ISSN 0234-0453 • INFORMATICS AND EDUCATION • 2026 • Volume 41 No 1

DOI: 10.32517/0234-0453-2026-41-1-37-48

О ВОЗМОЖНОСТЯХ ИСПОЛЬЗОВАНИЯ ГЛОБАЛЬНОГО ЯЗЫКА ПРИ ОБУЧЕНИИ ПРОГРАММИРОВАНИЮ

А. В. Желудков[1] ✉

[1] ***Московский финансово-юридический университет МФЮА, г. Москва, Россия***

✉ *zhantonv@gmail.com*

Аннотация

Существуют разнообразные методики для обучения созданию программных продуктов. Каждая из них предполагает использование различных инструментов. В статье выделяются две группы языков, используемые для обучения программированию: учебные и широкопрофильные. Учебные — это языки, которые специально разрабатывались для образовательных целей, например Пролог-Д. Они обладают простым и понятным синтаксисом, но имеют ограниченные возможности, и последующий переход к практической разработке может требовать дополнительных навыков. В то же время изучение широкопрофильных языков, таких как, например Java, изначально дает все необходимые прикладные навыки, но может быть слишком сложным для начинающих. В статье проводится сравнительный анализ достоинств и недостатков использования языков из данных групп при освоении основ разработки программного обеспечения.

Кратко описывается идея Глобального языка программирования, который служит промежуточным звеном при переводе текста программы между различными языками. Показывается возможность использования Глобального языка в качестве точки входа для обучения программированию. Обсуждаются преимущества данного подхода, в том числе: Глобальный язык имеет простой и лаконичный синтаксис из-за особенностей своего назначения; предоставляется способ наглядного сравнения реализации алгоритмов в различных парадигмах программирования; имеется возможность делать вставки в текст программы на различных языках, т. е. осуществлять плавный переход на требуемый широкопрофильный язык; присутствует возможность запускать полученную программу на различных устройствах путем конвертирования ее текста на поддерживаемый целевой язык.

В статье приводится пример задачи определения принадлежности точки многоугольнику, код решения которой с Глобального языка методом универсальных расширенных синтаксических деревьев автоматизированно переводится на Prolog, C, Java.

Обсуждаются достоинства использования в качестве начального именного Глобального языка, несмотря на то что его можно использовать только для перевода текстов программ, а начальным выбрать какой-либо другой язык, а именно: Глобальный язык специально разрабатывается с лаконичным и обобщенным синтаксисом; подобные знания помогут в будущем полноценно изучить работу метода универсальных расширенных синтаксических деревьев, что позволит, например, при создании собственного языка реализовать перевод на Глобальный язык и наоборот, тем самым давая возможность его быстрой интеграции с различными программными продуктами. Например, можно будет сразу подключать и использовать какие-либо готовые библиотеки.

Ключевые слова: Глобальный язык программирования, метод универсальных расширенных синтаксических деревьев, парадигма программирования, Prolog, Пролог-Д, Java, C, КуМир, Scratch.

Для цитирования:

Желудков А. В. О возможностях использования Глобального языка при обучении программированию. *Информатика и образование*. 2026;41(1):37–48. DOI: 10.32517/0234-0453-2026-41-1-37-48.

ON THE POSSIBILITIES OF USING THE GLOBAL LANGUAGE IN TEACHING PROGRAMMING

A. V. Zheludkov[1] ✉

[1] ***Moscow University of Finance and Law MFUA, Moscow, Russia***

✉ *zhantonv@gmail.com*

Abstract

There are various methods for teaching software development. Each of them involves the use of various tools. The article distinguishes two groups of languages used for teaching programming: educational and general-purpose. Educational languages are languages that were specially developed for educational purposes, for example, Prolog-D. They have a simple and clear syntax, but are limited in capabilities, and subsequent transition to practical development may require additional skills. At the same time, learning general-purpose languages, such as Java, initially provides all the necessary applied skills but may be too difficult for beginners. The article provides a comparative analysis of the advantages and disadvantages of using languages from these groups when mastering the fundamentals of software development.

The article briefly describes the idea of a Global programming language, which serves as an intermediate link in translating program text between different languages. The possibility of using a Global language as an entry point for teaching programming

is demonstrated. The advantages of this approach are discussed, including: the Global language has a simple and concise syntax due to the specifics of its purpose; a method for visually comparing algorithm implementations in different programming paradigms is provided; it is possible to insert program text in different languages, i. e., to seamlessly transition to the required general-purpose language; and it is possible to run the resulting program on various devices by converting its text to a supported target language.

The article provides an example of the problem of determining whether a point belongs to a polygon, the solution code for which is automatically translated from the Global Language into Prolog, C, and Java using the method of universal extended syntax trees.

The advantages of using the Global Language as the initial named language are discussed, despite the fact that it can only be used for translating program texts, while another language can be selected as the initial language. Specifically, the Global Language is specifically designed with a laconic and generalized syntax. Such knowledge will help in the future to fully understand the operation of the method of universal extended syntax trees, which will allow, for example, when creating a custom language, to implement translation into the Global Language and vice versa, thereby enabling its rapid integration with various software products. For example, it will be possible to immediately connect and use any ready-made libraries.

Keywords: Global programming language, method of universal extended syntax trees, programming paradigm, Prolog, Prolog-D, Java, C, KuMir, Scratch.

For citation:

Zheludkov A. V. On the possibilities of using the Global language in teaching programming. *Informatics and Education.* 2026;41(1): 37–48. (In Russian.) DOI: 10.32517/0234-0453-2026-41-1-37-48.

