и непредсказуемостью нерегулярных астрономических циклов, задающих интуитивно воспринимаемое время. Эта проблема вынуждает человечество пользоваться принципиально неалгоритмизируемой шкалой UTC и приводит к неизбежным дилеммам и компромиссам при конструировании языков программирования.

Часовые пояса и зоны. До середины 2-го тысячелетия нашей эры в каждом населенном пункте применялось местное истинное солнечное время. С распространением механических часов, для которых характерен равномерный ход времени, не учитывающий сезонные колебания продолжительности солнечных суток, люди стали переходить на использование местного среднего солнечного времени,

которое все еще отличалось от города к городу. Например, разница курантов на Спасской башне в Москве и Петропавловском соборе в Санкт-Петербурге, по приблизительным оценкам, могла составлять 29 мин 5 с.

С развитием скоростного железнодорожного транспорта (в первую очередь в Британии) и появлением расписаний стало понятно, что всюду различное местное время — это проблема. В таких условиях было сложно согласовывать движение составов, увеличивался риск отклонений от расписания и аварий. В середине XIX в. все железные дороги Британии перешли на единое время по Гринвичу, а затем оно было принято парламентом как официальное на территории всей Великобритании.

Канадский железнодорожный инженер Сэндфорд Флеминг обобщил эту идею. Он предложил поделить Землю на 24 часовых пояса по 15° с нулевым поясом, центр которого проходит через Гринвич [5, с. 88]. Разница между соседними поясами должна была составлять ровно 1 ч. Такая схема гарантировала бы отклонение местного поясного времени от среднего солнечного местного времени не более чем на 30 мин. Эта идея была частично реализована, но стройной и предсказуемой системы не получилось. Фактически время для гражданских нужд на Земле определяется согласно административным часовым поясам (именуемым также часовыми зонами), границы которых заданы административным делением или природными объектами. При этом смещение между соседними зонами может выражаться нецелым числом часов. Например, Непал живет по времени UTC+5:45, а Индия — по UTC+5:30. Отклонение от истинного местного солнечного времени может достигать 2 ч и более. Так, на территории всего Китая действует единая часовая зона, хотя по схеме Флеминга он лежит сразу в пяти часовых поясах. Еще одним интересным отхождением от изначальной задумки являются территории, живущие по времени, отклоняющемуся от UTC по модулю более чем на 12 ч. Например, Тонга (UTC+13) и острова Лайн (UTC+14), которые предпочли этот вариант из-за тесных экономических связей с Австралией. Это приводит к тому, что в один момент в разных регионах Земли могут быть три различные календарные даты.

Отдельную трудность для компьютерной формализации работы со временем представляет практика перехода на летнее время. Идея такого перехода не нова и восходит к периоду Нового времени, в частности, к сатирической заметке Бенджамина Франклина о необходимости внедрения налога на закрытые ставни. Суть идеи состоит в том, чтобы в летний период пробуждать людей раньше обычного, чтобы более эффективно использовать солнечный свет, за счет чего экономить электроэнергию и топливо на искусственное освещение.

Это отражено в англоязычном названии данной практики — Daylight Saving Time (DST). Обычно это достигается за счет перевода часов на 1 ч вперед относительно стандартного времени, действующего на данной территории в зимний период. При этом примечательно, что на территориях южнее экватора сезоны инвертированы, поэтому летнее время действует в те месяцы, которые в Северном полушарии считаются зимними. Широкое распространение такой подход получил в первой половине XX в. Одним из принципиальных затруднений является отсутствие общемирового стандарта, регулирующего даты и порядок перевода часов. В разных государствах летнее время внедрялось с учетом локальных экономических, политических и социальных факторов, и каждое государство (а иногда отдельные регионы внутри государства) самостоятельно принимало решение о необходимости и сроках перехода на летнее время. Например, в части штатов США переводят стрелки на час вперед во второе воскресенье марта в 2 часа ночи, а в Чили в первое воскресенье сентября в 12 часов дня.

