Федеральное агентство по образованию РФ
Дальневосточный федеральный университет
Институт математики и компьютерных наук
Кафедра информатики

Доработка системы генерации тестовых заданий

Курсовая работа
студента группы с8403а
Кевролетина В.В.
Руководитель:
ст. преподаватель кафедры информатики
Кленин А.С.

Содержание

Аннотация

Данная работа посвящена улучшению существующей системы генерации тестовых заданий ЕГЭ по информатике.

ЕГЭ — явление, с которым 5 лет назад столкнулось Российское общество. Среди последствий введения ЕГЭ выделятся появившаяся необходимость массово подготавливать выпускников средних учебных заведений к сдаче аттестационного тестирования. Для подготовки требуется большое число демонстрационных вариантов заданий.

Дорабатываемая мной система призвана удовлетворить потребность учебных заведений в заданиях по информатике.

1. Введение

1.1. Глоссарий

Вариант заданий - это набор сходных по структуре заданий, направленных на проверку одного и того же знания или навыка, отличающихся друг от друга формулировкой вопроса, или вариантами ответов.

Варианты ответа - это множество выражений, содержащее как правильный, так и неправильные ответы на вопрос задания.

Генерация задания - создание тестового задания на основе шаблона при помощи генератора случайных чисел.

Генерация результата - один из этапов генерации задания, в котором происходит построение множества ответов.

Генератор — процедура на ЯП, генерирующая схожие по структуре задания на основе случайных алгоритмов.

Единый государственный экзамен – одна из форм государственной итоговой аттестации выпускников 11-х классов и вступительных испытаний в высшие и средние специальные учебные заведения[1].

Задание типа А - задания этого типа содержат тестовые задания, в каждом из которых необходимо выбрать верный вариант ответа из нескольких предложенных[1].

Задание типа В - Задания этого типа содержат тестовые задания, в каждом из которых необходимо дать краткий ответ, состоящий из одного или нескольких слов, букв или чисел.

Задание типа С - Задания этого типа содержат тестовые задания, в каждом из которых необходимо дать ответ в виде одного или нескольких предложений или формул. Проверка правильности ответов на эти задания производится специально подготовленными независимыми экспертами-предметниками с использованием четко определенных критериев оценивания[1].

Дистрактор - вариант ответа в заданиях с выбором, не являющийся правильным решением, но внешне близкий к нему и полученный в результате ошибки в правильном ходе рассуждений [2].

Задание - это базовая единица проверки знаний. Под заданием в данной работе будем понимать тестовое задание, состоящее из условия задания, вопроса и вариантов ответа.

Результат тестирования - последовательность правильных и неправильных ответов, позволяющая сделать вывод о степени владения материалом теста, выявить наиболее трудные задания и наиболее частые ошибки учащихся.

Тест - стандартизированные задания, результат выполнения которых позволяет

измерить знания, умения и навыки испытуемого[3].

Тестирование - это процесс выполнения учащимся заданий теста.

Ход решения - это последовательность действий, приводящих к получению ответа.

1.2. Описание предметной области

Тестирование как метод проверки знаний стало широко использоваться с начала XX века [4]. Предпосылкой к этому явилось усовершенствование технологий производства, потребовавшее большого количества квалифицированных кадров, в силу чего возросли требования к образованию. Образовательная система должна была позволять обучать много людей и при этом обучать качественно.

Контроль знаний учащихся, служащий для осуществления обратной связи между преподавателем и студентами, является трудоемким процессом, который включает в себя составление заданий по учебному материалу и проверку результатов их выполнения, поэтому нуждается в оптимизации. Примером такой оптимизации является применение тестов. Обоснование этого подхода приведено в работах доктора педагогических наук, профессора В. И. Аванесова [5]. В частности, он пишет: «Учебные вопросы многословны и порождают ответы, полные и неполные, правильные и неправильные, разные по форме, содержанию и по структуре, вследствие чего оценка таких ответов требует обязательного участия преподавателя и сопровождается некоторой долей субъективизма. Вопросы и ответы на них иногда бывают столь неопределенными и многословными, что для выявления их истинности требуются большие затраты интеллектуальной энергии, в то время как технологичная методика тестирования предполагает четкую и быструю дифференцируемость ответов, что в свою очередь облегчает их проверку». Очевидные достоинства данной технологии проверки знаний: простота проведения тестирования, невысокая сложность проверки результатов и их статистической обработки привели к повсеместному внедрению технологии проведения тестирования среди учебных заведений и центров проверки знаний во многих странах мира.

Следующий шаг в оптимизации проверки знаний был сделан с введением компьютерного тестирования, которое позволило полностью автоматизировать проверку результатов. Широкое распространение персональных компьютеров в сочетании с простотой и эффективностью проведения компьютерного тестирования привело к повсеместному внедрению технологий автоматического тестирования в учебный процесс образовательных учреждений по всему миру.

В нашей стране тестирование играет особенно важную роль: с 2009г. в России введён единый государственный экзамен[6]. Теперь каждый выпускник среднего учебного заведения для того, чтобы подтвердить свои знания, полученные в ходе обучения и поступить в высшее учебное заведение, должен сдавать централизованное тестирование. Этот факт порождает

потребность готовить большое число выпускников к сдаче тестирования.

ЕГЭ проводится с использованием заданий стандартизированной формы и единой методики оценивания выполненных работ[7]. Другими словами, одной из главных особенностей ЕГЭ является «шаблонность» заданий: среди множества различных вариантов можно выделить группу задний, схожих по структуре и различающихся лишь некоторыми параметрами (числовыми значениями, именами людей, различной формулировкой одних и тех же предложений и т. д.). При этом для решения схожих задач требуются одинаковые навыки, так как для построения решения применяются одинаковые рассуждения. Поэтому учащиеся, знакомые со способами и приёмами решения основных типов заданий, получают дополнительное преимущество на экзамене.

Распространённая методика подготовки к тестированию состоит в систематическом прорешивании учащимся типовых заданий. В сложившейся обстановке школы и вузы уделяющие внимание подготовке выпускников и будущих абитуриентов к ЕГЭ, испытывают необходимость в большом количестве демонстрационных вариантов тестов и системе автоматизированного проведения тестирования.

