Е.В. Касьянова

АДАПТИВНАЯ СИСТЕМА ПОДДЕРЖКИ ДИСТАНЦИОННОГО ОБУЧЕНИЯ ПРОГРАММИРОВАНИЮ

ВВЕДЕНИЕ

В не столь далеком прошлом хороший почерк уже являлся гарантией спокойной и обеспеченной жизни до старости. Последние десятилетия характерны ускорением обновляемости технологий и знаний в различных сферах деятельности человека. Поэтому школьного и даже вузовского образования надолго уже не хватает. Сегодня особенно актуальна концепция непрерывного образования на протяжении всей жизни или, как говорят, пожизненного обучения (long-life education). Поиск соответствующей организационной структуры и учреждений образования (особенно образования взрослых), которые обеспечили бы переход от принципа «образование на всю жизнь» к принципу «образование через всю жизнь» — важнейшая проблема XXI века. Открытое образование — это образование, доступное всем. Его развитие неизбежно приведет к существенному пересмотру традиционных методик и технологий учебного процесса, а также к формированию единого открытого образовательного пространства.

Системы дистанционного обучения в настоящее время активно исследуются и развиваются и уже успели пройти путь в пять поколений, начиная от систем обучения по переписке, больше известных в СССР как системы заочного обучения, и кончая системами гибкого обучения и интеллектуального гибкого обучения, определяющими настоящее и будущее дистанционного образования и базирующимися на Web-технологиях.

Выгоды сетевого обучения ясны: аудиторная и платформенная независимости. Сетевое обучающее программное обеспечение, один раз установленное и обслуживаемое в одном месте, может использоваться в любое время и по всему миру тысячами учащихся, имеющих компьютеры, подключенные к Интернету. Тысячи программ сетевого обучения и других образовательных приложений стали доступны в сети за последние годы. Проблема состоит в том, что большинство из них являются не более чем статичными гипертекстовыми страницами.

Появившиеся в последнее время *адаптивные гипермедиа-системы* существенно повышают возможности обучающих систем [1–3]. Целью адап-

тивных систем является *персонализация* гипермедиа-системы, ее настройка на особенности индивидуальных пользователей. Поддержка адаптивных методов в гипермедиа-системах оказывается весьма полезной в тех случаях, когда имеется одна система, обслуживающая множество пользователей с различными целями, уровнем знаний и опытом, и когда лежащее в ее основе гиперпространство является относительно большим. Поэтому области применения адаптивной гипермедиа выходят далеко за границы обучающих систем [1] и включают, например, такие, казалось бы далекие от обучения, области применения гипермедиа-систем, как открытые адаптивные виртуальные музеи [4].

Статья посвящена проекту WAPE, работа над которым ведется в Институте систем информатики СО РАН [5–7]. Цель проекта — разработка адаптивной среды дистанционного обучения WAPE, поддерживающей активное индивидуальное обучение программированию в рамках проблемного подхода и соединяющей возможности адаптивных гипермедиа-систем и интеллектуальных обучающих систем.

Статья начинается с описания основных свойств адаптивных гипермедиа-систем (разд. 1). В разд. 2 дается общее описание системы WAPE с упором на ее возможности, предоставляемые студентам. Возможностям, предоставляемым системой другим пользователям (администраторам, лекторам и инструкторам), посвящен разд. 3. В разд. 4 рассматриваются модель знания курса, а в разделах 5 и 6 — вопросы моделирования знаний студента и использования при этом моделировании механизма сетей Байеса. Моделям тестирования и видам тестов посвящен разд. 7. В разд. 8 и 9 рассматриваются проекты (упражнения и задания) подсистемы CLASS, играющей роль виртуальной семинарской аудитории в системе WAPE. Разд. 10 содержит описание подсистемы PRACTICE, предназначенной для прохождения студентами компьютерного практикума по курсу, поддерживаемому системой WAPE. Виды аннотирования проектов и других указаний, помогающих работе студентов, рассматриваются в разд. 11. Разд. 12 посвящен режимам работы студентов, а вопросы обновления модели знаний студента и развития курса описаны в разд. 13 и 14. В разд. 15 представлен разработанный вузовский курс программирования на базе языка Zonnon. Завершает изложение заключение.

1. АДАПТИВНЫЕ ГИПЕРМЕДИА-СИСТЕМЫ

Класс адаптивных гипермедиа-систем состоит из всех таких гипертекстовых и гипермедиа-систем, которые отражают некоторые особенности пользователя в его модели и применяют эту модель для адаптации различных видимых для пользователя аспектов системы. Таким образом, каждый пользователь имеет свою собственную картину и индивидуальные навигационные возможности для работы с адаптивной гипермедиа-системой.

Отметим, что области применения адаптивных систем далеко выходят за границы обучающих систем. Например, другим важным приложением являются *онлайновые информационные системы* (on-line information systems), а также *онлайновые справочные системы* (on-line help systems). К онлайновым информационным системам относятся, например, электронные энциклопедии, хранилища документов или туристические справочники. Чтобы выдать правильную информацию пользователям с различным уровнем квалификации, этим системам также требуется модель знаний пользователя. Важен также контекст запроса: нужна ли информация пользователю для краткой справки, для разработки презентации, для восстановления знаний? Онлайновые справочные системы принимают во внимание конкретную среду, например, место вызова (контекстно-зависимые справочные системы).

Вместе с тем, обучающие гипермедиа-системы, в которых пользователь или ученик имеет конкретную цель обучения (включая и такую цель, как общее образование), являются типичным приложением адаптивных гипермедиа-систем. В этих системах основное внимание уделяется знаниям обучающихся, которые могут сильно различаться. Состояние знаний изменяется во время работы с системой. Таким образом, корректное моделирование изменяющегося уровня знаний, надлежащее обновление модели и способность делать правильные заключения на базе обновленной оценки знаний являются важнейшей составляющей обучающей гипермедиа-системы. Эти свойства стали особенно важны для Web-систем дистанционного обучения с тех пор, как обучаемые стали учиться в основном самостоятельно (обычно дома). Интеллектуальное и личное содействие, которое могут дать учитель или студент-сокурсник при обычном (аудиторном) обучении, при дистанционном обучении нелегко достижимо. Адаптивность важна для программного обеспечения дистанционного обучения еще и потому, что оно должно использоваться намного более разнообразным по уровню знаний множеством студентов, чем любое «однопользовательское» учебное приложение. Сетевое программное обеспечение, разработанное для одного

класса пользователей (с определенным складом ума), может совсем не подойти другим обучаемым.

Выделяются следующие характеристики пользователя обучающей системы, важные для ее адаптации: цель (или задача) пользователя, уровень его знаний, уровень его подготовки, имеющийся опыт работы пользователя с данной гипермедиа-системой, набор (система) предпочтений пользователя, личностные характеристики пользователя и характеристики пользовательской среды.

Сетевые обучающие системы успешно объединяют технологии адаптации, используемые в интеллектуальных обучающих системах и адаптивных гипермедиа-системах.

