

МИНОБРНАУКИ РОССИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧЕРЕЖДЕНИЕ  
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ  
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет компьютерных наук

Кафедра цифровых технологий

Программная реализация (на языке JavaScript) алгоритмов  
генерации ФОС ЕГЭ по геометрии в 2025 году

Курсовая работа

Направление 020301 Математика и компьютерные науки

Зав.кафедрой \_\_\_\_\_ д.физ.-мат.н., проф. С.Д. Кургалин

Обучающийся \_\_\_\_\_ Е.Ю. Колесникова

Руководитель \_\_\_\_\_ к.физ.-мат.н, доц. Н.П. Стадная

Воронеж 2025

# Содержание

<b>Введение</b>	<b>3</b>
<b>1 Планиметрия</b>	<b>5</b>
1.1 Вспомогательные функции . . . . .	5
1.1.1 Функции для работы с массивами . . . . .	5
1.1.2 Функции для работы с числами . . . . .	6
1.1.3 Функции для работы с canvas . . . . .	7
1.1.4 Элементы декларативного программирования . . . . .	11
1.2 Этапы разработки шаблона со вспомогательным чертежом по теме «Планиметрия» . . . . .	12
<b>2 Генерация текстовых задач</b>	<b>18</b>
2.1 Декорации . . . . .	18
2.2 Склонения существительных . . . . .	19
2.3 Проверка условий (генерация утверждений) . . . . .	21
2.4 Разнообразие в текстовых задачах . . . . .	24
<b>3 Задачи №4 ЕГЭ (преобразование выражений)</b>	<b>30</b>
3.1 Основные используемые команды . . . . .	30
3.2 Адаптирование команд . . . . .	31
3.3 Обратные задачи . . . . .	36
<b>4 Задачи №7 ОГЭ (координатная прямая)</b>	<b>38</b>
4.1 разработка функции . . . . .	38
4.2 Назначение функции . . . . .	38
4.3 Интерфейс функции . . . . .	39
4.4 Алгоритм работы . . . . .	40
4.5 Пример использования . . . . .	45

<b>5</b>	<b>Задачи №7 ОГЭ (сравнение чисел)</b>	<b>46</b>
<b>6</b>	<b>Стереометрия</b>	<b>63</b>
6.1	Разработка библиотек с помощью Gpt-Chat . . . . .	63
6.2	Применение объектно-ориентированного программирования для разработки шаблонов . . . . .	67
6.3	Вспомогательные функции . . . . .	69
6.3.1	Функции для работы с координатами . . . . .	69
6.3.2	Функции для работы с canvas . . . . .	69
6.4	Этапы разработки шаблоны с вспомогательным чертежом по теме «Стереометрия» . . . . .	70
	<b>Заключение</b>	<b>79</b>
	<b>Приложение</b>	<b>80</b>

# Введение

Единый государственный экзамен (ЕГЭ) — централизованно проводимый в Российской Федерации экзамен в средних учебных заведениях — школах, лицеях и гимназиях, форма проведения ГИА (Государственной Итоговой Аттестации) по образовательным программам среднего общего образования. Служит одновременно выпускным экзаменом из школы и вступительным экзаменом в вузы. Но за время обучения в 9, 10 и 11 классе при подготовке к ОГЭ и ЕГЭ школьники сталкиваются с дефицитом заданий по определённым категориям. Так, за последние 5 лет в список заданий ЕГЭ были добавлены новые задания под номером 1 по теме «Округление с недостатком» и «Округление с избытком», так же задания под номером 15 «проценты и округление», 21 задания «Текстовые задачи», количество которых для прорешивания было очень мало. Ко всему прочему в задании номер 7 по теме «Числовые неравенства, координатная прямая - числа на прямой» банк заданий расходуется при подготовке с невероятной скоростью: так как это преимущественно графические задания, решение их занимает менее минуты, а их составление вручную занимает несоразмерно много времени. ОГЭ и ЕГЭ является относительно неизменяемым экзаменом, поэтому все материалы, которые уже были выложены в открытый доступ, имеют полные решения, что приводит к списыванию учениками.

При этом существуют задания со вспомогательным чертежом. Чаще всего для целого ряда заданий используется одна и та же иллюстрация, которая не всегда соответствует условиям задачи, а иногда отвлекает от решения. Проект «Час ЕГЭ» позволяет решить все эти проблемы.

«Час ЕГЭ» — компьютерный образовательный проект, разрабатываемый при математическом факультете ВГУ в рамках «OpenSource кластера» и предназначенный для помощи учащимся старших классов при подготовке к тестовой части единого государственного экзамена. Задания в «Час ЕГЭ» генерируются случайным образом по специализированным алгоритмам, называемых шаблонами,

каждый из которых охватывает множество вариантов соответствующей ему задачи. Для пользователей предназначены четыре оболочки (режима работы): «Случайное задание», «Тесты на печать», «Полный тест» и «Мини-интеграция». «Час ЕГЭ» является полностью открытым (код находится под лицензией GNU GPL 3.0) и бесплатным. В настоящее время в проекте полностью реализованы тесты по математике с кратким ответом (бывшая «часть В»). [?] Планируется с течением времени включить в проект тесты по другим предметам школьной программы.

Первую главу посвятим обзору шаблонов для номеров 21,15, 1 из ЕГЭ базовой. Во второй главе рассмотрим функцию добавленному для упрощения отрисовки прямых для 7 задания ОГЭ.

# 1. Планиметрия

В этой главе мы приводим вспомогательные функции и алгоритм написания шаблона по планиметрии. И рассказываем о элементах декларативного программирования в проекте.

## 1.1. Вспомогательные функции

### 1.1.1. Функции для работы с массивами

```
Array.prototype.permuteCyclic = function(repeat)
```

Возвращает массив после циклической перестановки. В листинге 3 в строке 38 функция используется для перестановки букв в названии параллелограмма, а в строке 40 — в названии трапеции.

```
1  let array = [1, 2, 3, 4, 5];
2
3  array.permuteCyclic(1);
4  // [5, 1, 2, 3, 4]
5
6  array.permuteCyclic(-2);
7  // [3, 4, 5, 1, 2]
8
9  array.permuteCyclic(0);
10 // [1, 2, 3, 4, 5]
11
```

```
Array.prototype.mt_coordinatesOfIntersectionOfTwoSegments
= function()
```

Возвращает координаты пересечения двух отрезков, задаваемых первыми парами точек из массива. Является вспомогательной для функции `arcBetweenSegments`.

```
1  let array = [{x:0,y:5},{x:-4,y:4},{x:1,y:10},{x:-3,y:6}];
2
```

```

3   array.mt_coordinatesOfIntersectionOfTwoSegments()
4   //{ x: -5.333333333333333, y: 3.666666666666667, status: false }
5   //Если status - false, отрезки не пересекаются, но прямые проходящие через
   них пересекаются в точке {x,y}
6
7   array = [{x:0,y:5},{x:-4,y:4},{x:1,y:1},{x:-3,y:6}];
8   array.mt_coordinatesOfIntersectionOfTwoSegments()
9   //{ x: -1.833333333333333, y: 4.541666666666667, status: true }
10  //Если status - true, отрезки пересекаются в точке {x,y}

```

`Array.prototype.shuffleJoin = function(separator)`

Перемешивает и соединяет массив с разделителем `separator`. `separator` по умолчанию пустая строка. Функция используется в листинге 4 в строке 74 для отображения условий задачи в случайном порядке.

```

1   let array = ['A', 'B', 'C', 'D',];
2   array.shuffleJoin();
3   //ADBC
4
5   array.shuffleJoin('; ');
6   //C; D; B; A

```

`Array.prototype.joinWithConjunction = function(separator)`

Соединяет массив запятыми и соединяет два последних элемента союзом «и».

```

1   let array = ['A', 'B', 'C', 'D',];
2
3   array.joinWithConjunction();
4   //A, B, C и D

```

### 1.1.2. Функции для работы с числами

`Number.prototype.perfectCubicMultiplier = function()`

Возвращает максимальный делитель данного числа, куб которого также является делителем данного числа.

```

1    let number = 81;
2
3    number.perfectCubicMultiplier()
4    //3
5
6    number = 36;
7    number.perfectCubicMultiplier()
8    //1
9
10   number = -27;
11   number.perfectCubicMultiplier()
12   //3

```

`Number.prototype.texcbirt = function(p1, p2)`

TeX-представление кубического корня из данного числа.

Если данное число - полный куб, то корень из числа.

Если p1, то из-под корня будут вынесены возможные множители.

Если p1, p2 и из-под корня выносятся единица, то она будет опущена

Радиусы трёх шаров равны  $\sqrt[3]{3}$ ,  $\sqrt[3]{6}$ ,  $\sqrt[3]{18}$ . Найдите радиус шара, объем которого равен сумме их объемов.

### 1.1.3. Функции для работы с canvas

`CanvasRenderingContext2D.prototype.drawSection = function(vertex, fillStyle)`

Заполняет область цветом `fillStyle` по вершинам из массива `vertex`.

```

1    let paint1 = function(ctx) {
2        let h = 400;
3        let w = 400;
4        ctx.drawCoordinatePlane(w, h, {
5            hor: 1,
6            ver: 1

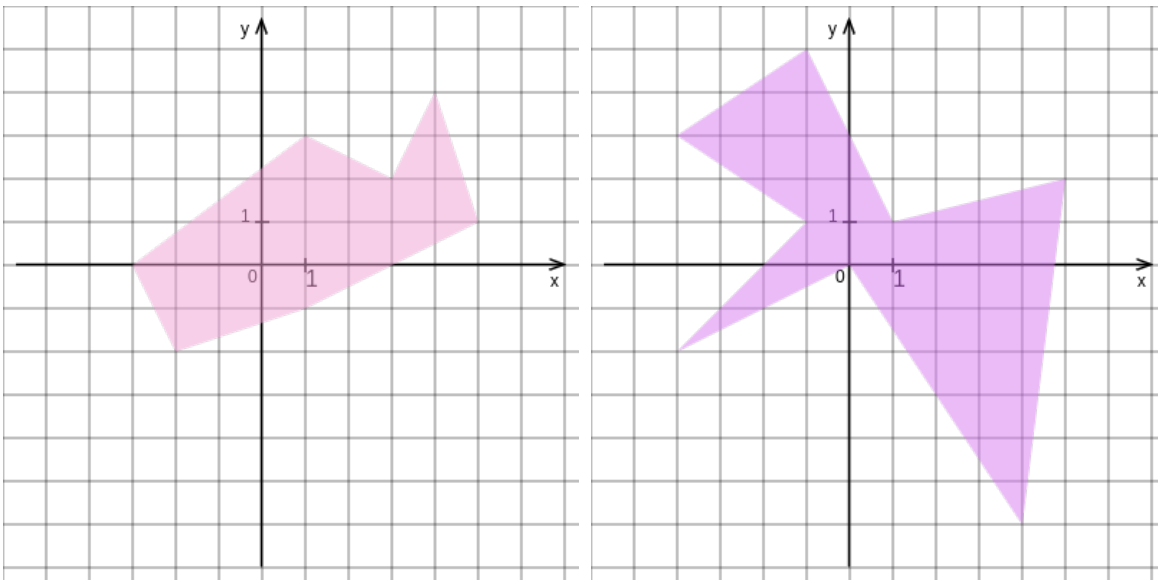
```



```

7      }, {
8          x1: '1',
9          y1: '1',
10         sh1: 16,
11     }, 30);
12     ctx.scale(30, -30);
13     ctx.drawSection([[1, 3], [-3, 0], [-2, -2], [1, -1], [5, 1], [4, 4], [3,
14         2]]);
15     ctx.drawSection([[-2, 0], [-1, 1], [-4, 3], [-1, 5], [1, 1], [5, 2], [4,
16         -6], [0, 0], [-4, -2],]);
17     };

```



`CanvasRenderingContext2D.prototype.drawLineAtAngle = function(x, y, angle, length)`

Рисует отрезок длины `length` под углом `angle` (в радианах). Пример использования можно найти в листинге 5 в строках 19 и 25 (применяется для отрисовки биссектрисы).

`CanvasRenderingContext2D.prototype.strokeInMiddleOfSegment = function(x1, y1, x2, y2, length, quantity)`

Ставит штрихи длины `length` на середине отрезка перпендикулярно ему. Функция используется в листинге 5 в строках 21-22 для обозначения равных по длине

сторон треугольника.

```
CanvasRenderingContext2D.prototype.markSegmentWithLetter  
= function(x, y, angle, letter, length, maxLength)
```

Вспомогательная функция для отрисовки текста около некоторого отрезка.

```
CanvasRenderingContext2D.prototype.signSegmentInMiddle  
= function(x1, y1, x2, y2, letter, length, maxLength)
```

Рисует строку `letter` на середине отрезка. В листинге 6 в строках 94 - 98 функция используется для отображения длин рёбер многогранника.

```
CanvasRenderingContext2D.prototype.arcBetweenSegments  
= function(coordinates, radius)
```

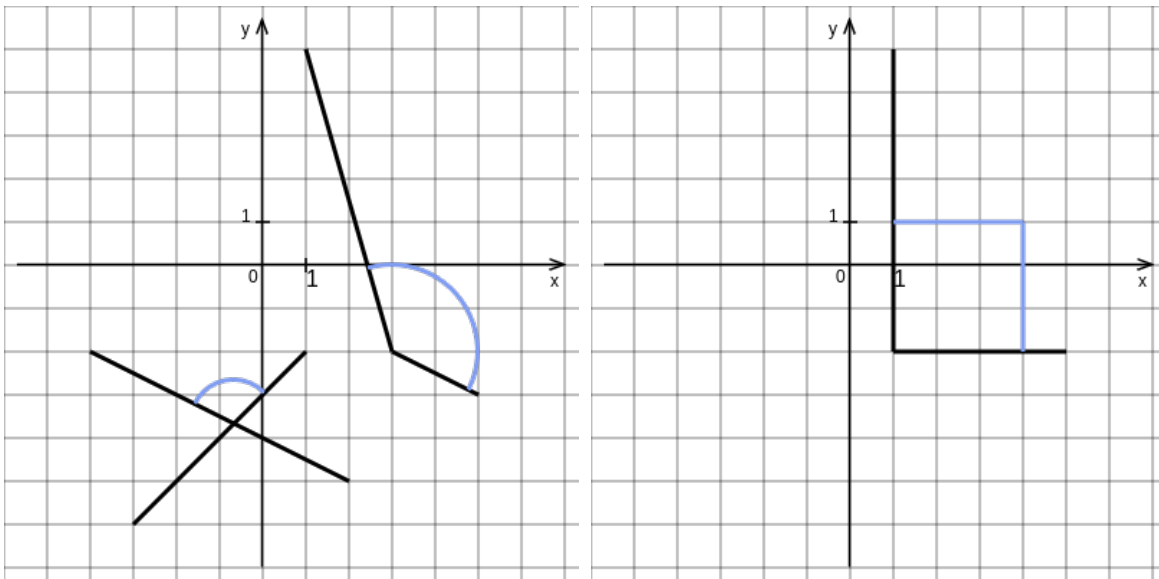
Рисует знак угла между двумя отрезками в месте их пересечения. `coordinates` - массив вида `[x1, y1, x2, y2]`.

```
1  let paint1 = function(ctx) {  
2      let h = 400;  
3      let w = 400;  
4      ctx.drawCoordinatePlane(w, h, {  
5          hor: 1,  
6          ver: 1  
7      }, {  
8          x1: '1',  
9          y1: '1',  
10         sh1: 16,  
11     }, 30);  
12     ctx.scale(30, -30);  
13  
14     ctx.lineWidth = 2 / 30;  
15     ctx.drawLine(1, 5, 3, -2);  
16     ctx.drawLine(3, -2, 5, -3);  
17     ctx.arcBetweenSegments([1, 5, 3, -2, 5, -3], 2);  
18  
19     ctx.drawLine(2, -5, -4, -2);  
20     ctx.drawLine(1, -2, -3, -6);
```

```

21     ctx.arcBetweenSegments([2, -5, -4, -2, -3, -6, 1, -2,], 1);
22
23     ctx.drawLine(1, 5, 1, -2);
24     ctx.drawLine(1, -2, 5, -2);
25     ctx.strokeStyle = om.secondaryBrandColors.iz();
26     ctx.arcBetweenSegments([1, 5, 1, -2, 5, -2], 3);
27
28     };

```



`CanvasRenderingContext2D.prototype.arcBetweenSegmentsCount`  
`= function(coordinates, radius, number, step)`

Рисует знак угла между двумя отрезками в месте их пересечения **number** раз с отступом **step**. В листинге 7 в строках 27 - 28 используется для обозначения двух равных углов.

`CanvasRenderingContext2D.prototype.drawEllipse`  
`= function(x, y, radiusX, radiusY, rotation, startAngle, endAngle, anticlockwise)`

Рисует эллипс.

`CanvasRenderingContext2D.prototype.drawArc`  
`= function(x, y, radius, startAngle, endAngle, anticlockwise)`

Рисует дугу.

#### 1.1.4. Элементы декларативного программирования

**Определение.** Декларативное программирование — парадигма программирования, в которой задается спецификация решения задачи, то есть описывается конечный результат, а не способ его достижения. [?]

Во время разработки шаблонов по теме «Графики функции» требовалось много раз генерировать коэффициенты функций через циклы `while` или `do...while`, пока они не начнут соответствовать заданным условиям (видимость графика, сливание его с осями, видимость целых точек). Это часто приводило к бесконечной работе шаблона, при этом сложно было определить, какое условие не выполняется.

Для этого было разработано окружение `retryWhileUndefined` для шаблонов, которое бы перезапускало их не более `maxIterations` раз, если одно из условий не удовлетворено.

```
function retryWhileUndefined(theFunction, maxIterations)
```

Но всё равно было тяжело определить, почему шаблон перезапускается. Для этого было разработано более совершенное окружение `retryWhileError`, которое не только могло бы ограничивать количество перезапусков, но и фиксировать, какие проверки не были пройдены и выводить их на экран (ошибки видны только для разработчика при отладке).

```
function retryWhileError(theFunction, maxIterations, maxCollectedErrors)
```

Для окружения были написаны функции-утверждения, которые имеют структуру: условие не выполнено - записать ошибку - перезапустить шаблон. Если максимальное количество повторений достигнуто, то вывести накопившиеся ошибки и количество их появлений.

```
function genAssert(condition, message)
```

Если условие `condition` ложно, то шаблон перезапускается.

```
function genAssertNonempty(array, message)
```

Если массив `array` пуст, то шаблон перезапускается.

```
function genAssertZ1000(number, message)
```

Если число `number` имеет более 3 знаков после запятой, то шаблон перезапускается.

```
function genAssertIrreducible(numerator, denominator, message)
```

Если дробь `numerator/denominator` сократима, то шаблон перезапускается.

```
function genAssertSaneDecomposition(number, maxFactor, message)
```

Если `number` число не раскладывается на простые множители, не более одного из которых превосходит `maxFactor`, то шаблон перезапускается.

