Научная статья
УДК 004.432
DOI: 10.24412/2072-9014-2026-276-29-40

ПОДХОД К СРАВНЕНИЮ ПАРАДИГМ ПРОГРАММИРОВАНИЯ С ПОМОЩЬЮ МЕТОДА УНИВЕРСАЛЬНЫХ РАСШИРЕННЫХ СИНТАКСИЧЕСКИХ ДЕРЕВЬЕВ

Антон Владимирович Желудков

Московский финансово-юридический университет,
Москва, Россия
zhantonv@gmail.com, https://orcid.org/0009-0006-9211-1008

Аннотация. В статье демонстрируется подход по увеличению эффективности создания программ, используемых для изучения различий в парадигмах программирования при обучении студентов. Данный подход заключается в применении метода универсальных расширенных синтаксических деревьев (УРСД) для перевода текстов программного обеспечения (ПО) между языками разработки. Рост эффективности достигается путем сокращения количества ПО, написанного в ручном или полуручном режиме.

Ключевые слова: конвергентный язык программирования; метод универсальных расширенных синтаксических деревьев; перевод текста программы; C; Java; обучение программированию.

Для цитирования: Желудков А. В. Подход к сравнению парадигм программирования с помощью метода универсальных расширенных синтаксических деревьев / А. В. Желудков // Вестник МГПУ. Серия «Информатика и информатизация образования». 2026. № 2 (76). С. 29–40. https://doi.org/10.24412/2072-9014-2026-276-29-40

Scientific article
UDC 004.432
DOI: 10.24412/2072-9014-2026-276-29-40

AN APPROACH TO COMPARING PROGRAMMING PARADIGMS USING THE UNIVERSAL EXTENDED SYNTAX TREES METHOD

Anton V. Zheludkov

Moscow University of Finance and Law,
Moscow, Russia
zhantonv@gmail.com, https://orcid.org/0009-0006-9211-1008

Abstract. This article presents an approach to improving the efficiency of creating programs used to study differences in programming paradigms. It involves using the UEST method to translate software text between programming languages. This efficiency gain is achieved by reducing the amount of software written manually or semi-manually.

Keywords: Convergent programming language; UEST method; program text translation; C; Java; programming training.

For citation: Zheludkov A. V. An approach to comparing programming paradigms using the universal extended syntax trees method / A. V. Zheludkov // MCU Journal of Informatics and Informatization of Education. 2026. № 2 (76). P. 29–40. https://doi.org/10.24412/2072-9014-2026-276-29-40

Введение

При обучении студентов разработке программного обеспечения (ПО) полезно проводить сравнения различных подходов к созданию ПО — парадигм программирования [1]. Один из способов сравнения — это задействование автоматизированных методов по переводу исходных текстов программных продуктов между различными языками. Существующие подходы можно разделить на две большие группы:

1. Методы, в основе которых лежит использование искусственного интеллекта (ИИ) [2–5]. Работают с большим количеством языков, но имеют серьезный недостаток в виде необходимости тестирования получаемого результата на корректность.

2. Точные средства [6–8], дающие возможность переводить текст ПО для ограниченного количества языков и требующие разработки отдельных модулей для каждого поддерживаемого направления перевода.

В представленной работе задействован метод универсальных расширенных синтаксических деревьев (УРСД) [9; 10], выполняющий точный перевод и использующий специальное промежуточное состояние, определяемое грамматикой разработанного конвергентного языка, что позволяет уменьшить количество

требуемых подсистем перевода. Трансляция исходного текста ПО сначала выполняется на конвергентный язык, а затем на целевой, что для 10 языков сокращает число преобразователей с 45 (число сочетаний без повторений из 10 по 2) до 10. Применение данного метода при обучении программированию позволяет также расширить возможности учебных языков разработки. Например, при разработке на Прологе-Д [11; 12] появляется возможность использования модулей, реализованных на других языках.

В более ранних работах, посвященных методу УРСД, встречаются термины «глобальный язык», «конкретное синтаксическое дерево» и «универсальное конкретное синтаксическое дерево», которые в дальнейшем были заменены на «конвергентный язык», «расширенное синтаксическое дерево» и «универсальное расширенное синтаксическое дерево», для более точного отражения их смысловой нагрузки.