Целью различных интеллектуальных обучающих систем является использование знаний о сфере обучения, обучаемом и о стратегиях обучения для обеспечения гибкого индивидуализированного изучения и обучения. Для ее достижения традиционно используются следующие основные технологии: построение последовательности курса обучения, интеллектуальный анализ ответов обучаемого и интерактивная поддержка в решении задач. В группу технологий интеллектуальных адаптаций сетевых обучающих систем входит также технология, получившая название *подбора моделей обучаемых* (или просто *подбор моделей*).

Что касается гипермедиа-систем, то в них адаптация в адаптивной гипермедиа может состоять в настройке содержания очередной страницы (*адаптация на уровне содержания*) или в изменении ссылок с очередной страницы, индексных страниц и страниц карт (*адаптация на уровне ссылок*).

Основные цели (методы) адаптации на уровне содержания гипермедиа-систем — это дополнительные, предварительные и сравнительные объяснения, варианты объяснений и сортировка.

Для достижения целей адаптации: на уровне адаптации разработаны такие техники, как условный и эластичный тексты, варианты страниц и фрагментов, а также технология, основанная на фреймах. Основные цели (методы) адаптации навигации — это глобальное и локальное руководство, поддержка локальной и глобальной ориентаций, управление индивидуализированными представлениями, а основные технологии адаптивной навигационной поддержки — это полное руководство, адаптивная сортировка (упорядочение) ссылок, адаптивное сокрытие ссылок, адаптивное аннотирование ссылок, адаптивное генерирование ссылок и адаптация карты.

2. СИСТЕМА WAPE

Система WAPE ориентирована на поддержку дистанционного обучения и предполагает четыре типа пользователей: студенты, инструкторы, лекторы и администраторы. Все пользователи осуществляют доступ к системе через стандартный Web-браузер, который представляет HTML-документы, предоставляемые HTML-сервером на стороне сервера.

После авторизации пользователя в качестве студента открывается подходящее меню команд.

WAPE система поддерживает три уровня процесса обучения:

- когда студент изучает теоретический материал в некоторой специфической области с использованием гипертекстовых учебников и задачников,
- когда система тестирует концептуальные знания студента, соответствующие изученному теоретическому материалу,
- когда студент под управлением системы выполняет учебные проекты, решая задания и упражнения.

Третий уровень рассматривается как основной в использовании WAPE системы; для того чтобы изучить курс, поддерживаемый WAPE системой, студент должен справиться с набором *проектов* (заданий и упражнений), который инструктор подбирает студенту строго индивидуально.

Другой тип задач, поддерживаемый системой WAPE, — это так называемые *тесты*. В отличие от проектов, решение о выполнении (или невыполнении) которых принимается инструктором, тесты — это вопросы, правильность ответов студентов на которые система оценивает полностью автоматически.

Ориентация на цели обучения является одним из важных свойств нашей WAPE среды. Поскольку мы не хотим фиксировать путь обучения студента или студенческой группы от начала до конца, студенты свободны в определении своих собственных целей обучения и своих собственных последовательностей обучения. На каждом шаге они могут обращаться за помощью к системе, запрашивая подходящий материал, последовательности обучения и советы по примерам и проектам. Если студенту необходим совет по нахождению своего собственного пути обучения, он может спросить систему о следующей подходящей цели обучения.

Система WAPE предназначена для обслуживания многих студентов с различными целями, знанием и опытом. В нашей системе основной упор делается на знание студентов, уровень которого может весьма сильно варьироваться у разных студентов. Более того, состояние знаний студента из-

меняется в процессе работы с системой. Поэтому большое внимание уделяется возможностям адаптивности в нашем проекте.

Система WAPE предоставляет лектору и инструкторам средства для управления мониторингом взаимодействия студентов с системой. Возможно определять те действия студента, которые нуждаются в реакции со стороны преподавателя. Например, когда студент завершает выполнение задания (или упражнения), сообщения посылаются инструктору, отвечающему за мониторинг работы данного студента.

Открытые дискуссии, поддерживаемые WAPE системой, обеспечивают полную виртуальную атмосферу телекласса, включая возможности кооперативного изучения курса вместе с другими студентами и средства кооперативного преподавания для инструкторов и лекторов.

3. ВОЗМОЖНОСТИ АДМИНИСТРАТОРОВ И ПРЕПОДАВАТЕЛЕЙ

Помимо студентов, система WAPE поддерживает три типа пользователей: инструкторы, лекторы и администраторы. Эти типы пользователей различаются как по своим правам, так и по возможностям работы с системой. Каждому типу пользователей соответствует свой интерфейс, поддерживаемый системой.

Интерфейс администратора поддерживает ряд административных функций организации учебного процесса, которые разбиваются на следующие две части.

1. Управление курсами и преподавателями.

В этой части можно осуществлять создание и удаление курсов и преподавателей, а также связывать преподавателей с курсами, потоками и группами в качестве лекторов и инструкторов, а также заменять одного лектора курса или инструктора учебной группы на другого.

2. Управление студентами.

В этой части можно создавать и удалять потоки, группы и отдельных студентов, а также переводить студентов из одной группы в другую.

Интерфейс лектора поддерживает следующие основные функции.

1. Редактирование учебной информации курса.

Сюда входят возможности по включению новых учебников в курс, пополнению гиперкниг курса новыми примерами, созданию или улучшению примеров проектов, по пополнению заданий новыми тестами и эталонными решениями, а также пополнению пространств тестов новыми тестами.

2. Общение со студентами и инструкторами.

Она реализовано в виде общих и преподавательских форумов, администратором которых является лектор. В общих форумах могут участвовать как студенты, так и преподаватели, а в преподавательских — только лектор и инструкторы. После включения лектором некоторой общей темы для обсуждения любой студент и любой инструктор могут написать свое мнение по обсуждаемому вопросу, но у лектора есть возможность удалять и редактировать любые сообщения.

3. Просмотр статистики.

Практически любые действия студента и инструктора заносятся в таблицу статистики и могут быть рассмотрены лектором. В частности, лектор может посмотреть, как часто и сколько времени студенты его потока тратят на обучение, сколько раз и какие тесты они проходили (с фиксацией времени и успеха прохождения). Здесь же он может узнать текущее состояние модели знаний каждого студента, а также какие задачи были им уже решены, а какие нет. Для каждой отдельной задачи можно также узнать количество раз, которое студент пытался ее решить, и посмотреть все варианты решений, которые студент предложил, вместе с комментариями инструктора.

4. Управление мониторингом.

Система предоставляет лектору возможности управления мониторингом взаимодействия студентов и инструкторов с системой. Здесь он может определить те действия студентов и инструкторов, которые нуждаются в его реакции. Каждый раз, когда выбранные действия будут происходить, лектор будет получать соответствующие сообщения.

Интерфейс инструктора поддерживает следующие основные функции.

1. Проверка и оценка заданий студентов.

Каждое задание, решенное студентом, должна быть проверено и оценено инструктором. После того как студент решает задание и проверяет его на имеющихся тестах, он отправляет ее решение на проверку инструктору. Для проверки правильности и оценки качества студенческого решения инструктор может использовать эталонные решения данного задания, если они есть. При этом он должен не только оценить данное решение (отклонить или принять с некоторой положительной оценкой), но и написать комментарий с разъяснением причин такой оценки.