1. Введение

Существует большое количество подходов к обучению разработке программного обеспечения [1]. Важную роль в каждом из них играет выбор языка программирования [2]. При решении данной задачи следует учитывать набор факторов, которые помогут сделать процесс получения знаний более эффективным и увлекательным, а вырабатываемые навыки актуальными. Ниже перечислены некоторые из них:

- Возраст и опыт учеников [3]. В зависимости от того, к какой возрастной группе относятся обучаемые, будут сильно различаться используемые подходы и инструменты. Для детей младшего школьного возраста больше подойдут графические языки, например Scratch, а для взрослой аудитории можно задействовать более строгие языки, например Пролог-Д [4].
- Простота и понятность синтаксиса [5]. Для начинающих критически важно, чтобы синтаксис языка был простым и интуитивно понятным. Сложные конструкции или многочисленные правила могут как отпугнуть новичков, так и просто сделать процесс сложным и неинтересным.
- Поддержка сообществом и наличие учебных материалов [6]. Хороший начальный язык должен иметь большое сообщество разработчиков, множество ресурсов и документации. Это позволит легко находить ответы на возникающие вопросы и пользоваться множеством учебных пособий, видеоуроков и примеров кода.
- Интерактивность и поддержка учебных инструментов [7]. Интерактивные среды для написания кода помогают новичкам сразу видеть результаты своих действий, что улучшает восприятие. Также важно наличие инструментов, которые поддерживают автодополнение и отладку, чтобы студентам было проще разбираться с ошибками.
- Простота установки и запуска. Чем меньше усилий требуется для того, чтобы начать программировать, тем лучше. Если, для того чтобы начать работу, нужно устанавливать много программного обеспечения или разбираться с конфигурацией среды, это может стать преградой для новичков.
- Поддержка фундаментальных концепций программирования [8]. Язык должен помогать изучать основные принципы программирования: переменные, условные операторы, циклы, функции, основные структуры данных и т. п.
- Безопасность и избегание сложных ошибок. Для новичков важно избегать языков, в которых легко сделать неочевидные критические ошибки, такие как ошибки с указателями в C/C++. Лучше задействовать языки с автоматическим управлением памятью и сборкой мусора, а также не требующие глубокого понимания среды, в которой запускается программа.
- Перспективы дальнейшего использования [9]. Язык должен давать навыки, которые впоследствии помогут в дальнейшем обучении или сразу будут востребованными на рынке труда. Так как данный критерий сильно зависит от того, в каком направлении развиваются ученики (написание ПО, драйверов, проведение научных вычислений и т. п.), то учебный язык должен быть универсальным с точки зрения базовых навыков написания и отладки кода.
- Сложность и стоимость экосистемы. Некоторые языки обладают сложными экосистемами и требуют глубоких знаний для настройки среды разработки, использования библиотек и работы с инструментами сборки и дорогих IDE для удобной работы. Для обучения важно выбирать язык с простой экосистемой, чтобы студенты могли сосредоточиться на программировании, а не на настройке и покупке громоздкого вспомогательного ПО.
- Платформенная независимость. Желательно, чтобы язык был кроссплатформенным и позволял работать в разных операционных системах (Windows, macOS, Linux). Это позволит использовать язык в разных контекстах, не привязываясь к конкретной платформе.

ISSN 0234-0453 • INFORMATICS AND EDUCATION • 2026 • Volume 41 No 1

Проблема поиска оптимального языка программирования для обучения сложна и актуальна, так как существует большое число разнообразных языков, применяемых для решения различных задач[1]. В представленной статье подобная задача обобщается тем, что все языки можно разделить на две группы:

- учебные языки, созданные специально для комфортного и постепенного погружения учеников в процесс. В качестве примера подобных языков можно привести КуМир [10], Пролог-Д [4] или Scratch [11];
- языки широкого профиля, созданные для решения прикладных задач, но которые все равно можно использовать в качестве учебных. Примеры: Пролог [12], C [13], Java [14].

В данной статье демонстрируются достоинства и недостатки использования данных групп языков в качестве начальных для обучения программированию. Описывается возможность задействовать в качестве точки входа для освоения основ создания ПО Глобальный язык [15]. Показываются все преимущества данного подхода на примере программы определения принадлежности точки многоугольнику на плоскости, код которой с помощью метода универсальных расширенных синтаксических деревьев (УРСД) [16] переводится с Глобального языка на Java, C, Prolog, тем самым наглядно демонстрируются особенности различных парадигм программирования.

В более ранних работах вместо названия «метод универсальных расширенных синтаксических деревьев» применяется название «метод универсальных КСД» (КСД — конкретное синтаксическое дерево) и дается определение универсального КСД, однако для лучшего отражения сути метода было принято решение в дальнейшем использовать новое название.

2. Материалы и методы

2.1. Учебные языки программирования

Учебные языки программирования — это языки, специально созданные для того, чтобы обучать концепциям создания ПО. Их главная цель — упростить изучение базовых понятий и структур программирования. Они отличаются от промышленных языков тем, что в них особое внимание уделено понятности, простоте и фокусировке на ключевых понятиях: переменная, функция, цикл, рекурсия, базовые структуры.

Вот некоторые важные особенности учебных языков программирования:

- Простота синтаксиса. Учебные языки обычно имеют упрощенный синтаксис, чтобы сосредоточиться на логике программирования, а не на запоминании сложных правил. Синтаксис стараются сделать максимально естественным и приближенным к человеческому языку. Например, в языке КуМир используется простой русскоязычный синтаксис, что упрощает его изучение для русскоговорящей аудитории.
- Минимизация ошибок. Учебные языки часто имеют системы, снижающие вероятность синтаксических и логических ошибок у новичков. Они могут быть менее требовательны к строгому форматированию кода или автоматически исправлять мелкие ошибки. Например, в языке Scratch элементы кода представляются в виде блоков, которые можно соединять, как пазлы, что предотвращает ошибки вроде забытых скобок или синтаксических неточностей.
- Абстрагирование от сложных деталей. Чтобы не отвлекать новичков на сложные технические детали, учебные языки часто скрывают низкоуровневые аспекты программирования (например, управление памятью, указатели и т. п.). Это помогает сосредоточиться на изучении фундаментальных концепций, таких как циклы, условия и функции. Например, язык КуМир позволяет писать простые программы по вышеуказанному принципу.
- Интерактивность и визуализация. Учебные языки часто поддерживают интерактивные элементы и визуализацию кода. Это помогает учащимся сразу видеть результат своей работы, что усиливает мотивацию и делает процесс обучения более наглядным. Например, в Scratch можно создавать игры и анимации, наблюдая за изменениями в реальном времени при изменении программы.
- Постепенное введение сложных концепций. Учебные языки вводят новые концепции шаг за шагом. Это помогает не перегружать новичков на начальных этапах сложными темами вроде многопоточности или объектно-ориентированного программирования.
- Поддержка игровой механики. Для повышения мотивации некоторые учебные языки интегрируют игровые элементы, что делает обучение программированию более увлекательным и интерактивным.
- Легкость развертывания. Учебные языки часто проектируются так, чтобы их можно было легко начать использовать без сложных установок и конфигураций. Это позволяет учащимся быстро приступить к работе. Например, для языка Пролог-Д требуется лишь запуск среды исполнения кода [17].
- Ориентация на начальные этапы обучения. Учебные языки обычно не предназначены для разработки коммерческого программного обеспечения. Их основная цель — обучение базовым принципам. Как только учащийся освоит основы, он обычно переходит на более сложные языки.

