

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

Математический факультет

Кафедра теории функций и геометрии

Программная реализация (на языке JavaScript) алгоритмов
генерации ФОС по геометрии в 2024 году

Курсовая работа

Направление 010501 Фундаментальные математика и механика

Зав.кафедрой _____ д.физ.-мат.н., проф. Е.М. Семёнов

Обучающийся _____ А.С. Суматохина

Руководитель _____ д.физ.-мат.н., проф. Е.М. Семёнов

Воронеж 2024

Содержание

Введение	3
1 Планиметрия	4
1.1 Вспомогательные функции	4
1.1.1 Функции для работы с массивами	4
1.1.2 Функции для работы с числами	5
1.1.3 Функции для работы с canvas	5
1.2 Этапы разработки шаблоны с вспомогательным чертежом по теме «Планиметрия»	8
2 Стереометрия	12
2.1 Разработка библиотек с помощью Gpt-Chat	12
2.2 Применение ООП для разработки шаблонов	15
2.3 Вспомогательные функции	16
2.3.1 Функции для работы с координатами	16
2.3.2 Функции для работы с canvas	16
2.4 Этапы разработки шаблоны с вспомогательным чертежом по теме «Стереометрия»	17
Заключение	23
Приложение	25

Введение

Единый государственный экзамен (ЕГЭ) — централизованно проводимый в Российской Федерации экзамен в средних учебных заведениях — школах, лицеях и гимназиях, форма проведения ГИА (Государственный Итоговая Аттестация) по образовательным программам среднего общего образования. Служит одновременно выпускным экзаменом из школы и вступительным экзаменом в вузы.

Но за время обучения в 10 и 11 классе при подготовке к ЕГЭ школьники сталкиваются с дефицитом заданий по определённым категориям. Так в конце 2021 года в список заданий ЕГЭ были добавлены новые задания под номером 11 по теме «Графики функции», а в конце 2023 - задание №2 по теме «вектора», количество которых, для прорешивания было очень мало. А по теме «Производная и первообразная» 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.

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 с 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»). [3] Планируется с течением времени включить в проект тесты по другим предметам школьной программы.

Первая глава этой работы посвящена обзору вспомогательных функций, которые ускоряют написание шаблонов по теме «Планиметрия». А также приведён алгоритм написания шаблона с чертежом.

Вторая глава представляет решение проблемы отрисовки фигур в трёхмерном пространстве на языке программирования JavaScript; рассказывает о применении объектно-ориентированного программирования для упрощения написания шаблонов с чертежом; приводит обзор вспомогательных функций и алгоритм написания шаблона по теме «Стереометрия».

1. Планиметрия

В этой главе мы приводим вспомогательные функции и алгоритм написания шаблона по планиметрии

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 10 функция используется для перестановки букв в названии угла.

```
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. В листинге 4 в строке 85 функция используется для нахождения точки пересечения рёбер составного многогранника.

```
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: 3.6666666666666667, status: false }
5   //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: -1.8333333333333333, y: 4.541666666666667, status: true }
10  //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 5 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

1.1.3. Функции для работы с canvas

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

Заполняет область цветом fillStyle по вершинам из массива vertex.

```

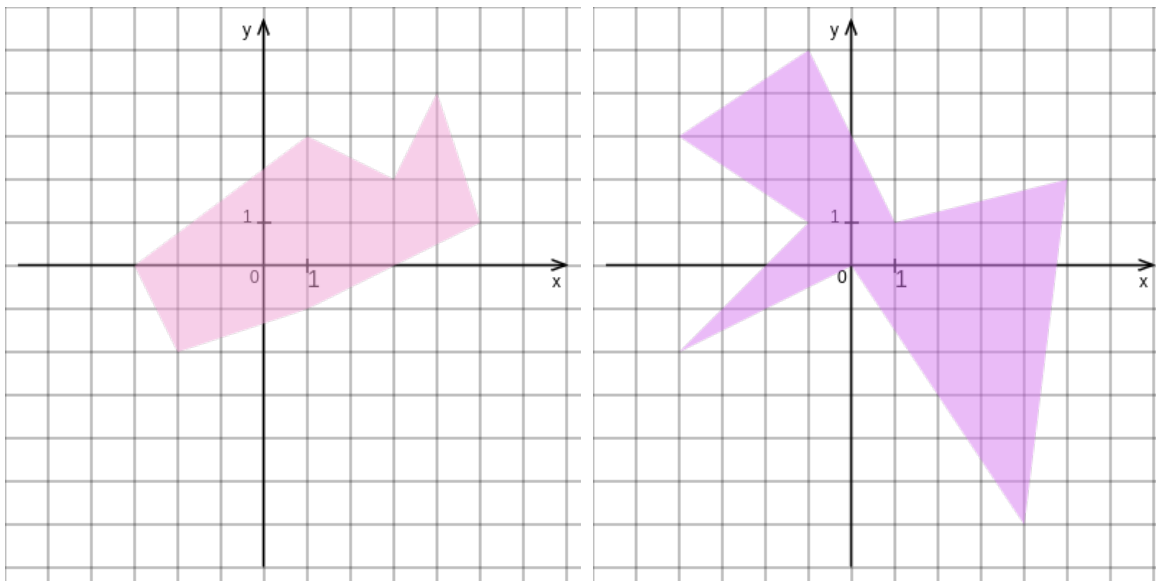
1 let paint1 = function(ctx) {
2   let h = 400;
3   let w = 400;
4   ctx.drawCoordinatePlane(w, h, {
5     hor: 1,

```

```

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
16    ctx.drawSection([[-2, 0], [-1, 1], [-4, 3], [-1, 5], [1, 1], [5, 2], [4,
17    -6], [0, 0], [-4, -2],]);
18    };

```



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

Рисует отрезок длины `length` под углом `angle` (в радианах). Пример использования в листинге 6 в строках 19 и 25 (используется 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 6 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 4 в строках 94 - 98 функция используется для отображения длин рёбер многогранника.

`CanvasRenderingContext2D.prototype.arcBetweenSegments`

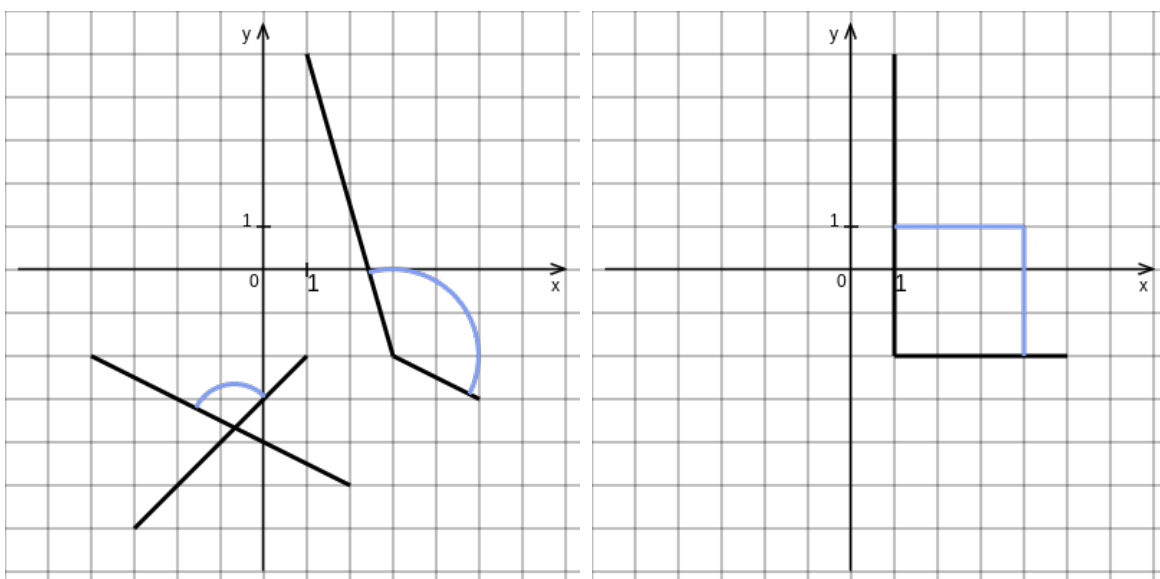
= function(coordinates, radius)

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.

1.2. Этапы разработки шаблоны с вспомогательным чертежом по теме «Планиметрия»

Для примера возьмём задание №19416 [2].

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, AHвысота-, BH=5. Найдите косинус
ABC ',
10      answers: 0,
11      author: ['Суматохина Александра']
12    });
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     ctx.drawLine(10, 370, 390, 370);
11     ctx.drawLine(10, 370, 180, 50);
12     ctx.drawLine(180, 50, 390, 370);
13
14     ctx.drawLine(280, 200, 10, 370);
15 };
16
17 NATask.setTask({
18     text: 'В треугольнике ABC AC=BC, AB=15, AHвысота-, BHНайдите=5.
косинус ABC.',
19     answers: 0,
20     author: ['Суматохина Александра']
21 });
22 NATask.modifiers.addCanvasIllustration({
23     width: 400,
24     height: 400,
25     paint: paint1,
26 });
27 }, 1000);
28 })();

```

2. Добавим на рисунок штрихи, указывающие на равенство сторон и обозначение прямого угла при помощи функций `strokeInMiddleOfSegment` и `arcBetweenSegments` соответственно. И подпишем вершины и точку перпендикуляра. 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            ctx.drawLine(10, 370, 390, 370);
13            ctx.drawLine(10, 370, 180, 50);
14            ctx.drawLine(180, 50, 390, 370);
15
16            ctx.drawLine(280, 200, 10, 370);
17
18            ctx.strokeInMiddleOfSegment(180, 50, 10, 370, 10);
19            ctx.strokeInMiddleOfSegment(180, 50, 390, 370, 10);
20
21            ctx.arcBetweenSegments([180, 50, 280, 200, 280, 200, 10, 370],
22            25);

```

```

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

```

3. Теперь добавим ответ в задание. Проверим при помощи `genAssertZ1000`, что ответ целый (если 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            ctx.drawLine(10, 370, 390, 370);
17            ctx.drawLine(10, 370, 180, 50);
18            ctx.drawLine(180, 50, 390, 370);
19
20            ctx.drawLine(280, 200, 10, 370);
21
22            ctx.strokeInMiddleOfSegment(180, 50, 10, 370, 10);
23            ctx.strokeInMiddleOfSegment(180, 50, 390, 370, 10);

```

