УДК 378.14.015.62
DOI: http://dx.doi.org/10.21686/1818-4243-2021-4-13-22

**А.П. Сергушичева, Е.Н. Давыдова**
Вологодский государственный университет, Вологда, Россия

# Разработка и применение программного инструмента для поддержки обучения формальным языкам

***Цель статьи*** *представить результаты исследования по возможности применения дедуктивного подхода в изучении языков программирования (от теории формальных языков к конкретным языкам программирования) и разработке обучающей системы для реализации этого подхода. Вопрос подготовки специалистов в области информационных технологий по-прежнему остается актуальным, а разнообразие языков программирование настолько велико, что далеко не всегда удается угадать, какой из них будет востребован в профессиональной деятельности. По мнению авторов, применение указанного подхода позволит видеть общие элементы и находить синтаксические различия языков программирования, а следовательно, упростить и ускорить их освоение. В статье проанализированы методы обучения программированию, обоснована актуальность изучения формальных языков будущими ИТ-специалистами, сформулированы требования к программному инструменту для поддержки обучения формальным языкам, описана его реализация.*

***Материалы и методы:*** *теория формальных языков, синтаксис и семантика языков программирования, лексический анализ, анализ различных источников информации по исследуемой тематике, систематизация собранных данных, технологии проектирования, реализации и тестирования программных продуктов, экспериментальные исследования.*

***Результаты.*** *Предложена методика изучения языков программирования посредством формальных языков. Создана программная обучающая система, которая позволяет связать теорию формальных языков с языками высокого уровня за счёт соответствующих примеров. Разработан и реализован в указанной системе алгоритм проверки корректности выполнения задания посредством синтаксического анализа введенной обучающимся программы и имитации ее выполнения. Эксперименты показали целесообразность подхода и работоспособность программного продукта. В настоящее время разработанная система применяется в Вологодском государственном университете при преподавании дисциплин «Теория языков программирования и методы трансляции» и «Теория автоматов и формальных языков».*

***Заключение.*** *Результаты исследования показывают приемлемость предложенного подхода и целесообразность применения разработанной программы при изучении языков программирования.*

***Ключевые слова:*** *программный продукт, языки программирования, формальные языки, методы обучения программированию, алгоритм.*

**Anna P. Sergushicheva, Elena N. Davydova**
Vologda State University, Vologda, Russia

# Development and Application of a Software Tool to Support the Teaching of Formal Languages

***The purpose*** *of the article is to present the results of a study on the possibility of using a deductive approach in the study of programming languages (from the theory of formal languages to specific programming languages) and the development of a training system for implementing this approach. The issue of training specialists in the field of information technology is still relevant, and the variety of programming languages is so great that it is not always possible to guess which of them will be in demand in professional activities. According to the authors, the application of this approach will allow you to see common elements and find syntactic differences in programming languages, and therefore simplify and speed up their development. The article analyzes the methods of teaching programming, substantiates the relevance of learning formal languages by future IT specialists, formulates the requirements for a software tool to support learning formal languages, describes its implementation.*

***Materials and methods.*** *Theory of formal languages, syntax and semantics of programming languages, lexical analysis, analysis of various sources of information on the subject under study, systematization of the collected data, technologies for designing, implementing and testing software products, experimental research.*

***Results.*** *A methodology for learning programming languages through formal languages is proposed. A software training system has been created that allows you to link the theory of formal languages with high-level languages through appropriate examples. The algorithm of checking the correctness of the task execution by means of syntactic analysis of the program entered by the student and imitation of its execution is developed and implemented in the specified system. Experiments have shown the feasibility of the approach and the performance of the software product. Currently, the developed system is used at Vologda State University when teaching the disciplines "Theory of programming languages and translation methods" and "Theory of automata and formal languages".*

***Conclusion.*** *The results of the study show the acceptability of the proposed approach and the expediency of using the developed program when learning programming languages.*

***Keywords:*** *software product, programming languages, formal languages, methods of teaching programming, algorithm.*

## Введение

В настоящее время информационные технологии и программное обеспечение применяются практически во всех сферах деятельности человека. Возрастает потребность в соответствующих специалистах и в ответ на это высшие и средние профессиональные учебные заведения увеличивают прием абитуриентов по направлениям подготовки, связанным с созданием и применением программного обеспечения. Писать программы стало проще: языки программирования высокого уровня скрывают множество рутинных, но обязательных операций и позволяют почти на родном языке описать алгоритм работы системы. Более того, появилось множество инструментов и библиотек, посредством которых сравнительно легко собрать несложную программу с заданным функционалом. С другой стороны, увеличились сложность и размер программных систем. Все чаще они наделяются элементами искусственного интеллекта: нас не устраивает просто выборка требуемых данных, надо, чтобы эти данные были проанализированы по ряду критериев, определены проблемы и предложены пути их решения. Как следствие, повысились требования к разработчикам таких систем. Приветствуется знание ими нескольких языков программирования. Вопросам обучения программированию, разработке методов и методик преподавания языков программирования посвящено значительное количество работ отечественных и зарубежных авторов, диссертационных исследований (например, работы [1–6]). Об актуальности проблемы говорит и наличие значительного количества Интернет-ресурсов, ей посвященных (например, работа [7]).