Учебные языки программирования фокусируются на доступности и удобстве, чтобы помочь начинающим программистам быстрее освоить основы

[1] Согласно индексу TIOBE: https://www.tiobe.com/tiobe-index/programminglanguages_definition/

и начать понимать принципы программирования, однако, из-за этого обладают рядом недостатков:

- Ограниченная применимость. Учебные языки редко применяются за пределами образовательной среды. Навыки, полученные при работе с ними, часто не переносятся напрямую на реальные задачи разработки, что может ограничить их практическую ценность. Например, Scratch или КуМир редко используются в промышленной разработке, поэтому студенты могут столкнуться с трудностями при переходе к языкам широкого профиля.
- Недостаток глубины. Учебные языки обычно скрывают сложные аспекты программирования (например, управление памятью, многопоточность, типизация данных и т. д.), что ограничивает возможность изучения более продвинутых тем. В результате учащиеся могут не получить полного понимания того, как работают программы на более низком уровне. Например, в Scratch нельзя научиться основам работы с памятью или сложными структурами данных, что критично для разработки на языках широкого профиля, таких как C++.
- Переход к реальным языкам может быть сложным. После обучения на учебных языках студентам может быть сложно перейти к промышленным языкам, так как учебные языки сильно абстрагируют многие аспекты программирования. Столкновение с реальными языками, требующими строгого соблюдения синтаксиса и управления ресурсами, может вызывать затруднения. Например, учащиеся, привыкшие к визуальным блокам Scratch, могут испытывать трудности при переходе на текстовые языки вроде C или Java.
- Ограниченные возможности для масштабных проектов. Учебные языки обычно не подходят для разработки больших и сложных проектов. Они созданы для простых упражнений и небольших программ, что ограничивает их использование для более серьезных задач. Например, на языках вроде КуМир невозможно создать полноценное веб-приложение или мобильное приложение.
- Отсутствие опыта работы с реальными инструментами. Учебные языки не знакомят студентов с профессиональными инструментами разработки, такими как системы контроля версий (Git), интегрированные среды разработки (IDE) и тестирование. Это может стать препятствием на пути к профессиональной карьере. Например, студенты, изучающие только Scratch, могут столкнуться со сложностями при работе с Git или при использовании таких IDE, как IntelliJ или Visual Studio.
- Избыточная абстракция. Слишком высокая степень абстракции может привести к тому, что учащиеся не будут понимать, как на самом деле работает компьютер и как программы взаимодействуют с аппаратным обеспечением. Это особенно важно для тех, кто хочет в будущем заниматься системным программированием или оптимизацией производительности.

Таким образом, можно сделать вывод о том, что выбор учебных языков программирования в качестве точки знакомства с разработкой ПО имеет свои плюсы и минусы, но следует учитывать, что переход между ними и тем более к широкопрофильным языкам сложен и требует дополнительных знаний об устройстве вычислителя и парадигмах программирования.

2.2. Языки программирования широкого профиля

Обучение программированию на промышленных языках, таких как Prolog, Java, C и других, имеет как достоинства, так и недостатки по сравнению с использованием специально разработанных учебных языков.

Рассмотрим сначала достоинства:

- Реальная практическая ценность. Учащиеся сразу осваивают инструменты, востребованные в индустрии. Знание промышленных языков повышает их шансы на трудоустройство и участие в реальных проектах. Например, знание Java открывает двери к разработке веб-приложений, анализу данных или мобильных приложений.
- Глубокое понимание программирования. Промышленные языки требуют изучения как базовых, так и продвинутых аспектов программирования. Это помогает студентам лучше понять структуры данных, алгоритмы, управление памятью и другие фундаментальные темы, которые могут быть скрыты в учебных языках.
- Широкая экосистема и поддержка. Промышленные языки поддерживаются огромным сообществом, для них существует множество библиотек, инструментов и фреймворков. Это дает студентам доступ к мощным инструментам разработки и готовым решениям, что позволяет им быстрее реализовывать проекты. Например, Java имеет богатую экосистему библиотек, которые помогают в таких областях, как разработка веб-приложений и автоматизация задач.
- Подготовка к реальным вызовам. Промышленные языки предъявляют требования к оптимизации, управлению памятью и обработке ошибок, что помогает студентам быть лучше подготовленными к задачам в реальной разработке. Они учатся решать реальные проблемы и оптимизировать код с учетом производительности.
- Долгосрочная перспектива. Промышленные языки развиваются и обновляются, что обеспечивает актуальность знаний на протяжении долгого времени. Начав учиться с подобных языков, учащиеся смогут поддерживать свои

навыки актуальными и развивать их на основе современных технологий.

Теперь опишем недостатки обучения на промышленных языках:

- Сложность для новичков. Промышленные языки зачастую сложнее в освоении для новичков. Они требуют понимания синтаксических деталей, управления памятью, обработки ошибок и других аспектов, которые могут отпугнуть начинающих программистов. Например, в языке C нужно управлять указателями и памятью, что может быть запутанным для новичков.
- Много «лишних» деталей на начальных этапах. В промышленных языках много технических аспектов, которые могут отвлекать от базовых концепций программирования. Например, новички могут тратить много времени на изучение нюансов работы с объектами, интерфейсами или типами данных, что усложняет процесс обучения. В Java даже простая программа требует писать классы, методы и часто запутанные для новичков конструкции, такие как наследование и интерфейсы.
- Крутая кривая обучения. Промышленные языки часто имеют более сложные синтаксические и семантические конструкции, что делает начальный этап обучения трудным и осложняет понимание базовых принципов программирования.
- Меньшая интерактивность. Промышленные языки не всегда предоставляют инструменты для визуализации кода или интерактивных экспериментов с программой, что может снижать мотивацию у студентов, особенно на ранних этапах обучения. Например, в отличие от таких визуальных языков, как Scratch, программы на Java или C нужно писать и запускать в текстовом виде, что может показаться менее наглядным для начинающих.
- Сложности с установкой и настройкой среды. Промышленные языки часто требуют установки сложных интегрированных сред разработки, настройки зависимостей и окружений. Это может стать препятствием на начальном этапе, особенно для тех, кто не знаком с разработкой программного обеспечения. Например, для начала работы с Java требуется установка JDK и настройка окружения.
- Накопление ошибок. Промышленные языки могут быть менее «терпимыми» к ошибкам. Малейшая ошибка в коде (например, забытая запятая или точка с запятой в C) может приводить к тому, что программа не скомпилируется, что увеличивает фрустрацию у начинающих [18].

