Web-сайт для создания и обмена генерируемыми задачами по математике

И.А.Посов
ассистент,
Санкт-Петербургский государственный электротехнический университет "ЛЭТИ",
ул. Профессора Попова, 5, г. Санкт-Петербург, 197376, (812)346-44-87
iposov@gmail.com

Аннотация

В статье описывается специальный web-сайт, представляющий собой базу многовариантных задач в помощь учителям и преподавателям для составления печатаных вариантов контрольных работ по любой предметной дисциплине школы и вуза. На примере математических курсов раскрываются проблемы структурирования данных, формата представления задач и технологий их создания, связанные с процессом наполнения базы задач сайта**,** обсуждается структурирование базы задач с помощью тегов и приводится ряд соображений по решению стандартных проблем, возникающих при их использовании. Представлен разработанный инструмент создания многовариантных задач, основанный на языке программирования JavaScript, имеющем удобные расширения, в частности, связь с бесплатной системой компьютерной алгебры Maxima.

The paper presents a specialized web-site with the collection of multivariant problems that may be used by teachers and professors for compiling printable variants of tests on any topic studied in school or university. On the example of mathematical courses the paper discusses questions of structuring problem base by means of tags and present several thoughts about solving standard issues usually arising when using tags. Finally, the paper presents the developed tool for creating multivariant problems based on JavaScript programming language with handy extensions, and particularly with the connection with the computer algebra system Maxima.

Ключевые слова

генерируемая задача, web-сайт, фолксономия, JavaScript, Maxima;
generated problems, web-site, JavaScript, Maxima.

Введение

Процесс обучения в любом высшем или среднем учебном заведении связан с периодической проверкой его результатов в виде многочисленных самостоятельных или контрольных работ. При проведении контрольных работ по математике в школе или вузе принято выдавать учащимся не один вариант заданий, а несколько. Это необходимо, чтобы усложнить списывание и тем самым уменьшить его влияние на

результаты работы. Обычно для проведения контрольной достаточно всего двух вариантов условий задач, в этом случае условия раздаются так, что сидящие рядом ученики имеют разные условия. Если работу предполагается решать дома, то экономить на вариантах не удается, и необходимо готовить столько же вариантов заданий, сколько учеников будут их решать.

Многовариантными будем называть далее задачи с несколькими вариантами условий, для уменьшения возможности списывания, которое снижает эффективность контроля над результатами обучения, искажая действительные данные о ходе учебного процесса. Существует, по крайней мере, два пути их подготовки.

Возможна подготовка "вручную", когда преподаватель сам придумывает похожие варианты условий и решает их для получения ответа. Этот процесс можно частично автоматизировать, сделав элементы условия и ответа зависящими от параметров. Значения параметров учащиеся берут либо в соответствии с номером своего варианта, либо вычисляют в зависимости от букв имени или даты рождения.

.Другой путь, доступный тем преподавателям, кто имеет навыки программирования, - это написание программы, которая автоматически создает все варианты условия задачи вместе с ответами.

Очевидно, что, независимо от выбранного метода создания, для реализации многовариантных задач необходимы навыки их разработки и привлечение вспомогательных ресурсов.

Например, в случае "ручной" подготовки неминуемы затраты личного времени преподавателя (учителя), при том, что необходимы особые аккуратность и усидчивость, при наличии же у кого-либо из них начальных навыков программирования – умение программировать, а также требуется время на написание и отладку программы

К сожалению, к большинству создаваемых многовариантных задач имеют доступ только авторы этих задач и, возможно, несколько их коллег. В лучшем случае создаваемые задачи включаются в методические пособия, которыми удается пользоваться внутри вуза.

Для решения именно проблемы доступности многовариантных задач нами создается web-сайт “База генерируемых задач” [4]. Основная цель сайта – предложить преподавателям (учителям) готовую базу многовариантных задач, в которой они смогли бы отыскать необходимые задачи для составления однотипных вариантов контрольных заданий, с возможность их распечатки. Помимо этого, сайт предлагает своим пользователям специальные инструменты для создания собственных многовариантных задач.

В настоящее время, за небольшим исключением, создание задач для нашего сайта требует от преподавателя (учителя) написания программы.

Преподаватель, не умеющий программировать, может поместить на сайт свою многовариантную задачу, введя вручную несколько условий задачи. Но количество и разнообразие таких заданий (с самостоятельно введенными условиями) будут отличаться от задач, создаваемых программистами. Приведем пример, поясняющий отличие.

Программист создаёт задачу «Вычислите наибольший общий делитель чисел *** и ***» (базовый пример возможностей автоматического создания условий задач). Программа автоматически генерирует числа, чтобы подставить их вместо звездочек, и сама же вычисляет ответ. Она может создать более тысячи условий этой задачи, подбирая числа так, чтобы задачи получались одинаковой сложности.

Например, чтобы оба числа были трехзначными и количество шагов в алгоритме Евклида, используемом для решения, было бы примерно одинаковым.

Подготовить вручную такой объём задач с каким-либо конкретным ограничением вряд ли возможно. (Другие примеры с более разнообразными условиями будут даны несколько позже).

В общем случае подобным автоматическим приёмом можно создавать условия заданий практически ко всем разделам школьной и университетской математики, информатики, в некоторой степени и по другим предметам.