2. Общение со студентами и преподавателями.

Указанные возможности образуют общую и приватную части. Общая часть реализована в виде форумов, администратором которых является лек-

тор, а также форумов, которые администрирует инструктор и создает их для своих студентов. Приватная часть дает инструктору возможность приватного общения с любым своим студентом. Она выполнена в виде гостевой книги, т.е. выводится просто последовательный список сообщений в порядке, обратном к порядку их написания (т.е. последнее сообщение всегда выводится первым). Преподаватель может отвечать на сообщения студентов, и наоборот. Ответы выводятся под соответствующими сообщениями и выделяются другим цветом и шрифтом.

3. Просмотр статистики.

Интерфейс просмотра статистики у инструктора имеет те же самые возможности, что и у лектора. Различие лишь в том, что инструктор может просматривать только свою статистику и статистику студентов своих групп.

4. Управление режимом работы и мониторингом.

Система предоставляет инструктору возможности управления режимом работы каждого своего студента, а также мониторингом взаимодействия этого студента с системой. Здесь он может определить, как знания студента будут влиять на его возможность решать проекты, а также выбрать те действия студента, которые нуждаются в его реакции. Каждый раз, когда студент будет совершать выбранные инструктором действия, соответствующие сообщения будут посылаться инструктору. Например, когда студент завершает выполнение задания (или упражнения), сообщения всегда посылаются инструктору, отвечающему за группу, в которой работает данный студент.

4. МОДЕЛЬ ЗНАНИЙ КУРСА

Центральным объектом модели обучения в WAPE является некоторый учебный курс (или просто курс), который представляет реальный курс, читаемый в некотором университете, реальном или виртуальном.

Каждый курс имеет своего лектора, который создает и поддерживает глоссарий, проекты, тесты, учебники и задачники по курсу (в качестве учебного материала), а также курирует проблемное обучение групп студентов, объединенных в студенческий курс (или поток), осуществляемое под руководством группы инструкторов (ассистентов лектора). Информация о завершении обучения одного потока студентов сохраняется в системе и может использоваться лектором для совершенствования его курса до того, как будет набран новый поток студентов для обучения.

В основе курса лежит его *модель знаний*, которая представляет собой конечное непустое множество *единиц знаний* S с двумя бинарными отношениями U и W на S, удовлетворяющими следующим свойствам для любых p, $q \in S$:

1) $(p, q) \in U$ тогда и только тогда, когда p является составной единицей знания, которая является объемлющей для q;

2) $(p, q) \in W$ тогда и только тогда, когда единица p должна быть изучена до изучения q.

Предполагается, что в модели знаний (S, U, W) пара (S, U) образует лес, а (S, W) является ациклическим графом.

На основе модели знаний курса строится *глоссарий* курса, в котором каждая единица знаний представлена одним (или несколькими) ключевым словом (или фразой) и дополнена множеством ссылок на те элементарные информационные ресурсы курса (элементарные информационные ресурсы учебников и задачников, а также примеры, тесты и проекты), содержание которых относится к данной единице знаний.

Каждый учебник или задачник имеет вид гипертекстовой книги (*гиперкниги*), обладающей иерархической структурой: книга, глава, параграф.

В общем случае гиперкнига содержит последовательность глав, каждая из которых может в свою очередь состоять из последовательности параграфов.

Элементарный информационный ресурс гиперкниги — это либо параграф, либо глава, не содержащая параграфов.

Особую группу элементарных информационных ресурсов гиперкниги образуют *примеры*. Они не содержат теоретического материала курса и служат для его пояснения.

Каждый элементарный информационный ресурс курса представлен отдельной Web-страницей и индексируется некоторым набором единиц знаний, описывающих содержание этого ресурса. Происхождение информационного ресурса несущественно для индексирования, только содержание определяет его информационный индекс.

Пусть $S \neq \varnothing$ — множество всех единиц знаний, $P(S)$ — множество всех подмножеств множества S, а H – множество всех информационных ресурсов, тогда *карта содержания* — это функция $I : H \to P(S) \setminus \varnothing$, которая каждому информационному ресурсу $h \in H$ сопоставляет *информационный индекс* этого ресурса $I(h)$ — непустое множество всех тех единиц знаний, которые связаны с данным информационным ресурсом: если h является тестом или проектом, то $I(h)$ состоит из тех единиц знания, которые используются в h, а если h является информационным элементом гиперкниги, то $I(h)$

состоит из тех единиц знания, объяснение которых в той или иной степени содержится в данном информационном ресурсе *h*.

5. МОДЕЛИРОВАНИЕ ЗНАНИЙ СТУДЕНТА

Знания студента *x* моделируются *вектором знаний* $K(x) = (p_1, \ldots p_n)$, где $n = |S|$ — число единиц знаний в модели знаний курса, а для любого i, $1 \le i \le n$, $p_i = p(s_i / E_i)$ — условная вероятность, описывающая предположение системы о том, что студент *x* обладает знанием единицы знания $s_i \in S = \{s_1, s_2, \ldots, s_n\}$ на базе тех свидетельств E_i, которые система собрала о знании студентом *x* единицы знания s_i в процессе мониторинга его работы над курсом.

Каждый элемент p_i вектора знаний $K(x)$ выражает степень знания единицы знания s_i, которым обладает студент *x* в данный момент.

Мы используем четыре степени такого знания:

- знания эксперта в области данной единицы знания — обозначаются *E* (отличные знания),
- знания продвинутого пользователя – обозначаются *F* (несколько затруднительных моментов в понимании s_i, но в целом понятие вполне усвоено),
- знания начинающего — обозначаются *A* (много затруднительных моментов, понятие s_i плохо усвоено),
- знания новичка — обозначаются *N* (пользователь еще не готов для работы с данным понятием, оно им не усвоено).

Таким образом, единицы знания являются с одной стороны понятиями, описывающими предметную область курса, а с другой — случайными переменными с четырьмя дискретными значениями *E*, *F*, *A* и *N*, кодирующими степень знаний данного студента.

Свидетельства, получаемые системой в процессе мониторинга действий студента, изменяются со временем. В типичном случае знание студента возрастает во время работы с курсом, хотя недостаток знания также воспринимается системой как свидетельство. Поскольку каждый отслеживаемый результат производимого наблюдения за студентом сразу же заносится в свидетельства, вектор знаний в любой момент дает моментальный снимок текущего уровня знаний студента.

Обновление свидетельств о студенте происходит только при выполнении им тестов и проектов. При этом никакое тестирование не позволяет студенту получить от системы оценки “отличные знания”, и он может дос-

тичь уровня знаний эксперта только за счет успешного выполнения проектов.

Пусть $P = (p_1, \ldots p_n)$ и $Q = (q_1, \ldots q_n)$ — два произвольных вектора знаний, тогда:

- $P \geq Q$, если $p_i \geq q_i$ для любого I;
- $min\ (P,\ Q)$ – это такой вектор знаний $(w_1, \ldots w_n)$, что $w_i = min(p_i, q_i)$ для любого I;
- $max\ (P,\ Q)$ – это такой вектор знаний $(w_1, \ldots w_n)$, что $w_i = max(p_i, q_i)$ для любого i.