Подводя итог, можно сделать вывод, что обучение на широкопрофильных языках возможно, однако, может быть непросто для начинающих и при желании перейти на какой-либо другой язык, особенно реализующий другую парадигму, могут возникнуть препятствия в необходимости начинать все с начала.

2.3. Глобальный язык программирования

Таким образом, видно, что индустрии не хватает языка, который являлся бы общей точкой входа для получения навыков написания ПО и позволял бы быстро демонстрировать, как изменяется программный продукт, при написании его на другом языке и в другой парадигме. Подобное сравнение в свою очередь формирует способность в будущем самостоятельно выбирать подход к написанию программ и развивает алгоритмическое и аналитическое мышление студентов [19]. Существуют различные расширения [20] и надстройки [21], позволяющие приблизиться к решению данной задачи, но не дающие комплексной возможности выполнять вышеописанные операции сравнения. На роль языка, при использовании которого это становится возможным, подходит Глобальный язык программирования, представляющий из себя промежуточное звено для трансформации кода между языками с помощью метода универсальных расширенных синтаксических деревьев (УРСД).

Подробно ознакомиться с методом можно в работе [16]. Пример синтаксиса языка представлен в работе [15]. Блок-схема алгоритма, реализующего метод УРСД, представлена на рисунке 1.

Основная идея заключается в том, что разрабатываемый Глобальный язык позволит переводить код между языками и его синтаксис будет специально адаптирован под метод универсальных РСД. То есть вместо необходимости писать конверторы для существующих языков можно будет автоматизировать их перевод в Глобальный и наоборот.

Таким образом, с точки зрения обучения программированию выделяется ряд преимуществ по сравнению с учебными и широкопрофильными языками, а именно:

- Зная Глобальный язык, можно перевести его код в другой и продолжить обучение в различных направлениях.
- Глобальный язык программирования позволяет наглядно изучить различные парадигмы, так как он дает возможность перевести код в языки, их реализующие.
- Программы на Глобальном языке программирования запускаются на различных устройствах, так как для этого они конвертируются в язык, поддерживаемый устройством.
- Мотивация изучения языка высока, так как он стирает границы между различными языками и позволяет делать вставки, т. е. писать части программы на различных языках. Таким образом, полученные знания и навыки можно применить в большом количестве проектов.
- Синтаксис Глобального языка максимально лаконичен и прост, что облегчает его понимание и освоение, так как этого требует сам смысл Глобального языка — быть прослойкой для перевода кода между различными языками.

ISSN 0234-0453 • ИНФОРМАТИКА И ОБРАЗОВАНИЕ • 2026 • Том 41 № 1

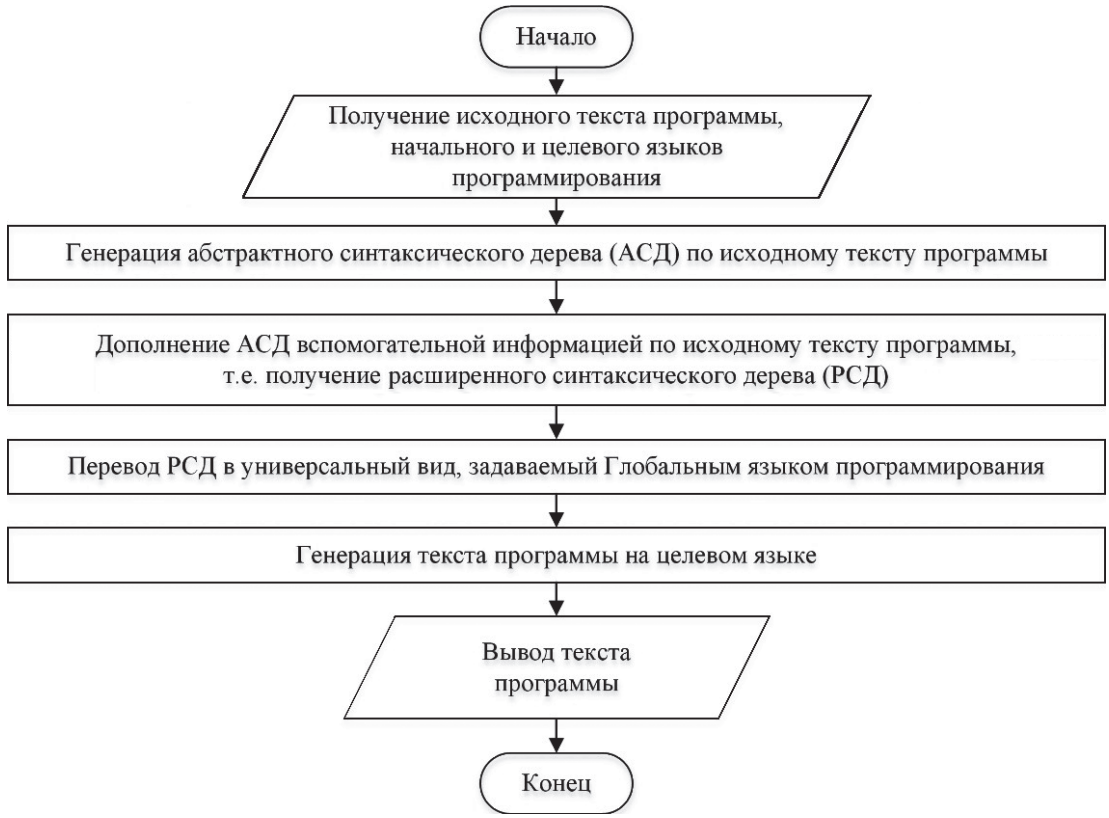

Рис. 1. Блок-схема алгоритма, реализующего метод УРСД
Fig. 1. Block diagram of the algorithm implementing the UEST method

Важно подчеркнуть, что Глобальный язык является возможной точкой входа для обучения программированию, но он не заменяет собой учебные языки, а дополняет их. Таким образом, обучение можно начать с Глобального языка, чтобы потом плавно перейти на другой — учебный или широкопрофильный — в зависимости от цели курса. Отдельно следует подчеркнуть, что Глобальный язык позволяет быстро переключаться между языками и проводить сравнения, наглядно показывая разницу в них и причины возникновения различных языков и парадигм программирования.

