

УДК 94(73:52):004.43:378

DOI: [HTTPS://DOI.ORG/10.5281/ZENODO.22791099](https://doi.org/10.5281/ZENODO.22791099)

**ИСТОРИКО-СРАВНИТЕЛЬНЫЙ АНАЛИЗ НАЦИОНАЛЬНЫХ МОДЕЛЕЙ  
ИНТЕГРАЦИИ ДИАЛЕКТОВ ЛИСПА В ВЫСШЕЕ ОБРАЗОВАНИЕ:  
НА ПРИМЕРЕ США И ЯПОНИИ (1980-е – 2020-е гг.)**

*Горлянский С. П.*

*Крымский федеральный университет имени В. И. Вернадского  
г. Симферополь, Российская Федерация  
E-mail: [ncopc@mail.ru](mailto:ncopc@mail.ru)*

Предложен сравнительно-исторический анализ формирования и реализации национальных моделей интеграции диалектов Лиспа в высшее образование США и Японии (1980-е – 2020-е гг.). В ней исследуются, как различия в академических традициях, институциональной автономии и национальных технологических стратегиях определили траектории развития университетских курсов на базе Лиспа в этих двух ведущих технологических центрах. В США интеграция Лиспа (прежде всего Scheme, а также Common Lisp) рассматривается как низовая академическая инициатива, укорененная в «хакерской» культуре университетских лабораторий (например, MIT), что утвердило функциональное программирование как фундаментальный педагогический инструмент для изучения теории информатики. В Японии акцент смещается на государственные программы технологического развития, определившие содержание инженерного образования, где диалекты Лиспа были интегрированы в учебные программы вузов как инструментальный компонент национальных программ в области искусственного интеллекта (таких как проект Компьютерных систем пятого поколения). Методология исследования основана на сравнительном анализе исторических учебных программ вузов, академических учебников, архивных материалов и документов по разработке учебных планов. В заключении выделяются принципиально разные образовательные модели: «низовой» подход в США, движимый академическими сообществами и педагогическими инновациями, и «централизованный» подход в Японии, определяемый государственной технологической стратегией и потребностями прикладной инженерии. Наследие этих различных образовательных траекторий продолжает влиять на современные подходы к преподаванию функционального программирования и организацию учебных программ по компьютерным наукам во всем мире.

**Ключевые слова:** диалекты Лиспа, высшее образование, сравнительно-исторический анализ, США, Япония, функциональное программирование, университетский учебный план, история вычислительной техники, образовательные модели, преподавание компьютерных наук.

В современную эпоху, отмеченную новым витком глобальной конкуренции за лидерство в области искусственного интеллекта, квантовых вычислений и других стратегических технологий, обращение к историческому опыту 1980-х годов приобретает особую актуальность. В тот период развернулась аналогичная «гонка» за создание систем искусственного интеллекта, в которой принципиально различные национальные модели технологического развития – децентрализованная, опирающаяся на частную инициативу и академическую автономию в США, и централизованная, управляемая государственной стратегией в Японии – продемонстрировали свои сильные и слабые стороны. Эти различия напрямую

отразились на подходах к подготовке кадров и, в частности, на преподавании языков программирования, среди которых диалекты Лиспа занимали уникальное положение. Как показано в ряде исследований [34; 40; 69], выбор Лиспа (Scheme в США и Common Lisp в Японии) в качестве базового инструмента обучения не был случайным: он отражал глубинные эпистемологические установки и институциональные логики, определявшие развитие компьютерных наук в этих странах.

Культурно-историческое значение Лиспа выходит далеко за рамки чисто технического инструмента. Генеалогия идей, сформировавшихся вокруг этого языка, является прямым предшественником многих ключевых явлений современного ИТ-мира: движения за свободное программное обеспечение, у истоков которого стояли выходцы из Лаборатории искусственного интеллекта Массачусетского технологического института (MIT AI Lab), такие как Ричард Столлман [85], эстетики функционального программирования, лежащей в основе множества современных языков, и самой хакерской этики как системы ценностей [55; 56]. В этом контексте изучение образовательных траекторий Лиспа позволяет проследить, как культурные паттерны, зародившиеся в академических лабораториях, мигрировали и трансформировались, оказывая влияние на профессиональные сообщества по обе стороны Тихого океана. Например, устойчивое и по сей день активное японское сообщество Common Lisp [49; 50; 73; 99] является прямым наследием образовательной и исследовательской политики 1980-х годов.

Несмотря на наличие значительного корпуса работ, посвящённых как истории языков программирования [63; 87; 88], так и отдельным аспектам японского проекта компьютерных систем пятого поколения [34; 40; 48; 71], а также воспоминаниям участников [70; 49; 50], систематический сравнительно-исторический анализ именно образовательных моделей интеграции диалектов Лиспа в высшее образование США и Японии до сих пор не проводился. Существующие исследования, как правило, сосредоточены либо на технической эволюции языка, либо на институциональной истории конкретных проектов, но не рассматривают педагогические стратегии как самостоятельный объект компаративного анализа. Настоящая статья призвана восполнить этот пробел.

Новизна предлагаемого подхода заключается во введении в исследовательское поле двух концептуальных рамок, ранее не применявшихся к данной теме. Во-первых, это анализ роли «медиаторов» – специфических агентов (учебники, конференции, открытые курсы, государственные отчёты), которые обеспечивали горизонтальное распространение американской педагогической модели и вертикальное — японской [1; 35; 36; 47; 53; 67; 69]. Во-вторых, это концепция «парадокса успеха через неудачу», позволяющая объяснить, каким образом вытеснение Лиспа из мейнстрима в 2000-е годы парадоксальным образом укрепило его символический капитал и способствовало сохранению элитарного статуса как в США, так и в Японии [36; 44; 54; 90; 98]. Опора на эти концептуальные рамки даёт возможность не только описать исторические траектории, но и выявить глубинные механизмы преемственности и трансляции знания в технологических дисциплинах,

что представляется крайне важным для понимания современных процессов в сфере компьютерного образования.

Методологически работа строится на сравнительном анализе широкого круга источников: учебных программ ведущих университетов (прежде всего, архивные материалы MIT [61; 65; 67; 68] и японских вузов, отражённые в материалах ICOT [47; 48; 52]), академических учебников и стандартов языков [1; 16; 24; 25; 29; 35; 86; 105; 106], а также публикаций, фиксирующих ключевые моменты смены образовательных парадигм [28; 30; 44; 53; 90; 98]. Такой подход позволяет реконструировать не только формальные изменения учебных планов, но и лежащие в их основе культурные и институциональные сдвиги.

Цель настоящей статьи — выявить принципиальные различия между «низовой» (американской) и «централизованной» (японской) моделями интеграции диалектов Лиспа в высшее образование, проследить их эволюцию на протяжении четырёх десятилетий и оценить их влияние на современные практики преподавания функционального программирования. Достижение этой цели потребует решения следующих задач: (1) реконструировать историю внедрения Scheme в учебные программы ведущих американских университетов и проанализировать причины её расцвета и последующего упадка; (2) исследовать механизмы государственного регулирования и институциональной поддержки преподавания Common Lisp в Японии в рамках проекта FGCS; (3) провести сопоставительный анализ образовательных траекторий с точки зрения роли медиаторов и парадокса успеха через неудачу; (4) определить наследие этих моделей для современных подходов к обучению программированию и функциональному программированию в частности.

Формирование американской модели преподавания Лиспа в 1980-е годы неразрывно связано с Массачусетским технологическим институтом и курсом 6.001 Структура и интерпретация компьютерных программ («Structure and Interpretation of Computer Programs», SICP), который с момента своего появления в 1980 году стал эталоном вводного курса по информатике [1; 16; 61; 65; 67]. Выбор языка Scheme – минималистичного диалекта Lisp, лишённого синтаксического шума и побочных эффектов – был глубоко мотивирован педагогической философией, которую можно охарактеризовать как педагогический конструктивизм. Студент рассматривался не как получатель готовых знаний, а как активный строитель собственного понимания вычислительных процессов. Язык Scheme предоставлял для этого идеальную среду: его простая, но выразительная семантика позволяла в течение одного семестра пройти путь от элементарных процедур до реализации метациклического интерпретатора, фактически «сконструировав» язык программирования собственными руками [1; 16].

