МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК
Кафедра программного обеспечения

ДОПУЩЕНО К ЗАЩИТЕ В ГЭК
И ПРОВЕРЕНО НА ОБЪЕМ
ЗАИМСТВОВАНИЯ
Заведующий кафедрой
д.п.н., профессор
_____ *И.Г. Захарова*
«20» июня 2016 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

РАЗРАБОТКА АЛГОРИТМОВ ПОДДЕРЖКИ МЕТОДОВ АКТИВНОГО ОБУЧЕНИЯ НА БАЗЕ МНОГОПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРАКТИВНОГО ВЗАИМОДЕЙСТВИЯ

02.04.03. Математическое обеспечение и администрирование информационных систем
Магистерская программа «Высокопроизводительные вычислительные системы»

Выполнила работу
Студентка 2 курса
очной формы обучения
Павлова
Елена
Александровна

Научный руководитель
к.т.н., доцент
Воробьева
Марина
Сергеевна

Рецензент
к.т.н., доцент
кафедры информационных систем
Григорьева
Инна
Ивановна

Тюмень 2016

Содержание

ВВЕДЕНИЕ

Сегодня многие университеты и институты выпускают специалистов в области ИТ, однако, традиционные формы обучения в вузе (изложение материала и контроль усвоения) не позволяют в полной мере развить компетенции, необходимые для дальнейшего профессионального роста. Известно, что студенты часто испытывают затруднения при выполнении заданий, требующих самостоятельного поиска знаний, творческого подхода к их выполнению, выделения главного из большого потока информации, представления результатов работы письменно и устно.

Внедрение интерактивных форм обучения – одно из важнейших направлений совершенствования подготовки студентов в современном вузе. Данные исследований подтверждают, что использование активных и интерактивных подходов является наиболее эффективным путём, способствующим обучению студентов и формированию необходимых компетенций.

Цель магистерской диссертации – разработка алгоритмов активного обучения на базе многопользовательского интерактивного взаимодействия.

Для достижения цели необходимо решить следующие задачи:

- Изучить предметную область.
- Разработать структуру автоматизированной системы для обучения программированию.
- Выявить особенности разработки модуля автоматической генерации заданий на основе шаблонов.
- Разработать алгоритм генерации набора заданий для выполнения лабораторных и контрольных работ.
- Выявить структуру тестового задания с учётом сложности и уровня интерактивности.
- Выделить этапы формирования учебного проверочного задания по программированию.

- Разработать алгоритм автоматической генерации тестовых заданий, содержащих программный код, с формированием вариантов ответа во время сеанса тестирования.
- Разработать автоматизированную систему, включающую алгоритмы создания, генерации и распределения заданий между студентами.

ГЛАВА 1. АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ В СФЕРЕ ОБРАЗОВАНИЯ

1.1. Основные формы взаимодействия преподавателей и студентов

В своих исследованиях Т.Г. Мухина [1], В.Н. Кругликов [14] отмечают, что в образовании сложились, утвердились и получили широкое распространение три формы взаимодействия преподавателя и студентов:

1. Пассивные методы
2. Активные методы
3. Интерактивные методы

Каждый из методов имеет свои особенности.

Пассивные методы предполагают авторитарный стиль взаимодействия, активные – демократический. Интерактивные методы можно рассматривать как наиболее современную форму активных методов.

Интерактивные методы (рис. 1) в отличие от активных ориентированы на более широкое взаимодействие студентов не только с преподавателем, но и друг с другом, и на доминирование активности студентов в процессе обучения. Место преподавателя на интерактивных занятиях сводится к направлению деятельности студентов на достижение целей занятия. Преподаватель также разрабатывает план занятия, часто это интерактивные упражнения и задания, в ходе выполнения которых студент изучает материал.

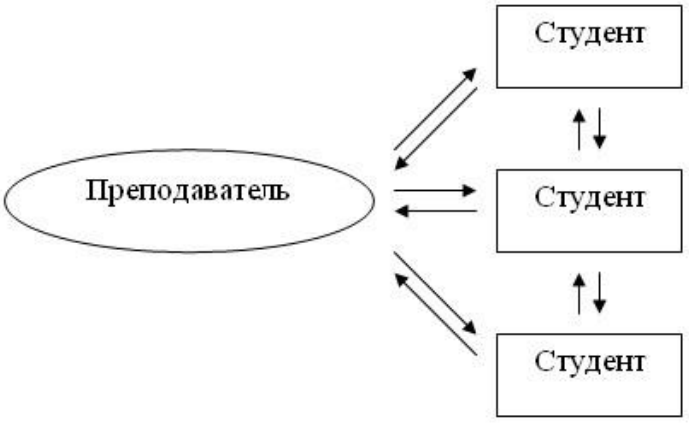

Рисунок 1. Схема взаимодействия преподавателя и студентов при интерактивном методе

Интерактивное обучение – это специальная форма организации познавательной деятельности. Цель состоит в создании комфортных условий обучения, при которых студент или слушатель чувствует свою успешность, свою интеллектуальную состоятельность, что делает продуктивным сам процесс обучения. Интерактивное обучение – это диалоговое обучение, в ходе которого осуществляется взаимодействие между студентом и преподавателем, между самими студентами.

Задачами интерактивных форм обучения являются:

- пробуждение у обучающихся интереса;
- эффективное усвоение учебного материала;
- самостоятельный поиск учащимися путей и вариантов решения поставленной учебной задачи (выбор одного из предложенных вариантов или нахождение собственного варианта и обоснование решения);
- установление воздействия между студентами, обучение работать в команде, проявлять терпимость к любой точке зрения, уважать право каждого на свободу слова, уважать его достоинства;
- формирование у обучающихся мнения и отношения; формирование жизненных и профессиональных навыков; выход на уровень осознанной компетентности студента.

При использовании интерактивных форм роль преподавателя резко меняется, перестаёт быть центральной. Преподаватель регулирует процесс и занимается общей организацией, готовит заранее необходимые задания и формулирует вопросы или темы для обсуждения в группах, даёт консультации, контролирует время и порядок выполнения намеченного плана. Участники обращаются к социальному опыту – собственному и других людей, при этом им приходится вступать в коммуникацию друг с другом, совместно

решать поставленные задачи, преодолевать конфликты, находить общие точки соприкосновения, идти на компромиссы.

1.2. Электронные образовательные системы

Существует большое количество современных систем проверки знаний. Системы проверки знаний реализованы в двух основных вариантах: как Интернет-проекты и как решения для локального использования, например, в локальных сетях.

Большинство учебных заведений предпочитают разработку собственных автоматизированных систем проверки знаний в силу разных причин: для одних важно иметь локальный вариант системы, другим необходимо иметь возможность генерировать варианты ответа во время сеанса тестирования, третьим важно развивать и пополнять функциональность системы.

Гриценко А.А., Анискин С.А., Мальчева Р.В. в работе [5] выделяют ряд характеристик автоматизированной системы проверки знаний системы, которые определяют, насколько эта система удовлетворяет современным требованиям и насколько затраты на разработку этой системы приносят результаты:

- Используемость – многие системы практически не используются в работе высших учебных заведений, для которых они проектировались и разрабатывались;
- Расширяемость – возможность поддержки работоспособности системы, ее адаптации и изменения функциональности студентами, которые не являются авторами системы. Это свойство является прямым следствием используемости;
- Распространенность – для современных систем, широта распространения является одной из базовых характеристик. Такая распространенность предполагает возможность использования (адаптации) системы независимо от разработчиков системы.

Автоматизированная обучающая система должна решать две основные задачи: обучение и проверку знаний. Подсистема проверки знаний, в свою очередь, бывает двух видов – статическая и динамическая. В статической подсистеме проверки знаний уже содержатся готовые вопросы и ответы, в динамической – вопросы и ответы генерируются по мере необходимости.

Генерация вариантов заданий может осуществляться разными методами: на основе шаблонов, на основе деревьев И/ИЛИ, на основе баз знаний. Наиболее полный обзор методов генерации приводится в монографиях [2, 16].

Авторы в работе [20] различают два принципа генерации вариантов: метод «подумать» и метод «потрясти».

Метод «подумать» заключается в том, что перед тем, как писать генератор, проектируется общий вид качественного (удовлетворяющего требованиям) варианта; далее составляется генератор таким образом, чтобы получались только такие варианты.

Метод «потрясти» представляет собой простой генератор вариантов, не учитывающий качество генерируемых вариантов, и программу-анализатор, составляющую оценку варианта с точки зрения предъявляемых требований. Далее для того, чтобы сгенерировать хороший вариант, простой генератор запускается несколько раз до тех пор, пока анализатор не признает сгенерированный вариант качественным.

Задача генерации индивидуальных вариантов заданий является важной с точки зрения повышения качества образования и усвоения учебного материала. Обучающая система, в которой предусмотрена генерация вариантов заданий, обладает следующими свойствами: с одной стороны, такая система максимально приближена к традиционному индивидуальному обучению с преподавателем (система поддерживает обратную связь с обучаемым и имеет способность адаптироваться к его уровню и потребностям); с другой стороны, такая система включает те возможности, которые предоставляет вычислительная техника.

Одним из самых распространенных методов генерации вариантов является метод шаблонов. Существуют различные преобразования этого метода в зависимости от особенности системы и требований. Одним из примеров видоизмененного метода шаблона является метод, основанный на клонировании шаблона. В этом методе шаблон документа представляет собой параметризованное задание или матрицу задания, а клоны – это варианты, где конкретизированы некоторые переменные. Условия перебора параметров также задаются в документе матрице отдельной формулой, что позволяет создать различные типы заданий, то есть преподаватель не ограничен в выборе типа задач и при желании может параметризировать любой пример.

1.3. Существующие аналоги

Среди современных систем проверки знаний можно выделить следующие:

- Программный комплекс «АСОП-Эксперт»[1].
- Автоматизированная проверка знаний сотрудников «Форум Тест»[2].
- Генераторы заданий для ЕГЭ по предмету ИКТ Полякова К., Сидорова А.[3]
- Система «Задачи»[4].
- Сайт «Генератор заданий для школы онлайн»[5].
- Портал «Якласс»[6].
- Программа тестирования знаний «Айрен»[7].

Основные возможности систем

[1] http://www.asop-expert.ru/
[2] http://www.forum-media.ru/catalog/detail.php?ID=1014
[3] http://kpolyakov.spb.ru/school/ege.htm
[4] http://www.problems.ru/about_system.php
[5] http://www.abakbot.ru/online-16/191-generator-zadanij-onlajn
[6] http://www.yaklass.ru/
[7] http://irenproject.ru/

1) «АСОП-Эксперт» – автоматизированная система обучения и проверки знаний персонала энергосистем – предназначена для обеспечения и оптимизации процессов изучения и проверки знаний персонала энергетической отрасли.

Структура АСОП представляет собой три ветви – мастер (набор функций ведения информационных таблиц), экзамен (для обучения, предэкзаменационной подготовки и проведения тестирования методом альтернативных ответов) и обучение (для проведения обучения и предэкзаменационной подготовки). Для каждой должности создана отдельная база программы обучения. По каждой теме есть список документов и контрольное тестирование. Имеется раздел «Библиотека», содержащий полный перечень документов по данному направлению. Условие приобретения: базовый комплект – 95000 руб., дополнительный комплект обучаемого – от 32000 руб. до 300000 руб. (на 20 рабочих мест), абонентское обновление ранее приобретенных базы тестовых заданий и электронной библиотеки нормативных документов в течение одного года – 46000 руб.

2) Автоматизированная проверка знаний сотрудников «Форум Тест» – это платная система для тестирования сотрудников организации по охране труда, пожарной безопасности и электробезопасности.

Тесты заполняются вручную или делаются на заказ индивидуально. На основе готовых тестов формируются билеты и производится проверка знаний сотрудников, на основе которой создается отчет.

3) Генератор заданий типа B4 для ЕГЭ по предмету ИКТ Константина Полякова представляет собой узко направленный генератор с возможностью задать конкретные параметры и получить на выходе одну задачу или сгенерировать много вариантов однотипных заданий.

Генератор заданий типа C2 для ЕГЭ по предмету ИКТ Сидорова представляет собой консольное приложение, где на вход подается только

количество вариантов заданий, которое должно быть на выходе. После генерации создается текстовый файл с готовым вариантами заданий.

4) Система «Задачи» – это Интернет-проект для преподавателей (как помощь при подготовке уроков, кружков и факультативных занятий в школе) и для школьников (для более глубокого изучения темы).

Задачи в системе расположены по темам. Преподаватель может взять любое задание за образец и изменить потом в соответствии с требованиями, либо составить задание «с нуля».

5) На сайте «Генератор заданий для школы онлайн» представлен довольно простой генератор математических выражений. На входе задаются такие параметры, как разрядность, операнды, количество операций и наличие скобок, на выходе двадцать (не параметр) вариантов математических выражений с указанием правильного и альтернативного ответов.

6) Портал «ЯКласс» предлагает школьникам задачи практически по всем темам школьного курса алгебры, математики и некоторых других предметов. Задачи решаются прямо на компьютере, у каждой задачи есть один или несколько вопросов, на каждый вопрос нужно ввести ответ, после чего ответы автоматически проверяются, и ученик сразу узнает, сколько он получил баллов. Просматривать задачи можно без регистрации, решать только после регистрации, а платная подписка дополнительно позволяет узнавать правильный ответ, если задача решена неверно, и позволяет просмотреть пошаговое решение задачи. Из недостатков проекта можно отметить то, что с помощью редактора генераторов получаются только простые генераторы, в математических задачах в основном варьируются только числа. Это хорошо видно, если пролистать предлагаемые на сайте задачи. Задачи больше подходят для индивидуализации, чем для тренировки навыков решения, хотя позиционируются как последнее. Такое ограничение сложности задач – это следствие необходимости сделать редактор простым и доступным для широкого круга учителей.