Реализация и поддержка системы генерации тестовых заданий ЕГЭ несёт с собой ряд сложностей, отсутствующих в традиционной технологией «бумажного тестирования»:

• процесс подготовки тестов требует специалистов высокой квалификации

• необходимо соответствующее аппаратное и программное обеспечение

• необходим квалифицированный специалист (администратор), осуществляющий техническую поддержку работы системы

С другой стороны, существуют многочисленные преимущества, которые и объясняют целесообразность создания использования подобных систем

• уменьшается занятость преподавателя в процессе составления тестовых заданий;

• исключается вероятность совершения ошибки составителем тестов

• подобную систему легко интегрировать с системами автоматического (в том числе удалённого) тестирования, применение которых значительно сокращает временные затраты преподавателя на обработку результатов тестирования

• при некоторых дополнительных условиях возможно неоднократное самостоятельное получение тестовых заданий учащимся, что не требует никаких усилий со стороны преподавателя

Весь процесс тестирования можно условно разделить на следующие этапы:

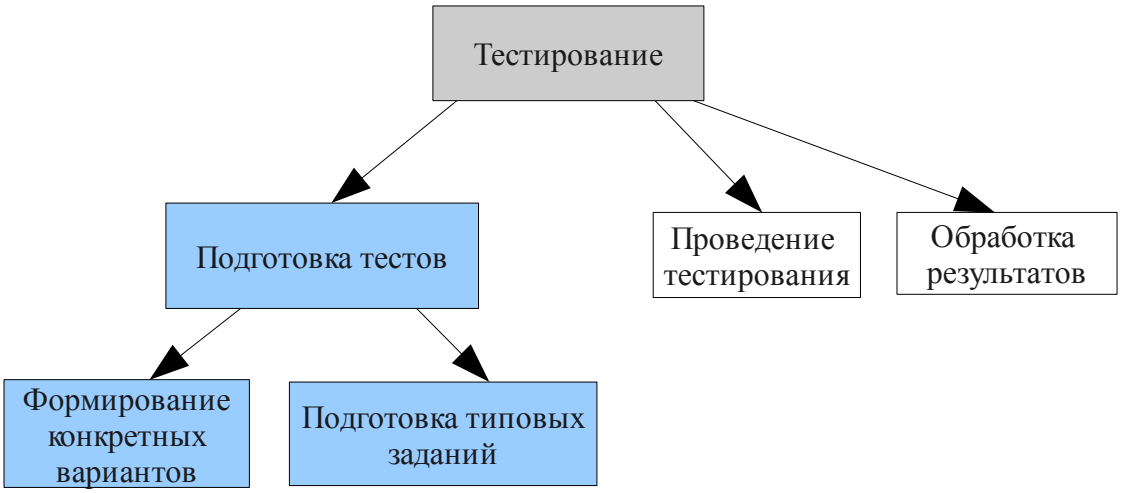

Диаграмма 1: Этапы проведения тестирования

Ниже приведена диаграмма, дающая сравнительную характеристику технологии компьютерного и бумажного тестирования на выделенных этапах подготовки и проведения тестирования:

Диаграмма 2: Сравнение компьютерного и бумажного тестирования

Институт математики и компьютерных наук[8] Дальневосточного федерального университета[9] ведёт подготовку специалистов в области математики и информатики. Залогом высокого уровня знаний выпускников является усердная работа сотрудников и студентов института на всех этапах обучения, которое начинается у многих ребят еще до поступления в вуз — в мастер классах по математике и информатике при институте. Работа с будущими абитуриентами является важной частью работы института, в которой участвуют как преподаватели, так и студенты.

Кафедра информатики ИМКН ведёт обучение по 2м специальностям: системное

программирование; математическое и программное обеспечение вычислительных систем. Очевидно, что студенты кафедры, овладевающие перечисленными специальностями, должны иметь глубокие знания в области информатики. Кроме того, для поступления на эти специальности абитуриенты сдают информатику в качестве вступительного экзамена. Для повышения уровня знаний будущих абитуриентов кафедра в рамках работы института со школьниками ведёт обучение ребят в области информатики и готовит их к сдаче экзаменов по этому предмету.

Для проведения более эффективных тренировок по решению типовых заданий ЕГЭ по информатике кафедра использует систему автоматической генерации тестовых заданий ЕГЭ.

На данный момент система содержит сравнительно небольшое число типовых задач. Кроме того каждый год выходят новые демонстрационные версии заданий. Так что перед кафедрой стоит задача доработки системы и дальнейшего поддержания её в актуальном состоянии.

Описание совместной деятельности

Отправной точкой развития проекта является дипломная работа выпускницы 2010 г. нашей кафедры Зенкиной А.О. «Библиотека алгоритмов для генерации задач»[10]. Работа написана под руководством преподавателя кафедры Кленина А.С., взявшего впоследствии на себя дальнейшее развитие системы.

Исходный код системы разрабатывается с использованием системы контроля версий git[11] и хостинга hithub.com[12], он свободно доступен по адресу [13] и распространяется по лицензии GPL[14] версии 2 или более поздней.

Предложенные модификации добавляются в проект только автором системы. Как следствие, абсолютно все изменения принимаются только после полного удовлетворения всем требованиям автора системы.

Моя основная задача состоит в разработке и добавлении в проект генераторов тестов и доработке уже существующих генераторов. В случае возникновения необходимости возможна доработка существующей библиотеки генерации путём реализации нового функционала.

1.3. Неформальная постановка задачи

Следует доработать существующую систему автоматической генерации тестовых заданий ЕГЭ по информатике. В ходе разработки следует учитывать специфику проблемной области, а именно:

• невысокие требования к производительности (задания можно генерировать заранее и сохранить для дальнейшего использования, важнее скорость разработки и гарантия корректности получаемых результатов)

• эксплуатация системы ведётся человеком, хорошо владеющим программированием

• в разработке системы задействовано несколько человек (что вынуждает хорошо комментировать исходный код и аккуратно писать документацию)

• важно гарантировать корректность генерируемых заданий (ошибка в генераторе может привести к развитию у учеников навыка неверно решать задачу)

• генерируемые задания должны с одной стороны отличаться от реальных заданий как можно меньше, а с другой охватывать как можно большее количество разных реальных заданий

Учитывая вышеизложенные требования, в рамках данной работы необходимо:

• доработать существующие генераторы, «разбавив» текст условия задачи разнообразными фразами, именами, названиями и т.д.

• добавить генераторы новых заданий, либо расширить старые генераторы для создания новых типов задач

• при необходимости улучшить существующую библиотеку добавлением новых модулей и функциональности в уже существующие модули

При этом следует аккуратно подходить к реализации, сохраняя принятый в системе формат кодирования.

1.4. Обзор существующих методов решения

Аналогичные (конкурирующие) решения

Изучив доступные источники информации, я не нашёл ни одного готового решения для автоматизированной генерации заданий ЕГЭ по информатике. На данный момент демонстрационные варианты заданий можно получить из печатных источников и в сети Интернет. Причем в сети найденные мной на различных сайтах[1] варианты заданий лишь дублировали материалы, доступные на официальном портале государственного экзамена. Ниже приведена сравнительная таблица источников демонстрационных заданий ЕГЭ по информатике.