В этой статье мы ограничимся обсуждением проблем, связанных с работой преподавателей, программирующих многовариантные задачи для наполнения сайта. Более конкретно, мы обсудим формат, в котором задачи хранятся в базе сайта, технологии их создания в этом формате, а также определимся с правилами классификации хранящихся на сайте материалов.

Правильно выбранная технология создания задач является одним из ключевых факторов, которые ускорят процесс наполнения сайта задачами. Основным руководством к выбору должно быть то, что бы преподавателю было легко и удобно создать задачу для себя именно на основе предлагаемой технологии. В этой ситуации доступность задачи другим пользователям сайта является, скорее, полезным побочным эффектом.

Выбор правильного метода категоризации публикуемых материалов необходим, чтобы сделать удобным поиск задач внутри сайта: преподавателю, определившемуся с темой интересующих его задач, должно быть просто найти их, (конечно, при наличии) в базе сайта.

База генерируемых задач

Перейдём к проблемам классификации материалов сайта и технологиям, позволяющим создавать для него новые задачи. Вопросы его устройства и возможностей: пользовательский интерфейс, процедуры составления контрольных из имеющихся задач и прочее, мы опускаем, т.к. они выходят за рамки этой статьи. Приведем лишь один пример работы с содержимым сайта.

На рисунке 1 изображен внешний вид одной из страниц web-сайта “База генерируемых задач”. На этом экране преподаватель просматривает созданную им "заготовку" контрольной работы (в данном случае она состоит из трех задач). Преподаватель может изменять порядок задач в списке, удалять задачи или добавлять другие задачи, найденные при поиске. Для добавления задач ему следует переключиться в режим поиска задач, и, если он найдёт необходимую, то переместить её в одну из своих "заготовок", расположенных на экране слева.

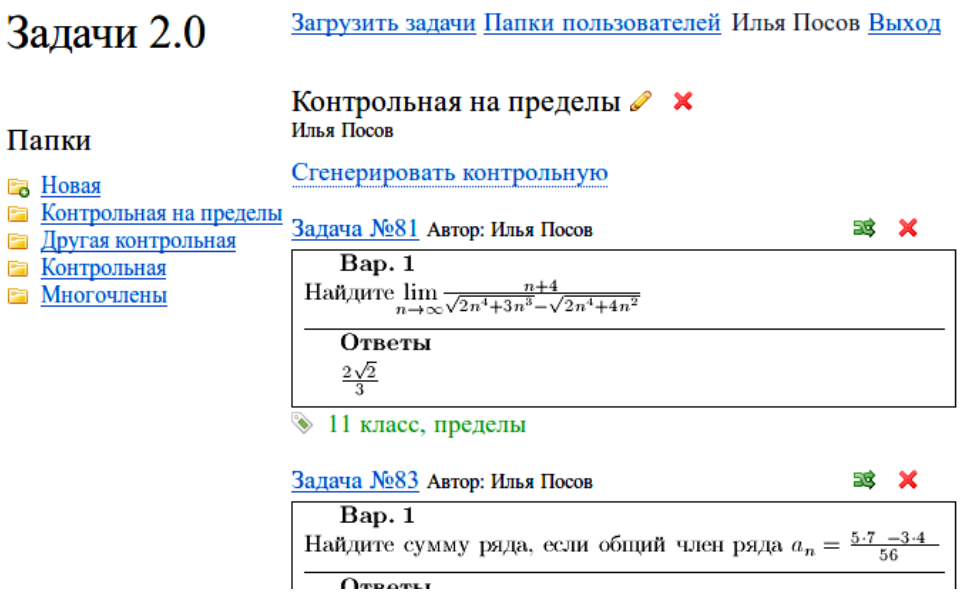

Рис. 1. Внешний вид сайта "База Генерируемых Задач"

Когда "заготовка контрольной" создана, можно “сгенерировать” саму контрольную работу, т.е. получить её печатаемый вариант с несколькими вариантами условий. Для этого используется ссылка “сгенерировать контрольную”, расположенная на экране сверху. Внешний вид получаемой контрольной дан на рисунке 2.

Вар. 1
1. Найдите $\lim\limits_{n\to\infty}\frac{\sqrt{n^2-5}-\sqrt{n^2+2}}{n^4-5n^2}$
2. Найдите сумму ряда, если общий член ряда $a_n = \frac{5\cdot 7^n - 4\cdot 2^n}{14^n}$
3. В последовательности $\frac{(-1)^{n-1}n}{6} + \frac{(-1)^{n+1}}{n+4}$ найдите a_{18}.

Вар. 2
1. Найдите $\lim\limits_{n\to\infty}\frac{\sqrt{n^5-3n}-\sqrt{n^5+4}}{2n^3+2n}$
2. Найдите сумму ряда, если общий член ряда $a_n = \frac{5\cdot 3^n + 3\cdot 2^n}{6^n}$
3. В последовательности $3(-1)^{n-1}n - \frac{2}{n-5}$ найдите a_{10}.