Основные недостатки существующих систем: программный продукт либо не обладает функцией автоматической генерации вариантов заданий, либо генерирует задания узкой направленности, либо служит для реализации нужд определенной предметной области.

1.4. Постановка задачи

Цель магистерской диссертации – разработка алгоритмов активного обучения на базе многопользовательского интерактивного взаимодействия.

Для достижения цели необходимо решить следующие задачи:

- Изучить предметную область.
- Разработать структуру автоматизированной системы для обучения программированию.
- Выявить особенности разработки модуля автоматической генерации заданий на основе шаблонов.
- Разработать алгоритм генерации набора заданий для выполнения лабораторных и контрольных работ.
- Выявить структуру тестового задания с учётом сложности и уровня интерактивности.
- Выделить этапы формирования учебного проверочного задания по программированию.
- Разработать алгоритм автоматической генерации тестовых заданий, содержащих программный код, с формированием вариантов ответа во время сеанса тестирования.
- Разработать автоматизированную систему, включающую алгоритмы создания, генерации и распределения заданий между студентами.

В рамках магистерской диссертации необходимо спроектировать webприложение, которое будет представлять собой автоматизированную систему для создания и распределения генерируемых заданий по учебным дисциплинам.

Система должна удовлетворять следующим требованиям:

- Система должна представлять собой кроссплатформенное webприложение, не требуемое установки на персональные компьютеры и не зависящее от технических характеристик ПК;
- Для развертки, первоначальной настройки и поддержки системы требуется администратор, на которого будут возложены функции организации доступа к системе преподавателей и студентов, добавления учебных групп и дисциплин в систему;
- В системе должна быть предусмотрена отказоустойчивость и максимальная надежность приложения;
- В системе должны быть три роли входа: администратор, преподаватель и студент, защищенные паролем во избежание утечки информации и несанкционированного доступа.

Разрабатываемая система должна обладать следующими функциями и особенностями:

1) Автоматическая генерация вариантов заданий по шаблону преподавателя;
2) Назначение студентов на выполнение заданий;
3) Градация заданий по уровню сложности;
4) Статистическая обработка данных по результатам сдачи работ студентами;
5) Автоматическая компоновка заданий в работе индивидуально для каждого студента в зависимости от предыдущих результатов.

Для каждой роли входа (преподаватель, студент, администратор) должны быть предусмотрены индивидуальные функции и возможности.

Функции преподавателя:

- Добавление нужных тем по дисциплинам;
- Формирование новых заданий (шаблона, по которому в дальнейшем система генерирует индивидуальные варианты);

- Формирование работ, состоящих из заданий, скомпонованных по какому-либо признаку;
- Назначение студентов, которые должны будут выполнить работу;
- Проставление баллов за работы студентов в электронный журнал.

Функции студента:

- Получение в системе работ по дисциплинам;
- Просмотр успеваемости и результатов сдачи работ.

Функции администратора:

- Обеспечение доступа к системе для студентов и преподавателей – добавление новых пользователей, назначение ролей, выдача реквизитов доступа;
- Редактирование учебной информации: добавление групп, дисциплин.

ГЛАВА 2. СТРУКТУРА АВТОМАТИЗИРОВАННОЙ ОБУЧАЮЩЕЙ СИСТЕМЫ

2.1. Особенности образовательного процесса при изучении программирования

Многие университеты и институты выпускают специалистов в сфере ИТ, однако, традиционные формы обучения в вузе (изложение материала и контроль усвоения) не позволяют в полной мере развить компетенции, необходимые для дальнейшего профессионального роста. Известно, что студенты часто испытывают затруднения при выполнении заданий, требующих самостоятельного поиска знаний, творческого подхода к их выполнению, выделения главного из большого потока информации, представления результатов работы письменно и устно.

Учебные планы специальностей и направлений подготовки в ИТ-сфере предусматривают изучение программирования на разных языках (Паскаль, C#, C++, Python). Успешное освоение подобных дисциплин требует от студентов написания большого числа программ на одном из языков программирования, а от преподавателя – проверки их за короткое время.

Изучение особенностей образовательного процесса в Институте математики и компьютерных наук Тюменского государственного университета позволяет выделить следующие проблемы, которые появляются при изучении компьютерных дисциплин, в частности, основ программирования:

1. на первый курс поступают студенты с разным уровнем подготовки, мотивации, скоростью обучения и потребностями в освоении дисциплин, что приводит к трудностям в построении лекционного материала и в организации лабораторных и практических занятий: с одной стороны, преподаватель должен уделить много времени базовым понятиям, структурам языка, с другой стороны – студентам, имеющим навыки программирования,

нужно предоставить возможность творческой реализации и развития этих навыков;

2. большую часть времени, отведённого на занятие, преподаватель вынужден тратить на проверку лабораторных работ, на оценку усвоения знаний, зачастую не успевая охватить всех студентов;

3. проверка задач по программированию предусматривает тестирование решений на достаточно большом объёме различных входных данных;

4. для успешного усвоения дисциплины и закрепления материала многим студентам требуется решить большое количество задач (в том числе однотипных), а преподавателю – сформировать банк различных и однотипных заданий, тестов и т. п., а затем много времени уделить их проверке.

Решение этих проблем требует нового подхода и особой формы организации лекционных, практических и лабораторных занятий. Здесь на помощь могут прийти современные ИТ. Это особенно актуально при внедрении рейтинговой системы для оценки качества обучения студентов, когда подсчёт рейтинга студента нужно вести на каждом занятии и в конце контрольной недели формировать итоговые ведомости.

2.2. Модули автоматизированной обучающей системы

Комплексное решение проблем, связанных с обучением программированию, может быть реализовано в рамках автоматизированной обучающей системы (АОС). Но стандартные АОС, как правило, предусматривают только теоретическое изложение материала и проверку усвоения знаний. В отличие от них, можно предложить АОС, которая имеет структуру (рис. 2).

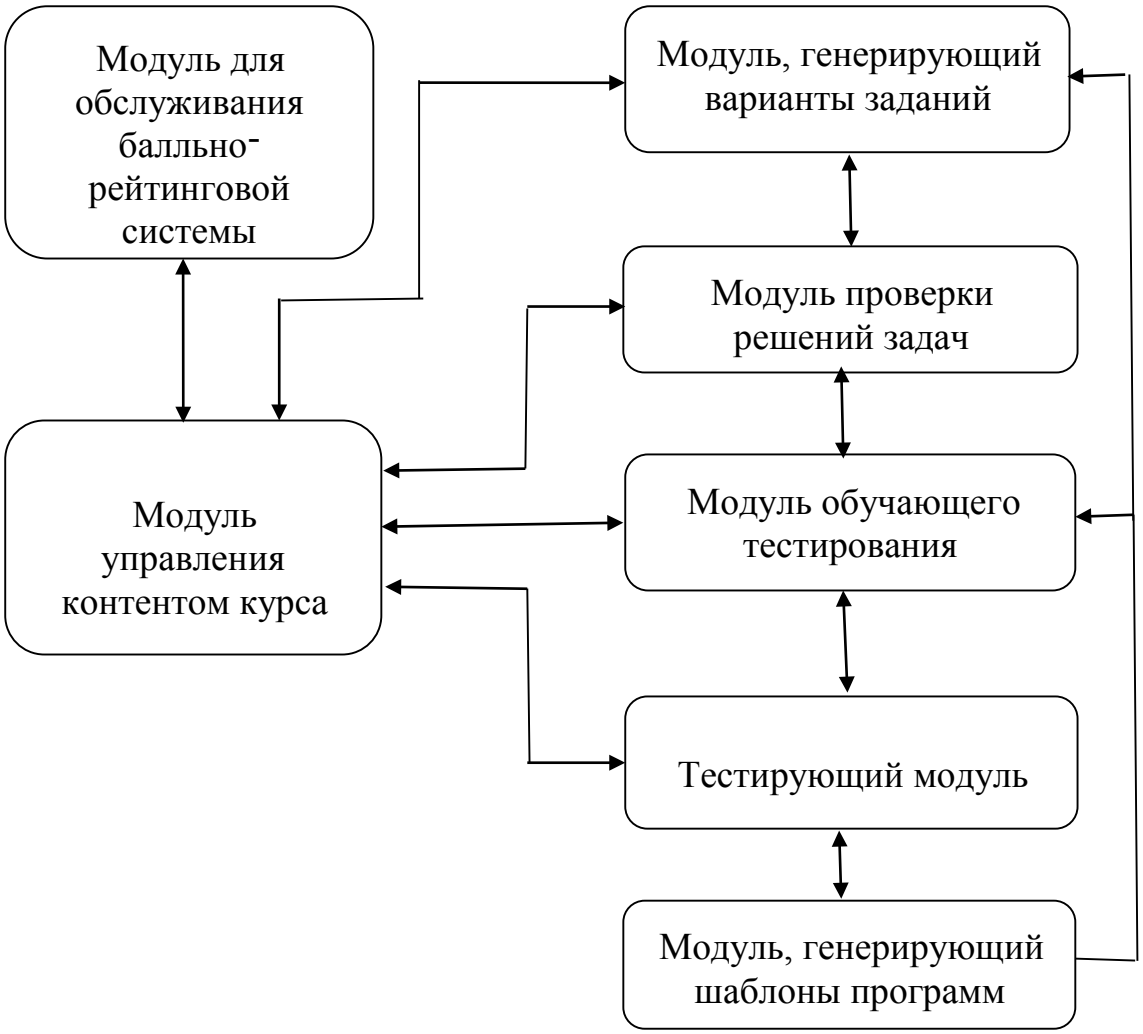

Рисунок 2. Схема взаимодействия модулей в АОС

Модуль управления контентом курса

В модуль управления контентом курса входит:

- опорный конспект лекций;
- список тем для самостоятельного изучения (по этим темам может быть подготовлен реферат или курсовая работа);
- задания для лабораторных работ и методические указания к их выполнению;
- упражнения, варианты контрольных и самостоятельных работ;

- список литературы (основной и вспомогательной); - список вопросов к экзамену или зачёту.

Модуль позволяет студенту увидеть структуру курса целиком и спланировать (возможно вместе с преподавателем) свою образовательную траекторию, у него появится возможность обучаться по индивидуальному графику, что будет способствовать развитию инициативности, активному восприятию учебного материала, учить грамотно формулировать вопрос, проблему, помогать развить навыки самостоятельной работы.

Тестирующий модуль

Функция тестирующего модуля – это текущий и промежуточный контроль (причем не только для студента, но и для преподавателя). Задания этого модуля:

- позволяют оценить степень готовности студента к сдаче экзамена;
- помогают преподавателю провести зачёт, затратив на проверку заданий минимум времени;
- могут являться одним из этапов экзамена;
- помогут преподавателю выявить темы, оставшиеся для обучаемых непонятными.

В качестве упражнения (или наказания за пропущенные занятия) студенту можно предложить самому составить тестовые задания с определёнными требованиями. Практика показывает, что такие упражнения способствуют усвоению и закреплению материала, а также помогают студенту проявить творческий подход, выразить себя.

Модуль обучающего тестирования

Обучающее тестирование помогает студенту контролировать свои знания в процессе обучения, позволяет выявить «белые пятна» и непонятные темы благодаря тому, что тестовые задания разбиты на темы и имеют 4 уровня сложности. Опыт применения подобного тестирования при изучении дисциплин «Основы программирования», «Технологии программирования»

показал, что такие задания помогают студенту ориентироваться в большом потоке информации по предмету, учат его выбирать главное, способствуют усвоению основных, базовых понятий (задания первого и второго уровня сложности) и применению их в нестандартных ситуациях (задания четвёртого уровня сложности).

Автоматизированные тесты привлекают обучаемых своей необычностью, по сравнению с традиционными формами контроля, возможностью проведения быстрого и объективного оценивания качества их знаний. Педагогические тесты при регулярном использовании побуждают к систематическим занятиям по предмету, что способствует формированию дополнительной мотивации к обучению. Оперативность обработки тестов обеспечивает эффективную обратную связь. А в условиях, когда обучаемые могут проходить испытания так часто, как им это потребуется, педагог может добиться, по меньшей мере, гарантированного усвоения базовых знаний.

Модуль, генерирующий типовые задания

Функция модуля – помощь преподавателю в оперативном формировании банка задач для самостоятельных и контрольных работ. Кроме того, этот модуль могут использовать студенты, которым требуется решение достаточно большого числа однотипных задач для закрепления материала.

Система автоматического формирования заданий может быть использована школьными учителями информатики для формирования большого количества заданий по разным темам

Модуль, генерирующий шаблоны программ

Модуль предоставляет преподавателю возможность конструировать вид заданий, когда в программу нужно вставить недостающие части так, чтобы она была работоспособной, или исправить ошибки в данном коде.

Модуль автоматизированной проверки решений

Автоматизированные системы проверки решений можно использовать для проверки домашних заданий, контрольных работ у студентов, изучающих дисциплины «Основы программирования», «Технологии программирования»

или другие дисциплины, которые подразумевают выполнение лабораторных работ на языках программирования, позволяя сократить время на их проверку. Эти системы содержат достаточное большое число тестов, которые должна пройти программа, чтобы данное задание было засчитано студенту.

Организация учебного процесса с применением автоматизированной системы проверки задач позволяет преподавателю больше времени уделить разборам алгоритмов, особенностям решения задач, а не их проверке. Применение таких систем способствует росту уровня самостоятельности студентов и поиску лучших решений при подготовке домашних заданий, лабораторных работ. Кроме того, автоматическое формирование «турнирной таблицы» помогает преподавателю выставить текущий рейтинг, а у студентов появляется соревновательных дух, повышается мотивация к обучению.

Модуль сопровождения балльно-рейтинговой системы

Функция модуля для сопровождения балльно-рейтинговой системы – это выставление баллов студентам за тот или иной вид работы, ранжирование студентов, а также формирование стандартных отчётов для потребностей деканата (например, рейтинг на определённую контрольную точку или за период).

