УДК 372.862:378

ДИСТАНЦИОННЫЕ СРЕДСТВА ОБУЧЕНИЯ ОСНОВАМ ПРОГРАММИРОВАНИЯ В ВУЗАХ С ПРИМЕНЕНИЕМ ПРОДУКЦИОННОГО ПОДХОДА

Денисов М.П., Габидуллина Л.И.

ФГАОУ ВО «Казанский (Приволжский) федеральный университет», Казань, e-mail: vmkksumaks@mail.ru

В статье рассматривается структура системы дистанционного обучения основам программирования. Система базируется на средствах Moodle, плагине CodeRunner и продукционных системах. Приведены основные компоненты и модули. Модуль адаптации материалов позволяет реализовать динамическое формирование материала курса в зависимости от знаний студента. Этот процесс использует поиск решений в продукционной базе знаний. Материалы хранятся фрагментами. В процессе поиска решений происходит определение порядка их изучения студентом, а также исключение отдельных фрагментов. Правила в базе знаний создаются для каждого конкретного курса. Знания студента проверяются с использованием тестирования в рамках модуля оценки знаний. В тесты включаются теоретические вопросы и практические задачи. Проверка теоретических вопросов проводится стандартными средствами Moodle. Проверка практических задач происходит в два этапа. На первом этапе код проходит unit-тестирование. Тесты задаются для каждой задачи средствами расширенного плагина CodeRunner. На втором этапе происходит экспертная оценка с использованием продукционной системы. База правил также задается для каждой задачи и включает в себя практические знания преподавателя о возможных ошибках, в том числе несоответствиях условиям, оптимальных решениях, рекомендациях и критериях оценки.

Ключевые слова: дистанционное обучение, Moodle, обучение программированию, экспертные системы, поиск решений, продукционные системы

DISTANCE LEARNING TOOLS FOR BASICS OF PROGRAMMING IN UNIVERSITIES USING RULE-BASED APPROACH

Denisov M.P., Gabidullina L.I.

Federal Autonomous Educational Institution of Higher Education Kazan (Volga region) Federal University, Kazan, e-mail: vmkksumaks@mail.ru

The article discusses the structure of distance learning system for basics of programming. The system is based on Moodle tools, CodeRunner plug-in and rules-based systems. The main components and modules of the system are given. The material adaptation module implements dynamic constructing of course material depending on a student's knowledge. This process uses the inference engine for the rule-based knowledge base. The materials of the courses are stored as fragments. The inference engine sets the order for a student to study these materials and excludes some of the fragments. The rules in the knowledge base are created for each course. Student's knowledge is checked by the knowledge assessment module using tests. Theoretical questions and practical tasks are included into tests. Verification of theoretical questions is processed by standard tools of Moodle. Solutions of the practical tasks are checked with two steps. At the first step the code goes through unit testing. Unit tests are set for each task by tools of the extended CodeRunner plug-in. At the second step the code is gone through the expert assessment using the rule-based expert system. The rules base is created for each task and includes professor's practical knowledge about possible mistakes (includes inconsistency with the conditions of the task), about optimal solutions, about recommendations and assessment criteria.

Keywords: distance learning, Moodle, learning programming, expert systems, inference engine, rule-based systems

В настоящее время область информационных технологий активно развивается – появляются новые подходы к решению прикладных задач. Такое развитие влечет постоянное повышение требований к специалистам, занятым в данной области, в особенности к разработчикам программного обеспечения. Это в свою очередь вызывает необходимость постоянной адаптации учебного процесса под современные реалии, а также увеличение объема материала, который должен усвоить обучаемый. Как следствие, обучение должно продолжаться даже когда у обучаемого нет возможности связаться с преподавателем. Один из вариантов решения проблемы – все больше появляющиеся образовательные структуры, реализующие площадки дистанционного обучения [1]. Использование исключительно таких площадок не позволяет получить студенту ту фундаментальную подготовку, которую дают вузы. Поэтому наилучший результат достигается при внедрении высшими учебными заведениями систем дистанционного обучения в существующие учебные процессы [2; 3].

