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 геометри 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

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 приводят 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 соответствуют 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 B»). [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 это 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 2 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 3 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: 3.666666666666667, 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 4 в строке 74 для отображения условий задачи в случайном порядке.

```
1     let array = ['A', 'B', 'C', 'D',];
2     array.shuffleJoin();
```

```
    //ADBC

    array.shuffleJoin('; ');
    //C; D; B; A
```

`Array.prototype.joinWithConjunction = function(separator)`
Соединяет массив запятыми и соединяет два последних элемента союзом «и».

```
    let array = ['A', 'B', 'C', 'D',];

    array.joinWithConjunction();
    //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.

```
    let number = 81;

    number.perfectCubicMultiplier()
    //3

    number = 36;
    number.perfectCubicMultiplier()
    //1

    number = -27;
    number.perfectCubicMultiplier()
    //3
```

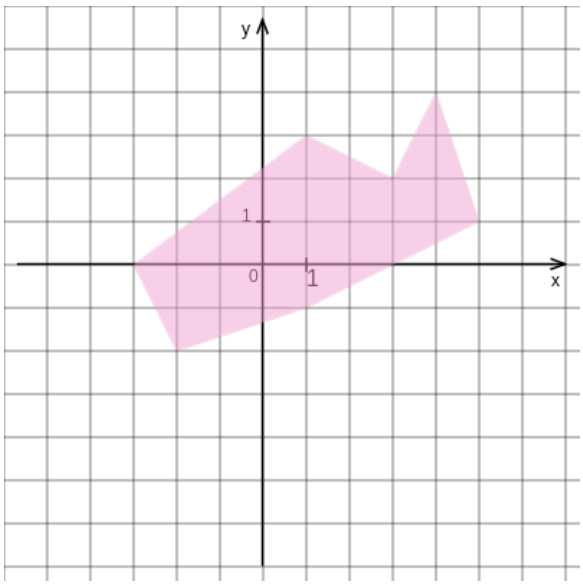
`Number.prototype.texcbrt = function(p1, p2)`
TeX-представление кубического корня из данного числа.
Если данное число - полный куб, то корень из числа.
Если `p1`, то из-под корня будут вынесены возможные множители.
Если `p1`, `p2` и из-под корня выносится единица, то она будет опущена

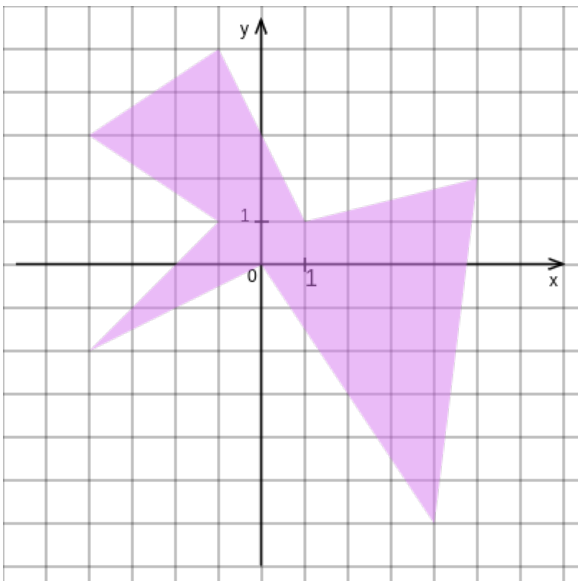
1.1.3. Функции для работы с canvas

`CanvasRenderingContext2D.prototype.drawSection = function(vertex, fillStyle)`
Заполняет область цветом `fillStyle` по вершинам из массива `vertex`.