2.3. Особенности разработки модуля автоматической генерации заданий на основе шаблонов

Для успешного усвоения дисциплин, связанных с программированием, необходимо выполнить лабораторные работы и набрать не менее определённого количества баллов.

При выполнении работы студенты, с одной стороны, желают минимизировать время, затраченное на выполнение заданий и усвоение темы, с другой стороны – набрать количество баллов, соответствующее уровню их знаний и притязаний по поводу оценки.

Преподавателю необходимо решать следующие задачи:

1. генерировать большое количество заданий разного уровня сложности по разным темам, обеспечивая уникальность каждого набора заданий;

2. подбирать задания таким образом, чтобы студент с любым уровнем знаний смог достичь целей лабораторной работы и усвоить тему;

3. развивать компетенции студента, необходимые для дальнейшего профессионального роста.

Комплексное решение этих проблем может быть реализовано с помощью модуля автоматизированного формирования индивидуальных заданий для студентов.

2.3.1. Схема работы автоматизированной системы

Создание и распределение генерируемых заданий состоит из следующих этапов:

1. Создание шаблонов.

Преподаватели имеют возможность создавать шаблоны, по которым в дальнейшем будут генерироваться индивидуальные варианты заданий.

В работе рассмотрены два способа для генерации вариантов заданий: метод шаблонов и деревья И/ИЛИ.

2. Формирование работы.

На данном этапе преподаватель формирует текст работы, состоящий из одного или нескольких шаблонов заданий, указывает общее количество баллов за работу и количество баллов за каждое из заданий. Далее преподаватель отмечает студентов, которые должны будут выполнить эту работу.

3. Автоматическая генерация индивидуального набора заданий в работе.

В том случае, если общее количество баллов за работу меньше суммы баллов за каждое из заданий, для каждого студента индивидуально на основе данных о результатах сдачи прошлых работ по дисциплине отбираются

подходящие по уровню задания, которые будут включены в работу. Кроме того, для каждого студента генерируется индивидуальный вариант каждого задания из работы по шаблону.

4. Проставление результатов за выполнение работы

После того, как студент лично сдаст преподавателю работу, преподаватель должен проставить результат сдачи в журнал успеваемости.

2.3.2. Деревья И/ИЛИ

Деревья И/ИЛИ – это модель представления данных в виде дерева, каждый из узлов которого имеет тип «И» или «ИЛИ». На рисунке 3 изображено схематическое представление каждого из типов узлов:

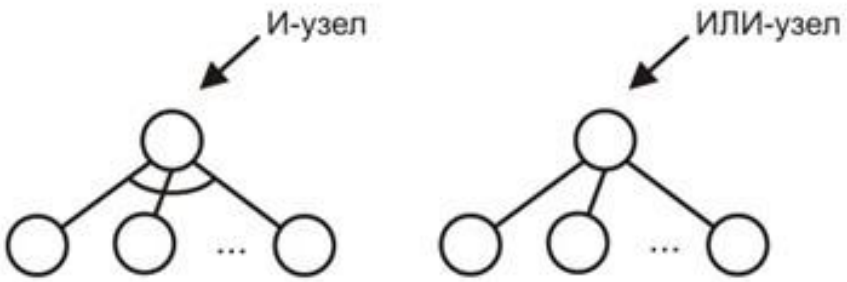

Рисунок 3. Типы узлов дерева И/ИЛИ

Вариантом дерева И/ИЛИ называется дерево, полученное из данного путём отсечения всех дуг, кроме одной, у ИЛИ-узлов. Корнем варианта будет корень исходного дерева. На рисунке 4 показано дерево И/ИЛИ и все его варианты:

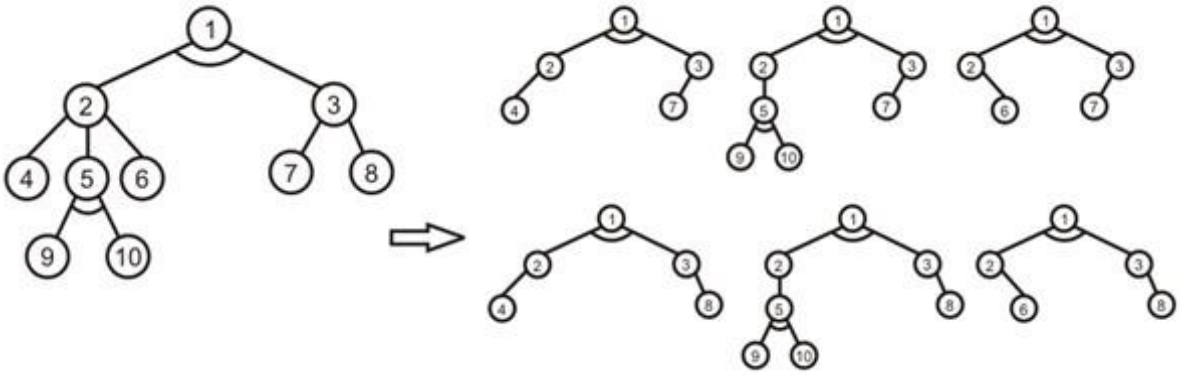

Рисунок 4. Варианты исходного дерева И/ИЛИ

Для подсчета количества вариантов можно использовать формулу:

$$\omega(z) = \begin{cases} \sum \omega(s_i^z) \text{ для И} - \text{узла} \\ \prod \omega s_i{}^z \text{ для ИЛИ} - \text{узла} \\ 1 \text{ для листа} \end{cases} \quad (1)$$

где:

z – рассматриваемый узел дерева; $\{s_i{}^z\}$

– множество сыновей узла z; $\omega(z)$ –

количество вариантов для узла z.

Перечислительные свойства деревьев И/ИЛИ представлены в работе Кручинина В.В. [15]:

Свойство 1.

Пусть дано два дерева И/ИЛИ d_1 и d_2. Известно, что деревья описывают два разных класса одного и того же объекта, тогда можно создать новое дерево И/ИЛИ Z, корнем которого будет ИЛИ-узел, а сыновьями – корни деревьев d_1 и d_2.

Свойство 2.

Пусть дано два дерева И/ИЛИ d_1 и d_2. Известно, что данные деревья описывают два разных объекта, на которых строится данный объект, тогда можно создать новое дерево И/ИЛИ Z, корнем которого будет И-узел, а сыновьями – корни деревьев d_1 и d_2.

Для рассмотрения построения дерева И/ИЛИ рассмотрим следующий пример:

С помощью логической переменной и оператора присваивания определить:

1. Каждое из чисел x, y, z положительно;

2. Цифра 3 входит в запись трехзначного числа k.

Для того, чтобы построить дерево И/ИЛИ по данному примеру, сначала необходимо перевести запись в форму шаблона. Для записи шаблона деревьев И/ИЛИ предлагается использовать скобочную нотацию [5]: круглыми скобками обозначаются И-узлы, фигурными – ИЛИ-узлы, узлы без скобок

являются листами. Рассмотренное задание в виде шаблона на программном языке выглядит следующим образом (см. листинг 1).

```
Main ("С помощью логической переменной и оператора присваивания определить:
    1)  " A{"каждое", "только одно"}, " из чисел x, y, z", B{"
положительно;"," отрицательно;"},"
    2)  Цифра ", [0..10], " входит в запись ", [2..6], "-значного числа k.")
```

Листинг 1. Шаблон задания для представления в виде деревьев И/ИЛИ.

Результат построения дерева И/ИЛИ по данному шаблону представлен на рисунке 5:

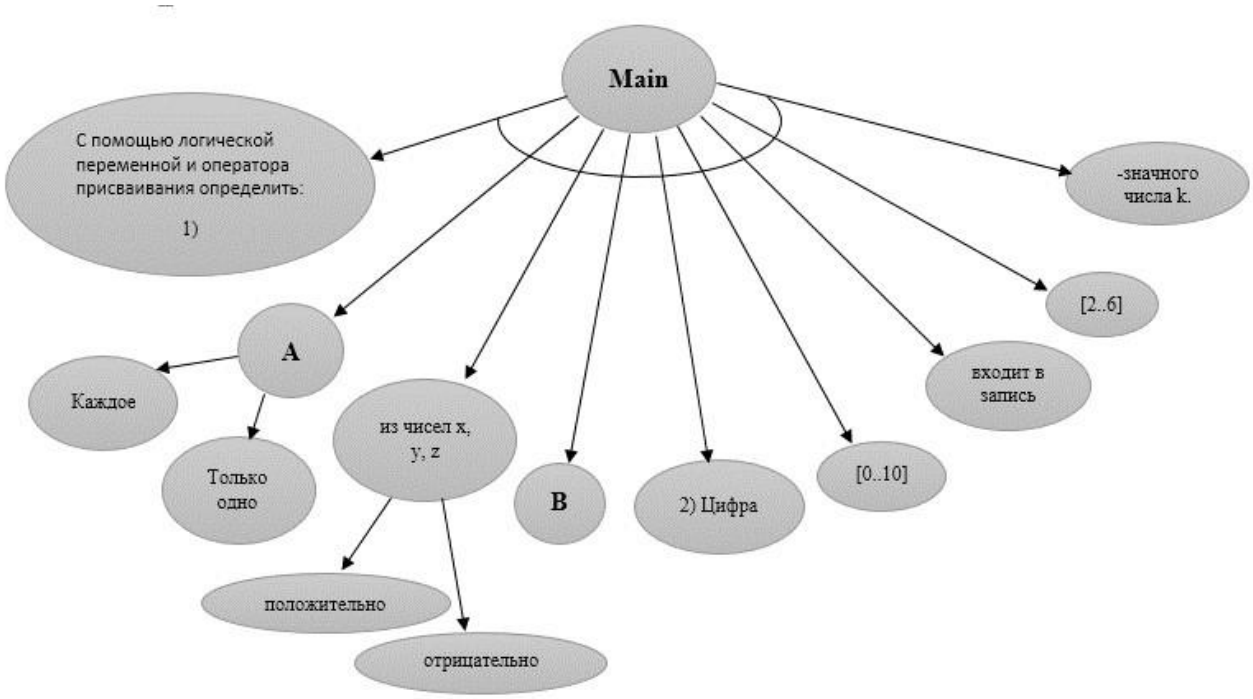

Рисунок 5. Пример построения дерева И/ИЛИ

Выполнив обход дерева и записав отобранные узлы, получим конкретный вариант заданий, например:

С помощью логической переменной и оператора присваивания определить:

• *Только одно из чисел x, y, z отрицательно;*

• *Цифра 0 входит в запись 4-значного числа k.*

Данный метод является довольно мощным инструментом. При дальнейшем его развитии можно описывать задания различного диапазона дисциплин, формировать блоки решений, ответов, подсказок и т.д. В приложении 2 представлены варианты шаблонов и заданий, сгенерированных по методу деревьев И/ИЛИ.

2.3.3.Метод шаблонов для генерации вариантов заданий

Метод шаблонов, описанный в монографии Кручинина В.В. [16], заключается в использовании типовых шаблонов и морфологического синтеза.

В качестве примера рассмотрим следующую задачу:

Заполнить одномерный массив из 100 элементов целыми значениями, заданными случайным образом из интервала [-50;50]. Вычислить сумму квадратов элементов данного массива, больших 15. На экран вывести исходный массив и полученный результат.

В качестве параметров-чисел могут выступать 100, -50 и 50. Можно обобщить эти параметры до N, a, b и установить естественные ограничения: $N > 0$, $a < b$. Не указанными остаются границы каждого из параметров a, b, способ задания N и его верхняя граница. Параметризировать можно и часть текста задания, например, вычислять можно: сумму квадратов элементов, произведение положительных элементов, количество чисел, кратных 3. Шаблон

Конкретный вариант параметра-текста всегда выбирается из некоторого фиксированного набора значений, а вариант параметра-числа может быть выбран из множества значений или вычислен с помощью некоторой функции.

Пусть в шаблоне имеется n параметров и пусть количество значения параметра p_i равно m_i. Тогда максимальное количество вариантов, генерируемое системой, будет вычисляться по формуле:

$$N = m_1 + m_2 + \ldots + m_n \qquad (2)$$

Некоторые параметры могут зависеть друг от друга, поэтому реальное количество вариантов будет меньше.

Для представления шаблона заданий требуется определить специальный язык представления.

Таким образом, задание в системе может быть представлено в виде шаблона (см. листинг 2), состоящего из тела задания и дополнительной информации, определяющей состав параметров и ограничения на генерируемые значения.

<body – начало тела**>**

Заполнить одномерный массив из <p1> элементов целыми значениями, заданными случайным образом из интервала [<p2>]. Вычислить <p3>, больших некоторого числа <p4>. На экран вывести исходный массив и полученный результат.

</body – конец тела**>**

<param – параметры**>**

<p1 – начало описания параметра**>**

<**type = number, integer** – тип параметра>

<condition – условие**>**

<v=random>

<v>0>

</condition – конец условия**>**

</p1 – конец описания параметра**>**

<p2 – начало описания параметра**>**

<**type = number, real** – тип параметра>

<**condition** – условие>

<v1=random>

<v2=random>

<v1<v2>

</condition – конец условия**>**

</p2 – конец описания параметра**>**

<p3 – начало описания параметра**>**

<type=text – тип параметра**>**

<select – начало выбора**>**

<v= “ сумму квадратов элементов ”**>**

<v= “ произведение положительных элементов ”>

<v= “ произведение положительных элементов”>

</select – конец выбора>

</p3 – конец описания параметра>

<p4 – начало описания параметра>

<type = number, integer – тип параметра>

<condition – условие>

<v=random>

<v>0>

</condition – конец условия>

</p4 – конец описания параметра>

</param – конец описания параметров>

Листинг 2. Шаблон задания

Возможно два подхода к разработке подобных языков: собственная разработка или использование языка XML. Второй вариант выгоднее, так как в первом случае придется разрабатывать дополнительно интерпретатор языка описания, в то время как во втором можно будет воспользоваться готовыми библиотеками для работы с XML.