[1] К примеру:

- http://www.rosbalt.ru/eg/
- http://egerf.ru/index.php?s=8&id=96&razd=11
- http://college.ru/how-buy/
- http://www1.ege.edu.ru/online-testing/inf

	Официальный портал ЕГЭ	Печатные источники	Неофициальные интернет-порталы подготовки к егэ	Наша система
Количество различных вариантов	мало	средне	мало	много
Автоматическое тестирование	1 вариант заданий	нет	максимум 1 вариант на сайт	простая интеграция с системами тестирования
Официальный источник информации	да	да	нет	нет
Задания типа А и В	да	да	не везде	да
Объяснение хода решения	только задания типа С	только задания типа С	обсуждения на форумах	нет, возможно в перспективе
Наличие ответов к заданиям	да	да	не везде	да

Таблица 1: Сравнение источников заданий ЕГЭ по информатике

После обзора источников заданий ЕГЭ по информатике становится очевидна необходимость системы генерации заданий.

Тестирование стало широко применяться задолго до появления ЕГЭ, поэтому сегодня существует множество различных систем автоматизированной генерации тестовых заданий. Во-первых, это редакторы тестов[15]. Они позволяют управлять внешним видом теста: формой ответа, наличием подсказок, добавлением графического материала, установкой сложности задания, однако содержание теста необходимо вводить вручную. Этим достигается гибкость в описании заданий, поскольку пользователь ничем не ограничен и может свободно набирать то задание, которое ему нужно. Однако это пример неавтоматизированного составления задач, поскольку он не освобождает пользователя от необходимости составлять множество однотипных заданий. Во-вторых, это редакторы курсов, предоставляющие возможность генерировать тесты на основе ранее введенного материала курса [16]. Как правило, они используют базу данных с понятиями и

определениями из курса и базу возможных вопросов и ответов. При этом вопросы могут комбинироваться при помощи логических связок: И, ИЛИ, НЕ, благодаря чему увеличивается разнообразие заданий. Они рассчитаны больше на преподавателей, не связанных с программированием, поэтому важной чертой для них является наличие простого и удобного графического интерфейса, что накладывает ограничения на содержание задания. Так, например, генерация выражений и формул в них уже не так просто осуществима. В-третьих, существуют различные генераторы выражений, создающие формулы и выражения на основе некоторого описания. Примером может служить библиотека MatLab для генерации случайных выражений [17] или Pinery, программа для генерации выражений на основе регулярных выражений Perl [18]. Однако такие генераторы могут быть использованы лишь как вспомогательные средства при генерации тестов, поскольку генерируют не задание в целом, а лишь некоторую его часть. При этом, чтобы их применять, пользователь должен быть знаком с программированием. В-четвертых, есть генераторы индивидуальных домашних заданий (ИДЗ) и тестов: например, генератор ИДЗ по мат. анализу [19] и генератор тестов по информатике [20]. Эти генераторы могут отличаться друг от друга функциональностью, алгоритмами и реализацией. В частности, приведенный выше генератор ИДЗ по мат. анализу позволяет генерировать условия заданий, решение и вычислять ответ, но в нем не предусмотрена генерация нескольких вариантов ответов. Генератор тестов по информатике позволяет генерировать условия заданий и 7 варианты ответов, но в нем не предусмотрено построение хода решения. Эти программы объединяет механизм составления шаблона. Как правило, шаблон содержит некоторое число параметризованных величин, для которых указана область их изменения и, возможно, алгоритм генерации. В процессе генерации величины заменяются константами из соответствующих областей и в зависимости от этих конкретных значений вводятся грамматические изменения в формулировку задачи. Кроме того, шаблон содержит вызов алгоритма решения задачи, по которому после генерации параметров вычисляется правильный ответ. При этом невозможно однозначно оценить простоту и гибкость таких систем. Если система ориентирована на какую-то определенную область и определенный вид теста, как, например, генератор ИДЗ по мат. анализу, то шаблоны тестов могут быть жестко зашиты внутри нее без возможности их изменения, что удобно с точки зрения использования: нужно лишь выбрать нужный шаблон и сгенерировать тест. Если область применения системы расширяется, то возникает необходимость не только в написании новых шаблонов, но и, как правило, в доработке самой системы, добавлении в нее новых функций и алгоритмов. В этом случае либо пользователь должен обладать навыками программиста и самостоятельно вносить изменения в систему, либо время от времени обращаться к программисту, что не очень удобно с точки зрения использования системы. Поскольку перед автором работы поставлена задача создания гибкой

системы генерации тестов, то создаваемый им проект также не освобожден от этого недостатка. В ходе обзора автору не удалось выяснить, поддерживают ли генераторы ИДЗ и тестов возможность генерации дистракторов вариантов ответов, которые являются результатом определенной ошибки в рассуждениях, основанной на недостаточном понимании материала. В свою очередь, такая возможность является одним из приоритетов в данной работе. Кроме того, особенностью данной работы является использование 8 для составления шаблонов языка программирования, что позволяет комбинировать уже существующие функции для описания новой задачи, а не реализовывать всякий раз программно новый алгоритм. Хотя перед автором не стоит задачи построения хода решения для заданий теста, использование языка программирования имеет потенциал к выполнению также и этой задачи. Следует также оговорить еще одну особенность создаваемой системы это достаточно большой объем текста, в котором выражается описание задания, что является оборотной стороной гибкости подхода. Однако этот недостаток компенсируется по мере увеличения числа сгенерированных по данному описанию тестов. Ниже приведена сводная таблица характеристик рассмотренных систем.

	Редактор тестов	Редактор курсов	Редактор выражений	Генератор ИДЗ	Наша система
Генерация условий	есть	есть	есть	есть	есть
Генерация ответов	нет	есть	нет	возможна	есть
Генерация дистракторов	нет	возможна	нет	возможна	есть
Построение хода решения	нет	нет	нет	есть	Нет, в перспективе возможно
Требования к пользователю	низкие	низкие	высокие	разные	высокие
Генерация заданий ЕГЭ по информатике	нет	нет	нет	нет	да

Таблица 2: Характеристики систем создания тестов

Описание предшествующих работ

В 2010г на кафедре информатики ИМКН Зенкиной А.О. в ходе создания дипломной работы на тему «Библиотека алгоритмов для генерации задач»[10] было проведено исследование в области генерации тестовых заданий. В данной работе были изучены различные способы создания систем автоматической генерации тестовых заданий. Из всех подходов был выбран, по мнению автора диплома, наиболее подходящий: написание генераторов на языке программирование perl[21] с использованием библиотеки Template::Tolkit[22] (от использования которой в дальнейшем Автор системы отказался) и собственной библиотеки для генерации тестовых заданий. Результатом работы стало создание системы, осуществляющей генерацию заданий на основе шаблонов с использованием библиотеки классов и функций, а также возможностей языка Template::Toolkit.