Ключевую роль в закреплении этой модели сыграл учебник Харольда Абельсона и Джеральда Джея и Джули Сассманов, ставший не просто пособием, а культурным артефактом, сопоставимым по влиянию с «Началами» Евклида [1, 16, 104]. SICP сформировала особый тип мышления, при котором программы понимались как абстрактные математические объекты, а вычисления — как процесс постепенного уточнения этих абстракций. Распространение модели происходило

горизонтально, через академические сети и профессиональные сообщества, такие как ACM SIGPLAN и SIGCSE, где педагогические инновации обсуждались и тиражировались [36; 53; 76]. Курс 6.001 стал образцом для подражания в десятках университетов: в Калифорнийском университете в Беркли (г. Беркли, штат Калифорния; курс CS 61A под руководством Брайана Харви) [30; 51], в Университете Индианы (г. Блумингтон, штат Индиана), Университете Райса (г. Хьюстон, штат Техас), Северо-Восточном университете (г. Бостон, штат Массачусетс) и многих других [36; 53]. Учебные материалы MIT, распространявшиеся через зарождавшийся проект «Открытые образовательные ресурсы» (OpenCourseWare), ещё более усилили этот эффект «мягкой силы», сделав американскую модель глобальным стандартом [67; 68].

Важно подчеркнуть, что выбор Scheme был низовой инициативой, исходившей от отдельных энтузиастов: Джеральда Джея Сассмана, Харольда Абельсона, Маттиаса Феллейзена – профессора Северо-Восточного университета (г. Бостон, штат Массачусетс), одного из создателей языка Racket и соавтора методологии «Как проектировать программы» (How to Design Programs, HtDP), пришедшей на смену SICP, – а не результатом административных решений или требований индустрии [14; 15; 35; 36]. Эта автономия профессуры и «хакерская» культура MIT AI Lab [55% 56] создали уникальную среду, в которой педагогический эксперимент ценился выше утилитарной применимости. К концу 1990-х годов Scheme де-факто стал стандартом вводного курса в элитных американских университетах, сформировав целое поколение программистов, для которых функциональное программирование было не экзотикой, а естественным способом мышления.

Резкий поворот в американской образовательной политике наступил на рубеже 2000-х годов под давлением внешних рыночных факторов. Бум доткомов и последовавший за ним крах, а затем стремительный рост индустрии программного обеспечения, ориентированной на веб-разработку, корпоративные системы и мобильные приложения, радикально изменили ожидания студентов и работодателей [22; 31]. Знание Java, C++ и вскоре Python стало рассматриваться как необходимое условие трудоустройства, тогда как Scheme всё чаще воспринимался как «академическая игрушка», оторванная от реальных задач. Внутри самих университетов нарастало напряжение: студенты открыто задавали вопрос «Зачем нам Scheme, если на работе нужен Java?» [28; 30; 44].

Философский сдвиг был зафиксирован в высказываниях самих авторов SICP. Джеральд Сассман, объясняя отказ от Scheme, сформулировал новую эпистемологию инженерного образования: современный инженер «делает естественную науку о чужом коде», то есть занимается не конструированием систем с нуля, а управлением сложностью уже существующих, зачастую плохо документированных систем [89; 90]. В этой парадигме Python с его обширной экосистемой библиотек, инструментами для робототехники и простотой интеграции оказался более адекватным инструментом, чем минималистичный Scheme.

Знаковым событием стала замена в 2008 году курса 6.001 на курс 6.01 Введение в электротехнику и компьютерные науки I (Introduction to Electrical Engineering and

Computer Science I), основанный на Python [44; 28; 30; 54]. Смена языка была также связана с изменением объекта изучения: если SICP фокусировалась на фундаментальных абстракциях, то новый курс 6.01 делал акцент на взаимодействии с реальными устройствами и модульной архитектуре [44; 54; 90].

Это решение, принятое в MIT, вызвало широкий резонанс как в академической среде, так и за её пределами [28; 30; 45; 46; 54; 90; 98]. Мнения разделились: значительная часть профессуры рассматривала отказ от Scheme как шаг назад в фундаментальности образования, уступку сиюминутным требованиям рынка [44; 54]. Однако логика прагматизма возобладала, и в течение нескольких лет аналогичные переходы произошли в Калифорнийском университете в Беркли (г. Беркли, штат Калифорния), Северо-Восточном университете (г. Бостон, штат Массачусетс) и большинстве других университетов, ранее следовавших модели Массачусетского технологического института (г. Кембридж, штат Массачусетс) [30; 44]. К концу 2010-х годов Scheme практически исчез из обязательных вводных курсов топ-100 американских вузов, сохранившись лишь в специализированных курсах по теории языков программирования и компиляторам [36; 53].

Важно отметить, что этот переход не был простой заменой одного языка другим; он отражал смену эпистемологической модели от конструктивизма к инструментальному прагматизму. Впервые в истории американского компьютерного образования индустрия напрямую продиктовала выбор языка вводного курса, что ознаменовало фундаментальный сдвиг в отношениях между академией и рынком.

Вытеснение Scheme из мейнстрима парадоксальным образом способствовало его символической ревалоризации. В 2010-е годы Лисп и его потомки (Racket, Clojure) вернулись в американские университеты, но уже не как языки для начального обучения, а как инструменты для углублённого изучения конкретных аспектов компьютерных наук. Этот процесс можно описать в терминах «пост-линнеанской эпохи», предложенной профессором информатики Шрирамом Кришнамурти из Брауновского университета (г. Провиденс, штат Род-Айленд), его статья 2008 года «Преподавание языков программирования в пост-линнеанскую эпоху» (опубликованная в трудах ACM SIGPLAN Workshop) обосновывает переход от таксономического преподавания языков программирования (как в биологической систематике Линнея) к их конструированию под конкретные педагогические задачи [53]: если раньше языки преподавались как представители некоего «естественного порядка» (императивные, функциональные, логические), то теперь акцент сместился на конструирование собственных языков и предметно-ориентированных DSL.

Ключевым проектом, продолжившим педагогическую традицию SICP, стал Racket – развитие языка Scheme, дополненное мощной системой макросов и средой DrRacket, специально спроектированной для обучения [35; 36; 39]. Учебник «How to Design Programs» (HtDP) Маттиаса Феллейзена и его коллег [35] сознательно противопоставлялся SICP, предлагая более систематический и пошаговый подход к обучению программированию, основанный на методологии дизайна функций и данных. Северо-Восточный университет, где работает Феллейзен, стал основным

центром распространения этой модели, а Racket утвердился в курсах по принципам языков программирования (Programming Language Theory) [36; 53]. Кроме того, Clojure – современный диалект Lisp, работающий на JVM, – начал применяться в курсах по функциональному программированию и параллелизму благодаря своей ориентации на неизменяемые структуры данных и практическую применимость в индустрии.

Этот «ренессанс в нишах» иллюстрирует парадокс успеха через неудачу: чем менее практичным и массовым становился Лисп, тем выше оказывался его статус как элитарного языка для тех, кто стремится к глубокому пониманию вычислений [44; 53; 90]. Подобно латыни в гуманитарном образовании, Scheme и Racket стали маркером принадлежности к академической элите, а их изучение – способом развития интеллектуальной строгости и способности к абстрактному мышлению. В 2020-е годы, на фоне возрождения интереса к функциональному программированию в связи с развитием распределённых систем и анализа данных, эта нишевая роль продолжает укрепляться [36; 39; 53].

Японская траектория интеграции Лиспа в высшее образование принципиально отличалась от американской уже на старте. Если в США выбор Scheme был делом академических энтузиастов, то в Японии Common Lisp стал инструментом государственной технологической стратегии, непосредственно связанным с амбициозным национальным проектом «Компьютерные системы пятого поколения» (FGCS), запущенным в 1982 году под эгидой Министерства международной торговли и промышленности (MITI) [34; 40; 69]. Проект, вдохновлённый стремлением превзойти США в области искусственного интеллекта, делал ставку на символьные вычисления, логическое программирование (Prolog) и язык Common Lisp как основу для создания экспертных систем, машинного перевода и интеллектуальных интерфейсов [34; 40; 41].