Цель исследования — с помощью метода УРСД продемонстрировать подход по увеличению эффективности создания программ, на примере которых показываются различия в парадигмах программирования. Рост эффективности предполагается получить путем сокращения трудозатрат, а именно количества ПО, написанного в ручном или полуручном режиме, то есть при непосредственном участии человека в процессе разработки.

Методы исследования

В описании метода УРСД используется следующая терминология:

- Абстрактное синтаксическое дерево (АСД) — структура данных, строящаяся по исходному тексту программы и грамматике исходного языка, в вершинах которой хранятся операции, производимые над переменными и константами, расположенными в листьях.
- Расширенное синтаксическое дерево (РСД) — АСД, содержащее вспомогательную информацию, используемую при переводе текста ПО на другой язык разработки.
- Универсальное расширенное синтаксическое дерево (УРСД) — РСД фиксированного вида, определяемого грамматикой конвергентного языка программирования.
- Конвергентный язык программирования — язык программирования, упрощающий и унифицирующий процесс перевода текста ПО между различными языками разработки.

Следует подчеркнуть, что поскольку каждый из существующих языков программирования создается и развивается для наиболее удобного решения каких-либо типов задач, то использование языка, отличного от конвергентного, в качестве промежуточного давало бы меньше контроля над процессом перевода и, как следствие, затрудняло бы его.

Основные этапы применения метода УРСД представлены на рисунке 1.

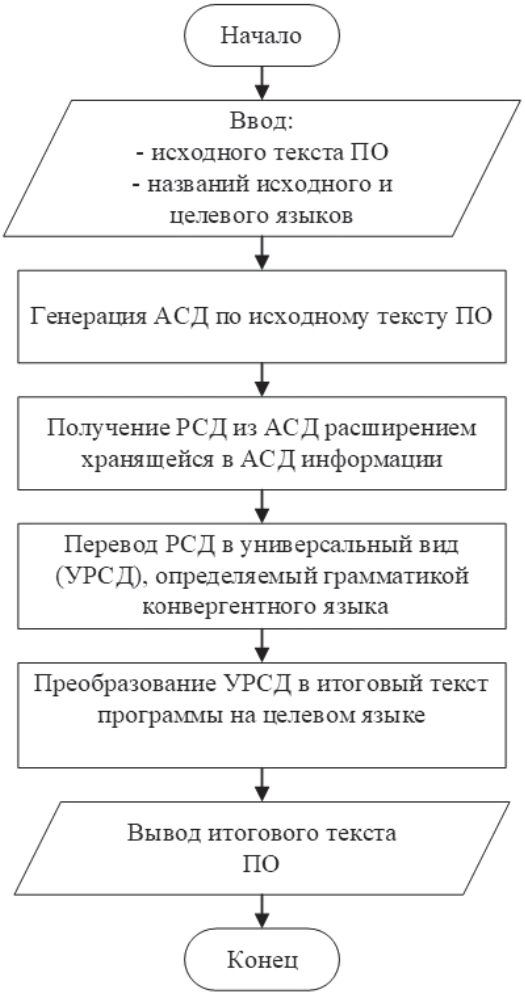

Рис. 1. Схема использования метода УРСД

Для сокращения количества трудозатрат при разработке ПО, на примере которого выполняется демонстрация различий в парадигмах программирования, можно задействовать подход, позволяющий реализовать одну программу в какой-либо парадигме, а затем автоматизированно с помощью метода УРСД перевести ее текст на другие языки разработки в требуемые парадигмы. Так как метод дает точный работоспособный результат, то полученные программы можно непосредственно задействовать, например, при составлении методических материалов.

Результаты исследования

Проведем эксперимент, показывающий работу метода УРСД по переводу текста программы из процедурной парадигмы в объектно-ориентированную. На вход разработанному программному комплексу [13] подается программа

линейного поиска элемента массива на процедурном языке C [14] (листинг 1). Для большей наглядности в представленной программе отсутствуют операторы ввода самого массива, однако при необходимости их можно добавить.