Вар. 3
1. Найдите $\lim\limits_{n\to\infty}\frac{\sqrt{2n^7+5n^2}-\sqrt{2n^7+5n^6}}{2n^3-3n^2}$
2. Найдите сумму ряда, если общий член ряда $a_n = \frac{3\cdot 4^n + 5^n}{40^n}$
3. В последовательности $n + \frac{2}{n+4}$ найдите a_{25}.

Вар. 4
1. Найдите $\lim\limits_{n\to\infty}\frac{\sqrt{n^5+5n}-\sqrt{n^5+n^4}}{2n^3+3n}$
2. Найдите сумму ряда, если общий член ряда $a_n = \frac{3-5\cdot 6^n}{18^n}$
3. В последовательности $2(-1)^n n - \frac{(-1)^n 2}{n+2}$ найдите a_{23}.

Вар. 5
1. Найдите $\lim\limits_{n\to\infty}\frac{\sqrt{n^8-n^4}-\sqrt{n^8-n^2}}{2n^3-4n^2}$
2. Найдите сумму ряда, если общий член ряда $a_n = \frac{5^{n+1}+4}{15^n}$
3. В последовательности $3(-1)^n n + \frac{(-1)^{n-1}}{n+1}$ найдите a_{25}.

Вар. 6
1. Найдите $\lim\limits_{n\to\infty}\frac{\sqrt{n^3+n^4}-\sqrt{n^3+2n^4}}{2n^4+5}$
2. Найдите сумму ряда, если общий член ряда $a_n = \frac{2^{n+1}+3\cdot 7^n}{28^n}$
3. В последовательности $2(-1)^{n-1}n - \frac{(-1)^{n-1}2}{n}$ найдите a_{30}.

Вар. 1
1. Найдите $\lim\limits_{n\to\infty}\frac{\sqrt{n^2-5}-\sqrt{n^2+2}}{n^4-5n^2}$
2. Найдите сумму ряда, если общий член ряда $a_n = \frac{5\cdot 7^n - 4\cdot 2^n}{14^n}$
3. В последовательности $\frac{(-1)^{n-1}n}{6} + \frac{(-1)^{n+1}}{n+4}$ найдите a_{18}.

Рис. 2. Внешний вид печатного варианта контрольной с увеличенным фрагментом

Рисунок 1 (справа) демонстрирует новые примеры задач, варианты условий которых могут создаваться (генерироваться) автоматически. На рисунке 2 показано 6 таких вариантов.

Первая задача контрольной (рис. 1) – это задача на поиск предела

последовательности, для нахождения которого необходимо домножить числитель дроби, на сопряженное выражение.

Вторая задача – это задача на поиск суммы ряда, для решения которой применяется формула суммы членов геометрической прогрессии.

Третья – задача на определение численного значения конкретного члена знакопеременного ряда.

Форматирование печатных вариантов контрольной основано на программном модуле А.В. Степанова, созданном им для своей системы автоматической генерации задач [6]. Модуль реализован на TeX- системе компьютерной верстки математических текстов. С его помощью условия задач располагаются в удобной "для нарезания" на варианты таблице.

Введем термины, которые будут использоваться в данном тексте.

На рисунке 3 изображен процесс создания и использования задач внутри сайта.

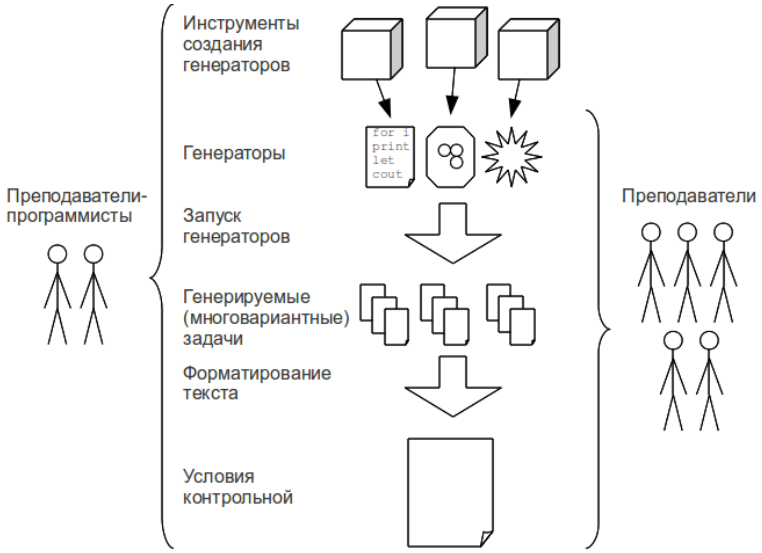

Рис. 3. Создание и использование задач

Задачам на сайте соответствуют *генераторы*. Генераторы – это некоторые объекты, которые могут быть *запущены* сайтом. Результатом одного запуска генератора является случайное условие задачи.

Например, после каждого запуска генератора задачи на вычисление НОД создается условие задачи, в котором вместо звездочек подставлены числа. Чтобы получить несколько вариантов контрольной, необходимо генератор каждой задачи запустить несколько раз.

Генераторы могут быть

либо запускаемыми программами, создающими условия,

либо подготовленным списком условий, из которых (при запуске случайным образом) выбирается одно из них,

либо быть устроены ещё каким-либо образом.