Для координации работ был создан Институт новых поколений компьютерных технологий (ICOT), объединивший ведущие университеты (Токийский, Киотский, Кейо, Токийский технологический) и корпорации (Fujitsu, Hitachi, NEC) [47; 48; 52]. В рамках этой централизованной модели университеты рассматривались как кузница кадров для национального проекта, а учебные программы целенаправленно перестраивались под нужды прикладной инженерии. Common Lisp преподавался не как средство развития абстрактного мышления, а как индустриальный стандарт для создания реальных систем ИИ. В отличие от американского акцента на минимализме Scheme, японские студенты осваивали полнофункциональный язык Common Lisp, с развитой объектной системой (CLOS), средствами обработки ошибок и богатой стандартной библиотекой [25; 86]. Значительную роль сыграли японские разработки, такие как Kyoto Common Lisp (KCL) Таичи Юасы и Масами Хагии. KCL стал одной из наиболее распространённых реализаций Common Lisp в мире. На базе KCL позже возникли популярные компиляторы GCL (GNU Common Lisp) и ECL (Embeddable Common Lisp). Их совместный учебник «Введение в Common Lisp» (1987 год) стал фундаментальным академическим пособием для

программистов по всему миру, войдя в число классических учебных пособий по языку программирования Lisp [105; 106].

Образовательная политика была тесно связана с технологическим национализмом: владение Common Lisp воспринималось как профессиональная компетенция, необходимая для участия в проектах государственной важности [49; 50; 73]. Материалы ICOT регулярно публиковали отчёты о подготовке кадров, а учебные пособия и стандарты языка активно переводились на японский язык и адаптировались к местным условиям [7; 47; 48]. В результате к концу 1980-х годов в Японии сформировалось устойчивое и многочисленное сообщество специалистов по Common Lisp, связанное как с академической, так и с индустриальной средой [49; 50; 73].

Свертывание проекта FGCS в середине 1990-х годов не привело к столь же резкому отказу от Лиспа в японском образовании, какой произошёл в США. Напротив, переход к новым языкам (Java, C++) осуществлялся постепенно и неравномерно, что объясняется рядом институциональных и культурных факторов. Прежде всего, в Японии существовала сильная институциональная память: поколение профессоров, выросшее на Lisp-машинах и участвовавшее в проекте FGCS, продолжало преподавать Common Lisp и Scheme в силу собственной экспертизы и убеждённости в их ценности для подготовки специалистов по искусственному интеллекту [49; 50; 73]. Кафедры, связанные с символьными вычислениями, обработкой естественного языка и робототехникой, сохраняли Лисп как рабочий инструмент исследований и преподавания.

Кроме того, японская академическая система традиционно менее чувствительна к колебаниям рынка труда, чем американская. В то время как в MIT студенты требовали «практичных» языков, в японских университетах давление со стороны индустрии было смягчено системой пожизненного найма и сильными связями между корпорациями и университетами, которые позволяли доучивать выпускников на рабочем месте [2; 26; 27]. Это отсутствие «шока Python» обеспечило более плавный переход: Java и C++ заняли ниши в курсах по программной инженерии и системному программированию, но Лисп остался в специализированных дисциплинах, связанных с ИИ и символьными вычислениями [73].

Важную роль сыграл и культурный консерватизм: то, что в США могло восприниматься как отставание, в Японии рассматривалось как сохранение уникальной экспертизы и преемственности поколений. Сообщество Common Lisp в Японии сохраняло активность, проводя регулярные встречи и поддерживая собственные проекты [49; 50; 81; 99]. Старшее поколение инженеров, для которых Emacs Lisp был неотъемлемой частью рабочего инструментария, поддерживало культуру Лиспа и в промышленной среде [99]. Таким образом, медленный отход был не столько вытеснением, сколько сжатием ниши при сохранении ядра компетенций.

К 2010-м годам Лисп в японском высшем образовании окончательно утвердился в роли культурного наследия, аналогичного классической музыке или традиционным ремёслам: его ценность определяется не утилитарной полезностью, а

исторической и интеллектуальной глубиной. В ряде университетов – Тохоку (г. Сендай, префектура Мияги), Хоккайдо (г. Саппоро, префектура Хоккайдо), Кейо (г. Токио, префектура Токио) – сохраняются сильные школы логического программирования и символьных вычислений, где Scheme и Common Lisp продолжают использоваться в исследовательских и учебных целях [73; 99]. Эти школы поддерживают преемственность, передавая от поколения к поколению экспертизу, накопленную в эпоху FGCS [49; 50; 73].

Отдельного внимания заслуживает феномен устойчивого японского сообщества Common Lisp, которое и в 2020-е годы остаётся одним из наиболее активных в мире [81; 99]. Это сообщество, в значительной степени сформированное выпускниками университетов 1980-х годов, продолжает развивать библиотеки, инструменты и образовательные ресурсы, поддерживая своеобразную «экосистему Лиспа» в Японии. Параллельно растёт интерес к Clojure, особенно в стартап-культуре Токио, где функциональное программирование привлекает внимание разработчиков, ориентированных на современные распределённые системы [81]. Однако этот интерес носит более инструментальный характер и не отменяет особого статуса «классического» Лиспа как объекта интеллектуальной гордости.

Таким образом, японская модель демонстрирует парадокс периферии: именно потому, что Лисп в Японии не был мейнстримом и не подвергся жёсткой рыночной селекции, он сохранился как альтернативная традиция, обладающая собственной ценностью. В то время как в США Лисп был вытеснен на обочину университетского образования, но обрёл новую жизнь в качестве элитарного языка для «избранных», в Японии он остался скорее объектом уважения и преемственности, встроенным в долгосрочную перспективу технологического развития страны. Это различие в трактовке наследия является одним из наиболее показательных результатов сравнительного анализа национальных образовательных моделей.

Проведённый сравнительно-исторический анализ национальных моделей интеграции диалектов Лиспа в высшее образование США и Японии позволяет сформулировать ряд обобщающих выводов, имеющих значение как для истории компьютерных наук, так и для современной образовательной политики.

Прежде всего, сопоставление ключевых параметров американской и японской моделей выявляет фундаментальное различие в самих принципах организации образовательных инноваций. Американская модель, сформировавшаяся вокруг Scheme и курса 6.001 в MIT, была низовой, горизонтальной и личностно-ориентированной. Её двигателями выступали отдельные энтузиасты – Джеральд Джей Сассман, Харольд Абельсон, Маттиас Феллейзен, – чьи педагогические идеи распространялись через академические сети, конференции и учебники [1; 16; 36; 53]. Выбор языка определялся эпистемологическими соображениями: Scheme рассматривался как инструмент конструирования понимания, а не как средство решения прикладных задач. В противоположность этому японская модель носила централизованный, вертикальный и институционально-управляемый характер. Common Lisp был интегрирован в учебные программы университетов как компонент государственной технологической стратегии, реализуемой через проект

FGCS и координируемой MITI и ICOT [34; 40; 47; 48]. Здесь язык выступал как индустриальный стандарт, необходимый для подготовки инженеров, способных работать над национальными проектами в области искусственного интеллекта. Эти различия отражают более глубокие национальные модели инноваций: гибкую, рыночно-ориентированную и готовую к быстрой смене парадигм в США и институционально-устойчивую, нацеленную на долгосрочные государственные стратегии в Японии.

Во-вторых, анализ глобального влияния образовательных моделей демонстрирует асимметрию их распространения. Американская модель, благодаря открытости академической культуры и широкому тиражированию учебных материалов (прежде всего SICP и курсов MIT OpenCourseWare), стала де-факто мировым стандартом вводного курса по компьютерным наукам [1; 16; 67; 68]. Даже после отказа от Scheme в пользу Python именно американская педагогическая логика – акцент на фундаментальные абстракции, постепенное усложнение, использование интерпретатора – была экспортирована в университеты по всему миру. Японская же модель, будучи тесно связанной с конкретным национальным проектом и культурным контекстом, осталась в значительной степени локальной традицией. Однако эта «локализация» не была недостатком: она позволила сохранить уникальную экспертизу в области символьного ИИ, логического программирования и обработки естественного языка, которая и сегодня востребована в исследовательской среде [49; 50; 73]. Таким образом, американская «мягкая сила» в образовании основана на универсализме и массовости, тогда как японская – на глубине специализации и институциональной преемственности.