```

24
25     ctx.arcBetweenSegments([180, 50, 280, 200, 280, 200, 10, 370],
26     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 $ABC$ '+'[$AC=BC$', '$AB='+a+'$',
37         '$AHвысота-$', '$BH='+b+'$'].shuffleJoin(', ')+'. Найдите косинус
38         $BAC$.',
39         answers: b / a,
40         author: ['Суматохина Александра']
41     });
42     NAtask.modifiers.variativeABC(vertices);
43     NAtask.modifiers.addCanvasIllustration({
44         width: 400,
45         height: 400,
46         paint: paint1,
47     });
48 }, 1000);
49 })();

```

2. Стереометрия

2.1. 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 подключение библиотеки значительно увеличивает вес проекта (на 2 GB). 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

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 самых 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.

2.2. Применение ООП 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 оснований, а так же тернарную матрицу связности и канонические координаты вершин.

Матрица может содержать значения: 1, 0, либо специальное значение, указывающий на отображении ребра пунктиром.

Пример канонической матрицы связей:

```

[
  [1],
  [0, 1],
  [1, 0, 1],
  [0, 0, 0, 1],
  [1, 0, 0, 0, 1],
  [0, 1, 0, 0, 0, 1],
  [0, 0, 1, 0, 1, 0, 1],
];

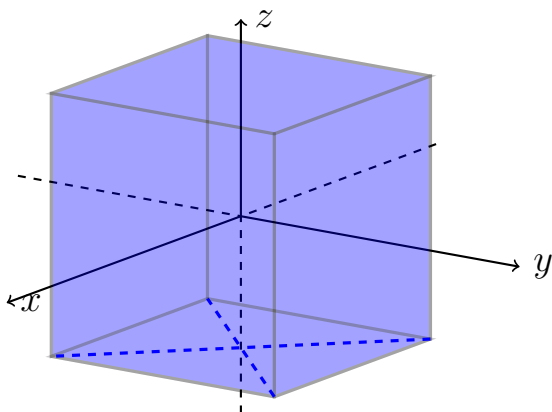
```

Листинг 2: Каноническая матрица связей для параллелепипеда

Мы можем опускать конец матрицы, если он состоит только из нулей. И нам не

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

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



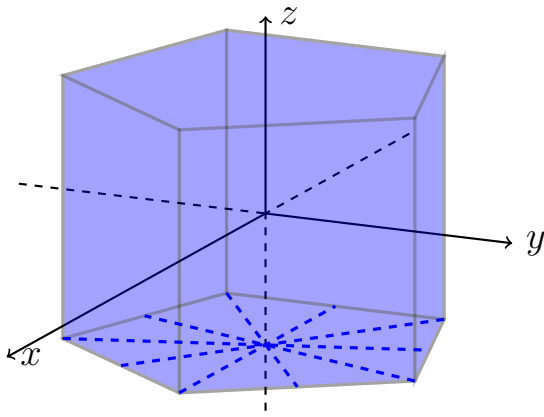


Рис. 2: Каноническое положение для правильной пятиугольной призмы

2.3. Вспомогательные функции

2.3.1. Функции для работы с координатами

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

Возвращает `true`, если двухмерная 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 двухмерные координаты `vertex2D` вида `{x,y}` находится в некоторой прямоугольной области. `step` по умолчанию 0.1.

function `distanceFromPointToSegment`(point, segmentStart, segmentEnd)

Возвращает длину перпендикуляра между двухмерной точкой `point` вида `{x,y}` до отрезка с концами в `segmentStart` и `segmentEnd`.

2.3.2. 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:

```
1  let matrixConnections = [
2      [1],
3      [strok, strok],
4      [0, 0, strok],
5      [1, 0, 0, 1],
6      [0, 1, 0, 1, 1]
7  ];
8
```



```
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:

```
1 let matrixConnections = {
2   0: [1, [3, stroke], 5],
3   2: [1, [3, stroke], 7],
4   4: [[3, stroke], 5, 7],
5   9: [1, 8, 10],
6  11: [8, 10, 12],
7  13: [5, 8, 12],
8  15: [7, 10, 12],
9 };
10
```

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

Для примера возьмём задание №27074 [5].

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
16      объем треугольной пирамиды ABCA_1.',
17      answers: 0,
18      author: ['Суматохина Александра']
19    });
20    NATask.modifiers.addCanvasIllustration({
21      width: 400,
22      height: 400,
23      paint: paint1,
24    }, 100000);
25  })();

```

2. Определим переменную **camera**, которая будет отвечать за положение наблюдателя. И спроецируем канонические координаты параллелепипеда на двухмерную плоскость при помощи функции **project3DTo2D**. И отмасштабируем полученные координаты так, чтобы они занимали максимально заполняли спрайт, 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 ABCA_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. Добавим модификаторы `NAtask.modifiers.assertSaneDecima` (исключает нецелый ответ) и `NAtask.modifiers.variativeABC(letter)` (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
64 $'+par.volume+'$. Найдите объем треугольной пирамиды $ABCA_1$.',
65         answers: par.volume/6,
66         author: ['Суматохина Александра']
67     });
68
69     NATask.modifiers.assertSaneDecimals();
70     NATask.modifiers.variativeABC(letter);
71
72     NATask.modifiers.addCanvasIllustration({
73         width: 400,
74         height: 400,
75         paint: paint1,
76     });
77     }, 100000);
78 })();

```

Заключение

За этот год был полностью покрыт открытый банк заданий ФИПИ по темам:

- Планиметрия — 26 шаблонов принято.
- Вектора — 18 шаблонов (10 принято, 8 на рецензировании).
- Стереометрия — 56 шаблонов (7 принято, 49 на рецензировании).
- Теория вероятности — 10 шаблонов на рецензировании.
- Теория вероятности (повышенной сложности) — 11 шаблонов (1 принят 10 на рецензировании).

В ядро проекта добавлены:

- Функции, упрощающие написание шаблонов по темам «Планиметрия» и «Стереометрия».
- Класс многогранников.
- Линейный проектор из $\mathbb{R}^3 \rightarrow \mathbb{R}^2$.

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 в проект класс плоских геометрических фигур и использовать в заданиях по теме «Планиметрия» динамические изображения.

Список литературы

- [1] Момот Е. А., Арахов Н. Д. Разработка и внедрение ПО для сбора статистики результатов подготовки к ЕГЭ по математике профильного уровня //Актуальные проблемы прикладной математики, информатики и механики. – 2021. – С. 1-2.
- [2] Открытый банк задач ЕГЭ по Математике.Профильный уровень. – URL: <https://prof.mathege.ru/>
- [3] Федеральный институт педагогических измерений. – URL: <https://fipi.ru/ege/otkrytyy-bank-zadaniy-ege>
- [4] Единый государственный экзамен. – URL: https://ru.wikipedia.org/wiki/Единый_государственный_экзамен
- [5] Решу ЕГЭ - Сдам ГИА. – URL: <https://ege.sdamgia.ru/problem?id=27074>

Приложение

```
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
8     let vertices = window.latbukv.iz(4);
9
10    let angle = sl1() ? vertices.slice(0, 3).permuteCyclic(1) :
vertices.slice(0, 3).permuteCyclic(2);
11
12    genAssertZ1000(b / a, 'Кривой ответ');
13
14    let paint1 = function(ctx) {
15      ctx.lineWidth = 2;
16      ctx.strokeStyle = om.primaryBrandColors [0];
17
18      ctx.drawLine(10, 370, 390, 370);
19      ctx.drawLine(10, 370, 180, 200);
20      ctx.drawLine(180, 200, 390, 370);
21
22      //высота
23      ctx.lineWidth = 1.2;
24      ctx.drawLine(180 - 40, 200 - 33, 390, 370);
25      ctx.drawLine(180 - 40, 200 - 33, 10, 370);
26
27      //прямой угол
28      ctx.strokeStyle = om.secondaryBrandColors.iz();
29      ctx.drawLine(180 - 52, 200 - 13, 180 - 35, 200 - 1);
30      ctx.drawLine(180 - 35, 200 - 1, 180 - 23, 200 - 17);
31
32      //штрихи
33      ctx.drawLine(275 + 10, 210 + 90, 300, 200 + 90);
34      ctx.drawLine(80, 200 + 90, 105 - 10, 210 + 90);
35
36      ctx.font = "23px liberation_sans";
37      ctx.fillText(vertices[0], 10 - 5, 370 + 25);
38      ctx.fillText(vertices[1], 180, 200 - 10);
39      ctx.fillText(vertices[2], 390 - 10, 370 + 25);
40      ctx.fillText(vertices[3], 180 - 40, 200 - 40);
41
42    };
43
44    NATask.setTask({
45      text: 'В треугольнике $' + vertices.slice(0, 3).shuffle().join('') +
'$ $' +
46      vertices.slice(0, 2).shuffle().join('') +
```

```

47     '=' + vertices.slice(1, 3).shuffle().join('') + '$, $' +
vertices[0] + vertices[2] + '=' + a + '$, высота $' +
48     [vertices[0], vertices[3]].shuffle().join('') +
49     '$ равна $' + b + '$. Найдите $\\sin \\angle ' + angle.join('') +
'$.',
50     answers: b / a,
51     analys: '',
52   });
53   NATask.modifiers.addCanvasIllustration({
54     width: 400,
55     height: 400,
56     paint: paint1,
57   });
58   }, 1000);
59   })();
60   // 105

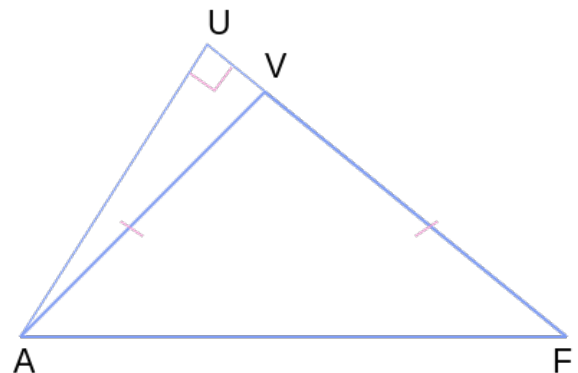
```

Листинг 3: 105.js

Примеры генерируемых задач 105.js

В треугольнике FAF $VA = VF$,
 $AF = 75$, высота AU равна 39. Най-
дите $\sin \angle VFA$.