```
    let paint1 = function(ctx) {
        let h = 400;
        let w = 400;
        ctx.drawCoordinatePlane(w, h, {
            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,
    2]]);
14
15          ctx.drawSection([[-2, 0],[-1, 1],[-4, 3],[-1, 5],[1, 1],[5, 2],[4,
    -6],[0, 0],[-4, -2],]);
16      };
17
```





`CanvasRenderingContext2D.prototype.drawLineAtAngle = function(x, y, angle, length)`
Рисует отрезок длины `length` под углом angle (в радианах). Пример использования в листинге 5 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 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.

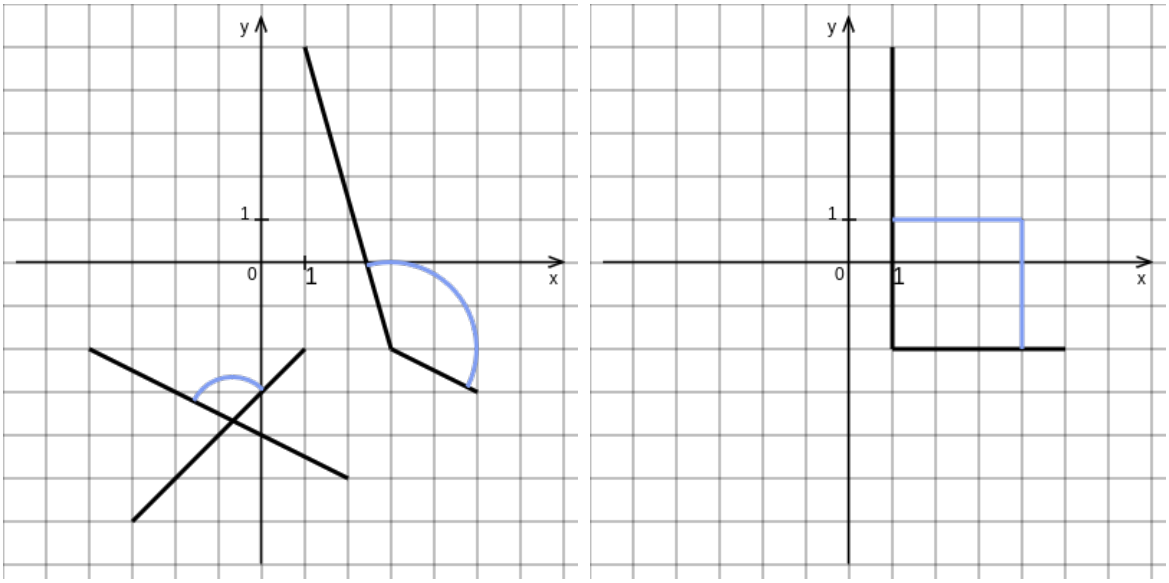
`CanvasRenderingContext2D.prototype.signSegmentInMiddle = function(x1, y1, x2, y2, letter, length, maxLength)`
Рисует строку `letter` на середине отрезка. В листинге 3 в строках 94 - 98 функция используется для отображения длин рёбер многогранника.

`CanvasRenderingContext2D.prototype.arcBetweenSegments`

= function(coordinates, radius)

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

```
    let paint1 = function(ctx) {
        let h = 400;
        let w = 400;
        ctx.drawCoordinatePlane(w, h, {
            hor: 1,
            ver: 1
        }, {
            x1: '1',
            y1: '1',
            sh1: 16,
        }, 30);
        ctx.scale(30, -30);

        ctx.lineWidth = 2 / 30;
        ctx.drawLine(1, 5, 3, -2);
        ctx.drawLine(3, -2, 5, -3);
        ctx.arcBetweenSegments([1, 5, 3, -2, 5, -3], 2);

        ctx.drawLine(2, -5, -4, -2);
        ctx.drawLine(1, -2, -3, -6);
        ctx.arcBetweenSegments([2, -5, -4, -2,  -3, -6,1, -2,], 1);

        ctx.drawLine(1, 5, 1, -2);
    ctx.drawLine(1, -2, 5, -2);
    ctx.strokeStyle = om.secondaryBrandColors.iz();
    ctx.arcBetweenSegments([1, 5, 1, -2, 5, -2], 3);

    };
```



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 6 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.

```
(function () {
  retryWhileError(function () {
    NAinfo.requireApiVersion(0, 2);

    let paint1 = function (ctx) {
    };

    NAtask.setTask({
      text: 'В треугольнике ABC AC=BC, AB=15, AHвысота-, BH=5. Найдите косинус
   ABC',
      answers: 0,
      author: ['Суматохина Александра']
    });
    NAtask.modifiers.addCanvasIllustration({
      width: 400,
      height: 400,
      paint: paint1,
    });
  }, 1000);
})();
```

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

```
(function () {
  retryWhileError(function () {
    NAinfo.requireApiVersion(0, 2);

    let paint1 = function (ctx) {

      ctx.lineWidth = 2;
```

```
      ctx.strokeStyle = om.secondaryBrandColors;

      ctx.drawLine(10, 370, 390, 370);
      ctx.drawLine(10, 370, 180, 50);
      ctx.drawLine(180, 50, 390, 370);

      ctx.drawLine(280, 200, 10, 370);
    };

    NAtask.setTask({
      text: 'В треугольнике ABC AC=BC, AB=15, AHвысота-, BHНайдите=5.
косинус ABC.',
      answers: 0,
      author: ['Суматохина Александра']
    });
    NAtask.modifiers.addCanvasIllustration({
      width: 400,
      height: 400,
      paint: paint1,
    });
  }, 1000);
})();
```

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

