УДК 004.9:378.146

О разработке многовариантных тестовых заданий с помощью JavaScript

Вильданов Алмаз Нафкатович
Нефтекамский филиал ФГБОУ ВО «Башкирский государственный университет»
к.ф.-м.н., доцент

Аннотация
Статья посвящена разработке инструментов для генерации многовариантных учебных заданий. Многовариантные задания решают проблему списывания и помогают лучше нацелить студента на обучаемость. Преподаватель может давать многовариантные задания обучающимся для тренировки и предварительной самопроверки, будучи уверенным в том, что на экзамене у студента будут пусть похожие, но другие задания. Приводятся фрагменты кода на языке JavaScript, иллюстрирующие некоторые особенные моменты разработки многовариантных тестовых вопросов. В работе указываются дополнительные инструментальные средства и библиотеки, которые могут оказаться полезными при создании таких тестовых задач. Результаты, изложенные в статье, могут быть использованы преподавателями для повышения качества учебных заданий.
Ключевые слова: генерация учебных заданий, методы генерации учебных заданий, многовариантная задача, редакторы генераторов, JavaScript

Development of multivariante test tasks using JavaScript

Vildanov Almaz Nafkatovich
Neftekamsk branch of Bashkir State University
Candidate of physico-mathematical sciences, associate professor

Abstract
The article deals with the development of tools for generating multivariante learning tasks. Multivariante tasks solve the problem of copying and help to better target the student to learnability. The teacher can give multivariante assignments to the trainees for training and preliminary self-examination, being sure that the exam will give the student alike, but other tasks. The code snippets are presented in the JavaScript language, illustrating some special moments in the development of multivariate test questions. The work indicates additional tools and libraries that can be useful in creating such test tasks. The results stated in the article can be used by teachers to improve the quality of study assignments.
Keywords: educational problems generation, methods of educational problems generation, multivariant problem, generators editors, JavaScript

Введение. Проверка и оценка успеваемости студентов занимает важную часть учебного процесса. Это может быть как «внутренняя» проверка – разнообразные зачеты, экзамены (в рамках промежуточной и итоговой аттестации) – так и внешняя, независимая оценка уровня образовательных достижений студентов. К внешним оценкам можно отнести, например, Федеральный Интернет-экзамен в сфере профессионального образования (ФЭПО), нацеленный на проверку результатов обучения студентов в рамках требований ФГОС ВО, Федеральный интернет-экзамен для выпускников бакалавриата (ФИЭБ), преподносимый как внешняя независимая сертификация выпускников бакалавриата, и т.д. Результаты ФЭПО обучающихся вуза могут учитываться при процедуре его государственной аккредитации (что сегодня актуально для многих вузов).

Тестирование полезно еще и тем, что может помочь студенту достичь понимания того, что конкретно от него требуется, какие знания он должен усвоить. Понятно, что чем больше вопросов в тесте, тем лучше и качественнее будет проверка знаний учащихся. Также желательно, чтобы при тестировании разным студентам попадались отличающиеся вопросы. Но тогда, например, для составления качественного теста из десяти вопросов, если на каждый вопрос придумывать до пяти его разновидностей, нужно подготовить до пятидесяти вопросов!

Очевидно, что приготовление такого теста становится очень трудоемким. Поэтому, несомненно, актуален вопрос создания инструментов для автоматической генерации многовариантных учебных заданий [1]. Многовариантные задания имеют несколько вариантов условия и могут использоваться, например, преподавателями во время контрольных работ для защиты от «списывания». В данной статье рассматривается использование возможностей браузерного языка JavaScript для создания таких многовариантных тестовых заданий.

Учащиеся могут использовать многовариантные задания и для тренировок в решении задач по определенной теме. Преподаватель может безбоязненно открыть обучающимся доступ к тестам для предварительной самопроверки, так как может быть уверен, что на экзамене у студента будут все равно пусть похожие, но другие задания.

Разработка многовариантных задач на знание языков программирования. Рассмотрим пример вопроса на знание функций работы со строками (язык Pascal):

```
Copy('technology', 7, 3) = ?
```

Правильный ответ: `'log'`. Понятно, что узнав ответ один раз, студент может его запомнить (или записать). Хорошо, если он узнал ответ самостоятельно. А если вспомнил или подсмотрел у соседа? Было бы хорошо, если во второй раз при обращении к тому же тесту сам вопрос выглядел бы несколько иначе, например, так:

```
Copy('catalogue', 1, 5) = ?
```

В таком случае студенту, чтобы ответить правильно, нужно будет приложить усилие и разобраться, как же все-таки работает метод Copy. Цель достигнута!