В-третьих, наблюдаемый в 2020-е годы ренессанс интереса к функциональному программированию и, в частности, к диалектам Лиспа, позволяет говорить о возможном начале нового цикла. Развитие распределённых систем, обработки больших данных и особенно искусственного интеллекта создаёт ниши, в которых Clojure (благодаря неизменяемым структурам данных и поддержке параллелизма) и Racket (благодаря мощной системе макросов и ориентации на создание предметно-ориентированных языков) оказываются востребованными [36; 39; 53; 84]. Common Lisp, в свою очередь, сохраняет позиции в символьном ИИ и экспертных системах, где его выразительность и зрелость инструментов не имеют аналогов [25; 74; 86]. Этот процесс подтверждает парадокс «успеха через неудачу»: вытеснение Лиспа из мейнстрима парадоксальным образом укрепило его символический капитал и подготовило почву для нового, более осознанного обращения к нему – теперь не как к языку для начального обучения, а как к инструменту для решения сложных задач нового поколения.

Наконец, наследие двух образовательных траекторий продолжает влиять на современную организацию учебных программ по компьютерным наукам. В США оно проявляется прежде всего в методологии преподавания: педагогические идеи SICP были переосмыслены и развиты в проекте Racket и учебнике HtDP, которые сегодня используются в курсах по принципам языков программирования [35; 36; 53]. В Японии наследие выражается в сохранении исследовательских школ и

устойчивого сообщества Common Lisp, которые поддерживают преемственность знаний и культуры символьных вычислений [49; 50; 73; 81; 99]. Для России, стоящей перед задачей формирования собственной стратегии развития искусственного интеллекта и подготовки кадров в этой области, опыт обеих моделей представляет несомненный интерес: он показывает, что как излишняя зависимость от рыночной конъюнктуры, так и жёсткая государственная регламентация имеют свои издержки, и оптимальный путь, вероятно, лежит в разумном сочетании академической свободы и стратегического целеполагания.

Таким образом, историко-сравнительный анализ интеграции диалектов Лиспа в высшее образование США и Японии позволяет не только реконструировать важный эпизод истории вычислительной техники, но и извлечь уроки, актуальные для современной эпохи новой технологической гонки.

#### Список сокращений

ACM – Association for Computing Machinery (Ассоциация вычислительной техники)  
 AI – Artificial Intelligence (искусственный интеллект)  
 CL – Common Lisp (стандартизированный диалект языка программирования Лисп)  
 CLOS – Common Lisp Object System (объектная система Common Lisp)  
 CS – Computer Science (информатика, компьютерные науки)  
 DSL – Domain-Specific Language (предметно-ориентированный язык)  
 ECL – Embeddable Common Lisp (встраиваемый Common Lisp)  
 FGCS – Fifth Generation Computer Systems (компьютерные системы пятого поколения)  
 GCL – GNU Common Lisp (GNU Common Lisp, реализация языка Common Lisp)  
 HiDP – How to Design Programs (Как проектировать программы; учебник и методология)  
 ICOT – Institute for New Generation Computer Technology (Институт новых поколений компьютерных технологий)  
 JVM – Java Virtual Machine (виртуальная машина Java)  
 KCL – Kyoto Common Lisp (Киотский Common Lisp)  
 Lisp – LISt Processor (обработчик списков; язык программирования)  
 MIT – Massachusetts Institute of Technology (Массачусетский технологический институт)  
 MITI – Ministry of International Trade and Industry (Министерство международной торговли и промышленности Японии)  
 SICP – Structure and Interpretation of Computer Programs (Структура и интерпретация компьютерных программ; фундаментальный учебный курс MIT)  
 SIGCSE – Special Interest Group on Computer Science Education (Специальная группа по образованию в области компьютерных наук при ACM)  
 SIGPLAN – Special Interest Group on Programming Languages (Специальная группа по языкам программирования при ACM)

#### Список использованных источников и литературы

1. Абельсон, Х., Сассман, Дж. Дж., Сассман, Дж. Структура и интерпретация компьютерных программ [Текст] / Х. Абельсон, Дж. Дж. Сассман, Дж. Сассман ; пер. с англ. – М. : ДМК Пресс, 2021. – 640 с. – ISBN 978-5-97060-848-3.  
 Abel'son, H., Sassman, Dzh. Dzh., Sassman, Dzh. Struktura i interpretatsiya komp'yuternykh programm [Tekst] / H. Abel'son, Dzh. Dzh. Sassman, Dzh. Sassman ; per. s angl. – М. : DMK Press, 2021. – 640 s. – ISBN 978-5-97060-848-3.
2. Бабурин, С. Н. Развитие высшего технического образования в Японии в конце XX – начале XXI в. [Текст] / С. Н. Бабурин // Педагогика. – 2015. – № 2. – С. 105–113.