Все языки программирования относятся к группе формальных языков. Существует большое количество языков программирования, которые могут синтаксически значительно отличаться друг от друга. Каждый язык имеет свои особенности, так как создавался под решение определённого класса задач. При этом в основе всех языков программирования лежит единая математическая модель, называемая формальным языком. Формальный язык определяет множество семантических, синтаксических и лексических правил, отвечающих как за действия, которые выполнит компьютер или иной исполнитель, так и за внешний вид программ.

Авторы предлагают использовать подход от общего к частному: от изучения теории формальных языков – к изучению конкретных языков. Знание теории формальных языков позволит обучающемуся акцентировать внимание на особенностях конкретного языка и тем самым упростить и ускорить процесс его освоения. В статье описан опыт авторов по реализации и применению программного инструмента для поддержки обучения формальным языкам с автоматической проверкой правильности выполнения заданий по написанию кода в соответствии с предложенным алгоритмом. Наличие системы автоматической проверки корректности выполнения заданий позволяет использовать его при дистанционном обучении.

## 1. Методы обучения программированию

Метод обучения – это способ совместной деятельности учителя и обучающихся с целью усвоения последними предусмотренных содержанием обучения знаний, умений и навыков. Первичные знания по программированию в настоящее время обучающиеся получают уже в школе, как правило, в рамках информатики или информационных технологий. Существуют три наиболее распространенных подхода к преподаванию программирования [8]:

1) как теоретической дисциплины (без освоения конкретных языков и систем). Далеко на все нынешние школьники свяжут свою будущую профессию с разработкой программного обеспечения, но подавляющее большинство из них уже сегодня не однажды это программное обеспечение использовали (смартфоны, гаджеты, бытовая техника). И задача общеобразовательного курса – дать представление, о том, как это работает, выработать определенный стиль мышления, развить логику. Однако, с отказом от языка программирования теряется возможность использовать соответствующий инструментарий, труднее обосновать необходимость его использования, снижается мотивация к изучению дисциплины;

2) преподавание на основе специально разработанного языка (Школьник, Рапира, SMR, LOGO, Scratch и др.). Такие языки предельно упрощены и рассчитаны на возможности младших школьников. Их применение дает видимый результат (а успех окрыляет);

3) преподавание на основе одного или нескольких широко используемых языков программирования. Такой курс может стать базой для последующего профессионального изучения программирования. И здесь возникает проблема, какой язык выбрать. Он должен быть современным, универсальным и достаточно простым для освоения одновременно. Однако, все языки программирования разрабатывались для решения конкретных задач и каждый из них ориентирован на определенную область применения, большинство их реализаций

перегружено техническими деталями и сложны в изучении.

Существуют различные классификации методов обучения. Так, по способу передачи информации их можно разделить на словесные, наглядные и практические. К словесным относятся лекция, рассказ, беседа, работа с книгой, консультация, дискуссия. Их целью является доведение актуальной информации по определённой дисциплине до обучающегося [9]. Процесс формирования умений, навыков и знаний здесь осуществляется путём рефлексии, усвоения и запоминания информации. Наглядные методы предполагают использование дополнительных инструментов (иллюстраций, схем, таблиц, моделей, технических приспособлений, видео- и аудиоматериалов), прямо или косвенно отражающих предмет изучения [10]. Практические методы основаны на вовлечении аудитории в учебный процесс: познавательные игры, упражнения, тренинги, практические задания и т.п. По логике изложения материала методы делят на дедуктивные и индуктивные; по характеру учебной деятельности – на исследовательские, проблемные и т.д.

При изучении языков программирования можно применять практически все перечисленные выше методы, как по отдельности, так и в различных сочетаниях. Проблема может заключаться лишь в отсутствии соответствующей методики. Традиционная методика предполагает, что после теоретического изучения алфавита и базовых конструкций языка переходят к решению простейших задач, и далее постепенно, по мере приобретения навыков в составлении программ, задачи усложняются.

Разработке методик применения известных методов к обучению программированию и разработке новых методов посвящены ряд научных статей и диссертаций.

Например, в работе [11] исследуется возможность применения кейс-метода в лабораторных занятиях по дисциплине «Информатика». Метод предполагает наличие модели некоторой социально-экономической системы. В системе конечно же есть проблемы, которые можно разрешить, если создать и внедрить соответствующее программное обеспечение. Определяется цель разработки. Преподаватель должен заранее продумать, как он будет направлять обучающихся к выбору этой единой цели, и позаботиться о наличии нескольких альтернативных путей достижения поставленной цели. Анализ конкретных ситуаций, выработка решений, ох оценка – работа коллективная. Ситуация, используемая на занятии, не должна быть перегружена лишней информацией, отвлекающей внимание от решаемой проблемы. Студентам предлагается проанализировать ситуацию, построить модель, найти различные варианты решения и выбрать из них оптимальное по ряду критериев. Определяется структура представления данных и их типы, эффективность используемых алгоритмов. Автор статьи приводит примеры ситуационного анализа и последующего написания программ в средах программирования Delphi и MatLab. Программное обеспечение, созданное в разных средах, также рассматривается как альтернативное.