3. Результаты

Для наглядной демонстрации достоинств использования Глобального языка как точки входа для обучения программированию на нем была реализована программа определения принадлежности точки многоугольнику на плоскости способом PNPOLY[1]. Из проверяемой точки в произвольном направлении проводится луч. Затем в цикле выполняется подсчет количества его пересечений с ребрами исследуемого многоугольника. Если полученное число нечетное, то исследуемая точка считается расположенной внутри многоугольника. В противном случае — снаружи. Этот вывод базируется на соображении, что если бы исследуемая точка двигалась по лучу, то при каждом пересечении она бы меняла свое положение относительно принадлежности многоугольнику, оказываясь то снаружи, то внутри него. Для простоты достаточно выпустить луч, параллельный оси абсцисс, в направлении увеличения значений на ней.

Данный метод известен как правило «четное-нечетное» или алгоритм подсчета пересечений. Однако возможны трудности, если луч проходит через вершину многоугольника. В таком случае предлагается обходной подход: считать, что вершины находятся чуть выше или ниже линии луча, чтобы избежать таких пересечений. Таким образом, пересечение засчитывается, если один конец стороны лежит выше луча, а другой — ниже или на линии луча.

В качестве многоугольника выберем фигуру в форме звезды (рис. 2).

В качестве исследуемой точки возьмем начало координат.

Сначала реализуем программу на Глобальном языке программирования. Ее код представлен в листинге 1.

Затем с помощью метода универсальных РСД автоматически переведем код на язык C. Результат представлен в листинге 2.

Далее аналогичным способом переведем программу из процедурной в объектно-ориентированную

[1] PNPOLY — Point Inclusion in Polygon Test — W. Randolph Franklin (WRF). https://wrfranklin.org/Research/Short_Notes/pnpoly.html

ISSN 0234-0453 • INFORMATICS AND EDUCATION • 2026 • Volume 41 No 1

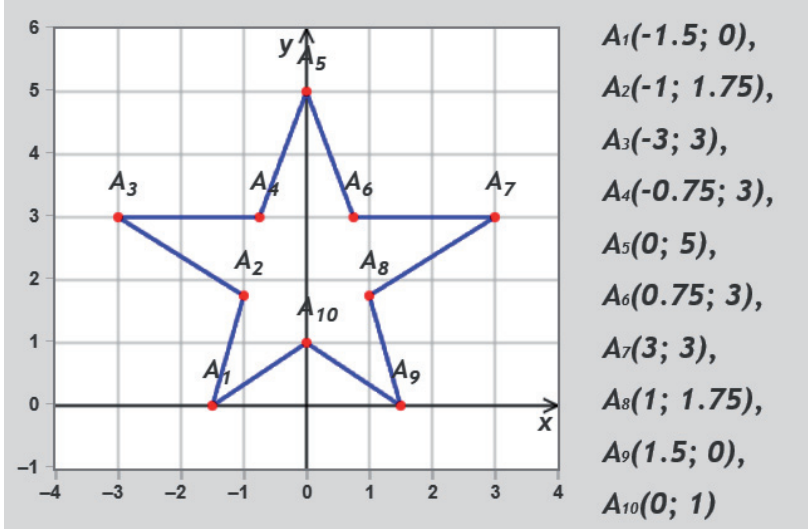

Рис. 2. Многоугольник, относительно которого проводится исследование принадлежности точки
Fig. 2. A polygon relative to which a point's belonging is investigated

```
начало

функция contains (xTest: дробь, yTest: дробь, n: число, xArray: [дробь], yArray: [дробь]) ->
число {
  пусть i: число = 0
  пусть j: число = n - 1
  пусть result: число = -1
  пока (i < n) {
    Если (((yArray[i] > yTest) != (yArray[j] > yTest)) && (xTest < ((xArray[j] - xArray[i]) *
    (yTest - yArray[i]) / (yArray[j] - yArray[i]) + xArray[i]))) {
      result = -1 * result
    }
    j = i
    i = i + 1
  }
  вернуть result
}

пусть xArray: [дробь] = [-1.5, -1.0, -3.0, -0.75, 0.0, 0.75, 3.0, 1.0, 1.5, 0.0]
пусть yArray: [дробь] = [0.0, 1.75, 3.0, 3.0, 5.0, 3.0, 3.0, 1.75, 0.0, 1.0]
пусть n: число = 10
пусть res: число = contains(xTest:0.0, yTest:0.0, n:n, xArray:xArray, yArray:yArray)
вывод(формат:"Res: %d", параметры:[res])

конец
```

Листинг 1. Программа определения принадлежности точки многоугольнику на Глобальном языке программирования
Listing 1. A program for determining whether a point belongs to polygon in Global programming language

парадигму на язык Java. Получившийся код представлен в листинге 3.

Затем в листинге 4 покажем, как будет выглядеть автоматизированный прямой перевод данной программы на язык логического программирования SWI-Prolog[1].

Очевидно, что получившийся код выглядит громоздким, однако его можно переписать вручную в соответствии с канонами логической парадигмы. Пример на языке Пролог-Д[2] приведен в листинге 5.

[1] Официальная страница языка SWI-Prolog. https://www.swi-prolog.org/

[2] Скачать среду исполнения кода на языке Пролог-Д для MacOS можно по ссылке: http:/cappa.csu.ru/media/filebrowser/prolog-d/prologformac.zip; для Windows — по ссылке: https://cappa.csu.ru/media/filebrowser/prolog-d/prolog-d.zip

```
#include <stdio.h>

int contains (double xTest, double yTest, int n, double * xArray, double * yArray) {
  int i = 0;
  int j = n - 1;
  int result = -1;
  while (i < n) {
    if (((yArray[i] > yTest) != (yArray[j] > yTest)) && (xTest < ((xArray[j] - xArray[i]) *
    (yTest - yArray[i]) / (yArray[j] - yArray[i]) + xArray[i]))) {
      result = -1 * result;
    }
    j = i;
    i = i + 1;
  }
  return result;
}

int main() {
  double xArray[] = {-1.5, -1.0, -3.0, -0.75, 0.0, 0.75, 3.0, 1.0, 1.5, 0.0};
  double yArray[] = {0.0, 1.75, 3.0, 3.0, 5.0, 3.0, 3.0, 1.75, 0.0, 1.0};
  int n = 10;
  int res = contains(0.0, 0.0, n, xArray, yArray);
  printf("Res: %d", res);
  return 0;
}
```