Листинг 1

Программа на языке C, выполняющая линейный поиск *n*-го элемента массива

```
#include <stdio.h>

int найтиИндекс(int * arr, int n, int target) {
   int i = 0;
   while (i < n) {
      if (arr[i] == target) {
         return i;
      }
      i = i + 1;
   }
   return -1;
}

int main() {
   int x[] = { 10, 5, 16, 7, 14, 8};
   int n = 6;
   printf("Введите элемент для поиска:\n");
   int target = 0;
   scanf("%d", &target);
   printf("Результат поиска: %d", найтиИндекс(x, n, target));
   return 0;
}
```

Под данному исходному коду выполняется построение АСД, а затем и РСД. Поддерево для подчеркнутых в листинге 1 строчек представлено на рисунке 2.

Затем выполняется преобразование РСД в УРСД, вид которого задается конвергентным языком программирования. Текст программы на конвергентном языке продемонстрирован в листинге 2. УРСД для подчеркнутого в листинге 1 фрагмента изображено на рисунке 3. На нем видны отличия в инициализации переменной, получении ее значения и вызове функции.

Далее по полученному УРСД генерируется текст программы на целевом языке Java [15] (см. листинг 3). В результате можно продемонстрировать некоторые отличия объектно-ориентированной парадигмы разработки от процедурной, а именно:

– основная логика работы программы обернута во вспомогательный класс;

– функция «найтиИндекс» преобразована в метод с приватным доступом;

– ввод данных происходит через вспомогательный объект Scanner, хранящийся в соответствующем поле.

Листинг 2

Программа на конвергентном языке, выполняющая линейный поиск *n*-го элемента массива

```
начало

функция найтиИндекс (arr: [число], n: число, target: число) -> число {
   пусть i: число = 0
   пока (i < n) {
      Если (arr[i] == target) {
         вернуть i
      }
      i = i + 1
   }
   вернуть -1
}

пусть x: [число] = [10, 5, 16, 7, 14, 8]
пусть n: число = 6
вывод(формат:"Введите элемент для поиска:\n", параметры:[])
пусть target: число = вводЧисла()
вывод(формат:"Результат поиска: %d", параметры:[найтиИндекс(arr:x, n:n,
 target:target)])
конец
```

Листинг 3

Программа на языке Java, выполняющая линейный поиск *n*-го элемента массива

```
import java.util.Scanner;
class LogicClass {
   private Scanner scanner;

   private int найтиИндекс (int[] arr, int n, int target) {
      int i = 0;
      while (i < n) {
       if (arr[i] == target) {
         return i;
      }
      i = i + 1;
   }
   return -1;
   }

   public void main() {
      scanner = new Scanner(System.in);
      int[] x = {10, 5, 16, 7, 14, 8};
      int n = 6;
```