## 1.2. Этапы разработки шаблона со вспомогательным чертежом по теме «Планиметрия»

Для примера возьмём задание №19416 [?].

**Задача №19376.** В треугольнике  $ABC$  известно, что  $AC = BC$ ,  $AB = 16$ ,  $AH$  — высота,  $BH = 4$ . Найдите косинус угла  $BAC$ .

Заготовка шаблона имеет вид.

```
1 (function () {
2   retryWhileError(function () {
3     NAinfo.requireApiVersion(0, 2);
4
5     let paint1 = function (ctx) {
6     };
7
8     NATask.setTask({
9       text: 'В треугольнике ABC AC=BC, AB=15, AH - высота, BH=5. Найдите
10      косинус ABC.',
11      answers: 0,
12      author: ['Суматохина Александра']
13    });
```

```

13   NATask.modifiers.addCanvasIllustration({
14     width: 400,
15     height: 400,
16     paint: paint1,
17   });
18 }, 1000);
19 })();

```

1. Начнём с отрисовки чертежа для задания. Отметим стороны треугольника так, чтобы он лежал в центре холста, а до краёв оставалось 10-20px. При отрисовке используем функцию `drawLine`. Добавим высоту

```

1  (function () {
2    retryWhileError(function () {
3      NAinfo.requireApiVersion(0, 2);
4
5      let paint1 = function (ctx) {
6
7        ctx.lineWidth = 2;
8        ctx.strokeStyle = om.secondaryBrandColors;
9
10       //стороны
11       ctx.drawLine(10, 370, 390, 370);
12       ctx.drawLine(10, 370, 180, 50);
13       ctx.drawLine(180, 50, 390, 370);
14       //высота
15       ctx.drawLine(280, 200, 10, 370);
16     };
17
18     NATask.setTask({
19       text: 'В треугольнике ABC AC=BC, AB=15, AH - высота, BH=5. Найдите
20       косинус ABC.',
21       answers: 0,
22       author: ['Суматохина Александра']
23     });

```

```

23     NATask.modifiers.addCanvasIllustration({
24         width: 400,
25         height: 400,
26         paint: paint1,
27     });
28 }, 1000);
29 })();

```

2. Добавим на рисунок штрихи, указывающие на равенство сторон и обозначение прямого угла при помощи функций `strokeInMiddleOfSegment` и `arcBetweenSe` соответственно. И подпишем вершины и точку пересечения высоты и основания. Добавим модификатор `NATask.modifiers.variativeABC(vertices)`, который заменяет все буквы в задании на случайные.

```

1  (function () {
2      retryWhileError(function () {
3          NAinfo.requireApiVersion(0, 2);
4
5          let vertices = ['A', 'B', 'C', 'H'];
6
7          let paint1 = function (ctx) {
8
9              ctx.lineWidth = 2;
10             ctx.strokeStyle = om.secondaryBrandColors;
11
12             //стороны
13             ctx.drawLine(10, 370, 390, 370);
14             ctx.drawLine(10, 370, 180, 50);
15             ctx.drawLine(180, 50, 390, 370);
16             //высота
17             ctx.drawLine(280, 200, 10, 370);
18
19             ctx.strokeInMiddleOfSegment(180, 50, 10, 370, 10);
20             ctx.strokeInMiddleOfSegment(180, 50, 390, 370, 10);

```

```

21
22     ctx.arcBetweenSegments([180, 50, 280, 200, 280, 200, 10, 370],
23     25);
24
25     ctx.font = "23px liberation_sans";
26     ctx.fillText(vertices[0], 10 - 5, 370 + 25);
27     ctx.fillText(vertices[1], 180, 50 - 10);
28     ctx.fillText(vertices[2], 390 - 10, 370 + 25);
29     ctx.fillText(vertices[3], ver2.x + 10, ver2.y);
30
31     NATask.setTask({
32         text: 'В треугольнике ABC AC=BC, AB=15, AH - высота, BH=5. Найдите
33         косинус ABC.',
34         answers: 0,
35         author: ['Сумагохина Александра']
36     });
37     NATask.modifiers.variativeABC(vertices);
38     NATask.modifiers.addCanvasIllustration({
39         width: 400,
40         height: 400,
41         paint: paint1,
42     });
43 }, 1000);
44 })();

```

3. Теперь добавим ответ в задание. Проверим при помощи `genAssertZ1000`, что ответ имеет не более трёх знаков после запятой (иначе шаблон запускается заново). Поместим все буквы и числа в `$...$`. Все условия из задачи преобразуем в массив и соединим случайным образом с помощью функции `shuffleJoin`.

```

1 (function () {
2     retryWhileError(function () {

```



```

3      NAinfo.requireApiVersion(0, 2);
4
5      let a = sl(2, 89);
6      let b = slKrome(a, 1, a - 1);
7      genAssertZ1000(b / a, 'Нецелый ответ');
8
9      let vertices = ['A', 'B', 'C', 'H'];
10
11     let paint1 = function (ctx) {
12
13         ctx.lineWidth = 2;
14         ctx.strokeStyle = om.secondaryBrandColors;
15
16         //стороны
17         ctx.drawLine(10, 370, 390, 370);
18         ctx.drawLine(10, 370, 180, 50);
19         ctx.drawLine(180, 50, 390, 370);
20         //высота
21         ctx.drawLine(280, 200, 10, 370);
22
23         ctx.strokeInMiddleOfSegment(180, 50, 10, 370, 10);
24         ctx.strokeInMiddleOfSegment(180, 50, 390, 370, 10);
25
26         ctx.arcBetweenSegments([180, 50, 280, 200, 280, 200, 10, 370],
27         25);
28
29         ctx.font = "23px liberation_sans";
30         ctx.fillText(vertices[0], 10 - 5, 370 + 25);
31         ctx.fillText(vertices[1], 180, 50 - 10);
32         ctx.fillText(vertices[2], 390 - 10, 370 + 25);
33         ctx.fillText(vertices[3], 280 + 10, 200);
34     };
35
36     NATask.setTask({
37         text: 'В треугольнике $ABC$ ' + ['$AC=BC$', '$AB=' + a + '$',

```

```

    '$АН - высота$', '$ВН=' + b + '$'].shuffleJoin(', ') + '. Найдите
    косинус $ВАС$.',
37     answers: b / a,
38     author: ['Суматохина Александра']
39   });
40   NATask.modifiers.variativeABC(vertices);
41   NATask.modifiers.addCanvasIllustration({
42     width: 400,
43     height: 400,
44     paint: paint1,
45   });
46   }, 1000);
47   })();

```

Примеры генерации задний приведены в листинге 8

## 2. Генерация текстовых задач

В Первом разделе представлены работы, связанные Текстовыми задачами. Так как в При построении текстовых шаблонов задач важно обеспечить как вариативность формулировок, так и корректность числовых и языковых выражений. Для этого в проекте применяются следующие инструменты.

### 2.1. Декорации

Чтобы увеличить количество уникальных вариантов задач, сохраняя их математическую суть, используются так называемые ``декорации'' --- элементы окружения, которые можно менять без потери смысла задачи: имена персонажей, место действия, цель, контекст. Для этого создаются массивы строк, а функция `iz()` случайным образом выбирает элемент из массива.

Пример:

```
1 let contract = ['о дружбе', 'во избежание двойного налогообложения',  
2   'о безвизовом режиме', 'об экологической среде',  
3   'по гуманитарным вопросам', 'по вопросам безопасности'].iz();
```

*Задача №514913.*

Если необходимо выбрать два случайных, но не повторяющихся элемента, используется форма `iz(2)`:

```
1 let tapeName = sklonlxkand(['лента', 'верёвка', 'нитка'].iz(2));
```

*Задача №2434.*

Чтобы каждый раз не задавать список декораций, у нас есть несколько заготовленных, которые используются от задачи к задаче. Например `om.meltov=` и они могут быть расписаны даже по нескольким падежам.

Например:

```
1 om.meltov={}
```

```

2 om.meltov.ie = ['фонарик', 'флакон шампуня', 'флэшка', 'компактдиск-',
    'сувенир', 'матрёшка', 'магнит на холодильник', 'сборник тестов для подготовки
    к ЕГЭ', 'тетрадь', 'учебник', 'цветочный горшок'];

```

В задаче же благодаря этому можно записать короче:

```

1 let item = sklonlxkand((om.meltov.ie).iz());

```

### *Задача №314120.*

Как вы могли заметить, некоторые декорации являются словосочетанием, например "флакон шампуня", "цветочный горшок", "магнит на холодильник" и так далее. И для более корректной работы `.ie` и других падежей у нас есть словарь `lxsoch.js`. Однако недавно было выявлено, что если рядом со словосочетанием стоит число, то оно должно изменяться. например "3 цветочных горшка" или "4 флакона шампуня". Они очень схожи с родительным падежом множественного числа, но отличаются. Так что мною был добавлен в `lxsoch.js` `.r2`. Это падеж, в котором учтено изменения слова из-за числительного.

## 2.2. Склонения существительных

В силу особенностей русского языка слова должны корректно изменяться по падежам и числам. Для этого применяется функция `sklonlxkand`, позволяющая генерировать правильные словоформы. После выбора слова к нему можно обратиться по падежу и числу. Например, `.ie` означает именительный падеж, единственное число.

Пример использования:

```

1 let typeOfFlowerInVases =
2     sklonlxkand(['роза', 'гвоздика', 'ромашка', 'лилия',
3                 'мак', 'ирис', 'лаванда', 'мимоза'].iz());
4 NATask.setTask({
5     text: 'На прилавке цветочного магазина стоят три вазы с '
6         + typeOfFlowerInVases.tm + ': ' + vaseColor[0] + 'ая, '
7         + vaseColor[1] + 'ая, ' + vaseColor[2] + 'ая.'

```

```

8   + ' Слева от ' + vaseColor[2] + 'ой вазы '
9   + chislitlx(leftOfThirdVase, typeOfFlowerInVases, '$')
10  + ', справа от ' + vaseColor[0] + 'ой вазы '
11  + chislitlx(rightOfFirstVase, typeOfFlowerInVases, '$') + '. '
12  + 'Всего в вазах '
13  + chislitlx(allFlowerInVases, typeOfFlowerInVases, '$') + '. '
14  + 'Сколько ' + typeOfFlowerInVases.rm + ' в '
15  + vaseColor[1] + 'ой вазе?',
16  answers: secondVaseCountFlower,
17 });

```

### Задача №515842.

Для корректного согласования числительных с существительными используется функция `chislitlx`. Она автоматически выбирает нужную форму слова в зависимости от числа.

Пример:

```

1  let numberOfApartamentPerFloor = sluchch(3, 12, 1);
2  NATask.setTask({
3    text: 'В доме, в котором живет ' + nameOfPerson + ', ' +
4          '$' + floorNumber + '$ этажей и несколько подъездов. На каждом этаже
           находится по ' +
5          chislitlx(numberOfApartamentPerFloor, 'квартира', '$') + '. '
6          + nameOfPerson + ' живет в квартире №' + '$' + apartmentNumber + '$' +
7          '. '
8          + 'В каком подъезде живет ' + nameOfPerson + '? ',
9    answers: '$' + (apartmentNumber /
10                 (floorNumber * numberOfApartamentPerFloor)).ceil() + '$',
11  });

```

### Задача №77351.

## 2.3. Проверка условий (генерация утверждений)

Помни во всех шаблонах вы так или иначе увидите окружение `retryWhileError`, которое ограничивает количество перезапусков, плюсом ещё и фиксирует какие проверки не были пройдены и выводить их на экран. Само собой ошибки видны только разработчикам Во время отладки. `function retryWhileError(theFunction, maxIterations, maxCollectedErrors)`

Пример:

```
1 (function () {
2     'use strict';
3     retryWhileError(function () {
4         NAinfo.requireApiVersion(0, 2);
5
6         let placeOfRest = ['пансионате', 'санатории', 'здравнице', 'курортном
7         доме', 'отеле', 'гостинице', 'доме отдыха'].iz();
8         let whoIsRest = ['дети', 'взрослые', 'пенсионеры', 'подростки'].iz();
9         let rand = sl(0, 6);
10        let theShareOfVacationers = ['Половина', 'Четверть', 'Одна пятая',
11        'Десятая часть', 'Двадцатая часть', 'Одна двадцать пятая', 'Пятидесятая']
12        let theShareOfVacationersInNumbers = [0.5, 0.25, 0.2, 0.1, 0.05,
13        0.04, 0.02][rand]
14
15        NATask.setTask({
16            text:
17                theShareOfVacationers + ' всех отдыхающих в ' + placeOfRest
18            + ' - ' + whoIsRest + '. Какой процент от всех отдыхающих составляют ' +
19            whoIsRest + '?',
20            answers: 100 * theShareOfVacationersInNumbers,
21        });
22    }, 100);
23})();
24//https://mathb-ege.sdamgia.ru/test?likes=509768
```

Для контроля корректности задачи применяются функции `genAssert` и её вариации. Принцип работы:

- если условие не выполнено --- фиксируется ошибка;
- шаблон перезапускается;
- если достигнуто максимальное количество перезапусков, выводятся накопившиеся ошибки с указанием количества.

Пример использования:

```
1 let students = sl(200, 10000, 10);
2 let percent = sl(10, 90, 1);
3 genAssert(students.kratno(100 / percent),
4           "количество учащихся не кратно процент100/");
```

*Задача №77340.*

Также часто используется функция `function genAssertZ1000(number, message)`, проверяющая точность чисел: если у числа более трёх знаков после запятой, шаблон перезапускается. `function genAssertIrreducible(numerator, denominator, message)` - проверка на несократимость дроби.

Пример:

```
1 let prise = sl(100, 2000, 10);
2 let percentSecondMonth = sl(10, 90, 1);
3 let percentThirdMonth = sl(10, 90, 1);
4
5 let middlePrise = prise * (1 + [1, -1][randFirst] * 0.01 *
6   percentSecondMonth);
7 let finalePrise = middlePrise * (1 + [1, -1][randSecond] * 0.01 *
   percentThirdMonth);
```

```

8 genAssertZ1000(finalePrise / 10,
9     'Число имеет более 2 знаков после запятой');

```

### Задача №77349.

В некоторых случаях может применяться `slKrome`, он работает как `sluchch` - создает случайное число. Однако с условием что оно отличное от первого параметра (первый параметр может быть числом, строкой, массивом или функцией), второй и третий параметры - это диапазон генерации, а четвертый параметр - шаг (по умолчанию это 1). Пример:

```

1 let mlnRuble = slKrome([100], 10, 200, 1);

```

### Задача №506346.

`kratno` является функцией, провищаяющее кратно ли данное число другому. Чаще всего используем в `genAssert` для проверки Пример: `let students = sl(200, 10000, 10); let percent = sl(10, 90, 1); genAssert(students.kratno(100 / percent), "количество учащихся не кратно 100/процент");` *Задача №77340.*

`NAtask.modifiers.allDecimalsToStandard` мы используем в случаях работы с десятичными числами и дробями, для более коррктенного расчёта java script формул. Это помогает избежать излишней точности.

Пример:

```

1 let difference = sl(2, 20, 1);
2     let minAngle = (360 / (2 * difference + 1));
3     let maxAngle = (360 / (difference + 2));
4
5     let result = maxAngle.floor() - minAngle.floor() - Number(maxAngle
6 % 1 == 0 minAngle % 1 == 0);
7
8     NAtask.setTask({
9         text: 'Три луча, выходящие из одной точки, разбивают плоскость на
$3$ ' +
        'разных угла, измеряемых целым числом градусов. Наи' +
        moreOrLess + 'ьший угол в ' +

```



```

10         chislitlx(difference, 'раз', 'v$') + ' ' +
moreThanLessThan +
11         ' наи' + lessOrMore + 'ьшего. Сколько значений может принимать
величина среднего угла?',
12         answers: result,
13     });
14     NATask.modifiers.allDecimalsToStandard();

```

*Задача №514918.*

## 2.4. Разнообразие в текстовых задачах

И самое главное для текстовых задач, возможность их разнообразить. Дело в том что изначально даётся достаточно много условий в задаче, от чего можно написать альтернативные варианты задачи. Что помогает ученикам лучше запоминать не просто последовательность решения, а принцип. Чаще всего от изменений условия задачи на альтернативное сильно меняется и текст, по этому мы используем questions. Он позволяет разбить на несколько заданий одно и для каждого приписать свой ответ. Если же у них общее начало то можно записать его в первый text, а если одинаковое окончание то дописать его в postquestion.