```
(function () {
  retryWhileError(function () {
    NAinfo.requireApiVersion(0, 2);

    let vertices = ['A', 'B', 'C', 'H'];

    let paint1 = function (ctx) {

      ctx.lineWidth = 2;
      ctx.strokeStyle = om.secondaryBrandColors;

      ctx.drawLine(10, 370, 390, 370);
      ctx.drawLine(10, 370, 180, 50);
      ctx.drawLine(180, 50, 390, 370);

      ctx.drawLine(280, 200, 10, 370);

      ctx.strokeInMiddleOfSegment(180, 50, 10, 370, 10);
      ctx.strokeInMiddleOfSegment(180, 50, 390, 370, 10);

      ctx.arcBetweenSegments([180, 50, 280, 200, 280, 200, 10, 370],
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, АНвысота-, ВННайдите=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],
      25);
26
27        ctx.font = "23px liberation_sans";
28        ctx.fillText(vertices[0], 10 - 5, 370 + 25);
29        ctx.fillText(vertices[1], 180, 50 - 10);
30        ctx.fillText(vertices[2], 390 - 10, 370 + 25);
31        ctx.fillText(vertices[3], 280+10, 200);
32      };
33
34      NAtask.setTask({
35        text: 'В треугольнике $ABC$ '+['$AC=BC$','$AB='+a+'$',
      '$AHвысота-$', '$BH='+b+'$'].shuffleJoin(', ')+'. Найдите косинус
      $BAC$.',
36        answers: b / a,
37        author: ['Суматохина Александра']
38      });
39      NAtask.modifiers.variativeABC(vertices);
40      NAtask.modifiers.addCanvasIllustration({
41        width: 400,
42        height: 400,
43        paint: paint1,
44      });
45    }, 1000);
46  })();
```

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 (Более 2 GB). При этом

```
import * as THREE from 'three';

// Создать сцену, камеру и рендерер
const scene = new THREE.Scene();
const camera = new THREE.PerspectiveCamera(75, window.innerWidth /
   window.innerHeight, 0.1, 1000);
camera.position.z = 2;
const renderer = new THREE.WebGLRenderer();
renderer.setSize(window.innerWidth, window.innerHeight);

// Создать геометрию куба
const geometry =new THREE.BoxGeometry(1, 1, 1);

// Создать материал куба
const material = new THREE.MeshBasicMaterial({ color: 0x00ff00 });

// Создать куб
const cube = new THREE.Mesh(geometry, material);

// Добавить куб в сцену
scene.add(cube);

// Отрендерить сцену
renderer.render(scene, camera);
```

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 1: 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.

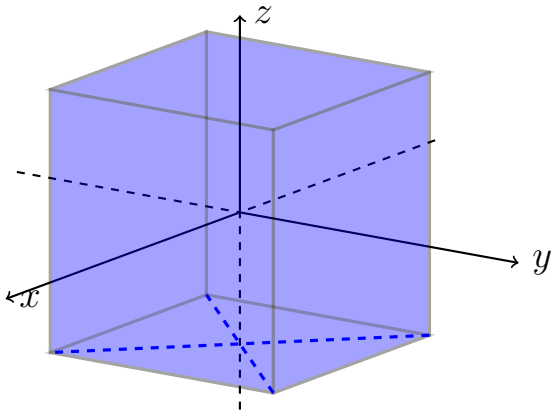


Рис. 1: Каноническое положение для параллелепипеда

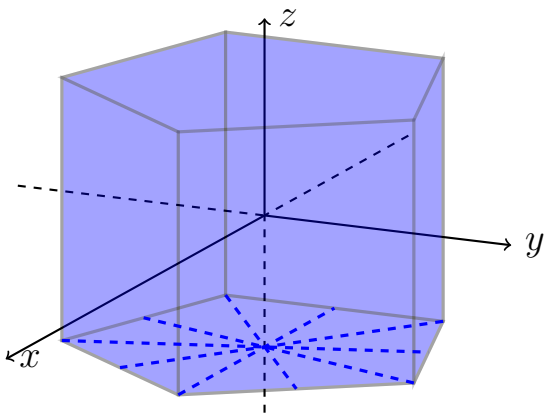


Рис. 2: Каноническое положение для правильной пятиугольной призмы

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 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, matrixC`
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`, которая является объектом с числовыми полями (номерами вершин), которые содержат в себе масив с номерами вершин для связи с ними.

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],
};

```

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

Для примера возьмём задание 27074

Заготовка шаблона имееет вид.

```
(function () {
  retryWhileError(function () {
    NAinfo.requireApiVersion(0, 2);

    let paint1 = function (ctx) {
    };

    NAtask.setTask({
      text: 'Объем параллелепипеда ABCDA_1B_1C_1D_1 равен 9. Найдите объем
   треугольной пирамиды ABCA_1.',
      answers: 0,
      author: ['Суматохина Александра']
    });
    NAtask.modifiers.addCanvasIllustration({
      width: 400,
      height: 400,
      paint: paint1,
    });
  }, 100000);
})();
```