Стоит отметить, что перед Зенкикой ставилась довольно общая задача: создание универсальной системы для генерации задач. Тем не менее для тестирования созданной системы в ней была реализована генерация нескольких заданий ЕГЭ по информатики. Это позволило автору работы сделать подкреплённый фактами вывод о возможности и целесообразности использования её программы для генерации заданий ЕГЭ по информатике. Кроме того были выявлены некоторые недостатки системы и были предложены способы её дальнейшего развития.

Вывод

Альтернатив системе автоматической генерации тестовых заданий А.С. Кленина не существует, либо информация о их создании не получала общественной огласки. Данная система нуждается в постоянной поддержке: необходимо реализовывать новые генераторы соответствие заданиям актуальных демонстрационных версий ЕГЭ.

В силу отсутствия альтернатив и востребованности системы очевидна необходимость её поддержки.

1.6. План работ

1. Изучить демонстрационные варианты тестовых заданий ЕГЭ по информатике. Выбрать задания, которые необходимо добавить.
2. Реализовать выбранные задания.

2. Требования к окружению

2.1. Требования к аппаратному обеспечению

Единственное требование к аппаратному обеспечению: возможность запустить

интерпретатор ЯП perl. Комплектация рабочей зависит от способа применения системы. Так, для генерации заданий с последующим их просмотром потребует наличие клавиатуры и монитора. Вариант получения пользователем заданий по сети посредством протокола ssh потребует наличие сетевой карты. Вариант, при котором задания генерируются и сохраняются в долговременной памяти для дальнейшего использования, вообще не требует наличия периферийных устройств.

2.2. Требования к программному обеспечению

Единственное требование к программному обеспечению: возможность запуска интерпретатора ЯП perl версии 5.10 и выше.

Официально поддерживаемые сообществом ЯП perl платформы [23]:

- Linux (x86, ARM, IA64)
- HP-UX
- AIX
- Win32
    - Windows 2000
    - Windows XP
    - Windows Server 2003
    - Windows Vista
    - Windows Server 2008
    - Windows 7
- Cygwin
- Solaris (x86, SPARC)
- OpenVMS
    - Alpha (7.2 and later)
    - I64 (8.2 and later)
- Symbian
- NetBSD
- FreeBSD
- Haiku
- Irix (6.5. What else?)
- OpenBSD
- Dragonfly BSD

- MirOS BSD
- Symbian (Series 60 v3, 3.2 and 5 - what else?)
- Stratus VOS
- AIX

Требования к пользователям

Пользователи системы делятся на 2 категории: составители генераторов и потребители сгенерированных заданий.

Для запуска генерации необходимо умение использования командной строки.

Для составления генераторов необходимо умение программирования с использованием ЯП perl и знание библиотеки генерации тестовых заданий данной системы.

3. Архитектура системы (Общие требования)

Программу можно разделить на 3 подсистемы:

1. подсистема запуска генераторов и вывода результатов
2. генераторы заданий
3. библиотека генерации тестовых заданий

Взаимодействие подсистем между собой, с пользовательским интерфейсом в виде командной строки, их зависимость от стандартной библиотеки языка perl отображена на следующей схеме:

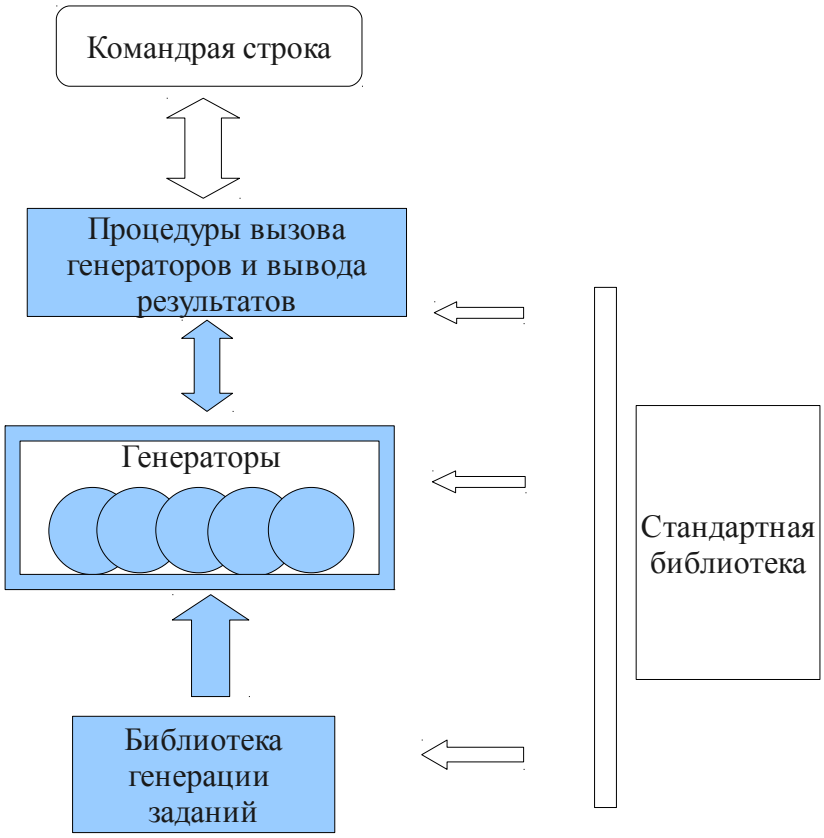

Диаграмма 3: Основные подсистемы программы

4. Спецификация данных

В соответствии с разделением заданий ЕГЭ на задания типа А, в которых необходимо выбирать правильный ответ из четырёх предложенных вариантов, и задания типа В , не предоставляющие никаких вариантов ответов, генераторы делятся на 2 вида:

1. генераторы заданий с выбором единственного варианта ответа
2. генераторы заданий, требующих от пользователя, непосредственного ввода ответа

Для генераторов первого типа требуется:

- создание текста условия задания
- создание нескольких вариантов ответов
- указания одного правильного варианта ответа

Для генераторов второго типа необходимо:

- создание текста условия задания
- создание единственного варианта решения

5. Функциональные требования

Система должна позволять пользователям:

- генерировать тестовые задания ЕГЭ

- добавлять генераторы новых типов заданий

При этом отдельные подсистемы должны выполнять следующие функции:

1. Подсистема запуска генераторов и вывода результатов
   - запускать множество генераторов
   - преобразования полученных от генераторов данных в формат html или json
2. Генераторы заданий
   - создавать варианты тестовых заданий ЕГЭ по информатике на основе случайных алгоритмов
3. Библиотека генерации тестовых заданий
   - предоставлять пользователю набор функций облегчающих процесс создания генераторов

Библиотека подпрограмм (классов)

Библиотека генерации заданий состоит из нескольких модулей, выполняющих определённые функции:

- Bin — вывод в html чисел в двоичной, восьмеричной или шестнадцатеричной системах исчисления
- Bits — работа с бинарными векторами
- GenBase — описание классов - генераторов
- Graph — работа с графами, вывод структур, описывающих графы в html и визуализация графов в svg
- Html — генерация html
- LagnTable — вывод заданий на программирование
- Logic — работа с логическими выражениями
- NotationBase — преобразование чисел из одной системы исчисления в другие
- NumText — вывод чисел и простых выражений, сосящих из числа и зависимого слова в виде текста
- Prog — генерация заданий на программирование
- Random — генерация случайных чисел, случайных выборок из последовательности, случайных букв и строк.
- Svg — генерация векторных изображений.

Кроме того библиотека содержит словари часто используемых в тексте заданий слов:

- Russian/Animals — названия животных
- Russian/FamilyNames — русские фамилии
- Russian/Jobs — профессии

- Russian/Names — русские имена
- Russian/SimpleNames — широко распространённые русские имена
- Russian/Subjects — названия школьных дисциплин

6. Требования к интерфейсу

Интерфейс командной строки

7. Прочие требования

Отсутствуют.

8. Проект

8.1. Средства реализации

Система целиком реализована на языке программирования Perl с использованием стандартной библиотеки этого языка.

Для работы с исходным кодом системы пригоден любой текстовый редактор. Я использую текстовый редактор emacs[24]. Среди преимуществ этого редактора стоит выделить режим отладки программ на ЯП perl.

Для просмотра сгенерированных тестов можно использовать любой современный веб-браузер. Я использую firefox[25].

8.2. Модули и алгоритмы

8.2.1. A01 «Количество единиц в двоичной записи числа»

Задание.

Сколько единиц в двоичной записи числа 1025?

1. 1
2. 2
3. 10
4. 11

Анализ решения.

Для того, чтобы быстро решить задание нужно заметить, что 1025 = 1024 + 1 = 2^10 + 2^0. Двоичная запись числа 2^n (n - натуральное число) содержит одну единицу. Тогда 1025 содержит 2 единицы.

Описание изменений в генераторе.

Реализованный в системе генератор A3 ones генерирует идентичные задания.

Единственное отличие - ограничения в генераторе. Генератор берёт большое число 2^n и прибавляет(отнимает) к нему 1. Ограничение генератора n <= 9 заменено на n <= 10.

8.2.2. A02 «Кратчайшего пути между населёнными пунктами »

Условие.

Между населёнными пунктами A, B, C, D, E, F построены дороги, протяжённость которых приведена в таблице. (Отсутствие числа в таблице означает, что прямой дороги между пунктами нет.)

	A	B	C	D	E	F
A	~	2	4			
B	2	~	1		7	
C	4	1	~	3	4	
D			3	~	3	
E		7	4	3	~	2
F					2	~

1. 9
2. 10
3. 11
4. 12

Анализ задания.

Дан ненаправленный взвешенный граф с циклами. Необходимо найти кратчайший маршрут между 2мя вершинами. На сложность задания влияет количество вершин, длина кратчайшего маршрута.

Алгоритм генерации.

Задание похоже на создаваемые генератором A6 bus_station варианты тестов. Используется тот же подход к генерации задания. Код переписан заново в лучшем виде (подразумевается последующее вынесение в отдельные процедуры общих частей с существующем генератором). Краткое описание алгоритма генерации:

1. Генерируется матрица смежности для неориентированного графа без петель (возможно) с циклами.
2. Используется алгоритм Флойда-Уоршолла для поиска расстояний между вершинами. Причем во время работы алгоритма при улучшении существующих значений в таблице маршрутов запоминаются предыдущие значения(будут использованы в качестве деструкторов).
3. Выбираются 2 вершины, между которыми существует маршрут с наибольшим числом деструкторов. В качестве недостающих вариантов ответов берутся длины маршрутов из других вершин в конечную и случайные числа.

8.2.3. A05 «Автомат, строящий числа по заданным правилам»

Условие.

Автомат получает на вход два трехзначных числа. По этим числам строится новое число по следующим правилам.

1. Вычисляются три числа – сумма старших разрядов заданных трехзначных чисел, сумма средних разрядов этих чисел, сумма младших разрядов.
2. Полученные три числа записываются друг за другом в порядке убывания (без разделителей).

Пример. Исходные трехзначные числа: 835, 196. Поразрядные суммы: 9, 12, 11. Результат: 12119

Определите, какое из следующих чисел может быть результатом работы автомата.

1. 151303
2. 161410
3. 191615
4. 121613

Анализ решения.

Необходимо отбросить неверные варианты:

- 151303 - 03 (лидирующий ноль)
- 191615 - 19 (максимальная сумма 2х 10тичных цифр 9+9==18)
- 121613 - Числа не в порядке убывания.

Оставшееся число 161410.

Алгоритм генерации:

Сгенерировать 3 числа, содержащих одну из перечисленных выше проблем и одно подходящее число.

8.2.4. A7 «Копирование ячейки с формулой в электронной таблице»

Условие.

В ячейке B4 электронной таблицы записана формула = $C3 * 2. Какой вид приобретет формула, после того как ячейку B4 скопируют в ячейку B6?

Примечание: знак $ используется для обозначения абсолютной адресации.

1. = $C5 * 4
2. = $C5 * 2
3. = $C3 * 4
4. = $C1 * 2

Анализ решения.

Номер ячейки состоит из двух частей: латинские буквы и цифры. Буквами обозначены

столбцы таблицы, цифрами строки. На пересечении строки и столбца находится ячейка с соответствующим номером. Если в ячейке записана формула, содержащая ссылку на другую ячейку, то при копировании содержимого в другую ячейку ссылка изменяется по следующему правилу:

1. координата ссылки, помеченная знаком $ остаётся неизменной
2. если знак $ отсутствует, то координата изменяется на столькоже, насколько изменилась соответствующая координата ячейки, в которой записана формула.