Еще большую проблему представляет тот факт, что на одной и той же территории наличие летнего времени и правила перехода могли неоднократно меняться даже в течение короткого срока. Например, жителям Египта в 2014 г. пришлось 4 раза переводить стрелки часов за 5 месяцев. При этом мотивация властей на очередной переход далеко не всегда очевидна. Изменения нередко происходят спонтанно, без анонсов, что приводит к невозможности настроить все компьютерные системы на новые правила отсчета времени заранее. Такая непредсказуемость особенно проблематична для авиаперевозок, логистических компаний, транснациональных корпораций, чьи базы данных содержат информацию о расписаниях и договорных сроках. Резкое изменение правил отсчета времени в стране может приводить к искажениям в арифметике дат (например, некоторые события «происходят раньше», чем были запланированы, или вовсе выпадают из хронологической логики системы), что усложняет как техническую, так и правовую сторону вопроса.

Перевод часов на летнее время и обратно порождает ситуации, когда одни календарные сутки могут составлять 23 или 25 ч. Такая система допускает метки времени, несуществующие в определенной часовой зоне. Например, местное время 2006-03-26T02:30:00 (записанное по стандарту ISO 8601) не существует в московской часовой зоне. При этом какие-то временны́е метки становятся неоднозначными и могут одновременно указывать на два разных фактических момента времени. Для разрешения неопределенности в такой ситуации требуется указание не только названия часовой зоны и местного времени, но и фактического смещения от UTC.

Моделирование шкалы UTC при проектировании языков программирования и решении других задач было бы невозможно без наличия упорядоченной базы данных с исторической информацией. Для учета и унификации исторических изменений в правилах перевода часов, а также административных корректировок часовых зон, в инженерной практике обычно применяется база данных часовых поясов (Time Zone Database, TZDB), которую публикует Администрация адресного пространства Интернет (IANA). В этой базе содержится детальное описание временны́х зон для разных территорий с указанием всех известных исторических сдвигов времени, дат начала и окончания действия летнего времени, а также исключений и поправок, связанных с административными решениями. Сложность и запутанность изменений часовых зон в прошлом задает основные ограничения TZDB. В связи с изменчивостью политических границ государств и их административных регионов, в качестве реперных точек преимущественно используются географические области и названия крупных городов, которые можно считать детерминантами целого региона: например, Asia/Tokyo или Europe/Moscow. В TZDB часовой зоной считается любой национальный регион, где все местные часы согласованы с 1970 г. При этом исторические изменения, происходившие на территориях за пределами выделенных городов до 1970 г., в TZDB не отражены. Таким образом, данную базу нельзя считать полной. Отдельно следует отметить, что в ней хранятся записи о введении дополнительных секунд шкалы UTC.

Может показаться, что, зная конкретный момент времени после 1970 г. и координаты места на земном шаре, с помощью TZDB возможно однозначно установить правила DST в заданной точке пространственно-временного континуума, но это неверно. Спорные территории и культурные разногласия приводят к коллизиям.

Принципиально непреодолимой методологической проблемой DST и часовых зон в целом остается неустранимая непредсказуемость грядущих изменений. Если физические факторы (например, сезонная разница в продолжительности дня или замедление вращения Земли) поддаются хотя бы приблизительному научному прогнозу, то политические и экономические решения — едва ли. Многие государства в короткие сроки могут отменить летнее время либо, наоборот, ввести его, изменить границы часовых зон, объединить две зоны или разбить одну на несколько. Подобная вариативность напрямую отражает фундаментальный тезис о том, что социальная реальность не поддается точному моделированию и алгоритмизации [6, с. 136].

Произвольность и изменчивость часовых зон является фундаментальной методологической проблемой при моделировании гражданского времени в языках программирования. Время в любой часовой

зоне строго привязано к UTC, поэтому иногда местное время пытаются представить как второстепенную точку зрения на единую первичную шкалу, сводя работу с часовыми зонами к форматированному вводу/выводу. Но это представление ошибочно: история изменений смещения на конкретной территории формирует уникальную шкалу местного времени, которая имеет собственный смысл при моделировании исторических событий или формальном описании юридических документов. Все это делает часовые зоны неотъемлемой частью бизнес-логики во многих задачах, поэтому многие языки программирования в той или иной степени поддерживают работу с часовыми зонами за счет предоставления доступа к базе данных исторических изменений TZDB в среде выполнения.