Пример1:

```

1 (function () {
2     'use strict';
3     retryWhileError(function () {
4         NAinfo.requireApiVersion(0, 2);
5
6         let rand = sl1();
7
8         let clothMR = sklonlxkand(['свитер', 'халат', 'плащ', 'дождевик',
'tовар'].iz());
9         let clothWR = sklonlxkand(['рубашка', 'футболка', 'куртка',
'кофта'].iz());
10        let clothers = [clothMR, clothWR][rand];

```

```

11
12     let cost = ['стоил', 'стоила'][rand];
13     let hasBecome = ['стал', 'стала'][rand]
14     let mw = ['', 'a'][rand];
15     let prise = sl(300, 5000, 10);
16     let percent = sl(10, 80, 1);
17
18     let result = (100 - percent) * 0.01 * prise;
19
20
21     NATask.setTask({
22         text:
23             clothers.ve.toZagl() + ' на распродаже уценили',
24         questions: [
25             {
26                 text: ' на ' + '$' + percent + '$' + '%, при этом он' +
27                 mw + ' ' + hasBecome + ' стоить ' + '$' + result + '$' + ' р. ' +
28                 'Сколько рублей ' + cost + ' ' + clothers.ie + ' до
29                 распродажи',
30                 answers: prise,
31             },
32             {
33                 text: ', при этом он' + mw + ' ' + hasBecome + ' стоить
34                 ' + '$' + result + '$' + ' р. ' + 'До распродажи ' + clothers.ie + ' ' +
35                 cost +
36                 ' ' + '$' + prise + '$' + ' р. Сколько процентов
37                 составляет скидка',
38                 answers: percent,
39             },
40             {
41                 text: ' на ' + '$' + percent + '$' + '%.' + ' До
42                 распродажи ' + clothers.ie + ' ' + cost + ' ' + '$' + prise + '$' +
43                 ' р. Сколько рублей стоит ' + clothers.ie + ' во
44                 время распродажи',
45                 answers: result,

```

```

39         },
40
41         ],
42         postquestion: '?',
43     });
44     NATask.modifiers.allDecimalsToStandard();
45     }, 100);
46 })();
47 //https://mathb-ege.sdamgia.ru/test?likes=506426
48 //zer00player

```

*Задача №506426.*

Пример2:

```

1 (function () {
2     'use strict';
3     retryWhileError(function () {
4         NAinfo.requireApiVersion(0, 2);
5
6         let rand = sl1();
7         let anotherRand = sl(0, 4, 1);
8
9         let endPrilag = ['й-', 'ю-'][rand];
10        let anotherEndPrilagFirst = ['ый', 'юй'][rand];
11        let anotherEndPrilagSecond = ['ой', 'уй'][rand];
12        let endChislit = ['х', 'х', 'ти', 'ми', 'ти'][anotherRand];
13        let oneItem = ['один', 'одна'][rand];
14
15        let clothMR = sklonlxkand(['свитер', 'халат', 'плащ',
16        'дождевик'].iz());
17        let clothWR = sklonlxkand(['рубашка', 'футболка', 'куртка',
18        'кофта'].iz());
19        let clothers = [clothMR, clothWR][rand];
20
21        let prise = sl(300, 5000, 10);
22        let percent = sl(10, 80, 1);

```

```

21     let countOfClother = sl(2, 4, 1);
22     let anotherCountOfClother = [2, 4, 6, 8, 10][anotherRand];
23
24     let result = (countOfClother - 1) * prise + (100 - percent) * 0.01
* prise;
25     let anotherResult = (anotherCountOfClother / 2) * prise +
(anotherCountOfClother / 2) * ((100 - percent) * 0.01 * prise);
26
27     NATask.setTask({
28         text:
29         'В спортивном магазине ' + ['любой', 'любая'][rand] + ' ' +
clothers.ie,
30         questions: [
31             {
32                 text: ' стоит ' + '$' + prise + '$' + 'p. ' +
33                 'Сейчас магазин проводит акцию: при покупке ' + '$' +
countOfClother + '$' + '-x ' + clothers.rm + ' - скидка на ' + '$' +
34                 countOfClother + '$' + endPrilag + ' ' +
clothers.ve + ' ' + '$' + percent + '$' + '%. ' +
35                 'Сколько рублей придётся заплатить за покупку ' + '$'
+ countOfClother + '$' + '-x ' + clothers.rm + ' в период действия акции',
36                 answers: result,
37             },
38             {
39                 text: ' стоит ' + '$' + prise + '$' + 'p. ' +
40                 'Сейчас магазин проводит акцию: при покупке ' +
clothers.rm +
41                 ' - скидка на кажд' + anotherEndPrilagFirst + '
втор' + anotherEndPrilagSecond + ' составляет ' + '$' + percent + '$' +
42                 '%. ' +
43                 'Сколько рублей придётся заплатить за покупку ' +
'$' + anotherCountOfClother + '$' + '-' +
endChislit + ' ' + clothers.rm + ' в период действия акции',
44                 answers: anotherResult,
45             },

```

```

46         {
47             text: ' стоит ' + '$' + prise + '$' + 'p. ' +
48                 'Сейчас магазин проводит акцию: при покупке ' + '$' +
countOfClother + '$' + '-x ' + clothers.rm + ' - будет скидка на ' + '$' +
49                 countOfClother + '$' + endPrilag + ' ' +
clothers.ve + '. ' +
50                 'Покупатель оплатил покупку ' + '$' + countOfClother
+ '$' + '-x ' + clothers.rm +
51                 ' в период действия акции, что стоило ему ' + '$' +
result + '$' + 'p. ' + 'Сколько процентов составляет скидка',
52             answers: percent,
53         },
54         {
55             text: ' попадает под акцию: при покупке ' + '$' +
countOfClother + '$' + '-x ' + clothers.rm + ' - скидка на ' + '$' +
56                 countOfClother + '$' + endPrilag + ' ' +
clothers.ve + ' ' + '$' + percent + '$' + '%. ' +
57                 'Покупатель оплатил покупку ' + '$' + countOfClother
+ '$' + '-x ' + clothers.rm +
58                 ' в период действия акции, что стоило ему ' + '$' +
result + '$' + 'p. ' +
59                 'Сколько стоит ' + oneItem + ' ' + clothers.ie + '
без акции',
60             answers: prise,
61         },
62         {
63             text: ' стоит ' + '$' + prise + '$' + 'p. ' +
64                 'Сейчас магазин проводит акцию: при покупке ' + '$' +
anotherCountOfClother + '$' + '-x ' + clothers.rm +
65                 ' - будет скидка на кажд' + anotherEndPrilagFirst +
' втроп' + anotherEndPrilagSecond + ' ' + clothers.ve + '. ' +
66                 'Покупатель оплатил покупку ' + '$' +
anotherCountOfClother + '$' + '-' + endChislit +
67                 ' ' + clothers.rm + ' в период действия акции, что
стоило ему ' + '$' + anotherResult + '$' + 'p. ' +

```

```

68         'Сколько процентов составляет скидка',
69         answers: percent,
70     },
71     {
72         text: ' попадает под акцию: при покупке ' + '$' +
anotherCountOfClother + '$' + '-x ' + clothers.rm +
73         ' - скидка на кажд' + anotherEndPrilagFirst + '
втор' + anotherEndPrilagSecond + ' составляет ' + '$' + percent + '$' +
74         '%. ' +
'Покупатель оплатил покупку ' + '$' +
anotherCountOfClother + '$' + '-' + endChislit + ' ' + clothers.rm +
75         ' в период действия акции, что стоило ему ' + '$' +
anotherResult + '$' + 'p. ' +
76         'Сколько стоит ' + oneItem + ' ' + clothers.ie + '
без акции',
77         answers: prise,
78     },
79 ],
80 postquestion: '?',
81 });
82 }, 100);
83 })();
84 //https://mathb-ege.sdamgia.ru/test?likes=522673
85 //zer00player

```

*Задача №522673.*

## 3. Задачи №4 ЕГЭ (преобразование выражений)

### 3.1. Основные используемые команды

Стандартные команды которыми мы используем из латеха это

`\cdot` и

`\mbox`

Пример:

```
1 NAtask.setTask({
2     text:
3         'В фирме «' + name + '» стоимость в( рублях) колодца из железобетонных
колец рассчитывается по формуле  $C = ' + plus + '+' + multiply +
4         ' \cdot n$ , где  $n$  - число колец, ' +
5         'установленных при рытье колодца. ' +
6         'Пользуясь этой формулой, ' +
7         'рассчитайте стоимость колодца из  $' + number + '$ колец.',
8     answers: cost,
9 });$ 
```

*Задача №124.*

```
1 NAtask.setTask({
2
3     text: 'Количество теплоты в( джоулях), полученное однородным телом при
нагревании, ' +
4         'вычисляется по формуле  $Q = cm(t_2 - t_1)$ , где  $c$  - удельная
теплоёмкость в( Дж), ' +
5         ' $m$  - масса тела в( килограммах),  $t_1$  - начальная температура тела
в( кельвинах), а  $t_2$  ' +
6         ' - конечная температура тела в( кельвинах). Пользуясь этой формулой, ' +
7         the_orderToFind + '  $Q$  в( джоулях), если  $t_2 = ' + t_2 + '$ К,  $c$ 
= ' + c + ' $\frac{\mbox{Дж}}{\mbox{кг}} \cdot \mbox{K}$ ', ' +
8         ' $m = ' + m + '$ кг и  $t_1 = ' + t_1 + '$ К.',
9     answers: Q,$$$ 
```

```
10
11 });
```

*Задача №509609.*

`frac`,

`sqrt` и

`sin` самые часто используемые LaTeX команды и задачах вида 4, для отображения дробей, корней и `sin`.

Пример:

```
1 NAtask.setTask({
2     text: 'Радиус окружности, описанной около треугольника, ' +
3         'можно вычислить по формуле  $R = \frac{a}{2\sin{\alpha}}$ , где  $a$ 
- сторона, ' +
4         ' $a$   $\alpha$  - противолежащий ей угол треугольника. ' +
5         'Пользуясь этой формулой, ' + the_orderToFind + '  $R$ ,
'a'] [rand] + '$' +
6         ', если  $a = ' + a, 'R = ' + R [rand] + '$ и  $\sin{\alpha} =
\frac{' + num + '}{' + deNum + '}$ .' ,
7         answers: [R, a] [rand],
8         preference: preference,
9     });$ 
```

*Задача №506300.*

## 3.2. Адаптирование команд

Но для более удобной работы с ними в тексте у нас имеется например `.texfrac()`

Пример:

```
1 NAtask.setTask({
2
3     text: 'Теорему синусов можно записать в виде  $\frac{a}{\sin{\alpha}} = \frac{b}{\sin{\beta}}$   $' +$ 
```



```

4      ', где $a$ и $b$ - две стороны треугольника, а $\alpha$ и $\beta$ -
углы треугольника, лежащие против них соответственно. ' +
5      ' Пользуясь этой формулой, ' + the_orderToFind + ' ' + '$' +
['\\sin{' + nameSin[0] + '}', nameLetter[0]][rand] + '$' +
6      ', если ' + '$' + [nameLetter[0] + ' =' + a, '\\sin{' + nameSin[0]
+ '}' = ' + sinA.texfrac(1)][rand] + '$' +
7      ', '$' + nameLetter[1] + ' =' + b + '$, $\\sin{' + nameSin[1] + '}'
= ' + sinB.texfrac(1) + '$.',
8      answers: [sinA, a][rand],
9      preference: preference,
10
11  });

```

*Задача №530329.*

(про `.shuffle()`) Пример:

```

1  let letter = ['a', 'b', 'c'].shuffle();

```

*Задача №2939.*

```

1  (function () {
2      'use strict';
3      retryWhileError(function () {
4          NAinfo.requireApiVersion(0, 2);
5
6          let the_orderToFind = decor.orderToFind.iz();
7
8          let a = sl(2, 50);
9          let b = slKrome([a], 2, 50);
10         let c = slKrome([a, b], 2, 50, 0.5);
11
12         let V = a * b * c;
13         let letter = ['a', 'b', 'c'].shuffle();
14
15         NATask.setTask({
16
17             text: 'Объём прямоугольного параллелепипеда вычисляется по формуле $V =

```

```

abc$, где $a$, $b$ и $c$ - длины трёх его рёбер, ' +
18     'выходящих из одной вершины. Пользуясь этой формулой, ' +
the_orderToFind + ' $' + letter[0] + '$, если $V = ' + V + '$, ' +
19     ' $' + letter[1] + ' = ' + b + '$ и $' + letter[2] + ' = ' + c +
'$.',
20     answers: a,
21
22     });
23     NATask.modifiers.allDecimalsToStandard();
24 }, 2000);
25 })();
26 //zer00player
27 //https://self-edu.ru/ege2021_base_30.php?id=6_4

```

(про .sqrt() .cbrt())

Пример:

```

1 (function () {
2     'use strict';
3     retryWhileError(function () {
4         NAinfo.requireApiVersion(0, 2);
5
6         let rand = sl(0, 8);
7         let the_orderToFind = decor.orderToFind.iz();
8         let planet = sklonlxkand(['Меркурий', 'Венера', 'Земля', 'Луна', 'Марс',
9             'Юпитер', 'Сатурн', 'Уран', 'Нептун'][rand]);
10
11         let m = sl(1, 20, 0.1);
12         let h = slKrome([m], 1, 20, 0.1);
13         let g = [3.7, 8.9, 9.8, 1.6, 3.9, 23.9, 10.4, 8.9, 11][rand];
14         let E = m * h * g;
15
16         NATask.setTask({
17             text: 'Потенциальная энергия тела в( джоулях) в поле тяготения ' +
planet.re + ' вблизи ' + ['его', 'её'][planet.rod] + ' поверхности

```

```

        вычисляется по формуле  $E = mgh$ , ' +
18         ' где  $m$  - масса тела в( килограммах),  $g$  - ускорение свободного
        падения в(  $m/s^2$ ), а  $h$  - высота в( метрах), ' +
19         ' на которой находится это тело относительно поверхности. ' +
20         'Пользуясь этой формулой, ' + the_orderToFind + '  $m$  в( килограммах),
        если  $g = ' + g + '$  $m/s^2$ ,  $h = ' + h + '$ м, а  $E = ' + E +
        '$ Дж. ',
21         answers: m,
22
23     });
24     NATask.modifiers.allDecimalsToStandard();
25 }, 2000);
26 })();
27 //zer00player
28 //https://mathb-ege.sdangia.ru/test?likes=506737$$$ 
```

`isValidTriangle()` - из `flatten-shape-geometry 1.8.2`, из 3 переменных проверяет могут ли они составить треугольник. и выдают `true` - если является, и `false` -если нет Пример:

```

1  let a = sl(2, 30);
2  let b = sl(2, 30);
3  let c = sl(2, 30);
4
5  genAssert(isValidTriangle(a, b, c), 'Должно являться треугольником');

```

*Задача №506550.*

`.rod` - помогает в определении рода существительного (где 0 - мужской, 1 - женский, 2 - средний, 3- всегда множественный). Очень помогает когда местоимение зависит от рода существительного.

Пример:

```

1  (function () {
2    'use strict';
3    retryWhileError(function () {
4      NAinfo.requireApiVersion(0, 2);

```

```

5
6   let rand = sl(0, 8);
7   let the_orderToFind = decor.orderToFind.iz();
8   let planet = sklonljkand(['Меркурий', 'Венера', 'Земля', 'Луна', 'Марс',
'Юпитер', 'Сатурн', 'Уран', 'Нептун'][rand]);
9
10  let m = sl(1, 20, 0.1);
11  let h = slKrome([m], 1, 20, 0.1);
12  let g = [3.7, 8.9, 9.8, 1.6, 3.9, 23.9, 10.4, 8.9, 11][rand];
13  let E = m * h * g;
14
15  NATask.setTask({
16
17      text: 'Потенциальная энергия тела в( джоулях) в поле тяготения ' +
planet.re + ' вблизи ' + ['его', 'её'][planet.rod] + ' поверхности
вычисляется по формуле  $E = mgh$ ,' +
18      ' где  $m$  - масса тела в( килограммах),  $g$  - ускорение свободного
падения в(  $\text{м/с}^2$ ), а  $h$  - высота в( метрах),' +
19      ' на которой находится это тело относительно поверхности. ' +
20      'Пользуясь этой формулой, ' + the_orderToFind + '  $m$  в( килограммах),
если  $g = ' + g + '$  $\text{м/с}^2$ ,  $h = ' + h + '$ м, а  $E = ' + E +
'$ Дж.',
21      answers: m,
22
23  });
24  NATask.modifiers.allDecimalsToStandard();
25  }, 2000);
26  })();
27  //zer00player
28  //https://mathb-ege.sdamgia.ru/test?likes=506737$$$ 
```

*Задача №512937.*

.isZ() проверяет является ли n целым числом.

Пример:

```

1 let second = sl(5, 30);
2     let amperage = sl(5, 30);
3     let voltage = sl(5, 30);
4     let resistance = slKrome([amperage, voltage, second], 5, 30);
5     let answer = [amperage ** 2 * resistance * second, voltage ** 2 *
second / resistance][rand];
6
7     genAssert(answer.isZ(), 'должно быть целым');

```

*Задача №523098.*

### 3.3. Обратные задачи

про preference

```

1 (function () {
2     'use strict';
3     retryWhileError(function () {
4         NAinfo.requireApiVersion(0, 2);
5
6         let key = '530329';
7         let preference = ['findSin', 'findSide'];
8         let rand = getSelectedPreferenceFromList(key, preference);
9
10        let the_orderToFind = decor.orderToFind.iz();
11
12        let a = sl(1, 30);
13        let b = slKrome([a], 1, 30);
14
15        let deNumA = sl(3, 50);
16        let numA = sl(1, deNumA - 1);
17        let sinA = numA / deNumA;
18
19        genAssertZ1000(sinA, 'sinA должно быть не более 3 знаков после запятой');
20
21        let sinB = (sinA * b) / a;

```

```

22
23     genAssertZ1000(sinB, 'sinB должно быть не более 3 знаков после запятой');
24     genAssert(sinB < 1, 'sinB не должен превышать 1');
25
26     let nameLetter = ['a', 'b'].shuffle();
27     let nameSin = ['\\alpha', '\\beta'].shuffle();
28
29     NATask.setTask({
30
31         text: 'Теорему синусов можно записать в виде $
\\frac{a}{\\sin{\\alpha}} = \\frac{b}{\\sin{\\beta}} $' +
32         ', где $a$ и $b$ - две стороны треугольника, а $\\alpha$ и $\\beta$ -
углы треугольника, лежащие против них соответственно. ' +
33         ' Пользуясь этой формулой, ' + the_orderToFind + ' ' + '$' +
['\\sin{' + nameSin[0] + '}', nameLetter[0]][rand] + '$' +
34         ', если ' + '$' + [nameLetter[0] + ' = ' + a, '\\sin{' + nameSin[0]
+ '}' = ' + sinA.texfrac(1)][rand] + '$' +
35         ', '$' + nameLetter[1] + ' = ' + b + '$, $\\sin{' + nameSin[1] + '}'
= ' + sinB.texfrac(1) + '$.',
36         answers: [sinA, a][rand],
37         preference: preference,
38
39     });
40     NATask.modifiers.allDecimalsToStandard();
41 }, 20000);
42 })();
43 //zer00player
44 //https://mathb-ege.sdamgia.ru/problem?id=530329

```

## 4. Задачи №7 ОГЭ (координатная прямая)

### 4.1. разработка функции

Наиболее значимая часть работы — это разработка функций для визуализации координатной прямой и точек на ней. Была создана универсальная функция, позволяющая отображать засечки и подписи в разных режимах.