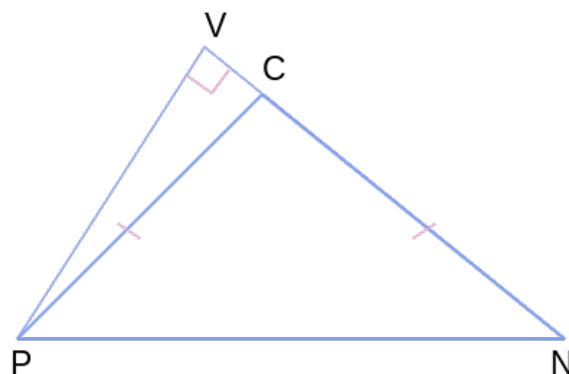
Ответ: 0,52



Приложение. 1

В треугольнике NCP $CP = CN$, $PN = 40$, высота VP равна 6. Найдите $\sin \angle CNP$.

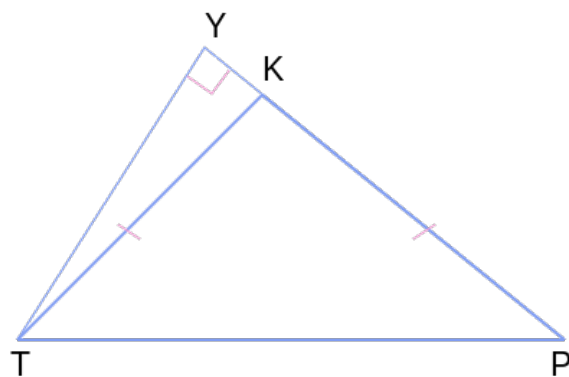
Ответ: 0,15



Приложение. 2

В треугольнике PKT $KT = PK$, $TP = 4$, высота TY равна 2. Найдите $\sin \angle KPT$.

Ответ: 0,5



Приложение. 3

```

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 =
37  par1.verticesOfFigure.concat(par2.verticesOfFigure.map((elem) =>
38  shiftCoordinate3D(elem, {
39      x: 0,
40      y: 0,
41      z: -0.5 * (par1.height + par2.height),
42  })));
43
44  vertex3D = vertex3D.map((elem) => shiftCoordinate3D(elem, {
45      x: 0,
46      y: 0,
47      z: 0.5 * par2.height,
48  })));
49
50  let camera = {
51      x: 0,
52      y: 0,
53      z: 0,
54      scale: 1,
55
56      rotationX: -Math.PI / 2 + Math.PI / 14,
57      rotationY: 0,
58      rotationZ: Math.PI / sl(10, 14),
59  };
60
61  let point2D = vertex3D.map((coord3D) => project3DTo2D(coord3D,
62  camera));
63
64  autoScale(vertex3D, camera, point2D, {
65      startX: -180,
66      finishX: 160,
67      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

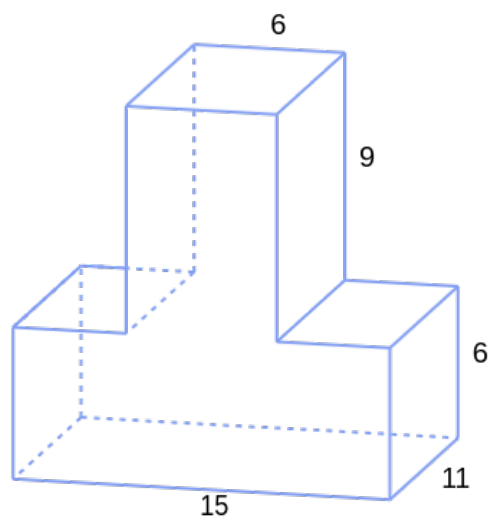
```

Листинг 4: 29193.js

Примеры генерируемых задач 27193.js

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

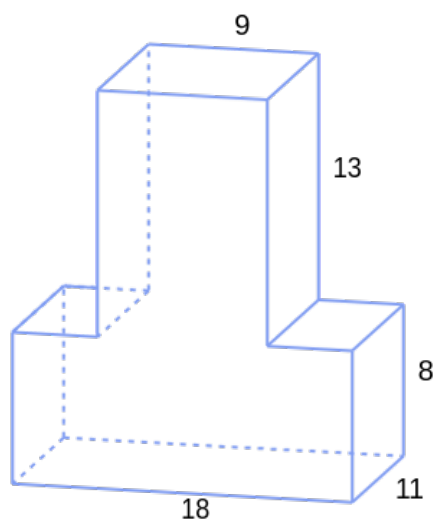
Ответ: 948



Приложение. 4

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

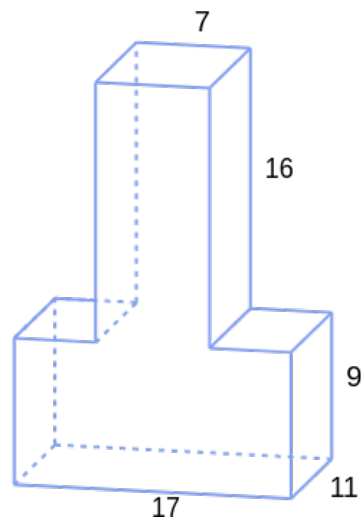
Ответ: 2871



Приложение. 5

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

Ответ: 1454



Приложение. 6

```

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,
elem.y + ((i != point2DPyr.length - 2) ? 15 : -
67       10)));
68   };
69
70   NATask.setTask({
71     text: 'В правильной четырёхугольной пирамиде $SABCD$ с ' + 'основанием
$ABCD$ ' +
72       [question[0][0].ie + ' рав' + ['ен', 'на',
'но']][question[0][0].rod] + ' $' + question[0][1].pow(2).texsqrt(1) +
'$ ',
73     question[1][0].ie + ' рав' + ['ен', 'на', 'но']][question[1][0].rod]
+ ' $' + question[1][1].pow(2).texsqrt(1) + '$ '
74     ].shuffleJoin(', ') +
75     '. Найдите ' + question[2][0].ve + ' пирамиды.',
76     answers: question[2][1],
77     author: ['Суматохина Александра'],
78   });

```



```

79   NAtask.modifiers.variativeABC(letters);
80   NAtask.modifiers.multiplyAnswerBySqrt(13);
81   NAtask.modifiers.allDecimalsToStandard(true);
82   NAtask.modifiers.assertSaneDecimals();
83   NAtask.modifiers.addCanvasIllustration({
84     width: 400,
85     height: 400,
86     paint: paint1,
87   });
88   },10);
89 })();
90 //https://ege314.ru/8-stereometriya-ege/reshenie-3011/
91 //3011

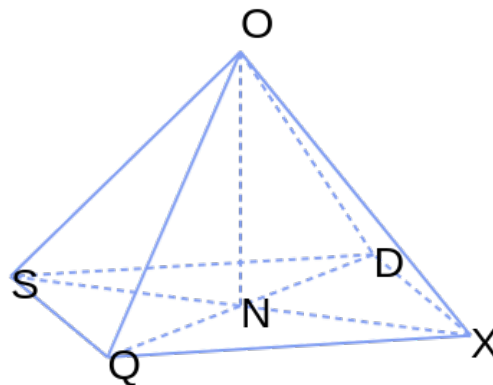
```

Листинг 5: 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.

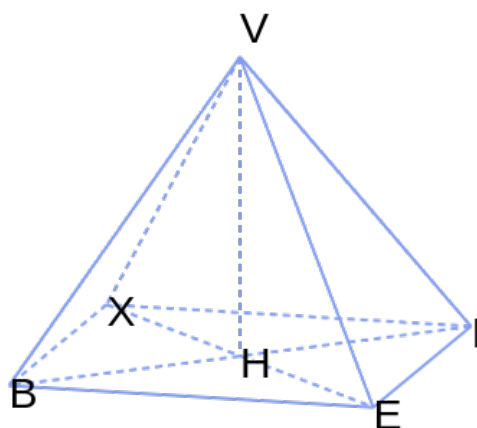
Ответ: 13689



Приложение. 7

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

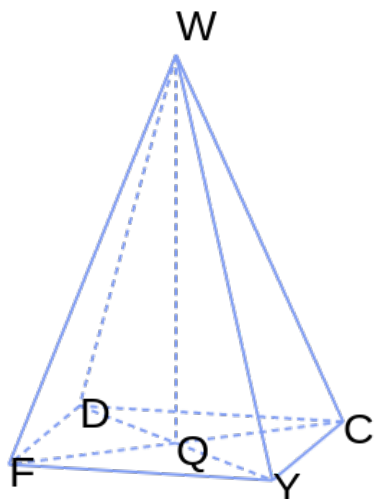
Ответ: 5346



Приложение. 8

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

Ответ: 6720



Приложение. 9

```

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

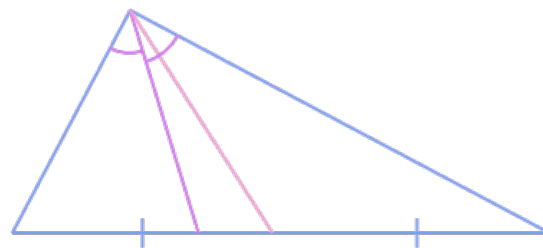
```

