Лукин Владимир Николаевич, *кандидат физико-математических наук, доцент*

Чернышов Лев Николаевич, *кандидат физико-математических наук, доцент*

Московский авиационный институт (национальный исследовательский университет), г. Москва

ПРОБЛЕМЫ ПОДГОТОВКИ СТУДЕНТОВ В ОБЛАСТИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ: КОНТРОЛЬ КАЧЕСТВА

Обсуждаются проблемы качества подготовки специалистов по информационным технологиям. Для качественной подготовки необходимо большое количество заданий на программирование, что увеличивает нагрузку на преподавателя. Предлагается подход, основанный на генерации контрольных заданий по дисциплине «Базы данных». Задания формируются на основе набора базовых заданий с использованием техники параметризации. Результат выполнения студенческого запроса к базе данных оценивается. В оценке учитывается количество попыток до достижения правильного результата. Подход может использоваться для различных дисциплин.

Ключевые слова: *контроль знаний, информационные технологии, генерация контрольных заданий, базы данных.*

Несмотря на постоянный устойчивый выпуск бакалавров и магистров, в дипломах которых записано, что они специалисты по информационным технологиям (формулировка может варьироваться), наблюдается острая нехватка именно специалистов, причём разных направлений, от простых программистов до руководителей проектов. Даже в Москве к авторам неоднократно обращались с просьбой порекомендовать толковых выпускников или, на худой конец, студентов старших курсов. Так как качество выпускника, в конечном счёте, определяется потребностями рынка труда, возникает вопрос, кого мы выпускаем? Он естественно перерастает в проблему качества преподавания, которое определяется тем,

чему учат, как учат, кто учит и кого учат. Сразу скажем, что мы не будем привязываться к официальным критериям качества, которые сводятся, в основном, к многочисленным отчётам о рейтинге вуза, организации учебного процесса, о соответствии программ формам, определённым очередным ФГОС и полностью соответствующим требованиям сверху. Нас интересует КПД нашей, преподавательской работы.

Определимся с тем, кого мы учим. Низкий уровень школьной подготовки — это уже общее место. И знания по информатике не исключение. Хотя на первом курсе и встречаются студенты с неплохой подготовкой, и их присутствие весьма полезно, они не определяют, к сожалению, общий уровень. Следовательно, надо вести занятия так, как будто в школе ничему не учили (знаменитая в своё время фраза Аркадия Райкина «забудьте школу как кошмарный сон» здесь даже и не нужна).

Теперь о тех, кто учит информационным технологиям. Если считать, что качество обучения определяется потребностями производства, необходимо, чтобы преподаватель информатики (программирования) был с ними знаком. По естественным причинам в массовом порядке подобрать таких преподавателей невозможно. Для решения этой проблемы А.Н. Терехов предлагает связать с кафедрой предприятие по разработке программного обеспечения, на котором, в частности, будут работать преподаватели [1]. Но это не так уж просто: во-первых, поддержкой подобного предприятия или постоянными контактами с ним должен заниматься специальный сотрудник, которого обычно нет, во-вторых, на предприятии не обязательно будут практикующие программисты, желающие заниматься со студентами. В этих условиях нужно, чтобы требования преподавателя были максимально близки к производственным. Если ограничиться программированием, следует потребовать соблюдения характеристик качества программного обеспечения из стандарта ГОСТ Р ИСО/МЭК 25010-2015 [2], наиболее важные из которых для программирования — это функциональная пригодность, надёжность, удобство использования, сопровождаемость.

На вопрос «чему учить» самый простой ответ — тому, что указано в учебной программе. Да, конечно, но в программе нет понятия «качественное программирование». Пробежать в течение семестра тему «программирование» можно без труда, но тогда результат будет такой, какой он есть. Если мы берёмся за обучение качеству, рассмотрим подробнее его характеристики.

Функциональная пригодность — наиболее очевидная характеристика. В студенческом варианте она означает, что студент в точности реализует те функции, которые требует преподаватель, играющий роль заказчика программного обеспечения.