Одним из ключевых элементов реализации автоматической генерации заданий стала функция *coordAxisDrawMarkPoint*. Она предназначена для отрисовки различных типов меток на координатной оси и их подписей.

### 4.2. Назначение функции

Функция решает задачу визуализации точек и вспомогательных обозначений на оси, что является неотъемлемой частью заданий ОГЭ и ЕГЭ по математике. С помощью данной функции возможно изображать:

- закрашенные точки (```dot''`),
- выколотые точки (```emptyDot''`),
- засечки (```line''`),
- отсутствие метки (```nothing''`).

```
1 function coordAxisdrawMarkPoint(ct, coord, text, markForm, textPosition,  
   options = {}) {  
2 //@author : zer00player  
3 //coordAxisdrawMarkPoint функция, которая рисует отметку на координатной оси  
   точку(, выколтую точку, засечку или ничего) с подписью.  
4 /*  
5  * @param {CanvasRenderingContext2D} ct Контекст отрисовки.  
6  * @param {number} coord Координата по оси X, где нужно поставить отметку.  
7  * @param {string} text Подпись
```

```

* @param {'dot'|'emptyDot'|'line'|'nothing' markForm Форма метки: точка,выколотая
точка, засечка, отсутствие метки.* @param 'underAxis'|"overAxis"|"onAxis"
textPosition Позиция текста относительно оси.* @param Object [options=]
Дополнительные параметры оформления.*/let font = "16px liberation sans",fillStyle
= om.secondaryBrandColors[0],strokeStyle = fillStyle,lineWidth = 1.5,textOffsetX =
options;/** @param string [options.font="16px liberationsans"] Шрифт подписи.*
@param string [options.fillStyle=om.secondaryBrandColors[0]] Цвет подписи и
заливки точки.* @param string [options.strokeStyle=fillStyle] Цвет засечки (по
умолчанию совпадает с fillStyle).* @param number [options.lineWidth=1.5] Толщина
линии засечки.* @param number [options.textOffsetX] смещение текста по X.*/let
prevFillStyle = ct.fillStyle;let prevStrokeStyle = ct.strokeStyle;let prevFont =
ct.font;let prevLineWidth = ct.lineWidth;/** @private* @type string prevFillStyle
сохраняет текущий цвет перед временной заменой* @type string prevStrokeStyle
сохраняет текущий цвет перед временной заменой* @type string prevFont сохраняет
текущий шрифт перед заменой* @type number prevLineWidth сохраняет текущую толщину
перез заменой*/ct.fillStyle = fillStyle;ct.strokeStyle = strokeStyle;ct.font =
font;ct.lineWidth = lineWidth;/** Устанавливает новые параметры рисования перед
отрисовкой объекта.* @param string fillStyle Цвет заливки (например, для текста
или точки).* @param string strokeStyle Цвет обводки (например, для засечек).*
@param string font Шрифт текста (например, "16px liberationsans").* @param number
lineWidth Толщина линий.*//* @param 'dot'|'emptyDot'|'line'|'nothing' markForm
Форма метки: точка,выколотая точка, засечка, отсутствие метки.switch (markForm)
case "dot":ct.drawFilledCircle(coord, 0, 5);break;case
"emptyDot":ct.drawFilledCircle(coord, 0, 5);ct.fillStyle =
"#ffffff";ct.drawFilledCircle(coord, 0, 3);ct.fillStyle = fillStyle;break; case
"line":ct.lineWidth = 1.5;ct.drawLine(coord, -6, coord, 6);break;case
"nothing":break;/** @param 'underAxis'|"overAxis"|"onAxis" textPosition Позиция
текста относительно оси.Под осью(для чёрточек), Над осью(для точек) и по линии
оси(после стрелки).switch (textPosition) case "underAxis":ct.fillText(text, coord
- (textOffsetX || 4), 20);break;case "overAxis":ct.fillText(text, coord -
(textOffsetX || 4), -10);break;case "onAxis":ct.fillText(text, coord +
(textOffsetX || 6), 5);break;ct.fillStyle = prevFillStyle;ct.strokeStyle =
prevStrokeStyle;ct.font = prevFont;ct.lineWidth = prevLineWidth;/** Возвращает
исходные параметры рисования после завершения работы функции.* @see prevFillStyle
возвращает исходный цвет* @see prevStrokeStyle возвращает исходный цвет* @see
prevFont возвращает исходный шрифт* @see prevLineWidth возвращает исходную
толщину*/});

```

## 4.3. Интерфейс функции

Функция имеет следующий набор параметров:

- `ct` --- графический контекст Canvas,
- `coord` --- координата по оси X,
- `text` --- подпись для метки,
- `markForm` --- форма метки: `dot`, `emptyDot`, `line`, `nothing`,
- `textPosition` --- расположение подписи: под осью (`underAxis`), над осью (`overAxis`), на оси (`onAxis`),



- `options` --- дополнительные параметры (шрифт, цвет текста, толщина линии, смещение).

## 4.4. Алгоритм работы

1. Сохраняются текущие параметры отрисовки (`fillStyle`, `strokeStyle`, `font`, `lineWidth`).
2. Устанавливаются новые параметры, переданные в `options`.
3. В зависимости от параметра `markForm` рисуется выбранный элемент:
  - точка --- закрашенный круг,
  - выколота точка --- окружность с заливкой белым цветом внутри,
  - засечка --- вертикальная черта,
  - отсутствие --- элемент не отрисовывается.
4. В зависимости от параметра `textPosition` подпись размещается под осью, над осью или на линии оси.
5. Восстанавливаются исходные параметры графического контекста.

`coordAxis_prepare`, что позволяет подготовить область для оси и рисует стрелку. И `coordAxis_drawAuto` она автоматически вычисляет масштаб оси и вызывает `coordAxis_drawMarkPoint` для всех точек.

Функция `coordAxis_prepare` выполняет подготовку холста для отрисовки горизонтальной координатной оси со стрелкой. Она:

- задаёт габариты рабочей области оси (`width`, `height`) и сохраняет их в контексте для последующего использования;
- вертикально центрирует ось (Ох) (смещение системы координат);
- настраивает стили (цвет линии и толщину) и рисует ось со стрелкой;

- бережно восстанавливает исходные графические параметры контекста.

Компонент рассчитан на дальнейшее использование вместе с *coordAxis<sub>d</sub>drawMarkPoint* и *coordAxis<sub>d</sub>drawAuto*.

```

1 function coordAxisprepare(ct, { width = 400, height = 100, strokeStyle =
   om.primaryBrandColors[0], lineWidth = 2 } = {}) {
2   /*
3    * coordAxisprepare подготавливает прямую со стрелкой к отрисовке.
4    * @param {CanvasRenderingContext2D} ct Контекст отрисовки.
5    * @param {Object} [params={}] Опции настройки оси.
6    * @param {number} [params.width=400] Ширина области.
7    * @param {number} [params.height=100] Высота области.
8    * @param {string} [params.strokeStyle=om.primaryBrandColors[0]] Цвет
   линии оси.
9    * @param {number} [params.lineWidth=2] Толщина оси.
10  */
11
12  ct.__coordAxisW = width;
13  ct.__coordAxisH = height;
14  /*
15   * @param {number} width Желаемая ширина координатной оси в пикселях.
16   * @param {number} height Желаемая высота координатной оси в пикселях.
17  */
18
19  let prevStroke = ct.strokeStyle;
20  let prevLineWidth = ct.lineWidth;
21  /*
22   * @private
23   * @type {string} prevStrokeStyle сохраняет текущий цвет перед временной
   заменой
24   * @type {number} prevLineWidth сохраняет текущую толщину через замену
25  */
26
27  ct.translate(0, height / 2);
28  /*

```

```

29  * Смещает систему координат: вертикально центрирует ось X.
30  * @param {number} x Горизонтальное смещение в( данном случае 0).
31  * @param {number} y Вертикальное смещение - половина высоты canvas.
32  * @this {CanvasRenderingContext2D}
33  */
34
35  ct.strokeStyle = strokeStyle;
36  ct.lineWidth = lineWidth;
37  /*
38  * @param {string} strokeStyle установка цвет линии оси.
39  * @param {number} lineWidth установка толщина оси.
40  */
41
42  ct.drawArrow(10, 0, width - 10, 0);
43  /*
44  * Рисует горизонтальную ось с направляющей стрелкой.
45  * @param {number} x1 Начальная координата оси.
46  * @param {number} y1 Начальная координата по оси Y (0).
47  * @param {number} x2 Конечная координата оси.
48  * @param {number} y2 Конечная координата по оси Y (0).
49  */
50
51  ct.strokeStyle = prevStroke;
52  ct.lineWidth = prevLineWidth;
53  /*
54  * Возвращает исходные параметры рисования после завершения работы функции.
55  * @see prevStrokeStyle
56  * @see prevLineWidth
57  */
58 };
59
60 function mathToCanvas(x, min, scale, x0) {
61     return x0 + (x - min) * scale;
62 };

```

**ct** графический контекст `CanvasRenderingContext2D`.

**width** ширина области оси в пикселях, по умолчанию (400).

**height** высота области в пикселях, по умолчанию (100).

**strokeStyle** цвет линии оси (интеграция со стилем проекта через `om.primaryBrandCol`).

**lineWidth** толщина линии оси, по умолчанию (2).

Функция явно сохраняет текущие значения **strokeStyle** и **lineWidth** в локальные переменные **prevStroke** и **prevLineWidth** и восстанавливает их к концу выполнения. Параметры `ct.__coordAxisW` и `ct.__coordAxisH` записываются в контекст как служебные метаданные — это упрощает доступ к габаритам при последующих рисованиях (например, при авторазметке меток).

Важно, что вызывается `ct.translate(0, height/2)`: система координат сдвигается на половину высоты вниз, чтобы ось (Ох) оказалась по центру холста. Этот сдвиг является *накопительным*; поэтому рекомендуется либо:

- вызывать `coordAxis_prepare` один раз в рамках одного цикла отрисовки, либо
- оборачивать работу в `ct.save() ... ct.restore()`, если требуется многократная подготовка в одном контексте.

`coordAxis_prepare` задаёт «сцену» — габариты, позицию оси и её визуальные атрибуты. Поверх этой сцены функции `coordAxis_drawMarkPoint` и `coordAxis_drawAutoLabel` размещают метки и подписи. Такое разделение обязанностей упрощает поддержку кода: изменения оформления оси не затрагивают логику генерации и размещения меток. Данная функция является частью связки:

Из особенностей можно подчеркнуть что функция поддерживает как закрашенные, так и выколотые точки, что позволяет формировать задания с открытыми и закрытыми интервалами. Так же она восстанавливает исходные параметры,

гарантирует корректную работу при множественной отрисовке. А так же есть возможность смещения текста по оси  $X$ , что помогает избежать наложений подписей.

```
1 function coordAxisdrawAuto(ct, {
2   points = [], // [{ value: число, label: строка, mark:
3     'dot' 'line' 'nothing', labelPos: 'overAxis' 'underAxis' 'onAxis' }]
4   margin = 20,
5   width = 400,
6   height = 100
7 } = {}) {
8   coordAxisprepare(ct, { width, height });
9
10  if (points.length === 0) {
11    return;
12  };
13
14  let values = points.map(p => p.value);
15  let min = values.minE();
16  let max = values.maxE();
17
18  if (min === max) {
19    min -= 1;
20    max += 1;
21  } else {
22    let padding = (max - min) * 0.05;
23    min -= padding;
24    max += padding;
25  }
26
27  let range = max - min;
28
29  let x0 = margin;
30  let x1 = width - margin;
31  let scale = (x1 - x0) / range;
32
33  for (let { value, label, mark, labelPos } of points) {
```

```
32     let x = mathToCanvas(value, min, scale, x0);
33     coordAxisdrawMarkPoint(ct, x, label, mark, labelPos);
34 }
35 };
```

## 4.5. Пример использования

// Отрисовка закрашенной точки A с подписью под осью

```
coordAxis_drawMarkPoint(ct, 100, "A", "dot", "underAxis");
```

// Отрисовка выколотой точки B с подписью над осью

```
coordAxis_drawMarkPoint(ct, 200, "B", "emptyDot", "overAxis");
```

## 5. Задачи №7 ОГЭ (сравнение чисел)

В проекте присутствует отдельный шаблон, посвящённый задачам на сравнение действительных чисел. Основная цель подобных заданий --- научить учащегося ориентироваться в числовой прямой, дробях и приближённых значениях корней.

Так как в задачах встречаются как обыкновенные дроби, так и десятичные приближения квадратных корней, важно уметь корректно округлять результаты. Для этого используется метод `toFixed(1)`, позволяющий оставить одно десятичное число. Например:

```
1 let correctVal = ((frac1 + frac2) / 2);
2 let correct = correctVal.toFixed(1).ts();
```

Здесь мы берём среднее значение между двумя дробями, а затем округляем его до одного знака после запятой, чтобы получить корректный ответ, который предлагается ученику.

```
1 (function () {
2   'use strict';
3   retryWhileError(function () {
4     NAinfo.requireApiVersion(0, 2);
5
6     let count = sl(0, 3);
7
8     let denominatorFirst = sl(2, 25);
9     let denominatorSecond = slKrome([denominatorFirst], 2, 25);
10    let numeratorFirst = sl(1, denominatorFirst - 1) + count *
        denominatorFirst;
11    let numeratorSecond = sl(1, denominatorSecond - 1) + count *
        denominatorSecond;
12
13    let frac1 = numeratorFirst / denominatorFirst;
14    let frac2 = numeratorSecond / denominatorSecond;
```

```

15
16     genAssert((frac1 - frac2).abs() >= 0.1, "разница между двумя значениями
должна составлять 0,1");
17
18     let text1 = numeratorFirst.texfrac(denominatorFirst);
19     let text2 = numeratorSecond.texfrac(denominatorSecond);
20
21     if (frac1 > frac2) {
22         [frac1, frac2] = [frac2, frac1];
23         [text1, text2] = [text2, text1];
24     }
25     let correctVal = ((frac1 + frac2) / 2);
26     let correct = correctVal.toFixed(1).ts();
27
28
29     let wrong = new Set();
30     while (wrong.size < 3) {
31         let noise = slKrome([0], -7, 7) * 0.1;
32         let candidate = +(correctVal + noise).toFixed(1);
33
34
35         if (candidate <= 0 || candidate > frac1 && candidate < frac2
candidate === +correctVal.toFixed(1)) {
36             continue
37         };
38
39         wrong.add(candidate.ts());
40     }
41
42     NATask.setTask({
43         text: 'Какое из следующих чисел заключено между числами ${' + text1 + '}$
и ${' + text2 + '}$? В ответе укажите номер правильного варианта.',
44         answers: correct,
45         wrongAnswers: Array.from(wrong)
46     });

```



```

47
48     AtoB(3, { autoLaTeX: true });
49 }, 1000);
50 })();
51
52 //zer00player
53 //https://oge.sdamgia.ru/test?likes=311420

```

Для проверки знаний учащегося необходимо не только предъявить правильный ответ, но и сформировать несколько правдоподобных «ловушек». В нашем проекте это реализовано через генерацию трёх ложных ответов. Ложные варианты создаются с помощью небольших шумов (**noise**), добавляемых к правильному значению.

```

1 let wrong = new Set();
2 while (wrong.size < 3) {
3     let noise = slKrome([0], -7, 7) * 0.1;
4     let candidate = +(correctVal + noise).toFixed(1);
5
6     if (candidate <= 0 || candidate > frac1 && candidate < frac2 ||
7         candidate === +correctVal.toFixed(1)) {
8         continue;
9     };
10    wrong.add(candidate.ts());
11 }

```

Таким образом, в итоговом задании всегда предлагается **4 варианта ответа**: один правильный и три ложных.

После задания параметров задачи и вариантов ответа вызывается функция **AtoB(3)**. Она отвечает за автоматическую генерацию списка вариантов с правильным ответом, расположенным случайным образом.

```

1 NAtask.setTask({
2     text: 'Какое из следующих чисел заключено между числами ${' + text1 + '}$ и
        ${' + text2 + '}$?',

```

```

3     answers: correct,
4     wrongAnswers: Array.from(wrong)
5 });
6
7 AtoB(3, { autoLaTeX: true });

```

Чтобы не окружать каждую формулу знаками  $\dots$ , используется параметр `{ autoLaTeX: true }`. Это позволяет сразу включать математические выражения (например, дроби и корни) прямо в текст задачи. В результате формулы корректно отображаются в интерфейсе без дополнительной ручной разметки.