Идея метода проектов состоит в том, что обучающийся в приобретении знаний не ограничивается знакомством с мировыми достижениями, а на основании собственного опыта приходит к известным заключениям. Любой учебный предмет должен быть связан с какой-нибудь жизненной ситуацией, таким образом, чтобы студент смог увидеть область приложения своих знаний [12–13]. Использование метода проектов в процессе обучения программированию естественно (создание любой программы осуществляется через проектирование и реализацию) и позволяет приблизить обучающегося к научно-исследовательской и практико-ориентированной деятельности. Выбор темы проекта – не проблема: сегодня программное обеспечение востребовано почти во всех отраслях деятельности человека. Проект может быть достаточно объемным, обладать высоким уровнем системности и разрабатываться коллективно (группами программистов). Жемчужников Д.Г. в своей диссертации [14 ] исследует возможность применения в качестве сквозных проектных задач разработку динамических компьютерных игр. Создавать игру гораздо интереснее, чем реализовывать однотипные вычислительные алгоритмы.

Участие в проектной деятельности позволяет обучающимся приобрести уникальный опыт, невозможный при других формах обучения. Метод обладает высоким мотивационным потенциалом и позволяет развить самостоятельность и творческие способности. К недостаткам метода можно отнести слабую разработанность его теоретических положений, процессуально-результативной технологической базы; сложность достижения запланированных результатов обучения, систематичности и фундаментальности знаний; повышенные требования к преподавателю – организатору и руководителю проекта.

Значительной части этих недостатков лишен метод открытых программ [15 ]. Под открытой программой понимается написанная на изучаемом языке модель, представляющая некоторый класс программ и предназначенная для передачи обучающимся знаний об операторах, синтаксисе, логике, структуре и назначении программ своего класса. Модель должна быть работаю-

щей, сопровождаться подробными комментариями и быть доступной для модификации. Целью обучения может быть изучение структуры программы, результатов и алгоритма ее работы (в том числе и посредством запуска программы на исполнение), документирование логики алгоритма, поиск и исправление ошибок. В зависимости от учебных целей она может содержать пропуски, которые следует заполнить соответствующим кодом (естественно, что возможно несколько вариантов кода), ошибки, которые необходимо найти и устранить.

Школьникам можно предложить игровой метод обучения программированию: задачи вычислительного типа обычно для учащихся не представляют интереса и воспринимаются ими как повинность. Разработаны и применяются ряд компьютерных игр, в ходе которых создается программа для управления поведением какого-либо объекта. Например, игра Game Maker содержит набор игровых объектов, поведение которых задаётся путём программирования реакции на события. Для представления программ используют графические блоки (подобно элементам блок-схем) в режиме drag-n-drop или скриптовый язык GML, похожий на JavaScript. В фокусе игры ПиктоМир – космодром, с которого стартуют космические корабли, и при старте они выжигают покрытие. Для ремонта покрытия посылают робота-вертуна. Необходимо задать роботу такую программу, чтобы он всё выгоревшее залил защитным составом и при этом сам не разбился о бордюрчик. В 3d-стратегии Колобот игра заключается в написании программ для роботов, которые должны подготовить планеты для заселения и добычи полезных ископаемых. Заставляя роботов выполнить задание, дети учатся составлять алгоритмы.

Таким образом, можно сделать вывод, что разнообразие методов обучения программированию достаточно велико и что изучение программирования в большинстве случаев осуществляется через освоение какого-либо языка программирования и соответствующей среды программирования. Авторы статьи считают перспективным и другое направление: от изучения теории формальных языков – к конкретным языкам.

## 2. Актуальность изучения формальных языков будущими ИТ-специалистами

Обычные разговорные языки считают естественными. С развитием науки, техники, искусств возникли языки искусственные, предназначенные для решения различных классов задач в некоторой предметной области: язык математики, дорожных знаков, нотная грамота и другие. Язык, характеризующийся точными правилами построения и трактовки выражений, назвали формализованным (или формальным). Формальный язык однозначен: четко сформулированные правила семантической интерпретации и синтаксического преобразования используемых знаков обеспечивают независимость его инструкций и результата выполнения этих инструкций от каких-либо прагматических обстоятельств (например, от контекста). Формальные языки широко применяются в науке и технике в тесной взаимосвязи с естественным языком.

Элементами естественного языка являются звуки речи, морфемы (части слова), слова, предложения. Звуки образуют фонетический уровень языка, морфемы – морфемный, слова и фразеологизмы – лексический, словосочетания и предложения – синтаксический. **Лексемы** выполняют номинативную функцию (называют вещи и явления действительности) и считаются основными единицами языка. На лексическом уровне наиболее полно представлена семантика – раздел лингвистики, изучающий смысловое значение единиц языка [16–17].

Закономерности построения правильных осмысленных речевых отрезков на данном языке задача грамматики. Чтобы определить грамматику, требуется задать алфавиты терминальных и нетерминальных символов, набор правил вывода, а также выделить в множестве нетерминалов начальный. Словами языка, заданного грамматикой, являются все последовательности терминалов, выводимые (порождаемые) из начального нетерминала по правилам вывода. Теория формальных языков базируется на идее порождающих грамматик Н. Хомского для естественных языков. Эта идея получила значительное развитие в области разработки языков программирования.