Надёжность в «большом мире» определяется вероятностью появле-

ния ошибки, наносящей существенный ущерб. Обычно она оценивается общим количеством ошибок при тестировании. Отсюда следует, что преподаватель должен учитывать количество ошибок, которые делает студент при сдаче работы, а особенно — ошибок при обработке нештатных ситуаций. Именно они приводят к падению космических аппаратов, сбоям в банковских системах и т.п.

Удобство использования в студенческих работах сводится к пользовательскому интерфейсу. К сожалению, этой, в наибольшей степени влияющей на пользователя, характеристике качества уделяется в учебном процессе слишком мало внимания. А огромное количество потерянного времени, значительное число ошибок пользователя получается из-за безобразного интерфейса, сработанного нашими, по сути, выпускниками. Эта проблема очень старая (см., например, [3, 4]), но воз и ныне там. Посмотрите хотя бы на стиль взаимодействия с системой, которая работает в поликлиниках и которую продвигают под маркой «цифровизации» медицины.

Сопровождаемость программной системы — важнейшая её характеристика, но реально обучать сопровождению в учебном процессе невозможно. Однако можно требовать соблюдения стиля, удобного для понимания программного текста. И конечно, можно и нужно обучать студентов навыкам анализа программ.

Проблема обучения созданию качественного программного обеспечения достаточно сложна, она охватывает все разделы технологии разработки программного обеспечения. Мы остановимся лишь на двух темах, наиболее близких к студенческим работам: программировании и работе с базами данных.

И тут мы переходим к вопросу «как учат». Мы видим, что для обучения качественной разработке требуется тщательный контроль студенческой продукции. Значит, для проверки только одной программы необходимо подготовить пакет «правильных» тестов, проверяющих функциональность, и пакет «неправильных», проверяющих надёжность. Проверка стиля выполняется визуально, для этого тесты не нужны. А учитывая тезис Р. Гласса [5] «Большинство задач допускают множество одинаково хороших программных решений», признаем, что автоматическая проверка текста — весьма непростая задача.

Чтобы быть уверенным в качественной подготовке будущего специалиста, следует для каждой темы иметь несколько задач, в том числе и задач на анализ текста. Отдельно стоят контрольные мероприятия: здесь нужно подготовить пакет однотипных задач. При современных технологиях списывания, задачи для каждого экзамена (зачёта) должны быть уникальными. Мы приходим к выводу, что объём работы для подготовки преподавателем добротного курса становится чрезмерным, и приходится, к сожалению, жертвовать качеством. Итак, мы добрались

до одной из ключевых причин выпуска некачественных программистов: естественная перегрузка преподавателя на подготовку курса, осложнённая необходимостью выполнять всё больше бессмысленных, по сути, работ по отчётам и контролям.

Авторы совершенно независимо друг от друга пришли к одной и той же идее: автоматизировать, по возможности, как процесс генерации заданий и тестов, так и процесс проверки студенческих работ. Результатом стали совместно разработанные системы поддержки деятельности преподавателя по дисциплинам программирования и баз данных (см., например, [6]). Системы изначально не предполагались для широкого распространения, авторы их разрабатывали для себя. Но близость получившихся решений дала повод задуматься о пользе подобных разработок и для других преподавателей, желающих выпускать качественных программистов.

В процессе обучения программированию используются все возможные формы обучения: курсовые работы, практические и лабораторные занятия, домашние задания, тестирование и т.п. Объём работы для студентов может варьироваться и даже зависеть от индивидуальных предпочтений преподавателя, который зачастую не строго соблюдает рабочую программу. Но для хорошей подготовки очевидно одно: чем больше программ напишет студент за время прохождения какой-либо дисциплины, тем лучше. И это повышает актуальность автоматизации подготовки и проверки контрольных заданий. При этом задания не должны повторяться, иначе мы получаем проблему ГДЗ (готовых домашних заданий). Генерация тестовых заданий применяется в различных дисциплинах и реализована в нескольких системах [7], но для программирования имеется своя специфика.