В результате ученик видит задачу: «*Какое из следующих чисел заключено между числами ...?*», к которой автоматически предлагаются четыре варианта ответа.

```

1 (function () {
2     'use strict';
3     retryWhileError(function () {
4         NAinfo.requireApiVersion(0, 2);
5
6         let labels = window.smallLatinLetters.iz(2);
7         let labelA = labels[0];
8         let labelB = labels[1];
9
10        let variant = sl1();
11        let rand = [variant, 1 - variant].iz();
12        let conditionText = labelA + [' < ', ' > '][variant] + labelB;
13        let reverseconditionText = labelA + [' < ', ' > '][1 - variant] +
14        labelB;
15
16        let minus = sl(1, 99);
17        let plus = slKrome([minus], 1, 99);
18        let denominatorMinus = slKrome([minus, plus], 1, 99);
19        let denominatorPlus = slKrome([minus, plus, denominatorMinus], 1, 99);

```

```

20     let wrongForm = '';
21     let correctForms = [
22         labelA + ' - ' + minus + [' < ', ' > '][variant] + labelB + ' - ' +
minus,
23         labelA + ' + ' + plus + [' < ', ' > '][variant] + labelB + ' + ' +
plus,
24         labelA.texfrac(denominatorPlus) + [' < ', ' > '][variant] +
labelB.texfrac(denominatorPlus),
25     ];
26     wrongForm = '-' + labelA.texfrac(denominatorMinus) + [' < -', ' >
-'][variant] + labelB.texfrac(denominatorMinus);
27
28     NATask.setTask({
29         text: 'Какое из данных утверждений ' + ['не', ''][rand] + 'верно, если
$' + [conditionText, reverseconditionText][rand] + '$?',
30         answers: wrongForm,
31         wrongAnswers: correctForms
32     });
33
34     AtoB(3, { autoLaTeX: true });
35 }, 1000);
36 })();
37
38 //zer00player
39 //https://oge.sdamgia.ru/problem?id=205843

```

```

1 (function () {
2     'use strict';
3     retryWhileError(function () {
4         NAinfo.requireApiVersion(0, 2);
5
6         let count = sl(0, 3);
7
8         let denominatorFirst = sl(2, 25);
9         let denominatorSecond = slKrome([denominatorFirst], 2, 25);

```

```

10     let numeratorFirst = sl(1, denominatorFirst - 1) + count *
denominatorFirst;
11     let numeratorSecond = sl(1, denominatorSecond - 1) + count *
denominatorSecond;
12
13     let frac1 = numeratorFirst / denominatorFirst;
14     let frac2 = numeratorSecond / denominatorSecond;
15
16     genAssert((frac1 - frac2).abs() >= 0.1, "разница между двумя значениями
должна составлять 0,1");
17
18     let text1 = numeratorFirst.texfrac(denominatorFirst);
19     let text2 = numeratorSecond.texfrac(denominatorSecond);
20
21     if (frac1 > frac2) {
22         [frac1, frac2] = [frac2, frac1];
23         [text1, text2] = [text2, text1];
24     }
25     let correctVal = ((frac1 + frac2) / 2);
26     let correct = correctVal.toFixed(1).ts();
27
28
29     let wrong = new Set();
30     while (wrong.size < 3) {
31         let noise = slKrome([0], -7, 7) * 0.1;
32         let candidate = +(correctVal + noise).toFixed(1);
33
34
35         if (candidate <= 0  candidate > frac1 && candidate < frac2
candidate === +correctVal.toFixed(1)) {
36             continue
37         };
38
39         wrong.add(candidate.ts());
40     }

```

```

41
42   NATask.setTask({
43     text: 'Какое из следующих чисел заключено между числами  $\{$ ' + text1 + ' $\}$ 
и  $\{$ ' + text2 + ' $\}$ ? В ответе укажите номер правильного варианта.',
44     answers: correct,
45     wrongAnswers: Array.from(wrong)
46   });
47
48   AtoB(3, { autoLaTeX: true });
49 }, 1000);
50 })();
51
52 //zer00player
53 //https://oge.sdangia.ru/test?likes=311420

```

```

1 (function () {
2   'use strict';
3   retryWhileError(function () {
4     NAinfo.requireApiVersion(0, 2);
5
6     let key = "317132";
7     let preference = ['frac', 'square'];
8     let rand = getListedPreference(key, preference.map((pref, index) => ({
9       preference: pref,
10      preferenceValue: index
11    })), sl(preference.length - 1));
12
13     let countDrob = sl(1, 5);
14     let denominator = sl(2, 25);
15     let numerator = sl(1, denominator - 1);
16
17     let numDrob = countDrob * denominator + numerator;
18     let valueDrob = numDrob / denominator;
19
20     genAssert(!(valueDrob * 100).isAlmostInteger(), "дробь должна иметь 2

```

```

или более знака после запятой");
21
22     let exprStrDrob = numDrob.texfrac(denominator);
23
24     let countSqrt = sl(5, 25);
25     let numSqrt = countSqrt * countSqrt + denominator;
26
27     genAssert(!numSqrt.isPolnKvadr(), "корень не должен быть полным
квадратом");
28
29     let valueSqrt = numSqrt.sqrt();
30
31     let value = [valueDrob, valueSqrt][rand];
32     let step = [0.1, 1][rand];
33     let format = rand === 0 ? x => ((x * 10).round() / 10).ts() : x => x;
34
35     let start = Math.floor(value / step) * step;
36     let end = start + step;
37     let correct = `[${format(start)}; ${format(end)}]`;
38
39     let wrongAnswers = new Set();
40     wrongAnswers.add(correct);
41
42     let tries = 0;
43     while (wrongAnswers.size < 4 && tries < 50) {
44         let offset = slKrome([0], -3, 3);
45         let fakeStart = start + offset * step;
46         let fakeEnd = fakeStart + step;
47         if (fakeStart < 0) continue;
48
49         let fake = `[${format(fakeStart)}; ${format(fakeEnd)}]`;
50         wrongAnswers.add(fake);
51         tries++;
52     }
53     wrongAnswers.delete(correct);

```

```

54
55     NAtask.setTask({
56         text: 'Какому из данных промежутков принадлежит число $' + [exprStrDrob,
57             '\\sqrt{' + numSqrt + '}'][rand] + '$? В ответе укажите номер правильного
58             варианта.',
59         answers: correct,
60         wrongAnswers: Array.from(wrongAnswers),
61         preference: preference,
62     });
63
64     AtoB(3, { autoLaTeX: true });
65 }, 1000);
66 }));
67 //zer00player
68 //https://oge.sdamgia.ru/test?likes=317132
69 //https://oge.sdamgia.ru/test?likes=317223

```

```

1 (function () {
2     'use strict';
3     retryWhileError(function () {
4         NAinfo.requireApiVersion(0, 2);
5
6         let key = "337311";
7         let preference = ['frac', 'square'];
8         let rand = getListedPreference(key, preference.map((pref, index) => ({
9             preference: pref,
10            preferenceValue: index
11        })), sl(preference.length - 1));
12         let randMinus = [-1, 1].iz();
13
14         let countDrob = sl(3, 12);
15         let denominator = sl(2, 25);
16         let numerator = sl(1, denominator - 1);
17
18         let numDrob = countDrob * denominator + numerator;

```

```

19     let valueDrob = (numDrob / denominator) * randMinus;
20     let exprStrDrob = (randMinus * numDrob).texfrac(denominator);
21
22     let countSqrt = sl(5, 25);
23     let numSqrt = countSqrt * countSqrt + denominator;
24
25     genAssert(!numSqrt.isPolnKvadr(), "корень не должен быть полным
квадратом");
26
27     let valueSqrt = numSqrt.sqrt();
28
29     let value = [valueDrob, valueSqrt][rand];
30
31     let floor = Math.floor(value);
32     let correct = `${floor}` + '\\mbox{ и }' + `${floor + 1}`;
33
34     let wrongAnswers = new Set();
35     let usedOffsets = new Set([0]);
36
37     while (wrongAnswers.size < 3) {
38         let offset = slKrome([0], -3, 3);
39         if (usedOffsets.has(offset)) {
40             continue
41         };
42         usedOffsets.add(offset);
43
44         let start = floor + offset;
45         let variant = `${start}` + '\\mbox{ и }' + `${start + 1}`;
46         wrongAnswers.add(variant);
47     }
48
49     NAtask.setTask({
50         text: 'Между какими целыми числами заключено число $' + [exprStrDrob,
'\\sqrt{' + numSqrt + '}'][rand] + '$? В ответе укажите номер правильного
варианта.',

```



```

51     answers: correct,
52     wrongAnswers: Array.from(wrongAnswers),
53     preference: preference,
54 });
55
56     AtoB(3, { autoLaTeX: true });
57 }, 1000);
58 })();
59 //zer00player
60 //https://oge.sdangia.ru/test?likes=337311

```

```

1 (function () {
2     'use strict';
3     retryWhileError(function () {
4         NAinfo.requireApiVersion(0, 2);
5
6         let labels = window.smallLatinLetters.iz(2);
7         let labelA = labels[0];
8         let labelB = labels[1];
9
10        let plusOrMinus = sl1();
11        let isALessThanB = sl1();
12
13        let word = ['положительные', 'отрицательные'][plusOrMinus];
14
15        let aSign = labelA + [' > ', ' < '][isALessThanB] + labelB;
16
17        let numerator = sl(1, 9);
18        let exprA = numerator.texfrac(labelA);
19        let exprB = numerator.texfrac(labelB);
20
21        let correct = exprA + [' < ', ' > '][isALessThanB] + exprB;
22
23        let wrong = [
24            exprA + ' < ' + exprB,

```

```

25     exprA + ' > ' + exprB,
26     exprA + ' = ' + exprB,
27     '\\mboxневозможно{ определить}'
28 ].filter(ans => ans !== correct);
29
30     NATask.setTask({
31         text: 'Сравните числа, если $' + labelA + '$, $' + labelB + '$ - ' +
32         word + ' числа и $' + aSign + '$. В ответе укажите номер правильного
33         варианта.',
34         answers: correct,
35         wrongAnswers: wrong
36     });
37     AtoB(3, { autoLaTeX: true });
38 }(), 1000);
39 //zer00player
40 //https://oge.sdangia.ru/test?pid=337381

```

```

1 (function () {
2     'use strict';
3     retryWhileError(function () {
4         NAinfo.requireApiVersion(0, 2);
5
6         let key = "369728";
7         let preference = ['frac', 'sqrt', 'advancedFrac', 'advancedSqrt'];
8         let rand = getListedPreference(key, preference.map((pref, index)
=> ({
9             preference: pref,
10            preferenceValue: index
11        })), sl(preference.length - 1));
12
13        let randMinus = [-1, 1].iz();
14        let section = ['отрезку', 'промежутку'].iz();
15

```

```

16     let countFrac = sl(3, 12);
17     let denominator = sl(2, 25);
18     let numerator = sl(1, denominator - 1);
19     let numFrac = countFrac * denominator + numerator;
20     let valueFrac = (numFrac / denominator) * randMinus;
21     let correctFrac = (randMinus * numFrac).texfrac(denominator);
22
23     let countSqrt = sl(5, 25);
24     let numSqrt = countSqrt.sqr() + denominator;
25     let valueSqrt = numSqrt.sqrt();
26
27     genAssert(!numSqrt.isPolnKvadr(), "корень не должен быть полным
квадратом ");
28
29     let correctSqrt = '\\sqrt{' + numSqrt + '}';
30
31     let value = [valueFrac, valueSqrt, valueFrac, valueSqrt][rand];
32     let correctExpr = [correctFrac, correctSqrt, correctFrac,
correctSqrt][rand];
33     let floor = Math.floor(value);
34     let intervalText = `[${floor}; ${floor + 1}]`;
35
36     let wrongAnswers = [];
37     let ifFracAdvanced = '';
38
39     if (rand === 0 || rand === 2) { //дробь
40
41         let usedNumerators = [numFrac];
42         let variants = [{ val: valueFrac, expr: correctFrac }];
43
44         while (variants.length < 4) {
45             let delta = sl(1, 4).pm();
46             let newCount = countFrac + delta;
47             if (newCount < 1) {
48                 continue

```

```

49         };
50
51         let fakeNumerator = newCount * denominator + numerator;
52         if (usedNumerators.includes(fakeNumerator)) {
53             continue
54         };
55         usedNumerators.push(fakeNumerator);
56
57         let val = fakeNumerator / denominator;
58         if (randMinus === -1) {
59             val *= -1
60         };
61         if (val >= floor && val <= floor + 1) {
62             continue
63         };
64
65         let expr = (randMinus *
fakeNumerator).texfrac(denominator);
66         variants.push({ val, expr });
67     }
68     variants.shuffle();
69     correctExpr = correctFrac;
70     wrongAnswers = variants.filter(v => v.expr !==
correctFrac).map(v => v.expr);
71
72     if (rand === 2) {
73         ifFracAdvanced = variants.map(x => '$' + x.expr +
'$').join(', ') + ' ';
74     }
75     } else if (rand === 1 || rand === 3) { // корни
76
77         let usedSqrts = [numSqrt];
78         let variants = [{ val: valueSqrt, expr: correctSqrt }];
79
80         while (variants.length < 4) {

```

```

81         let offset = sl(1, 10).pm();
82         let fakeRoot = (countSqrt + offset).sqr() + sl(1, 10);
83         if (fakeRoot <= 0  usedSqrts.includes(fakeRoot)) {
84             continue
85         };
86
87         let val = fakeRoot.sqr();
88         if (val >= floor && val <= floor + 1) {
89             continue
90         };
91
92         let expr = `\\sqrt{${fakeRoot}}`;
93         variants.push({ val, expr });
94     }
95     variants.shuffle();
96     correctExpr = correctSqrt;
97     wrongAnswers = variants.filter(v => v.expr !==
correctSqrt).map(v => v.expr);
98
99     if (rand === 3) {
100         ifFracAdvanced = variants.map(x => '$' + x.expr +
'$').join(', ') + ' ';
101     }
102
103 }
104 let dataName = ['имеющихся', 'данных', '', ''][rand];
105
106 NATask.setTask({
107     text: 'Какое из ' + dataName + ' чисел ' + ifFracAdvanced +
'принадлежит ' + section + ' ${' + intervalText + '} $? В ответе укажите
номер правильного варианта.',
108     answers: correctExpr,
109     wrongAnswers: wrongAnswers,
110     preference: preference,
111 });

```

```

112
113         AtoB(3, { autoLaTeX: true });
114     }, 1000);
115 }));
116 //zer00player
117 //https://oge.sdamgia.ru/test?likes=369728
118 //https://oge.sdamgia.ru/test?likes=337389

```

```

1 (function () {
2     'use strict';
3     retryWhileError(function () {
4         NAinfo.requireApiVersion(0, 2);
5
6         let rand = sl1();
7
8         let numberSqrtFisrt = slKrome([4, 9, 16], 2, 20);
9         let numberSqrtSecond = numberSqrtFisrt * slKrome([numberSqrtFisrt, 4,
10         9, 16, 25], 2, 35);
11
12         genAssert(!numberSqrtSecond.isPolnKvadr(), "корень не должен быть полным
13         квадратом");
14
15         let numberSqrtWithRatio = slKrome([4, 9, 16, numberSqrtFisrt], 2, 20);
16
17         let val1 = [numberSqrtFisrt.sqrt(), numberSqrtWithRatio *
18         numberSqrtFisrt.sqrt()][rand];
19         let val2 = [numberSqrtSecond.sqrt(), numberSqrtFisrt *
20         numberSqrtWithRatio.sqrt()][rand];
21
22         let minV = Math.min(val1, val2);
23         let maxV = Math.max(val1, val2);
24
25         // Количество целых чисел между ними
26         let left = Math.ceil(minV);
27         let right = Math.floor(maxV);

```

```

24     let count = Math.max(0, right - left) + 1;
25
26     let text1 = [`${\sqrt{\${numberSqrtFisrt}}}`,
    `${numberSqrtWithRatio}\sqrt{\${numberSqrtFisrt}}`][rand];
27     let text2 = [`${\sqrt{\${numberSqrtSecond}}}`,
    `${numberSqrtFisrt}\sqrt{\${numberSqrtWithRatio}}`][rand];
28
29     NATask.setTask({
30         text:
31             'Сколько целых чисел расположено между ${' + text1 + '}$ и ${' + text2
    + '}$?',
32         answers: count,
33     });
34     NATask.modifiers.allDecimalsToStandard();
35 }, 2000);
36 })();
37 //zer00player
38 //https://oge.sdamgia.ru/test?likes=401572

```

## 6. Стереометрия

В этой главе решается проблема отрисовки трёхмерных многогранников в JavaScript. Поднимается вопрос об использовании нейросетей для написания программного обеспечения. Рассказывается о применении ООП для написания шаблонов по теме «Стереометрия» и приводится пример написания шаблона с чертежом.

### 6.1. Разработка библиотек с помощью Gpt-Chat

На данный момент в языке JavaScript отсутствуют встроенные средства для изображения трёхмерных фигур. И существует только одна подходящая библиотека `Three.js`, которая могла бы выполнить проецирование координат фигуры на плоскость с учётом положения наблюдателя. Но при этом для создания любого объекта необходима не только камера, но и сцена с рендерингом, что значительно замедляет работу проекта.

```
1 import * as THREE from 'three';
2
3 // Создать сцену, камеру и рендер
4 const scene = new THREE.Scene();
5 const camera = new THREE.PerspectiveCamera(75, window.innerWidth /
    window.innerHeight, 0.1, 1000);
6 camera.position.z = 2;
7 const renderer = new THREE.WebGLRenderer();
8 renderer.setSize(window.innerWidth, window.innerHeight);
9
10 // Создать геометрию куба
11 const geometry = new THREE.BoxGeometry(1, 1, 1);
12
13 // Создать материал куба
14 const material = new THREE.MeshBasicMaterial({ color: 0x00ff00 });
15
16 // Создать куб
```



```

17 const cube = new THREE.Mesh(geometry, material);
18
19 // Добавить куб в сцену
20 scene.add(cube);
21
22 // Отрендерить сцену
23 renderer.render(scene, camera);

```

### Листинг 1: Код, необходимый для отрисовки куба

Другие подобные ей библиотеки проводят проецирование на плоскость с поворотом только вокруг осей  $Ox$  и  $Oz$ , что при отрисовке более сложных чертежей является недостаточным.