В связи с вышесказанным появляется актуальность адаптации существующих

или создания новых средств, позволяющих реализовать дополнительный дистанционный доступ обучаемого к материалам, а также позволяющих частично автоматизировать работу педагога в рамках конкретных учебных курсов вузов. В данной статье предлагается подход к частичной автоматизации обучения студентов первого курса «Основам программирования с применением языка C++».

Цель исследования: разработка архитектуры системы дистанционного обучения основам программирования, включающей модуль экспертной оценки решений практических задач.

Материалы и методы исследования

В рамках электронных ресурсов студентам должен быть предоставлен теоретический учебный материал, справочная информация по языку программирования, а также должна быть обеспечена возможность тренировки практических навыков (практикум на компьютере) [4].

Для реализации описанного используется система Moodle [4; 5] и существующие плагины, а также производится их доработка и разработка модулей для работы с продукционными базами знаний.

Выбор системы Moodle обосновывается: развитой системой плагинов, что позволяет расширять возможности системы без ее модификации; большим количеством существующих плагинов, в рамках которых уже реализовано множество возникающих при разработке технических задач; развитой документацией и сообществом системы, что позволяет ускорить решение проблем при разработке; открытым исходным кодом и модульной структурой, что позволяет производить доработки для еще нерешенных задач.

Составление теоретического материала и задач основывается на опыте проведения практических занятий по курсам «Основы программирования» и «Объектно-ориентированное программирование» в КФУ. Предоставление теоретических материалов с удобной навигацией реализовано в Moodle через создание «лекций» и «книг». В рамках «книг» может быть предоставлен лекционный материал, части которого отнесены к иерархическим категориям (главы, разделы). В рамках «лекций» материал может быть представлен в виде отдельных страниц. При определении страницы для перехода студенту может быть задан простой вопрос.

Применение функционала вопросов при навигации по лекциям предполагает некоторую индивидуализацию материала в зависимости от уровня знаний студента. Однако таких простых вопросов недостаточно для определения знаний по программированию. Предлагается формировать материал и последовательность его изучения на основе всестороннего тестирования обучаемого. Одним из важнейших требований к такому тестированию является наличие вопросов, требующих написания программ. Ранее некоторые аспекты такого подхода были реализованы в рамках инструментальной экспертной системы ExPRO [6].

Тестирование по пройденному материалу, предполагающее автоматическую проверку ответа, реализовано в системе Moodle в виде «викторин» с разными типами вопросов, в том числе выбор верного ответа из нескольких вариантов, сопоставление вариантов и т.д. Для вопросов, связанных с написанием программ, существуют различные плагины и расширения системы.

Использование плагина CodeRunner позволяет создавать в «викторинах» вопросы, в ответ на которые студент будет вводить код функции. На основании этого кода система формирует код полной программы. Далее эта программа запускается в песочнице и проверяется тестами, которые задает составитель вопроса. Изначально могут быть созданы тесты, проверяющие точное соответствие вывода функции заданным строкам при передаче ей различных аргументов. Для некоторых аргументов есть возможность задать проверку по случайно сгенерированным значениям [7].

Использование плагина Virtual programming lab позволяет студенту писать и выполнять программы в браузере, а преподавателю запускать проверку программы по заранее заданным тестам [5].

Нужно понимать, что описанные варианты проверки программы – это только unit-тестирование. Такое тестирование сильно зависит от правильности составления самих тестов или случая (если используются случайные значения). Кроме того, оно не позволяет оценить сам код и посоветовать студенту более эффективное или лаконичное решение. Такой подход автоматизирует первичную проверку программы и увеличивает время, которое преподаватель может выделить на объяснение менее тривиальных проблем. Но при этом крайне желательна всесторонняя проверка преподавателем программ, которые успешно прошли тестирование.

В качестве примера могут быть приведены задачи: «пересечение двух целочисленных массивов» и «пересечение двух упорядоченных целочисленных массивов».