Листинг 2. Программа определения принадлежности точки многоугольнику на языке C
Listing 2. A program for determining whether a point belongs to a polygon in C

```
class LogicClass {
  private int contains (double xTest, double yTest, int n, double[] xArray,
double[] yArray) {
    int i = 0;
    int j = n - 1;
    int result = -1;
    while (i < n) {
      if (((yArray[i] > yTest) != (yArray[j] > yTest)) && (xTest < ((xArray[j] -
          xArray[i]) * (yTest - yArray[i]) / (yArray[j] - yArray[i]) + xArray[i]))) {
        result = -1 * result;
      }
      j = i;
      i = i + 1;
    }
    return result;
  }

  public void main() {
    double[] xArray = {-1.5, -1.0, -3.0, -0.75, 0.0, 0.75, 3.0, 1.0, 1.5, 0.0};
    double[] yArray = {0.0, 1.75, 3.0, 3.0, 5.0, 3.0, 3.0, 1.75, 0.0, 1.0};
    int n = 10;
    int res = contains(0.0, 0.0, n, xArray, yArray);
    System.out.printf("Res: %d", res);
  }
}

class Main {

  public static void main(String[] args) {
    LogicClass logic = new LogicClass();
    logic.main();
  }
}
```

Листинг 3. Программа определения принадлежности точки многоугольнику на языке Java
Listing 3. A program for determining whether a point belongs to a polygon in Java

ISSN 0234-0453 • INFORMATICS AND EDUCATION • 2026 • Volume 41 No 1

```
contains(XTest, YTest, N, XArray, YArray, Результат) :-
  I is 0,
  J is N - 1,
  Result is -1,
  loop_0(XTest, XTest_1, YArray, YArray_1, XArray, XArray_1, N, N_1, I, I_2, J, J_2, YTest,
  YTest_1, Result, Result_2),
  Результат is Result_2.

loop_0(XTest, Результат_XTest, YArray, Результат_YArray, XArray, Результат_XArray, N,
Результат_N, I, Результат_I, J, Результат_J, YTest, Результат_YTest, Result, Результат_Result) :-
  I < N,
  ((Индекс_0 is I,
    nth0(Индекс_0, YArray, YArray_at_Индекс_0),
    set_boolean((YArray_at_Индекс_0 > YTest), Boolean_0),
    Индекс_1 is J,
    nth0(Индекс_1, YArray, YArray_at_Индекс_1),
    set_boolean((YArray_at_Индекс_1 > YTest), Boolean_1),
    Индекс_2 is J,
    nth0(Индекс_2, XArray, XArray_at_Индекс_2),
    Индекс_3 is I,
    nth0(Индекс_3, XArray, XArray_at_Индекс_3),
    Индекс_4 is I,
    nth0(Индекс_4, YArray, YArray_at_Индекс_4),
    Индекс_5 is J,
    nth0(Индекс_5, YArray, YArray_at_Индекс_5),
    Индекс_6 is I,
    nth0(Индекс_6, YArray, YArray_at_Индекс_6),
    Индекс_7 is I,
    nth0(Индекс_7, XArray, XArray_at_Индекс_7),
    ((Boolean_0 =\= Boolean_1),
    (XTest < ((XArray_at_Индекс_2 - XArray_at_Индекс_3) * (YTest - YArray_at_Индекс_4) /
    (YArray_at_Индекс_5 - YArray_at_Индекс_6) + XArray_at_Индекс_7)))) ->
    Result_1 is -1 * Result
  ;
    Result_1 is Result
  ),
  J_1 is I,
  I_1 is I + 1,
  loop_0(XTest, Результат_XTest, YArray, Результат_YArray, XArray, Результат_XArray, N,
  Результат_N, I_1, Результат_I, J_1, Результат_J, YTest, Результат_YTest, Result_1,
  Результат_Result).
loop_0(XTest, XTest, YArray, YArray, XArray, XArray, N, N, I, I, J, J, YTest, YTest, Result,
Result) :-
  not(I < N).

main :-
  XArray = [-1.5, -1.0, -3.0, -0.75, 0.0, 0.75, 3.0, 1.0, 1.5, 0.0],
  YArray = [0.0, 1.75, 3.0, 3.0, 5.0, 3.0, 3.0, 1.75, 0.0, 1.0],
  N is 10,
  contains(0.0, 0.0, N, XArray, YArray, Результат_contains_0),
  Res is Результат_contains_0,
  format("Res: ~d", [Res]), !.

set_boolean(A, R) :- A -> R = 1; R = 0.
```

Листинг 4. Программа определения принадлежности точки многоугольнику на языке SWI-Prolog
Listing 4. A program for determining whether a point belongs to a polygon in SWI-Prolog

```
add_tail([],X,[X]).
add_tail([H|T],X,[H|L]) :- add_tail(T,X,L).
copy_first_to_last([H|T],[H|L]) :- add_tail(T,H,L).

check(F, S, X, Y) :- ИЛИ(checkA(F, S, X, Y), checkB(F, S, X, Y)), checkC(F, S, X, Y).
checkA([F , L], [Q , Z], X, Y) :- НЕ(МЕНЬШЕ(L, Y)),
НЕ(БОЛЬШЕ(Z, Y)),
НЕ(РАВНО(L, Z)).
checkB([F , L], [Q , Z], X, Y) :- НЕ(БОЛЬШЕ(L, Y)),
НЕ(МЕНЬШЕ(Z, Y)),
НЕ(РАВНО(L, Z)).
checkC([F , L], [Q , Z], X, Y) :- МЕНЬШЕ(X, # (Q - F) * (Y - L) / (Z - L) + F #).

contains(X, Y, [F | L], R) :- contains(X, Y, F, L, -1, R).

contains(X, Y, F, L, T, R) :- ИЛИ(containsA(X, Y, F, L, T, R), containsB(X, Y, F, L, T, R)).
containsA(X, Y, F, [S | L], T, R) :- check(F, S, X, Y), contains(X, Y, S, L, # -1 * T #, R).
containsB(X, Y, F, [S | L], T, R) :- contains(X, Y, S, L, T, R).
contains(_, _, _, [], R, R).

main :- main(0, 0, [[-1.5, 0], [-1.0, 1.75], [-3.0, 3.0], [-0.75, 3.0], [0, 5.0], [0.75, 3.0],
[3.0, 3.0], [1.0, 1.75], [1.5, 0], [0, 1.0]], R), ВЫВОД("Res: ", R).

main(X, Y, L, R) :- copy_first_to_last(L, T), contains(X, Y, T, R).
?main.
```