В приложении 1 представлены варианты шаблонов и заданий, сгенерированных по методу шаблонов.

2.4. Генерация вариативных тестовых заданий

При реализации модуля обучающего тестирования нужно учесть возможность получения заданий на нескольких языках программирования.

Пусть Gk – множество характеристик сложности объекта, Qk – множество показателей уровня интерактивности объекта, Z^k – множество структур учебных заданий, состоящих из базовых элементов (множество A),

основных зависимостей (множество B), функциональных элементов

(множество C), функциональных зависимостей (множество D).

$$z_i^k = ({}_c b_1^1 \ \ldots\ {}_b b_m^p),\ a_j \in A,\ b_j \in B,\ c_j \in C,\ d_j \in D \qquad (3)$$

$$\begin{matrix} a_1 & \ldots & a_r \\ d_1 & \ldots & d_s \end{matrix}$$

Множество объектов учебного проверочного задания описывает неоднородную систему T, где t_i^k – объект учебного проверочного задания на языке программирования k. Данный объект можно описать набором характеристик.

$$t_i^k = \{z_i^k, g_i^k, q_i^k\},\ z_i^k \in Z^k,\ g \in G^k,\ q_i^k \in Q^k. \qquad (4)$$

Характеристиками объекта могут быть: уровень сложности задания, уровень интерактивности задания, совокупность основных, функциональных элементов и функциональных зависимостей, которые определяются в зависимости от показателя сложности задания.

Авторами работы предложен алгоритм формирования тестового задания, исходными данными для которого являются алгоритмы на псевдокоде (рис. 6).

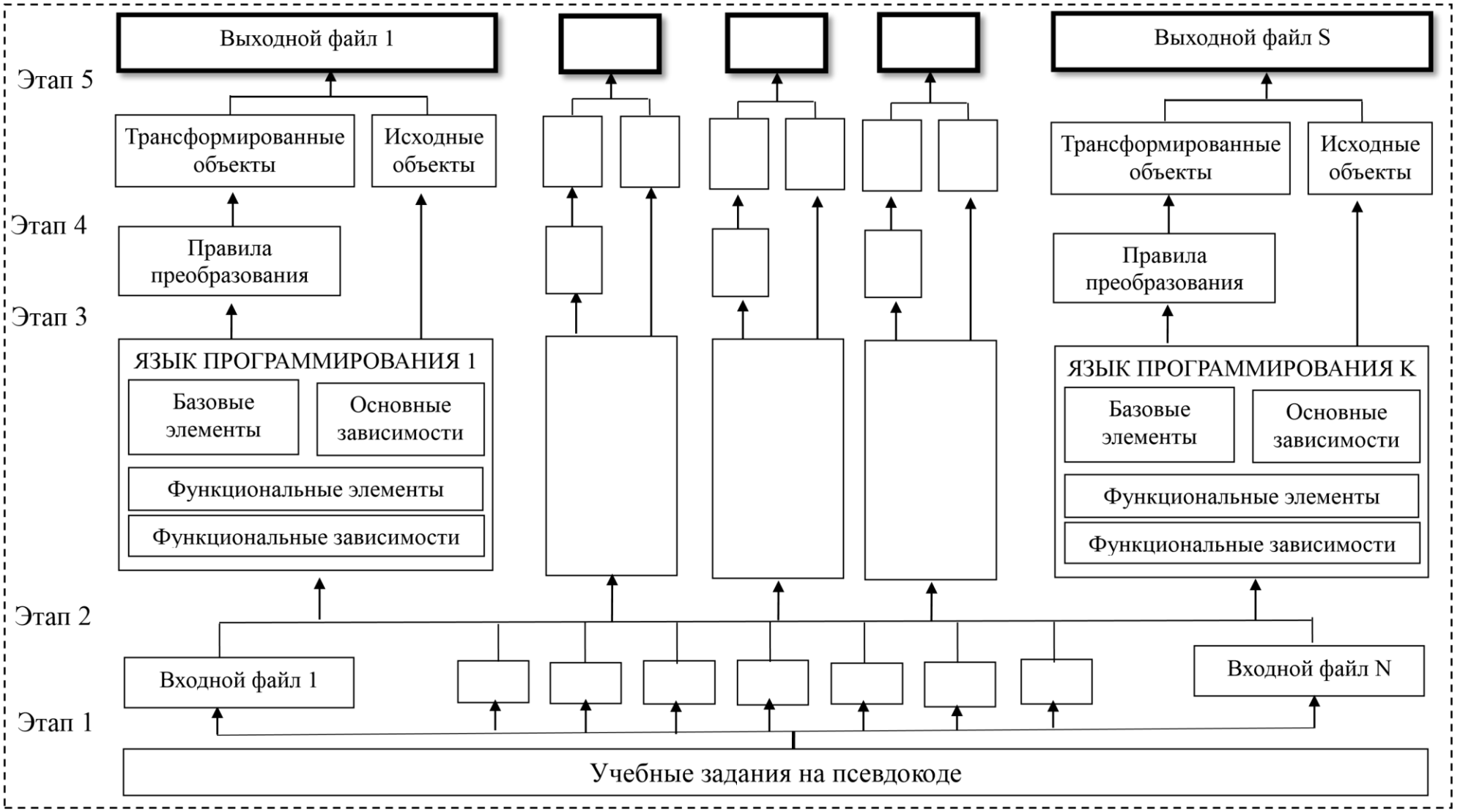

Рисунок 6. Схема формирования учебного проверочного задания по программированию

Формирование тестового задания состоит из пяти этапов.

Этап 1 «Генерирование входных данных». На данном этапе алгоритмы, которые будут включены в тестовые задания, записываются на псевдокоде, и формируются входные файлы.

Этап 2 «Трансляция в язык программирования». Этот этап подразумевает перевод алгоритмов из входных файлов на выбранный язык программирования, при этом в структуре файла выделяются следующие объекты: базовые элементы, основные зависимости, функциональные элементы, функциональные зависимости.

Этап 3 «Модифицирование объектов» подразумевает искажение некоторых объектов (или их частей) по заданным правилам. Правила и их количество определяется уровнем сложности и интерактивности задания.

Этап 4 «Распределение объектов». На этом этапе базовые элементы, основные зависимости, функциональные элементы, функциональные зависимости сортируются на *трансформированные* и *исходные*.

Этап 5 «Сборка учебного проверочного задания». На этом этапе происходит объединение *трансформированных* и *исходных* объектов и формирование тестовых заданий с учётом выбранного языка программирования, уровня сложности, интерактивности, правил трансформирования объектов.

Пример. Необходимо сформировать тестовое задание по теме «Нахождение чисел Фибоначчи».

Этап 1. «Генерирование входных данных». Формируется входной файл с заданием, записанным на псевдокоде (см. листинг 3).

Этап 2. «Трансляция в язык программирования». Используя словарь лексем соответствующего языка программирования, выделили объекты: входные данные, выходные данные, цикл for, начало и конец блока операторов внутри цикла (см. листинг 4).

```
НАЧАЛО
ВВОД n
f0 = 0 ;
f1 = 1 ;
ПОКА_С i = 2 ДО_ВВ n+1 ДЕЛАЙ_СВВ
  НАЧАЛО
    fn = f0 + f1
;     f0 = f1 ;
f1 = fn ;
  КОНЕЦ
ВЫВОД fn
КОНЕЦ
```

Листинг 3.Учебное задание на псевдокоде

```
cin >> n f0 = 0; f1 = 1;
for (i = 2; i < n+1; ++ i )
  {       fn = f0 +
f1 ;      f0 = f1 ;
f1 = fn ;
  }
cout << fn
```

Листинг 4.Учебное задание на языке программирования C++

Этап 3. «Модифицирование объектов». Определяются объекты строки кода, которые будут модифицированы. Например, в строке «for (i = 2; i < n; ++ i)» меняется знак «<» на знак «>».

Этап 4. «Распределение объектов». Строки кода с модифицированными объектами записывается в отдельный список (трансформированные объекты).

Этап 5. «Сборка учебного проверочного задания». Строка с модифицированным объектом помещается на то же место в исходный код на выбранном языке программирования (см. листинг 5).

```
cin >> n f0 = 0; f1 = 1;
for (i = 2; i > n+1; ++ i )
  {       fn = f0 +
f1 ;      f0 = f1 ;
f1 = fn ;
  }
cout << fn
```

Листинг 5. Код на языке программирования C++ к тестовому заданию

Тестовое проверочное задание формулируется так: «Приведённый код должен выводить на консоль n-ое число Фибоначчи (n >=2). Укажите, в какой строке допущена ошибка?»

ГЛАВА 3. ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

3.1. Используемое программное обеспечение

Для разработки системы использовался язык программирования PHP MVC с использованием yii2 framework (среда разработки NetBeansv.8.0.2).

Используемая база данных располагается в СУБД PostgreSQL (используемый клиент – pgAdminIII v.1.20.0).

PostgreSQL – это SQL СУБД с открытым исходным кодом. PostgreSQL является кроссплатформенным продуктом и работает не только на платформе Microsoft Windows, но и в широком диапазоне диалектов Unix (Linux, FreeBSD, Solaris и т.д.). Для работы с PostgreSQL существует множество интерфейсов и библиотек взаимодействия из других языков программирования: Java, Perl, Python, Ruby, C, C++, PHP, Lisp.

PHP – скриптовый язык программирования общего назначения, широко применяемый для разработки web-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков программирования, применяющихся для создания web-сайтов.

Yii2 – это высокопроизводительный PHP-фреймворк, реализующий концепцию MVC и предназначенный для быстрой разработки современных web-приложений.

Концепция MVC – это шаблон проектирования Модель-ПредставлениеКонтроллер (model-view-controller), который широко применяется в webпрограммировании. Концепция MVC предназначена для разделения бизнеслогики и пользовательского интерфейса, чтобы разработчики могли легко изменять отдельные части приложения, не затрагивая другие. В MVC модель предоставляет данные и правила бизнес-логики, представление отвечает за пользовательский интерфейс, а контроллер обеспечивает взаимодействие между моделью и представлением (см. рис. 7).

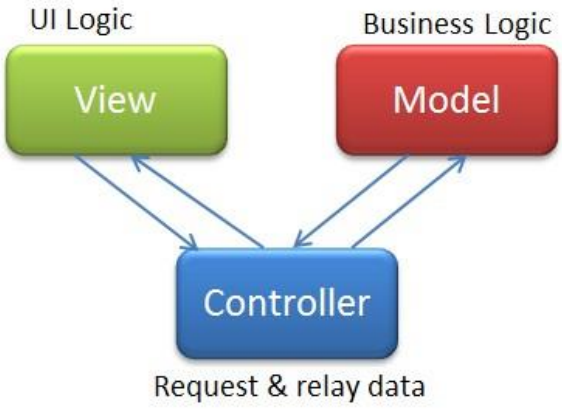

Рисунок 7. Схема шаблона MVC

3.2. Проектирование информационного обеспечения

Для хранения необходимой информации была спроектирована база данных, содержащая 19 таблиц (см. рис. 8).

Таблицы Users и Persons служат для идентификации в системе пользователей. В таблице Roles содержится информация о ролях входа пользователей.

Таблица 1. Структура таблицы Users

Имя поля	**Тип**	**Описание**
id	serial	Первичный ключ, идентификатор, автоинкремент.
person	integer	Внешний ключ, ссылка на Persons – пользователь.
login	character varying	Логин пользователя.
password	character varying	Пароль пользователя.
role	integer	Внешний ключ, ссылка на Roles – роль пользователя.

Таблица 2. Структура таблицы Persons

Имя поля	**Тип**	**Описание**

id	serial	Идентификатор записи, первичный ключ, автоинкремент.
sname	character varying	Фамилия пользователя.
name	character varying	Имя пользователя.
lname	character varying	Отчество пользователя.

Таблица 3. Структура таблицы Roles

Имя поля	**Тип**	**Описание**
id	serial	Идентификатор записи, первичный ключ, автоинкремент.
rolename	character varying	Имя роли пользователя.

В таблицах Students, Groups и Specialities хранится информацию о студентах.

Таблица 4. Структура таблицы Students

Имя поля	**Тип**	**Описание**
id	serial	Идентификатор записи, первичный ключ, автоинкремент.
person	integer	Внешний ключ, ссылка на Person – пользователь.
group	integer	Внешний ключ, ссылка на Groups – группа.