Программы состоят из синтаксических конструкций, называемых командами (операторами, указаниями, предложениями). Команды строятся из *лексем* – неделимых элементов языка: слов, цифр, символов операций и тому подобное. Слова делятся на служебные, стандартные имена и идентификаторы, которые пользователь дает разным объектам. Набор правил, описывающий комбинации символов алфавита, считающиеся правильно структурированной программой или её фрагментом называется синтаксисом языка программирования. Семантика приписывает значения (действия) различным синтаксическим конструкциям. Практически во всех языках программирования присутствуют следующие конструкции:

1. конструкции для задания констант или переменных;

2. конструкции для ввода и вывода данных;

3. условные конструкции (ветвления);

4. конструкции циклов;

5. конструкции для создания функций и/или методов.

Предложенная Дж.Бэкусом и П.Науром форма определения синтаксиса языков (БНФ – Бэкуса-Наура форма, РБНФ – расширенная форма Бэкуса-Наура) позволила дать формальное описание синтаксических конструкций языка и формализовать методы синтаксического анализа при проектировании компиляторов. Синтаксис обычно проверяется на ранних стадиях трансляции, а при использовании IDE это возможно непосредственно при редактировании исходных текстов программ. В настоящее время методы и алгоритмы из теории формальных языков и грамматик широко используются при проектировании инструментов анализа и трансляции языков разметки документов и гипертекста, языков описания интерфейсов и языков спецификаций, языков описания аппаратуры, языков описания распределённых систем и коммуникационных протоколов и т.п.

Поэтому изучение теории формальных языков будущими разработчиками программного обеспечения дает им представление об общих элементах и конструкциях языков, а также правилах их построения и описания. В результате, освоение первого и последующих языков программирования сводится к выявлению их лексических и семантических особенностей, а сам механизм работы языка обучающемуся будет известен. Это способствует расширению списка языков и инструментов, которыми владеет программист, увеличению сложности и разнообразия решаемых им задач и позволяет при необходимости ускорить переход с одного языка на другой при выполнении проекта. Также ожидается сокращение числа ошибок, вызванных непониманием работы того или иного элемента. Дополнительно, изучение формальных языков помогает получить базовое представление о работе компиляторов, трансляторов, интерпретаторов.

## 3. Программный инструмент для поддержки обучения формальным языкам

При разработке программного инструмента для поддержки обучения формальным языкам были выдвинуты следующие требования:

– реализация функций предъявления учебных материалов и разбора примеров практических заданий для каждой формальной конструкции, выдачи заданий для самостоятельной работы и автоматической проверки решений;

– запись формальных конструкций должна осуществляться в нотации Бэкуса-Наура;

– задачи, даваемые обучающимся, должны строиться на основе формальных конструкций, а для реализации их решения система должна поддерживать минимум два синтаксически различных языка программирования;

– программный продукт должен позволять загружать и сохранять решения заданий в файлах формата txt;

– наличие возможности запуска продукта на любой ОС (кросплатформенность решения).

Для реализации программы использованы следующие языки и инструменты: язык программирования Java (позволяет обеспечить кросплатформенность приложения), среда разработки – IntelleJIdea 2019.3.4, средство разработки интерфейса JavaFX, легковесная база данных Java с открытым исходным кодом H2, средство контроля версий Git, средство статического анализа кода SonarQube, средство документации кода JavaDoc.

В состав программы вошли модуль навигации, модуль предъявления учебных материалов, модуль практикум, база данных, интерпретатор программы.

Модуль навигации обеспечивает переход со стартовой страницы к конкретному разделу и обратно и перемещения между разными страницами выбранного учебного модуля. Он реализован посредством классов Navigation (отвечает за навигацию, загрузку контента, его отображение, переходы между страницами), Controller (осуществляет передачу управления соответствующей странице), Module представляет информацию о конкретном модуле).

Модуль предъявления учебных материалов направлен на теоретическое освоение курса «Теория формальных языков». Теоретический материал по этой дисциплине разбит на разделы: основные понятия; простые, условные и циклические конструкции; лексический, синтаксический и семантический анализ, интерпретация программ. Рекомендуется последовательное изучение тем, однако навигация по разделам свободная: теорию можно почерпнуть и в других источниках, в этом случае к теории можно обращаться в произвольном порядке при возникновении затруднений в решении задач.

Модуль практикум может быть вызван из разделов лексического, синтаксического, семантического анализа и интерпретации программ. Предыдущие учебные модули разъясняют базовые понятия формальных языков. В их числе алфавит (совокупность исходных символов, из которых будут строиться все выражения языка), грамматика (свод правил, определяющих правильность построения предложений языка и смысловую правильность предложений), правила описания грамматики

языка (наибольшее распространение получили БНФ и РБНФ) и типовые конструкции языка (присваивание, чтение, запись, ветвление и множественное повторение ряда действий). И авторы считают, что здесь достаточно ограничиться рассмотрением примеров. Страница с примером состоит из 2 частей: верхней, где отображается пример на РБНФ (рис. 1), и нижней, где пользователь может увидеть аналог этого же описания для выбранного языка программирования.