Рис. 2. РСД части программы линейного поиска элемента в массиве

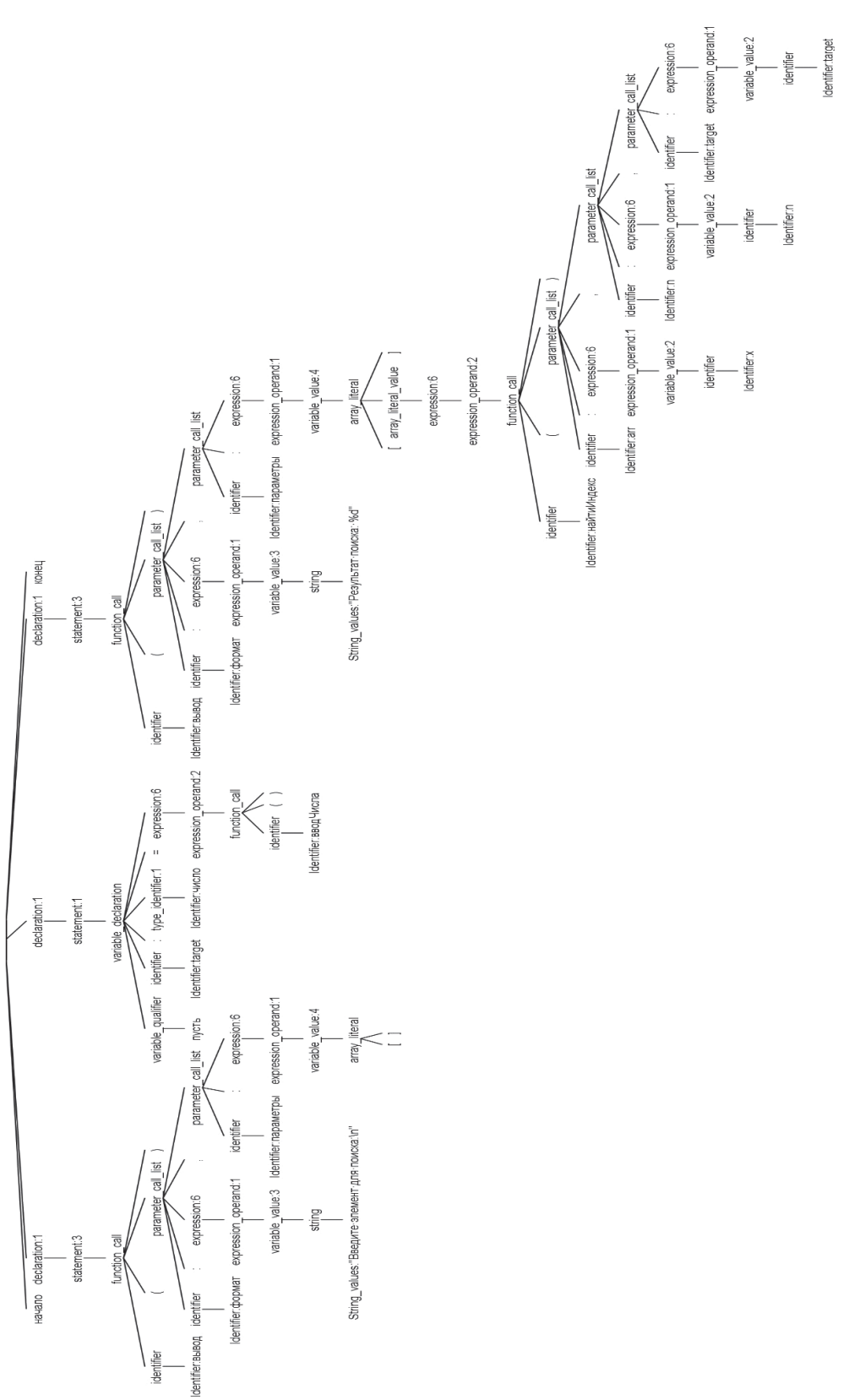

Рис. 3. УРСД части программы линейного поиска элемента в массиве

Продолжение Листинга 3

```
        System.out.printf(«Введите элемент для поиска:\n»);
        int target = scanner.nextInt();
        System.out.printf(«Результат поиска: %d», найтиИндекс(x, n, target));
        scanner.close();
    }
}

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

Таким образом, при использовании метода УРСД количество программ в различных парадигмах, которые требуется реализовать вручную, сокращается в общем случае n раз, где n — количество языков, поддерживающих отличающиеся парадигмы и для которых применим метод УРСД. В текущем примере не требуется отдельно писать программу на языке Java, то есть вместо двух программ необходимо разработать только одну, что приводит к двукратному сокращению трудозатрат и соответственно повышению эффективности в два раза.

Данный подход упрощает подготовку программ, используемых в качестве примеров для изучения способов и реализаций решения какой-либо задачи на различных языках программирования. Следовательно, можно приводить больше наглядных сравнений, что позволит улучшить понимание материала обучающимися. Также подход позволяет выполнять плавный переход для изучения нового языка на основе уже известного. Например, можно часть программы реализовать на изученном языке, а часть — на новом, автоматизированно переведя результат как к одному, так и к другому языку, тем самым наглядно исследовать разницу в синтаксисе и в принципах построения программы. Например, в приведенном в данной работе эксперименте демонстрируются различия между программами на языках C и Java.

Заключение

В результате проведенного исследования описан подход по увеличению эффективности разработки ПО, используемого для изучения отличий в парадигмах программирования. В основе предложенного способа лежит метод УРСД. Также приведено описание эксперимента, демонстрирующего двукратное повышение эффективности создания программ, на примере которых показываются различия между процедурной и объектно-ориентированной парадигмами, что доказывает применимость метода УРСД при обучении разработке ПО.

Список источников

1. *Floyd W.* The Paradigms of Programming / W. Floyd // Communications of the ACM. 1979. Vol. 22. No. 8. P. 455–460.

2. *Pathak R.* GPT-5 Prompt Migration and Improvement Using the New Optimizer / R. Pathak, C. Cheung. 2025. URL: https://cookbook.openai.com/examples/gpt-5/prompt-optimization-cookbook (дата обращения: 04.10.2026).

3. LLMs for science: Usage for code generation and data analysis // Journal of Software: Evolution and Process / M. Nejjar [et al.]. 2025. Vol. 37. P. e2682. DOI: 10.1002/smr.2723

4. *Odeh A.* Comparative Review of AI Techniques for Automated Code Generation in Software Development: Advancements, Challenges, and Future Directions / A. Odeh, N. Odeh, A. S. Mohammed // TEM Journal. 2024. Vol. 13. No. 1. P. 726–739. DOI: 10.18421/TEM131-76