Как можно реализовать многовариантный вопрос такого типа? Нужно, во-первых, создать словарь слов, а во-вторых, написать аналог функции Copy на JavaScript и вызывать ее со случайными параметрами.

Итак, создаем словарь:

```
var dictionary = new Array( dictionaryLength );
     dictionary[0] = "academy";
     dictionary[1] = "atomic";
     dictionary[2] = "balance";
…
```

Чем больше будет слов в словаре, тем лучше. Выбирая случайный номер, будем легко менять слово в задании. Поскольку параметры вопроса будут меняться, ответ на вопрос также будет каждый раз другим. Поэтому нужно разработать подпрограмму для вычисления правильного ответа. Для этого создаем на JavaScript аналог функции Copy:

```
function copy( source, index, count ) {
     return source.substr( index - 1, count );
}
```

Для создания многовариантных заданий нам понадобится функция, которая генерирует целые случайные значения параметров будущего задания, тем самым делая его практически уникальным. Стандартная функция Javascript Math.random() генерирует случайные числа от нуля до единицы. Построим на ее основе функцию для случайного целого числа в указанном диапазоне (от min до max) [2]:

```
function random( min, max ) {
     var range = max - min + 1;
     return Math.floor( Math.random() * range ) + min;
}
```

Теперь создадим функцию для генерации вопроса со случайными параметрами, и отображения его в окне браузера. Вопрос хранится в переменной question, а правильный ответ на него – в переменной answer:

```
function test1() {
// выбираем случайное слово из словаря
```

```
        k = random( 0, dictionaryLength - 1 );
        word =  dictionary[ k ];
    // генерируем случайные параметры вопроса
        index = random( 1, 3 );
        count = random( 2, 4 );
    // вычисляем ответ на вопрос
        answer = copy( word, index, count );
    // формулируем вопрос со случайными параметрами
        question = "Copy('" +  word + "'," + String(index)
         + "," + String(count) + ") = ? ";
    // выводим на экран вопрос
        document.getElementById('query').innerHTML =
                                                question;
    }
```

Теперь вопрос, каждый раз при обновлении веб-страницы, уникален. Параметры `index`, `count` берутся случайно, не слишком большие, так как желательно, чтобы ответы на вопросы были не пустыми строками. Поэтому и слова в словаре должны иметь длину, не меньшую семи.

Используем данную методологию для программирования задач на нахождение результатов программного кода. Рассмотрим пример (такая задача предлагается в ФЭПО по дисциплине «Языки программирования», направление «Информационная безопасность») (рис. 1):

В результате выполнения следующей программы:

```
var n,k: word;
begin
  n:=1818;
  k:=0;
  repeat
    k:=10*k+n mod 10;
    n:=n div 10;
  until n<>0;
write(k);
end.
```

будет выведено…

Рисунок 1 – Найти результат выполнения программы на Pascal

Подобную задачу также можно сгенерировать так, чтобы каждый раз она была «уникальна». Уникальности можно добиться, меняя начальную переменную `n`. В `JavaScript` отсутствует конструкция типа `repeat until`. Решим подобную задачу с помощью конструкции `while`. Следующий код выводит на экран задачу, подобную той, что изображена на рис. 1, и вычисляет результат:

```
function test2() {
// генерируем случайный параметр вопроса
     var n = 1800 + random( 12, 99 ), k = 0;
// формулируем вопрос со случайным параметром n
     question = "<pre>В результате выполнения
     следующей программы: ' + '<br>' +
     ' var n, k: word;' + '<br>' +
     ...
     + 'будет выведено… <br>';
// выводим на экран вопрос
  document.getElementById('query').innerHTML = question;
// и вычисляем ответ
     k = ( 10 * k + n ) % 10;
     n = Math.floor( n / 10 );
     while ( n == 0 ) {
          k = ( 10 * k + n ) % 10;
          n = Math.floor( n / 10 );
     }
     answer = k;
}
```

Рассмотрим теперь пример создания задач на решение блок-схем (подобная задача предлагается в ФЭПО по дисциплине «Языки программирования» на разных направлениях) (рис. 2):