По остальным разделам в программе также предусмотрено наличие двух видов заданий: 1) задания с фиксированным правильным ответом и 2) задания требующие проверки корректности выполнения.

В первом случае ответ полностью берётся из поля с ответом в соответствующей таблице базы данных. Если проверка на соответствие с ним введённого обучающимся варианта ответа пройдена успешно, пользователю отображается результат. В противном случае после анализа ответа выводится информация о найденном несоответствии.

В втором случае, как правило, существует несколько решений поставленной задачи. Например, сравнительно простая задача, представленная на рис. 2, может быть решена посредством любого из итерационных конструктивов. Для таких ситуаций требуется производить анализ и интерпретацию программы. Анализ начинается с проверки корректности формата: необходимо убедиться, что пользователь ввёл решение в формате, который сможет интерпретировать программа. В данном случае, неверным форматом будет написание кода вне метода Main, а некорректным с точки зрения задания является решение, где пользователь выводит на экран числа от 1 до 5 в обход циклических конструкций.

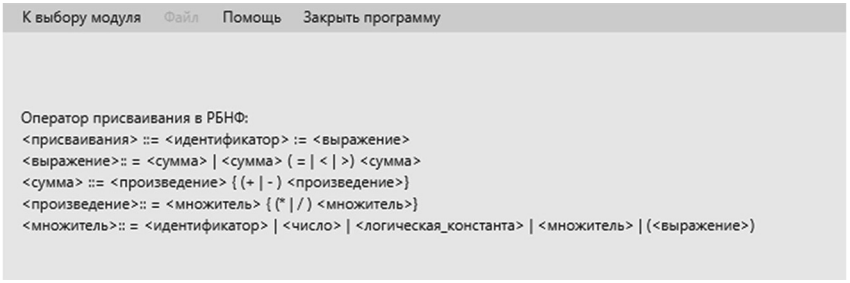


**Рис. 1.** Верхняя часть страницы с примером

**Fig. 1.** The upper part of the page with an example

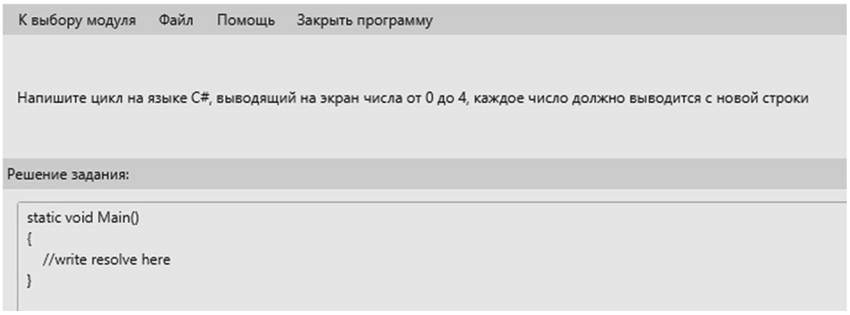


**Рис. 2.** Страница с примером задания

**Fig. 2.** A page with an example of the task

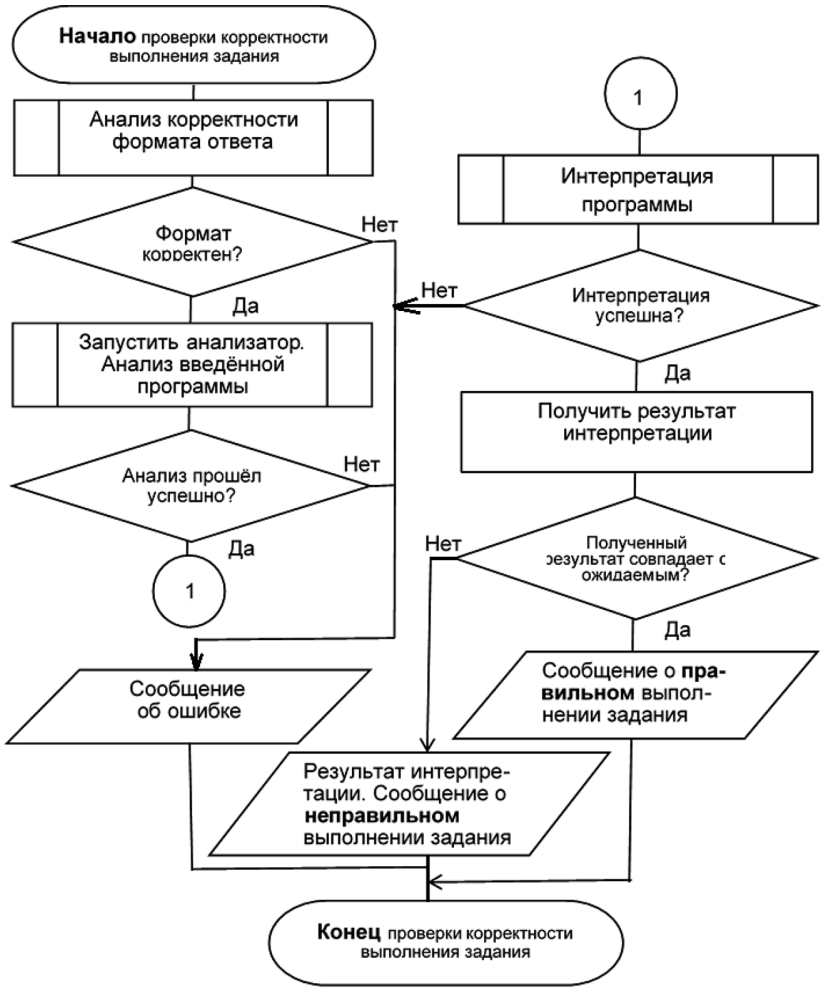


**Рис. 3.** Алгоритм проверки корректности выполнения задания

**Fig. 3.** Algorithm for checking the correctness of the task

Правильный ответ для задания также лежит в поле с ответом базы данных, но определяется он на основе метаинформации, в данном случае показывающей, что требуется интерпретация программы на языке C#. Соответственно, вызывается анализатор C#, который осуществляет лексический, синтаксический и семантический анализ решения. После интерпретации программы анализируется результат ее выполнения: он должен точно совпадать с тем, что приведён в базе данных. Алгоритм анализа введенной в качестве ответа на задание программы приведен на рис. 3.

Интерпретатор программы реализует выполнение написанных обучающимися программ посредством запуска аналогичных конструкций в языке Java. В данной версии реализована интерпретация с языков Python и C#.

Для хранения информации было решено применить **базу данных** H2 [18]. Она поддерживает синтаксис SQL, написана на Java, является открытым ресурсом (к ней можно подключиться используя JDBC API). Для взаимодействия с базой данных используется объектно-ориентированный запрос в Java. База данных включает следующие таблицы:

– MODULE_TYPES – типы учебных модулей. Каждая кнопка на стартовой странице ведёт к открытию модуля определённого типа. Таблица MODULE содержит описание контента модулей, а таблица MODULE_PAGES – страниц модуля. При этом возможный тип страницы задаётся в таблице PAGE_TYPE: теория, пример, задание с выбором варианта ответа и задание с вводом ответа обучающимся;

– PRIMER. Каждый пример содержит описание языковой конструкции в разных нотациях (пока в РБНФ, Python и C#) и связан с конкретной страницей конкретного учебного модуля;