5. Out of the BLEU: How should we assess quality of the Code Generation models? / M. Evtikhiev [et al.] // Journal of Systems and Software. 2023. Vol. 203. DOI: 10.1016/j.jss.2023.111741

6. *Грачев Д. А.* Разработка многоязыкового редактора на основе семантической модели программы / Д. А. Грачев, В. В. Лаптев // Вестник АГТУ. Серия: Управление, вычислительная техника и информатика. 2013. № 2. С. 191–201.

7. *Spöri Y.* Haxe as a Swiss knife for bioinformatic applications: the SeqPHASE case story / Y. Spöri, J. Flot // Briefings in Bioinformatics. 2024. Vol. 25. No. 5. DOI: 10.1093/bib/bbae367

8. Reengineering C++ Component Models via Automatic Program Transformation / R. Akers [et al.] // 12th Working Conference on Reverse Engineering (WCRE 2005). Pittsburgh (PA), 2005.

9. *Zheludkov A. V.* Software Development Using the Global Programming Language / A. V. Zheludkov, S. G. Grigoriev // Transforming Business, Industry, and Education for a Dynamic World / edited by G. S. Prakasha, et al. N. Y.: IGI Global Scientific Publishing, 2026. P. 131–164. DOI: 10.4018/979-8-3373-7927-2.ch008

10. *Желудков А. В.* Разработка глобального языка программирования с помощью метода универсальных ксд-деревьев / А. В. Желудков // Физико-математические, естественно-научные и социальные аспекты современного развития науки, техники и общества: материалы III Региональной со всерос. участием молодежной науч. конф. Казань: Сагиев А. Р., 2023. С. 31–35.

11. *Григорьев С. Г.* Программирование на Прологе-Д / С. Г. Григорьев // Информатика и образование. – 1990. № 5. С. 50–56.

12. *Вахитов Р. Х.* Пролог Д: Учебная система — интерпретатор // Информационные технологии в образовательном процессе вуза и школы: материалы XV Всерос. науч.-практ. конф. Воронеж: ВГПУ, 2021. С. 81–85.

13. Свидетельство о регистрации программы для ЭВМ № 2025665487 Российская Федерация, Программа перевода исходного текста алгоритмов с одного языка программирования на другой с помощью Глобального языка: № 2025664320: заявл. 05.06.2025: опубл. 17.06.2025.

14. *Seacord R. C.* Effective C: An Introduction to Professional C Programming / R. C. Seacord. San Francisco: No Starch Press, 2020. 272 p.

15. *Schildt H.* Java: The Complete Reference, Thirteenth Edition / H. Schildt, D. Coward. N. Y.: McGraw Hill, 2024. 1280 p.

References

1. *Floyd W.* The Paradigms of Programming / W. Floyd // Communications of the ACM. 1979. Vol. 22. No. 8. P. 455–460.

2. *Pathak R.* GPT-5 Prompt Migration and Improvement Using the New Optimizer / R. Pathak, C. Cheung. 2025. URL: https://cookbook.openai.com/examples/gpt-5/prompt-optimization-cookbook (accessed: 04.10.2026).

3. LLMs for science: Usage for code generation and data analysis // Journal of Software: Evolution and Process / M. Nejjar [et al.]. 2025. Vol. 37. P. e2682. DOI: 10.1002/smr.2723

4. *Odeh A.* Comparative Review of AI Techniques for Automated Code Generation in Software Development: Advancements, Challenges, and Future Directions / A. Odeh, N. Odeh, A. S. Mohammed // TEM Journal. 2024. Vol. 13. No. 1. P. 726–739. DOI: 10.18421/TEM131-76