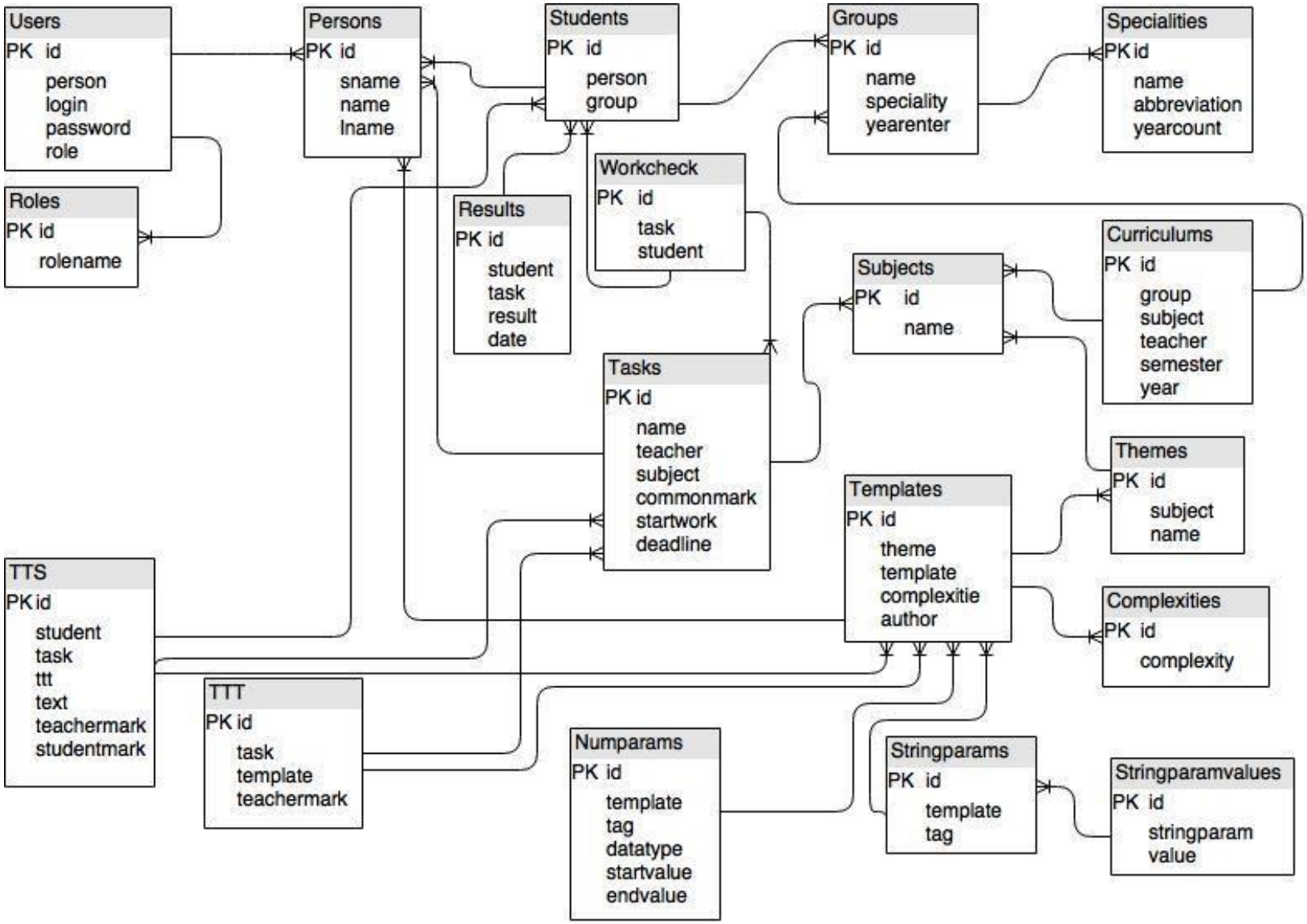

Рисунок 8. Схема базы данных

Таблица 5. Структура таблицы Groups

Имя поля	**Тип**	**Описание**
id	serial	Идентификатор записи, первичный ключ, автоинкремент.
name	character varying	Название и/или номер группы.
speciality	integer	Внешний ключ, ссылка на Specialities – специальность.
yearenter	integer	Год потока группы.

Таблица 6. Структура таблицы Specialities

Имя поля	**Тип**	**Описание**
id	serial	Идентификатор записи, первичный ключ, автоинкремент.
name	character varying	Название специальности.
abbreviation	character varying	Аббревиатура специальности.
yearcount	character varying	Количество лет обучения.

Таблица Curriculums представляет собой учебный план – посеместровое распределение дисциплин с указанием учебных групп и преподавателей, назначенных на дисциплины.

Таблица 7. Структура таблицы Curriculums

Имя поля	**Тип**	**Описание**
id	serial	Идентификатор записи, первичный ключ, автоинкремент.
group	integer	Внешний ключ, ссылка на Groups – номер группы.
subject	integer	Внешний ключ, ссылка на Subjects – дисциплина.
semester	integer	Номер семестра обучения.
teacher	integer	Внешний ключ, ссылка на Persons – пользователи (преподаватель).
year	character varying	Учебный год.

Таблица 8. Структура таблицы Subjects

Имя поля	**Тип**	**Описание**
id	serial	Идентификатор записи, первичный ключ, автоинкремент.
name	character varying	Название дисциплины.

Таблица 9. Структура таблицы Themes

Имя поля	**Тип**	**Описание**
id	serial	Идентификатор записи, первичный ключ, автоинкремент.
subject	integer	Внешний ключ, ссылка на Subjects – дисциплина.
name	character varying	Название темы дисциплины.

В таблице Templates расположена информация о шаблонах заданий с указанием конкретной темы и сложности. Справочная таблица Complexities

содержит информацию о уровнях сложности. Данные о работах хранятся в таблице Tasks.

Таблица 11. Структура таблицы Templates

Имя поля	**Тип**	**Описание**
id	serial	Идентификатор записи, первичный ключ, автоинкремент.
theme	integer	Внешний ключ, ссылка на Themes – тема шаблона.
template	character varying	Текст шаблона с параметрами.
author	integer	Внешний ключ, ссылка на Persons – пользователь (преподаватель).
complexitie	integer	Внешний ключ, ссылка на Complexities – сложность шаблона.

Таблица 10. Структура таблицы Complexities

Имя поля	**Тип**	**Описание**
id	serial	Идентификатор записи, первичный ключ, автоинкремент
complexity	character varying	Сложность задания: «1» – просто; «2» – средней сложности; «3» – сложно.

Таблица 12. Структура таблицы Tasks

Имя поля	**Тип**	**Описание**
id	serial	Идентификатор записи, первичный ключ, автоинкремент.
name	character varying	Название работы.
subject	integer	Внешний ключ, ссылка на Subjects – дисциплина.

teacher	integer	Внешний ключ, ссылка на Persons – пользователь (преподаватель).
commonmark	integer	Общее количество баллов за работу.
startwork	date	Дата начала выполнения работы.
deadline	date	Срок сдачи работы.

Для формирования работ используются сводные таблицы TTT и TTS, которые хранят ссылки на Templates и Tasks (многие задания Tasks могут содержать многие шаблоны Templates).

Таблица 13. Структура таблицы TTT

Имя поля	**Тип**	**Описание**
id	serial	Идентификатор записи, первичный ключ, автоинкремент.
task	integer	Внешний ключ, ссылка на Tasks – работа.
template	integer	Внешний ключ, ссылка на Templates – шаблон задания.
teachermark	integer	Количество баллов, начисляемое за успешное выполнение задания.

Таблица 14. Структура таблицы TTS

Имя поля	**Тип**	**Описание**
id	serial	Идентификатор записи, первичный ключ, автоинкремент.
task	integer	Внешний ключ, ссылка на Tasks – работа.
template	integer	Внешний ключ, ссылка на Templates – шаблон задания.
student	integer	Внешний ключ, ссылка на Students – студент.
teachermark	integer	Количество баллов, начисляемое за успешное выполнение задания.

studentmark	integer	Фактически набранное количество баллов.
text	character varying	Текст задания (сгенерированный по шаблону).

За числовые и строковые параметры отвечают таблицы Numparams и Stringparams соответственно, дополнительная таблица Stringparamvalues необходима для хранения вариантов значений строковых параметров.

Таблица 15. Структура таблицы Numparams

Имя поля	**Тип**	**Описание**
id	serial	Идентификатор записи, первичный ключ, автоинкремент.
datatype	integer	Тип данных: «1» – целые числа; «2» – дробные числа.
startvalue	real	Начальное значение диапазона для генерации числа.
endvalue	real	Конечное значение диапазона для генерации числа.
template	integer	Внешний ключ, ссылка на Templates – шаблон, в котором присутствует параметр.
tag	character varying	Название параметра в шаблоне.

Таблица 16. Структура таблицы Stringparams

Имя поля	**Тип**	**Описание**
id	serial	Идентификатор записи, первичный ключ, автоинкремент.
template	integer	Внешний ключ, ссылка на Templates – шаблон, в котором присутствует параметр.
tag	character varying	Название параметра в шаблоне.

Таблица 17. Структура таблицы Stringparamvalues

Имя поля	**Тип**	**Описание**

id	serial	Идентификатор записи, первичный ключ, автоинкремент.
stringparam	integer	Внешний ключ, ссылка на Stringparams – строковый параметр.
value	character varying	Значение строкового параметра.

Назначение студентов на выполнение заданий осуществляется посредством записи данных в таблицу Workcheck, а результаты работ хранятся в таблице Results.

Таблица 18. Структура таблицы Workcheck

Имя поля	**Тип**	**Описание**
id	serial	Идентификатор записи, первичный ключ, автоинкремент.
task	integer	Внешний ключ, ссылка на Tasks – работа.
student	integer	Внешний ключ, ссылка на Students – студент, которому назначена на выполнение работа.

Таблица 19. Структура таблицы Results

Имя поля	**Тип**	**Описание**
id	serial	Идентификатор записи, первичный ключ, автоинкремент.
student	integer	Внешний ключ, ссылка на Students – студент.
task	integer	Внешний ключ, ссылка на Tasks – работа.
result	integer	Количество баллов, начисленное за выполнение работы.
date	date	Дата сдачи работы студентом.

3.3. Проектирование программного обеспечения автоматизированной системы обмена генерируемыми заданиями

Программный продукт представляет собой web-приложение, реализованное по шаблону MVC – model-view-controller (см. рис. 9).

Главный контроллер сайта – SiteController – содержит основные функции, управляющие работой приложения, такие как: переходы на разные представления, обработка данных для вывода их на экран, выполнение действий на стороне сервера.

Представления представляют собой PHP-файлы, содержащие разметку web-страниц приложения и предназначенные для вывода на экран данных. В большинстве случаев к представлениям подключены JS-скрипты для обработки событий и действий пользователя в web-приложении.

Модели являются по сути своей описанием и определением структур данных. Большая часть моделей соответствуют таблицам в базе данных (добавляются в проект с помощью Gii Extension for Yii2), однако могут быть дополнены необходимыми методами.

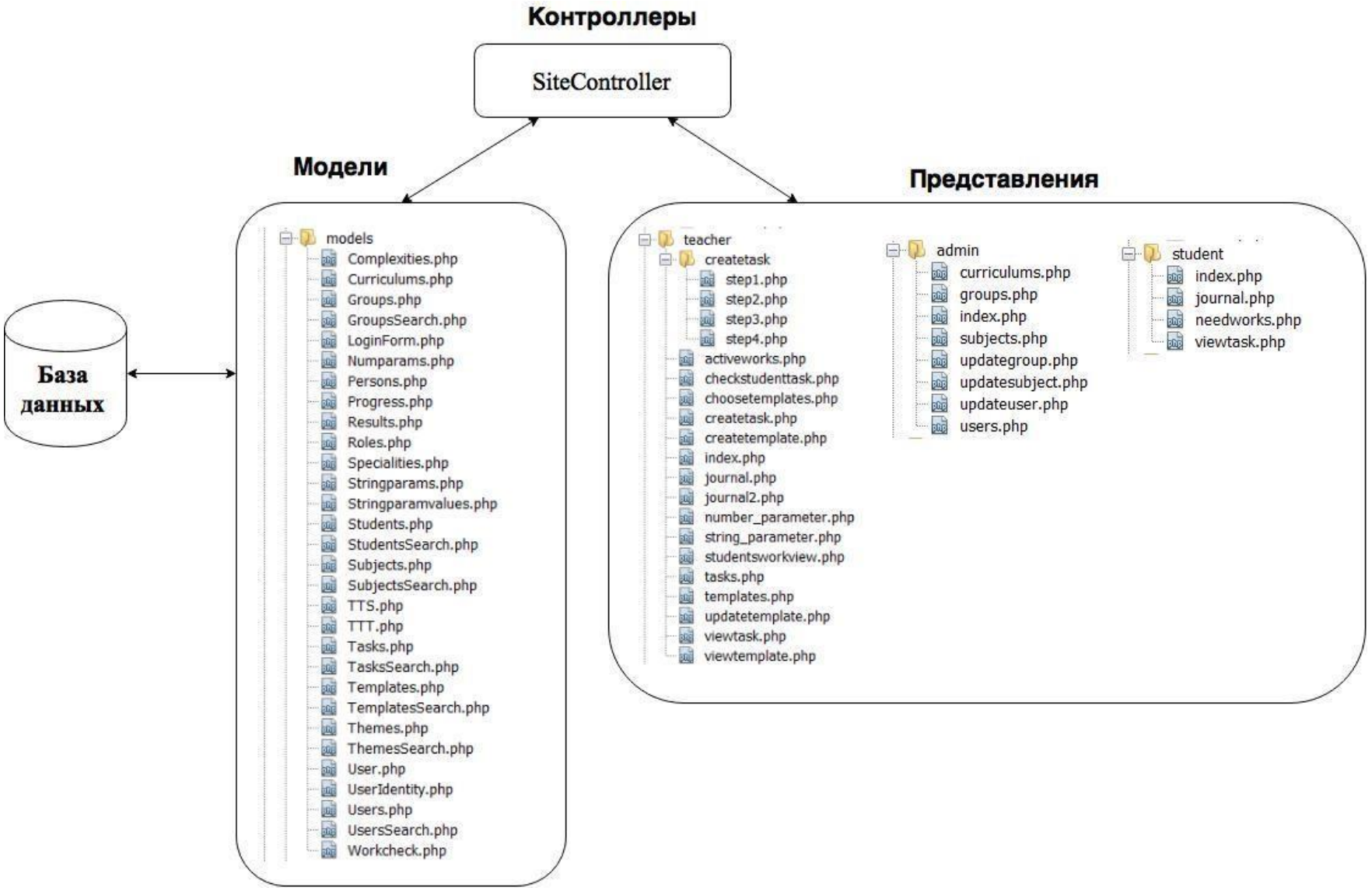

Рисунок 9. Логическая организация данных

Одной из основных моделей является UserIdentity, которая содержит данные о пользователе и с помощью SQL-запросов получает сводную информацию из базы данных. Методы UserIdentity:

1) getPerson() – получение идентификатора текущего пользователя;
2) ActiveWorks($teacher) – получение данных о текущих работах преподавателя на проверку;
3) JournalTeacher($teacher) – получение информации обо всех дисциплинах и работах преподавателя за все время;
4) JournalStudent($person) – получение информации обо всех дисциплинах и работах студента по всем семестрам;
5) getAllStudents() – получение данных по всем студентам;
6) LoadStudentsWorkView($task) – получение текста работы для студента;
7) LoadNeedWorks($studID) – получение информации о работах, которые необходимо выполнить студенту;
8) LoadAlreadyDoneWorks ($studID) – получение информации об уже выполненных студентом работах;
9) getTemplates($task, $user) – получение списка шаблонов работы для студента;
10) getStats($person, $subject) – определение коэффициентов выполнения студентом работ по данной дисциплине;
11) getTask($person, $task) – получение индивидуального варианта работы для студента (выполняется в момент первого обращения к работе);
12) getParams($template) – получение конкретных значений параметризированной части шаблона;
13) getUserInfo($id) – получение информации о текущем пользователе;
14) getGroupInfo($id) – получение информации об учебной группе;
15) getStudentsOfThisGroupOrEmpty($id) – получение списка студентов, которые принадлежат к данной группе либо группа которых не определена;

16) getCurriculum($id) – получение информации об учебном плане;
17) getSubjectsAndThemes($id) – получение информации о дисциплинах и темах по этим дисциплинам.

