TECH SCIENCE

ISSN 3030-3702

TEXNIKA FANLARINING DOLZARB MASALALARI

TOPICAL ISSUES OF TECHNICAL SCIENCES

№ 12 (3) 2025

TECHSCIENCE.UZ

№ 12 (3)-2025

TEXNIKA FANLARINING DOLZARB MASALALARI

TOPICAL ISSUES OF TECHNICAL SCIENCES

TOSHKENT-2025

BOSH MUHARRIR:

KARIMOV ULUG‘BEK ORIFOVICH

TAHRIR HAY’ATI:

Usmankulov Alisher Kadirkulovich - Texnika fanlari doktori, professor, Jizzax politexnika universiteti

Fayziyev Xomitxon – texnika fanlari doktori, professor, Toshkent arxitektura qurilish instituti;

Rashidov Yusuf Karimovich – texnika fanlari doktori, professor, Toshkent arxitektura qurilish instituti;

Adizov Bobirjon Zamirovich– Texnika fanlari doktori, professor, O‘zbekiston Respublikasi Fanlar akademiyasi Umumiy va noorganik kimyo instituti;

Abdunazarov Jamshid Nurmuxamatovich - Texnika fanlari doktori, dotsent, Jizzax politexnika universiteti;

Umarov Shavkat Isomiddinovich – Texnika fanlari doktori, dotsent, Jizzax politexnika universiteti;

Bozorov G‘ayrat Rashidovich – Texnika fanlari doktori, Buxoro muhandislik-texnologiya instiuti;

Maxmudov Muxtor Jamolovich – Texnika fanlari doktori, Buxoro muhandislik-texnologiya instiuti;

Asatov Nurmuxammat Abdunazarovich – Texnika fanlari nomzodi, professor, Jizzax politexnika universiteti;

Mamayev G‘ulom Ibroximovich – Texnika fanlari bo‘yicha falsafa doktori (PhD), Jizzax politexnika universiteti;

Ochilov Abduraxim Abdurasulovich – Texnika fanlari bo‘yicha falsafa doktori (PhD), Buxoro muhandislik-texnologiya instiuti.

OAK Ro‘yxati
Mazkur jurnal O‘zbekiston Respublikasi Oliy ta’lim, fan va innovatsiyalar vazirligi huzuridagi Oliy attestatsiya komissiyasi Rayosatining 2025-yil 8-maydagi 370-son qarori bilan texnika fanlari bo‘yicha ilmiy darajalar yuzasidan dissertatsiyalar asosiy natijalarini chop etish tavsiya etilgan ilmiy nashrlar ro‘yxatiga kiritilgan.

Muassislar: “SCIENCEPROBLEMS TEAM” mas’uliyati cheklangan jamiyati;
Jizzax politexnika insituti.

TECHSCIENCE.UZ- TEXNIKA FANLARINING DOLZARB MASALALARI elektron jurnali 15.09.2023-yilda 130343-sonli guvohnoma bilan davlat ro‘yxatidan o‘tkazilgan.

TAHRIRIYAT MANZILI:
Toshkent shahri, Yakkasaroy tumani, Kichik Beshyog‘och ko‘chasi, 70/10-uy.
Elektron manzil:
scienceproblems.uz@gmail.com

Barcha huqular himoyalangan.
© Sciencesproblems team, 2025-yil
© Mualliflar jamoasi, 2025-yil

TEXNIKA FANLARINING DOLZARB MASALALARI
3-jild, 12 -son (Dekabr, 2025). – 205 bet.