1. Создадим объект класса `Parallelepiped` со случанной высотой, шириной и глубиной в заданном диапазоне.

```
(function () {
```

```
  retryWhileError(function () {
    NAinfo.requireApiVersion(0, 2);

    let par = new Parallelepiped({
      depth: sl(10, 50),
      height: sl(10, 50),
      width: sl(10, 50),
    });

    let paint1 = function (ctx) {
    };

    NAtask.setTask({
      text: 'Объем параллелепипеда ABCDA_1B_1C_1D_1 равен 9. Найдите
   объем треугольной пирамиды ABCA_1.',
      answers: 0,
      author: ['Суматохина Александра']
    });
    NAtask.modifiers.addCanvasIllustration({
      width: 400,
      height: 400,
      paint: paint1,
    });
  }, 100000);
})();
```

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`.

```
(function () {
    retryWhileError(function () {
        NAinfo.requireApiVersion(0, 2);

        let par = new Parallelepiped({
            depth: sl(10, 50),
            height: sl(10, 50),
            width: sl(10, 50),
        });

        let camera = {
            x: 0,
            y: 0,
            z: 0,
            scale: 5,

            rotationX: -Math.PI / 2 + Math.PI / 14,
            rotationY: 0,
            rotationZ: 2 * Math.PI / 3,
```

```
        };

        let point2DPar = par.verticesOfFigure.map((coord3D) =>
    project3DTo2D(coord3D, camera));

        autoScale(par.verticesOfFigure, camera, point2DPar, {
            startX: -180,
            finishX: 160,
            startY: -160,
            finishY: 160,
            maxScale: 50,
        });

        point2DPar = par.verticesOfFigure.map((coord3D) =>
    project3DTo2D(coord3D, camera));

        let paint1 = function (ctx) {
        };

        NAtask.setTask({
            text: 'Объем параллелепипеда ABCDA_1B_1C_1D_1 равен 9.
    Найдите объем треугольной пирамиды ABCA_1.',
            answers: 0,
            author: ['Суматохина Александра']
        });
        NAtask.modifiers.addCanvasIllustration({
            width: 400,
            height: 400,
            paint: paint1,
        });
    }, 100000);
})();
```

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

```
(function () {
    retryWhileError(function () {
        NAinfo.requireApiVersion(0, 2);

        let par = new Parallelepiped({
            depth: sl(10, 50),
            height: sl(10, 50),
            width: sl(10, 50),
        });

        let camera = {
            x: 0,
            y: 0,
            z: 0,
            scale: 5,

```

```
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 ABCA_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).

```
(function() {
  retryWhileError(function() {
    NAinfo.requireApiVersion(0, 2);

    let par = new Parallelepiped({
      depth: sl(10, 50),
      height: sl(10, 50),
      width: sl(10, 50),
    });

    let pyr = new Pyramid({
      height: par.height,
      baseArea: 0.5 * par.baseArea,
    });

    let camera = {
      x: 0,
      y: 0,
      z: 0,
      scale: 5,

      rotationX: -Math.PI / 2 + Math.PI / 14,
      rotationY: 0,
      rotationZ: 2 * Math.PI / 3,
    };

    let point2DPar = par.verticesOfFigure.map((coord3D) =>
   project3DTo2D(coord3D, camera));

    autoScale(par.verticesOfFigure, camera, point2DPar, {
      startX: -180,
      finishX: 160,
      startY: -160,
      finishY: 160,
      maxScale: 50,
    });

    point2DPar = par.verticesOfFigure.map((coord3D) =>
   project3DTo2D(coord3D, camera));

    let letter = ['A', 'B', 'C', 'D', '□D', '□A', '□B', '□C',];

    let paint1 = function(ctx) {
      let h = 400;
      let w = 400;
      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 $ABCDA_1B_1C_1D_1$ 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

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 $\mathbb{R}^3 \to \mathbb{R}^2$.

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

Приложение

```
(function() {
  retryWhileError(function() {
    NAinfo.requireApiVersion(0, 2);

    let a = sl(2, 89);
    let b = slKrome(a, 1, a - 1);

        let vertices = window.latbukv.iz(4);

    let angle = sl1() ? vertices.slice(0, 3).permuteCyclic(1) :
   vertices.slice(0, 3).permuteCyclic(2);

    genAssertZ1000(b / a, 'Кривой ответ');

    let paint1 = function(ctx) {
      ctx.lineWidth = 2;
      ctx.strokeStyle = om.primaryBrandColors [0];

      ctx.drawLine(10, 370, 390, 370);
      ctx.drawLine(10, 370, 180, 200);
      ctx.drawLine(180, 200, 390, 370);

      //высота
      ctx.lineWidth = 1.2;
      ctx.drawLine(180 - 40, 200 - 33, 390, 370);
      ctx.drawLine(180 - 40, 200 - 33, 10, 370);

      //прямой угол
      ctx.strokeStyle = om.secondaryBrandColors.iz();
      ctx.drawLine(180 - 52, 200 - 13, 180 - 35, 200 - 1);
      ctx.drawLine(180 - 35, 200 - 1, 180 - 23, 200 - 17);

      //штрихи
      ctx.drawLine(275 + 10, 210 + 90, 300, 200 + 90);
      ctx.drawLine(80, 200 + 90, 105 - 10, 210 + 90);

      ctx.font = "23px liberation_sans";
      ctx.fillText(vertices[0], 10 - 5, 370 + 25);
      ctx.fillText(vertices[1], 180, 200 - 10);
      ctx.fillText(vertices[2], 390 - 10, 370 + 25);
      ctx.fillText(vertices[3], 180 - 40, 200 - 40);

    };

    NAtask.setTask({
      text: 'В треугольнике  $' + vertices.slice(0, 3).shuffle().join('') +
  '$ $' +
        vertices.slice(0, 2).shuffle().join('') +
```