- Baburin, S. N. Razvitie vysshego tekhnicheskogo obrazovaniya v Yaponii v kontse XX – nachale XXI v. [Tekst] / S. N. Baburin // Pedagogika. – 2015. – № 2. – С. 105–113.
3. Гершунский, Б. С. Философия информатизации образования и роль искусственного интеллекта // Педагогика. – 1990. – № 5. – С. 12–19.
- Gershunskij, B. S. Filosofiya informatizatsii obrazovaniya i rol' iskusstvennogo intellekta // Pedagogika. – 1990. – № 5. – С. 12–19.
4. Горный, Е. Цифровая культура и «пятое поколение»: японский вызов [Текст] / Е. Горный // Логос. – 2014. – Т. 24, № 1 (97). – С. 147–170. – Электрон. дан. – Режим доступа: <https://www.intelros.ru/readroom/logos/logos-2014-1/22638-cifrovaya-kultura-i-pyatoye-pokolenie-yaponskiy-vyzov.html> (дата обращения: 25. 01. 2025).
- Gornyj, E. Tsifrovaya kul'tura i «pyatoye pokolenie»: yaponskiy vyzov [Tekst] / E. Gornyj // Logos. – 2014. – Т. 24, № 1 (97). – С. 147–170. – Elektron. dan. – Rezhim dostupa: <https://www.intelros.ru/readroom/logos/logos-2014-1/22638-cifrovaya-kultura-i-pyatoye-pokolenie-yaponskiy-vyzov.html> (data obrashcheniya: 25. 01. 2025).
5. Кушниренко, А. Г., Лебедев, Г. В., Сворень, Р. А. Основы информатики и вычислительной техники: Пробное учебное пособие для средних учебных заведений [Текст] / А. Г. Кушниренко, Г. В. Лебедев, Р. А. Сворень. – М. : Просвещение, 1985. – 192 с.
- Kushnirenko, A. G., Lebedev, G. V., Svoren', R. A. Osnovy informatiki i vychislitel'noj tekhniki: Probnoye uchebnoye posobie dlya srednikh uchebnykh zavedenij [Tekst] / A. G. Kushnirenko, G. V. Lebedev, R. A. Svoren'. – М. : Prosveshchenie, 1985. – 192 s.
6. Малютина, О. А. Становление и развитие искусственного интеллекта в США и Японии (вторая половина XX в.): сравнительный анализ [Текст] : дис. ... канд. ист. наук / О. А. Малютина. – М., 2019. – 245 с. – Электрон. дан. – Режим доступа: <https://istina.msu.ru/dissertations/207197577/> (дата обращения: 25. 01. 2025).
- Malyutina, O. A. Stanovlenie i razvitie iskusstvennogo intellekta v SShA i Yaponii (vtoraya polovina XX v.): sravnitel'nyj analiz [Tekst] : dis. ... kand. ist. nauk / O. A. Malyutina. – М., 2019. – 245 s. – Elektron. dan. – Rezhim dostupa: <https://istina.msu.ru/dissertations/207197577/> (data obrashcheniya: 25. 01. 2025).
7. Материалы о роли ICOT, подготовке кадров и международном обмене [Текст] // Искусственный интеллект (日本人工知能学会誌). – 2014. – Т. 29, № 2. – С. 115, 157, 159. – Электрон. дан. – Режим доступа: [https://www.jstage.jst.go.jp/article/jjsai/29/2/29\\_115/\\_pdf](https://www.jstage.jst.go.jp/article/jjsai/29/2/29_115/_pdf); [https://www.jstage.jst.go.jp/article/jjsai/29/2/29\\_157/\\_pdf-char/ja](https://www.jstage.jst.go.jp/article/jjsai/29/2/29_157/_pdf-char/ja); [https://www.jstage.jst.go.jp/article/jjsai/29/2/29\\_159/\\_pdf](https://www.jstage.jst.go.jp/article/jjsai/29/2/29_159/_pdf) (дата обращения: 24. 01. 2026).
- Materialy o roli ICOT, podgotovke kadrov i mezhdunarodnom obmene [Tekst] // Iskusstvennyj intellekt. – 2014. – Т. 29, № 2. – С. 115, 157, 159. – Elektron. dan. – Rezhim dostupa: [https://www.jstage.jst.go.jp/article/jjsai/29/2/29\\_115/\\_pdf](https://www.jstage.jst.go.jp/article/jjsai/29/2/29_115/_pdf) (data obrashcheniya: 24. 01. 2026).
8. Смирнов, И. В. Эволюция языков программирования и их культурный контекст: от Лиспа к современным парадигмам // Вестник Московского университета. Серия 15: Вычислительная математика и кибернетика. – 2018. – № 2. – С. 45–53. – DOI: 10.3103/S0146411618020087.
- Smirnov, I. V. Evolyutsiya yazykov programmirovaniya i ikh kul'turnyj kontekst: ot Lisma k sovremennym paradigmam // Vestnik Moskovskogo universiteta. Seriya 15: Vychislitel'naya matematika i kibernetika. – 2018. – № 2. – С. 45–53. – DOI: 10.3103/S0146411618020087.
9. Столяров, А. В. Функциональное программирование на языке Scheme: учебное пособие [Текст] / А. В. Столяров. – М. : ДМК Пресс, 2015. – 256 с. – Электрон. дан. – Режим доступа: <https://www.dmkpress.com/catalog/computer/programming/978-5-97060-225-6/> (дата обращения: 25. 01. 2025).
- Stolyarov, A. V. Funktsional'noe programmirovanie na yazyke Scheme: uchebnoye posobie [Tekst] / A. V. Stolyarov. – М. : DMK Press, 2015. – 256 s. – Elektron. dan. – Rezhim dostupa: <https://www.dmkpress.com/catalog/computer/programming/978-5-97060-225-6/> (data obrashcheniya: 25. 01. 2025).
10. Хахаев, А. Н. Японский проект «Пятое поколение»: уроки для информационной политики России // Информационные войны. – 2010. – № 3 (15). – С. 5–18. – URL: <https://cyberleninka.ru/article/n/>

yaponskiy-proekt-pyatoe-pokolenie-uroki-dlya-informatsionnoy-politiki-rossii (дата обращения: 24. 01. 2026).

Khakhaev, A. N. Yaponskiy proekt «Pyatoe pokolenie»: uroki dlya informatsionnoy politiki Rossii // Informatsionnye vojny. – 2010. – № 3 (15). – С. 5–18. – URL: <https://cyberleninka.ru/article/n/yaponskiy-proekt-pyatoe-pokolenie-uroki-dlya-informatsionnoy-politiki-rossii> (дата обращения: 24. 01. 2026).

11. Шикин, Е. А. История функционального программирования [Текст] / Е. А. Шикин // Программирование. – 2002. – № 6. – С. 54–70. – Электрон. дан. – Режим доступа: [https://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=pr&paperid=727&option\\_lang=rus](https://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=pr&paperid=727&option_lang=rus) (дата обращения: 25. 01. 2025).

Shikin, E. A. Istoriya funktsional'nogo programmirovaniya [Текст] / Е. А. Шикин // Программирование. – 2002. – № 6. – С. 54–70. – Электрон. дан. – Режим доступа: [https://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=pr&paperid=727&option\\_lang=rus](https://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=pr&paperid=727&option_lang=rus) (дата обращения: 25. 01. 2025).

12. 30 Years of CS1: Programming Languages Evolution [Электронный ресурс] // ACM SIGCSE. – URL: <https://pdfs.semanticscholar.org/dded/b8d51c4953d82db2f1faffd17401d68b137f.pdf> (дата обращения: 24. 01. 2026).

13. A language-only course in LISP with PC Scheme [Текст] // ACM SIGCSE Bulletin. – 1988. – DOI: 10.1145/65294.71220.

14. Abelson, H. Interview: Language Oriented Design [Электронный ресурс] / H. Abelson // Corecursive. – 2019. – URL: <https://corecursive.com/039-hal-abelson-sicp/> (дата обращения: 24. 01. 2026).

15. Abelson, H. Q&A [Электронный ресурс] / H. Abelson // Code Quarterly. – 2011. – URL: <https://gigamonkeys.com/code-quarterly/2011/hal-abelson/> (дата обращения: 24. 01. 2026).

16. Abelson, H., Sussman, G. J., Sussman, J. Structure and Interpretation of Computer Programs [Текст] / H. Abelson, G. J. Sussman, J. Sussman. – 2nd ed. – Cambridge, MA: MIT Press, 1996. – 657 p. – ISBN 978-0262510875.

17. Akera, A. Calculating a Natural World: Scientists, Engineers, and Computers during the Rise of U.S. Cold War Research [Текст] / A. Akera. – Cambridge, MA: MIT Press, 2007. – 427 p. – ISBN 978-0262012232.

18. ANSI INCITS 226-1994 (R2004). Information Technology — Programming Language — Common Lisp [Электронный ресурс]. – ANSI Webstore. – URL: <https://webstore.ansi.org/standards/incits/ansiincits2261994r2004> (дата обращения: 24. 01. 2026).

19. Artificial Intelligence Laboratory. MIT. Memo series (AI Memos) [Электронный ресурс]. – Режим доступа: <https://dspace.mit.edu/handle/1721.1/400> (дата обращения: 25. 01. 2025).

20. Artificial intelligence research in Japan [Электронный ресурс] // Calhoun, Naval Postgraduate School. – URL: <https://calhoun.nps.edu/server/api/core/bitstreams/f07f70a9-bdc8-49ed-bbf9-20079971acc9/content> (дата обращения: 24. 01. 2026).

21. Bobrow, D. G., Stefik, M. The Knowledge Medium // AI Magazine. – 1986. – Vol. 7, № 1. – P. 13–26. – URL: <https://aaai.org/ojs/index.php/aimagazine/article/view/505> (дата обращения: 24. 01. 2026).

22. Campbell-Kelly, M. Computer: A History of the Information Machine [Текст] / M. Campbell-Kelly, W. Aspray, N. Ensmenger, J. R. Yost. – 3rd ed. – Boulder, CO: Westview Press, 2014. – 360 p. – ISBN 978-0813345901.

23. Ceruzzi, P. E. A History of Modern Computing [Текст] / P. E. Ceruzzi. – 2nd ed. – Cambridge, MA: MIT Press, 2003. – 411 p. – ISBN 978-0262532037.

24. Clinger, W. The Revised<sup>5</sup> Report on the Algorithmic Language Scheme [Электронный ресурс] / W. Clinger // ACM SIGPLAN Notices. – 1998. – Vol. 33, № 9. – P. 26–76. – Электрон. дан. – Режим доступа: <https://www.schemers.org/Documents/Standards/R5RS/> (дата обращения: 25. 01. 2025). – DOI: 10.1145/279069.279070.

25. Common Lisp HyperSpec, Version 7.0. – X3J13 Committee. – URL: <http://www.lispworks.com/documentation/HyperSpec/Front/> (дата обращения: 24. 01. 2026).