MUNDARIJA

Texnika fanlarining dolzarb masalalari
Topical Issues of Technical Sciences
2025-yil | 3-jild | 12-son
ISSN (onlayn): 3030-3702

ПРОГРАММИРОВАНИЕ ДЛЯ СИСТЕМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА И АВТОМАТИЗАЦИИ

Тураев Хуршид Салим угли
Кандидат технических наук (кандидат наук)
Ташкентский государственный технический университет, доцент
Email: xturayev@mail.ru
Тель: +998 90 325 85 55
Orcid: https://orcid.org/0000-0002-6779-3645

Аннотация. Искусственный интеллект (ИИ) — междисциплинарная область, направленная на разработку методов и технологий, имитирующих интеллектуальное поведение человека. Выбор подходящего языка программирования имеет ключевое значение для разработки эффективных систем ИИ и автоматизированных технологических процессов. В данной статье представлен обзор четырёх широко используемых языков программирования в области ИИ — LISP, Prolog, C++ и Python, с акцентом на их сильные и слабые стороны, а также практические области применения в интеллектуальных и автоматизированных системах. Особое внимание уделено их пригодности для логического вывода, числовых вычислений, быстрого прототипирования, а также интеграции в масштабные автоматизированные проекты. Анализ показывает, что для оптимизации производительности, поддержки и гибкости часто используется комбинация нескольких языков, что особенно важно для автоматизации и управления сложными системами.

Ключевые слова: Искусственный интеллект, автоматизация, LISP, Prolog, C++, Python, логическое программирование, машинное обучение, глубокое обучение, языки программирования, интеллектуальные системы.

PROGRAMMING FOR ARTIFICIAL INTELLIGENCE AND AUTOMATION SYSTEMS

Turaev Khurshid Salim ugli
Candidate of Technical Sciences (PhD)
Tashkent State Technical University, Associate Professor

Annotation. Artificial Intelligence (AI) is an interdisciplinary field focused on developing methods and technologies that replicate human intellectual behavior. Choosing the appropriate programming language is crucial for developing effective AI systems and automated technological processes. This paper provides an overview of four widely used programming languages in AI — LISP, Prolog, C++, and Python — highlighting their strengths, weaknesses, and practical applications in intelligent and automated systems. Special attention is given to their suitability for logical reasoning, numerical computations, rapid prototyping, and integration into large-scale automated projects. The analysis shows that to optimize performance, maintainability, and flexibility, a combination of several languages is often used, which is particularly important for automation and the management of complex systems.

Keywords: Artificial Intelligence, automation, LISP, Prolog, C++, Python, logical programming, machine learning, deep learning, programming languages, intelligent systems.

DOI: https://doi.org/10.47390/ts-v3i12y2025N05

1. Введение

Искусственный интеллект (ИИ) — это область, направленная на создание методов и технологий, позволяющих воспроизводить элементы человеческого интеллектуального поведения. Задачи ИИ выходят далеко за рамки изучения природы разума: ключевая цель состоит в разработке практических систем, способных демонстрировать интеллектуальные функции и принимать обоснованные решения [1-3].

Развитие ИИ сформировало особые требования к используемым языкам программирования. Такие языки должны обеспечивать эффективные средства для представления и обработки знаний, а также для построения **символьных (знаковых) систем**, поскольку именно они позволяют моделировать мыслительные процессы. В основе данного подхода лежит идея о том, что человеческое мышление можно рассматривать как процесс преобразования структур из символов по заранее определённым правилам в рамках некоторой логической системы [3].

Как и в других областях разработки программного обеспечения, специалисты по ИИ располагают широким набором языков программирования. Однако ни один из них не может считаться универсальным для всех задач ИИ. Оптимальный выбор зависит от целей будущего приложения и требуемой функциональности.

В данной работе рассматриваются четыре наиболее распространённых языка программирования, применяемых в сфере искусственного интеллекта, их сильные стороны и возможные недостатки. Идеальный язык для ИИ должен предоставлять удобные механизмы для моделирования знаний о реальном мире и для логического вывода новых сведений на основе имеющейся информации [4].

Особенно важную роль такие возможности играют в **системах управления**, где ИИ используется для повышения точности регулирования, диагностики, прогнозирования и оптимизации процессов. Интеллектуальные алгоритмы позволяют адаптировать управление к изменяющимся условиям и значительно повышают надёжность работы сложных технических систем.