6. МЕХАНИЗМ СЕТЕЙ БАЙЕСА: ВЫЧИСЛЕНИЕ ВЕРОЯТНОСТЕЙ

Сети Байеса являются мощным инструментом вывода в графах с зависимыми случайными вершинами. Мы используем такую сеть для вычисления распределения вероятностей для каждой единицы знаний и, следовательно, для вычисления векторов знаний пользователей.

Сеть Байеса представляет собой ориентированный ациклический граф со следующими свойствами (рис. 1):

- каждая вершина графа представляет случайную переменную;
- имеется дуга из X в $Y \neq X$ в каждом случае, когда Y зависит от X;
- каждая вершина помечена таблицей условной вероятности, которая определяет воздействие на вершину ее непосредственных предшественников.

Чтобы построить сеть Байеса, которая вычисляет распределение вероятностей для каждой единицы знаний курса для конкретного студента, требуется выполнить два действия. Во-первых, сгенерировать некоторый ациклический граф, имеющий единицы знаний в качестве вершин и обучающие зависимости между ними в качестве дуг, и, во-вторых, определить таблицы вероятности для всех вершин.

Для наших целей мы строим сеть Байеса со случайными переменными, которые дают распределение вероятностей для вычисления знаний пользователя. Мы используем в качестве вершин случайные переменные с четырьмя дискретными значениями из множества $\{E, F, A, N\}$ и проводим дугу из вершины X в вершину Y в том и только том случае, когда $Y < X$ и не существует такого Z, что $Y < Z < X$, где $Y < X$, если Y зависит от X. т.е. если $(X, Y) \in W$ или $(Y, X) \in U$.

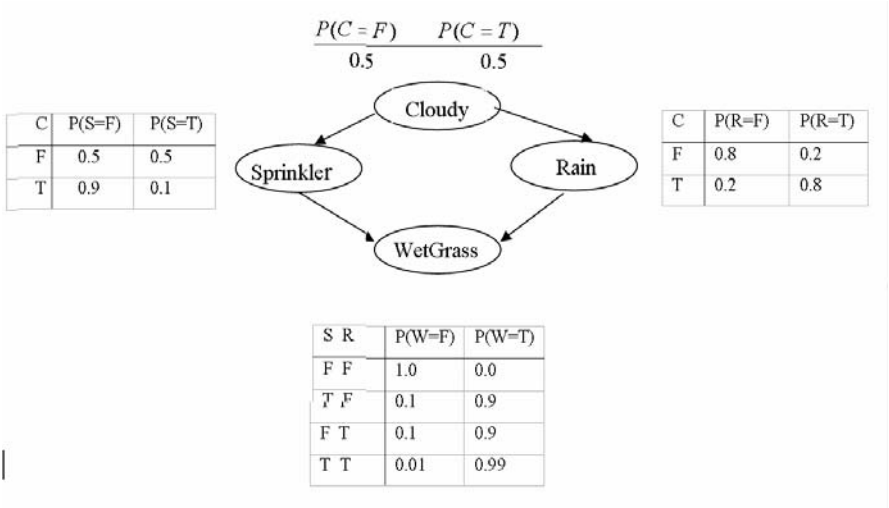

Рис. 1. Простая сеть Байеса, вершины которой представляют случайные логические переменные, принимающие значения: T (истина) или F (ложь)

Известно, что точный вывод в сетях Байеса является *NP*-трудной проблемой [8]. Вместе с тем существуют линейные по времени алгоритмы для тех сетей Байеса, базовые графы которых не содержат циклов, а также существует несколько методов обработки таких не полностью ориентированных циклов: кластеризация, кондиционирование и стохастическая симуляция.

7. МОДЕЛЬ ТЕСТИРОВАНИЯ

Тесты – это вопросы студенту, ответы на которые оцениваются системой без участия преподавателя.

По виду различается три типа тестовых вопросов: выборные, мультивыборные и наборные. Выборный тест предполагает выбор одного варианта ответа из предлагаемого списка (рис. 2), а мультивыборный — нескольких вариантов (рис. 3). Наборный тест требует, чтобы студент ввел правильный ответа в специальное текстовое поле (рис. 4). По существу, каждый мультивыборный тест предлагает студенту список вариантов отве-

тов, некоторое (возможно пустое) множество из которых является правильными.

Мы используем компьютерное тестирование со случайным позиционированием ответов для выборных и мультивыборных тестов, чтобы ответы по номерам позиций могли привести к ошибкам студентов, заучивающих позиционные номера правильных ответов. Таким образом, вместо запоминания вопросов и номеров строк вариантов правильных ответов студенты вынуждены в вопросах и ответах на вопросы фокусироваться на содержательной стороне дела. Поэтому подход к тестированию, используемый нашей системой, включает случайную генерацию вопросов в заданной предметной области и случайное расположение вариантов ответов.

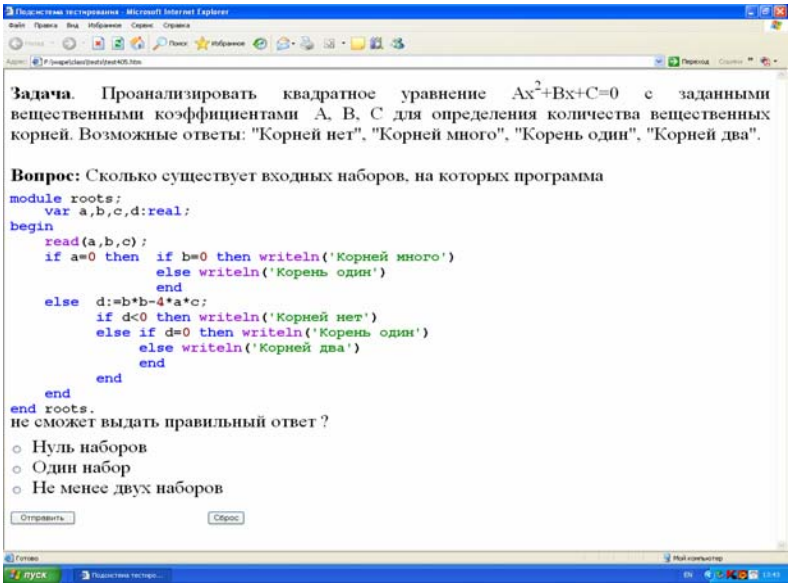

Рис. 2. Выборный тест

Предполагается, что в процессе тестирования студент, как правило, получает не один, а целую серию тестов, ответы на которые оцениваются суммарно. Такой подход еще более усложняет работу тех студентов, которые пытаются пройти тестирование путем заучивания позиционных номеров правильных ответов.