Неправильные ответы в приведённом задании содержат следующие ошибки:

1. сдвиг в обратную сторону
2. сдвиг по другой координате
3. сдвиг по обеим координатам

Алгоритм генерации.

Выбираются начальные параметры: какие координаты зафиксированы; направление сдвига (из 8ми возможных: 4 по горизонтали, 4 по диагонали).

Производится сдвиг и по правилам вычисляется верный ответ. Варьируя фиксаторы координат получается еще несколько неверных значений для дистракторов. Если значений не хватает выбирается другое направление сдвига и таким же образом генерируются неверные результаты.

8.2.5. A9 «Неравномерный код»

Условие.

Для кодирования некоторой последовательности, состоящей из букв А, Б, В, Г и Д, решили использовать неравномерный двоичный код, позволяющий однозначно декодировать двоичную последовательность, появляющуюся на приёмной стороне канала связи. Использовали код: А–1, Б–000, В–001, Г–011. Укажите, каким кодовым словом может быть закодирована буква Д. Код должен удовлетворять свойству однозначного декодирования.

1. 00
2. 01
3. 11
4. 010

Анализ решения.

Необходимо отбросить неверные варианты, нарушающие однозначность декодирования:

1. новый код не должен быть префиксом существующих кодов:

- 00 - префикс 000
- 01 - префикс 011

существующие коды не должны быть префиксами нового:

- 1 - префикс 11

Остаётся 010.

Алгоритм генерации.

1. Случайным образом строится двоичное дерево.
2. В процессе обхода дерева в глубину строятся двоичные коды.
3. Выбирается один код для ответа и несколько кодов для условия.
4. В качестве деструкторов берутся либо префиксы кодов из условия, либо к кодам из условия добавляются суффиксы.

8.2.6. A11 «Определить размер пароля, созданного по заданным правилам»

Условие.

Для регистрации на сайте некоторой страны пользователю требуется придумать пароль. Длина пароля – ровно 11 символов. В качестве символов используются десятичные цифры и 12 различных букв местного алфавита, причём все буквы используются в двух начертаниях: как строчные, так и заглавные (регистр буквы имеет значение!).

Под хранение каждого такого пароля на компьютере отводится минимально возможное и одинаковое целое количество байтов, при этом используется посимвольное кодирование и все символы кодируются одинаковым и минимально возможным количеством битов.

Определите объём памяти, который занимает хранение 60 паролей.

1. 540 байт
2. 600 байт
3. 660 байт
4. 720 байт

Анализ решения.

Ход решения.

1. Определить число бит, необходимых для кодирования 1 буквы алфавита. Для этого нужно найти число, являющееся степенью 2 не меньшее, чем длина алфавита.
2. Определить количество байт, необходимых для кодирования одного пароля. Необходимо умножить число бит для хранения 1 символа на длину пароля, разделить на 8 с округлением вверх.
3. Определить количество байт для хранения 60 паролей. Умножить полученный на предыдущем шаге результат на 60.

Возможные ошибки:

1. в ходе решения можно не учесть, что буквы используются в 2х регистрах

2. можно случайно отвести под каждый символ целое число байт

3. можно неверно посчитать количество бит, необходимых для хранения одного пароля

Описание изменений.

Задание аналогично результатам работы генератора A2 car_numbers. Есть отличия:

• другая легенда

• дополнительная сложность - буквы алфавита используются в двух начертания: строчные и заглавные.

Для реализации общий код из существующего генератора был вынесен в отдельные процедуры и использован для создания заданий нового типа.

8.2.7. A12 «Переворот массива»

Условие.

В программе используется одномерный целочисленный массив A с индексами от 0 до 9. Ниже представлен фрагмент программы, записанный на разных языках программирования, в котором значения элементов сначала задаются, а затем меняются.

Бейсик

```
 FOR i=0 TO 9
   A(i) = 9-i
 NEXT i
 FOR i = 0 TO 4
   k = A(i)
   A(i) = A(9-i)
   A(9-i) = k
 NEXT i
```

Паскаль

```
for i:=0 to 9 do
  A[i] := 9-i;
for i:=0 to 4 do
begin
  k := A[i];
  A[i] := A[9-i];
  A[9-i] := k;
end;
```

Си

```
for (i=0;i<=9;i++)
  A[i] = 9-i;
for (i=0;i<=4;i++)
{
  k = A[i];
  A[i] = A[9-i];
  A[9-i] = k;
}
```

Алгоритмический язык

```
нц для i от 0 до 9
  A[i]:= 9-i
кц
нц для i от 0 до 4
  k := A[i]
  A[i] := A[9-i]
  A[9-i] := k
кц
```

Чему будут равны элементы этого массива после выполнения фрагмента программы?

1. 9 8 7 6 5 4 3 2 1 0
2. 0 1 2 3 4 5 6 7 8 9
3. 9 8 7 6 5 5 6 7 8 9
4. 0 1 2 3 4 4 3 2 1 0

Анализ решения.

Необходимо понять, что в первом цикле массив A инициализируется числами от 9 до 0. Во втором цикле элементы массива переупорядочиваются в обратном порядке.

Параметрами для генератора могут быть:

1. размер массива
2. инициализация массива: цикл вперёд или назад, заполнения от большего числа к меньшему или наоборот
3. операции, производимые во 2м цикле, цикл вперёд или назад

Дистракторы:

- элементы массива в обратном порядке
- копирование элементом (слева направо/справа налево) вместо обмена

Алгоритм генерации и ограничения.

1. Ограничение на размерность массива 8 .. 12 = 10(в оригинале условия) +- 2.
2. Для разнообразия в задание добавлено:
    - инициализация массива целыми числами: n-1 .. 0 (а не 0 .. n-1)
    - обратный порядок присваиваний при перестановке элементов в массиве

8.2.8. B01 «Изменение размера перекодированного сообщения»

Условие.

Автоматическое устройство осуществило перекодировку информационного сообщения на русском языке длиной в 20 символов, первоначально записанного в 2-байтном коде Unicode, в 8-битную кодировку КОИ-8. На сколько бит уменьшилась длина сообщения?

В ответе запишите только число.

Анализ решения.

8 бит в 2 раза меньше, чем 2 байта, следовательно каждый символ после перекодировки занимает на 1 байт меньше места. Значит сообщение стало занимать на 20 байт меньше, 20 байт это 160 бит.

Описание изменений.

Задание похоже на существующее задание A1 recode, где необходимо по изменению(после перекодировки) размера сообщения в байтах узнать его длину. Названия и размеры кодировок, использующиеся в генераторах вынесены в отдельную процедуру.

8.2.9. B04 «Список слов, составленных из нескольких букв»

Условие.