2. LISP

LISP появился в 1958 году и считается одним из самых старых языков программирования, который активно используется и сегодня. По времени создания он идёт сразу после Fortran (1957). Изначально LISP был задуман как удобная математическая нотация для записи вычислительных процедур, а его концепции во многом основаны на λ-исчислении Алонсо Чёрча. Спустя короткое время язык стал основным инструментом для исследований в области искусственного интеллекта.

LISP дал толчок множеству идей, ставших классическими в компьютерных науках:

— работа с древовидными структурами данных,
— автоматическое управление памятью,
— динамическая типизация,
— условные конструкции,
— рекурсивные вызовы,
— цикл «чтение–вычисление–вывод» (read–eval–print) [9].

В языке существует всего два вида данных: атомы и S-выражения. **Атомы** — это простейшие строковые обозначения, используемые как символы. **S-выражение** может

быть атомом или списком, состоящим из других S-выражений. Благодаря этому в S-выражениях можно кодировать структуры любой сложности, включая деревья:

(temp-23 petar mihaela)

(students (temp-23 petar mihaela))

И программы, и данные в LISP имеют форму S-выражений и записываются как вложенные списки. Философия языка состоит в том, чтобы использовать одну универсальную структуру данных — список — вместо множества разных типов. Это позволяет программисту сосредоточиться на задаче, а не на обслуживании синтаксических ограничений.

Как правило, первый элемент списка интерпретируется как имя функции, а последующие — как её аргументы. LISP позволяет определять новые функции, а их описание основано на λ-исчислении Чёрча. Пример определения функции, вычисляющей площадь круга:

Функция def не вычисляет аргументы, а просто связывает второе поле списка (здесь — circle-area) с λ-выражением, указанным в третьем элементе.

При построении базы данных LISP допускает, чтобы каждому атому соответствовал список пар «свойство—значение» (property list, p-list). В задачах искусственного интеллекта такие списки часто используются для моделирования графов: атомы служат узлами, свойства — помеченными рёбрами.

Когда же требуется более строгий формализованный подход, факты предметной области начинают представлять не в виде свойств, а в виде отдельных утверждений или n-кортежей [5]:

(student name temp-23)

(teacher class robotics)

Одним из ключевых механизмов доступа к подобной базе является сопоставление с шаблоном (pattern matching). Шаблон структурно похож на утверждение, но может содержать переменные. Если переменные можно заменить атомами так, чтобы шаблон совпал с некоторым утверждением, то сопоставление считается успешным:

(student name ?x)

(student name temp-23)

Здесь переменная ?x заменяется атомом temp-23.

Потребность в ещё более мощных средствах работы с фактологическими базами данных — включая поиск с возвратом — привела к появлению логически ориентированных языков программирования, основанных на формальной логике [6-9].

3. Prolog

Идея создания программ, которые могут выполняться компьютером и используют аппарат предикатной логики вместе с принципом резолюции (то есть программ, работающих с логическими формулами), оказалась настолько плодотворной, что положила начало целому направлению — логическому программированию. Наиболее известным языком, представляющим эту парадигму, является **Prolog**[2] (от *programmation en logique* — «программирование в логике»), разработанный примерно в 1972 году Аленом Колмерауэром (Alain Colmerauer) и Филиппом Русселем (Philippe Roussel). Основой послужила процедурная интерпретация Хорновских клауз, предложенная Робертом Ковальски (Robert Kowalski)[10].

Синтаксис Prolog основан на *обратных* Хорновских клаузах вида:

u ← p ∧ q ∧ r ∧ ... ∧ s

Для удобства в языке используются следующие обозначения:

- символ импликации заменён на :-
- логическое «И» заменено запятыми

Таким образом, клауза записывается как:

u :- p, q, r, ..., s.