Автоматизированная проверка ответов на контрольные задания аналогична проверке открытых ответов в системах автоматизированного тестирования, но для программистских дисциплин задача несколько упрощается. Можно было бы воспользоваться опытом проверки правильности работы программ при проведении олимпиад, тем более что для этого существует несколько систем. Но там задачи уникальные, и тестовые данные составляются каждый раз заново. Для контроля в типовом учебном процессе необходим набор типовых задач, а их уникальность обеспечивается генерацией вариантов на некотором достаточно большом наборе «базовых» задач. Эти базовые задачи могут а) составляться самими студентами и б) накапливаться от семестра к семестру.

Рассмотрим дисциплину программирования, где необходимо проверять умение читать и писать программы на некотором языке программирования. В базе тестовых заданий (БТЗ) хранится набор готовых программ вместе с исходными данными и результатами работы программ

на них. Возможны следующие типы тестовых заданий с открытыми ответами:

1. Студенту предъявляется программа и данные. В ответе он должен указать результат работы программы. Система проверяет правильность результата по хранящемуся эталону.
2. Студенту предлагается написать программу по заданному условию. Система выполняет его программу на хранящихся данных и сверяет полученный результат с эталонным.
3. Студенту предъявляется программа и результат её работы. В ответе он должен указать данные, на которых программа получила указанный результат. Система сопоставляет данные с хранящимися эталонами или выполняет программу и сравнивает полученный результат с указанным в задании.

В первом и третьем случаях проверяется умение читать программу, во втором — писать.

Предметной областью, для которой составляются алгоритмы и формулируются условия программ, являются, как правило, абстрактные объекты: массивы, матрицы, строки и т.п. Именно на таких объектах изучаются типовые задачи на алгоритмизацию.

Простой способ генерации программ, предъявляемых студенту, — замена имён переменных и объектов программы. Немного сложнее варьировать алгоритм путём замены конструкций, например замена цикла типа «for» на тип «while». Большие возможности появляются, если программу хранить в виде синтаксического дерева, на котором можно проводить эквивалентные преобразования, а затем генерировать программу.

Принципы организации проверки контрольных заданий по дисциплине программирования подходят и для других дисциплин, к примеру, дисциплинам по базам данных, в которых изучается язык SQL. Роль программы играет SQL-запрос, данных — набор таблиц (база данных), а результат выполнения программы — результирующая таблица.

Рассмотрим разработанные нами две близкие системы для подготовки контрольных заданий и проведения тестирования для дисциплины «Базы данных» (БД) по теме «Язык запросов SQL». Первая система реализована как web-приложение, что позволяет формировать БТЗ одновременно нескольким преподавателям.

Обе системы используют следующую технологию для создания базы тестовых заданий.

Подготовка тестового материала:

- определить предметную область для формирования базы данных;
- определить информационные потребности и ограничения;
- спроектировать БД в 3-й нормальной форме и сгенерировать её в заданной СУБД;
- исходя из сложности курса, определить множество запросов и реализовать их на SQL;

● протестировать реализованные запросы на тестовых вариантах.

На данный момент созданы десятки предметных областей — основ для «базовых» заданий. Описание БД для каждой предметной области создаётся в виде SQL-скрипта, который может динамически выполнятся как на этапе подготовки БТЗ, так и на этапе тестирования. Для каждой БД создаётся несколько заданий в виде формулировки запроса к БД на естественном языке и на языке SQL.

Для пополнения БТЗ и проверки корректности описания БД и запросов создано отдельное web-приложение, которое используется и преподавателями, и студентами. В приложении автоматически формируется экран с кнопками «создать БД» и кнопками для выполнения запросов «1 run», «2 run» и т.д. (*рис. 1*). По радиокнопкам «SQL» и «текст» в правой части формы отображаются скрипты создания БД и описания заданий соответственно, которые могут редактироваться.