Листинг 6: 2069.js

Примеры генерируемых задач 2069.js

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

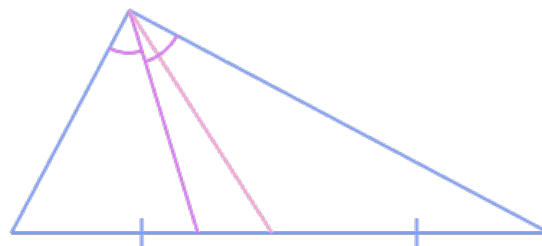
Ответ: 83



Приложение. 10

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

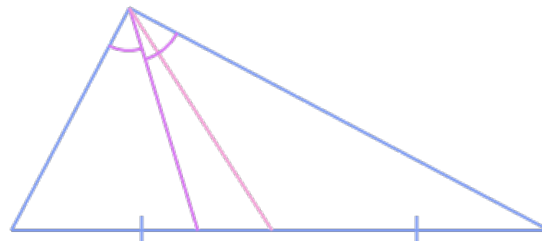
Ответ: 37



Приложение. 11

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

Ответ: 14



Приложение. 12

```

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 +
24        vertex.y, -390 + vertex.x) / 2 - Math.PI / 2, 317);
25
26      //Углы
27      ctx.strokeStyle = om.primaryBrandColors[rand];
28      ctx.arcBetweenSegmentsCount([vertex.x, vertex.y, 10,
29        370].concat([bisector1.x, bisector1.y]), 30, 2);
30      ctx.arcBetweenSegmentsCount([bisector1.x, bisector1.y].concat([10,
31        370, 390, 370])), 40, 2);
32
33      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     '~{\c} $, 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

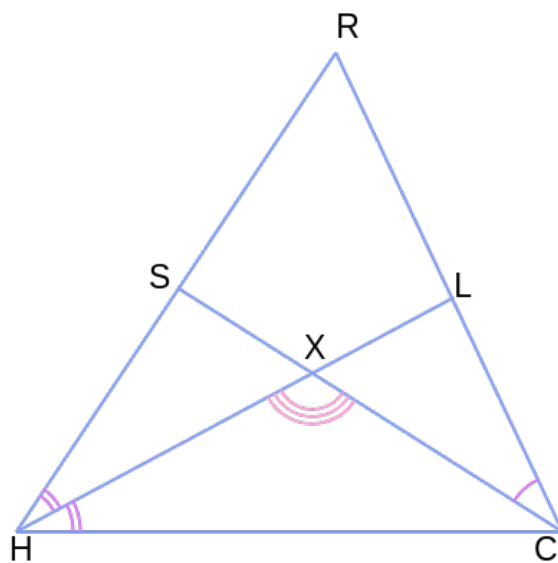
```

Листинг 7: 27764.js

Примеры генерируемых задач 27764.js

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

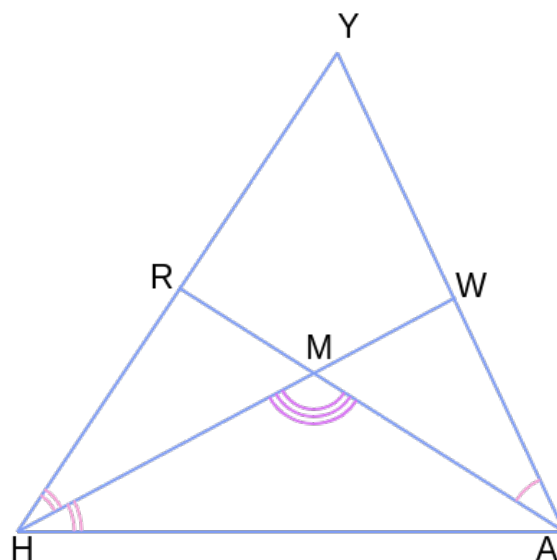
Ответ: 123,5



Приложение. 13

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

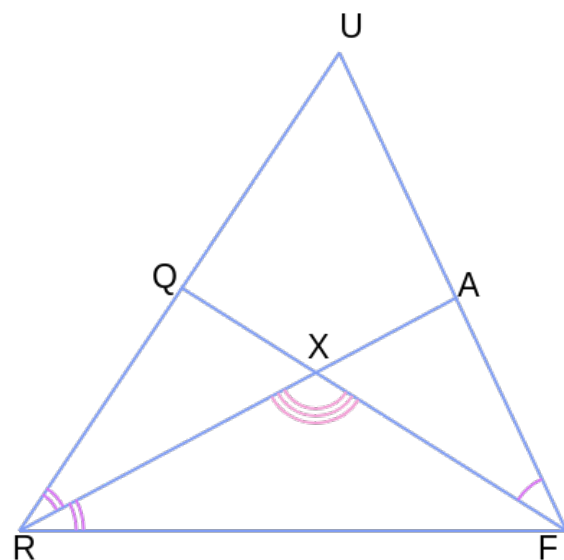
Ответ: 112,5



Приложение. 14

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

Ответ: 93,5



Приложение. 15

Шаблоны по теме «Стереометрия»

```
1 (function() {
2   retryWhileError(function() {
3
4     let pyr1 = new RegularPyramid({
5       height: sl(30, 70),
6       baseSide: sl(20, 50),
7       numberSide: 4
8     });
9
10    let pyr2 = new Pyramid({
11      height: 0.5 * pyr1.height,
12      baseArea: 0.5 * pyr1.baseArea,
13    });
14
15    pyr1.verticesOfFigure =
coordinatesMiddleOfSegment3D(pyr1.verticesOfFigure[0],
pyr1.verticesOfFigure[4]);
16
17    let camera = {
18      x: 0,
19      y: 0,
20      z: 0,
21      scale: 5,
22
23      rotationX: -Math.PI / 2 + Math.PI / 14,
24      rotationY: 0,
25      rotationZ: [1, 2].iz() * Math.PI / 3,
26    };
27
28    let point2DPyr = pyr1.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));
29
30    autoScale(pyr1.verticesOfFigure, camera, point2DPyr, {
31      startX: -180,
32      finishX: 160,
33      startY: -160,
34      finishY: 160,
35      maxScale: 50,
36    });
37
38    point2DPyr = pyr1.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));
39
40    let letters = ['A', 'B', 'C', 'D', 'S', 'E'];
41    let strok = [5, 4];
42
43    let paint1 = function(ctx) {
44      let h = 400;
45      let w = 400;
46      ctx.translate(h / 2, w / 2);
```

```

47     ctx.lineWidth = 2;
48     ctx.strokeStyle = om.secondaryBrandColors;
49     ctx.drawFigure(point2DPyr, [
50         [1],
51         [0, strok],
52         [1, strok, strok],
53         [1, 1, strok, 1, ],
54         [0, 1, 0, 1, 0]
55     ]);
56
57     ctx.font = "30px liberation_sans";
58     point2DPyr.forEach((elem, i) => ctx.fillText(letters[i], elem.x,
59         elem.y + ((i <= point2DPyr.length - 3) ? 15 : -
60             10)));
61
62     };
63
64     NATask.setTask({
65         text: 'Объём правильной четырёхугольной пирамиды $SABCD$ равен $' +
66         pyr1.volume.pow(2).texsqrt(1) + '$. ' +
67         'Точка $E$ - середина ребра $SA$. Найдите объём треугольной пирамиды
68         $EABD$.',
69         answers: pyr2.volume,
70         author: ['Суматохина Александра'],
71     });
72     NATask.modifiers.variativeABC(letters);
73     NATask.modifiers.multiplyAnswerBySqrt(13);
74     NATask.modifiers.allDecimalsToStandard(true);
75     NATask.modifiers.assertSaneDecimals();
76     NATask.modifiers.addCanvasIllustration({
77         width: 400,
78         height: 400,
79         paint: paint1,
80     });
81
82 //27114 75015 75063 519535 75017 75019 75021 75023 75025 75027 75029 75031
83     75033 75035 75037 75039 75041 75043 75045 75047 75049 75051 75053 75055
84     75057 75059 75061

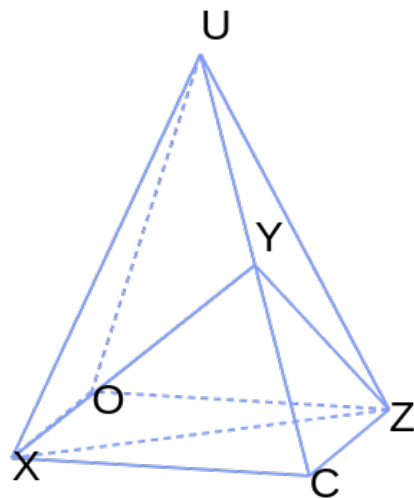
```

Листинг 8: 27114.js

Примеры генерируемых задач 27114.js

Объём правильной четырёхугольной пирамиды $UCXOZ$ равен 31164. Точка Y – середина ребра UC . Найдите объём треугольной пирамиды $YCXZ$.

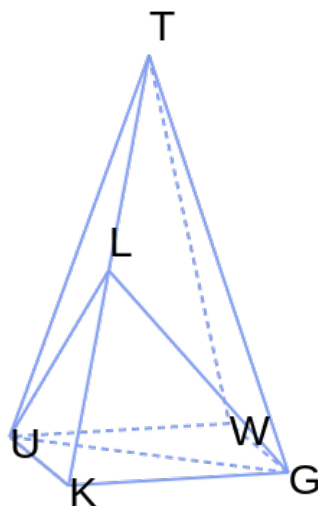
Ответ: 7791



Приложение. 16

Объём правильной четырёхугольной пирамиды $TKUWG$ равен 4800. Точка L – середина ребра TK . Найдите объём треугольной пирамиды $LKUG$.