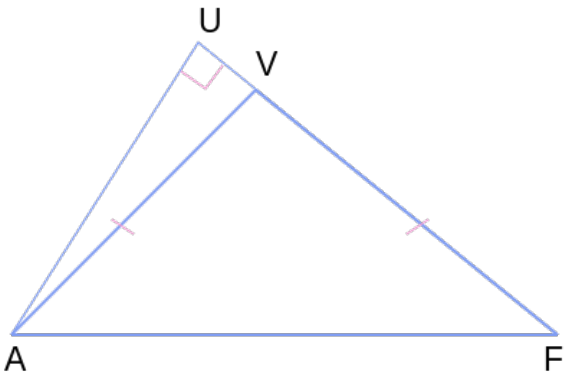
```
47          '=' + vertices.slice(1, 3).shuffle().join('') + '$, $' +
    vertices[0] + vertices[2] + '=' + a + '$, высота $' +
48          [vertices[0], vertices[3]].shuffle().join('') +
49          '$ равна $' + b + '$. Найдите $\\sin \\angle ' + angle.join('') +
    '$.',
50        answers: b /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

Листинг 2: 105.js

Примеры генерируемых задач 105.js

В треугольнике VAF $VA = VF$, $AF = 75$, высота AU равна 39. Найдите $\sin \angle VFA$.

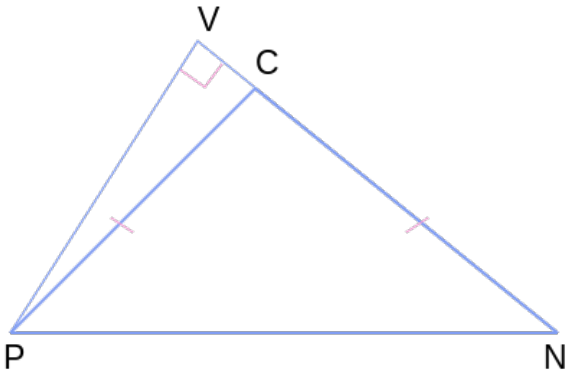
Ответ: 0,52



Приложение. 1

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

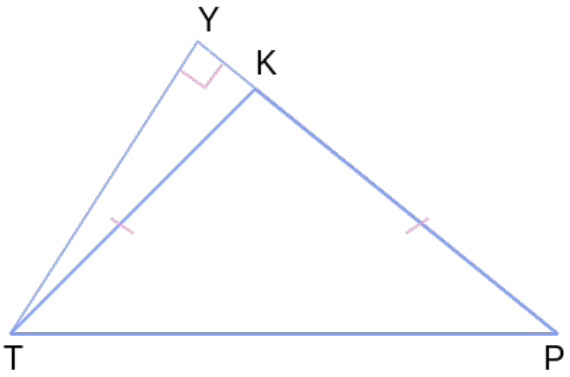
Ответ: 0,15



Приложение. 2

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

Ответ: 0,5



Приложение. 3

```
(function() {
  retryWhileError(function() {
      NAinfo.requireApiVersion(0, 2);

      let stroke = [4, 5];

      let matrixConnections = {
        0: [1, [3, stroke], 5],
        2: [1, [3, stroke], 7],
        4: [
          [3, stroke],
          [5, stroke],
          [11, stroke]
        ],
        6: [1, 7, 9],
```

```
      8: [5, [11, stroke], 13],
      10: [7, 9, 15],
      12: [
        [11, stroke], 13, 15
      ],
      14: [9, 13, 15],
    };

    let par1 = new Parallelepiped({
      depth: sl(10, 20),
      height: sl(5, 20),
      width: sl(10, 20),
    });

    let par2 = new Parallelepiped({
      depth: par1.depth,
      height: sl(5, 20),
      width: slKrome(par1.width, 5, par1.width - 5),
    });

    let vertex3D =
par1.verticesOfFigure.concat(par2.verticesOfFigure.map((elem) =>
shiftCoordinate3D(elem, {
      x: 0,
      y: 0,
      z: -0.5 * (par1.height + par2.height),
    })));