При выполнении запроса на экран выдаётся его формулировка, соответствующий SQL-запрос, результат запроса в виде таблицы, а также содержимое участвующих в запросе таблиц (*рис. 2*).

По созданной таким образом БТЗ в дальнейшем проводятся контрольные мероприятия.

Рассматриваются два основных типа заданий.

1. Студенту, как и описывалось ранее, предлагается составить SQL-запрос по заданной формулировке на естественном языке и по заданной БД. Для проверки ответа система динамически генерирует БД, выполняет SQL-запрос, введённый студентом, выполняет SQL-запрос, имеющийся в БТЗ, и сравнивает результаты. Задание упрощается, если студенту предложить ввести только часть запроса.
2. Студенту предлагается заполнить таблицу результата запроса по заданному SQL-запросу. На экране отображаются значения исходных таблиц и SQL-запрос. Для проверки ответа система динамически генерирует БД,

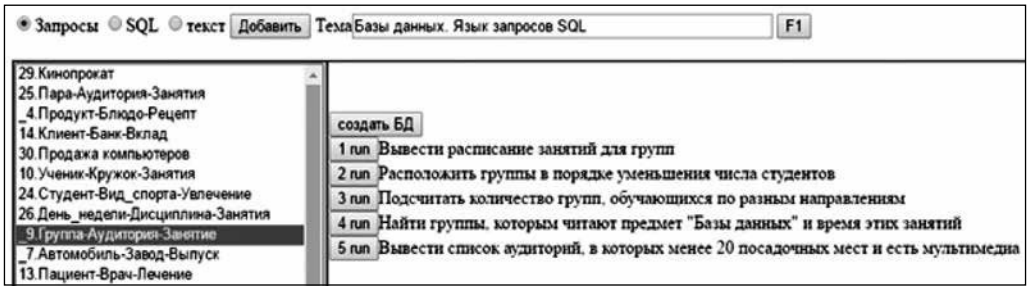

Рис. 1. *Экран контроля правильности тестовых заданий*

БД db_09.
Вывести расписание занятий для групп
SELECT Name,Datas,Subject FROM Zanyatie LEFT JOIN Groups ON Zanyatie.Group_ID = Groups.ID

РЕЗУЛЬТАТ:

Name	Datas	Subject
БИ	2014-02-16 09:00:00	Математика
БИ	2014-02-16 10:30:00	Анализ данных
БИ	2014-02-16 09:00:00	Программирование
БИ	2014-02-16 10:30:00	Базы данных
ПМ	2014-02-17 00:00:00	Моделирование БП

auditoriya

ID	Nomer	Vmestimost	Multimedia
1	101	20	1
2	102	30	0

groups

ID	Name	Course	Students	Starosta
1	БИ	1	24	Иванов
2	БИ	2	18	Синяков

zanyatie

Group_ID	Auditoriya_ID	Datas	Subject
1	2	2014-02-16 09:00:00	Математика
1	2	2014-02-16 10:30:00	Анализ данных

Рис. 2. *Результат выполнения запроса*

выполняет SQL-запрос и сравнивает с тем, что ввёл студент.

В первом случае проверяется умение решить задачу, во втором — умение проанализировать решение. Это различные умения, каждое из которых требуется в практической работе программиста.

Проведение контрольного мероприятия:

- студент регистрируется в системе;
- самостоятельно или по указанию преподавателя выбирает предметную область;
- автоматически или от преподавателя получает вариант задания;
- решает задачу на SQL, вносит полученный запрос в систему и получает результат;
- получает из системы правильный ответ;
- совместно с преподавателем визуально сравнивает результат и ответ;
- если выборки семантически эквивалентны, задача считается зачтённой, работа заканчивается, в противном случае увеличивается счётчик подходов, и задача решается повторно;
- если количество подходов превышает критический уровень, задача считается незачтённой, работа заканчивается.