Листинг 5. Программа определения принадлежности точки многоугольнику на языке Пролог-Д
Listing 5. A program for determining whether a point belongs to a polygon in Prolog-D

4. Обсуждение и выводы

Как видно из приведенных выше примеров, Глобальный язык программирования может служить точкой входа при обучении написанию кода, так как:

1. Синтаксис Глобального языка прост, лаконичен и интуитивно понятен для русскоязычных пользователей.
2. При помощи автоматического преобразования кода на широкопрофильные языки даже на представленном компактном примере можно сразу наглядно увидеть, чем процедурная парадигма отличается от объектно-ориентированной и логической, а именно:
   2.1. В процедурной парадигме код составляется из функций подобно слоеному пирогу, а при объектно-ориентированном подходе методы изолируются в классы с разными уровнями доступа, что хотя и увеличивает количество строк, но делает работу с программой более понятной и наглядной. Можно сразу запустить ПО и провести исследование в разнице производительности.
   2.2. При прямом автоматизированном переводе предложенной программы на Swi-prolog получается тяжеловесный код, так как логическая парадигма служит для других целей и создана не для описания алгоритма получения ответа как такового, а для формулирования критериев проверки результата с последующим автоматическим выводом ответа по ним и заданному запросу. Если же переписать программу вручную, в соответствии с заданными принципами, то эти отличия сразу становятся понятны. Однако следует отметить, что можно комбинировать подобные подходы и использовать автоматизированный перевод с помощью Глобального языка для задействования готовых библиотек, реализованных на других языках, например при разработке на Prolog. То есть Глобальный язык программирования позволяет писать каждый из блоков программы на наиболее оптимальных для этого языках и сводить результат к целевому языку.
3. Глобальный язык программирования позволяет быстро и удобно изучить особенности других языков, например:
   3.1. Работа с массивами в C идет через указатели, тогда как в Java через ссылки.
   3.2. Схожесть названия базовых типов в Java и C.
   3.3. Отсутствие возможности перезаписывать значение переменной в Swi-prolog, что подводит к понятию унификации.
4. На Глобальном языке программирования можно реализовать плавный переход на другие языки путем осуществления вставок кода, тем самым позволяя упростить и сгладить данный процесс. Например, можно часть программы

написать на уже известном Глобальном языке с последующим автоматизированным переводом на C, а часть реализовать вручную с использованием указателей.

5. Программы, реализованные на Глобальном языке, можно запускать на различных устройствах путем перевода их текста в язык, поддерживаемый требуемым оборудованием.

При этом отдельно следует еще раз уточнить, что Глобальный язык не заменяет собой ни учебные, ни широкопрофильные языки. Он может служить точкой входа, изучение которой дополняет их и позволяет затем перейти к освоению языка, который наиболее актуален в зависимости от потребностей учащегося. Также он стирает границы между какими-либо языками, что стимулирует их развитие.

Еще один момент, на котором стоит остановиться отдельно, — это ответ на вопрос: зачем изучать и использовать для обучения программированию синтаксис именно Глобального языка, когда можно учиться на другом и получать на первый взгляд аналогичные результаты, точно так же автоматизированно переводя программы методом универсальных РСД? Ответ затрагивает сразу несколько аспектов:

- Каждый из существующих языков создавался для решения определенных задач, и поэтому их синтаксис проектировался для удобства написания кода именно под эти задачи. Глобальный язык разрабатывается для удобства перевода кода между различными языками, поэтому его синтаксис должен быть лаконичным и обобщенным.
- При знании синтаксиса Глобального языка можно будет более грамотно понимать возможности данного инструмента, впоследствии подключиться к работе по переводу кода на него и с него для других языков или даже разработать свой собственный язык, в котором можно будет использовать готовые вычислительные библиотеки, например из Java. То есть изучение синтаксиса Глобального языка программирования создает задел для будущего развития обучаемого.

Список источников / References

1. *Perugini S.* Emerging languages: An alternative approach to teaching programming languages. *Journal of Functional Programming*. 2019;29:e13. DOI: 10.1017/S095679681900011X.

2. *Паволоцкий А. В., Королев Д. А., Левицкая Н. И.* Проблема выбора языка для начала обучения программированию в техническом вузе. *Качество. Инновации. Образование*. 2015;(12(127)):23–31. EDN: VOTMWL.

[*Pavolotsky A., Korolev D., Levitskaya N.* Choosing a language to start learning programming in a technical university. *Quality. Innovations. Education*. 2015;(12(127)):23–31. (In Russian.) EDN: VOTMWL.]

3. *Третьяков О. А., Федоркевич Е. В.* Выбор первого языка для обучения программированию. *Мир науки. Педагогика и психология*. 2020;8(5):44. EDN: JMQMKX.

[*Tretiakov O. A., Fedorkevich E. V.* Choosing the first language for teaching programming. *The World of Science. Pedagogy and Psychology*. 2020;8(5):44. (In Russian.) EDN: JMQMKX.]

4. *Григорьев С. Г.* Программирование на Прологе-Д. *Информатика и образование*. 1990;(5):50–56.

[*Grigoriev S. G.* Programming in Prolog-D. *Informatics and Education*. 1990;(5):50–56. (In Russian.)]

5. *Омарова Г. Р., Шимов И. В.* Современные языки программирования при обучении программированию школьников. *Актуальные вопросы преподавания математики, информатики и информационных технологий*. 2018;(3):270–275. EDN: UORVGR.

[*Omarova G. R., Shimov I. V.* Modern programming languages for teaching students programming. *Aktual'nye voprosy prepodavaniya matematiki, informatiki i informatsionnykh tekhnologij*. 2018;(3):270–275. (In Russian.) EDN: UORVGR.]

6. *Бабенко В. В., Гольчевский Ю. В.* Выбор языков программирования и средств проектирования для обучения специалистов по направлению «Прикладная информатика». *Прикладная информатика*. 2013;(4(46)):43–49. EDN: QGCKPZ.