При автоматической генерации индивидуального варианта задания по шаблону используются методы getTask($person, $task) и getParams($template), с помощью которых в каждом шаблоне задания, которые включены в данную работу, параметризуемые части заменяются конкретными значениями.

Для определения коэффициентов по выполнению работ различного уровня сложности для студента используется метод getStats ($person, $subject).

3.4. Руководство пользователя

Главная страница web-приложения содержит поля для ввода реквизитов доступа: Администратор, Студент, Преподаватель.

Для каждой роли определено собственная навигационная панель, расположенная вверху страницы, собственный набор возможных действий и функций системы.

Роль «Студент»

У студента в навигационной панели находятся пункты «Текущие работы», «Учебный план» и «Выход».

На странице «Текущие работы» (рис. 10) выводится информация о работах, которые студент должен выполнить в ближайшее время (срок сдачи работы указан). Выбранную работу студент может распечатать или просмотреть в браузере.

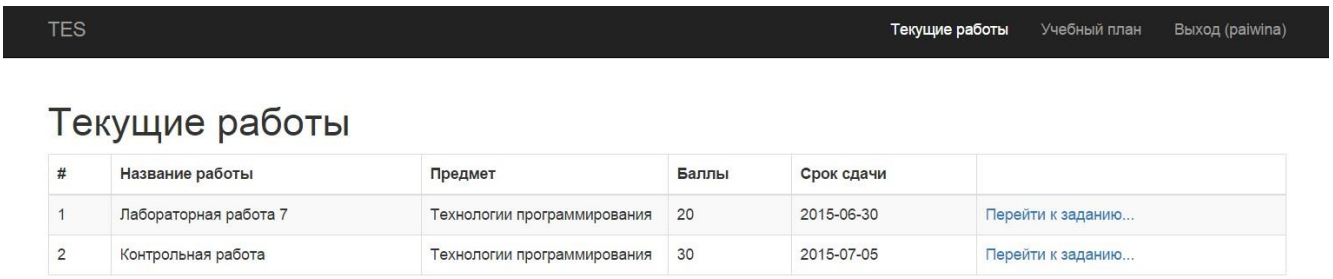

Рисунок 10. Список текущих работ студента

Работы в кабинете студента представлены уже в виде сгенерированного индивидуального варианта по исходному шаблону, составленному преподавателем (рис. 11).

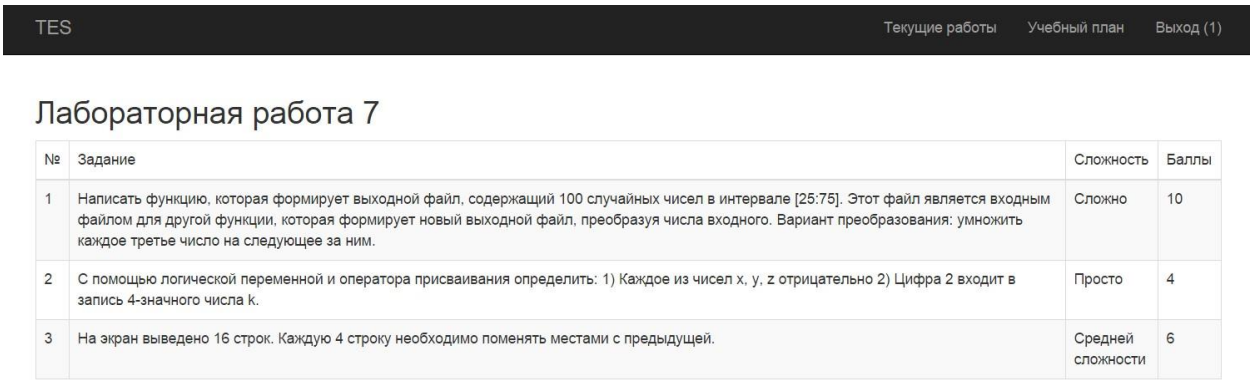

Рисунок 11. Просмотр работы студентом

Во вкладке «Учебный план» располагается вся учебная информация по семестрам. При переключении вкладок семестров выводится информация о дисциплинах и преподавателях данной дисциплины выбранного семестра, а также общее число работ по дисциплине.

Для просмотра подробной информации о работах по дисциплинам и результатов за их выполнение необходимо перейти по ссылке «Просмотр заданий»

В окне просмотра работ по дисциплине (рис. 12) отображаются все работы с указанием количества набранных баллов за их выполнение (зеленым цветом

отмечены работы, за которые студент набрал максимум баллов, красным – работы, за которых студент получил 0 баллов).

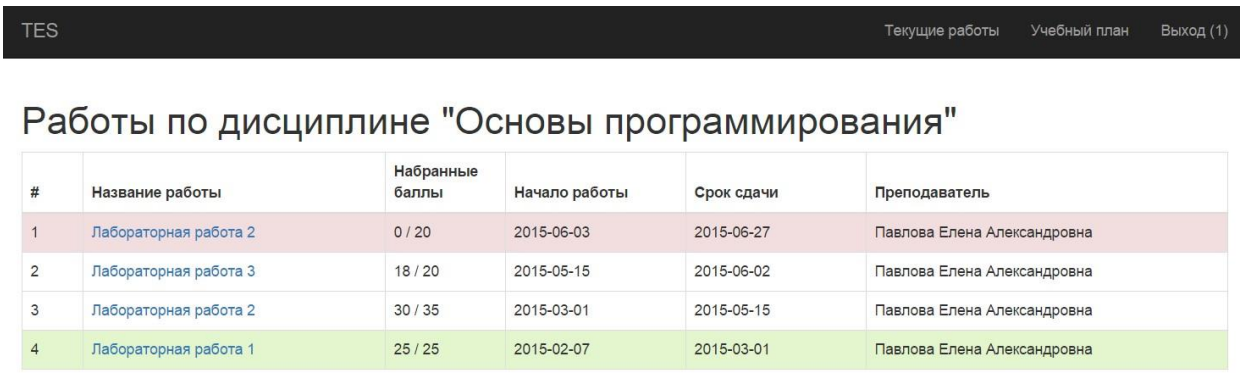

Рисунок 12. Просмотр списка заданий по дисциплине учебного плана

Роль «Преподаватель»

Навигационная панель преподавателя содержит пункты «Проверка работ», «Журнал успеваемости», «Работы», «Шаблоны» и «Выход».

На странице проверки работ отображены все текущие (дата сдачи которых не истекла) работы, сгруппированные по дисциплинам преподавателя (рис. 13).

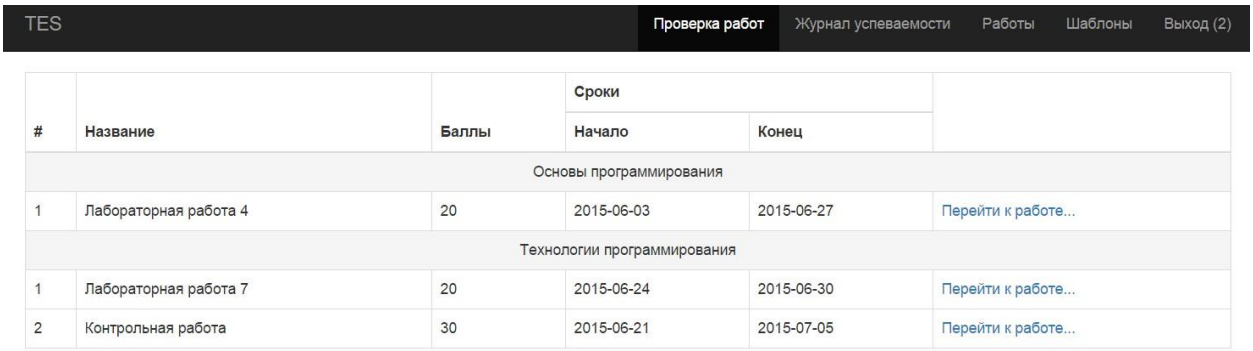

Рисунок 13. Список текущих работ для проверки преподавателем

Для просмотра списка работ студентов необходимо перейти по ссылке «Перейти к работе…».

В открывшемся окне будет приведен список всех назначенных на выполнение этой работы студентов с указанием набранных ими баллов. Непроверенные работы выделены красным цветом (рис. 14).

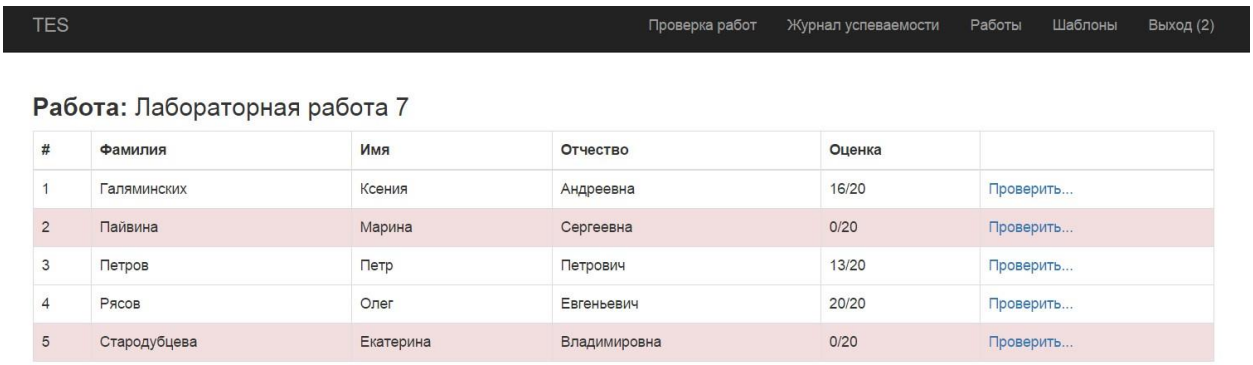

Рисунок 14. Список работ для проверки

Для проверки работы (проставления баллов за задания) необходимо перейти по ссылке «Проверить…».

В открывшемся окне будет представлен текст работы студента и элементы для ввода баллов за каждое задание с указанием максимально возможного количества баллов (рис. 15). В том случае, если преподаватель по ошибке введет значение, большее максимального, данное задание будет выделено красным цветом и сохранение результатов будет невозможно до устранения ошибочного значения.

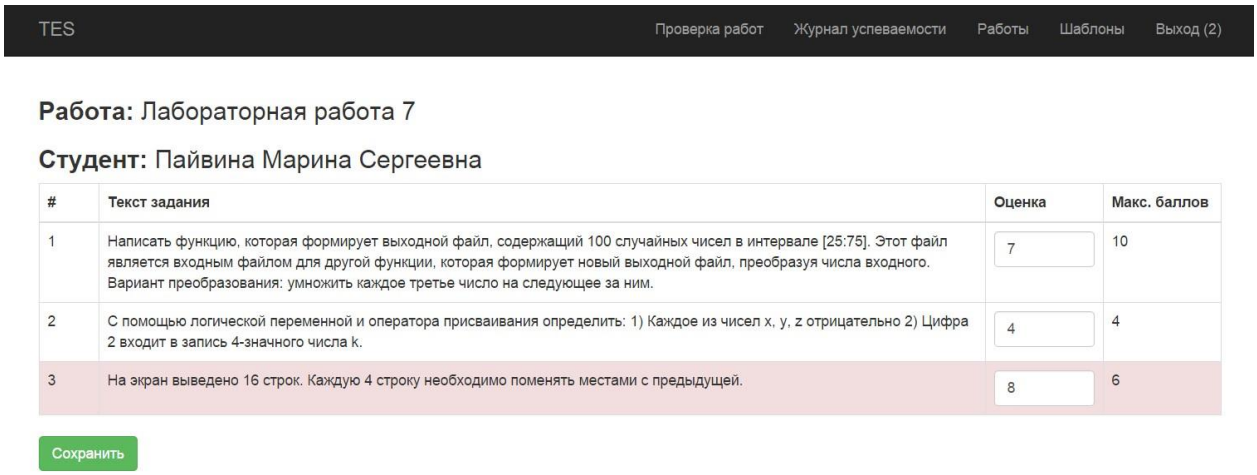

Рисунок 15. Проверка работы студента

В разделе «Журнал успеваемости» преподаватель имеет возможность просмотреть всю историю работ по дисциплинам (рис. 16). При выборе дисциплины в правой таблице подгружается информация обо всех работах по данной дисциплине.