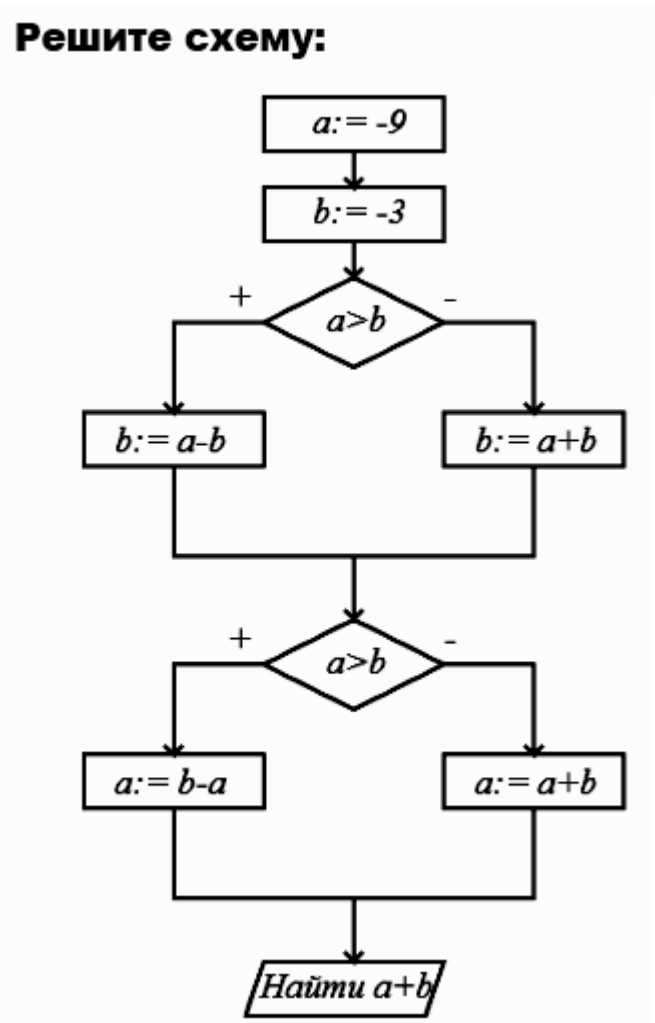


Рисунок 2 – Найти результат блок-схемы

Для создания такой многовариантной задачи можно генерировать случайные начальные значения переменных `a` и `b`, и вычислять соответствующий результат. Для рисования схемы отлично подходит элемент `Canvas`. `Canvas` предназначен для создания растрового двухмерного изображения при помощи языка `JavaScript` и используется для построения графики, простой анимации и игр в браузерах [4]. Создадим функцию:

```
function test3() {
// генерируем случайные параметры блок-схемы
     var a = random( -10, 12 );
     var b = random( -10, 12 );
// Рисуем блок-схему с произвольными параметрами
     var example = document.getElementById("example"),
     ctx = example.getContext('2d');
// сначала рисуем прямоугольник с оператором внутри
     ctx.strokeRect( 400, 10, 100, 30 );
     ctx.font = 'italic 20px none';
     ctx.textBaseline = 'top';
```

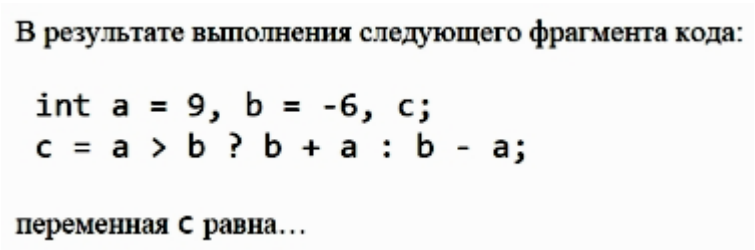
```
        ctx.fillText ( 'a:= '+ a, 427, 13 );
    ... // и т.д.
```

Далее вычисляем ответ:

```
        if ( a > b ) {
            b = a - b;
        }
        else b = a + b;
        if ( a > b ) {
            a = b - a;
        }
        else a = a + b;
        answer = (a + b);
        document.getElementById('query').innerHTML =
                                            'Решите блок-схему';
    }
```

Данный код поможет сгенерировать задачу, похожую на ту, что изображена на рис. 2, но практически всегда с разными начальными значениями (a, значит, и с разными ответами).

Во многих случаях задача на проверку СИ-подобных языков выглядит гораздо проще, так как `JavaScript` относится к этому же семейству языков. Рассмотрим пример на знание тернарного оператора в `C#` (рис. 3).