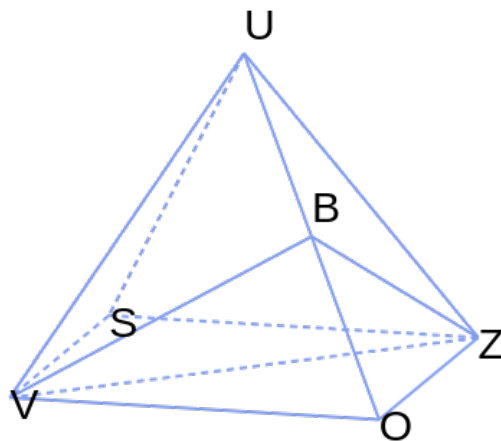
Ответ: 1200



Приложение. 17

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

Ответ: 6412,5



Приложение. 18

```

1 (function() {
2   retryWhileError(function() {
3
4     let pyr1 = new RegularPyramid({
5       height: sl(10, 30)*(3).sqrt(),
6       baseSide: sl(20, 50),
7       numberSide: 3
8     });
9
10    let pyr2 = new RegularPyramid({
11      height: pyr1.height,
12      baseSide: 0.5 * pyr1.baseSide,
13      numberSide: 3
14    });
15
16    pyr1.verticesOfFigure.push(coordinatesMiddleOfSegment3D(pyr1.verticesOfFigure[0],
17      pyr1.verticesOfFigure[1]));
18
19    pyr1.verticesOfFigure.push(coordinatesMiddleOfSegment3D(pyr1.verticesOfFigure[0],
20      pyr1.verticesOfFigure[2]));
21
22    let strok = [5, 4];
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: sl(1,2)* Math.PI / 8,
33    };
34  });
35 }());

```

```

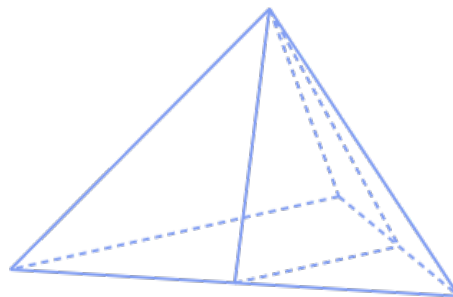
30 };
31
32 let point2DPyr = pyr1.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));
33
34 autoScale(pyr1.verticesOfFigure, camera, point2DPyr, {
35     startX: -180,
36     finishX: 160,
37     startY: -160,
38     finishY: 160,
39     maxScale: 50,
40 });
41
42 point2DPyr = pyr1.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));
43
44 let paint1 = function(ctx) {
45     let h = 400;
46     let w = 400;
47     ctx.translate(h / 2, w / 2);
48     ctx.lineWidth = 2;
49     ctx.strokeStyle = om.secondaryBrandColors;
50     ctx.drawFigure(point2DPyr, [
51         [1],
52         [strok, strok],
53         [1, 1, strok, 0, 1, strok],
54         [0, 0, 0, 0, 0, strok]
55     ]);
56 };
57
58 NATask.setTask({
59     text: ' Объем треугольной пирамиды равен $' + pyr1.volume + '$. ' +
60         'Через вершину пирамиды и среднюю линию её основания проведена плоскость
см(. рисунок). ' +
61         'Найдите объем отсечённой треугольной пирамиды.',
62     answers: pyr2.volume,
63     author: ['Суматохина Александра'],
64 });
65 NATask.modifiers.multiplyAnswerBySqrt(13);
66 NATask.modifiers.allDecimalsToStandard(true);
67 NATask.modifiers.assertSaneDecimals();
68 NATask.modifiers.addCanvasIllustration({
69     width: 400,
70     height: 400,
71     paint: paint1,
72 });
73 }, 1000);
74 })();
75 //27115 75065 75109 75113 514460 75067 75069 75071 75073 75075 75077 75079
75081 75083 75085 75087 75089 75091 75093 75095 75097 75099 75101 75103
75105 75107 75111

```

Примеры генерируемых задач 27115.js

Объём треугольной пирамиды равен 2560. Через вершину пирамиды и среднюю линию её основания проведена плоскость (см. рисунок). Найдите объём отсечённой треугольной пирамиды.

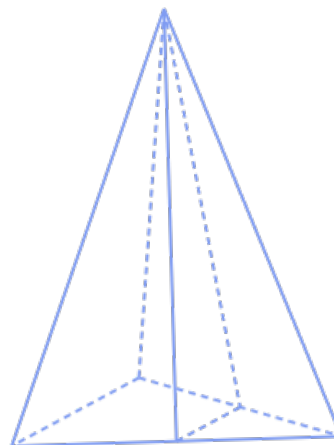
Ответ: 640



Приложение. 19

Объём треугольной пирамиды равен 4950. Через вершину пирамиды и среднюю линию её основания проведена плоскость (см. рисунок). Найдите объём отсечённой треугольной пирамиды.

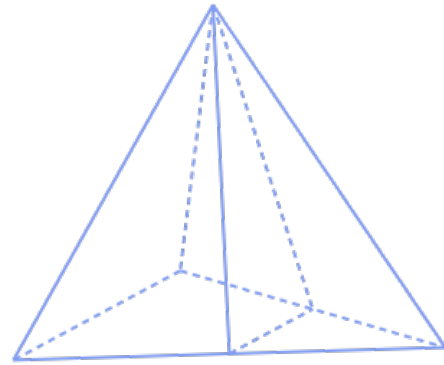
Ответ: 1237,5



Приложение. 20

Объём треугольной пирамиды равен 12096. Через вершину пирамиды и среднюю линию её основания проведена плоскость (см. рисунок). Найдите объём отсечённой треугольной пирамиды.

Ответ: 3024



Приложение. 21

```

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: [[3, stroke], 5, 7],
11      9: [1, 8, 10],
12      11: [8, 10, 12],
13      13: [5, 8, 12],
14      15: [7, 10, 12],
15    };
16
17    let par1 = new Parallelepiped({
18      depth: sl(10, 20),
19      height: sl(10, 20),
20      width: sl(10, 20),
21    });
22
23    let par2 = new Parallelepiped({
24      depth: slKrome(par1.depth, 5, par1.depth - 3),
25      height: slKrome(par1.height, 5, par1.height - 3),
26      width: slKrome(par1.width, 5, par1.width - 5),
27    });
28
29    let vertex3D =
30    par1.verticesOfFigure.concat(par2.verticesOfFigure.map((elem) =>
31      shiftCoordinate3D(elem, {
32        x: -0.5 * (par1.width - par2.width),
33        y: -0.5 * (par1.depth - par2.depth),

```

```

32     z: -0.5 * (par1.height - par2.height),
33   }));
34
35   let camera = {
36     x: 0,
37     y: 0,
38     z: 0,
39     scale: 1,
40
41     rotationX: -Math.PI / 2 + Math.PI / 14,
42     rotationY: 0,
43     rotationZ: Math.PI / sl(10, 14),
44   };
45
46   let point2D = vertex3D.map((coord3D) => project3DTo2D(coord3D,
47     camera));
48
49   autoScale(vertex3D, camera, point2D, {
50     startX: -180,
51     finishX: 160,
52     startY: -160,
53     finishY: 160,
54     maxScale: 100,
55   });
56
57   point2D = vertex3D.map((coord3D) => project3DTo2D(coord3D, camera));
58   genAssert((point2D[3].y - point2D[11].y).abs() > 20);
59   genAssert((point2D[3].y - point2D[8].y).abs() > 20);
60   genAssert((point2D[13].x - point2D[14].x).abs() > 20);
61
62   let rand = sl1();
63
64   let paint1 = function(ctx) {
65     let h = 400;
66     let w = 400;
67     ctx.translate(w / 2, h / 2);
68     ctx.lineWidth = 2;
69     ctx.strokeStyle = om.secondaryBrandColors;
70     ctx.drawFigureVer2(point2D, matrixConnections);
71
72     ctx.font = "20px liberation_sans";
73     ctx.signSegmentInMiddle(point2D[4].x, point2D[4].y, point2D[7].x,
74       point2D[7].y, par1.height, -10, 20);
75     ctx.signSegmentInMiddle(point2D[10].x, point2D[10].y,
76       point2D[15].x, point2D[15].y, par2.height, -20, 20);
77     ctx.signSegmentInMiddle(point2D[12].x, point2D[12].y,
78       point2D[15].x, point2D[15].y, par2.width, 20, 20);
79     ctx.signSegmentInMiddle(point2D[0].x, point2D[0].y, point2D[1].x,
80       point2D[1].y, par1.width, 18, 20);
81     ctx.signSegmentInMiddle(point2D[1].x, point2D[1].y, point2D[2].x,
82       point2D[2].y, par1.depth, 18, 20);
83     ctx.signSegmentInMiddle(point2D[9].x, point2D[9].y, point2D[10].x,

```



```

point2D[10].y, par2.depth, 15, 20);
78     };
79     NATask.setTask({
80         text: 'Найдите ' + ['площадь поверхности', 'объём'][rand] +
81             ' многогранника, изображённого на рисунке все( двугранные углы -
прямые).',
82         answers: [par1.surfaceArea, par1.volume - par2.volume][rand],
83     });
84     NATask.modifiers.addCanvasIllustration({
85         width: 400,
86         height: 400,
87         paint: paint1,
88     });
89 },
90 1000);
91
92 })();
93 //27193 25671 25673 25675 25677 25679

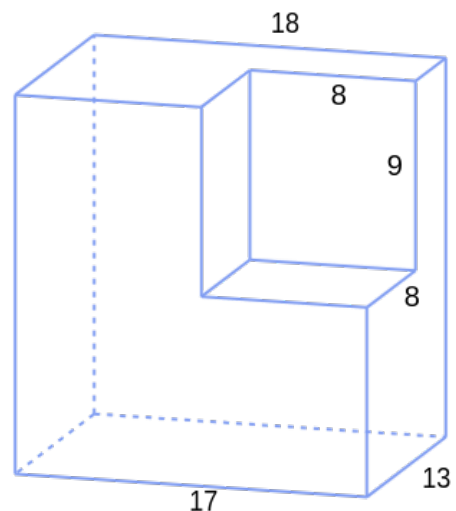
```