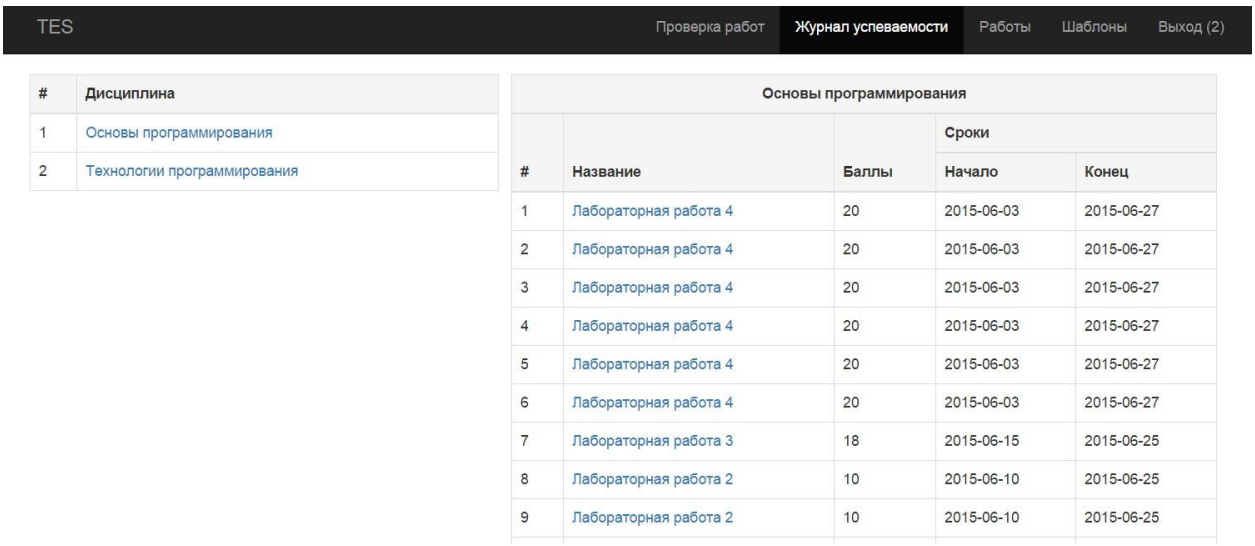

Рисунок 16. Просмотр журнала успеваемости

На странице работ отображены все работы, созданные преподавателем (рис. 17).

Имеется возможность сортировать данные в таблице или использовать фильтр для поиска данных.

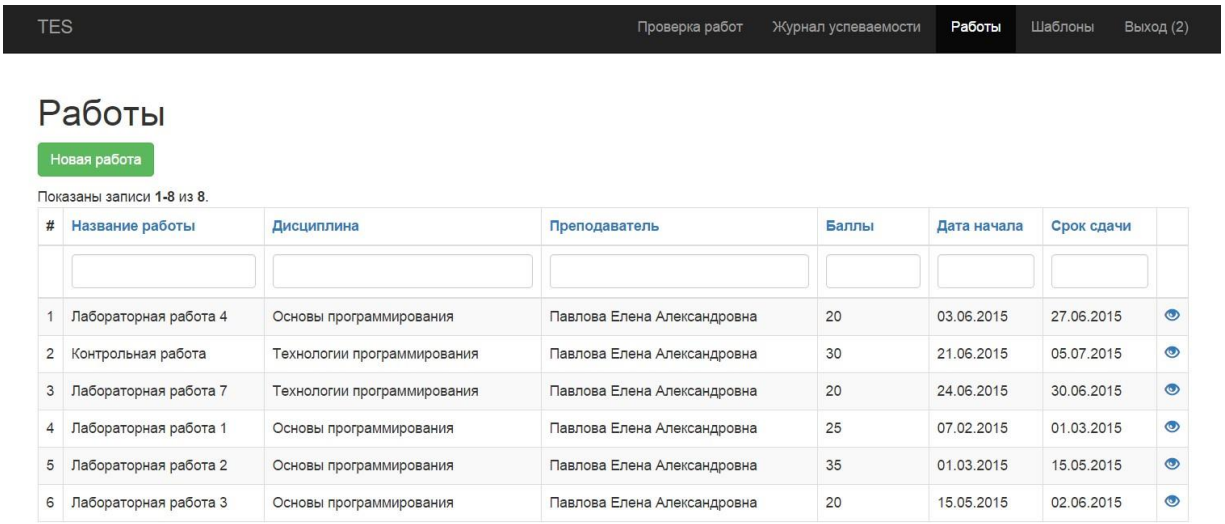

Рисунок 17. Просмотр списка работ

При нажатии на кнопку «Новая работа» открывается форма для формирования новой работы, выполняемое пошагово.

На первом шаге заполняется общая информация о работе: название работы, дисциплина, начало и срок сдачи, баллы за работу.

На втором шаге выбираются шаблоны заданий, которые будут включены в работу (рис. 18).

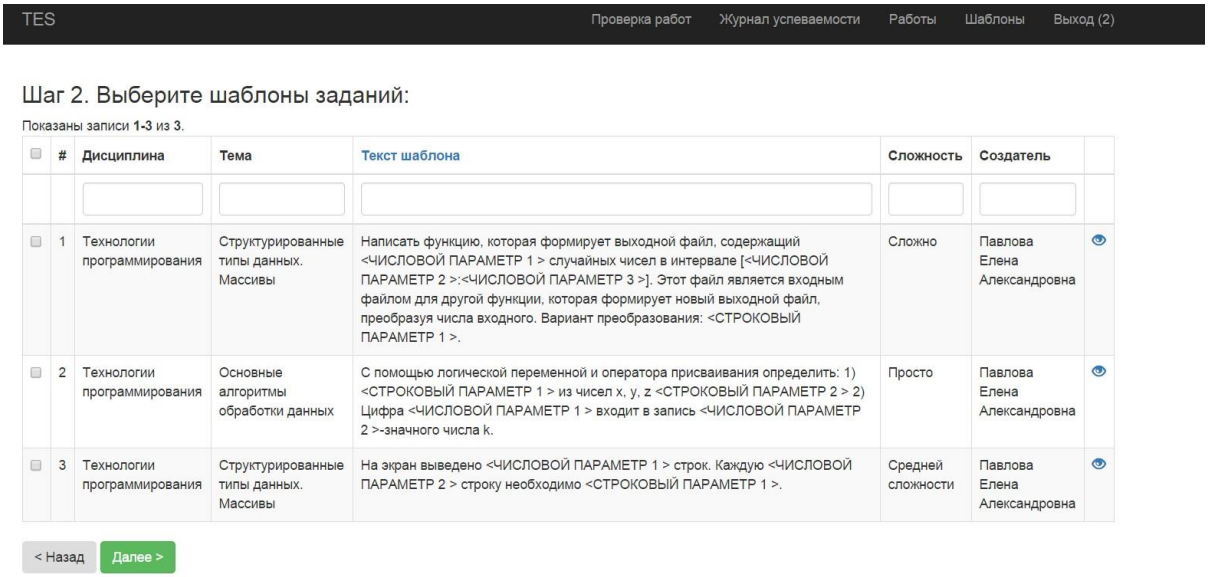

Рисунок 18. Второй шаг создания работы

На третьем шаге проставляются баллы за каждое из заданий в работе (рис. 19). Сумма балов за задания может быть больше либо равна указанному ранее количеству баллов за работу. В том случае, если сумма будет равна общему количеству баллов, все задания будут включены в работу; если сумма больше общего количества баллов, то для каждого студента будет сформирован индивидуальный набор заданий на основе результатов сдачи студентом предыдущих работ по дисциплине.

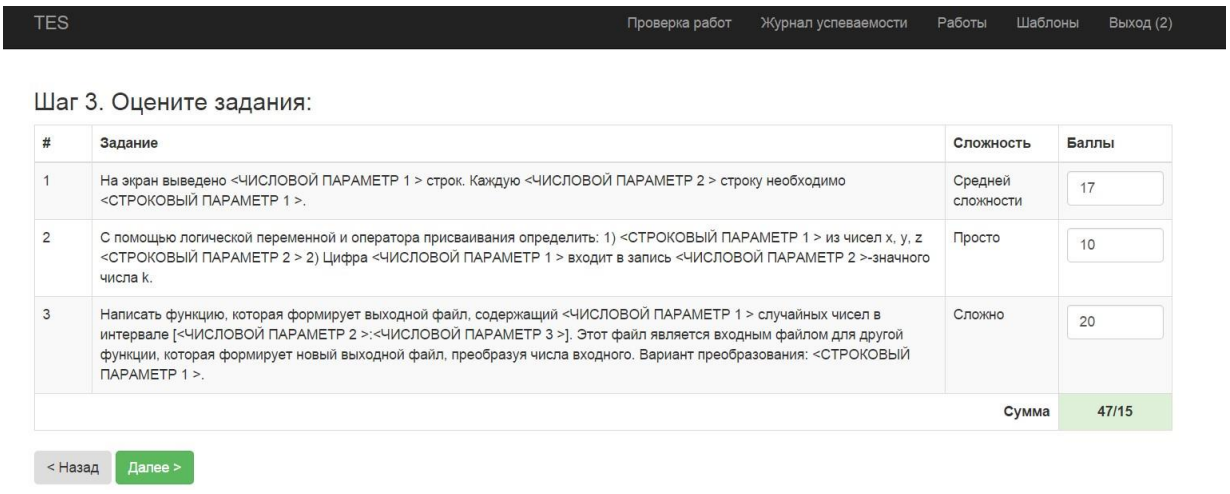

Рисунок 19. Третий шаг создания работы

На четвертом шаге назначаются студенты, которые должны выполнить данную работу (рис. 20). Предусмотрена фильтрация данных для более удобного поиска студентов.

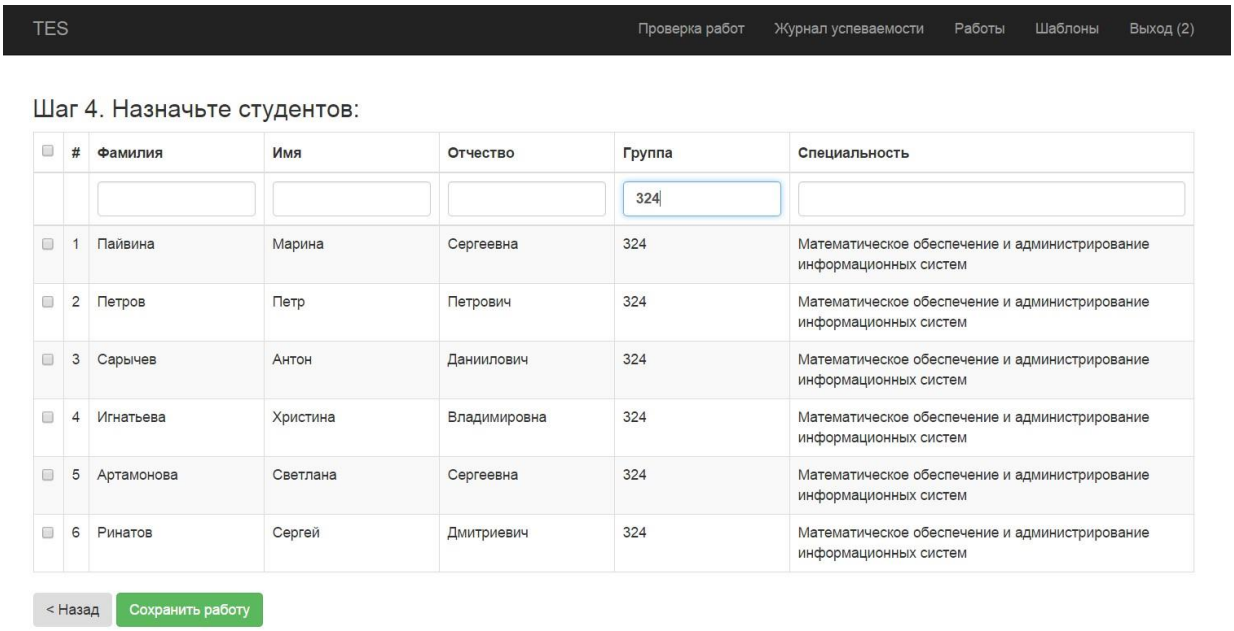

Рисунок 20. Четвертый шаг создания работы

При нажатии на кнопку «Сохранить работу» данная работа сохраняется и будет отображена у студентов в личном кабинете с той даты, которая была указана как начало работы.

На странице шаблонов выводятся все имеющиеся шаблоны (рис. 21).

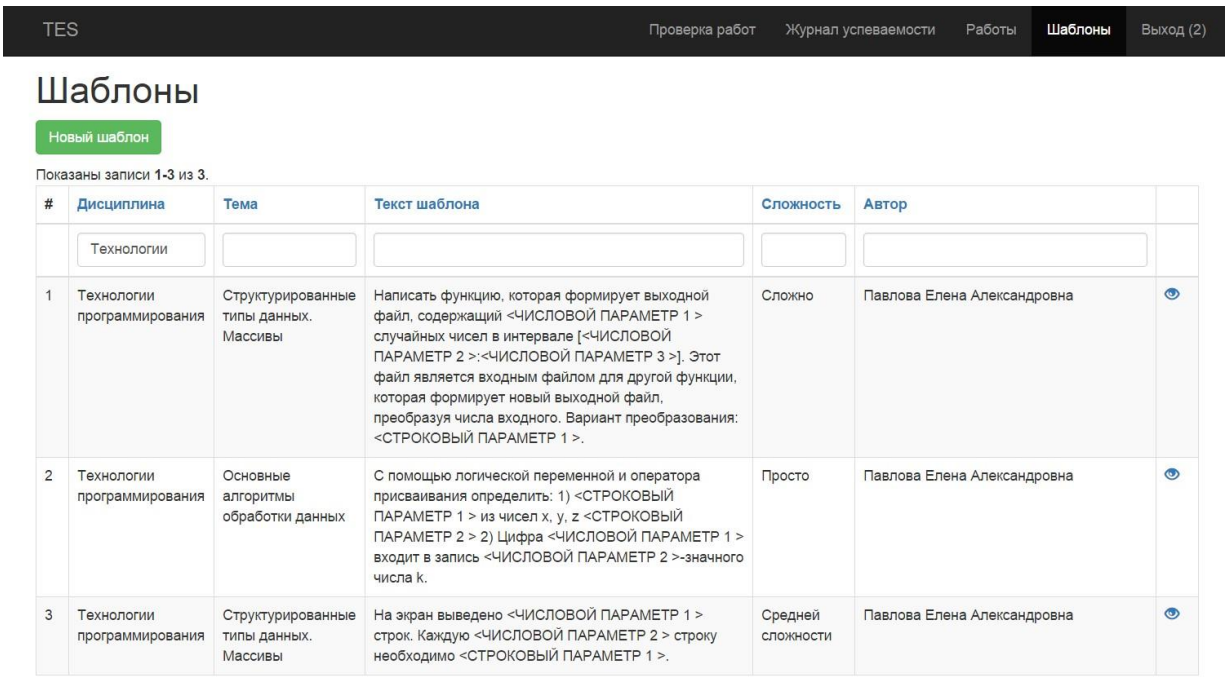

Рисунок 21. Просмотр списка шаблонов

При нажатии на кнопку «Новый шаблон» открывается форма создания нового шаблона: текстовое поле для ввода текста, кнопки для вставки числовых и строковых параметров; также указывается дисциплина, тема и сложность шаблона. После того, как текст шаблона готов, необходимо перейти к редактированию параметров.