В результате выполнения следующего фрагмента кода:

```
int a = 9, b = -6, c;
c = a > b ? b + a : b - a;
```

переменная c равна…

Рисунок 3 – Задача на знание тернарного оператора

В данном случае реализация многовариантного вопроса становится практически повторением заданного фрагмента кода:

```
function test4() {
    var a = random( -9, 9 ), b = random( -9, 9 ), c;
        c = a > b ? b + a : b - a;
        answer = c;
    question = 'В результате выполнения' +
    'следующего фрагмента кода:' + ' <br><br>  <pre>' +
    ' int a = ' + a + ', b = ' + b + ', c; ' + '<br>' +
    ' c = a > b ? b + a : b - a; ' + '</pre><br>' +
    'переменная <code>c</code> равна… <br>';
```

```
        document.getElementById('query').innerHTML =
                                                    question;
    }
```

Вкупе с языком разметки HTML и каскадной таблицей стилей CSS можно создавать многовариантные задачи для самых разных разделов высшей математики. Следующее приложение на `JavaScript` проверяет знание функций математической логики [5]. Требуется найти таблицу истинности логической функции, выбрать правильные варианты для совершенной дизъюнктивной нормальной формы (СДНФ) и совершенной конъюнктивной нормальной формы (СКНФ) (рис. 4). Операнды и знаки операций логической функции генерируются случайным образом, тем самым образуя многовариантные задачи.

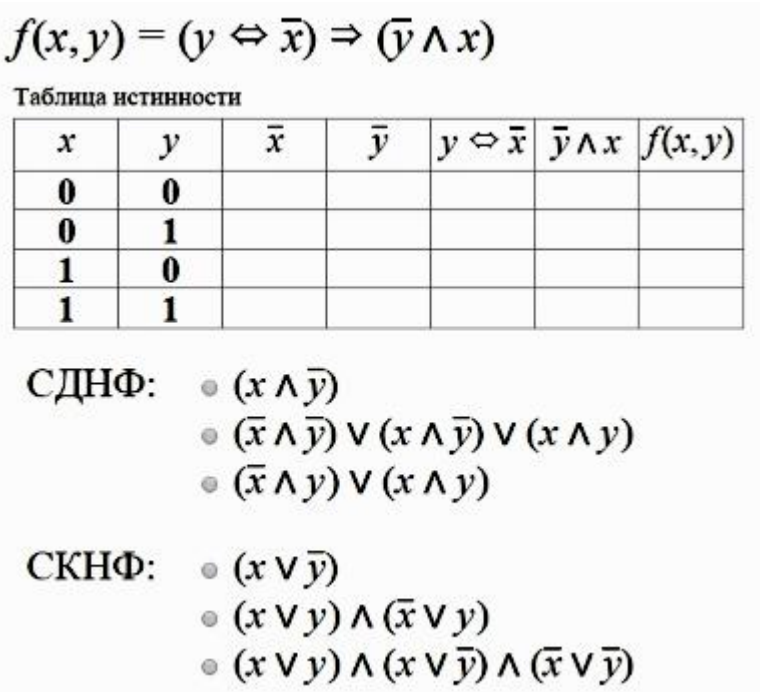
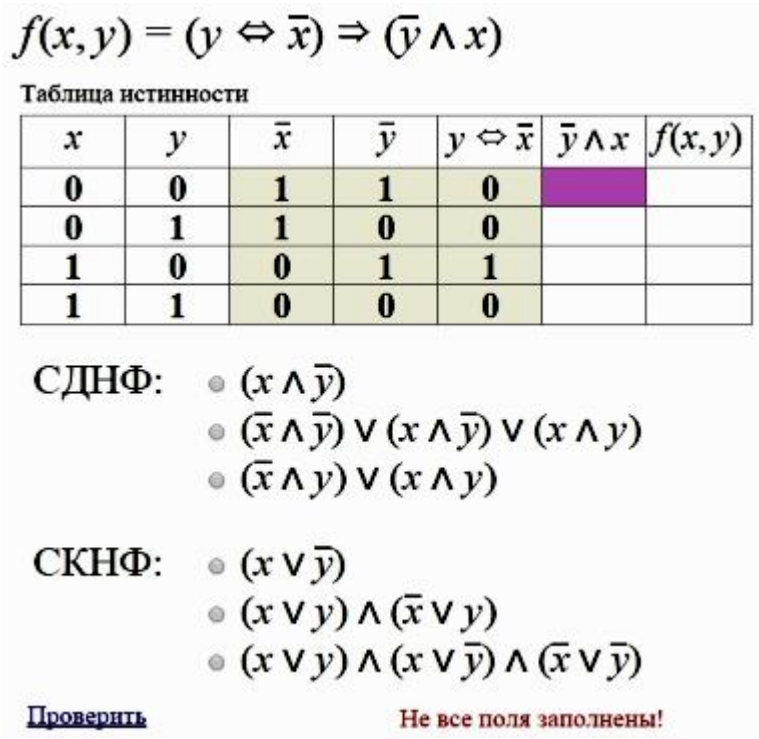