Здесь требуется небольшой комментарий, связанный с проверкой результата. В описании проведения контрольного мероприятия говорится о визуальном сравнении результатов, а в описании типов заданий — об автоматическом. Автоматическое сравнение результатов — не такая простая задача, хотя и разрешимая. Поэтому для сложных запросов на практике удобнее посмотреть на результат и оценить его, а для простых лучше сравнивать автоматически. Простейший признак ошибки — различный объём выборки в результате и ответе.

Разнообразие вариантов заданий обеспечивается переходом от фиксированных формулировок к параметризированным шаблонам, по которым генератор формирует уникальные задания. Простой способ генерации вариантов — случайные размеры таблиц в базовом задании. Здесь необходимо только следить за соответствием

связанных таблиц, а также за сохранением некоторого числа записей, от которых зависят запросы. Это соблюдение целостности можно упростить, если после выбора случайных значений проверять корректность запросов. При этом некоторые запросы могут быть отброшены (забракованы).

Другим способом генерации является изменение содержимого таблиц. Для определенных полей записи создаются таблицы возможных значений, например, список фамилий FIO. Шаблон добавления информации представляется в параметризованном виде, например:

```
INSERT INTO Groups values ([1-3],
'ПИ-' + id, 3, {20-30}, dm = {FIO});
```

Здесь значение «[1–3]» показывает, что сгенерируется три команды для вставки записей со значениями первого атрибута от 1 до 3. Конструкция «{20–30}» — генерация случайных значений в диапазоне от 20 до 30. Если значение атрибута символьное, указывается имя домена — заранее созданного справочника значений, например фамилий.

Генерация содержимого БД (тестовых заданий) может производиться как однократно при создании БТЗ, так и непосредственно при проведении контрольного мероприятия. Первый вариант используется при групповом контроле, второй — при индивидуальном или при самоконтроле. В любом случае при генерации может возникнуть нарушение целостности БД, что приводит к различным аномалиям (например, результат запроса будет пустым). Такой случай исключается созданием заведомо избыточного количества записей и отбрасыванием неподходящих вариантов в дополнительном проходе.

На следующих рисунках представлена вторая система, реализованная как обычное приложение на базе Foxpro. Сценарий использован тот же.

Например, преподаватель определяет предметную область «Экзаменационная ведомость», описание которой приводится в первом поле. Затем задаётся структура БД, которая используется для генерации. По кнопке «Добавить» создаётся БД и добавляется в репозиторий (*рис. 3*). Затем для выбранной предметной области преподаватель формулирует запросы и реализует их на SQL (*рис. 4*).

Загрузка тестовых данных производится либо исходя из характера запросов, либо массово, для большего количества вариантов.

Рис. 5 и 6 демонстрируют процесс решения задачи студентом. В данном случае оба решения ошибочны. Верный ответ выдаётся по нажатию кнопки «Ответ».

Приведённая технология контроля знаний по дисциплине «База данных» используется авторами достаточно долго. До этого использовалось как традиционное решение задач «на бумажке», так и обычное тестирование. Оба эти подхода проигрывают: при решении неизбежна небреж-