Введенные параметры редактируются во всплывающих окнах: для числовых параметров указывается числовой тип данных, а также диапазон значений; для строковых параметров – набор значений.

Заключение

В рамках выполнения магистерской диссертации предложена структура автоматизированной обучающей системы, состоящей из семи модулей, спроектирована и разработана система автоматической генерации вариантов заданий по программированию на основе шаблонов преподавателя, разработан и реализован алгоритм формирования интерактивных тестовых заданий.

Разработанная система позволяет:

-генерировать необходимое количество индивидуальных вариантов заданий по шаблону преподавателя, группировать задания по темам, формам контроля и другим параметрам, важным для преподавателя;

-распределять баллы между заданиями лабораторной работы в соответствии с ограничениями, указанными преподавателем;

-устранять возможные ошибки, связанные с неправильным назначением баллов (например, если сумма баллов, назначенных преподавателем за выполнение всех лабораторных работ, превышает установленный максимум и т.п.);

-выбирать необходимую совокупность заданий из лабораторной работы для набора нужного количества баллов;

-назначить студентов на выполнение конкретных заданий, при этом «сильный» студент не может решать только лёгкие, простые задания, а «слабому» надо дать возможность решить несколько простых задач;

-автоматически генерировать задания для «работы над ошибками»;

-вести учёт сданных лабораторных работ для балльно-рейтинговой системы с формированием необходимых отчётов.

Система представляет собой кроссплатформенное web-приложение с разным уровнем доступа.

При построении системы использовались методы разработки webприложений, объектно-ориентированного анализа и программирования, математической статистики, дискретной математики, реляционной теории баз данных, метод XML-шаблонов.

Внедрение в учебный процесс автоматизированной обучающей системы способствует развитию электронного обучения, так как известно, что адекватная поддержка студентов современными ИТ побуждает интерес студентов к самостоятельной работе, к поиску лучших решений при

выполнении лабораторных работ, повышает степень ответственности за результаты своего обучения.

Результаты, полученные при выполнении магистерской диссертации, были опубликованы в журнале «Современные исследования социальных проблем», № 1(21) и представлены на Международной научной студенческой конференции МНСК-2015 в подсекции «информационные технологии обучения».

Систему автоматической генерации заданий можно использовать в разных дисциплинах гуманитарных и естественно-научных направлений, а также в средней школе для формирования обычных учебных заданий, для генерации однотипных олимпиадных задач с целью отработки школьниками навыков их решения.

Список литературы

1. Активные и интерактивные образовательные технологии (формы проведения занятий) в высшей школе: учебное пособие / Сост. Т. Г. Мухина. – Н. Новгород, ННГАСУ, 2013. – 97 с.

2. Башмаков А.И., Башмаков И.А. Разработка компьютерных учебников и обучающих систем. – М.: Информационно-издательский дом "Филинъ", 2003, - 616 с.

3. Воробьева, М.С., Павлова Е.А. Один из этапов расширения информационного образовательного пространства / М. С. Воробьева, Е.А. Павлова // Современные исследования социальных проблем. – 2015. №1 (21). – С. 181 – 187.

4. Воробьева, М.С., Павлова Е.А. Адаптивная система автоматической генерации заданий по программированию //53-я международная научная студенческая конференция «МНСК-2015», 2015. С. 185 — 186.

5. Гриценко А.А., Анискин С.А., Мальчева Р.В. Автоматизированная система проверки знаний студентов // II международная научная конференция студентов, аспирантов и молодых ученых «Компьютерный мониторинг и информационные технологии», 2006. – С.176-177.

6. Документация PHP // php.net: URL: https://php.net/manual/ru/, 06.02.2015 г.

7. Документация PHP Yii2 framework // GitHub: URL: https://github.com/yiisoft/yii2/tree/master/docs/guide-ru, 13.02.2015 г.

8. Документация PostgreSQL // PostgreSQL.ru.net: URL: http://postgresql.ru.net/userdocs.html, 22.01.2015 г.

9. Захарова И. Г. Возможности информационных технологий в совершенствовании образовательного процесса высшей школы: Монография. - Тюмень: Издательство Тюменского государственного университета, 2002. 176 с.

10. Зорин Ю.А. Интерпретатор языка построения генераторов тестовых заданий на основе деревьев И/ИЛИ // Доклады Томского государственного университета систем управления и радиоэлектроники, 2013, №1. – С.75-79.

11. Зорин Ю.А. Использование алгоритмов комбинаторной генерации при построении генераторов тестовых заданий // Дистанционное и виртуальное обучение, 2013, №6. – С.54-59.

12. Карпова И.П. Исследование и разработка подсистемы контроля знаний в распределенных автоматизированных обучающих системах: дис. канд. тех. наук: 05.13.13. - М., 2002.

13. Княжева В. В. Теория и практика внедрения интерактивных форм обучения на уроках общественных дисциплин в профессиональном образовании // Молодой ученый. — 2015. — №21. — С. 784-788.

14. Кругликов. В. Н. Активное обучение в техническом вузе (Теоретико-методологический аспект): дис. д-ра пед. наук. Санкт-Петерб. гос. университет: СПбГУ, 2000.

15. Кручинин В.В. Алгоритмы и перечислительные свойства деревьев И/ИЛИ // Вестник Томского государственного университета, 2004, №284. – С.181-184.

16. Кручинин В.В. Генераторы в компьютерных учебных программах. – Томск: Изд-во Том. ун-та, 2003. – 200 с.

17. Кручинин В.В., Морозова Ю.В. Модели и алгоритмы генерации задач в компьютерном тестировании // Известия Томского политехнического университета, 2004, № 5. – С.127-131.

18. Лаптев В.В., Толасова В.В. Генерация вариантов заданий для лабораторных работ по программированию // Вестник Архангельского государственного университета, 2010, №1. – С.127-131.

19. Левинская М. А. Автоматизированная генерация заданий по математике для контроля знаний учащихся // Образовательные технологии и общество, 2002, №5. – с.214-221.

20. Маврин П.Ю., Григорьева М.В. Автоматизация составления вариантов заданий для проверочных работ // Информационные технологии для новой школы, 2010. – С.206-209.

21. Посов И. А. Web-сайт для создания и обмена генерируемыми задачами по математике // Образовательные технологии и общество, 2014, №3. – С. 360-373.

22. Посов И. А. Обзор генераторов и методов генерации учебных заданий // Образовательные технологии и общество, 2014, №17. – С. 593605.

23. Профессиональные стандарты. Режим доступа: http://www.apkit.ru/committees/education/meetings/standarts.php (дата обращения 29.06.2015).

24. Пруцков А. В. Статический и динамический подходы к проектированию подсистем проверки знаний автоматизированных обучающих систем // Информационные ресурсы России, 2006, №1. – С.27-29.

ПРИЛОЖЕНИЕ 1

Задания, сгенерированные по методу шаблонов

Шаблон 1

<body – начало тела**>**

Дан одномерный массив из <p1> элементов. Элементы массива могут принимать целые значения от -10000 до 10000 включительно. Напишите программу, позволяющую найти и вывести количество пар элементов массива, в которых <p2> число делится на 3. В данной задаче под парой подразумевается два подряд идущих элемента массива.

</body – конец тела**>**

<param – параметры**>**

<p1 – начало описания параметра**>**

<**type = number, integer** – тип параметра>

<condition – условие**>**

<v=random>

<v>10>

</condition – конец условия**>**

</p1 – конец описания параметра**>**

<p2 – начало описания параметра**>**

<type=text – тип параметра**>**

<select – начало выбора**>**

<v= “ каждое ”**>**

<v= “ только одно ”**>**

<v= “ хотя бы одно ”**>**

</select – конец выбора>

</p2 – конец описания параметра>

</param – конец описания параметров>

Полученные варианты заданий

Вариант 1.

Дан одномерный массив из 25 элементов. Элементы массива могут принимать целые значения от -10000 до 10000 включительно. Напишите программу, позволяющую найти и вывести количество пар элементов массива, в которых хотя бы одно число делится на 3. В данной задаче под парой подразумевается два подряд идущих элемента массива.

Вариант 2.

Дан одномерный массив из 35 элементов. Элементы массива могут принимать целые значения от -10000 до 10000 включительно. Напишите программу, позволяющую найти и вывести количество пар элементов массива, в которых только одно число делится на 3. В данной задаче под парой подразумевается два подряд идущих элемента массива.

Вариант 3.

Дан одномерный массив из 28 элементов. Элементы массива могут принимать целые значения от -10000 до 10000 включительно. Напишите программу, позволяющую найти и вывести количество пар элементов массива, в которых каждое число делится на 3. В данной задаче под парой подразумевается два подряд идущих элемента массива.

Шаблон 2

<body – начало тела>

Вычислить при a = <p1>, b = <p1>

1. *a <p2> b*
2. *not a <p2> b*
3. *<p3> a <p2> b <p2> <p3> <p4>*

</body – конец тела>

<param – параметры>

<p1 – начало описания параметра>

<type=text – тип параметра>

<select – начало выбора>

<v= “ true ”>

<v= “ false ”>

</select – конец выбора>

</p1 – конец описания параметра>

<p2 – начало описания параметра>

<type=text – тип параметра>

<select – начало выбора>

<v= “ and ”>

<v= “ or ”>

<v= “ xor ”>

</select – конец выбора>

</p2 – конец описания параметра>

<p3 – начало описания параметра>

<type=text – тип параметра>

<select – начало выбора>

<v= “ ”>

<v= “ not ”>

</select – конец выбора>

</p3 – конец описания параметра>

<p4 – начало описания параметра>

<type=text – тип параметра>

<select – начало выбора>

<v= “ a ”>

<v= “ b ”>

</select – конец выбора>

</p4 – конец описания параметра>

</param – конец описания параметров>

Полученные варианты заданий

Вариант 1.

Вычислить при a = false, b = true:

1. a and b
2. not a and b
3. a xor b and not b

Вариант 2.

Вычислить при a = false, b = false:

1. a or b
2. not a or b
3. a or b and not a

ПРИЛОЖЕНИЕ 2

Задания, сгенерированные по методу деревьев И/ИЛИ

Шаблон 1

Main("Дан одномерный массив из ",[20..40]," элементов. Элементы массива могут принимать целые значения от -10000 до 10000 включительно. Напишите программу, позволяющую найти и вывести количество пар элементов массива, в которых ", A{"каждое","только одно","хотя бы

одно"}," число делится на ",[3..7],". В данной задаче под парой подразумевается два подряд идущих элемента массива.") Вариант 1.

Дан одномерный массив из 24 элементов. Элементы массива могут принимать целые значения от -10000 до 10000 включительно. Напишите программу, позволяющую найти и вывести количество пар элементов массива, в которых хотя бы одно число делится на 4. В данной задаче под парой подразумевается два подряд идущих элемента массива.

Вариант 2.

Дан одномерный массив из 20 элементов. Элементы массива могут принимать целые значения от -10000 до 10000 включительно. Напишите программу, позволяющую найти и вывести количество пар элементов массива, в которых каждое число делится на 3. В данной задаче под парой подразумевается два подряд идущих элемента массива.

Вариант 3.

Дан одномерный массив из 23 элементов. Элементы массива могут принимать целые значения от -10000 до 10000 включительно. Напишите программу, позволяющую найти и вывести количество пар элементов массива, в которых только одно число делится на 6. В данной задаче под парой подразумевается два подряд идущих элемента массива.

Шаблон 2.

Main("С помощью логической переменной и оператора присваивания определить:

1) x принадлежит [",[-2,0..3],";",[3..5],"];
2) "A{"каждое","только одно"}, " из чисел x, y, z", B{" положительно;"," отрицательно;"},"
3) Цифра ",[0..10]," входит в запись ",[2..6],"-значного числа k.")

Вариант 1.

С помощью логической переменной и оператора присваивания определить:

1) x принадлежит [-1,7; 4];

2) каждое из чисел x, y, z отрицательно; 3)

Цифра 5 входит в запись 3-значного числа k.

Вариант 2.

С помощью логической переменной и оператора присваивания определить:

1) x принадлежит [1,9; 3];

2) только одно из чисел x, y, z отрицательно;

3) Цифра 8 входит в запись 4-значного числа k.

Шаблон 3.

Main("Вычислить при a = ",A{"true","false"}", b = ",B{"true","false"},":

1. a ",C{"and","or","xor"}," b
2. not a ", E{"and","or","xor"}," b
3. ",F{"not ",""},"a ", G{"and","or","xor"},"

b ",I{"and","or","xor"},F{" not",""},K{" a"," b"}".")

Вариант 1.

Вычислить при a = true, b = true:

1. a or b
2. not a and b
3. a xor b and not a

Вариант 2.

Вычислить при a = false, b = true:

1. a xor b
2. not a or b

3. not a and b or not a

4. not a or b and b