– TASK. Показывает тип задания и его связь с учебным модулем. Возможные типы заданий приведены в таблице TASK_TYPE. Это – задание с единичным или множественным выбором правильного ответа и задание, требующее от пользователя написание кода в качестве ответа. Описание заданий заносится в таблицы TEST_TASK и ANSWER_TASK соответственно;

– LEXICL_TASK и LEXEMES содержат информацию, необходимую для лексического анализа введённой обучающимся программы.

Обучающимся доступен пункт меню “Помощь”, который включает список всех элементов управления и подробное руководство по работе с продуктом

## 4. Методика применения программы для изучения формальных языков

Начинаем с изучения теории формальных языков. Один из лучших способов знакомства с теорией – прослушать соответствующие лекции: преподаватель даёт хорошо систематизированный материал, расставляет акценты на важных моментах, отвечает на возникшие вопросы. Кроме того, обучающийся получает список литературы, в которой он может почерпнуть дополнительную информацию. Не исключается и самостоятельный поиск и анализ источников информации, в том числе и на просторах Интернета. В лекционном курсе обязательно должны быть представлены примеры применения теории формальных языков к конкретным языкам программирования (от общего к частному). Выбор языка программирования (не исключительно Pythoon и C#, реализованные в программе) осуществляется преподавателем и может зависеть от контингента обучающихся. Теоретический материал также входит в контент разработанной программы.

Следующий шаг – знакомство с инструментом. Простой, интуитивно-понятный пользовательский интерфейс позволяет студентам самостоятельно разобраться с функциональностью программы. При этом можно воспользоваться пунктом меню “Помощь”. Он содержит два подпункта:” Инструкция” (подробное руководство по работе с продуктом) и “Навигация” (включает список всех элементов управления), каждый из которых открывается в модальном окне поверх основного приложения. Дополнительно справочная информация поставляется в архиве с программным продуктом в виде файла txt. Также информацию по работе с программой предоставляет педагог в методических указаниях к выполнению лабораторных и/или практических работ.

Теперь можно переходить к выполнению заданий. Решение задачи вводится студентом вручную в специально предназначенное для этого поле программы. Есть возможность загрузки решения из файлов. В случае затруднений всегда можно обратиться к теоретическому материалу и приведенным примерам. Проверка корректности выполнения задания инициируется нажатием кнопки «Проверить». Решение можно сохранить в файле. Правильное решение полностью или частично включается в отчёт по работе. Если решение оказалось некорректным, студент выполняет работу над ошибками. В результате автоматизации проверки решения обучающийся получает быструю обратную связь, а преподаватель – сэкономленное время.

В заключение необходимо оформить и защитить отчет по выполненной работе. Окончательная проверка знаний обучающихся проходит в форме,

предусмотренной рабочей программой дисциплины: в виде зачёта или экзамена.