Функция, решающая первую задачу, в большинстве случаев подойдет для решения второй задачи. Однако такое решение не будет верным, т.к. не будут использованы особенности упорядоченных массивов и алгоритм будет требовать выполнения гораздо большего числа операций, чем нужно. При этом unit-тестирование покажет, что вторая задача решена верно, если верно решена первая задача. Исключением является подход, при котором студенту будет предоставлен формат и структура вывода всей промежуточной информации, которая также будет проверяться в рамках тестов. Но тогда студент будет выполнять больше механическую работу, т.к. формат вывода будет уже предполагать некоторую конкретную реализацию конкретного алгоритма.

Кроме ошибок несоответствия условиям задачи и неверных ответов, код может содержать и другие ошибки. Самые простые для выявления – синтаксические ошибки. Они определяются любым компилятором. CodeRunner для языка «C++» использует песочницу Jobe с установленным компилятором GCC. Синтаксические ошибки в этом случае будут показаны студенту сразу при запуске проверки вопроса перед проведением unit-тестирования. Однако в процессе компиляции не проверяются ошибки неопределенного поведения, переполнения буфера и другие. Частично эти вопросы могут быть решены с использованием статического анализа кода [8]. Из существующих средств проверки можно выделить Clang. Данные средства могут осуществлять как компиляцию программы, так и проверку с использованием статического анализа, в том числе на этапе компиляции. Это означает, что при использовании Clang вместо GCC программа будет проверяться строже. В итоге студент получит больше сообщений об ошибках. В том числе и об ошибках работы с памятью, которые могут не проявляться на проверяемых тестах или проявляться не всегда.

Тем не менее и такие ошибки полностью не определяются существующими средствами. Это связано со сложностью анализа программ, о которых ничего заранее неизвестно. Но студентам, только начинающим изучение программирования, сначала предлагаются типовые проработанные задачи, что может быть использовано для анализа кода.

Для решения описанных вопросов предлагается интегрировать в систему Moodle подход на основе экспертных систем с использованием процедуры вывода в продукционных моделях [9].

За годы работы преподаватель накапливает знания о том, какие ошибки чаще делают студенты при решении задач; как именно эти ошибки проявляются в коде; на что следует обратить внимание в первую очередь при проверке; какой материал следует объяснить студенту, если найдена та или иная ошибка; какие задачи следует дать для закрепления материала.

Для использования той части этих знаний, которая не представима средствами, описанными выше, может быть использована продукционная модель представления знаний. При таком подходе знания представляются в виде правил вида: ЕСЛИ «условия», ТО «действия». В дальнейшем запускается процесс поиска решений, который для текущих данных и условий определяет список и порядок выполнения «действий».

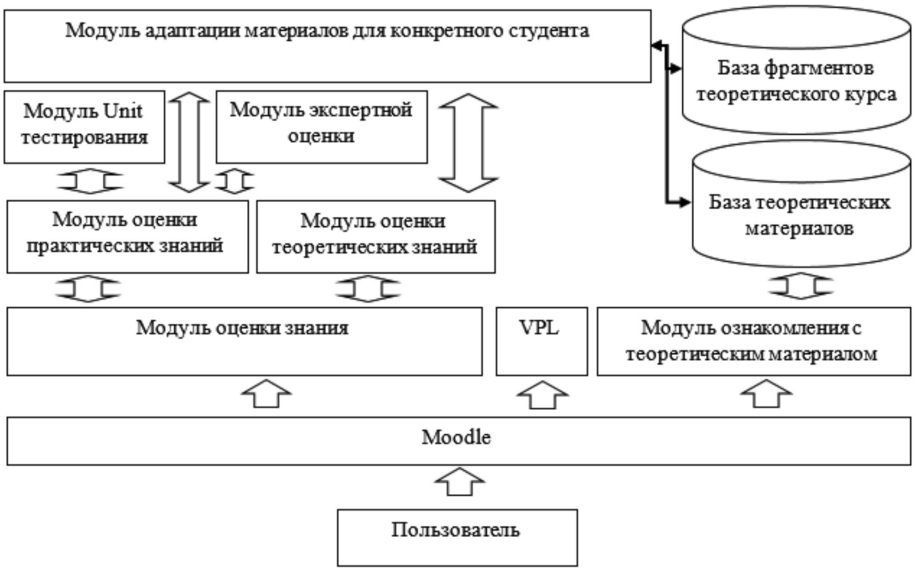
Результаты исследования и их обсуждение

Общая структура предлагаемой системы приведена на рисунке.