В Prolog различают четыре типа клауз:

– **факты** — литералы вида u :- . (чаще — просто u.)

– **правила** — положительные клаузы вида u :- p, q, r, ..., s.

– **цели (запросы)** — отрицательные клаузы вида :- p, q, r, ..., s.

– **пустая клауза** — обозначает логическое противоречие :- .

Факт — утверждение, которое считается истинным без условий в выбранной предметной области.
Правило истинно тогда, когда истинна конъюнкция литералов в его правой части. **Негативные клаузы** выступают как инвертированные цели, а **пустая клауза** указывает на возникновение логического конфликта.

При создании программы разработчик всегда опирается на некоторую предметную интерпретацию, связывая константы, функции и предикаты с объектами реального мира. Его задача — сформировать базу знаний, которой затем можно задавать вопросы, получая логические выводы.

Например, пусть база содержит:

likes(adam, X) :- likes(X, apples).

likes(eve, apples).

Тогда ответ на запрос:

:- likes(adam, X).

будет:

likes(adam, eve).

Смысл логической программы — это набор всех базовых фактов, логически выводимых из неё.

Помимо декларативного подхода Prolog допускает и процедурное толкование: головная часть правила u рассматривается как имя процедуры, выполнение которой требует поочерёдного вызова p, q, r, ..., s. В этом случае порядок литералов имеет значение, и коммутативность не действует.

Аргументы головного предиката играют роль входных и выходных параметров, передаваемых между подцелями. Каждая подцель в цели интерпретируется как вызов соответствующей процедуры.

Таким образом, база данных, содержащая обратные Хорновские клаузы, может быть рассмотрена и декларативно (как логическая теория), и процедурно (как программа, выполняющая логические вычисления).

Схематично любую программу на Prolog можно представить так:

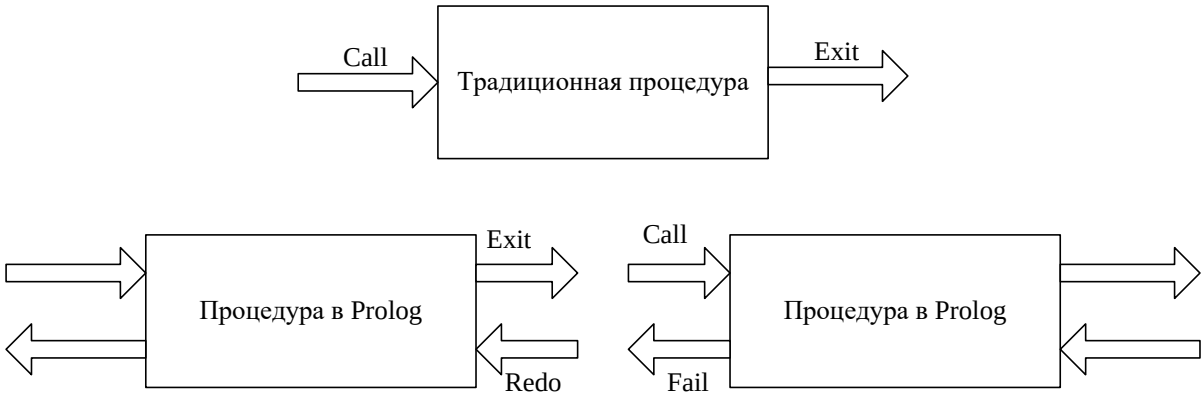
ПРОГРАММА = ПРАВИЛА + ФАКТЫ

ВЫПОЛНЕНИЕ = ЛОГИЧЕСКИЙ ВЫВОД

Одним из важных отличий Prolog от языков других парадигм является **недетерминизм**: в момент вызова не всегда заранее известно, какая конкретная клауза

будет использована. Кроме того, возможен повторный вход в уже выполненную процедуру.