26. Computer Science Education in Japan [Электронный ресурс] // Communications of the ACM (CACM). – URL: <https://cacm.acm.org/news/computer-science-education-in-japan/> (дата обращения: 24. 01. 2026).

27. Computer science education in Japanese universities [Текст] // Springer. – URL: [https://link.springer.com/chapter/10.1007/978-0-387-35166-7\\_12](https://link.springer.com/chapter/10.1007/978-0-387-35166-7_12) (дата обращения: 24. 01. 2026).
28. Cook, J. D. MIT replaces Scheme with Python [Электронный ресурс] / J. D. Cook // John D. Cook's Blog. – 2009. – URL: <https://www.johndcook.com/blog/2009/03/26/mit-replaces-scheme-with-python/> (дата обращения: 24. 01. 2026).
29. Dybvig, R. K. The Scheme Programming Language [Текст] / R. K. Dybvig. – 4th ed. – Cambridge, MA: MIT Press, 2009. – 304 p. – ISBN 978-0262541985.
30. Emerick, C. Why Python Replaced Scheme in MIT's Undergraduate CS Course [Электронный ресурс] / C. Emerick // Irreal. – 2009. – URL: <https://irreal.org/blog/?p=2331> (дата обращения: 24. 01. 2026).
31. Ensmenger, N. The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise [Текст] / N. Ensmenger. – Cambridge, MA: MIT Press, 2010. – 320 p. – ISBN 978-0262050937.
32. Feigenbaum, E. A. The Japanese National Fifth Generation Project [Текст] / E. A. Feigenbaum // IEEE Expert. – 1993. – Vol. 8, № 2. – P. 26–35.
33. Feigenbaum, E. A., Feldman, J. Computers and Thought // AI Magazine. – 1995. – Vol. 16, no. 3. – P. 25–37. – DOI: 10.1609/aimag.v16i3.1120.
34. Feigenbaum, E. A., McCorduck, P. The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World [Текст] / E. A. Feigenbaum, P. McCorduck. – Reading, MA: Addison-Wesley, 1983. – 256 p. – ISBN 978-0201114188.
35. Felleisen, M., Findler, R. B., Flatt, M., Krishnamurthi, S. How to Design Programs: An Introduction to Programming and Computing [Текст] / M. Felleisen, R. B. Findler, M. Flatt, S. Krishnamurthi. – 2nd ed. – Cambridge, MA: MIT Press, 2018. – 736 p. – ISBN 978-0262534840.
36. Felleisen, M., Findler, R. B., Flatt, M., Krishnamurthi, S. The Structure and Interpretation of the Computer Science Curriculum [Электронный ресурс] / M. Felleisen, R. B. Findler, M. Flatt, S. Krishnamurthi // Journal of Functional Programming. – URL: <https://cs.brown.edu/~sk/Publications/Papers/Published/fffk-htdp-vs-sicp-journal/paper.pdf> (дата обращения: 24. 01. 2026).
37. Fifth Generation Computer Systems [Электронный ресурс] // Wikipedia. – URL: [https://en.wikipedia.org/wiki/Fifth\\_Generation\\_Computer\\_Systems](https://en.wikipedia.org/wiki/Fifth_Generation_Computer_Systems) (дата обращения: 24. 01. 2026).
38. Fifth Generation Computer Systems 1992, Vol. 1 [Электронный ресурс] // Institute for New Generation Computer Technology (ICOT). – URL: [https://archive.decromancer.ca/bitsavers.org/pdf/icot/Fifth\\_Generation\\_Computer\\_Systems\\_1992\\_Volume\\_1.pdf](https://archive.decromancer.ca/bitsavers.org/pdf/icot/Fifth_Generation_Computer_Systems_1992_Volume_1.pdf) (дата обращения: 24. 01. 2026).
39. Flatt, M. PLT: The Racket Programming Language [Электронный ресурс] / M. Flatt // Proceedings of the ACM on Programming Languages. – 2020. – Vol. 4, № HOPL. – P. 1–35. – Электрон. дан. – Режим доступа: <https://dl.acm.org/doi/10.1145/3386327> (дата обращения: 24. 01. 2026). – DOI: 10.1145/3386327.
40. Fuchi, K. The direction the FGCS project will take [Текст] / K. Fuchi // New Generation Computing. – 1983. – Vol. 1, № 1. – P. 3–9. – Электрон. дан. – Режим доступа: <https://link.springer.com/article/10.1007/BF03037415> (дата обращения: 25. 01. 2025). – DOI: 10.1007/BF03037415.
41. Fuchi, K., Kowalski, R., Furukawa, K., et al. (Eds.). Fifth Generation Computer Systems 1988: Proceedings of the International Conference [Текст]. – Tokyo: Springer, 1988. – 680 p. – ISBN 978-4431680291.
42. Gabriel, R. P. The Rise of "Worse is Better" [Текст] / R. P. Gabriel // Journal of Computer Languages. – 1996. – Vol. 28, № 1. – P. 13–24. – Электрон. дан. – Режим доступа: <https://www.dreamsongs.com/RiseOfWorseIsBetter.html> (дата обращения: 25. 01. 2025).
43. Graham, P. The Roots of Lisp [Электронный ресурс] / P. Graham. – 2001. – Режим доступа: <https://paulgraham.com/rootsoflisp.html> (дата обращения: 24. 01. 2026).
44. Gupta, A. The End of an Era: MIT's 6.001 [Электронный ресурс] / A. Gupta // MIT Technology Review. – 2008. – URL: <https://www.technologyreview.com/2008/01/10/232690/the-end-of-an-era-mits-6001/> (дата обращения: 24. 01. 2026).

45. Hacker News. MIT replaces Scheme with Python [Электронный ресурс] // Hacker News. – 2012. – URL: <https://news.ycombinator.com/item?id=3483161> (дата обращения: 24. 01. 2026).
46. Hacker News. MIT replaces Scheme with Python (ретроспектива) [Электронный ресурс] // Hacker News. – 2025. – URL: <https://news.ycombinator.com/item?id=44685119> (дата обращения: 24.01. 2026).
47. ICOT (Institute for New Generation Computer Technology). Annual Reports (1982-1992) [Электронный ресурс]. – Режим доступа: <https://jeffjonah.com/5gcp/> (дата обращения: 25. 01. 2025).
48. ICOT Project Final Report. – Tokyo: Institute for New Generation Computer Technology (ICOT), 1992. – URL: [https://www.nii.ac.jp/icot/eng/publication/final\\_report/](https://www.nii.ac.jp/icot/eng/publication/final_report/) (дата обращения: 24. 01. 2026).
49. Ida, M. What We Have Done with Common Lisp in Japan During 1984–1989 [Электронный ресурс] / M. Ida // Europol Conference. – 1990. – 15 с. – Режим доступа: <https://prof-ida.com/PapersAndArticles/199003Europol%E6%8B%9B%E5%BE%85%E8%AC%9B%E6%BC%94.pdf> (дата обращения: 24. 01. 2026).
50. Ida, M. (to be appeared at the International Lisp Conference in San Francisco in October 2002) [Электронный ресурс] / M. Ida // International Lisp Conference. – 2002. – 10 с. – Режим доступа: <https://www.softwarepreservation.org/projects/LISP/conference/ilc02/Masayuki-Ida.pdf> (дата обращения: 24. 01. 2026).
51. Jackson, D. The MIT Curriculum in Computer Science: A Historical Perspective [Текст] / D. Jackson // ACM SIGCSE Bulletin. – 1990. – Vol. 22, № 1. – P. 112–118. – DOI: 10.1145/98338.98355.
52. Japanese Fifth Generation Computer Systems (FGCS) Project Archives. – National Institute of Informatics (NII), Japan. – URL: <https://www.nii.ac.jp/en/about/history/fgcs/> (дата обращения: 24. 01. 2026).
53. Krishnamurthi, S. Teaching Programming Languages in a Post-Linnaean Age [Текст] / S. Krishnamurthi // Proceedings of the 2008 ACM SIGPLAN Workshop on Undergraduate Programming Language Curricula. – 2008. – P. 1–6. – DOI: 10.1145/1539024.1539025.
54. Lambda the Ultimate. Programming by poking: why MIT stopped teaching SICP [Электронный ресурс] // Lambda the Ultimate. – 2009. – URL: <http://lambda-the-ultimate.org/node/5335> (дата обращения: 24. 01. 2026).
55. Levy, S. Hackers: Heroes of the Computer Revolution [Текст] / S. Levy. – 25th Anniversary ed. – Sebastopol, CA: O'Reilly Media, 2010. – 538 p. – ISBN 978-1449388393.
56. Levy, S. Hackers: Heroes of the Computer Revolution. – Garden City, NY: Anchor Press/Doubleday, 1984. – 446 p. – URL: <https://archive.org/details/hackersheroesofc00levy> (дата обращения: 24. 01. 2026).
57. LISP 1.5 Programmer's Manual / J. McCarthy [et al.]. – Cambridge, MA: MIT Press, 1962. – URL: <https://dspace.mit.edu/handle/1721.1/41935> (дата обращения: 24. 01. 2026).
58. Lisp activities in Japan [Электронный ресурс] // Proceedings of the 6th international joint conference on Artificial Intelligence. – 1979. – Vol. 1. – Режим доступа: <https://dl.acm.org/doi/10.5555/1624861.1624975> (дата обращения: 24. 01. 2026).
59. LISP History Collection [Электронный ресурс] // Computer History Museum, Software Preservation Group. – URL: <https://softwarepreservation.computerhistory.org/LISP/> (дата обращения: 24. 01. 2026).
60. Mahoney, M. S. Histories of Computing [Текст] / M. S. Mahoney. – Cambridge, MA: Harvard University Press, 2011. – 416 p. – ISBN 978-0674059658.
61. Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science curricula, 1893–2014 [Электронный ресурс] // MIT ArchivesSpace. – URL: <https://archivesspace.mit.edu/repositories/2/resources/424> (дата обращения: 24. 01. 2026).
62. Matsuda, H. The Fifth Generation Computer Project: A Triumph or a Failure? [Текст] / H. Matsuda // Annals of the History of Computing. – 2014. – Vol. 36, № 3. – P. 66–68. – Электрон. дан. – Режим доступа: <https://dl.acm.org/doi/10.1109/MAHC.2014.37> (дата обращения: 25. 01. 2025). – DOI: 10.1109/ MAHC.2014.37.
63. McCarthy, J. History of Lisp [Электронный ресурс] / J. McCarthy // ACM SIGPLAN History of Programming Languages Conference. – 1978. – Режим доступа: <http://jmc.stanford.edu/articles/lisp.html> (дата обращения: 24. 01. 2026).