## Заключение

Таким образом, в результате исследования:

– предложена методика изучения языков программирования посредством формальных языков;

– создана программная обучающая система, которая позволяет обучающемуся связать теорию формальных языков с языками высокого уровня за счёт соответствующих примеров;

– разработан и реализован в указанной системе алгоритм проверки корректности выполнения задания посредством синтаксического анализа введенной обучающимся программы и имитации ее выполнения.

Эксперименты показали приемлемость предложенного подхода, работоспособность разработанного программного инструмента и целесообразность его использования при изучении языков программирования. В настоящее созданная система применяется в Вологодском государственном университете при преподавании дисциплин “Теория языков программирования и методы трансляции” и “Теория автоматов и формальных языков”.

## Литература


1. Шефер О.Р., Носова Л.С., Лебедева Т.Н. Современная методология изучения программирования в вузе // Научно-техническая информация. Серия 1: Организация и методика информационной работы. 2018. № 5. С. 6–12.
2. Баженова И.В., Пак Н.И. Разработка электронного учебника-трансформера при обучении программированию на основе самопознавательной деятельности студента // Вестник Московского городского педагогического университета. Серия: Информатика и информатизация образования. 2019. № 1(47). С. 20–28.
3. Моглан Д.В. Дидактический потенциал использования систем визуализации алгоритмов в процессе обучения программированию // Открытое образование. 2019. Т. 23. № 2. С. 31–41.
4. Шарипов Ф.Ф., Мараджабов С.И. Теоретическая модель формирования алгоритмического мышления студентов вузов в процессе обучения объектно-ориентированному программированию // Балтийский гуманитарный журнал. 2017. Т. 6. № 3(20). С. 313–316.
5. Шкарбан Ф.В. Методика обучения основам объектно-ориентированного программирования бакалавров прикладной информатики с использованием визуальных учебных сред: диссертация кандидата педагогических наук: 13.00.02. Волгоград, 2018. 212 с.
6. Касьянов В.Н., Касьянова Е.В. Методы и средства обучения программированию в вузе // Информатика: проблемы, методы, технологии: материалы XX международной научно-методической конференции. 2020. С. 1989–1998.
7. Самбо Е. Какие языки программирования учить в 2021 (для начинающих) [Электрон. ресурс]. Режим доступа: https://videoinfographica.com/programming-languages/.
8. Милова Е.А. Методика обучения программированию [Электрон. ресурс]. Режим доступа: https://pandia.ru/text/79/134/22349.php.
9. Педагогика. Урок 4: традиционные методы обучения [Электрон. ресурс]. Режим доступа: https://4brain.ru/pedagogika/new-methods.php.
10. Методы обучения: понятие, виды и классификация в педагогике [Электрон. ресурс]. Режим доступа: https://nauka.club/podsovet/metody-obucheniya.html.
11. Юрьева Т.А., Чалкина Н.А., Лебедь О.А. Применение кейс-метода в обучении бакалавров основам программирования // Педагогические науки. 2016. № 7. С. 78–82.
12. Слинкин Д.А. Использование метода проектов при обучении программированию в курсе информатики: диссертация кандидата педагогических наук: 13.00.02. Екатеринбург, 2001 166 с.
13. Лебедева Т.Н. Метод проектов в обучении студентов // Актуальные проблемы развития среднего и высшего образования: XV межвузовский сборник научных трудов. Челябинск, 2019. С. 204–207.
14. Жемчужников Д.Г. Методика обучения программированию, основанная на создании школьниками динамических компьютерных игр: диссертация кандидата педагогических наук: 13.00.02. Москва, 2013. 230 с.
15. Сидорик В.В., Костина Е.Н. Метод открытых программ в изучении программирования // Наука – образованию, производству, экономике: материалы 11-й международной научно-технической конференции. Минск: БНТУ, 2013. Т. 4. С. 268.
16. Гладкий А.В. Формальные грамматики и языки. М.: Наука, 1973. 368 с.
17. Лаздин А.В. Формальные языки, грамматики, автоматы. СПб.: Университет ИТМО, 2019. 99 с.
18. H2 Database Engine [Электрон. ресурс]. Режим доступа: https://www.h2database.com/html/main.html.

**References**

1. Shefer O.R., Nosova L.S., Lebedeva T.N. Modern methodology for studying programming at a university. Nauchno-tekhnicheskaya informatsiya. Seriya 1: Organizatsiya i metodika informatsionnoy raboty = Scientific and technical information. Series 1: Organization and methodology of information work. 2018; 5: 6-12. (In Russ.)

2. Bazhenova I.V., Pak N.I. Development of an electronic textbook-transformer for teaching programming based on the student's self-cognitive activity. Vestnik Moskovskogo gorodskogo pedagogicheskogo universiteta. Seriya: Informatika i informatizatsiya obrazovaniya = Bulletin of the Moscow City Pedagogical University. Series: Informatics and informatization of education. 2019; 1(47): 20-28. (In Russ.)

3. Moglan D.V. Didactic potential of using algorithms visualization systems in the process of teaching programming. Otkrytoye obrazovaniye = Open Education. 2019; 23; 2: 31-41. (In Russ.)