Недетерминизм проявляется при неудачном завершении вычисления (**Fail**): запускается механизм возврата, который автоматически переходит к предыдущей точке выбора и пытается альтернативный путь (**Redo**).



Prolog в основном применяется для решения логических задач, а не для числовых вычислений. Одной из наиболее успешных областей его использования является разработка экспертных систем, которые способны самостоятельно решать сложные проблемы — например, выполнять автоматическое планирование, мониторинг, управление или диагностику неисправностей в больших технических комплексах.

С помощью этого языка удобно решать задачи, основанные на логических правилах, такие как поиск информации в базах данных, работа голосовых интерфейсов, автоматическое заполнение шаблонных структур.

Изначально Prolog был создан для задач обработки естественного языка. Он широко используется при сопоставлении шаблонов на деревьях синтаксического разбора, поскольку позволяет естественно описывать правила соответствия и обеспечивает механизм их эффективного выполнения.

C++ — это высокоуровневый язык программирования общего назначения, разработанный в 1983 году Бьёрном Страуструпом как расширение языка C. Он поддерживает императивное, объектно-ориентированное и обобщённое программирование, а также предоставляет возможности работы с памятью на низком уровне. Чаще всего C++ реализуется как компилируемый язык, и многие компании предлагают компиляторы, благодаря чему его можно использовать на различных платформах.

Базой для C++ служит язык C, работающий на более низком уровне, что позволяет получать ряд преимуществ в скорости, которые отсутствуют у большинства высокоуровневых языков:

– Если структуры в C++ не используют виртуальные функции, дополнительная память не выделяется, как это происходит, например, для объектов в Java.

– Возможно создание временных массивов и объектов в стеке без лишних затрат памяти, в то время как многие высокоуровневые языки вынуждают размещать объекты в куче (heap), управление которой сложнее и медленнее.

– C++ позволяет использовать массивы объектов, а не только массивы указателей на объекты, как в других языках. Это уменьшает количество указателей, экономит память и снижает уровень косвенной адресации.

– Локальные массивы (объявленные внутри функций) по умолчанию остаются неинициализированными, то есть их содержимое неопределенно. Такой подход может экономить время по сравнению с языками, где все элементы инициализируются нулями, как в Java.

– В C++ отсутствуют проверки границ переменных или индексов массивов, что также ускоряет выполнение программ.

Эти особенности дают программисту большую свободу в управлении памятью и системными ресурсами, что повышает скорость, но одновременно увеличивает риск ошибок. Поэтому многие считают, что работа на C++ менее «безопасна», чем на других языках.

В области искусственного интеллекта возможности языка имеют особое значение. C++ предоставляет удобные и эффективные структуры данных для реализации алгоритмов, многие популярные алгоритмы уже включены в стандартную библиотеку STL. Программист может управлять выполнением на низком уровне через указатели, что особенно полезно при разработке алгоритмов.

Благодаря этим преимуществам C++ часто выбирают для приложений с нейронными сетями и машинным обучением, где требуется интенсивная обработка больших массивов данных и использование статистических методов. Кроме того, многие программы ИИ работают на встроенных системах и устройствах, например на мобильных роботах или камерах для машинного зрения, где выбор языка ограничен C или C++.

К недостаткам C++ можно отнести необходимость писать больше «базового» кода (boilerplate) перед основной реализацией и относительную сложность изучения для начинающих из-за широких возможностей языка.

Высокая скорость выполнения делает C++ востребованным в проектах ИИ, где критично время работы — например, в поисковых системах и компьютерных играх. Многие библиотеки для других языков, включая Python, написаны на C++ или C именно из-за высокой производительности.

Python — это язык программирования, созданный нидерландским программистом Гвидо ван Россумом в 1991 году. Python является многопарадигменным языком: он полностью поддерживает объектно-ориентированное и структурное программирование, а многие его возможности позволяют использовать функциональный и аспектно-ориентированный подход, включая метапрограммирование и работу с метаобъектами. Вместо того чтобы вся функциональность была встроена в ядро, Python имеет широкие возможности для расширения, включая поддержку контрактного и логического программирования. Его модульная архитектура делает язык особенно популярным для создания программируемых интерфейсов к существующим приложениям.