    vertex3D = vertex3D.map((elem) => shiftCoordinate3D(elem, {
      x: 0,
      y: 0,
      z: 0.5 * par2.height,
    }));

    let camera = {
      x: 0,
      y: 0,
      z: 0,
      scale: 1,

      rotationX: -Math.PI / 2 + Math.PI / 14,
      rotationY: 0,
      rotationZ: Math.PI / sl(10, 14),
    };

    let point2D = vertex3D.map((coord3D) => project3DTo2D(coord3D,
camera));

    autoScale(vertex3D, camera, point2D, {
      startX: -180,
      finishX: 160,
      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,
```

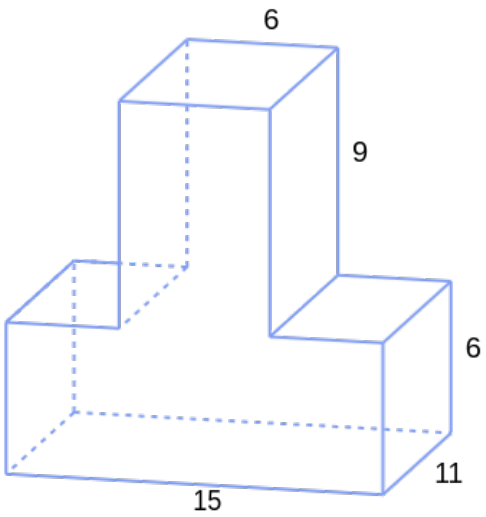
```
        height: 400,
        paint: paint1,
      });
    },
    1000);

})();
//27193 25671 25673 25675 25677 25679
```

Листинг 3: 29193.js

Примеры генерируемых задач 27193.js

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

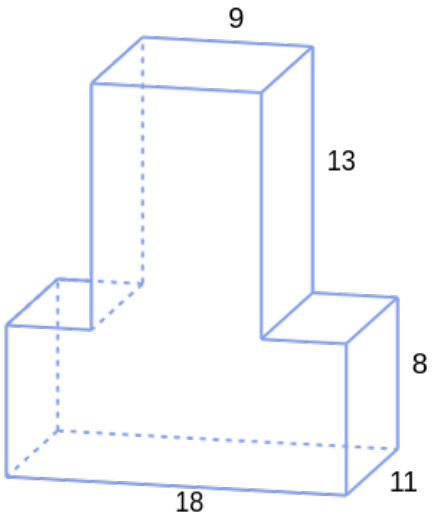
Ответ: 948



Приложение. 4

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

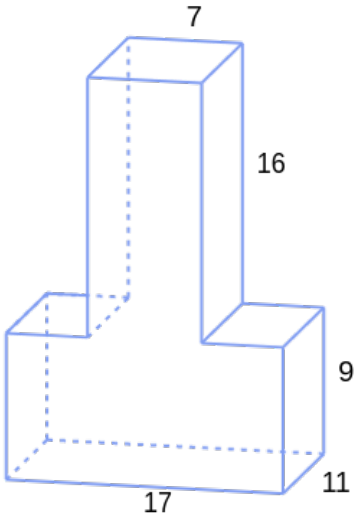
Ответ: 2871



Приложение. 5

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

Ответ: 1454



Приложение. 6

```
(function() {
  retryWhileError(function() {
    lx_declareClarifiedPhrase('сторона', 'основания');
    NAinfo.requireApiVersion(0, 2);

    let pyr = new RegularPyramid({
      height: sl(20, 50),
      baseSide: sl(20, 40),
      numberSide: 4
    });

    pyr.verticesOfFigure.push({
      x: 0,
      y: 0,
      z: pyr.verticesOfFigure[0].z
    });

    let question = [
      [sklonlxkand('боковое ребро'), pyr.sideEdge],
      [sklonlxkand('объём'), pyr.volume],
    ].shuffle();
    question.unshift([sklonlxkand('сторона основания'), pyr.baseSide]);

    let camera = {
      x: 0,
      y: 0,
      z: 0,
      scale: 5,

      rotationX: -Math.PI / 2 + Math.PI / 14,
      rotationY: 0,
      rotationZ: [1, 2].iz() * Math.PI / 3,
    };
```

```
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      });
```

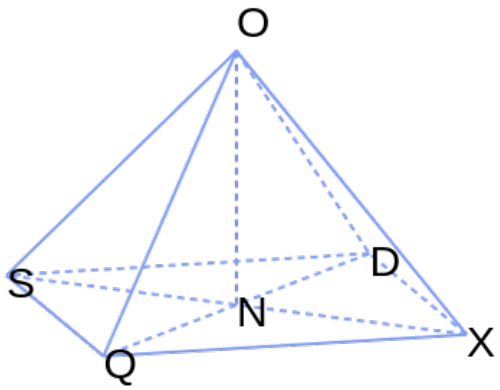
```
    NAtask.modifiers.variativeABC(letters);
    NAtask.modifiers.multiplyAnswerBySqrt(13);
    NAtask.modifiers.allDecimalsToStandard(true);
    NAtask.modifiers.assertSaneDecimals();
    NAtask.modifiers.addCanvasIllustration({
      width: 400,
      height: 400,
      paint: paint1,
    });
  },10);
})();
//https://ege314.ru/8-stereometriya-ege/reshenie-3011/
//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.