5. Out of the BLEU: How should we assess quality of the Code Generation models? / M. Evtikhiev [et al.] // Journal of Systems and Software. 2023. Vol. 203. DOI: 10.1016/j.jss.2023.111741

6. Grachev D. A. Development of a multilingual editor based on a semantic model of a program / D. A. Grachev, V. V. Laptev // Bulletin of ASTU. Series: Management, Computer Engineering and Informatics. 2013. P. 191–201.

7. *Spöri Y.* Haxe as a Swiss knife for bioinformatic applications: the SeqPHASE case story / Y. Spöri, J. Flot // Briefings in Bioinformatics. 2024. Vol. 25. No. 5. DOI: 10.1093/bib/bbae367

8. Reengineering C++ Component Models via Automatic Program Transformation / R. Akers [et al.] // 12th Working Conference on Reverse Engineering (WCRE 2005). Pittsburgh (PA), 2005.

9. *Zheludkov A. V.* Software Development Using the Global Programming Language / A. V. Zheludkov, S. G. Grigoriev // Transforming Business, Industry, and Education for a Dynamic World / edited by G. S. Prakasha, et al. N. Y.: IGI Global Scientific Publishing, 2026. P. 131–164. DOI: 10.4018/979-8-3373-7927-2.ch008

10. *Zheludkov A. V.* Development of a global programming language using the method of universal kst-trees / A. V. Zheludkov // Physical, mathematical, natural science and social aspects of the modern development of science, technology and society: Proceedings of the III regional with all-Russian participation of youth scientific conference. Kazan: Sagiev A. P., 2023. P. 31–35.

11. *Grigoriev S. G.* Programming in Prolog-D / S. G. Grigoriev // Computer Science and Education. 1990. No. 5. P. 50–56.

12. *Vakhitov R. Kh.* Prolog D: Educational system — interpreter. Information technology in the educational process of universities and schools: proceedings of the 15th All-Russian Scientific and Practical Conference.. Voronezh: VSPUб 2021. P. 81–85.

13. Certificate of Registration of Computer Program No. 2025665487 Russian Federation, Program for translating the source text of algorithms from one programming language to another using the Global language: No. 2025664320: declared 05.06.2025: published 17.06.2025.

14. *Seacord R. C.* Effective C: An Introduction to Professional C Programming / R. C. Seacord. San Francisco: No Starch Press, 2020. 272 p.

15. *Schildt H.* Java: The Complete Reference, Thirteenth Edition / H. Schildt, D. Coward. N. Y.: McGraw Hill, 2024. 1280 p.

Статья поступила в редакцию: 05.03.2026;
одобрена после рецензирования: 15.04.2026;
принята к публикации: 15.04.2026.

The article was submitted: 05.03.2026;
approved after reviewing: 15.04.2026;
accepted for publication: 15.04.2026.

Информация об авторе / Information about the author

Антон Владимирович Желудков — выпускник аспирантуры Института информационных технологий, Московский финансово-юридический университет, Москва, Россия.

Anton V. Zheludkov — Graduate of the postgraduate, Institute of Information Technologies, Moscow University of Finance and Law, Moscow, Russia.

zhantonv@gmail.com, https://orcid.org/0009-0006-9211-1008