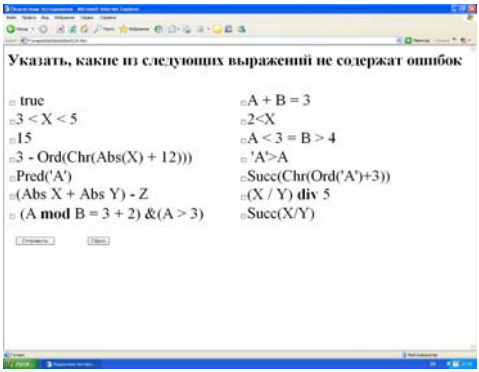

Рис. 3. Мультивыборный тест

Каждый тест измеряет вербальные или аналитические умения, относящиеся к конкретной области изучаемого курса. Различные временные ограничения ассоциируются с каждым вопросом. Мы различаем три вида тестов: вербальные, качественные и аналитические.

Вербальный тест определяет некоторую конкретную концепцию определения и имеет временное ограничение в 60 секунд. Вербальный тест проверяет способность анализировать и оценивать написанный материал и синтезировать информацию, полученную из него, для анализа отношений между отдельными частями предложений и для распознавания отношений между словами и понятиями.

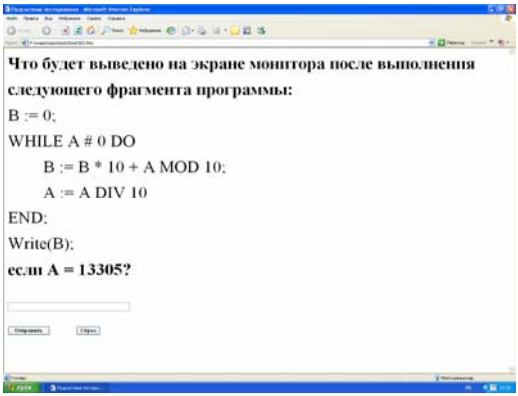

Рис. 4. Наборный тест

В случае *качественных* тестов, в которых должны быть объяснены более сложные понятия, обычно ответ ожидается между 120 и 240 секундами. Качественный тест проверяет базисные умения и понимание элементарных понятий, а также способность студента качественно мыслить и решать задачи в качественной постановке или объяснять более сложные понятия.

Для *аналитического* теста, при котором должно быть объяснено некоторое трудное понятие, ответ обычно ожидается внутри временного диапазона в 360 — 480 секунд. Аналитический тест проверяет способность студента понимать структурированные множества отношений, выводить новую информацию из множеств отношений, анализировать и оценивать аргументы, идентифицировать центральные вопросы и гипотезы, делать правильные выводы и опознавать хорошо обоснованные объяснения. Тесты аналитического вида обычно измеряют умение рассуждать, связанное с несколькими разделами изучаемого курса.

С каждой единицей знания курса связываются два пространства тестов, первое из которых используется для тестирования студентов, претендующих на знание данной единицы знаний на уровне “начинающего”, а второе — на уровне “продвинутого” студента.

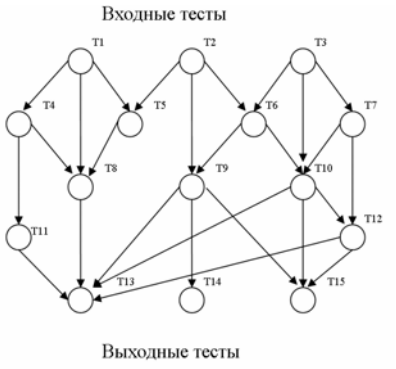

Рис. 5. Пространство из 15 тестов

Пространство тестов — это ациклический ориентированный граф $G = (X, V)$, вершинами которого являются тесты, а дуги отражают последовательность их прохождения: $(p, q) \in V$ тогда и только тогда, когда после выполнения студентом теста p ему может быть предложен тест q.

В пространстве тестов выделяются множества входных и выходных тестов. Тест p является входным, если он не имеет предшественников в G, и является выходным, если p не имеет преемников в G.

С каждым пространством тестов лектор связывает две константы границ знаний C_1 и C_2, где C_1 — минимальный уровень знаний для начинающего студента в данном пространстве тестов, а C_2 — минимальный уровень знаний для продолжающего студента для данного пространства тестов. По умолчанию предполагается, что пространство тестов для начинающих студентов состоит из вербальных и качественных тестов и имеет $C_2 > 1$, а пространство для продвинутых студентов состоит из аналитических тестов, и в нем $C_2 = 1$.

Путь тестирования в пространстве тестов — это путь по G от некоторого входного теста до некоторого выходного. В процессе тестирования студенту предлагается серия тестов $T = \{p_1, \ldots, p_n\}$, образующих путь тестирования $(p_1, \ldots, p_n)$, генерируемый случайным образом. Например, последовательность (T3, T6, T9, T13) образует один из возможных путей тестирования для пространства тестов рис. 5.

Пусть студент выполнил серию тестов T и

$$\gamma_x = \left(\sum_{t \in T} \alpha(t)\right) / \left(\sum_{t \in T} \beta(t)\right), \text{ где}$$

$\alpha(t)$ — число правильных ответов данного студента в серии тестов T,

$\beta(t)$ — максимальное число правильных ответов для серии тестов T, которые мог бы дать студент. Заметим, что в случае выборного или наборного ответа студент может дать только один правильный ответ, а в мультивыборном тесте максимальное число правильных ответов совпадает с длиной списка вариантов ответов.

Студент x демонстрирует на серии тестов T знания

- новичка, если $\gamma_x < C_1$,
- начинающего студента, если $C_1 \le \gamma_x < C_2$, и
- продвинутого студента, если $C_2 \le \gamma_x$.

8. ПРОЕКТЫ СИСТЕМЫ CLASS

Системы CLASS и PRACTICE являются основными подсистемами системы WAPE.

Подсистема CLASS является виртуальной семинарской аудиторией, которая предназначена для получения опыта программирования на некотором

языке высокого уровня. Это среда проблемного обучения, в которой студенты группы под руководством инструктора обучаются конструировать корректные и эффективные программы для решения достаточно простых задач.

Любой курс, поддерживаемый системой CLASS, включает набор проектов, предназначенных для выполнения студентами. Каждый проект P — это упорядоченное множество однотипных задач $\{P_1, P_2,\ldots,P_n\}$, где $n \geq 0$ — количество вариантов проекта P.

В системе CLASS имеются проекты двух видов: упражнения и задания, различающиеся по виду предполагаемого решения.

Каждое *упражнение* P — это набор вопросов $\{P_1, P_2,\ldots,P_n\}$, предполагаемые ответы на которые не рассматриваются системой как исполняемые программы.

В отличие от упражнений любое *задание* P — это задача на программирование; так что решить вариант i задания P, т.е. P_i, — это значит написать исполняемую программу.

По уровню сложности решаемых задач различаются проекты двух типов: стандартной и повышенной сложности.

Проекты стандартной сложности не имеют пометок и образованы из тех задач, которые проверяют понимание студентом изложенного в учебнике теоретического материала. В частности, к проектам стандартной сложности относятся те упражнения и задания, которые направлены на обучение студентов использованию новых языковых конструкций совместно с уже усвоенными, а также на отработку описанных в учебнике схем решения задач.

Другой тип проектов выделен специальной пометкой «проект повышенной сложности» и содержит задачи, которые добавляют новую или требующую размышлений информацию к материалу, изложенному в учебнике. Такие проекты заставляют студентов обдумывать некоторую важную концепцию, относящуюся к теоретическому материалу учебника, или находить ответ на вопрос, который может возникать у студента во время чтения учебника.