Рисунок 4 – Задание на знание логических функций

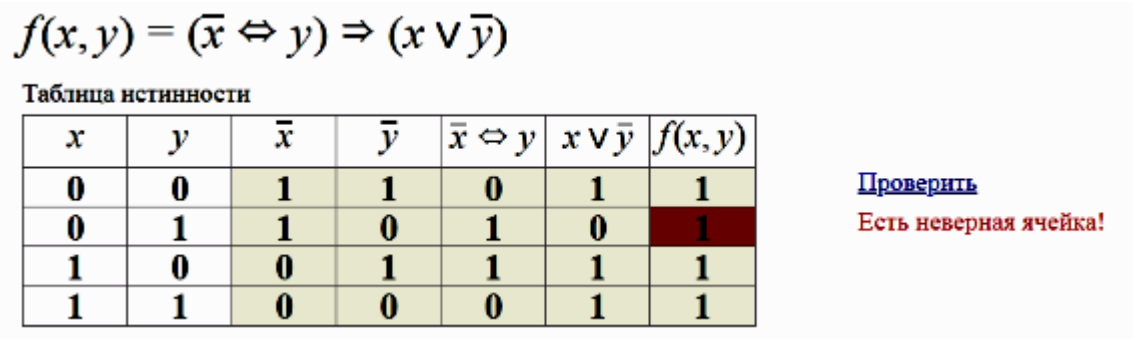
С помощью обработчиков событий на `JavaScript` создается дружественный интерфейс. Студент может легко набрать ответ лишь при помощи мышки, щелкая по соответствующим полям. При щелчке значения последовательно меняются на 0, 1, или пусто. Если студент заполнил не все поля, то при проверке выводится соответствующее предупреждение (рис. 5).



x	y	$\bar{x}$	$\bar{y}$	$y \Leftrightarrow \bar{x}$	$\bar{y} \wedge x$	$f(x,y)$
0	0	1	1	0		
0	1	1	0	0		
1	0	0	1	1		
1	1	0	0	0		

Рисунок 5 – Ошибка заполнения полей

Можно запрограммировать подсказку в случае неверного ответа (но не более одной в ходе выполнения задания) (рис. 6):



x	y	$\bar{x}$	$\bar{y}$	$\bar{x} \Leftrightarrow y$	$x \vee \bar{y}$	$f(x,y)$
0	0	1	1	0	1	1
0	1	1	0	1	0	1
1	0	0	1	1	1	1
1	1	0	0	0	1	1

Рисунок 6

Для проверки правильности программа должна уметь вычислять значения логической функции $f(x, y) = (y \Leftrightarrow \bar{x}) \Rightarrow (\bar{y} \wedge x)$. Для этих целей можно использовать парсер [6], который позволяет по данному тестовому представлению функции (формулы) вычислять ее числовые значения при различных значениях аргумента.

Логическая функция, как уже было сказано, генерируется случайно. При обновлении страницы будет уже другая задача (рис. 6):

$f(x,y) = (x \vee y) \Rightarrow (\bar{y} \Leftrightarrow \bar{x})$

Таблица истинности

x	y	$\bar{x}$	$\bar{y}$	$x \vee y$	$\bar{y} \Leftrightarrow \bar{x}$	$f(x,y)$
0	**0**					
0	**1**					
1	**0**					
1	**1**					

СДНФ:
- $(\bar{x} \wedge \bar{y}) \vee (\bar{x} \wedge y)$
- $(x \wedge \bar{y}) \vee (x \wedge y)$
- $(\bar{x} \wedge \bar{y}) \vee (x \wedge y)$

СКНФ:
- $(\bar{x} \vee y) \wedge (\bar{x} \vee \bar{y})$
- $(x \vee y) \wedge (x \vee \bar{y})$
- $(x \vee \bar{y}) \wedge (\bar{x} \vee y)$