Многовариантные задачи, которые получаются в результате запуска генераторов, мы называем генерируемыми. Генераторы однозначно соответствуют создаваемым им генерируемым задачам, поэтому часто эти термины используются взаимозаменяемо.

К примеру, в названии статьи вынесено “создание и обмен генерируемыми задачами”, хотя в соответствии с приведенными выше определениями, точнее было бы сказать “создание и обмен генераторами”.

Для составления контрольных работ с помощью генераторов специальных знаний не требуется, поэтому их могут использовать любые преподаватели. Запуском генераторов и созданием печатных вариантов контрольной работы сайт "занимается" автоматически.

Разработка же самих генераторов является более сложной задачей, доступной в общем случае только узкому классу преподавателей-программистов, умеющих применять инструменты, предназначенные для их создания.

Такие инструменты могут быть и требующими полноценного программирования, и достаточно простыми, как инструменты, предлагающие для создания генератора ввести вручную несколько вариантов условий (выше уже отмечалось, что автоматически можно создать намного больше условий задачи, чем вручную).

Структурирование материалов сайта

Для успешной работы с "Базой Генерируемых Задач" недостаточно удобных технологий создания задач, необходимо структурирование материала, позволяющее легко находить добавленные задачи, в противном случае пользоваться сайтом будет невозможно.

Для решения проблемы структурирования материалов традиционно используются два подхода: таксономия и фолксономия.

Таксономию в случае web-сайта можно понимать как рубрикатор. Каждому элементу базы назначается одна или несколько рубрик рубрикатора, а сами рубрики образуют древовидную структуру.

Например, подобный подход можно встретить на известном в Интернете сайте, базе математических задач problems.ru [7]. На сайте проекта собраны задачи, используемые в работе математических кружков. Это задачи из специализированных задачников, математических олимпиад и соревнований и других источников.

Рубрики рубрикатора содержат разбиение по темам и используемым в решении методам. Одна и та же задача может быть занесена, к примеру, и в рубрику “теория чисел” и в рубрику “принцип Дирихле”. Показательно, что наполняют базу только владельцы сайта, посторонние пользователи могут лишь указать на задачу или источник задач. Такой подход гарантирует качество задач сайта, а так же актуальность и полноту рубрикатора.

Применение рубрикатора для сайтов, которые наполняются распределенно большим количеством участников, представляется маловозможным.

Пользователь после добавления информации на такой сайт (в нашем случае – после добавления генератора) зачастую не хочет тратить силы на заполнение информации о добавленной информации (метаинформации), т.е. не желает выбирать необходимые рубрики рубрикатора, руководствуясь при этом сводом правил, определяющим требование к содержимому каждой рубрики.

Рубрикатор на сайте с распределенным наполнением обычно поддерживается модераторами, которые самостоятельно назначают рубрики материалам или, по крайней мере, следят за правильностью назначения рубрик.

Фолксономию можно определить как "народную" таксономию. Структуризацией материалов занимаются пользователи сайта, пополняющие его содержимое.

Традиционно для этого используются теги (ключевые слова). При добавлении информации пользователь перечисляет ряд ключевых слов, которые, по его мнению, наиболее точно характеризуют этот материал. Использование тегов сопряжено с рядом трудностей.

Во-первых, по-прежнему существует нежелание пользователей тратить время

на указание тегов.

Во-вторых, при вводе данных происходит разрастание банка "разных" тегов из-за опечаток, синонимов и сокращений (например, “теорвер” и “теория вероятностей”).

В-третьих, использование тегов задаёт плоскую структуру, что не позволяет выстраивать иерархиею понятий.

Например, теги “планиметрия” и “подобие треугольников” невозможно связать без внесения принципиальных изменений в технологию использования тегов.

Во всех случаях описанные трудности приводят к уменьшению полноты поиска, т. е. пользователи находят по заданной теме меньше задач, чем их имеется на сайте.

На сайте "База Генерируемых Задач" предполагается нецентрализованное появление задач за счёт работы преподавателей, поэтому было принято решение использовать *фолксономию*.

Автору генерируемой задачи (генератора) при её добавлении на сайт необходимо вводить характеризующие теги.

Чтобы подтолкнуть пользователей к самостоятельному структурированию материалов и помочь им справиться с трудностями, возникающими при использовании тегов, предлагается следующий ряд мер:

1.Проверка орфографии для вводимых тегов;

2.Выбор из уже существующих тегов.

При вводе слова появляется для выбора список тегов, содержащих это слово или его введенное начало;

3.Интеллектуальная подсказка тегов для задачи.

Для подсказки анализируется текст условия задачи, ищутся ключевые слова и выражения, которые могут свидетельствовать о вероятных тегах. Связь ключевых слов и тегов определяется на основе всей базы задач;

4.Предупреждения о нерекомендуемых тегах.

К таковым относятся теги, помечающие изменяющиеся параметры

Например, сложность и класс, для которого предназначена задача. Эти параметры изменяются в зависимости от учебного курса. При попытке ввода тегов пользователь предупреждается об этом.

5.Рекомендации автору задачи по добавлению новых тегов на основе информации об её использовании другими пользователями.

6.Синонимичные теги.

При обнаружении модераторами синонимов (например, «теорвер» и «теория вероятностей»), они помечаются как синонимичные и не различаются в процессе поиска;

7.Обобщающие теги.