Каждый проект стандартной сложности требует, чтобы у студента, допускаемого к его выполнению, уровень знания для всех тех единиц, которые образуют индекс данного проекта, был бы не ниже уровня продвинутого студента, а любой проект повышенной сложности может начать решать только тот студент, который является экспертом во всех единицах знаний, входящих в индекс данного проекта.

В каждом проекте можно выделить некоторый вариант в качестве примера. Решение такого варианта, который получает номер 0, оформляется в

виде гипертекстового документа, содержащего формулировку задачи, ее решение и обоснование решения. В частности, в случае задания он содержит описание задания, текст программы и комментарии к программе, поясняющие и обосновывающие данное решение.

Для поддержки данной возможности используются виртуальные и пользовательские номера вариантов проектов.

Виртуальные номера используются при описании проекта лектором на языке описания проектов, а пользовательские — это номера, в терминах которых варианты данного проекта используются студентами в курсе, поддерживаемом системой CLASS. В частности, вариант, имеющий в качестве пользовательского номера нуль, является примером данного проекта.

Лектор, описывая проект в терминах виртуальных номеров, одновременно задает функцию преобразования виртуальных номеров в пользовательские. Указанная функция используется подсистемой генерации проектов каждый раз, когда происходит обращение к проекту за формулировкой некоторого его варианта или за его тестами.

9. ЗАДАНИЯ СИСТЕМЫ CLASS

Каждое задание системы CLASS предполагает составление программы, читающей входные данные из файла *input.txt* и записывающие результаты своего исполнения (выходные данные) в файл *output.txt.*

С каждым заданием α связывается множество эталонных решений $S(\alpha)$, множество тестов задания $T(\alpha)$, множество правил вывода знаний студента $R(\alpha)$ и два вектора знаний: $P(\alpha)$ — вектор предварительных знаний и $F(\alpha)$ — вектор итоговых знаний.

Эталонные решения $S(\alpha)$ — это прокомментированные программы решения задания α. В комментариях описываются особенности данного решения задания α и приводится мотивированная его оценка. Эталонные решения используются преподавателями (лекторами и инструкторами) и не доступны студентам.

Тесты задания $T(\alpha)$ используются системой для проверки правильности понимания студентом условия задания и автоматической проверки правильности составленной студентом программы. Каждый тест из множества $T(\alpha)$ — это пара, состоящая из входных и соответствующих выходных данных.

Множество тестов $T(\alpha)$ задания α распадается на два непересекающихся подмножества $T_1(\alpha)$ и $T_2(\alpha)$, т.е. $T(\alpha) = T_1(\alpha) \cup T_2(\alpha)$ и $T_1(\alpha) \cap T_2(\alpha) = \varnothing$,

называемых тестами *правильности понимания задания* и тестами *правильности программы* (*решения задания*).

Перед тем как начать решать задание, студент должен подтвердить, что он правильно понимает условие задания. Для этого он должен для каждого теста $t \in T_1(\alpha)$ правильно ввести выходные данные теста t по заданным входным. Тесты из $t \in T_1(\alpha)$ могут использоваться студентом также на начальном этапе отладки программы решения задания α.

Перед тем, как студенту будет разрешено отдать программу (решение задания α) на проверку инструктору, он должен продемонстрировать системе ее правильность на всех тестах из $T_2(\alpha)$. Для этого его программа должна на входных данных каждого теста $t \in T_2(\alpha)$ выдавать выходные данные, которые соответствуют ожидаемым.

Каждому тесту $t \in T(\alpha)$ могут быть сопоставлены два текста: диагностическое сообщение и общедоступный комментарий. *Диагностическое сообщение* — это тот текст, который выдается студенту тогда, когда программа студента является неправильной относительно теста t. В тексте комментария указываются предполагаемые ограничения на рабочие характеристики программы для данного теста.

Каждое *правило вывода* знаний студента из множества $R(\alpha)$ имеет вид: $B \to N$, где

- B — логическое выражение, содержащее литералы вида x или $\neg x$,
- $x \in T_2(\alpha)$,
- N — вектор знаний.

Правило $B \to N$ работает в два этапа.

1. На первом этапе в B вместо каждого вхождения теста $x \in T_2(\alpha)$ подставляется либо значение *true,* если на входных данных теста x программа выдает предполагаемые результаты, либо значение *false* в противном случае.
2. На втором этапе вычисляется выражение B, и, если оно принимает истинное значение, то происходит перевычисление вектора знаний $K(x)$ студента x, решающего данное задание, по следующему правилу $K(x) := min(N, K(x))$.

Вектора предварительных и итоговых знаний $P(\alpha)$ и $F(\alpha)$, приписанные заданию α, имеют следующий смысл.

Предполагается, что студент x может приступить к выполнению проекта α только в том случае, если $K(x) \geq P(\alpha)$.

Успешное выполнение проекта α приводит к перевычислению вектора знаний $K(x)$ студента x, решающего данное задание, по следующему правилу $K(x) := max(F(\alpha), K(x))$.

10. СИСТЕМА PRACTICE

Система PRACTICE является виртуальной лабораторией, предназначенной для прохождения студентами компьютерного практикума по курсу. Основная цель, которая ставится перед студентом при выполнении индивидуальных заданий (интегральных проектов), составляющих компьютерный практикум, — это практическое освоение всех этапов разработки надежной и наглядной интерактивной (диалоговой) программы для компьютерного решения некоторой нетривиальной задачи, требующей разработки алгоритма, обработки сложных структур данных и создания дружественного интерфейса.

С каждым интегральным заданием α связывается множество эталонных решений $S(\alpha)$ и два вектора знаний: $P(\alpha)$ — вектор предварительных знаний и $F(\alpha)$ — вектор итоговых знаний. Они имеют тот же смысл, что и у заданий системы CLASS, с тем отличием, что *эталонные решения* $S(\alpha)$ — это прокомментированные решения задания α, которые имеют вид гипертекстовых отчетов (см. ниже).

Тематика индивидуальных заданий для компьютерного практикума определяется, в первую очередь, всеми видами работ, которые должен освоить студент, чтобы научиться создавать качественные (эффективные, наглядные и надежные) нетривиальные программы. В большинстве случаев задача, решаемая во время выполнения индивидуального задания, — это задача невычислительного характера, имеющая краткую и точную (содержательную) формулировку и допускающая большое разнообразие решений.

При выполнении проектов практикума студенты должны интегрировать все, что они изучили ранее (при работе в системе CLASS), а также получить навыки, являющиеся базовыми для разработки программного обеспечения на всех уровня и для программирования как дисциплины. Это включает использование в той или иной мере формальных методов анализа задач и конструирования программ с упором на создание эффективных и надежных программ, которые удовлетворяют заданным спецификациям и поддерживают дружественный интерфейс.

Работая в виртуальной лаборатории, студент изучает методический материал, содержащийся в задачнике курса, тестирует свои знания теоретического материала, прочитанного им, или решает интегральные проекты.