При подключении студента к курсу срабатывает процедура клонирования курса с предоставлением доступа только для этого студента. В рамках курса добавлены новые типы ресурсов: «База фрагментов материала»; «База правил адаптации материала». Эти ресурсы видны только составителю курса.

Весь теоретический материал в рамках курса разбит на небольшие фрагменты («База фрагментов теоретического материала», добавленная как ресурс курса типа «База фрагментов материала»), такие как «Оператор сложений», «Оператор инкремента», «Понятие списка», «Понятие наследования». Каждый фрагмент отнесен к одной из категорий. Категории имеют древовидную структуру. Отнесение производится только к листовым узлам. При определении на этих фрагментах материала порядка формируется теоретический материал для конкретного студента («База теоретических материалов») с навигацией по категориям. Практические задачи и отдельные вопросы для оценки знаний студента представлены аналогичным образом («База задач», «База вопросов» входят в «Модуль оценки знаний»), но используется функционал базы вопросов Moodle. В рамках каждой задачи также загружен файл, содержащий «Базу правил оценки решения».

При начале работы студента с курсом материал включает все фрагменты в заданном начальном порядке. При этом сформирована «книга» и первая «викторина». Это формирование происходит в момент клонирования курса.



Общая структура системы обучения программированию для первых курсов вузов

При прохождении первой «викторины», включающей в себя общие вопросы и простые задачи, в системе появляются первые результаты тестирования студента. Далее срабатывает разработанный «Модуль адаптации материалов», включающий в себя экспертную систему с продукционной базой знаний (используются правила из ресурса курса типа «База правил адаптации материала»). То есть запускается процесс поиска решений. В результате этого происходит переоценка фрагментов теоретического материала и тестовых заданий (меняется порядок, принимается решение о включении или удалении отдельных фрагментов). После чего в списке курса студента появляется вторая «книга», которая будет меняться в процессе курса; «лекция» из нескольких страниц, содержащая первые фрагменты из сформированного порядка материалов; вторая «викторина», содержащая новые сформированные вопросы. База правил при этом содержит как общие правила, которые могут быть использованы в различных курсах, так и правила для конкретного курса. Например: «ЕСЛИ задания.решенные.содержат([«быстрая сортировка»]) И задания.решенные.категории ([«сортировка»]).число > 2 ТО фрагменты. категории([[«сортировка», «алгоритмы»]]). метка(закреплено)».

Каждый раз после прохождения последней «викторины» (на прохождение каждой из «викторин» дается одна попытка) снова запускается поиск решений. В результате происходит переформирование второй «книги», формируется следующая «лекция» и «викторина». Материалы из списка участвуют в формировании до тех пор, пока экспертная система не отметит их как закрепленные. Это означает, что каждая следующая «лекция» может содержать материалы предыдущей. В рамках «книги» меняется порядок материала и исключается материал, знание которого студент демонстрирует до включения его в «лекции». Каждая следующая «викторина» наполовину состоит из вопросов, еще не освещенных в рамках «лекций» (первая «викторина» полностью состоит из таких вопросов). Если студент проходит всю «викторину» без ошибок, предупреждений и советов, то следующая «лекция» не формируется и студенту предлагается пройти новую «викторину».

Для оценки практических знаний был расширен плагин CodeRunner. Оценка проводится в два этапа.

На первом этапе решение проходит unit-тестирование. Для компиляции и выполнения кода на C++ используется песочница с установленными инструментами Clang. Переход на второй этап проверки производится в случае, если первый этап пройден успешно (успешно пройдены все unit-тесты, и статический анализ кода не вывел критичных предупреждений). Если на первом этапе возникли ошибки, то задача отмечается как нерешенная. Если возникли не критичные предупреждения (хранится список кодов таких предупреждений), то происходит переход на второй этап с записью кодов предупреждений в рабочую память.