Модераторы, как и в предыдущем случае, обнаруживают популярные теги, один из которых обобщает другой, и фиксируют этот факт. Информация об обобщении учитывается при поиске задач;

8.Рубрикатор рекомендуемых тегов.

Создается древовидный рубрикатор, элементами которого являются теги. При описании задачи пользователь имеет возможность свериться с рубрикатором и выбрать предлагаемые им теги.

Перечисленные возможности будут реализовываться по мере развития сайта, так как при маленьких размерах базы задач нужны не все из них.

На сегодняшний день на сайте

для нахождения интересующих пользователя задач предлагается использовать теги и тексты условий, причём поиску доступны математические символы внутри формул;

для структурирования найденных задач формируются списки, т.е.

упорядоченные последовательности задач, собранные для произвольных целей, в том числе, для создания контрольных работ.

В дальнейшем планируется следующая поддержка зарегистрированных пользователей: возможность создавать контрольные работы, т. е. упорядоченные списки задач, из которых сайт "позволяет" генерировать и форматировать для вывода на печать необходимое количество вариантов.

Разновидности методов создания генераторов задач

Известные автору данной статьи методы создания генераторов можно объединить в четыре условных класса:

ручной ввод условий,
использование предметно-ориентированных редакторов,
параметризация,
программирование.

Генераторы, в соответствии с используемым методом создания, называются

выбирающими,
предметно-ориентированными,
параметрическими,
программируемыми.

Программируемые генераторы, – это генераторы**,** требующие написания полноценной программы.

Например, такие генераторы необходимы для создания задач дискретной математики, требующих применения некоторых алгоритмов, таких, как алгоритмов Прима и Краскала для построения минимального остовного дерева, или алгоритма RSA, применяемого для шифрования сообщений.

Для комфортной работы преподавателя-программиста необходимо предоставить ему среду для разработки генераторов, снабженную всеми необходимыми инструментами, а также язык программирования, имеющий доступ к полезным при создании генераторов библиотекам, системам символьных и численных вычислений.

В рамках сайта был создан язык программирования, основанный на JavaScript. При его использовании требуется только лишь описывать логику создания задачи и нет необходимости работать с файлами или генератором случайных чисел, также в язык добавлены возможности доступа к системе символьных вычислений Maxima, подробнее о языке написано ниже.

Параметрические генераторы, создание которых доступно более широкому кругу преподавателей, в том числе и не обладающим полноценными навыками программирования.

Здесь пользователю достаточно написать текст условия задачи и её ответ, вставив в отдельных местах текста выражения, зависящие от параметров, а далее описать условия на эти параметры.

Приведем пример описания параметрического генератора (задача на поиск наибольшего общего делителя). Он не привязан к какой-либо существующей технологии, а лишь демонстрирует подход. Похожий подход используется в работе [2].

Условие:
С помощью алгоритма Евклида вычислите
наибольший общий делитель чисел {a} и {b}.
Ответ:
НОД({a}, {b}) = {gcd(a,b)}
Параметры:

a = случайно(1,100)
b = случайно(1,100)
Условия на параметры:
gcd(a,b) > 2
gcd(a,b) < 10
a mod b != 0
b mod a != 0

Этот текст необходимо записать с помощью соответствующего инструмента для создания параметрического генератора.

В первом и втором разделе заданы условие и ответ к задаче, выражения в фигурных скобках вычисляются отдельно для каждого условия задачи. gcd(*a*,*b*) – это выражение, которое вычисляется в наибольший общий делитель чисел *a* и *b*.

В разделе “Параметры” указывается, как выбираются значения параметров, в данном случае это случайные числа в диапазоне от 1 до 100. Условия на параметры нужны, чтобы генерировать задачи одинаковой заранее заданной сложности. Два условия на НОД требуют, чтобы получившийся НОД был в диапазоне от 3 до 9. Последние два условия требуют, чтобы числа не делились друг на друга, т. е. чтобы при выполнении алгоритма Евклида нужно было бы совершить хотя бы два шага.

Более правильно было бы указать условие на параметры, что количество шагов в алгоритме Евклида находилось бы в определенном диапазоне, например, от 4 до 5. Мы не готовы реализовывать здесь полностью алгоритм Евклида, чтобы посчитать количество его шагов, потому что в этом случае этот генератор пришлось бы отнести к предыдущему рассмотренному классу программируемых генераторов. Вместо этого можно было бы написать условия gcd_steps(*a*, *b*) > 3 и gcd_steps(*a*, *b*) < 6, но только в том случае, если эти функции доступны при создании генератора. Как видно, преподаватель, создающий параметрические генераторы, ограничен набором доступных функций для задания ограничений на значения параметров.

Некоторые математические аспекты генерирования задачи на алгоритм Евклида обсуждаются в работе [5].

Предметно-ориентированные генераторы, создание которых доступно более широкому кругу преподавателей, в том числе и не обладающих знаниями программирования.

Для создания генератора преподаватель запускает программу-редактор, с помощью пользовательского интерфейса задаёт необходимые настройки будущего генератора, нажимает кнопку “создать” и получает его.

Приведем пример описания возможного редактора по теме “системы счисления”.