В процессе решения проекта в системе PRACTICE студент составляет гипертекстовый отчет, который включает:

1) формулировку задачи;

2) описание программы для пользователя, ее внешнюю спецификацию, т.е. описание способа задания входных данных, вида результатов программы при заданных входных данных и сценария диалога в процессе исполнения программы;

3) словесное описание алгоритма и обоснование его правильности и эффективности;

4) текст программы;

5) описание тестового набора и его обоснование.

Для удобства все интегральные задания разбиты на три группы:

- обычной сложности (они не помечены),
- повышенной сложности,
- пониженной сложности.

При оценке сложности задания рассматриваются три показателя:

- сложность структур данных,
- сложность вычислений,
- изобретательность.

В показатель “изобретательность” включаются такие свойства задания, как непривычность для студента понятий, используемых в задании, сложность извлечения из определений тех свойств, на которых должен базироваться алгоритм решения задания, а также сложная связь между структурами данных и вычислениями. Каждый из указанных трех показателей задания оценивается в баллах 0, 1 или 2. Пометку “задание повышенной сложности” получают задания, набирающие в сумме по трем показателям всего 1 балл, а пометку “задание пониженной сложности ”— задания, суммарный балл которых больше 3.

11. АННОТАЦИЯ ПРОЕКТОВ

Для вычисления релевантности доступного по ссылке проекта текущему уровню знаний студента используется простая метафора «семафора» для аннотаций. Ссылки на проекты, которые должен решить в курсе студент, помечены одним из трех цветовых кружков: готов-для-решения (зеленый

кружок рядом с текстом ссылки), не-готов-для-решения (красный кружок) или уже-решен (серый кружок), что позволяет студенту выбрать подходящие проекты.

Студенту часто требуется информация по определенным темам, но не хватает предварительных знаний для понимания этой информации. К примеру, студент хочет работать над проектом построения ветвящихся программ, но не понимает таких тем, как логические выражения; в таком случае ему не стоит начинать чтение сразу со страниц, содержащих описание условных операторов. Для поддержки студента система сравнивает его текущий уровень знаний с необходимым для понимания рассматриваемой темы. Если у студента не хватает некоторых предварительных знаний, система может сгенерировать последовательность информационных элементов (след), который направляет его обучение в рамках данной темы.

Генерация такого следа реализована как алгоритм обхода в глубину, который проверяет оценку системой знаний студента о тех единицах знаний, которые предварительно необходимы для его текущей цели. Алгоритм проверяет, все ли предварительные знания достаточно усвоены студентом. Если нет, алгоритм находит единицы знаний, нуждающиеся в изучении. После этого генерируется последовательность подцелей и последовательность информационных элементов, которая последовательно направляет работу студента через необходимые темы вплоть до выбранной им.

Если студент хочет получить более конкретные указания во время работы с системой, он может запросить у системы следующий разумный шаг обучения. Этот запрос на прямое руководство выполняется путем определения подходящей цели обучения, зависящей от текущего состояния знаний студента и состояния его проектов. Цель определяется как набор единиц знаний. Чтобы определить следующую подходящую цель обучения, вычисляется последовательный след, покрывающий все нерешенные проекты студента. Для каждого элемента этого следа проверяется оценка системой уровня знаний студента. Если студенту не хватает знания какой-либо единицы знаний, тогда она предлагается в качестве следующей подходящей цели.

12. РЕЖИМЫ РАБОТЫ СТУДЕНТА

Инструктор может управлять тем, как знания студента влияют на возможность его работы с системой.

В *жестко контролируемом* режиме студент x не может выполнять ни один из тех проектов, для которых он не имеет достаточно знаний. Другими словами, при этом режиме разрешается студенту x приступить к выполнению некоторого проекта α только в том случае, когда $K(x) \geq P(\alpha)$.

В *слабо контролируемом* режиме студент x может выполнять не только те проекты α, для которых у него хватает знаний (т.е. $K(x) \geq P(\alpha)$), но и любой такой проект α, что для любого i либо $K(x)(i) \geq P(\alpha)(i)$, либо $K(x)(i) = F$ и $P(\alpha)(i) = E$.

В *свободном режиме* студент может выполнять любой проект.

При этом вне зависимости от того режима, который установлен студенту x инструктором, студент всегда получает предупреждение о недостаточности своих знаний, если он пытается начать выполнять такой проект α, для которого у него не хватает компетентности в некоторой единице знаний i (т.е. $K(x)(i) < P(\alpha)(i)$). Одновременно с предупреждением студенту демонстрируются те единицы знаний (вместе с их уровнями), знание которых он должен повысить до выполнения данного проекта.

13. ОБНОВЛЕНИЕ МОДЕЛИ ЗНАНИЙ СТУДЕНТА

Многие адаптивные системы фиксируют факт чтения пользователем определенной информации и на этой основе обновляют оценку его знаний. Некоторые из них учитывают время чтения или последовательность прочитанных страниц для углубления этой оценки. Хотя это оправданный подход, его недостатком является трудность измерения знания, приобретенного пользователем во время чтения Web-страницы. Вместо этого мы используем для обновления модели знаний только тесты и проекты. Это мотивировано проблемным подходом к обучению в тех курсах, на поддержку которых ориентирована система WAPE.

Предполагается, что после чтения того или иного теоретического материала система организует обратную связь. Студент последовательно указывает те темы, которые были целью изучения данного материала. Для каждой из этих тем (единиц знаний) студент переоценивает свои изменившиеся знания, выбирая одну из двух категорий: «тема была несложной — я овладел материалом без труда», «тема была непростой — у меня были некоторые проблемы в понимании». Эти категории соответствуют знаниям продвинутого студента и начинающего. После этого система генерирует индивидуальный набор тестов, успешное прохождение которых позволяет студенту изменить желаемым образом оценку своих знаний.

Обновление модели знаний автоматически происходит во время работы студента над некоторым проектом каждый раз, когда решение студента не проходит некоторый тест, а также тогда, когда инструктор завершает оценку текущего (посылаемого инструктору на проверку) варианта решения проекта студентом.

Студент имеет право в любой момент понизить оценку уровня своих знаний любой единицы знаний, а также попытаться повысить её. В последнем случае в зависимости от его претензий он получает индивидуальный набор тестов, сгенерированный системой, успешное прохождение которых позволяет студенту изменить желаемым образом оценку своих знаний.

14. РАЗВИТИЕ КУРСА

В процессе функционирования предполагается развитие каждого курса, поддерживаемого системой, по следующим направлениям:

1) включение новых учебников в курс,

2) создание или улучшение примеров решения проектов,

3) пополнение заданий новыми тестами и эталонными решениями.

Указанные усовершенствования осуществляются лектором и не требуют от него каких-либо специальных программистских знаний. При включении некоторого нового гипер-учебника в курс нет необходимости его преобразования. Единственное, что требуется — это построение информационных индексов элементарных информационных элементов, составляющих данный учебник. Эти индексы можно строить как автоматически (путем поиска вхождений ключевых фраз), так и вручную (путем просмотра учебника лектором). В основе других направлений развития курса лежат наработки студентов, которые при включении их преподавателем в курс могут подвергнуться редактированию и комментированию.