Листинг 10: 12.js

Примеры генерируемых задач 12.js

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

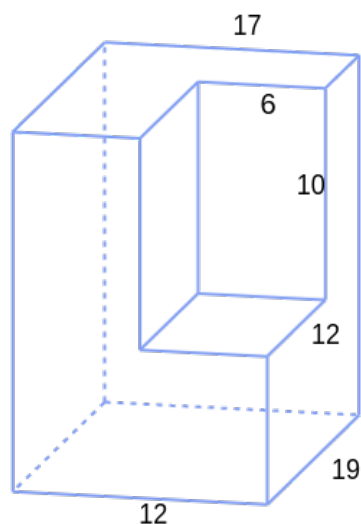
Ответ: 3402



Приложение. 22

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

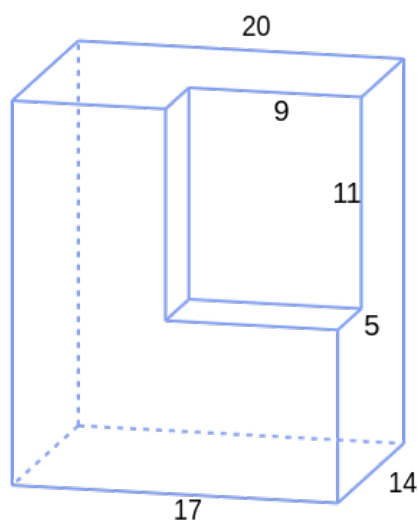
Ответ: 3156



Приложение. 23

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

Ответ: 4265



Приложение. 24

```
1 (function() {  
2   retryWhileError(function() {  
3     NAinfo.requireApiVersion(0, 2);  
4  
5     let stroke = [4, 2];  
6  
7     let matrixConnections = {  
8       0: [1, [3, stroke], 5],  
9       2: [1, [3, stroke], 7],  
10      4: [  
11        [3, stroke], 5, [11, stroke]  
12      ],  
13      6: [1, 7, 9],  
14      8: [5, [11, stroke], 13],  
15      10: [  
16        [3, stroke], 5, [11, stroke]  
17      ],  
18      12: [1, 7, 9],  
19      14: [5, [11, stroke], 13],  
20      16: [  
21        [3, stroke], 5, [11, stroke]  
22      ],  
23      18: [1, 7, 9],  
24      20: [5, [11, stroke], 13],  
25      22: [  
26        [3, stroke], 5, [11, stroke]  
27      ],  
28      24: [1, 7, 9],  
29      26: [5, [11, stroke], 13],  
30      28: [  
31        [3, stroke], 5, [11, stroke]  
32      ],  
33      30: [1, 7, 9],  
34      32: [5, [11, stroke], 13],  
35      34: [  
36        [3, stroke], 5, [11, stroke]  
37      ],  
38      36: [1, 7, 9],  
39      38: [5, [11, stroke], 13],  
40      40: [  
41        [3, stroke], 5, [11, stroke]  
42      ],  
43      42: [1, 7, 9],  
44      44: [5, [11, stroke], 13],  
45      46: [  
46        [3, stroke], 5, [11, stroke]  
47      ],  
48      48: [1, 7, 9],  
49      50: [5, [11, stroke], 13],  
50      52: [  
51        [3, stroke], 5, [11, stroke]  
52      ],  
53      54: [1, 7, 9],  
54      56: [5, [11, stroke], 13],  
55      58: [  
56        [3, stroke], 5, [11, stroke]  
57      ],  
58      60: [1, 7, 9],  
59      62: [5, [11, stroke], 13],  
60      64: [  
61        [3, stroke], 5, [11, stroke]  
62      ],  
63      66: [1, 7, 9],  
64      68: [5, [11, stroke], 13],  
65      70: [  
66        [3, stroke], 5, [11, stroke]  
67      ],  
68      72: [1, 7, 9],  
69      74: [5, [11, stroke], 13],  
70      76: [  
71        [3, stroke], 5, [11, stroke]  
72      ],  
73      78: [1, 7, 9],  
74      80: [5, [11, stroke], 13],  
75      82: [  
76        [3, stroke], 5, [11, stroke]  
77      ],  
78      84: [1, 7, 9],  
79      86: [5, [11, stroke], 13],  
80      88: [  
81        [3, stroke], 5, [11, stroke]  
82      ],  
83      90: [1, 7, 9],  
84      92: [5, [11, stroke], 13],  
85      94: [  
86        [3, stroke], 5, [11, stroke]  
87      ],  
88      96: [1, 7, 9],  
89      98: [5, [11, stroke], 13],  
90      100: [  
91        [3, stroke], 5, [11, stroke]  
92      ],  
93      102: [1, 7, 9],  
94      104: [5, [11, stroke], 13],  
95      106: [  
96        [3, stroke], 5, [11, stroke]  
97      ],  
98      108: [1, 7, 9],  
99      110: [5, [11, stroke], 13],  
100     112: [  
101       [3, stroke], 5, [11, stroke]  
102     ],  
103     114: [1, 7, 9],  
104     116: [5, [11, stroke], 13],  
105     118: [  
106       [3, stroke], 5, [11, stroke]  
107     ],  
108     120: [1, 7, 9],  
109     122: [5, [11, stroke], 13],  
110     124: [  
111       [3, stroke], 5, [11, stroke]  
112     ],  
113     126: [1, 7, 9],  
114     128: [5, [11, stroke], 13],  
115     130: [  
116       [3, stroke], 5, [11, stroke]  
117     ],  
118     132: [1, 7, 9],  
119     134: [5, [11, stroke], 13],  
120     136: [  
121       [3, stroke], 5, [11, stroke]  
122     ],  
123     138: [1, 7, 9],  
124     140: [5, [11, stroke], 13],  
125     142: [  
126       [3, stroke], 5, [11, stroke]  
127     ],  
128     144: [1, 7, 9],  
129     146: [5, [11, stroke], 13],  
130     148: [  
131       [3, stroke], 5, [11, stroke]  
132     ],  
133     150: [1, 7, 9],  
134     152: [5, [11, stroke], 13],  
135     154: [  
136       [3, stroke], 5, [11, stroke]  
137     ],  
138     156: [1, 7, 9],  
139     158: [5, [11, stroke], 13],  
140     160: [  
141       [3, stroke], 5, [11, stroke]  
142     ],  
143     162: [1, 7, 9],  
144     164: [5, [11, stroke], 13],  
145     166: [  
146       [3, stroke], 5, [11, stroke]  
147     ],  
148     168: [1, 7, 9],  
149     170: [5, [11, stroke], 13],  
150     172: [  
151       [3, stroke], 5, [11, stroke]  
152     ],  
153     174: [1, 7, 9],  
154     176: [5, [11, stroke], 13],  
155     178: [  
156       [3, stroke], 5, [11, stroke]  
157     ],  
158     180: [1, 7, 9],  
159     182: [5, [11, stroke], 13],  
160     184: [  
161       [3, stroke], 5, [11, stroke]  
162     ],  
163     186: [1, 7, 9],  
164     188: [5, [11, stroke], 13],  
165     190: [  
166       [3, stroke], 5, [11, stroke]  
167     ],  
168     192: [1, 7, 9],  
169     194: [5, [11, stroke], 13],  
170     196: [  
171       [3, stroke], 5, [11, stroke]  
172     ],  
173     198: [1, 7, 9],  
174     200: [5, [11, stroke], 13],  
175     202: [  
176       [3, stroke], 5, [11, stroke]  
177     ],  
178     204: [1, 7, 9],  
179     206: [5, [11, stroke], 13],  
180     208: [  
181       [3, stroke], 5, [11, stroke]  
182     ],  
183     210: [1, 7, 9],  
184     212: [5, [11, stroke], 13],  
185     214: [  
186       [3, stroke], 5, [11, stroke]  
187     ],  
188     216: [1, 7, 9],  
189     218: [5, [11, stroke], 13],  
190     220: [  
191       [3, stroke], 5, [11, stroke]  
192     ],  
193     222: [1, 7, 9],  
194     224: [5, [11, stroke], 13],  
195     226: [  
196       [3, stroke], 5, [11, stroke]  
197     ],  
198     228: [1, 7, 9],  
199     230: [5, [11, stroke], 13],  
200     232: [  
201       [3, stroke], 5, [11, stroke]  
202     ],  
203     234: [1, 7, 9],  
204     236: [5, [11, stroke], 13],  
205     238: [  
206       [3, stroke], 5, [11, stroke]  
207     ],  
208     240: [1, 7, 9],  
209     242: [5, [11, stroke], 13],  
210     244: [  
211       [3, stroke], 5, [11, stroke]  
212     ],  
213     246: [1, 7, 9],  
214     248: [5, [11, stroke], 13],  
215     250: [  
216       [3, stroke], 5, [11, stroke]  
217     ],  
218     252: [1, 7, 9],  
219     254: [5, [11, stroke], 13],  
220     256: [  
221       [3, stroke], 5, [11, stroke]  
222     ],  
223     258: [1, 7, 9],  
224     260: [5, [11, stroke], 13],  
225     262: [  
226       [3, stroke], 5, [11, stroke]  
227     ],  
228     264: [1, 7, 9],  
229     266: [5, [11, stroke], 13],  
230     268: [  
231       [3, stroke], 5, [11, stroke]  
232     ],  
233     270: [1, 7, 9],  
234     272: [5, [11, stroke], 13],  
235     274: [  
236       [3, stroke], 5, [11, stroke]  
237     ],  
238     276: [1, 7, 9],  
239     278: [5, [11, stroke], 13],  
240     280: [  
241       [3, stroke], 5, [11, stroke]  
242     ],  
243     282: [1, 7, 9],  
244     284: [5, [11, stroke], 13],  
245     286: [  
246       [3, stroke], 5, [11, stroke]  
247     ],  
248     288: [1, 7, 9],  
249     290: [5, [11, stroke], 13],  
250     292: [  
251       [3, stroke], 5, [11, stroke]  
252     ],  
253     294: [1, 7, 9],  
254     296: [5, [11, stroke], 13],  
255     298: [  
256       [3, stroke], 5, [11, stroke]  
257     ],  
258     300: [1, 7, 9],  
259     302: [5, [11, stroke], 13],  
260     304: [  
261       [3, stroke], 5, [11, stroke]  
262     ],  
263     306: [1, 7, 9],  
264     308: [5, [11, stroke], 13],  
265     310: [  
266       [3, stroke], 5, [11, stroke]  
267     ],  
268     312: [1, 7, 9],  
269     314: [5, [11, stroke], 13],  
270     316: [  
271       [3, stroke], 5, [11, stroke]  
272     ],  
273     318: [1, 7, 9],  
274     320: [5, [11, stroke], 13],  
275     322: [  
276       [3, stroke], 5, [11, stroke]  
277     ],  
278     324: [1, 7, 9],  
279     326: [5, [11, stroke], 13],  
280     328: [  
281       [3, stroke], 5, [11, stroke]  
282     ],  
283     330: [1, 7, 9],  
284     332: [5, [11, stroke], 13],  
285     334: [  
286       [3, stroke], 5, [11, stroke]  
287     ],  
288     336: [1, 7, 9],  
289     338: [5, [11, stroke], 13],  
290     340: [  
291       [3, stroke], 5, [11, stroke]  
292     ],  
293     342: [1, 7, 9],  
294     344: [5, [11, stroke], 13],  
295     346: [  
296       [3, stroke], 5, [11, stroke]  
297     ],  
298     348: [1, 7, 9],  
299     350: [5, [11, stroke], 13],  
300     352: [  
301       [3, stroke], 5, [11, stroke]  
302     ],  
303     354: [1, 7, 9],  
304     356: [5, [11, stroke], 13],  
305     358: [  
306       [3, stroke], 5, [11, stroke]  
307     ],  
308     360: [1, 7, 9],  
309     362: [5, [11, stroke], 13],  
310     364: [  
311       [3, stroke], 5, [11, stroke]  
312     ],  
313     366: [1, 7, 9],  
314     368: [5, [11, stroke], 13],  
315     370: [  
316       [3, stroke], 5, [11, stroke]  
317     ],  
318     372: [1, 7, 9],  
319     374: [5, [11, stroke], 13],  
320     376: [  
321       [3, stroke], 5, [11, stroke]  
322     ],  
323     378: [1, 7, 9],  
324     380: [5, [11, stroke], 13],  
325     382: [  
326       [3, stroke], 5, [11, stroke]  
327     ],  
328     384: [1, 7, 9],  
329     386: [5, [11, stroke], 13],  
330     388: [  
331       [3, stroke], 5, [11, stroke]  
332     ],  
333     390: [1, 7, 9],  
334     392: [5, [11, stroke], 13],  
335     394: [  
336       [3, stroke], 5, [11, stroke]  
337     ],  
338     396: [1, 7, 9],  
339     398: [5, [11, stroke], 13],  
340     400: [  
341       [3, stroke], 5, [11, stroke]  
342     ],  
343     402: [1, 7, 9],  
344     404: [5, [11, stroke], 13],  
345     406: [  
346       [3, stroke], 5, [11, stroke]  
347     ],  
348     408: [1, 7, 9],  
349     410: [5, [11, stroke], 13],  
350     412: [  
351       [3, stroke], 5, [11, stroke]  
352     ],  
353     414: [1, 7, 9],  
354     416: [5, [11, stroke], 13],  
355     418: [  
356       [3, stroke], 5, [11, stroke]  
357     ],  
358     420: [1, 7, 9],  
359     422: [5, [11, stroke], 13],  
360     424: [  
361       [3, stroke], 5, [11, stroke]  
362     ],  
363     426: [1, 7, 9],  
364     428: [5, [11, stroke], 13],  
365     430: [  