[*Babenko V., Golchevskiy Yu.* Problem of programming languages and design tools choosing for educational process in direction "Applied computer science". *Journal of Applied Informatics*. 2013;(4(46)):43–49. (In Russian.) EDN: QGCKPZ.]

7. *Spinellis D.* Choosing a programming language. *IEEE Software*. 2006;23(4):62–63. DOI: 10.1109/MS.2006.97.

8. *Петров Ю. И.* Выбор языка для обучения студентов программированию для IT-направления в аграрном вузе. *Сибирский педагогический журнал*. 2020;(4):64–74. DOI: 10.15293/1813-4718.2004.07. EDN: VWTSEO.

[*Petrov Yu. I.* The choice of language for teaching students programming in the IT field at an agricultural university. *Siberian Pedagogical Journal*. 2020;(4):64–74. (In Russian.) DOI: 10.15293/1813-4718.2004.07. EDN: VWTSEO.]

9. *Иванов С. О., Ильин Д. В., Большаков И. Ю.* Сравнительное тестирование языков программирования. *Вестник Чувашского университета*. 2017;(3):222–227. EDN: ZGQAMF.

[*Ivanov S., Ilyin D., Bolshakov I.* Benchmark of programming languages. *Vestnik Chuvashskogo universiteta*. 2017;(3):222–227. (In Russian.) EDN: ZGQAMF.]

10. *Аникеева Л. А., Гусева О. Б.* Программирование на алгоритмическом языке КуМир. М.: Солон-пресс; 2012. 48 с. Режим доступа: http://m.ibooks.ru/bookshelf/344907/reading

[*Anelikova L. A., Guseva O. B.* Programming in the algorithmic language KuMir. Moscow, Solon-press; 2012. 48 p. (In Russian.) Available at: http://m.ibooks.ru/bookshelf/344907/reading]

11. *Bagge P.* Teaching primary programming with Scratch Teacher book: Research-informed approaches. London, University of Buckingham Press; 2022. 112 p.

12. *Bramer M.* Logic programming with Prolog. London, Springer; 2005. 237 p. DOI: 10.1007/978-1-4471-5487-7.

13. *Seacord R. C.* Effective C: An introduction to professional C programming. San Francisco, No Starch Press; 2020. 272 p.

14. *Schildt H., Coward D.* Java: The complete reference. NYC, McGraw Hill; 2024. 1280 p.

15. *Желудков А. В.* Глобальный язык программирования как промежуточное звено для трансформации кода между любыми языками. *Современное программное обеспечение, математическое моделирование и обеспечение информационной безопасности в компьютерных системах и комплексах. Сборник научных статей аспирантов*. М.: МФЮА; 2024:23–30. EDN: BRVFAA.

[*Zheludkov A. V.* Global programming language as an intermediate element for transforming code within any languages. *Modern Software, Mathematical Modeling and Information Security in Computer Systems and Complexes.*

Collection of Scientific Articles by Postgraduate Students. Moscow, MFUA; 2024:23–30. (In Russian.) EDN: BRVFAA.]

16. *Желудков А. В.* Разработка Глобального языка программирования с помощью метода универсальных КСД-деревьев. *Физико-математические, естественно-научные и социальные аспекты современного развития науки, техники и общества. Материалы III Региональной со всероссийским участием молодежной научной конференции*. Казань: ИП Сагиев А. Р.; 2023:31–35. EDN: TDAVMN.

[*Zheludkov A. V.* Development of a Global programming language using the method of universal CST–trees. *Physical, Mathematical, Natural Science and Social Aspects of Modern Development of Science, Technology and Society. Proc. 3th Regional Youth Scientific Conf. with All-Russian Participation*. Kazan, I P Sagiyev A. R.; 2023:31–35. (In Russian.) EDN: TDAVMN.]

17. *Вахитов Р. Х.* Пролог-Д: учебная система–интерпретатор. *Информационные технологии в образовательном процессе вуза и школы. Материалы XV Всероссийской научно-практической конференции*. Воронеж: ВГПУ; 2021:81–85. EDN: GTZXQZ.

[*Vakhitov R. Kh.* Prolog-D: Educational system–interpreter. *Information Technologies in the Educational Process of Universities and Schools. Proc. 15th All-Russ. Scientific and Practical Conf*. Voronezh, VSPU; 2021:81–85. (In Russian.) EDN: GTZXQZ.]

18. *Ali A., Smith D.* Teaching an introductory programming language in a general education course. *Journal of Information Technology Education: Innovations in Practice*. 2014;13:57–67. DOI: 10.28945/1992.

19. *Газейкина А. И.* Обучение программированию будущего учителя информатики. *Педагогическое образование в России*. 2012;(5):45–48. EDN: PHYBON.

[*Gazeykina A. I.* Training future teacher of computer science in programming. *Pedagogical Education in Russia*. 2012;(5):45–48. (In Russian.) EDN: PHYBON.]

20. *Allen O., Downs X., Varoy E., Luxton-Reilly A., Giacaman N.* Block–based object-oriented programming. *IEEE Transactions on Learning Technologies*. 2022;15(4):439–453. DOI: 10.1109/TLT.2022.3190318.

21. *Грачёв Д. А., Лаптев В. В.* Разработка многоязыкового редактора на основе семантической модели программы. *Вестник Астраханского государственного технического университета. Серия: Управление, вычислительная техника и информатика*. 2013;(2):191–201. EDN: QJBSIH.

[*Grachev D. A., Laptev V. V.* Development of the multilanguage editor on the basis of semantic model of the program. *Vestnik of Astrakhan State Technical University. Series: Management, Computer Science and Informatics*. 2013;(2):191–201. (In Russian.) EDN: QJBSIH.]

Информация об авторе

Желудков Антон Владимирович, аспирант кафедры информационных и математических инноваций, Институт информационных технологий, Московский финансово-юридический университет МФЮА, г. Москва, Россия; *ORCID:* https://orcid.org/0009-0006-9211-1008; *e-mail:* zhantonv@gmail.com

Information about the author

Anton V. Zheludkov, a postgraduate student at the Department of Information and Mathematical Innovations, Institute of Information Technologies, Moscow University of Finance and Law MFUA, Moscow, Russia; *ORCID:* https://orcid.org/0009-0006-9211-1008; *e-mail:* zhantonv@gmail.com

Поступила в редакцию / Received: 24.03.25.
Поступила после рецензирования / Revised: 16.06.25.
Принята к печати / Accepted: 17.06.25.