Все 5-буквенные слова, составленные из букв А, О, У, записаны в алфавитном порядке. Вот начало списка:

1. ААААА
2. ААААО
3. ААААУ
4. АААОА
5. …

Запишите слово, которое стоит на 240-м месте от начала списка.

Анализ решения.

Необходимо, отталкиваясь от последнего элемента списка, построить несколько предыдущих. Число 240 близко к 243, а 243 - это номер последнего слова в списке. Тогда последние элементы списка следующие:

243. УУУУУ
242. УУУУО
241. УУУУА
240. УУУОУ

Возможные параметры для генератора:

- Мощность алфавита
- Длина слова
- Отступ от последнего слова

Алгоритм генерации.

1. Генерация задания заключается в выборе предложенных выше параметров и составлении в соответствии с ними текста.
2. Решение вычисляется по алгоритму изложенному выше при рассмотрения решения.

8.2.10. B05 «Недостающее значение в электронной таблице»

Условие.

Дан фрагмент электронной таблицы:

	A	**B**	**C**	**D**
1	3		3	2
2	=(C1+A1)/2	=C1–D1	=A1–D1	=B1/2

Какое число должно быть записано в ячейке B1, чтобы построенная после выполнения вычислений диаграмма по значениям диапазона ячеек A2:D2 соответствовала рисунку?

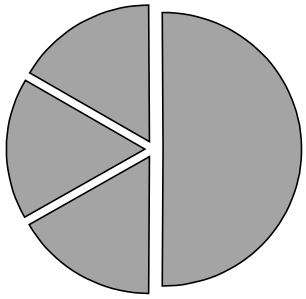

Диаграмма 4: Задание B05

Анализ решения.

Рассмотрим приведённое выше задания. Значения всех ячеек второй строки, кроме одной, вычисляются на основе известных значений, их стоит вычислить в первую очередь. После этого по диаграмме определяется значение последней ячейки во второй строке. После этого определяется значение в пустой ячейке.

Возможные параметры генератора:

- Количество столбцов в таблице
- Значения в известных ячейках
- Формулы в ячейках второй строки

Алгоритм генерации.

В случае случайной генерации формул необходимо следить за сложностью получаемого задания, а так же за существованием и единственностью решения. Это сильно усложняет генерацию, поэтому в целях упрощения процедуры генерации уравнения записываются в ручную программистом. Для внесения разнообразия в получаемые варианты заданий ячейки в 1й и 2й строке случайным образом перемешиваются, в соответствии с этим меняются координаты в формулах.

8.2.11. B08 «Основание системы счисления»

Условие.

Запись числа 67_10 в системе счисления с основанием N оканчивается на 1 и содержит 4 цифры. Чему равно основание этой системы счисления N?

Анализ решения.

Решать перебором, отбросив заведомо неверные варианты ответа.

Алгоритм генерации и ограничения.

1. Выбираются параметры - число(10 .. 100) и основание системы счисления (2 .. 9).
2. Перебором проверяется, есть ли другие системы исчисления, в которых

результат имеет столько же цифр и такую же последнюю цифру. Если другая система счисления есть - параметры генерируются заново.

8.2.12. B11 «Маска подсети»

Условие.

В терминологии сетей TCP/IP маской сети называется двоичное число, определяющее, какая часть IP-адреса узла сети относится к адресу сети, а какая — к адресу самого узла в этой сети. Обычно маска записывается по тем же правилам, что и IP-адрес. Адрес сети получается в результате применения поразрядной конъюнкции к заданному IP-адресу узла и маске. По заданным IP-адресу узла и маске определите адрес сети.

IP –адрес узла:	217.233.232.3
Маска:	255.255.252.0

При записи ответа выберите из приведенных в таблице чисел четыре элемента IP-адреса и запишите в нужном порядке соответствующие им буквы. Точки писать не нужно.

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
0	3	217	233	232	244	252	255

Пример. Пусть искомый IP-адрес 192.168.128.0, и дана таблица

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
128	168	255	8	127	0	17	192

В этом случае правильный ответ будет записан в виде: HBAF

Анализ решения.

Основная сложность задания в выполнении побитовой конъюнкчии чисел 252 и 232. Оба числа не являются степенью 2ки.

Алгоритм генерации.

1. Генерируются ip-адрес не равный нулю. И маска, содержащая и нули и единицы.
2. Вычисляется ответ.
3. Генерируются деструкторы:
   - применение маски к ip-адресу, используя побитовое “или” вместо побитового “и”
   - числа из исходного ip-адреса
   - числа из маски

8.2.13. B12 «Запрос к поисковой системе»

Условие.

В языке запросов поискового сервера для обозначения логической операции «ИЛИ» используется символ «|», а для логической операции «И» – символ «&». В таблице приведены запросы и количество найденных по ним страниц некоторого сегмента сети Интернет.

Запрос	Найдено страниц(в тысячах)
Шахматы | Теннис	7770
Теннис	5500
Шахматы & Теннис	1000

Какое количество страниц (в тысячах) будет найдено по запросу Шахматы? Считается, что все запросы выполнялись практически одновременно, так что набор страниц, содержащих все искомые слова, не изменялся за время выполнения запросов.

Анализ решения.

Обозначим:

A	Теннис
B	Шахматы
A|B	Шахматы | Теннис
A&B	Шахматы & Теннис

Тогда имеет место формула A|B = A + B - A&B. Любое недостающее значение может быть восстановлено, если известны остальные 3.

Алгоритм генерации.

1. Генерируются значения A, B, A&B. По формуле приведенной выше вычисляется A|B.
2. В тексте задания показываются произвольные 3 значения, 4е необходимо найти.

8.2.14. B «Программа из прибавлений и вычитаний»

Условие.

У исполнителя Кузнечик две команды:

1. прибавь 3,
2. вычти 2.

Первая из них увеличивает число на экране на 3, вторая – уменьшает его на 2 (отрицательные числа допускаются).

Программа для Кузнечика – это последовательность команд. Сколько различных чисел можно получить из числа 1 с помощью программы, которая содержит ровно 5 команд?

Анализ задания.

Используются только сложение и вычитание, следовательно, в силу коммутативности сложения, порядок применения операций не важен. Результат зависит только от количества применений 1й или 2й операции. Если длина программы n, то существует n + 1 вариантов по-разному применить команды исполнителя.

Алгоритм генерации.

Случайным образом выбираются длина программы и величины, на которые исполнитель увеличивает или уменьшает исходное число. Далее, генерируется верный ответ и текст задания.

8.2.15. B15 «Специфичная формула математической логики»

Условие.