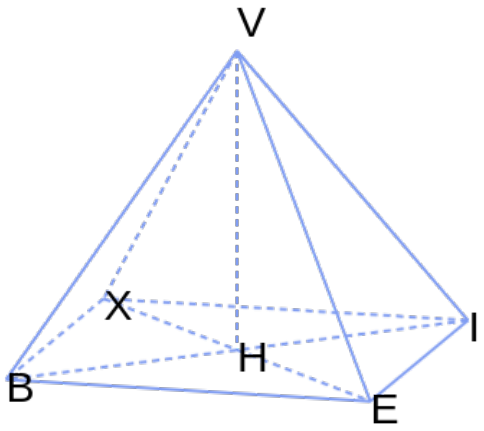
Ответ: 13689



Приложение. 7

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

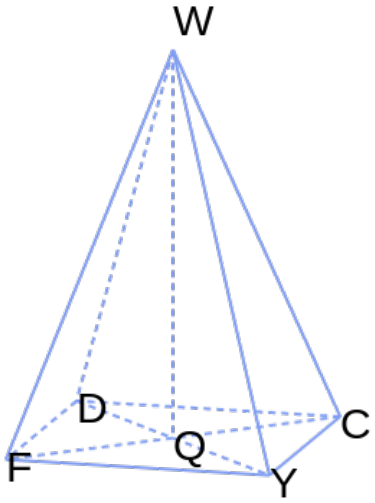
Ответ: 5346



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

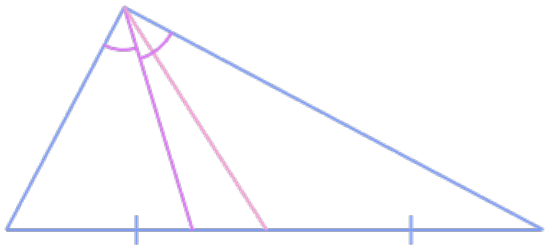
```
(function() {
  retryWhileError(function() {
    NAinfo.requireApiVersion(0, 2);

    let angle = sl(2, 44);
    let condition = [
      ['больший', 45 + angle],
      ['меньший', 45 - angle]
    ].iz();

    let paint1 = function(ctx) {
      ctx.lineWidth = 2;

      let angle = Math.PI/2.9;

```

```
      ctx.strokeStyle = om.secondaryBrandColors.iz();
      ctx.drawLine(10, 250, 390-8, 250);
      let ver = ctx.drawLineAtAngle(10, 250, -angle, 200-25);
      ctx.drawLineAtAngle(ver.x, ver.y, -angle+Math.PI/2, 350-20);
      //штрихи
      ctx.strokeInMiddleOfSegment(10, 250, (390-8)/2, 250, 10);
      ctx.strokeInMiddleOfSegment(10, 250, 3*(390-8)/2, 250, 10);
      //биссектриса
      ctx.strokeStyle = om.primaryBrandColors [0];
      let bis = ctx.drawLineAtAngle(ver.x, ver.y,
  (-angle+Math.PI/2)+Math.PI/4, 160+2);
      //медиана
      ctx.strokeStyle = om.primaryBrandColors [1];
      ctx.drawLine(ver.x, ver.y,(390-8)/2, 250);

      ctx.strokeStyle = om.primaryBrandColors [0];
      ctx.arcBetweenSegments([10, 250,ver.x, ver.y, bis.x, bis.y], 30);
      ctx.arcBetweenSegments([390-8, 250,ver.x, ver.y, bis.x, bis.y], 38);

    };

    NAtask.setTask({
      text: 'Угол между биссектрисой и медианой прямоугольного треугольника, ' +
        'проведёнными из вершины прямого угла, равен $' + angle + '^{\\circ}$.
  ' +
        'Найдите ' + condition[0] + ' угол прямоугольного треугольника. Ответ
  дайте в градусах.',
      answers: condition[1],
      analys: '',
    });
    NAtask.modifiers.addCanvasIllustration({
      width: 400,
      height: 400,
      paint: paint1,
    });
  }, 1000);
})();
//2069
```

Листинг 5: 2069.js

Примеры генерируемых задач 2069.js

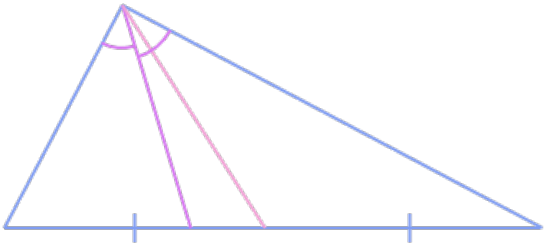
Угол между биссектрисой и медианой прямоугольного треугольника, проведёнными из вершины прямого угла, равен 38°. Найдите больший угол прямоугольного треугольника. Ответ дайте в градусах.