С учётом прогресса современных технологий встал закономерный вопрос, сможет ли нейросеть ChatGPT 3.5 сгенерировать код проектора. Ниже представлен диалог с нейросетью:

write a 2D to 3D projector in JS without using libraries. input: point = {x,y,z}, camera= {x,y,z, angleX, angleY, angleZ, scale}

```

// Define the point in 2D
const point = { x: 100, y: 50, z: 0 };

// Define the camera parameters
const camera = { x: 0, y: 0, z: 0, angleX: 0, angleY: 0, angleZ: 0,
  scale: 1 };

// Function to project 2D point to 3D
function project2Dto3D(point, camera) {
  // Apply rotation around X axis
  const rotatedY = point.y * Math.cos(camera.angleX) - point.z *
Math.sin(camera.angleX);
  const rotatedZ = point.y * Math.sin(camera.angleX) + point.z *

```

```

Math.cos(camera.angleX);

    // Apply rotation around Y axis
    const rotatedX = point.x * Math.cos(camera.angleY) + rotatedZ *
Math.sin(camera.angleY);
    const rotatedZ2 = -point.x * Math.sin(camera.angleY) + rotatedZ *
Math.cos(camera.angleY);

    // Apply rotation around Z axis
    const rotatedX2 = rotatedX * Math.cos(camera.angleZ) - rotatedY *
Math.sin(camera.angleZ);
    const rotatedY2 = rotatedX * Math.sin(camera.angleZ) + rotatedY *
Math.cos(camera.angleZ);

    // Apply scaling
    const scaledX = rotatedX2 * camera.scale;
    const scaledY = rotatedY2 * camera.scale;

    // Translate to camera position
    const translatedX = scaledX + camera.x;
    const translatedY = scaledY + camera.y;
    const translatedZ = rotatedZ2 * camera.scale + camera.z;

    return { x: translatedX, y: translatedY, z: translatedZ };
}

// Project the point from 2D to 3D
const projectedPoint = project2Dto3D(point, camera);

// Output the result
console.log(»Projected 3D point:», projectedPoint);

```

shorten function project2Dto3D

```

function project2Dto3D(point, camera) {
    const { x, y, z } = point;
    const { angleX, angleY, angleZ, scale, x: cx, y: cy, z: cz } =
camera;

    const cosX = Math.cos(angleX);
    const sinX = Math.sin(angleX);
    const cosY = Math.cos(angleY);
    const sinY = Math.sin(angleY);
    const cosZ = Math.cos(angleZ);
    const sinZ = Math.sin(angleZ);

    const rx = x * cosY * cosZ - y * cosY * sinZ + z * sinY;
    const ry = (x * (sinX * sinY * cosZ + cosX * sinZ) + y * (sinX *
sinY * sinZ - cosX * cosZ) + z * sinX * cosY) * scale;
    const rz = (-x * (cosX * sinY * cosZ - sinX * sinZ) - y * (cosX *
sinY * sinZ + sinX * cosZ) + z * cosX * cosY) * scale;

    return { x: rx + cx, y: ry + cy, z: rz + cz };
}

```

add comments in code

```

function project2Dto3D(point, camera) {
    // Extract point coordinates
    const { x, y, z } = point;
    // Extract camera parameters
    const { angleX, angleY, angleZ, scale, x: cx, y: cy, z: cz } =
camera;

    // Precompute trigonometric values for rotation angles

```

```

const cosX = Math.cos(angleX);
const sinX = Math.sin(angleX);
const cosY = Math.cos(angleY);
const sinY = Math.sin(angleY);
const cosZ = Math.cos(angleZ);
const sinZ = Math.sin(angleZ);

// Apply rotation around X axis
const rx = x * cosY * cosZ - y * cosY * sinZ + z * sinY;
// Apply rotation around Y axis and scale
const ry = (x * (sinX * sinY * cosZ + cosX * sinZ) + y * (sinX *
sinY * sinZ - cosX * cosZ) + z * sinX * cosY) * scale;
// Apply rotation around Z axis and scale
const rz = (-x * (cosX * sinY * cosZ - sinX * sinZ) - y * (cosX *
sinY * sinZ + sinX * cosZ) + z * cosX * cosY) * scale;

// Translate to camera position
const translatedX = rx + cx;
const translatedY = ry + cy;
const translatedZ = rz + cz;

return { x: translatedX, y: translatedY, z: translatedZ };
}

```

За несколько шагов удалось получить корректный, оптимизированный код.

## 6.2. Применение объектно-ориентированного программирования для разработки шаблонов

Банк заданий содержит большое количество разнообразных задач по теме «Стереометрия». Поэтому одной из первостепенных задач было сократить код шаблонов и исключить вычислительные ошибки. Для этого были разработаны классы многогранников, которые содержат в себе длины рёбер, объем, площади основа-

ний, а так же тернарную матрицу связности и канонические координаты вершин.

Матрица может содержать значения: 1, 0, либо специальное значение, указывающие на отображение ребра пунктиром.

Пример канонической матрицы связей:

```

1  [  [1],
2      [0, 1],
3      [1, 0, 1],
4      [0, 0, 0, 1],
5      [1, 0, 0, 0, 1],
6      [0, 1, 0, 0, 0, 1],
7      [0, 0, 1, 0, 1, 0, 1],
8  ];

```

#### Листинг 2: Каноническая матрица связей для параллелепипеда

Мы можем опускать конец строки матрицы, если он состоит только из нулей, и главную диагональ матрицы (на ней всегда стоят нули).

**Определение.** Каноническим положением будем называть такое расположение многогранника, когда его высота, проходящая через центр масс его основания, совпадает с осью аппликат и началом координат делится пополам (Рис. 1- 2).

При таком расположении начало координат можно расположить в центре иллюстрации. Тогда чертёж не будет чрезмерно смещён ни в одну из сторон.

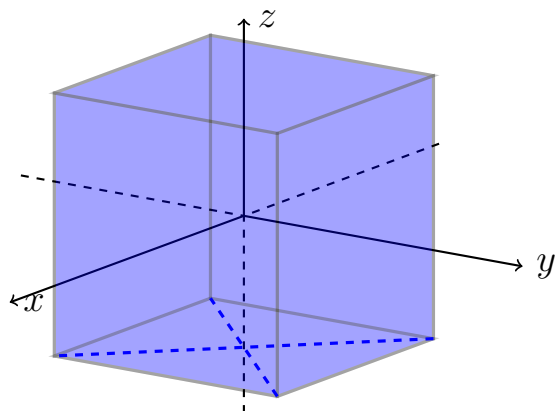


Рис. 1: Каноническое положение для параллелепипеда

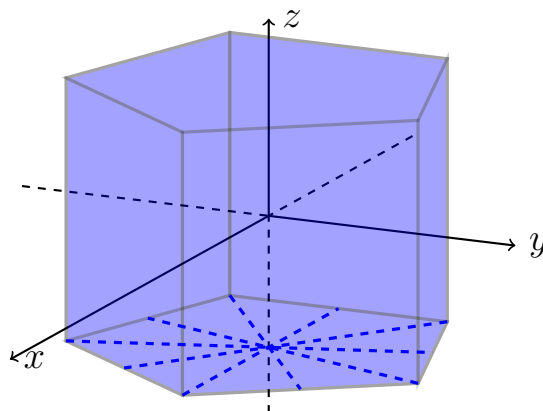


Рис. 2: Каноническое положение для правильной пятиугольной призмы

## 6.3. Вспомогательные функции

### 6.3.1. Функции для работы с координатами

```
function verticesInGivenRange(vertex, startX, finishX, startY, finishY)
```

Возвращает `true`, если двумерная координата точки `vertex` вида `{x,y}` находится в некоторой прямоугольной области, иначе `false`.

```
function autoScale(vertex3D, camera, vertex2D, startX, finishX, startY, finishY, step, maxScale)
```

Увеличивает свойство объекта `camera.scale` до тех пор, пока все двумерные координаты `vertex2D` вида `{x,y}` находится в некоторой прямоугольной области. `step` по умолчанию 0.1.

```
function distanceFromPointToSegment(point, segmentStart, segmentEnd)
```

Возвращает длину перпендикуляра между двумерной точкой `point` вида `{x,y}` и прямой, проходящей через точки `segmentStart` и `segmentEnd`.

### 6.3.2. Функции для работы с canvas

```
CanvasRenderingContext2D.prototype.drawFigure = function(vertex, matrixConnections)
```

Соединяет линиями точки массива `vertex` с элементами `{x,y}` в соответствии с матрицей связей `matrixConnections`, которая является массивом, содержащим в себе 0, 1 или массив `step`, указывающий на отрисовку пунктиром.

Пример матрицы связей:

```
let matrixConnections = [
  [1],
  [strok, strok],
  [0, 0, strok],
  [1, 0, 0, 1],
  [0, 1, 0, 1, 1]
```

```
];
```

```
CanvasRenderingContext2D.prototype.drawFigureVer2 = function(  
vertex, matrixConnections)
```

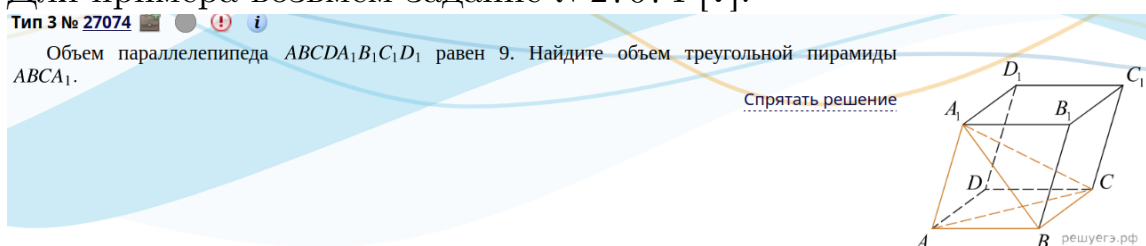
Соединяет линиями точки массива `vertex` с элементами `{x,y}` в соответствии с матрицей связей `matrixConnections`. Эта матрица представляет собой объект, где каждое числовое поле соответствует номеру вершины в массиве `vertex`. В каждом поле находится массив номеров других вершин, с которыми должна быть соединена данная вершина.

Пример матрицы связей:

```
let matrixConnections = {  
  0: [1, [3, stroke], 5],  
  2: [1, [3, stroke], 7],  
  4: [[3, stroke], 5, 7],  
  9: [1, 8, 10],  
  11: [8, 10, 12],  
  13: [5, 8, 12],  
  15: [7, 10, 12],  
};
```

## 6.4. Этапы разработки шаблоны с вспомогательным чертежом по теме «Стереометрия»

Для примера возьмём задание №27074 [?].



Заготовка шаблона имеет вид:

```

1 (function () {
2   retryWhileError(function () {
3     NAinfo.requireApiVersion(0, 2);
4
5     let paint1 = function (ctx) {
6       };
7
8     NATask.setTask({
9       text: 'Объем параллелепипеда ABCDA_1B_1C_1D_1 равен 9. Найдите объем
треугольной пирамиды ABCA_1.',
10      answers: 0,
11      author: ['Суматохина Александра']
12    });
13    NATask.modifiers.addCanvasIllustration({
14      width: 400,
15      height: 400,
16      paint: paint1,
17    });
18  }, 100000);
19 })();

```

1. Создадим объект класса **Parallelepiped** со случайной высотой, шириной и глубиной в заданном диапазоне.

```

1 (function () {
2   retryWhileError(function () {
3     NAinfo.requireApiVersion(0, 2);
4
5     let par = new Parallelepiped({
6       depth: sl(10, 50),
7       height: sl(10, 50),
8       width: sl(10, 50),
9     });
10

```



```

11     let paint1 = function (ctx) {
12     };
13
14     NATask.setTask({
15         text: 'Объем параллелепипеда ABCDA_1B_1C_1D_1 равен 9. Найдите
объем треугольной пирамиды ABCA_1.',
16         answers: 0,
17         author: ['Суматохина Александра']
18     });
19     NATask.modifiers.addCanvasIllustration({
20         width: 400,
21         height: 400,
22         paint: paint1,
23     });
24 }, 100000);
25 })();

```

2. Определим переменную **camera**, которая будет отвечать за положение наблюдателя. Спроецируем канонические координаты параллелепипеда на двумерную плоскость при помощи функции **project3DTo2D**, отмасштабируем полученные координаты так, чтобы они занимали максимально заполняли иллюстрацию, функцией **autoScale**.

```

1 (function () {
2     retryWhileError(function () {
3         NAinfo.requireApiVersion(0, 2);
4
5         let par = new Parallelepiped({
6             depth: sl(10, 50),
7             height: sl(10, 50),
8             width: sl(10, 50),
9         });
10
11         let camera = {

```

```

12         x: 0,
13         y: 0,
14         z: 0,
15         scale: 5,
16
17         rotationX: -Math.PI / 2 + Math.PI / 14,
18         rotationY: 0,
19         rotationZ: 2 * Math.PI / 3,
20     };
21
22     let point2DPar = par.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));
23
24     autoScale(par.verticesOfFigure, camera, point2DPar, {
25         startX: -180,
26         finishX: 160,
27         startY: -160,
28         finishY: 160,
29         maxScale: 50,
30     });
31
32     point2DPar = par.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));
33
34     let paint1 = function (ctx) {
35     };
36
37     NATask.setTask({
38         text: 'Объем параллелепипеда ABCDA_1B_1C_1D_1 равен 9.
Найдите объем треугольной пирамиды ABСA_1.',
39         answers: 0,
40         author: ['Суматохина Александра']
41     });
42     NATask.modifiers.addCanvasIllustration({
43         width: 400,

```

```

44         height: 400,
45         paint: paint1,
46     });
47     }, 100000);
48 })();

```

3. Перемещаемся в середину иллюстрации. Отрисовываем фигуру функцией `drawFigure`, передав в неё матрицу связей для параллелепипеда.

```

1  (function () {
2      retryWhileError(function () {
3          NAinfo.requireApiVersion(0, 2);
4
5          let par = new Parallelepiped({
6              depth: sl(10, 50),
7              height: sl(10, 50),
8              width: sl(10, 50),
9          });
10
11         let camera = {
12             x: 0,
13             y: 0,
14             z: 0,
15             scale: 5,
16
17             rotationX: -Math.PI / 2 + Math.PI / 14,
18             rotationY: 0,
19             rotationZ: 2 * Math.PI / 3,
20         };
21
22         let point2DPar = par.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));
23
24         autoScale(par.verticesOfFigure, camera, point2DPar, {
25             startX: -180,

```

```

26         finishX: 160,
27         startY: -160,
28         finishY: 160,
29         maxScale: 50,
30     });
31
32     point2DPar = par.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));
33
34     let paint1 = function (ctx) {
35         let h = 400;
36         let w = 400;
37         ctx.translate(h / 2, w / 2);
38         ctx.lineWidth = 2;
39         ctx.strokeStyle = om.secondaryBrandColors;
40         let strok = [5, 4];
41         ctx.drawFigure(point2DPar, [
42             [strok],
43             [0, 1],
44             [strok, 0, 1],
45             [0, 0, 0, 1],
46             [strok, 0, 0, 0, 1],
47             [0, 1, 0, 0, 0, 1],
48             [0, 0, 1, 0, 1, 0, 1],
49         ]);
50     };
51
52     NAtask.setTask({
53         text: 'Объем параллелепипеда ABCDA_1B_1C_1D_1 равен 9.
Найдите объем треугольной пирамиды ABСA_1.',
54         answers: 0,
55         author: ['Суматохина Александра']
56     });
57     NAtask.modifiers.addCanvasIllustration({
58         width: 400,

```

```

59         height: 400,
60         paint: paint1,
61     });
62 }, 100000);
63 })();

```

4. Далее вырезаем из условия значения и заменяем их данными из класса. Впишем ответ. Обособляем имена фигур в `$...$`. Добавляем буквы на вершины параллелепипеда. Добавляем модификаторы `NAtask.modifiers.assertSaneDecimals()` (исключает нецелый ответ) и `NAtask.m` (заменяет все буквы в задании на случайные).

```

1  (function() {
2      retryWhileError(function() {
3          NAinfo.requireApiVersion(0, 2);
4
5          let par = new Parallelepiped({
6              depth: sl(10, 50),
7              height: sl(10, 50),
8              width: sl(10, 50),
9          });
10
11         let pyr = new Pyramid({
12             height: par.height,
13             baseArea: 0.5 * par.baseArea,
14         });
15
16         let camera = {
17             x: 0,
18             y: 0,
19             z: 0,
20             scale: 5,
21
22             rotationX: -Math.PI / 2 + Math.PI / 14,

```

```

23     rotationY: 0,
24     rotationZ: 2 * Math.PI / 3,
25 };
26
27 let point2DPar = par.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));
28
29 autoScale(par.verticesOfFigure, camera, point2DPar, {
30     startX: -180,
31     finishX: 160,
32     startY: -160,
33     finishY: 160,
34     maxScale: 50,
35 });
36
37 point2DPar = par.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));
38
39 let letter = ['A', 'B', 'C', 'D', '⊠D', '⊠A', '⊠B', '⊠C',];
40
41 let paint1 = function(ctx) {
42     let h = 400;
43     let w = 400;
44     ctx.translate(h / 2, w / 2);
45     ctx.lineWidth = 2;
46     ctx.strokeStyle = om.secondaryBrandColors;
47     let strok = [5, 4];
48     ctx.drawFigure(point2DPar, [
49         [strok],
50         [0, 1],
51         [strok, 0, 1],
52         [0, 0, 0, 1],
53         [strok, 0, 0, 0, 1],
54         [0, 1, 0, 0, 0, 1],
55         [0, 0, 1, 0, 1, 0, 1],

```

```

56     });
57
58     ctx.font = "25px liberation_sans";
59     point2DPar.forEach((elem, i) => ctx.fillText(letter[i],
elem.x, elem.y + ((i < point2DPar.length / 2) ? 15 : -10)));
60     });
61
62     NAtask.setTask({
63         text: 'Объем параллелепипеда $ABCD A_1 B_1 C_1 D_1$ равен
$'+par.volume+'$. Найдите объем треугольной пирамиды $ABCA_1$.',
64         answers: par.volume/6,
65         author: ['Суматохина Александра']
66     });
67
68     NAtask.modifiers.assertSaneDecimals();
69     NAtask.modifiers.variativeABC(letter);
70
71     NAtask.modifiers.addCanvasIllustration({
72         width: 400,
73         height: 400,
74         paint: paint1,
75     });
76 }, 100000);
77 })();

```

## Заключение

В ходе выполнения курсовой работы за 3 курс был покрыт открытый банк заданий ФИПИ по темам:

- Текстовые задачи (на смекалку) — 12 шаблонов принято.
- Текстовые задачи (проценты и дроби) — 29 шаблонов принято.
- Преобразования выражений — 29 шаблонов (25 принято 4 на внутреннем рецензировании).
- Задачи с прямыми — 10 шаблонов принято. (3 с рисунком и 7 на сравнение чисел )

В ядро проекта добавлены:

- Функции, упрощающие написание шаблонов по теме «Координатная прямая».
- $r^2$

А также сокращён технический долг проекта.

Все добавленные в проект задания можно использовать для составления контрольных работ, проведения текущего контроля знаний учащихся, подготовки к ЕГЭ. [?]