Python является интерпретируемым языком, что означает, что код можно выполнять напрямую без предварительной компиляции в машинные инструкции. Интерпретируемые языки обычно позволяют быстрее создавать прототипы, хотя в

некоторых случаях они не используют оптимизации, доступные компилируемым языкам.

Python динамически типизирован: имя переменной связано с объектом во время выполнения, и одно и то же имя может последовательно ссылаться на объекты разных типов. При этом язык является строго типизированным: интерпретатор следит за типами объектов и запрещает операции, несовместимые с типом данных (например, сложение числа и строки), что помогает избежать ошибок.

Современные возможности Python включают использование графического процессора (GPU) для ускорения вычислений. Библиотеки, такие как CUDA Python и cuDNN, позволяют эффективно обрабатывать большие объёмы данных, что особенно важно для задач глубокого обучения, аналитики и инженерных приложений, приближая производительность Python к языкам вроде C++.

При разработке систем искусственного интеллекта часто важен не только сам алгоритм, но и проектирование подходов к его реализации. Python хорошо подходит для программного прототипирования — создания прототипов или неполных версий программ, позволяющих исследовать функциональность перед финальной реализацией.

Динамическая типизация позволяет писать универсальные функции, работающие с разными типами данных. Например, функция:

```
def max_val(a, b):
return a if a > b else b
```

может принимать числа, строки, списки, словари и другие объекты. В Python коллекции данных могут быть неоднородными, что облегчает упаковку и преобразование данных в структуры, подходящие для статически типизированных языков, таких как C++. Работа со структурами данных интуитивна и делает код более читаемым. Например, создание списка и проверка наличия элемента:

```
list = [1, 2, 3]
result = 2 in list
```

в C++ выглядело бы сложнее, с использованием итераторов и явного добавления элементов.

Python позволяет писать алгоритмы ИИ с меньшим объёмом кода — иногда до одной пятой по сравнению с другими языками. Интерпретация кода даёт возможность проверок во время разработки. Среди преимуществ также стоит отметить открытый исходный код и большое сообщество, обеспечивающее множество библиотек, расширяющих функциональность языка, включая ИИ.

Популярные библиотеки для ИИ на Python: TensorFlow (потоковое и дифференцируемое программирование), Keras (нейронные сети) и Scikit-learn (машинное обучение). Благодаря скриптовой природе, модульной архитектуре и простому синтаксису Python часто используется для обработки естественного языка.

Наиболее подходящие области применения Python — машинное и глубокое обучение, прогнозирование, управление рисками, медицинская диагностика и другие системы, где требуется обработка больших объёмов данных и гибкость в программировании.

Соображения при выборе подходящего языка программирования

При разработке систем искусственного интеллекта сообщество специалистов разделено во мнениях о том, какой язык является наиболее подходящим для создания программ и приложений. Часто программист выбирает язык, исходя из собственного комфорта при работе с ним, знаний и предыдущего опыта. Такой подход облегчает быстрое прототипирование и разработку, но в долгосрочной перспективе может оказаться не оптимальным для построения моделей, учитывающих конкретную задачу и объём данных, подлежащих обработке.

Компилируемые языки обычно обеспечивают более высокую общую производительность по сравнению с интерпретируемыми, так как при компиляции код преобразуется в набор машинных инструкций и сохраняется в виде исполняемого файла. В случае интерпретации код сохраняет исходную форму и преобразуется в инструкции только во время выполнения. Однако интерпретируемые языки предоставляют дополнительные возможности: например, программы могут изменять себя во время работы за счёт добавления или изменения функций. Также процесс разработки часто проще и быстрее, так как не требуется перекомпиляция для проверки даже небольших изменений.