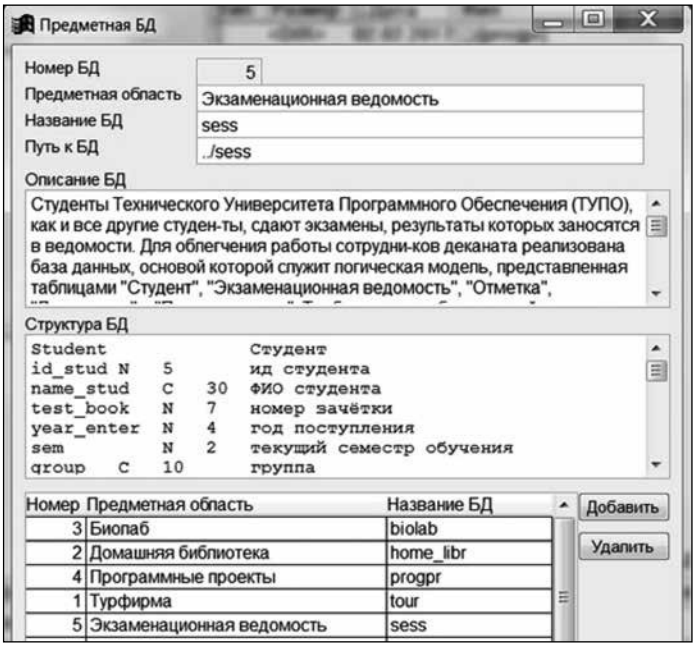

Рис. 3. *Преподаватель определяет предметную область и создаёт БД*

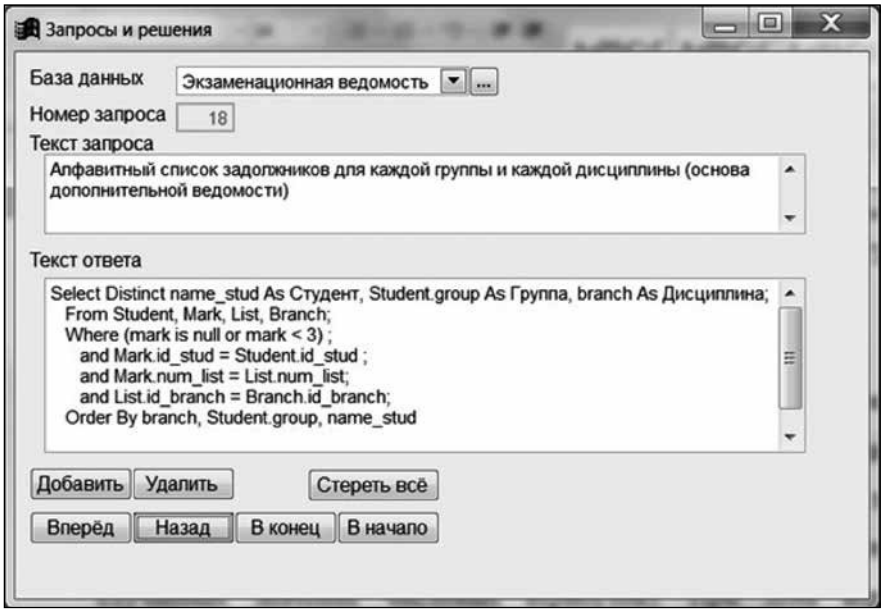

Рис. 4. *Преподаватель готовит задания*

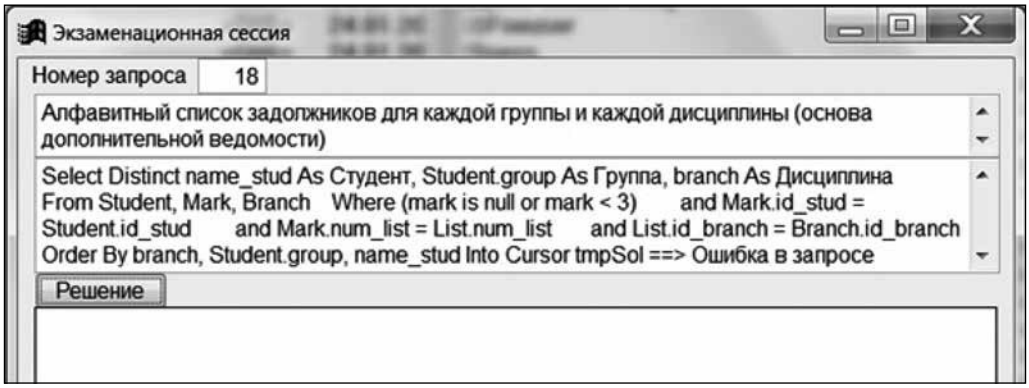

Рис. 5. *Студент решает задачу. Ошибка: в списке таблиц не указан List*

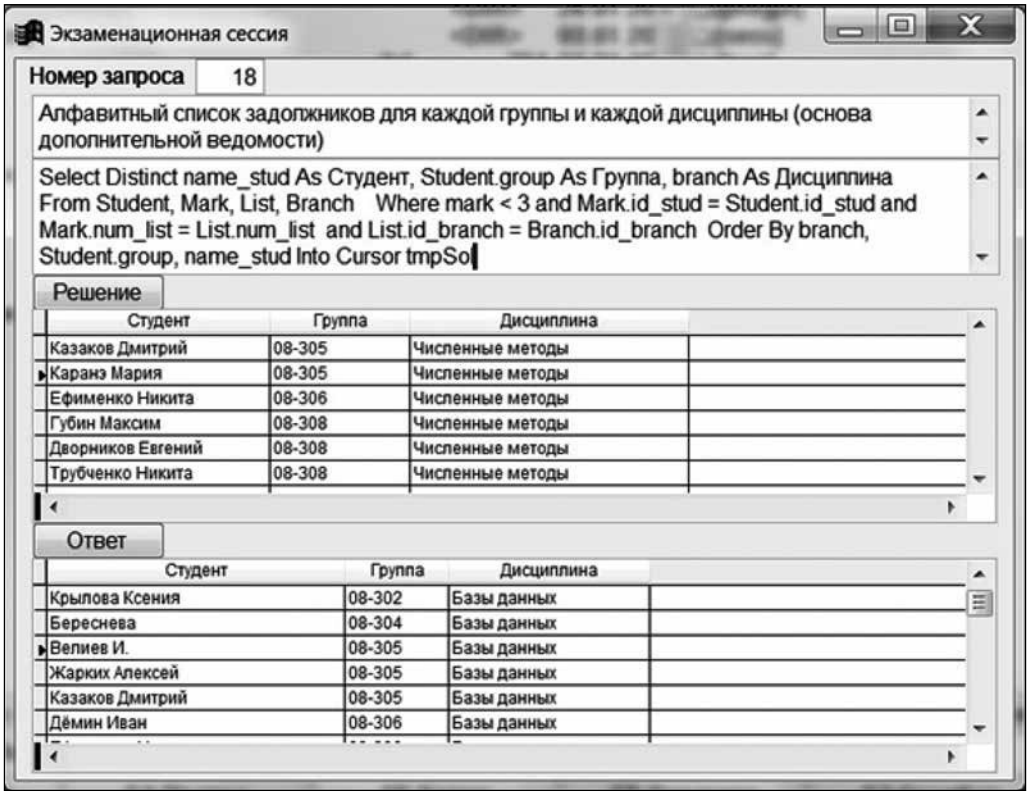

Рис. 6. *Студент решает задачу. Содержательная ошибка: не учтены не явившиеся студенты: в условии вместо (mark is null or mark<3) задано mark<3*

ность, которая, войдя в привычку, затруднит профессиональную работу будущего программиста, а тестирование проверяет только наличие сведений, а не умения. И очень важно то, что предлагаемая технология существенно снижает трудоёмкость подготовки материала к контрольным мероприятиям, позволяет проводить контроль в условиях, похожих на производственные, и не вызывает споров и выпрашивания оценки: «Ну я же почти правильно всё сделал!..» А здесь система объективно принимает решение, и вопросов нет.

Заметим, что подобный подход годится и для таких дисциплин, как программирование. Здесь роль SQL-запросов играет программа, условие — это формулировка задачи, а тестовые варианты готовятся даже проще. В режиме самоподготовки студенту может дополнительно предъявляться корректное решение (*рис. 7*), но нужно, чтобы он понимал, что оно обычно не единственное.

Подобный подход применялся одним из авторов и для дисциплины «Специальные разделы программирования» по теме «Конечные автоматы и регулярные грамматики» [8].