Для создания генерируемой задачи на перевод чисел между системами счислений преподаватель выбирает исходную систему счисления, диапазон, из которого выбирается число для перевода, выбирает вторую систему счисления для перевода.

Таким образом можно получить, например, генератор задач на перевод из десятичной системы в двоичную, где число для перевода выбираются случайно из заданного диапазона.

Системы счисления для переводов также могут выбираться случайно из определенного заданного преподавателем диапазона. Редактор может предлагать настраивать, разрешено ли при генерации использовать отрицательные или нецелые числа, еще более функциональный редактор позволит создавать генераторы на простейшие арифметические действия в различных системах счисления и на решение линейных уравнений.

Описанный редактор позволяет формировать генераторы заданий на перевод чисел между системами счислений, на арифметические действия и на решение линейных уравнений одной переменной в недесятичных системах счисления

Подобные редакторы способны создавать только генераторы очень узкого класса задач, так как нелегко совместить удобный пользовательский интерфейс для непрофессионалов в программировании, и широкую функциональность.

Выбирающие генераторы

Последний метод позволяет превратить любую многовариантную задачу в генерируемую, не требует специальных знаний и доступен многим. Преподаватель, создавая генератор, вводит в базу несколько условий одной задачи. При запуске генератора для формирования задания не создается условие задачи, как при работе предыдущих видов генераторов, а лишь выбирается одно из хранящихся в нем условий. Единственная сложность использования этого метода на сайте состоит в том, что преподавателю необходимо оформлять условия в формате TeX.

Один из вариантов использования этой технологии состоит в том, что преподаватель в качестве условий задачи вводит вопросы к экзамену. При создании печатного варианта контрольной работы с использованием такого генератора будут получаться билеты к экзамену.

Представление генераторов на сайте

Обсудим, как именно сайт хранит генераторы задач. При выборе их формата представления учитывается, что он должен удовлетворять следующим требованиям:

1. Необходимо уметь хранить все разновидности генераторов, созданных описанными выше методами;
2. Должна существовать возможность преобразования любого генератора, созданного "независимо от сайта", в формат, используемый на сайте.

Произвольные программные генераторы, создаваемые преподавателями-программистами своими средствами, нельзя хранить внутри сайта, так как,

во-первых, сайт "не обучен" с ними взаимодействовать для построения многовариантных задач;

во-вторых, программы могут содержать вредоносный кода;

в-третьих, можно быть практически уверенными, что они не запустятся из-за несоответствия платформ.

Для выбирающих генераторов задач, которые работают с уже созданными условиями, таких проблем не возникает. То есть, любой выбирающий генератор, оформленный в "понятном" для сайта формате, можно поместить в "Базу генерируемых задач".

Программный генератор может служить основой для создания выбирающего. Для этого достаточно запустить его несколько раз и получить желаемое количество условий задачи. Затем, оформив соответствующим образом, в виде выбирающего генератора разместить на сайте.

Такое преобразование используется в настоящее время для наполнения сайта.

Генераторы задач, созданные в прошлом преподавателями кафедры ВМ-2 СПбГЭТУ для системы генерации задач А.В. Степанова, не были предназначены для сайта, и не могут использоваться в нем в “первоначальном” виде, но они преобразуются в выбирающие и помещаются в базу.

За основу формата представления генераторов на сайте выбран формат представления задач в системе проведения удаленных соревнований DCES [3]. В упрощенном понимании задача этого формата является архивом, содержащим файлы (ресурсы) с информацией о задаче и файлом метаинформации, описывающим смысл ресурсов для системы. В нашем случае генератор задач – это архив, в котором содержится информация для генерации.

Для выбирающих генераторов внутри архива содержатся файл с перечислением условий задач, файл с перечислением соответствующих условиям

ответов, а также файл с метаинформацией, который содержит информацию о том, что задача генерируется именно с помощью выбирающего генератора и содержит ссылки на файлы с условиями и ответами. Помимо файлов с условиями и ответами архив также может содержать и другую информацию.

Для программных генераторов имеет смысл хранить в архиве запускаемую программу и её исходный код. Сайт не будет использовать файлы с кодами, но к ним можно будет обратиться в случае обнаружения ошибок в работе программы.

Инструментарий для создания генераторов

Выбирающие генераторы, несмотря на свою универсальность, создаются "напрямую" лишь для задач с небольшим количеством вариантов условий. Во всех остальных случаях они формируется из других разновидностей генераторов.

Здесь мы обсудим создание программируемых генераторов, как разновидности, имеющей больше всего возможностей по созданию задач.

На данный момент в рамках проекта по созданию сайта "База генерируемых задач" разработан инструмент создания программных генераторов задач. Активное участие в разработке принимала магистрант Санкт-Петербургского Академического университета РАН С. Александрова.

Из существующих языков программирования выбран JavaScript, причины выбора этого языка приведены ниже.

При выборе языка программирования первым делом была отброшена идея создавать язык с нуля.

С одной стороны создать собственный язык программирования генераторов – совсем не плохая идея. Язык может быть похож на описанный выше гипотетический язык создания параметрических генераторов.