4. Sharipov F.F., Maradzhabov S.I. A theoretical model of the formation of algorithmic thinking of university students in the process of teaching object-oriented programming. Baltiyskiy gumanitarnyy zhurnal = Baltic Humanitarian Journal. 2017; 6; 3(20): 313-316. (In Russ.)

5. Shkarban F.V. Metodika obucheniya osnovam ob»yektno-oriyentirovannogo programmirovaniya bakalavrov prikladnoy informatiki s ispol'zovaniyem vizual'nykh uchebnykh sred: dissertatsiya kandidata pedagogicheskikh nauk = Methods of teaching the basics of object-oriented programming for bachelors of applied informatics using visual learning environments: dissertation of the candidate of pedagogical sciences: 13.00.02. Volgograd; 2018. 212 p. (In Russ.)

6. Kas'yanov V.N., Kas'yanova Ye.V. Methods and means of teaching programming at the university. Informatika: problemy, metody, tekhnologii: materialy XX mezhdunarodnoy nauchno-metodicheskoy konferentsii = Informatics: problems, methods, technologies: materials of the XX international scientific and methodological conference. 2020: 1989-1998. (In Russ.)

7. Sambo Ye. Kakiye yazyki programmirovaniya uchit' v 2021 (dlya nachinayushchikh) = What programming languages to learn in 2021 (for beginners) [Internet]. Available from: https://videoinfographica.com/programming-languages/. (In Russ.)

8. Milova Ye.A. Metodika obucheniya programmirovaniyu = Programming teaching method [Internet]. Available from: https://pandia.ru/text/79/134/22349.php. (In Russ.)

9. Pedagogika. Urok 4: traditsionnyye metody obucheniya = Pedagogy. Lesson 4: Traditional Teaching Methods [Internet]. Available from: https://4brain.ru/pedagogika/new-methods.php. (In Russ.)

10. Metody obucheniya: ponyatiye, vidy i klassifikatsiya v pedagogike = Teaching methods: concept, types and classification in pedagogy [Internet]. Available from: https://nauka.club/podsovet/metody-obucheniya.html. (In Russ.)

11. Yur'yeva T.A., Chalkina N.A., Lebed' O.A. Application of the case method in teaching bachelors the basics of programming. Pedagogicheskiye nauki = Pedagogical sciences. 2016; 7: 78-82. (In Russ.)

12. Slinkin D.A. Ispol'zovaniye metoda proyektov pri obuchenii programmirovaniyu v kurse informatiki: dissertatsiya kandidata pedagogicheskikh nauk= Using the project method in teaching programming in the course of computer science: dissertation of the candidate of pedagogical sciences: 13.00.02. Yekaterinburg; 2001 166 p. (In Russ.)

13. Lebedeva T.N. The method of projects in teaching students. Aktual'nyye problemy razvitiya srednego i vysshego obrazovaniya: XV mezhvuzovskiy sbornik nauchnykh trudov = Actual problems of the development of secondary and higher education: XV interuniversity collection of scientific papers. Chelyabinsk; 2019: 204-207. (In Russ.)

14. Zhemchuzhnikov D.G. Metodika obucheniya programmirovaniyu, osnovannaya na sozdanii shkol'nikami dinamicheskikh komp'yuternykh igr: dissertatsiya kandidata pedagogicheskikh nauk = Methodology of teaching programming based on the creation of dynamic computer games by schoolchildren: dissertation of the candidate of pedagogical sciences: 13.00.02. Moscow; 2013. 230 p. (In Russ.)

15. Sidorik V.V., Kostina Ye.N. The method of open programs in the study of programming. Nauka – obrazovaniyu, proizvodstvu, ekonomike: materialy 11-y mezhdunarodnoy nauchno-tekhnicheskoy konferentsii = Science - education, production, economics: materials of the 11th international scientific and technical conference. Minsk: BNTU; 2013; 4: 268. (In Russ.)

16. Gladkiy A.V. Formal'nyye grammatiki i yazyki = Formal grammars and languages. Moscow: Nauka; 1973. 368 p. (In Russ.)

17. Lazdin A.V. Formal'nyye yazyki, grammatiki, avtomaty = Formal languages, grammars, automata. Saint Petersburg: ITMO University; 2019. 99 p. (In Russ.)

18. H2 Database Engine [Internet]. Available from: https://www.h2database.com/html/main.html.

**Сведения об авторах**

***Анна Павловна Сергушичева***
*К.т.н., доцент кафедры автоматики и вычислительной техники*
*Вологодский государственный университет, Вологда, Россия*
*Эл. почта: annpas@list.ru*

***Елена Николаевна Давыдова***
*К.т.н., доцент кафедры автоматики и вычислительной техники*
*Вологодский государственный университет, Вологда, Россия*
*Эл. почта: Davidova_EN@mail.ru*

**Information about the authors**

***Anna P. Sergushicheva***
*Cand. Sci. (Engineering), Aassociate Professor in the Department of automation and computer engineering*
*Vologda State University,*
*Vologda, Russia*
*E-mail: annpas@list.ru*

***Elena N. Davydova***
*Cand. Sci. (Engineering), Associate Professor of the Department of automation and computer engineering*
*Vologda State University,*
*Vologda, Russia*
*E-mail: Davidova_EN@mail.ru*