Сколько существует различных наборов значений логических переменных $x_1, x_2, \ldots x_9, x_{10}$, которые удовлетворяют всем перечисленным ниже условиям?

$$((x_1 \equiv x_2) \vee (x_3 \equiv x_4)) \wedge (\neg(x_1 \equiv x_2) \vee \neg(x_3 \equiv x_4)) = 1$$
$$((x_3 \equiv x_4) \vee (x_5 \equiv x_6)) \wedge (\neg(x_3 \equiv x_4) \vee \neg(x_5 \equiv x_6)) = 1$$
$$((x_5 \equiv x_6) \vee (x_7 \equiv x_8)) \wedge (\neg(x_5 \equiv x_6) \vee \neg(x_7 \equiv x_8)) = 1$$
$$((x_7 \equiv x_8) \vee (x_9 \equiv x_{10})) \wedge (\neg(x_7 \equiv x_8) \vee \neg(x_9 \equiv x_{10})) = 1$$

В ответе не нужно перечислять все различные наборы значений $x_1, x_2, \ldots x_9, x_{10}$, при которых выполнена данная система равенств. В качестве ответа вам нужно указать количество таких наборов.

Анализ решения.

1. Для начала рассмотрим случай 4х переменных:

((x1 ≡ x2) ∨ (x3 ≡ x4)) \ (¬(x1 ≡ x2) \ ¬(x3 ≡ x4)) = 1

Уравнение выше можно записать так:

x1 = x2 <=> x3 != x4 x1 != x2 <=> x3 = x4

8 наборов переменных удовлетворяют этому уравнению.

2. Теперь переходим к случаю 6ти перменых: Для каждого из уже выбранных наборов для случая 4х переменных можем добавить по 2 набора из x5, x6

3. Для n = 2*k переменных получаем:

Всего k-1 пар. Первая пара даёт 8 вариантов, каждая последующая увеличивает число вариантов вдвое. Имеем:

8 * 2 ^ (k - 2) = 2 ^ (k + 1)

Алгоритм генерации.

Единственный параметр для: количество переменных. Ответ вычисляется по формуле, приведённой выше.

8.3. Стандарт кодирования

Принят автором системы, соответствует рекомендациям автора ЯП perl[26] и состоит в следующем:

1. Всегда использовать «use strict» и «use warnings»

2. Отступ состоит из 4х пробелов.
3. Закрывающаяся фигурная скобка многострочного блока должна находиться на таком же уровне отступа, как и ключевое слово, с которого начинается блок.
4. Пробел перед открывающейся фигурной скобкой многострочного блока.
5. Нет пробелов перед точной с запятой.
6. Точка с запятой отсутствует в конце коротких однострочных блоков.
7. Пробелы вокруг большинства операторов (в частности вокруг ..).
8. Пробелы вокруг сложных выражений внутри скобок.
9. Пустые строчки отделяют логические блоки кода, делающие разные вещи.
10. Нет пробела между именем функции и открывающейся круглой скобкой.
11. Пробел после каждой запятой.
12. Длинные строки переносятся после операторов (за исключением операторов and и or).

8.4. Проект интерфейса

Результаты могут выводиться в следующих форматах:

• html

• json

9. Реализация и тестирование

В результате моей работы в проект добавлена 1601 строка кода и удалены 74 строки кода на языке программирования perl.

Генераторы заданий, реализованные мной, использовались на практике в ходе проведения сертификации по ЕГЭ в рамках Весеннего турнира юных программистов 12-15 мая 2011 г.

Заключение

Таким образом, в процессе выполнения курсовой работы мною был доработан существующий существующий открытый проект — система генерации тестовых заданий ЕГЭ. В систему были добавлены генераторы 8 типов заданий, что значительно увеличило актуальность системы.

Проект всё еще нуждается в доработке — необходимо добавлять новые генераторы. Кроме того, возможно добавление новых особенностей в систему, например осуществлять вместе с созданием варианта задания еще и описания процесса построения правильного решения.

Пользу от выполнения данной работы получил и я. Мною были изучены ЯП perl[21],

система контроля версий git[11] и текстовый редактор GNU/Emacs[24]

Список литературы

1: Словарь ЕГЭ - http://www1.ege.edu.ru/brief-glossary

2: Пси-шпаргалка психологический образовательный сайт - http://psylist.net/slovar/5a73.htm

3: Яндекс словари - http://slovari.yandex.ru

4: Just Whose Idea Was All This Testing? - http://www.washingtonpost.com/wp-dyn/content/article/2006/11/13/AR2006111301007_2.html

5: Теория и методика педагогических измерений - old.ustu.ru/Аванесов%20B.C.pdf?id=2421&jf=yes

6: Нормативно-правовые документы ЕГЭ - http://www1.ege.edu.ru/legal-documents

7: Основные сведения о ЕГЭ - http://www1.ege.edu.ru/main

8: Кафедра информатики ИМиКН ДВФУ - http://imcs.dvgu.ru/works/kif.html

9: Сайт ДВФУ - http://двфу.рф/

10: Зенкина А.О., курсовая работа на тему "Библиотека алгоритмов для генерации задач"

11: Система контроля версий git - http://git-scm.com/

12: Git-хостинг github.com - https://github.com/

13: Система автоматической генерации тестовых заданий ЕГЭ - https://github.com/klenin/EGE

14: Лицензия GPL2 - http://www.gnu.org/licenses/gpl-2.0.html

15: Описаниепрограммногокомплекса SuperTest - http://ipg.h1.ru/tests/mirsanov.les/super_test.html

16: Дьяченко Е.П, Редактор курсов. Дипломная работа

17: Random Expression Generator - http://matlabdb.mathematik.uni-stuttgart.de/download.jsp?MC_ID=9&MP_ID=118

18: Pinery: Генерация на основе регулярных выражений Perl - http://www.unitesk.ru/pinery/pinery-regexp.php

19: Методика конструирования систем генерацииндивидуальных домашних заданий по математическому анализу спримененим пакетов прикладных программ - http://window.edu.ru/window_catalog/les/r24265/2000_3-4_032.pdf

20: Модели и алгоритмы генерациизадач в компьютерном тестировании. Известия Томскогополитехнического университета - http://www.duskyrobin.com/tpu/2004-05-00030.pdf

21: Язык программирования perl - http://www.perl.org/

22: Язык шаблоно Template::Toolkit - http://template-toolkit.org/

23: Writing portable Perl - http://perldoc.perl.org/perlport.html

24: Многофункциональный текстовый редактор emacs - http://www.gnu.org/software/emacs/

25: Веб браузер Mozilla Firefox - http://www.mozilla.com
26: perl style guide - http://perldoc.perl.org/perlstyle.html