В будущем планируется добавить



# Приложение

```
1 (function() {
2   retryWhileError(function() {
3     NAinfo.requireApiVersion(0, 2);
4
5     let a = sl(2, 89);
6
7     let vertices = om.latbukv.iz(5);
8
9     let trapezoid = [vertices[0], vertices[1], vertices[4],
vertices[3]].randomReverse();
10
11    let paint1 = function(ctx) {
12      ctx.lineWidth = 2;
13      ctx.strokeStyle = om.secondaryBrandColors.iz();
14
15      ctx.drawLine(100, 80, 350, 80);
16      ctx.drawLine(10, 320, 270, 320);
17      ctx.drawLine(100, 80, 10, 320);
18      ctx.drawLine(270, 320, 350, 80);
19
20      ctx.strokeStyle = om.primaryBrandColors[0];
21      let middle = coordinatesMiddleOfSegment(270, 320, 350, 80);
22      ctx.drawLine(10, 320, middle[0], middle[1]);
23
24      ctx.strokeStyle = om.primaryBrandColors[1];
25      ctx.strokeInMiddleOfSegment(350, 80, middle[0], middle[1], 10);
26      ctx.strokeInMiddleOfSegment(270, 320, middle[0], middle[1], 10);
27
28      ctx.font = "23px liberation_sans";
29      ctx.fillText(vertices[0], 100, 80 - 10);
30      ctx.fillText(vertices[1], 350, 80 - 10);
31      ctx.fillText(vertices[2], 270, 320 + 20);
```

```

32     ctx.fillText(vertices[3], 10, 320 + 20);
33     ctx.fillText(vertices[4], middle[0] + 5, middle[1] + 5);
34
35 };
36
37 NATask.setTask({
38     text: 'Площадь параллелограмма $' + vertices.slice(0,
39         4).permuteCyclic(sl(0, 3)).join('') + '$ равна $' + a +
40         '$. Точка $' + vertices[4] + '$ - середина стороны $' +
41         vertices.slice(1, 3).shuffleJoin() +
42         '$. Найдите площадь трапеции $' + trapezoid.permuteCyclic(sl(0,
43         3)).join('') + '$.',
44     answers: a * 0.75,
45 });
46 NATask.modifiers.addCanvasIllustration({
47     width: 400,
48     height: 400,
49     paint: paint1,
50 });
51 }, 1000);
52 })();
53 //92

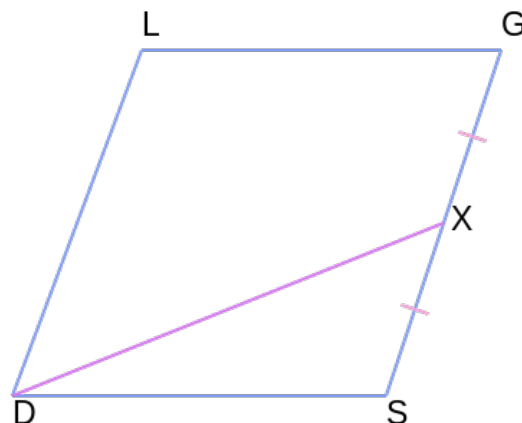
```

Листинг 3: 92.js

## Примеры генерируемых задач 92.js

Площадь параллелограмма  $DLGS$  равна 45. Точка  $X$  – середина стороны  $GS$ .  
Найдите площадь трапеции  $XGLD$ .

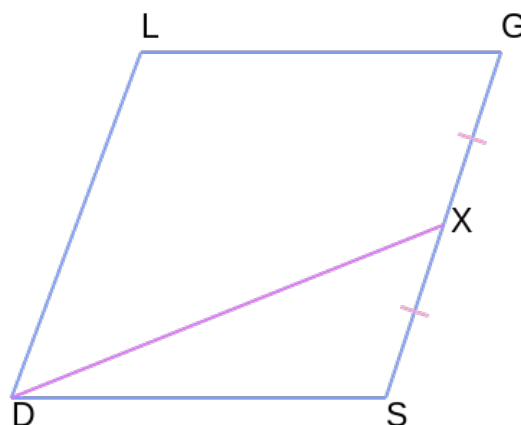
Ответ: 0,25



Задача. 1: Пример генерации задания с помощью 92.js

Площадь параллелограмма  $AHPJ$  равна 47. Точка  $G$  – середина стороны  $PJ$ . Найдите площадь трапеции  $PGAH$ .

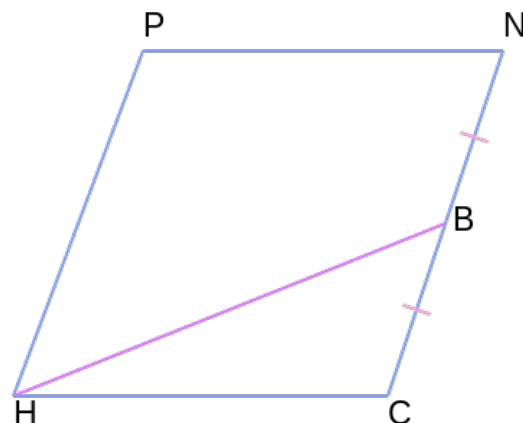
Ответ: 0,75



Задача. 2: Пример генерации задания с помощью 92.js

Площадь параллелограмма  $CHPN$  равна 16. Точка  $B$  – середина стороны  $CN$ . Найдите площадь трапеции  $HPNB$ .

Ответ: 0,25



Задача. 3: Пример генерации задания с помощью 92.js

```

1 (function() {
2   retryWhileError(function() {
3     lx_declareClarifiedPhrase('сторона', 'основания');
4     NAinfo.requireApiVersion(0, 2);
5
6     let pyr = new RegularPyramid({
7       height: sl(20, 50),
8       baseSide: sl(20, 40),
9       numberSide: 4
10    });
11
12    pyr.verticesOfFigure.push({
13      x: 0,
14      y: 0,
15      z: pyr.verticesOfFigure[0].z
16    });
17
18    let question = [
19      [sklonlxkand('боковое ребро'), pyr.sideEdge],
20      [sklonlxkand('объём'), pyr.volume],
21    ].shuffle();
22    question.unshift([sklonlxkand('сторона основания'), pyr.baseSide]);

```

```

23
24     let camera = {
25         x: 0,
26         y: 0,
27         z: 0,
28         scale: 5,
29
30         rotationX: -Math.PI / 2 + Math.PI / 14,
31         rotationY: 0,
32         rotationZ: [1, 2].iz() * Math.PI / 3,
33     };
34
35     let point2DPyr = pyr.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));
36
37     autoScale(pyr.verticesOfFigure, camera, point2DPyr, {
38         startX: -180,
39         finishX: 160,
40         startY: -160,
41         finishY: 160,
42         maxScale: 50,
43     });
44
45     point2DPyr = pyr.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));
46
47     let letters = ['A', 'B', 'C', 'D', 'S', 'O'];
48
49     let strok = [5, 4];
50
51
52     let paint1 = function(ctx) {
53         let h = 400;
54         let w = 400;
55         ctx.translate(h / 2, w / 2);

```

```

56     ctx.lineWidth = 2;
57     ctx.strokeStyle = om.secondaryBrandColors;
58     ctx.drawFigure(point2DPyr, [
59         [1],
60         [strok, strok],
61         [1, strok, strok],
62         [1, 1, strok, 1, 0, strok],
63     ]);
64
65     ctx.font = "30px liberation_sans";
66     point2DPyr.forEach((elem, i) => ctx.fillText(letters[i], elem.x,
67         elem.y + ((i != point2DPyr.length - 2) ? 15 : -
68             10)));
69
70     NATask.setTask({
71         text: 'В правильной четырёхугольной пирамиде $SABCD$ с ' + 'основанием
72 $ABCD$ ' +
73         [question[0][0].ie + ' рав' + ['ен', 'на',
74 'но']][question[0][0].rod] + ' $' + question[0][1].pow(2).texsqrt(1) +
75 '$',
76         question[1][0].ie + ' рав' + ['ен', 'на', 'но']][question[1][0].rod]
77 + ' $' + question[1][1].pow(2).texsqrt(1) + '$'
78         ].shuffleJoin(', ') +
79         '. Найдите ' + question[2][0].ve + ' пирамиды.',
80         answers: question[2][1],
81         author: ['Суматохина Александра'],
82     });
83     NATask.modifiers.variativeABC(letters);
84     NATask.modifiers.multiplyAnswerBySqrt(13);
85     NATask.modifiers.allDecimalsToStandard(true);
86     NATask.modifiers.assertSaneDecimals();
87     NATask.modifiers.addCanvasIllustration({
88         width: 400,
89         height: 400,

```

```

86     paint: paint1,
87   });
88 },10);
89 })();
90 //https://ege314.ru/8-stereometriya-ege/reshenie-3011/
91 //3011

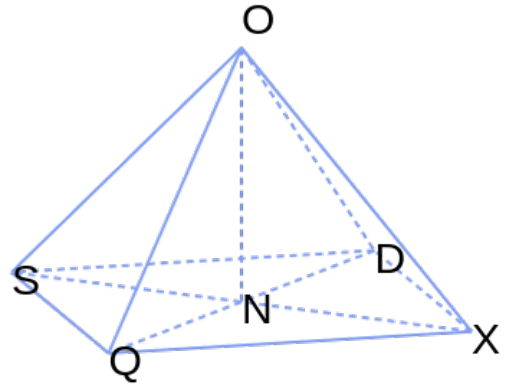
```

Листинг 4: 3011.js

### Примеры генерируемых задач 3011.js

В правильной четырёхугольной пирамиде  $OQSDX$  с основанием  $QSDX$  боковое ребро равно  $\sqrt{1489,5}$ , сторона основания равна 39. Найдите объём пирамиды.

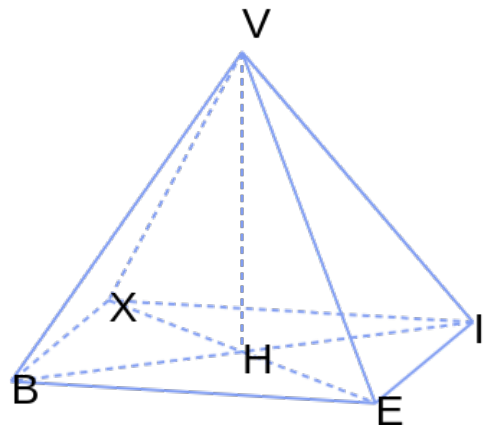
Ответ: 13689



Задача. 4: Пример генерации задания с помощью 3011.js

В правильной четырёхугольной пирамиде  $VEBXI$  с основанием  $EBXI$  боковое ребро равно  $\sqrt{848,5}$ , сторона основания равна 27. Найдите объём пирамиды.

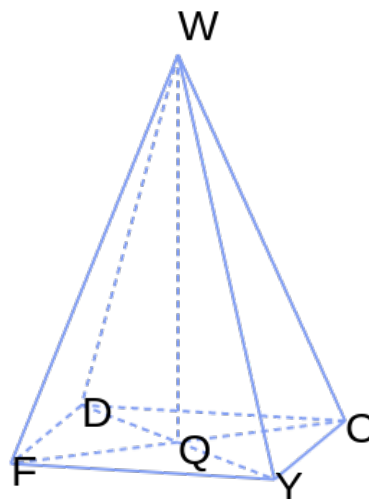
Ответ: 5346



Задача. 5: Пример генерации задания с помощью 3011.js

В правильной четырёхугольной пирамиде  $WYFDC$  с основанием  $YFDC$  боковое ребро равно  $\sqrt{1513}$ , сторона основания равна 24. Найдите объём пирамиды.

Ответ: 6720



Задача. 6: Пример генерации задания с помощью 3011.js

```

1 (function() {
2   retryWhileError(function() {
3     NAinfo.requireApiVersion(0, 2);
4
5     let angle = sl(2, 44);
6     let condition = [
7       ['большой', 45 + angle],
8       ['меньший', 45 - angle]
9     ].iz();
10
11    let paint1 = function(ctx) {
12      ctx.lineWidth = 2;
13
14      let angle = Math.PI/2.9;
15
16      ctx.strokeStyle = om.secondaryBrandColors.iz();
17      ctx.drawLine(10, 250, 390-8, 250);
18      let ver = ctx.drawLineAtAngle(10, 250, -angle, 200-25);
19      ctx.drawLineAtAngle(ver.x, ver.y, -angle+Math.PI/2, 350-20);
20      //штрихи
21      ctx.strokeInMiddleOfSegment(10, 250, (390-8)/2, 250, 10);
22      ctx.strokeInMiddleOfSegment(10, 250, 3*(390-8)/2, 250, 10);

```



```

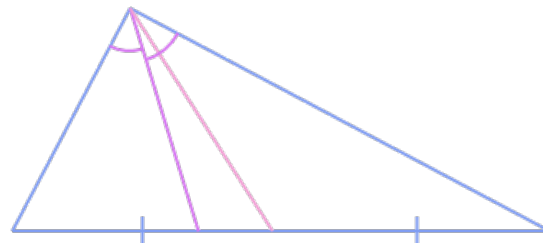
23     //биссектриса
24     ctx.strokeStyle = om.primaryBrandColors [0];
25     let bis = ctx.drawLineAtAngle(ver.x, ver.y,
(-angle+Math.PI/2)+Math.PI/4, 160+2);
26     //медиана
27     ctx.strokeStyle = om.primaryBrandColors [1];
28     ctx.drawLine(ver.x, ver.y,(390-8)/2, 250);
29
30     ctx.strokeStyle = om.primaryBrandColors [0];
31     ctx.arcBetweenSegments([10, 250,ver.x, ver.y, bis.x, bis.y], 30);
32     ctx.arcBetweenSegments([390-8, 250,ver.x, ver.y, bis.x, bis.y], 38);
33
34 };
35
36 NATask.setTask({
37     text: 'Угол между биссектрисой и медианой прямоугольного треугольника, ' +
38         'проведёнными из вершины прямого угла, равен $' + angle + '^{\circ}$.'
39     'Найдите ' + condition[0] + ' угол прямоугольного треугольника. Ответ
дайте в градусах.',
40     answers: condition[1],
41     analys: '',
42 });
43 NATask.modifiers.addCanvasIllustration({
44     width: 400,
45     height: 400,
46     paint: paint1,
47 });
48 }, 1000);
49 })();
50 //2069

```

Листинг 5: 2069.js

## Примеры генерируемых задач 2069.js

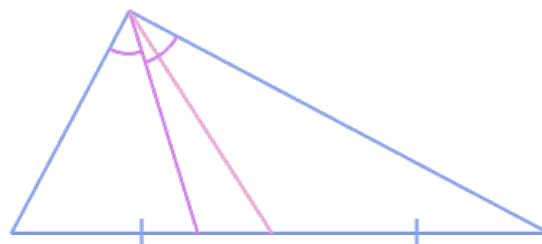
Угол между биссектрисой и медианой прямоугольного треугольника, проведёнными из вершины прямого угла, равен  $38^\circ$ . Найдите больший угол прямоугольного треугольника. Ответ дайте в градусах.



Ответ: 83

### Задача. 7: Пример генерации задания с помощью 2069.js

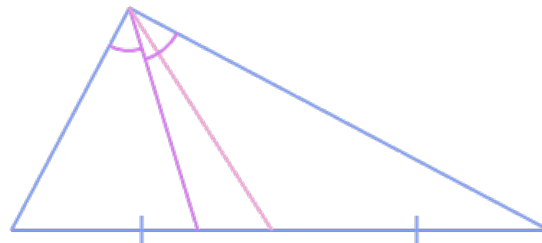
Угол между биссектрисой и медианой прямоугольного треугольника, проведёнными из вершины прямого угла, равен  $8^\circ$ . Найдите меньший угол прямоугольного треугольника. Ответ дайте в градусах.



Ответ: 37

### Задача. 8: Пример генерации задания с помощью 2069.js

Угол между биссектрисой и медианой прямоугольного треугольника, проведёнными из вершины прямого угла, равен  $31^\circ$ . Найдите меньший угол прямоугольного треугольника. Ответ дайте в градусах.



Ответ: 14

#### Задача. 9: Пример генерации задания с помощью 2069.js

```

1 (function() {
2   retryWhileError(function() {
3     NAinfo.requireApiVersion(0, 2);
4
5     let stroke = [4, 5];
6
7     let matrixConnections = {
8       0: [1, [3, stroke], 5],
9       2: [1, [3, stroke], 7],
10      4: [
11        [3, stroke],
12        [5, stroke],
13        [11, stroke]
14      ],
15      6: [1, 7, 9],
16      8: [5, [11, stroke], 13],
17      10: [7, 9, 15],
18      12: [
19        [11, stroke], 13, 15
20      ],
21      14: [9, 13, 15],
22    };

```

```

23
24 let par1 = new Parallelepiped({
25     depth: sl(10, 20),
26     height: sl(5, 20),
27     width: sl(10, 20),
28 });
29
30 let par2 = new Parallelepiped({
31     depth: par1.depth,
32     height: sl(5, 20),
33     width: slKrome(par1.width, 5, par1.width - 5),
34 });
35
36 let vertex3D =
par1.verticesOfFigure.concat(par2.verticesOfFigure.map((elem) =>
shiftCoordinate3D(elem, {
37     x: 0,
38     y: 0,
39     z: -0.5 * (par1.height + par2.height),
40 })))));
41
42 vertex3D = vertex3D.map((elem) => shiftCoordinate3D(elem, {
43     x: 0,
44     y: 0,
45     z: 0.5 * par2.height,
46 })));
47
48 let camera = {
49     x: 0,
50     y: 0,
51     z: 0,
52     scale: 1,
53
54     rotationX: -Math.PI / 2 + Math.PI / 14,
55     rotationY: 0,

```

```

56     rotationZ: Math.PI / sl(10, 14),
57 };
58
59     let point2D = vertex3D.map((coord3D) => project3DTo2D(coord3D,
camera));
60
61     autoScale(vertex3D, camera, point2D, {
62         startX: -180,
63         finishX: 160,
64         startY: -160,
65         finishY: 160,
66         maxScale: 100,
67     });
68
69     point2D = vertex3D.map((coord3D) => project3DTo2D(coord3D, camera));
70     genAssert((point2D[4].x - point2D[8].x).abs() > 20);
71     genAssert((point2D[4].y - point2D[13].y).abs() > 50);
72     genAssert((point2D[12].x - point2D[14].x).abs() > 40);
73
74     let rand = sl1();
75
76     let paint1 = function(ctx) {
77         let h = 400;
78         let w = 400;
79         ctx.translate(w / 2, h / 2);
80         ctx.lineWidth = 2;
81         ctx.strokeStyle = om.secondaryBrandColors;
82         ctx.drawFigureVer2(point2D, matrixConnections);
83
84         if (point2D[4].x > point2D[8].x) {
85             let point = [point2D[4], point2D[5], point2D[8],
point2D[13]].mt_coordinatesOfIntersectionOfTwoSegments();
86             ctx.drawLine(point2D[5].x, point2D[5].y, point.x, point.y);
87         } else {
88             ctx.drawLine(point2D[4].x, point2D[4].y, point2D[5].x,

```

```

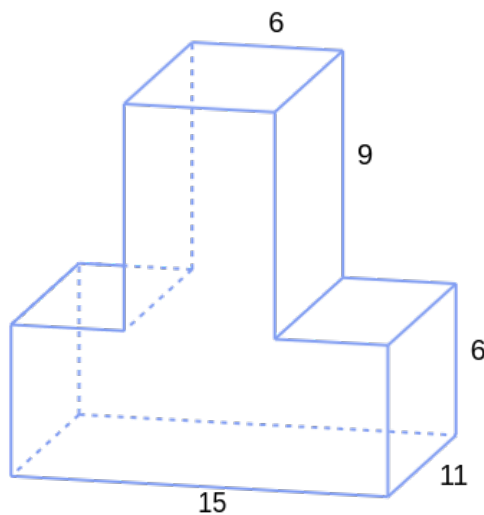
point2D[5].y);
89     let point = [point2D[4], point2D[11], point2D[8],
point2D[13]].mt_coordinatesOfIntersectionOfTwoSegments();
90     ctx.drawLine(point2D[4].x, point2D[4].y, point.x, point.y);
91 }
92
93     ctx.font = "20px liberation_sans";
94     ctx.signSegmentInMiddle(point2D[2].x, point2D[2].y, point2D[7].x,
point2D[7].y, par1.height, 10, 20);
95     ctx.signSegmentInMiddle(point2D[10].x, point2D[10].y,
point2D[15].x, point2D[15].y, par2.height, 10, 20);
96     ctx.signSegmentInMiddle(point2D[12].x, point2D[12].y,
point2D[15].x, point2D[15].y, par2.width, -10, 20);
97     ctx.signSegmentInMiddle(point2D[0].x, point2D[0].y, point2D[1].x,
point2D[1].y, par1.width, 18, 20);
98     ctx.signSegmentInMiddle(point2D[1].x, point2D[1].y, point2D[2].x,
point2D[2].y, par1.depth, 18, 20);
99 };
100     NATask.setTask({
101         text: 'Найдите ' + ['площадь поверхности', 'объём'][rand] +
102         ' многогранника, изображённого на рисунке все( двугранные углы -
прямые). ',
103         answers: [par1.surfaceArea + par2.surfaceArea - 2 * par2.baseArea,
par1.volume + par2.volume][rand],
104     });
105     NATask.modifiers.addCanvasIllustration({
106         width: 400,
107         height: 400,
108         paint: paint1,
109     });
110 },
111 1000);
112
113 })();
114 //27193 25671 25673 25675 25677 25679

```

### Примеры генерируемых задач 27193.js

Найдите площадь поверхности многогранника, изображённого на рисунке (все двугранные углы – прямые).