Лёгкость использования языка зависит от множества факторов, один из которых — ограничения на тип данных. Динамическая типизация ускоряет разработку, но в языках со статической типизацией проще проводить рефакторинг и поддерживать код в долгосрочной перспективе.

Другой фактор — степень многословности (verbosity), необходимая для описания логики программы. Важным также является состояние сообщества разработчиков, поддерживающего язык: возможность обмена идеями, поиска решений проблем и обсуждения будущих улучшений.

В конечном счёте выбор языка для ИИ зависит от конкретных задач, сложности реализации, объёма данных, масштаба проекта, предыдущего опыта и навыков программиста.

Заключение

Искусственный интеллект — это междисциплинарная область, предоставляющая широкие возможности для исследований и практического применения. Разнообразие программных языков и их развитые особенности усложняют выбор подходящего языка для разработки интеллектуальных систем.

Из проведённого обзора можно сделать вывод, что для решения строго логических задач наиболее подходящим является Prolog, структура и синтаксис которого максимально приближены к естественной логике. Однако он недостаточно эффективен для выполнения вычислительных операций с числовыми объектами. В этом плане большое преимущество имеет C++, который обеспечивает высокую скорость и предоставляет возможности управления выполнением на низком уровне.

Python же продолжает набирать популярность благодаря лёгкости изучения и удобству использования, что сокращает время, необходимое для достижения конечных целей. Он способен выполнять разнообразные функции достаточно эффективно благодаря широкому набору библиотек.

При создании полноценной программы с элементами искусственного интеллекта часто используют несколько языков программирования. Цель заключается в том, чтобы максимально использовать возможности всех них и одновременно минимизировать их

основные недостатки. Часто, особенно в крупных проектах, основная структура реализуется на удобном или предпочитаемом языке высокого уровня, к которому через интерфейсы подключаются отдельные модули, написанные на разных языках в зависимости от выполняемых функций.

Adabiyotlar/Литература/References:

1. Russell, S., Norvig, P. *Artificial Intelligence: A Modern Approach*. 4th Edition, Pearson, 2020.
2. Luger, G. F. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. 7th Edition, Pearson, 2009.
3. Van Rossum, G., Drake, F. L. *Python Reference Manual*. Python Software Foundation, 2001.
4. Stroustrup, B. *The C++ Programming Language*. 4th Edition, Addison-Wesley, 2013.
5. Colmerauer, A., Roussel, P. *The Birth of Prolog. History of Programming Languages*, ACM Press, 1996.
6. McCarthy, J. *LISP 1.5 Programmer's Manual*, MIT Press, 1962.
7. Chollet, F. *Deep Learning with Python*. 2nd Edition, Manning, 2021.
8. Goodfellow, I., Bengio, Y., Courville, A. *Deep Learning*. MIT Press, 2016.
9. Kowalski, R. *Predicate Logic as a Programming Language*. IFIP Congress, 1974.

ISSN: 3030-3702 (Online)
САЙТ: https://techscience.uz

TECHSCIENCE.UZ

TEXNIKA FANLARINING DOLZARB MASALALARI

№ 12 (3)-2025

TOPICAL ISSUES OF TECHNICAL SCIENCES

TECHSCIENCE.UZ- TEXNIKA FANLARINING DOLZARB MASALALARI elektron jurnali 15.09.2023-yilda 130346-sonli guvohnoma bilan davlat ro‘yxatidan o‘tkazilgan.
Muassislar: “SCIENCEPROBLEMS TEAM” mas’uliyati cheklangan jamiyati;
Jizzax politexnika insituti.

TAHRIRIYAT MANZILI:
Toshkent shahri, Yakkasaroy tumani, Kichik Beshyog‘och ko‘chasi, 70/10-uy.
Elektron manzil:
scienceproblems.uz@gmail.com