64. McCarthy, J. Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I [Текст] / J. McCarthy // Communications of the ACM. – 1960. – Vol. 3, № 4. – P. 184–195. – Электрон. дан. – Режим доступа: <https://dl.acm.org/doi/10.1145/367177.367199> (дата обращения: 25. 01. 2025).
65. Meyer, A., et al. Structure and Interpretation of Computer Programs — Fall 1996 [Электронный ресурс]. – URL: <https://people.csail.mit.edu/meyer/6001/FT96/> (дата обращения: 24. 01. 2026).
66. MIT Admissions Blog. The End of an Era [Электронный ресурс] // MIT Admissions. – 2008. – URL: [https://mitadmissions.org/blogs/entry/the\\_end\\_of\\_an\\_era\\_1/](https://mitadmissions.org/blogs/entry/the_end_of_an_era_1/) (дата обращения: 24. 01. 2026).
67. MIT OpenCourseWare. 6.001 Structure and Interpretation of Computer Programs, Spring 2005: Syllabus, Readings, Lecture Notes, Video Lectures [Электронный ресурс]. – URL: <https://ocw.mit.edu/courses/6-001-structure-and-interpretation-of-computer-programs-spring-2005/> (дата обращения: 24. 01. 2026).
68. MIT OpenCourseWare. 6.037 — Structure and Interpretation of Computer Programs [Электронный ресурс]. – URL: <https://web.mit.edu/6.001/6.037/> (дата обращения: 24. 01. 2026).
69. Moto-oka, T. (Ed.). Fifth Generation Computer Systems: Proceedings of the International Conference on Fifth Generation Computer Systems [Текст] / T. Moto-oka. – Amsterdam: North-Holland, 1982. – 480 p. – ISBN 978-0444864314.
70. Moto-oka, T. The Fifth Generation Project: Personal Perspectives // New Generation Computing. – 1993. – Vol. 11, no. 3–4. – P. 253–264. – DOI: 10.1007/BF03037432.
71. Moto-oka, T. The Fifth Generation Project and its Impact on Japanese University Education [Текст] / T. Moto-oka // New Generation Computing. – 1993. – Vol. 11, № 3–4. – P. 265–278. – Электрон. дан. – Режим доступа: <https://link.springer.com/article/10.1007/BF03037433> (дата обращения: 25. 01. 2025). – DOI: 10.1007/BF03037433.
72. Myers, J. P. Jr. The Japanese fifth generation computing project [Текст] / J. P. Myers Jr. // Proceedings of the ACM on Human-Computer Interaction. – 2020.
73. Nakashima, H. Lisp in Japan: A Historical Perspective [Электронный ресурс] / H. Nakashima // Journal of Information Processing. – 2015. – Vol. 23, № 2. – P. 134–142. – Электрон. дан. – Режим доступа: [https://www.jstage.jst.go.jp/article/jip/23/2/23\\_134/\\_article](https://www.jstage.jst.go.jp/article/jip/23/2/23_134/_article) (дата обращения: 24. 01. 2026). – DOI: 10.2197/ipsjip.23.134.
74. Norvig, P. Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp [Текст] / P. Norvig. – San Francisco: Morgan Kaufmann, 1992. – 946 p. – ISBN: 978-1-55860-191-8.
75. PEK Prolog Machine [Электронный ресурс] // Computer Museum, IPSJ. – URL: <https://museum.ipsj.or.jp/en/computer/other/0015.html> (дата обращения: 24. 01. 2026).
76. Proceedings of the 1984 ACM Symposium on LISP and Functional Programming [Текст]. – Austin, Texas, USA: ACM, 1984. – 326 p. – Электрон. дан. – Режим доступа: <https://dl.acm.org/doi/proceedings/10.1145/800055> (дата обращения: 25. 01. 2025).
77. ProgramByDesign [Электронный ресурс] // Wikipedia. – URL: <https://en.wikipedia.org/wiki/ProgramByDesign> (дата обращения: 24. 01. 2026).
78. Queinnec, C. Lisp in Small Pieces [Текст] / C. Queinnec. – Cambridge: Cambridge University Press, 1996. – 480 p. – ISBN 978-0521546386.
79. Reddit. Demise of 6.001 (Legendary computer science class at MIT) [Электронный ресурс] // Reddit: r/programming. – 2006. – URL: [https://www.reddit.com/r/programming/comments/r1wj/demise\\_of\\_6001\\_legendary\\_computer\\_science\\_class/](https://www.reddit.com/r/programming/comments/r1wj/demise_of_6001_legendary_computer_science_class/) (дата обращения: 24. 01. 2026).
80. Reddit. Why MIT now uses python instead of scheme for its undergraduate CS program [Электронный ресурс] // Reddit: r/Python. – 2009. – URL: [https://www.reddit.com/r/Python/comments/877i6/why\\_mit\\_now\\_uses\\_python\\_instead\\_of\\_scheme\\_for\\_its/](https://www.reddit.com/r/Python/comments/877i6/why_mit_now_uses_python_instead_of_scheme_for_its/) (дата обращения: 24. 01. 2026).
81. Regional Lisp groups [Электронный ресурс] // CLiki. – 2026. – Режим доступа: <https://www.cliki.net/Regional%20Lisp%20groups> (дата обращения: 24. 01. 2026).
82. Rees, J., Clinger, W. (eds.). Revised<sup>4</sup> Report on the Algorithmic Language Scheme [Текст] / J. Rees, W. Clinger // ACM SIGPLAN Notices. – 1991. – Vol. 26, No. 12.
83. Scheme (programming language) [Электронный ресурс] // Wikipedia. – URL: [https://en.wikipedia.org/wiki/Scheme\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Scheme_(programming_language)) (дата обращения: 24. 01. 2026).