366       [3, stroke], 5, [11, stroke]  
367     ],  
368     432: [1, 7, 9],  
369     434: [5, [11, stroke], 13],  
370     436: [  
371       [3, stroke], 5, [11, stroke]  
372     ],  
373     438: [1, 7, 9],  
374     440: [5, [11, stroke], 13],  
375     442: [  
376       [3, stroke], 5, [11, stroke]  
377     ],  
378     444: [1, 7, 9],  
379     446: [5, [11, stroke], 13],  
380     448: [  
381       [3, stroke], 5, [11, stroke]  
382     ],  
383     450: [1, 7, 9],  
384     452: [5, [11, stroke], 13],  
385     454: [  
386       [3, stroke], 5, [11, stroke]  
387     ],  
388     456: [1, 7, 9],  
389     458: [5, [11, stroke], 13],  
390     460: [  
391       [3, stroke], 5, [11, stroke]  
392     ],  
393     462: [1, 7, 9],  
394     464: [5, [11, stroke], 13],  
395     466: [  
396       [3, stroke], 5, [11, stroke]  
397     ],  
398     468: [1, 7, 9],  
399     470: [5, [11, stroke], 13],  
400     472: [  
401       [3, stroke], 5, [11, stroke]  
402     ],  
403     474: [1, 7, 9],  
404     476: [5, [11, stroke], 13],  
405     478: [  
406       [3, stroke], 5, [11, stroke]  
407     ],  
408     480: [1, 7, 9],  
409     482: [5, [11, stroke], 13],  
410     484: [  
411       [3, stroke], 5, [11, stroke]  
412     ],  
413     486: [1, 7, 9],  
414     488: [5, [11, stroke], 13],  
415     490: [  
416       [3, stroke], 5, [11, stroke]  
417     ],  
418     492: [1, 7, 9],  
419     494: [5, [11, stroke], 13],  
420     496: [  
421       [3, stroke], 5, [11, stroke]  
422     ],  
423     498: [1, 7, 9],  
424     500: [5, [11, stroke], 13],  
425     502: [  
426       [3, stroke], 5, [11, stroke]  
427     ],  
428     504: [1, 7, 9],  
429     506: [5, [11, stroke], 13],  
430     508: [  
431       [3, stroke], 5, [11, stroke]  
432     ],  
433     510: [1, 7, 9],  
434     512: [5, [11, stroke], 13],  
435     514: [  
436       [3, stroke], 5, [11, stroke]  
437     ],  
438     516: [1, 7, 9],  
439     518: [5, [11, stroke], 13],  
440     520: [  
441       [3, stroke], 5, [11, stroke]  
442     ],  
443     522: [1, 7, 9],  
444     524: [5, [11, stroke], 13],  
445     526: [  
446       [3, stroke], 5, [11, stroke]  
447     ],  
448     528: [1, 7, 9],  
449     530: [5, [11, stroke], 13],  
450     532: [  
451       [3, stroke], 5, [11, stroke]  
452     ],  
453     534: [1, 7, 9],  
454     536: [5, [11, stroke], 13],  
455     538: [  
456       [3, stroke], 5, [11, stroke]  
457     ],  
458     540: [1, 7, 9],  
459     542: [5, [11, stroke], 13],  
460     544: [  
461       [3, stroke], 5, [11, stroke]  
462     ],  
463     546: [1, 7, 9],  
464     548: [5, [11, stroke], 13],  
465     550: [  
466       [3, stroke], 5, [11, stroke]  
467     ],  
468     552: [1, 7, 9],  
469     554: [5, [11, stroke], 13],  
470     556: [  
471       [3, stroke], 5, [11, stroke]  
472     ],  
473     558: [1, 7, 9],  
474     560: [5, [11, stroke], 13],  
475     562: [  
476       [3, stroke], 5, [11, stroke]  
477     ],  
478     564: [1, 7, 9],  
479     566: [5, [11, stroke], 13],  
480     568: [  
481       [3, stroke], 5, [11, stroke]  
482     ],  
483     570: [1, 7, 9],  
484     572: [5, [11, stroke], 13],  
485     574: [  
486       [3, stroke], 5, [11, stroke]  
487     ],  
488     576: [1, 7, 9],  
489     578: [5, [11, stroke], 13],  
490     580: [  
491       [3, stroke], 5, [11, stroke]  
492     ],  
493     582: [1, 7, 9],  
494     584: [5, [11, stroke], 13],  
495     586: [  
496       [3, stroke], 5, [11, stroke]  
497     ],  
498     588: [1, 7, 9],  
499     590: [5, [11, stroke], 13],  
500     592: [  
501       [3, stroke], 5, [11, stroke]  
502     ],  
503     594: [1, 7, 9],  
504     596: [5, [11, stroke], 13],  
505     598: [  
506       [3, stroke], 5, [11, stroke]  
507     ],  
508     600: [1, 7, 9],  
509     602: [5, [11, stroke], 13],  
510     604: [  
511       [3, stroke], 5, [11, stroke]  
512     ],  
513     606: [1, 7, 9],  
514     608: [5, [11, stroke], 13],  
515     610: [  
516       [3, stroke], 5, [11, stroke]  
517     ],  
518     612: [1, 7, 9],  
519     614: [5, [11, stroke], 13],  
520     616: [  
521       [3, stroke], 5, [11, stroke]  
522     ],  
523     618: [1, 7, 9],  
524     620: [5, [11, stroke], 13],  
525     622: [  
526       [3, stroke], 5, [11, stroke]  
527     ],  
528     624: [1, 7, 9],  
529     626: [5, [11, stroke], 13],  
530     628: [  
531       [3, stroke], 5, [11, stroke]  
532     ],  
533     630: [1, 7, 9],  
534     632: [5, [11, stroke], 13],  
535     634: [  
536       [3, stroke], 5, [11, stroke]  
537     ],  
538     636: [1, 7, 9],  
539     638: [5, [11, stroke], 13],  
540     640: [  
541       [3, stroke], 5, [11, stroke]  
542     ],  
543     642: [1, 7, 9],  
544     644: [5, [11, stroke], 13],  
545     646: [  
546       [3, stroke], 5, [11, stroke]  
547     ],  
548     648: [1, 7, 9],  
549     650: [5, [11, stroke], 13],  
550     652: [  
551       [3, stroke], 5, [11, stroke]  
552     ],  
553     654: [1, 7, 9],  
554     656: [5, [11, stroke], 13],  
555     658: [  
556       [3, stroke], 5, [11, stroke]  
557     ],  
558     660: [1, 7, 9],  
559     662: [5, [11, stroke], 13],  
560     664: [  
561       [3, stroke], 5, [11, stroke]  
562     ],  
563     666: [1, 7, 9],  
564     668: [5, [11, stroke], 13],  
565     670: [  
566       [3, stroke], 5, [11, stroke]  
567     ],  
568     672: [1, 7, 9],  
569     674: [5, [11, stroke], 13],  
570     676: [  
571       [3, stroke], 5, [11, stroke]  
572     ],  
573     678: [1, 7, 9],  
574     680: [5, [11, stroke], 13],  
575     682: [  
576       [3, stroke], 5, [11, stroke]  
577     ],  
578     684: [1, 7, 9],  
579     686: [5, [11, stroke], 13],  
580     688: [  
581       [3, stroke], 5, [11, stroke]  
582     ],  
583     690: [1, 7, 9],  
584     692: [5, [11, stroke], 13],  
585     694: [  
586       [3, stroke], 5, [11, stroke]  
587     ],  
588     696: [1, 7, 9],  
589     698: [5, [11, stroke], 13],  
590     700: [  
591       [3, stroke], 5, [11, stroke]  
592     ],  
593     702: [1, 7, 9],  
594     704: [5, [11, stroke], 13],  
595     706: [  
596       [3, stroke], 5, [11, stroke]  
597     ],  
598     708: [1, 7, 9],  
599     710: [5, [11, stroke], 13],  
600     712: [  
601       [3, stroke], 5, [11, stroke]  
602     ],  
603     714: [1, 7, 9],  
604     716: [5, [11, stroke], 13],  
605     718: [  
606       [3, stroke], 5, [11, stroke]  
607     ],  
608     720: [1, 7, 9],  
609     722: [5, [11, stroke], 13],  
610     724: [  
611       [3, stroke], 5, [11, stroke]  
612     ],  
613     726: [1, 7, 9],  
614     728: [5, [11, stroke], 13],  
615     730: [  
616       [3, stroke], 5, [11, stroke]  
617     ],  
618     732: [1, 7, 9],  
619     734: [5, [11, stroke], 13],  
620     736: [  
621       [3, stroke], 5, [11, stroke]  
622     ],  
623     738: [1, 7, 9],  
624     740: [5, [11, stroke], 13],  
625     742: [  
626       [3, stroke], 5, [11, stroke]  
627     ],  
628     744: [1, 7, 9],  
629     746: [5, [11, stroke], 13],  
630     748: [  
631       [3, stroke], 5, [11, stroke]  
632     ],  
633     750: [1, 7, 9],  
634     752: [5, [11, stroke], 13],  
635     754: [  
636       [3, stroke], 5, [11, stroke]  
637     ],  
638     756: [1, 7, 9],  
639     758: [5, [11, stroke], 13],  
640     760: [  
641       [3, stroke], 5, [11, stroke]  
642     ],  
643     762: [1, 7, 9],  
644     764: [5, [11, stroke], 13],  
645     766: [  
646       [3, stroke], 5, [11, stroke]  
647     ],  
648     768: [1, 7, 9],  
649     770: [5, [11, stroke], 13],  
650     772: [  
651       [3, stroke], 5, [11, stroke]  
652     ],  
653     774: [1, 7, 9],  
654     776: [5, [11, stroke], 13],  
655     778: [  
656       [3, stroke], 5, [11, stroke]  
657     ],  
658     780: [1, 7, 9],  
659     782: [5, [11, stroke], 13],  
660     784: [  
661       [3, stroke], 5, [11, stroke]  
662     ],  
663     786: [1, 7, 9],  
664     788: [5, [11, stroke], 13],  
665     790: [  
666       [3, stroke], 5, [11, stroke]  
667     ],  
668     792: [1, 7, 9],  
669     794: [5, [11, stroke], 13],  
670     796: [  
671       [3, stroke], 5, [11, stroke]  
672     ],  
673     798: [1, 7, 9],  
674     800: [5, [11, stroke], 13],  
675     802: [  
676       [3, stroke], 5, [11, stroke]  
677     ],  
678     804: [1, 7, 9],  
679     806: [5, [11, stroke], 13],  
680     808: [  
681       [3, stroke], 5, [11, stroke]  
682     ],  
683     810: [1, 7, 9],  
684     812: [5, [11, stroke], 13],  
685     814: [  
686       [3, stroke], 5, [11, stroke]  
687     ],  
688     816: [1, 7, 9],  
689     818: [5, [11, stroke], 13],  
690     820: [  
691       [3, stroke], 5, [11, stroke]  
692     ],  
693     822: [1, 7, 9],  
694     824: [5, [11, stroke], 13],  
695     826: [  
696       [3, stroke], 5, [11, stroke]  
697     ],  
698     828: [1, 7, 9],  
699     830: [5, [11, stroke],
```

```

16         [7, stroke],
17         [9, stroke],
18         [15, stroke]
19     ],
20     12: [
21         [11, stroke], 13, 15
22     ],
23     14: [9, 13, 15],
24 };
25
26 let par1 = new Parallelepiped({
27     depth: sl(10, 20),
28     height: sl(10, 20),
29     width: sl(10, 20),
30 });
31
32 let par2 = new Parallelepiped({
33     depth: par1.depth,
34     height: sl(5, par1.height - 4),
35     width: slKrome(par1.width, 5, par1.width - 5),
36 });
37
38 let deltaWidth = par1.width - par2.width;
39 let diagonal = (0.25 * deltaWidth.pow(2) +
40 par2.height.pow(2)).sqrt();
41
42 let vertex3D =
43 par1.verticesOfFigure.concat(par2.verticesOfFigure.map((elem) =>
44 shiftCoordinate3D(elem, {
45     x: 0,
46     y: 0,
47     z: -0.5 * (par1.height - par2.height),
48 })));
49
50 let camera = {
51     x: 0,
52     y: 0,
53     z: 0,
54     scale: 1,
55
56     rotationX: -Math.PI / 2 + Math.PI / 14,
57     rotationY: 0,
58     rotationZ: Math.PI / sl(12, 14),
59 };
60
61 let point2D = vertex3D.map((coord3D) => project3DTo2D(coord3D,
62 camera));
63
64 autoScale(vertex3D, camera, point2D, {
65     startX: -180,
66     finishX: 160,
67     startY: -160,