Возвращаясь к проблеме подготовки качественных программистов, можно сказать, что, конечно, так просто её не решить, необходим комплексный подход, который затрагивает практически все дисциплины из этого направления. Но даже такие относительно простые решения позволяют заметно улучшить понимание предмета, уровень владения им, а порой и вызывают к нему дополнительный интерес, о чём свидетельствуют работы [9, 10], выполненные совместно со студентами. И, конечно, экономия времени преподавателя на рутинные действия даёт возможность больше работать со студентами.

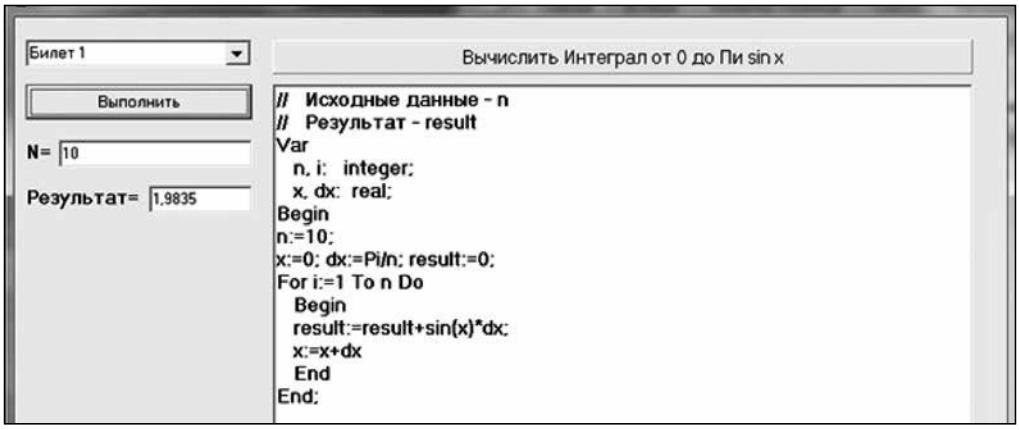

Рис. 7. *Самоподготовка к экзамену по программированию на 1-м курсе*

ЛИТЕРАТУРА

1. *Терехов А.Н.* Технология программирования. — М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2006.
2. Стандарты: ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models ГОСТ Р ИСО/МЭК 25010-2015 Информационные технологии. Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программных продуктов.
3. *Купер А.* Психбольница в руках пациентов. — СПб: Символ-Плюс, 2004.
4. *Платт Д.* Софт — отстой! И что с этим делать? — СПб: Символ-Плюс, 2007.
5. *Гласс Р.* Факты и заблуждения профессионального программирования. — СПб: Символ-Плюс, 2008.
6. *Лукин В.Н., Чернышов Л.Н.* Технология контроля знаний студентов по дисциплинам программирования. Материалы XX Международной конф. по вычисл. механике и совр. прикладным программным системам, Алушта. — М.: Изд-во МАИ, 2017. — С. 785–787.
7. *Братчиков И.Л.* Генерация тестовых заданий в экспертно-обучающих системах / И.Л. Братчиков // Вестник РУДН. Серия «Информатизация образования» — 2012. — № 2.
8. *Чернышов Л.Н.* Программа-тренажер по теории формальных языков и конечных автоматов. / Тезисы докладов Международной конференции по вычислительной механике и современным прикладным программным системам. — М., 2013(ВМСППС), Алушта, Крым, 2013, — М.: Вузовская книга, 2013. — С. 862–863.
9. *Рыжов А.А., Чернышов Л.Н., Булыгин А.С.* Система автоматической проверки ответов. / Двенадцатая конференция «Свободное программное обеспечение в высшей школе»: Материалы конференции / Переяславль, 27–28 января 2017 года. — М.: Basealt, 2017. — С. 22–24.
10. *Махлай В.С., Чернышов Л.Н.* Web-приложение для проведения контрольных и практических работ по программированию с автоматической генерацией заданий. / Десятая конференция «Свободное программное обеспечение в высшей школе»: Тезисы докладов / Переяславль, 24–25 января 2015 года. — М.: Альт Линукс, 2015. — С. 86–88.