Для описания новых проектов можно использовать следующий специальный *язык задания проектов* (ЯЗП), позволяющий задавать одновременно все варианты проекта компактным образом.

Ниже для описания синтаксиса языка ЯЗП используется Расширенный Бекуса-Наура Формализм (Extended Backus-Naur Formalism, EBNF), который характеризуется следующими свойствами:

- альтернативы разделяются символом |;
- скобки [и] обозначают факультативность выражения в скобках;
- скобки { и } обозначают повторение (возможно 0 раз);
- скобки (и) используются для формирования групп элементов;

- нетерминальные символы начинаются с прописной буквы (например, Statement);
- терминальные символы либо начинаются со строчной буквы, либо записывается целиком прописными буквами (например, BEGIN), или представляются в виде строк (например, ":=").

Правила описания проекта на языке ЯЗП имеют следующий вид, где Символ — это произвольный символ, отличный от знака # :

Проект={Условие Образец}.
Образец={{Символ}{#Выражение}{Символ}}.
Условие= ИстинноеУсловие | ВычисляемоеУсловие.
ВычисляемоеУсловие = #<Выражение, Интервал> |
#<Выражение, Число>.
ИстинноеУсловие = ##.
Выражение= Номер | Номер * Число | Номер DIV Число.
Интервал=ОткрывСкобка Число, Число ЗакрСкобка.
ОткрывСкобка = (| [.
ЗакрСкобка =) |].
Число = {Цифра}.
Номер = # Число.

Каждый Проект на языке ЯЗП представляет собой фактически последовательность строк вида Условие Образец, по которым строятся все конкретные варианты проекта по следующим правилам. Вначале *N* подставляется в каждое Выражение вместо Номера, преобразуя их в константные. Затем полученные константные выражения вычисляются, и вычисленные значения заменяют вхождения выражений. Данная подстановка преобразует образцы в строки факультативных и обязательных частей текста формулировки соответствующего варианта. Обязательная часть образуется из образца, которому предшествует ИстинноеУсловие. Условие, соответствующего образца, при котором данная факультативная строка должна включаться в формулировку конкретного варианта проекта, — это принадлежность вычисленного значения выражения интервалу, заданному в вычисляемом условии, или его равенство заданному значению.

15. КУРС ПРОГРАММИРОВАНИЯ НА БАЗЕ ЯЗЫКА ZONNON

Для включения в систему нами разработан курс начального обучения программированию на базе нового языка Zonnon [9, 10], работа над которым ведется в Цюрихском институте информатики. Язык Zonnon задуман

как дальнейшая эволюция хорошо известного и широко применяемого на западе в учебных целях языка Оберон, являющегося преемником языков Паскаль и Модула-2. Язык Zonnon сохраняет стремление к простоте, ясному синтаксису и независимости концепций, а также уделяет внимание параллельности и легкости композиции и выражения. Унификация абстракций является стержнем проектирования языка Zonnon, и она отражается в его концептуальной модели, основанной на модулях, объектах, определениях и реализациях. Язык Zonnon содержит такие новые черты, как активность в объектах, основанный на межобъектном взаимодействии диалог, перегрузка операций и обработка исключительных ситуаций. Он специально разрабатывается как платформно-независимый язык, и его первая реализация выполнена для платформы .NET.

Разработанный курс базируется на ряде методических и технологических принципов, основными из которых являются следующие:

- принцип концентрического изложения материала, когда обучаемый осваивает языковые средства и приемы программирования постепенно, слой за слоем;
- принцип обучения конструированию программ на подробно комментированных образцах решения тщательно подобранных задач;
- принцип доказательного программирования, когда программа строится вместе с доказательством ее правильности;
- принцип пошаговой разработки программ, когда программа строится из формальной спецификации задачи с помощью мелких формально проверяемых шагов преобразования;
- принцип модульного программирования, позволяющий проектировать, разрабатывать и собирать программу по частям и с использованием библиотек уже готовых частей;
- принцип объектно-ориентированного программирования, позволяющий разработчикам программ легко создавать все более сложные приложения с помощью инкапсуляции, наследования и полиморфизма.

Подготовлены два гипертекстовых учебных пособия «Введение в программирование» и «Практикум по программированию», поддерживающие курс, которые размещены на сайте русскоязычной библиотеки учебных курсов международной программы MSDN Academic Alliance [11].

ЗАКЛЮЧЕНИЕ

В статье описывается архитектура адаптивной среды дистанционного обучения, которая поддерживает активное индивидуальное обучение программированию в рамках проблемного подхода и соединяет возможности адаптивных гипермедиа-систем и интеллектуальных обучающих систем. Среда нацелена на поддержку обучения конструированию алгоритмов и -- разработки эффективных и надежных программ, в процессе которой обучаемый, решая поставленные ему индивидуальные задачи, действует вполне самостоятельно, но постоянно имеет возможность получения квалифицированной помощи, корректирующей и направляющей его усилия, начиная с этапа понимания условия задачи и кончая этапом оценки правильности решения.

СПИСОК ЛИТЕРАТУРЫ

1. Brusilovsky P. Adaptive hypermedia // User Modeling and User-Adapted Interaction. — 2001. — Vol 11. — P. 87–110.
2. Касьянов В.Н., Касьянова Е.В. Дистанционное обучение: методы и средства адаптивной гипермедиа // Программные средства и математические основы информатики. — Новосибирск, ИСИ СО РАН, 2004. — С. 80–141.
3. Касьянов В.Н, Касьянова Е.В. Адаптивные системы и методы дистанционного обучения // Информационные технологии в высшем образовании. — 2004.— Т.1, N 4. — С. 40–60.
4. Kasyanov V. SVM — Siberian Virtual Museum of Informatics History // Innovation and the Knowledge Economy: Issues, Applications, Case Studies. — Amsterdam, IOS Press, 2005. — Part 2. — P. 1014–1021.
5. Kasyanov V.N., Kasyanova E.V. Web-based systems for supporting computer-science teaching and learning // SIGCSE Bulletin. — New York: ACM Press, 2002. — Vol.34, N 3. — P. 238. — (Proc. of the 7th ACM SIGCSE Conf. on Innovation and Technology in Computer Science Education).
6. Kasyanov V.N., Kasyanova E.V. An environment for Web-based education of programming // HCI Internat. 2003. — Heraklion, Crete University Press, 2003. — P. 179–180.
7. Kasyanova E.V. WAPE: an adaptive environment for Web-based education of programming // Proc. of the 17th IMACS World Congress. — Paris, 2005. — P. 681–685.
8. Cooper G. The computational complexity of probabilistic inference using Bayesian belief networks // Artificial Intelligence. — 1990. — Vol. 42. — P. 393–405.

9. Касьянова Е.В. Язык программирования Zonnon для платформы .NET // Программные средства и математические основы информатики. — Новосибирск, ИСИ СО РАН, 2004. — С. 189–205.
10. Касьянова Е.В. Вводный курс программирования на базе языка Zonnon // Методы и инструменты конструирования и оптимизации программ. – Новосибирск, ИСИ СО РАН, 2005. — С. 95–116.
11. http://www.microsoft.com/Rus/Msdnaa/Curricula/Default.msp