Подводя итог, можно отметить, что ключевыми проблемами работы с часовыми зонами являются: изменчивость границ и правил, неполнота исторических данных, запаздывание обновлений, потенциальные конфликты политических и культурных интересов, принципиальная невозможность предвидеть и заложить будущие изменения. Данные обстоятельства приводят к вопросу о том, в какой мере возможно (и вообще нужно) стремиться к универсальной системе измерения времени, когда само понятие «правильного» времени суток или даты оказывается укорененным в конкретной историко-культурной ситуации. С одной стороны, в эпоху глобализации информационные системы требуют стабильного и предсказуемого исчисления. С другой стороны, социальная изменчивость и неоднородность мира обрекает такие попытки на постоянную «догоняющую» корректировку. Сегодня не вполне ясно, как соотнести идею абстрактной унификации с исторической динамикой множества локальных практик.

Таким образом, историческая эволюция методов измерения времени привела к множеству проблем при их формализации и алгоритмизации в языках программирования. Наличие различных календарных систем, неравномерность солнечных суток, многообразие шкал времени, зависимость часовых зон и правил перехода на летнее время от непредсказуемых административных решений делают задачу создания унифицированной и удобной модели времени практически неразрешимой.

Попытки создания универсальных шкал времени, таких как TAI и UTC, неизбежно приводят к компромиссам между точностью, удобством вычислений и учетом социальной реальности. Дополнительные секунды и изменения правил часовых зон требуют постоянного обновления базы данных временны́х сдвигов, что снижает предсказуемость вычислений и создает зависимость результата от версии базы данных в среде выполнения программы. Стандарты времени, такие как Unix

time, решают некоторые из этих проблем, но не могут устранить фундаментальный разрыв между равномерными шкалами и неравномерными астрономическими циклами.

В контексте языков программирования проблема представления времени выходит за рамки сугубо технического вопроса и становится частью концептуального моделирования. Из-за инерции в социуме и технических стандартах встроенные механизмы работы со временем в популярных языках программирования часто оказываются несовершенными из-за необходимости поддерживать различные исторически сложившиеся инженерные практики, что приводит к несоответствию поведения программы ожиданиям разработчиков и программным ошибкам. Введение новых стандартов, таких как Temporal API в JavaScript, решает часть проблем, но не устраняет методологическую и концептуальную сложность моделирования в задачах обработки и измерения времени.

Универсальная репрезентация времени в языках программирования остается одной из сложных и нерешенных проблем в компьютерных науках, требующей дальнейших фундаментальных исследований для поиска баланса между удобством и универсальностью, с одной стороны, и физическими, историко-культурными и технико-экономическими ограничениями — с другой.

ЛИТЕРАТУРА

[1] Денисова Т.Ю. Онтология времени у Аристотеля. *Идеи и идеалы*, 2017, № 3, с. 100–109.

[2] Левич А.П. Мотивы и задачи изучения времени. *Конструкции времени в естествознании: на пути к пониманию феномена времени. Ч. I. Междисциплинарное исследование.* Москва, Издательство Московского университета, 1996, с. 9–27.

[3] Ивлев В.Ю., Ивлева М.Л., Иноземцев В.А. Эволюция концепций компьютерной репрезентации знания и эпистемологического содержания искусственного интеллекта. *Известия МГТУ*, 2012, № 2, с. 294–298.

[4] Meisner D. The Evolution of the Roman Calendar. *Past Imperfect,* 2009, vol. 15, pp. 290–321. URL: https://journals.library.ualberta.ca/pi/index.php/pi/article/view/6634 (дата обращения 28.02.2025).

[5] Вознесенский И.С. Тайм-менеджмент часовых зон: уроки большой Евразии. *Большая Евразия: развитие, безопасность, сотрудничество*, 2019, № 2–2, с. 87–91.

[6] Пахарь Л.И. Социальная реальность: анализ механизма функционирования. *Ученые записки ОГУ. Сер.: Гуманитарные и социальные науки*, 2014, № 5, с. 136–143.

Статья поступила в редакцию 11.06.2025

Ссылку на эту статью просим оформлять следующим образом:

Соловьев С.А. Проблема репрезентации времени в языках программирования: философско-методологические аспекты. *Гуманитарный вестник*, 2025, вып. 3. EDN YUYXIQ

Соловьев Сергей Александрович — ведущий разработчик ООО «ТЦР».
e-mail: sergey.soloviev@inbox.ru