```

```

64     finishY: 160,
65     maxScale: 100,
66 });
67
68     point2D = vertex3D.map((coord3D) => project3DTo2D(coord3D, camera));
69     genAssert((point2D[4].x - point2D[8].x).abs() > 20);
70     genAssert((point2D[4].y - point2D[13].y).abs() > 50);
71     genAssert((point2D[12].x - point2D[14].x).abs() > 40);
72     genAssert([point2D[0], point2D[3], point2D[8]].mt_is3ug(), 'Точки
лежат на одной прямой');
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         let point = [];
85         ctx.drawLine(point2D[4].x, point2D[4].y, point.x, point.y);
86         if (point2D[6].x < point2D[15].x)
87             point = [point2D[10], point2D[15], point2D[6],
point2D[7]].mt_coordinatesOfIntersectionOfTwoSegments();
88         else
89             point = [point2D[10], point2D[15], point2D[6],
point2D[9]].mt_coordinatesOfIntersectionOfTwoSegments();
90         ctx.drawLine(point2D[15].x, point2D[15].y, point.x, point.y);
91
92         point = [point2D[12], point2D[13], point2D[4],
point2D[11]].mt_coordinatesOfIntersectionOfTwoSegments();
93         ctx.drawLine(point2D[4].x, point2D[4].y, point.x, point.y);
94
95         ctx.font = "20px liberation_sans";
96         ctx.signSegmentInMiddle(point2D[2].x, point2D[2].y, point2D[7].x,
point2D[7].y, par1.height, 10, 20);
97         ctx.signSegmentInMiddle(point2D[9].x, point2D[9].y, point2D[14].x,
point2D[14].y, par2.height, -24, 20);
98         ctx.signSegmentInMiddle(point2D[12].x, point2D[12].y,
point2D[15].x, point2D[15].y, par2.width, -10, 20);
99         ctx.signSegmentInMiddle(point2D[0].x, point2D[0].y, point2D[1].x,
point2D[1].y, par1.width, 18, 20);
100        ctx.signSegmentInMiddle(point2D[1].x, point2D[1].y, point2D[2].x,
point2D[2].y, par1.depth, 18, 20);
101    };
102
103    NATask.setTask({
104        text: 'Найдите ' + ['площадь поверхности', 'объём'][rand] + '
многогранника, изображённого на рисунке.',
105        answers: [par1.surfaceArea - deltaWidth * (par2.height +

```

```

106     par2.depth) + 2 * par2.height * par2.depth + 2 *
107         diagonal * par2.depth,
108     par1.volume - par2.height * (par1.width - par2.width) *
109     par2.width
110     ][rand],
111     author: ['Суматохина Александра']
112     });
113     NAtask.modifiers.multiplyAnswerBySqrt(13);
114     NAtask.modifiers.addCanvasIllustration({
115         width: 400,
116         height: 400,
117         paint: paint1,
118     });
119     },
120     1000);
121 //144526144

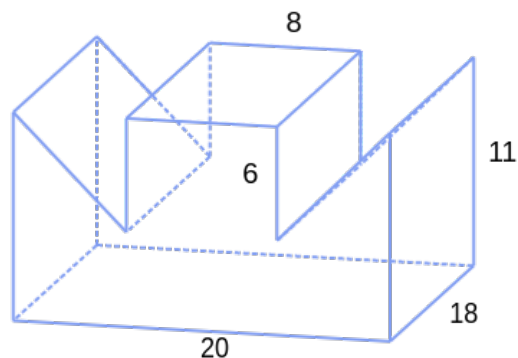
```

Листинг 11: 144526144.js

Примеры генерируемых задач 144526144.js

Найдите площадь поверхности многогранника, изображённого на рисунке.

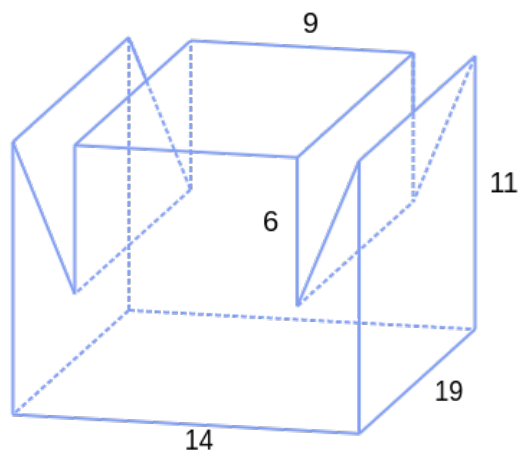
Ответ: 3384



Приложение. 25

Найдите площадь поверхности многогранника, изображённого на рисунке.

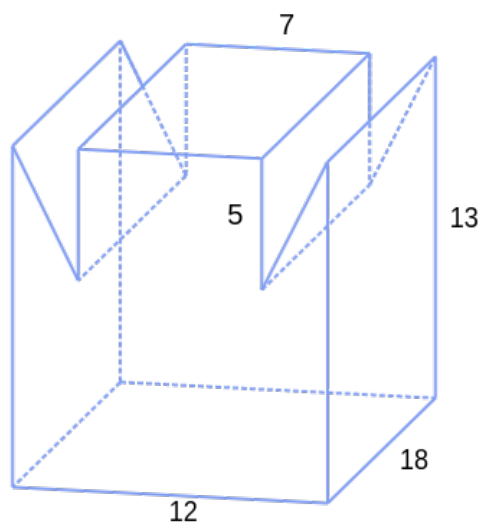
Ответ: 2656



Приложение. 26

Найдите площадь поверхности многогранника, изображённого на рисунке.

Ответ: 2633



Приложение. 27