Действительно, весь код программы сконцентрирован именно на создании условия задачи, параметрах, методах подбора параметров и т. п. Хорошим решением могло бы быть создание собственного языка с помощью системы метапрограммирования MPS (Meta Programming System) фирмы JetBrains [1], которая как раз позволяет создавать языки сразу вместе со средой разработки.

Но, с другой стороны, система MPS еще недостаточно “созрела” для повсеместного использования, она требует мощный компьютер и имеет сложный для понимания интерфейс.

Перечислим достоинства JavaScript, на основе которых было принято решение использовать именно его для разработки генераторов.

Во-первых, JavaScript распространен и прост в изучении. Если даже программист не сталкивался с этим языком в своей практике, то он быстро изучит его на требуемом уровне.

Во-вторых, JavaScript способен исполняться внутри браузера. Это означает, что для создания, отладки и преобразования созданного генератора в универсальный выбирающий не требуется никакого дополнительного программного обеспечения.

Возможно даже отказаться от такого преобразования, потому что программу можно запускать каждый раз при необходимости создать условия, но в этом случае надо учитывать, что пользователи будут запускать у себя программы, полученные из непроверенных источников, что создаст угрозу безопасности их компьютерам.

При этом JavaScript, помимо прочего, уже имеет ряд сред разработки, например, его поддержка существует в популярной среде разработки Eclipse.

Приведем набор особенностей языка и дополнительных библиотек, упрощающих процесс создания генераторов, которые включены в разработанный язык.

1.Отсутствие в коде лишних, повторяющихся инструкций.

Например, отсутствуют команды на открытие и закрытие файлов для вывода в них условий и ответов, инициализация генератора случайных чисел и т. п.;

2.Поддержка максимального набора возможностей форматирования текста;

3.Разнообразные способы доступа к случайным числам;

4.Связь с системами символьных и численных вычислений;

5.Создание и использование собственных и сторонних библиотек;

6.Поддержка повторения генерации при неудачных значениях параметров;

7.Наличие средств отладки, как минимум, возможность вывода отладочных сообщений.

Поясним пункты подробнее.

Во-первых, программист не должен заботится о том, что может сделать за него среда исполнения.

Например, если создаваемое условие необходимо сохранить в файл, именно внешняя среда должна его открыть и потом "не забыть" закрыть.

При таком понимании разрабатываемый язык является встроенным, т. е. он не существует отдельно сам по себе, а работает только в рамках окружающей среды..

Во-вторых, существенной особенностью языка являются возможности форматирования текста.

Действительно, условие и ответ являются текстом, и создание текста должно происходить максимально удобно.

Пусть частью условия является “У Василия было 5 яблок”. Предположим, что в переменной *apples* содержится количество яблок, которое надо выписать в условие. Стандартное для JavaScript выражение выглядит так:

```
'У Василия было ' + apples + ' яблок'.
```

Но более удобно было бы написать это в php или perl стиле:

```
'У Василия было %apples яблок'.
```

Вместо стандартного в данной ситуации символа $ используется символ %, потому что создаваемые условия имеют формат TeX. В TeX символ $ используется для включения или выключения режима ввода формул. Символ % используется в TeX для комментариев, и этой его возможностью можно пожертвовать.

Помимо прочего вспомним, что условие – это текст на естественном языке, который имеет сложные для программирования нерегулярности.

Например, если бы яблок было двадцать одно, то написать следовало бы “У Василия было 21 яблок**о**”. Пример оформления этого в коде:

```
'У Василия было %{r(apples, "яблок|о|а|")}'.
```

В-третьих, особенности доступа к случайным числам, которые очень часто используются при программировании генераторов.

Стандартная библиотека инструмента разработки генераторов содержит большой набор функций случайных чисел.

Пример полезной функции – coin(), которая выдает истину или ложь с вероятностью половина. Если указать необязательный параметр, можно изменить вероятность выпадения истины.

В-четвёртых, связь с системой символьных вычислений Maxima является одной из самых важных особенностей.

Несмотря на то, что Maxima не является наиболее функциональной системой компьютерной алгебры и по возможностям отстает от лидеров отрасли Mathematica и Maple, она единственная из них бесплатна, (распространяется под лицензией GPL).

При программировании генераторов нужны системы компьютерной алгебры, так как они уже содержат реализацию большого числа алгоритмов, (способны брать интегралы, производные, считать наибольший общий делитель), и способны форматировать результаты вычислений.

Проблему у программиста вызывает даже форматирование многочленов.

Допустим, мы имеем три переменных, содержащих коэффициенты квадратного трехчлена *a*, *b* и *c*. Если написать выражение '%a x^2 + %b x + %c', то некорректный вывод произойдет в случае отрицательных коэффициентов, нулях и единицах.
В-пятых, в данный инструментарий включена возможность создания и подключения дополнительных библиотек.

Примером полезной при создании генераторов библиотеки может являться библиотека геометрических объектов, позволяющая совершать над ними геометрические действия, преобразования, а также выводить их в виде текста для использования в условиях.
В-шестых, обеспечена поддержка повторения генерации при неудачных значениях параметров.

Вспомним устройство кода из примера параметрического генератора. Последним разделом является список условий на параметры. При генерации, если эти условия не выполняются, значения для параметров выбираются заново.