Рисунок 6 – Окно приложения после обновления страницы

Наконец, рассмотрим еще пример генерации многовариантных задач по теории графов (рис.7):

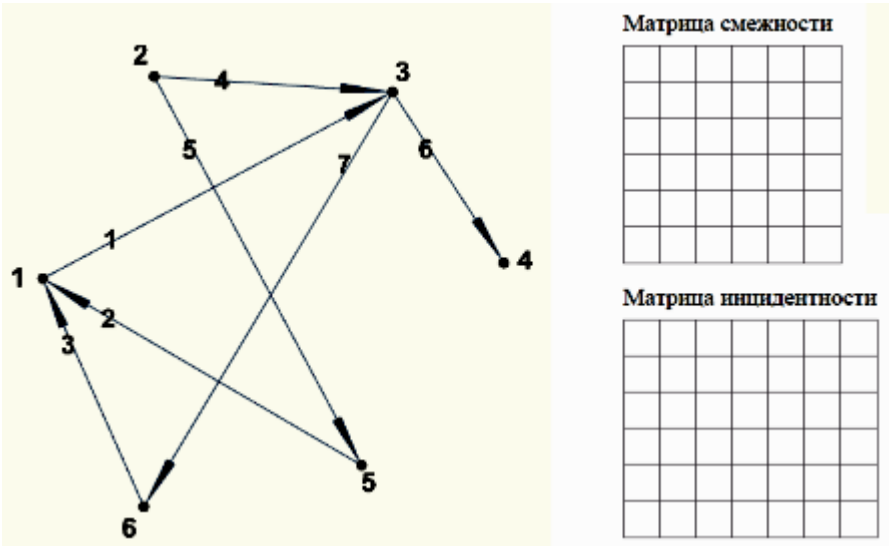


Рисунок 7 – Задача на теорию графов

Достаточно сгенерировать случайную последовательность ребер, вычислить соответствующие матрицы смежности и инцидентности, и предложить учащемуся самому вычислить их. Для изображения векторов удобно использовать готовые классы JavaScript-библиотеки Three.js [7].

Заключение. С помощью браузерного языка JavaScript преподаватель имеет возможность создавать тесты с уникальными неповторяющимися вопросами по языкам программирования и высшей математике, тем самым повышая качество контроля знаний учащихся по этим дисциплинам.

В приведенных в статье примерах можно разрабатывать многовариантные задания как «открытого» типа (свободного изложения), когда испытуемый должен предложить свой ответ и вписать его в указанное поле. Вполне возможно и создание заданий «закрытого» типа, т.е. заданий, сопровождающихся готовыми вариантами ответов, из которых необходимо выбрать один или несколько правильных. Но при этом следует учитывать ряд моментов, таких как правдоподобность неправильных вариантов ответов, их различие между собой и т.д.

Ознакомиться с разработанными тестами и их исходными кодами можно по адресам https://gevaraweb.github.io/iatests/cisharp, https://gevaraweb.github.io/iatests/graph.

Библиографический список

1. Посов И. А. Обзор генераторов и методов генерации учебных заданий // Образовательные технологии и общество. 2014. Т.17. №4. С. 3-10.
2. Кантор И. Центральный Javascript-ресурс. Учебник с примерами скриптов. Форум. Книги и многое другое [Электронный ресурс] – Режим доступа: http://javascript.ru, свободный (дата обращения: 01.02.18).
3. MathJax [Электронный ресурс] / MathJax.org – Режим доступа: http://www.mathjax.org, свободный (дата обращения: 01.03.18).
4. Вильданов А.Н., Шафеева Е.П. Построение объектов двумерной графики в HTML5 с помощью Canvas // Информатика в школе. 2015. № 4 (107). С. 24-28.
5. Вильданов А.Н., Шафеева Е.П. Браузерное приложение «Генератор теста с уникальными вопросами по дискретной математике» на JavaScript и WebGL // Хроники объединенного фонда электронных ресурсов «Наука и образование». 2014. Т. 1. № 12 (67). С. 129.
6. Matthew Crumley. JavaScript Expression Evaluator [Электронный ресурс] – Режим доступа: https://silentmatt.com/javascript-expression-evaluator, свободный (дата обращения: 01.05.18).
7. Вильданов А.Н. 3D-моделирование на WebGL с помощью библиотеки Three.js: учебное пособие. Уфа : РИЦ БашГУ, 2014. 114 с.