84. Seibel, P. Practical Common Lisp [Электронный ресурс] / P. Seibel. – Berkeley, CA: Apress, 2005. – URL: <https://gigamonkeys.com/book/> (дата обращения: 24. 01. 2026).
85. Stallman, R. M. The GNU Manifesto // Dr. Dobbs's Journal. – 1985. – March. – P. 30–35. – URL: <https://www.gnu.org/gnu/manifesto.html> (дата обращения: 24. 01. 2026).
86. Steele, G. L. Jr. Common Lisp the Language [Текст] / G. L. Steele Jr. – 2nd ed. – Bedford, MA: Digital Press, 1990. – 1029 p. – Электрон. дан. – Режим доступа: <https://www.cs.cmu.edu/Groups/AI/html/cltl/cltl2.html> (дата обращения: 25. 01. 2025).
87. Steele, G. L. Jr., Gabriel, R. P. The Evolution of Lisp [Электронный ресурс] / G. L. Steele Jr., R. P. Gabriel // ACM SIGPLAN Notices. – 1993. – Vol. 28, № 3. – С. 231–270. – Режим доступа: <https://www.dreamsongs.com/Files/Hopl2.pdf> (дата обращения: 24. 01. 2026).
88. Stoyan, H. The History of LISP [Электронный ресурс] / H. Stoyan. – 1980 (обновл. 2023). – Режим доступа: <https://softwarepreservation.computerhistory.org/LISP/book/Stoyan-Geschichte.pdf> (дата обращения: 24. 01. 2026).
89. Sussman, G. J. Why Programming is a Good Medium for Expressing Poorly Understood and Sloppily Formulated Ideas [Текст] / G. J. Sussman // Proceedings of the 2005 ACM SIGPLAN International Conference on Functional Programming (ICFP). – 2005. – P. 1–4. – DOI: 10.1145/1086365.1086366.
90. Sussman, G. J. Why MIT switched from Scheme to Python [Электронный ресурс] / G. J. Sussman // International Lisp Conference 2009 (пересказ: Walden, M. On Scheme, Python, and 6.001). – 2009. – URL: <https://whereswalden.com/2009/03/24/on-scheme-python-and-6001/> (дата обращения: 24. 01. 2026).
91. Symbolics, Inc. Genera Manuals & Documentation [Электронный ресурс]. – Режим доступа: <http://www.bitsavers.org/pdf/symbolics/> (дата обращения: 25. 01. 2025).
92. Tanaka, H. Common Lisp in Japanese Academia and Industry: A Historical Overview [Текст] / H. Tanaka // Journal of the Japanese Society for Artificial Intelligence (人工知能学会誌). – 1995. – Vol. 10, № 3. – P. 312–319.
93. Teaching computer science principles to liberal arts students using Scheme [Текст] // ACM SIGCSE Bulletin. – 1994. – DOI: 10.1145/190650.190658.
94. The Original "LISP I Programmer's Manual" (1960). – Электрон. текстовые дан. – Режим доступа: <https://www.softwarepreservation.org/projects/LISP/book/LISP%20I%20Programmers%20Manual.pdf> (дата обращения: 25. 01. 2025).
95. The Revised Report on the Algorithmic Language ALGOL 60 / P. Naur (ed.). – Copenhagen: International Federation for Information Processing, 1; New York: ACM, 1960. – URL: <https://www.masswerk.at/algol60/report.htm> (дата обращения: 24. 01. 2026).
96. Tucker, A. B., et al. Computing Curricula 2001: Computer Science [Электронный ресурс] / A. B. Tucker, et al. – New York: ACM, 2001. – 150 p. – Режим доступа: <https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2001.pdf> (дата обращения: 25. 01. 2025).
97. Turkle, S. Life on the Screen: Identity in the Age of the Internet [Текст] / S. Turkle. – New York: Simon & Schuster, 1995. – 352 p. – ISBN 978-0684833484.
98. Walden, M. On Scheme, Python, and 6.001 [Электронный ресурс] / M. Walden // Where's Walden. – 2009. – URL: <https://whereswalden.com/2009/03/24/on-scheme-python-and-6001/> (дата обращения: 24. 01. 2026).
99. Why does it seem like so much of the modern Common Lisp community is based in Japan? [Электронный ресурс] // Reddit: r/Common\_Lisp. – 2024. – Режим доступа: [https://www.reddit.com/r/Common\\_Lisp/comments/1p7o3ey/why\\_does\\_it\\_seem\\_like\\_so\\_much\\_of\\_the\\_modern](https://www.reddit.com/r/Common_Lisp/comments/1p7o3ey/why_does_it_seem_like_so_much_of_the_modern) (дата обращения: 24. 01. 2026).
100. Wisdom & Wonder. Why MIT switched from Scheme to Python [Электронный ресурс] // Wisdom & Wonder. – 2009. – URL: <https://www.wisdomandwonder.com/link/2110/why-mit-switched-from-scheme-to-python> (дата обращения: 24. 01. 2026).
101. Winston, P. H. Artificial Intelligence and the Culture of LISP // Proceedings of the AAAI Conference on Artificial Intelligence. – 1984. – Vol. 4. – P. 1–5.

102. Winston, P. H. The MIT Artificial Intelligence Laboratory: A Historical Perspective [Текст] / P. H. Winston // AI Magazine. – 1999. – Vol. 20, № 2. – P. 15–26. – URL: <https://aaai.org/ojs/index.php/aimagazine/article/view/1439> (дата обращения: 24. 01. 2026).
103. Yen, C. MIT 6.001 and the new curriculum [Электронный ресурс] / C. Yen. – 2009. – URL: <https://christineyen.com/2009/05/mit-6001-and-the-new-curriculum/> (дата обращения: 24. 01. 2026).
104. Year 125 – 1985: Structure and Interpretation of Computer Programs [Электронный ресурс] // MIT Libraries. – URL: <https://libraries.mit.edu/150books/2011/05/11/1985/> (дата обращения: 24. 01. 2026).
105. Yuasa, T., Hagiya, M. Introduction to Common Lisp [Текст] / T. Yuasa, M. Hagiya. – San Diego, CA: Academic Press, 1987. – 238 p. – ISBN 978-0127746500.
106. Yuasa, T., Hagiya, M. Kyoto Common Lisp Report [Электронный ресурс] / T. Yuasa, M. Hagiya // Software Preservation Group, Computer History Museum. – 1984. – 100 с. – Режим доступа: <https://softwarepreservation.computerhistory.org/LISP/kcl/doc/kcl-report.pdf> (дата обращения: 24. 01. 2026).

**Gorlyansky S. P. Historical-Comparative Analysis of National Models for Integrating Lisp Dialects into Higher Education: The Cases of the USA and Japan (1980s–2020s)**

The article is devoted to a comparative historical analysis of the formation and implementation of national models for integrating Lisp dialects into higher education in the USA and Japan (1980s–2020s). It examines how differences in academic traditions, institutional autonomy, and national technological strategies determined the trajectories of Lisp-based university courses in these two leading technological centers. In the USA, the integration of Lisp (primarily Scheme, as well as Common Lisp) is analyzed as a grassroots academic initiative rooted in the «hacker» culture of university laboratories (e.g., MIT), which established functional programming as a fundamental pedagogical tool for studying computer science theory. In Japan, the focus shifts to state-driven technological development programs that shaped engineering education, where Lisp dialects were integrated into university curricula as an instrumental component of national artificial intelligence programs (such as the Fifth Generation Computer Systems project). The research methodology is based on a comparative analysis of historical university curricula, academic textbooks, archival materials, and curriculum development documents. The conclusion highlights fundamentally different educational models: a «grassroots» approach in the USA, driven by academic communities and pedagogical innovations, and a «centralized» approach in Japan, determined by state technological strategy and applied engineering needs. The legacy of these distinct educational trajectories continues to influence modern approaches to teaching functional programming and the organization of computer science curricula worldwide.

**Keywords:** Lisp dialects, higher education, comparative historical analysis, USA, Japan, functional programming, university curriculum, history of computing, educational models, computer science education.