На втором этапе действует разработанная внешняя экспертная система оценки решения, которая интегрируется с CodeRunner. Данная система работает с базой правил, составленных на основе практических знаний преподавателя по каждой конкретной задаче («База правил оценки решения»). Например: «ЕСЛИ максимальная_вложенность_циклов > 1 ТО расчет_суммарного_числа_итераций()»; «ЕСЛИ суммарное_число_итераций > min(размерность_массива_1,размерность_массива_2) ТО предупреждения.добавить(«задача решена для обычных массивов, а не упорядоченных»)». Правила могут изменять балл за решение (в начале второго этапа задача имеет максимальный балл); устанавливать сообщения, которые будут выведены пользователю; передавать факты для записи в рабочую память «Модуля адаптации материалов». Например, если студент решил задачу по нахождению пересечения для обычных массивов, а по условию требовалось для упорядоченных, то задача отмечается как решенная неверно, но в рабочую память «модуля адаптации материалов» будет передана информация о том, что студент освоил работу с обычными массивами на уровне решения задачи о пересечении.

Для отладки программ и проведения собственных экспериментов студент может использовать инструменты VPL, которые устанавливаются в системе без изменений.

Заключение

Таким образом, предлагаемая система позволяет реализовать комплексное решение для обучения студентов основам программирования на C++. При этом появляется возможность дистанционного обучения с адаптацией курса под конкретного студента, более «глубокой» проверкой задач и рекомендациями по оптимизации кода по различным критериям. Кроме того, еще больше уменьшается время, затрачиваемое преподавателем на проверку (кроме unit-тестирования, представленного в ранее существующих решениях, система производит проверку на соответствие условиям задачи, предоставляет первичные рекомендации). При этом преподаватель в любом случае просматривает решения и ответы, если они оказываются не типовыми, и при необходимости дополняет базы знаний экспертных систем. Система с данной архитектурой также может быть реализована для разных материалов, задач и языков программирования. Это потребует создания баз правил для конкретных материалов и задач.

Список литературы

1. Орлова Е.Р., Кошкина Е.Н. Эволюция технологий обучения в аспекте развития информационных технологий (первая половина XX в. – начало XXI в.) // Образовательные ресурсы и технологии. 2017. № 4 (21). С. 68–75.

2. Никольская В.А., Родькина О.Я. Применение современных систем дистанционного обучения в образовательном процессе ВУЗов для практической реализации новых требований стандартов последнего поколения фгос 3+ // Научные ведомости БелГУ. Серия: Гуманитарные науки. 2016. № 28 (240). С. 157–155.

3. Чеботарёв В.Г., Громов А.И. Автоматизация процесса обучения // Бизнес-информатика. 2014. № 4 (30). С. 45–52.

4. Касьянова Е.В. Методы и средства обучения программированию в вузе // Образовательные ресурсы и технологии. 2016. № 2 (14). С. 23–30.

5. Aldo Von Wangenheim. Developing Programming Courses with Moodle and VPL. Bookess Editora LTDA – ME, 2017. 210 p.

6. Юрин А.М., Денисов М.П. Инструментальные средства создания экспертно-обучающих систем // Информационные технологии в обществе, образовании и науке: материалы Международной научно-практическая интернет-конференция (26–27 ноября 2013 г.) / отв. ред. Т.А. Брачун. Магадан: СВГУ, 2014. 291 с.

7. Nedeva V., Kiryakova G. Moodle plugins coderunner for assessment code in training programming. International Conference on Technics, Technologies and Education. 2019. P. 89–94. DOI: 10.15547/ictte.2019.02.005.

8. Меркулов А.П., Поляков С.А., Белеванцев А.А. Анализ программ на языке Java в инструменте Svace // Труды ИСП РАН. 2017. № 3. С. 57–74.

9. Maxim P. Denisov and Arnold M. Jurin. Search of Solutions with the Variable Strategy in Static Expert Systems. International Business Management. 2015. Vol. 9. P. 998–1006.