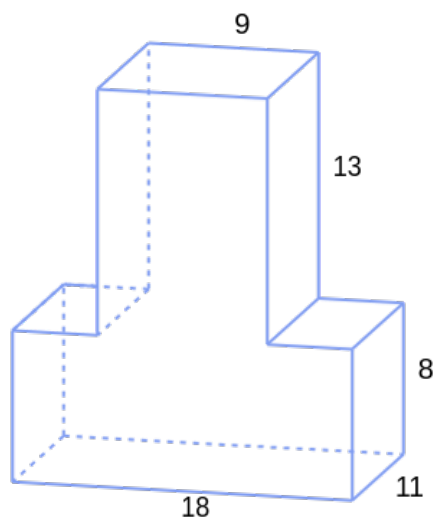
Ответ: 948



Задача. 10: Пример генерации задания с помощью 27193.js

Найдите объём многогранника, изображённого на рисунке (все двугранные углы – прямые).

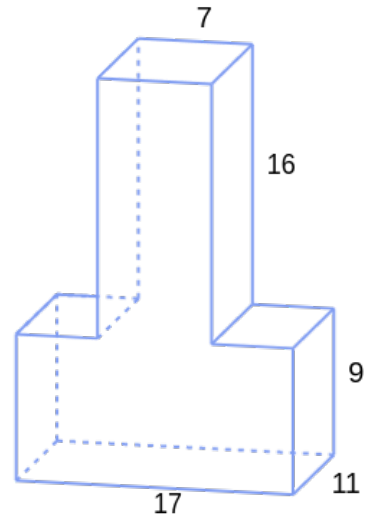
Ответ: 2871



Задача. 11: Пример генерации задания с помощью 27193.js

Найдите площадь поверхности многогранника, изображённого на рисунке (все двугранные углы – прямые).

Ответ: 1454



Задача. 12: Пример генерации задания с помощью 27193.js

```
1 (function() {  
2   retryWhileError(function() {  
3     NAinfo.requireApiVersion(0, 2);  
4  
5     let angle = sl(2, 89);  
6  
7     let vertices = window.latbukv.iz(6);  
8  
9     let rand = sl1();  
10  
11    let paint1 = function(ctx) {  
12      ctx.lineWidth = 2;  
13  
14      let angle = -Math.PI / 3.2;  
15      ctx.strokeStyle = om.secondaryBrandColors.iz();  
16  
17      let vertex = ctx.drawLineAtAngle(10, 370, angle, 400);  
18      ctx.drawLine(10, 370, 390, 370);  
19      ctx.drawLine(390, 370, vertex.x, vertex.y);  
20  
21      //Биссектрисы  
22      let bisector1 = ctx.drawLineAtAngle(10, 370, angle / 2, 345);
```



```

23     let bisector2 = ctx.drawLineAtAngle(390, 370, Math.atan2(-370 +
vertex.y, -390 + vertex.x) / 2 - Math.PI / 2, 317);
24
25     //УГЛЫ
26     ctx.strokeStyle = om.primaryBrandColors[rand];
27     ctx.arcBetweenSegmentsCount([vertex.x, vertex.y, 10,
370].concat([bisector1.x, bisector1.y]), 30, 2);
28     ctx.arcBetweenSegmentsCount([bisector1.x, bisector1.y].concat([10,
370, 390, 370])), 40, 2);
29
30     ctx.strokeStyle = om.primaryBrandColors[rand];
31     ctx.arcBetweenSegments([10, 370, 390, 370].concat([bisector2.x,
bisector2.y]), 30);
32     ctx.arcBetweenSegments(([bisector2.x, bisector2.y].concat([390, 370,
vertex.x, vertex.y])), 40);
33
34     ctx.strokeStyle = om.primaryBrandColors[1 - rand];
35     ctx.arcBetweenSegmentsCount([bisector1.x, bisector1.y].concat([10,
370]).concat([bisector2.x, bisector2.y]).concat([390, 370])), 25, 3);
36
37     ctx.font = "23px liberation_sans";
38     ctx.fillText(vertices[0], vertex.x, vertex.y - 10);
39     ctx.fillText(vertices[1], 10 - 5, 370 + 20);
40     ctx.fillText(vertices[2], 390 - 20, 370 + 20);
41
42     ctx.fillText(vertices[3], bisector1.x, bisector1.y);
43     ctx.fillText(vertices[4], bisector2.x - 20, bisector2.y);
44
45     ctx.fillText(vertices[5], 210, 250);
46 };
47
48 NAtask.setTask({
49     text: 'В треугольнике $' + vertices.slice(0, 3).join('') + '$ угол $'
+ vertices[0] + '$ равен $' + angle +
50     '~{\circ}$, углы $' + vertices[1] + '$ и $' + vertices[2] + '$ -

```

```

острые, ' +
51     'биссектрисы $' + [vertices[1], vertices[3]].shuffleJoin() + '$ и
    '$' + [vertices[2], vertices[4]].shuffleJoin() +
52     '$ пересекаются в точке $' + vertices[5] + '$. Найдите угол $' +
    vertices[2] + vertices[5] + vertices[1] +
53     '$. Ответ дайте в градусах.',
54     answers: 90 + 0.5 * angle,
55     analys: '',
56 });
57 NATask.modifiers.addCanvasIllustration({
58     width: 400,
59     height: 400,
60     paint: paint1,
61 });
62 }, 1000);
63 }());
64 //27764 628357 628475 47369 47371 47373 47375 47377 47379 47381 47383
    47385 47387 47389 47391 47393 47395 47397 47399 47401 47403 47405 47407
    47409 47411 47413 47415 47417 47419 47421 47423 47425 47427 47429 47431
    47433 47435 47437 47439 47441 47443 47445 47447 47449 47451 47453

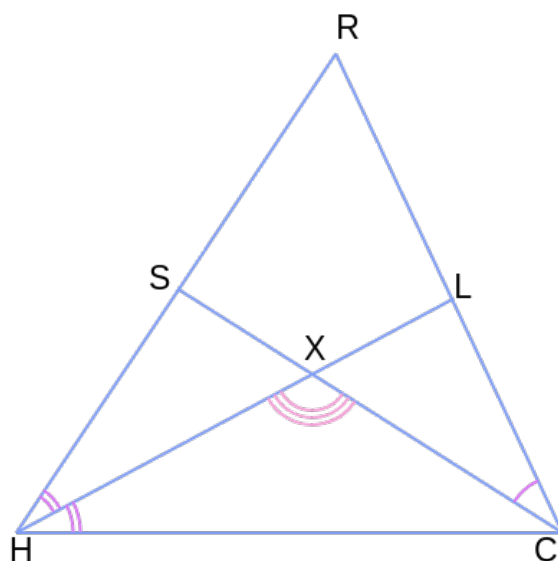
```

Листинг 7: 27764.js

## Примеры генерируемых задач 27764.js

В треугольнике  $RHC$  угол  $R$  равен  $67^\circ$ , углы  $H$  и  $C$  – острые, биссектрисы  $LH$  и  $SC$  пересекаются в точке  $X$ . Найдите угол  $CXH$ . Ответ дайте в градусах.

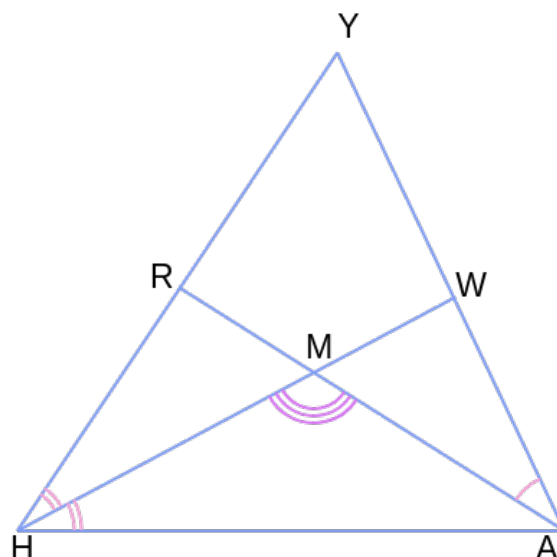
Ответ: 123,5



Задача. 13: Пример генерации задания с помощью 27764.js

В треугольнике  $YHA$  угол  $Y$  равен  $45^\circ$ , углы  $H$  и  $A$  – острые, биссектрисы  $HW$  и  $AR$  пересекаются в точке  $M$ . Найдите угол  $AMH$ . Ответ дайте в градусах.

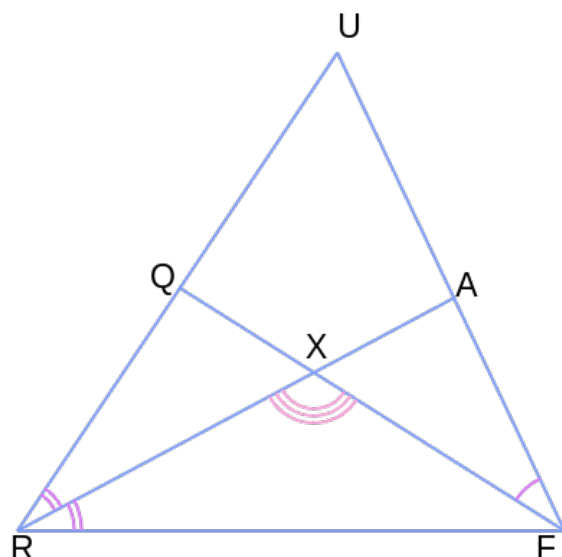
Ответ: 112,5



Задача. 14: Пример генерации задания с помощью 27764.js

В треугольнике  $URF$  угол  $U$  равен  $7^\circ$ , углы  $R$  и  $F$  – острые, биссектрисы  $AR$  и  $FQ$  пересекаются в точке  $X$ . Найдите угол  $FXR$ . Ответ дайте в градусах.

Ответ: 93,5



Задача. 15: Пример генерации задания с помощью 27764.js

```

1 (function () {
2   retryWhileError(function () {
3     NAinfo.requireApiVersion(0, 2);
4
5     let a = sl(2, 89);
6     let b = slKrome(a, 1, a - 1);
7     genAssertZ1000(b / a, 'Нецелый ответ');
8
9     let vertices = ['A', 'B', 'C', 'H'];
10
11    let paint1 = function (ctx) {
12
13      ctx.lineWidth = 2;
14      ctx.strokeStyle = om.secondaryBrandColors;
15
16      //стороны
17      ctx.drawLine(10, 370, 390, 370);
18      ctx.drawLine(10, 370, 180, 50);
19      ctx.drawLine(180, 50, 390, 370);
20      //высота
21      ctx.drawLine(280, 200, 10, 370);
22

```

```

23     ctx.strokeInMiddleOfSegment(180, 50, 10, 370, 10);
24     ctx.strokeInMiddleOfSegment(180, 50, 390, 370, 10);
25
26     ctx.arcBetweenSegments([180, 50, 280, 200, 280, 200, 10, 370], 25);
27
28     ctx.font = "23px liberation_sans";
29     ctx.fillText(vertices[0], 10 - 5, 370 + 25);
30     ctx.fillText(vertices[1], 180, 50 - 10);
31     ctx.fillText(vertices[2], 390 - 10, 370 + 25);
32     ctx.fillText(vertices[3], 280 + 10, 200);
33 };
34
35 NAtask.setTask({
36     text: 'В треугольнике  $ABC$  ' + [' $AC=BC$ ', ' $AB=$ ' + a + ' $,$ ' + ' $AN -$   

высота', ' $BH=$ ' + b + ' $,$ '].shuffleJoin(', ') + '. Найдите косинус  

 $BAC$ .' ,
37     answers: b / a,
38     author: ['Суматохина Александра']
39 });
40 NAtask.modifiers.variativeABC(vertices);
41 NAtask.modifiers.addCanvasIllustration({
42     width: 400,
43     height: 400,
44     paint: paint1,
45 });
46 }, 1000);
47 })();

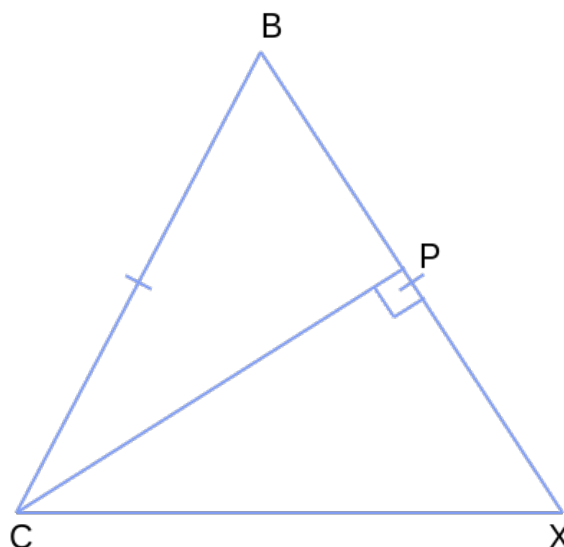
```

Листинг 8: 109.js

## Примеры генерируемых задач 109.js

В треугольнике  $CBX$   $BP = 5$ ,  $CB = 20$ ,  $CX = BX$ ,  $CP \perp BX$ . Найдите косинус  $\angle BCX$ .

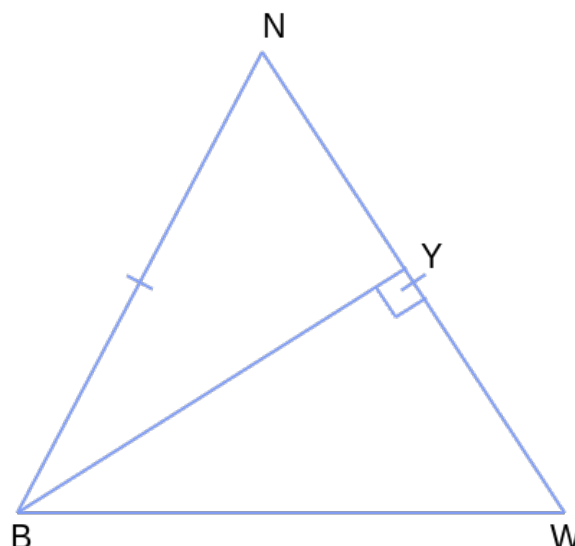
Ответ: 33,75



Задача. 16: Пример генерации задания с помощью 109.js

В треугольнике  $BNW$   $BW = NW$ ,  $BY \perp NW$ ,  $NY = 6$ ,  $BN = 8$ . Найдите косинус  $\angle NBW$ .

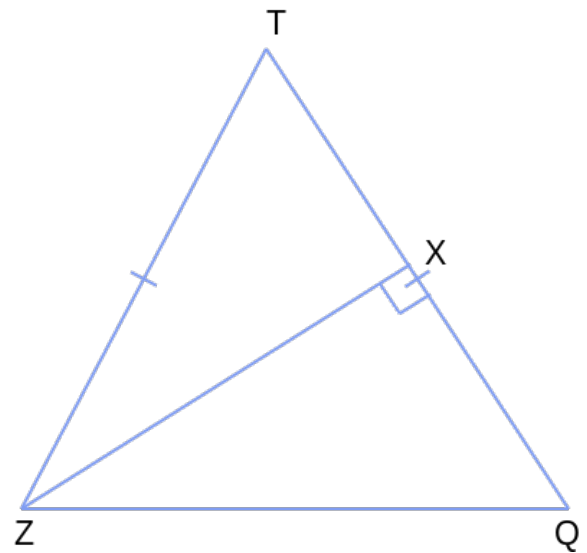
Ответ: 35,25



Задача. 17: Пример генерации задания с помощью 109.js

Объём правильной четырёхугольной пирамиды  $UOVSZ$  равен 25650. Точка  $B$  – середина ребра  $UO$ . Найдите объём треугольной пирамиды  $BOVZ$ .

Ответ: 12



Задача. 18: Пример генерации задания с помощью 109.js