При программировании генераторов такой подход является практически стандартом, программа генератора должна исполняться, пока возникают проблемы со значениями параметров, из-за которых либо вообще не удается создать условие, либо условие оказывается слишком простым, либо слишком сложным. Поэтому операция перезапуска генератора удобна в генераторах, и такой операцией в инструменте создания генераторов является функция assert(). Если записанное в параметрах условие оказывается неверным, генератор запускается сначала.
В-седьмых, возможность вывода на экран отладочной информации.

Если программист не хочет настраивать настоящий отладчик JavaScript, чтобы пошагово следить за ходом исполнения программы, он может пользоваться отладочным выводом. Функции print() и println() добавлены в инструмент создания генераторов для печати строки текста в отладочную консоль.

Обобщим все вышесказанное на конкретном примере кода генератора, создающего задачу на поиск корней квадратного уравнения.

```
statement = 'Решите квадратное уравнение $%m=0$';
answer = '$x_1=%x1, x_2=%x2$';
var x1 = rnd(-5, 5);
var x2 = rnd(-5, 5);
assert (x1 != x2);
var a = rnd(-5,5);
var m = tex('%a * x^2 - %a*(%x1+%x2)*x + % a*x1*x2');
```

В первой и второй строке заполняются переменные, содержащие текст условия и ответа. Из-за того, что условие и ответ должны иметь формат TeX, необходимо использовать символ $ для обозначения формул. Внутри переменных statement и answer есть ссылки на параметры, они начинаются на символ %. После окончания исполнения программы все параметры в переменных statement и answer будут заменены на значения соответствующих переменных. Никаких дополнительных команд на подстановку значений параметров писать не нужно.

Переменные с условием и ответом могли бы быть заполнены и в конце программы, но хорошим тоном считается располагать условие и ответ в начале кода, чтобы при просмотре кода генератора сразу было ясно, какую задачу он создает.

Далее по коду создаются переменные *x*1 и *x*2, корни уравнения. Они, как только что было сказано, будут подставлены в текст ответа. Функция rnd() выбирает случайное число в заданном диапазоне, исключая нуль. Если ноль также нужно выбирать, используется функция rnd0(). Операция assert() проверяет, что корни разные и не равны нулю. Будем считать, что этого условия достаточно, чтобы получать условия одинаковой сложности.

Последняя строка создает переменную, в которой хранится строка с многочленом. Функция tex() вначале заполняет значения параметров внутри строки,

т. е. все выражения %a, %x1, %x2 будут заменены на соответствующие числа. Далее она посылает выражение Maxima. Maxima упрощает выражение, вычисляя все коэффициенты, убирая их по необходимости, если они равны 0 и 1. Далее Maxima преобразует выражение в формат TeX и возвращает в качестве результата функции.

Заключение

Разработка web-сайта “База генерируемых задач” связана с процессом наполнения сайта задачами, для структурирования которых предложена технология *фолксономии*. Для авторов, добавляющих в общую базу свои собственные задачи, предложен ряд мер для решения проблем, возникающих при использовании тегов.

В качестве формата для представления на сайте генерируемых задач выбран формат задач системы проведения удаленных соревнований DCES. Он удобен тем, что допускает подготовку задач вручную, т.к. для текстовых файлов с условиями задач и ответов к ним имеется возможность размещения их в архивах.

На данный момент в рамках сайта реализованы два метода создания генераторов задач, позволяющие избежать этой ручной работы.

Первый предназначен для пользователей и предполагает создание выбирающего генератора ручным вводом нескольких условий и ответов к задачам с небольшим количеством вариантов условий, в том числе и для создания билетов к экзамену.

Второй рассчитан на программистов, – это инструмент создания генераторов на языке JavaScript, для которого доступны функции стандартной библиотеки, упрощающие создание генераторов, обеспечен доступ к системе компьютерной алгебры Maxima, осуществляющей вычисления, помогающей упрощать выражения и форматирование их для вывода на печать.

Литература

1. Посов И.А. Смирнов И. Б. Интернет-сервис для хранения базы генерируемых задач по математике // Современное образование: содержание, технологии, качество: материалы международного форума (21-22 апреля, СПб), – СПбГЭТУ 2010, – Т. 2. – С. 84-85
2. Степанов А.В. Система компьютерной генерации заданий по математике // Компьютерные инструменты в образовании, – 2000. – № 3/4, – С. 28-31.
3. Интернет-проект “Задачи”, URL: http://problems.ru (дата обращения: 15.07.2010)
4. Левинская М.А, Автоматизированная генерация заданий по математике для контроля знаний учащихся // Educational Technology & Society . – 2002. – 5(4) – С. 214-221.
5. Посов И. А. Автоматическая генерация задач // Компьютерные инструменты в школе, – 2007. – № 1, – С. 54-62
6. Посов И.А., Рукшин С.Е. Архитектура системы проведения удаленных соревнований и организации работы с математическими задачами // Научно-технические ведомости СПбГПУ, – 2010. – № 3(101), – С. 49-53.
7. Dmitriev S. Language Oriented Programming: The Next Progrmming Paradigm // JetBrains, ноябрь 2004. URL: http://www.jetbrains.com/mps/docs/Language_Oriented_Programming.pdf (дата обращения: 15.07.2010)