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.

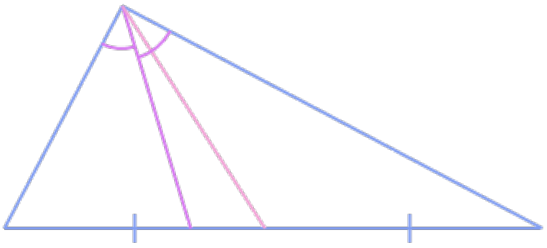


Ответ: 37

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

```
(function() {
  retryWhileError(function() {
    NAinfo.requireApiVersion(0, 2);

    let angle = sl(2, 89);

    let vertices = window.latbukv.iz(6);

    let rand = sl1();

    let paint1 = function(ctx) {
      ctx.lineWidth = 2;

      let angle = -Math.PI / 3.2;
      ctx.strokeStyle = om.secondaryBrandColors.iz();

      let vertex = ctx.drawLineAtAngle(10, 370, angle, 400);
      ctx.drawLine(10, 370, 390, 370);
      ctx.drawLine(390, 370, vertex.x, vertex.y);

      //Биссектрисы
      let bisector1 = ctx.drawLineAtAngle(10, 370, angle / 2, 345);
      let bisector2 = ctx.drawLineAtAngle(390, 370, Math.atan2(-370 +
  vertex.y, -390 + vertex.x) / 2 - Math.PI / 2, 317);

      //Углы
      ctx.strokeStyle = om.primaryBrandColors[rand];
      ctx.arcBetweenSegmentsCount([vertex.x, vertex.y, 10,
  370].concat([bisector1.x, bisector1.y]), 30, 2);
      ctx.arcBetweenSegmentsCount([bisector1.x, bisector1.y].concat([10,
  370, 390, 370]), 40, 2);

      ctx.strokeStyle = om.primaryBrandColors[rand];
```

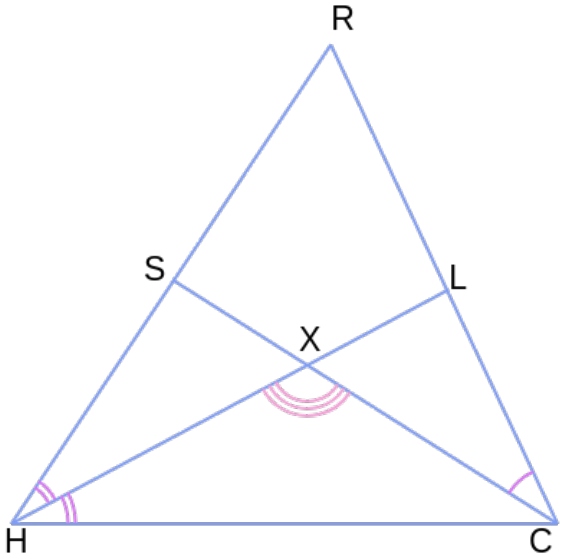
```
31      ctx.arcBetweenSegments([10, 370, 390, 370].concat([bisector2.x,
    bisector2.y]), 30);
32      ctx.arcBetweenSegments(([bisector2.x, bisector2.y].concat([390, 370,
    vertex.x, vertex.y])), 40);
33
34      ctx.strokeStyle = om.primaryBrandColors[1 - rand];
35      ctx.arcBetweenSegmentsCount([bisector1.x, 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 6: 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.

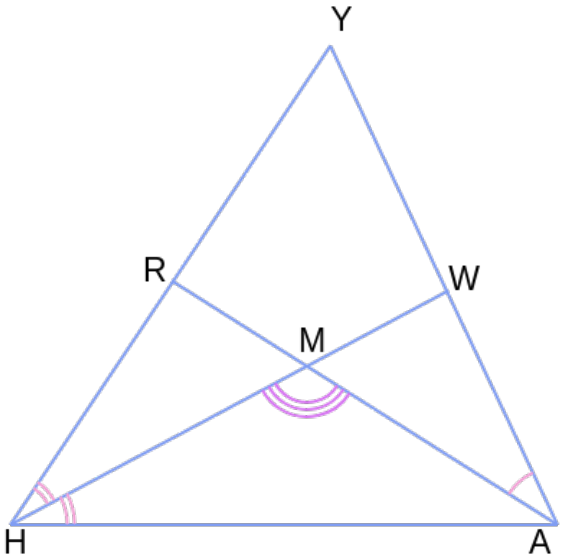
Ответ: 123,5



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.

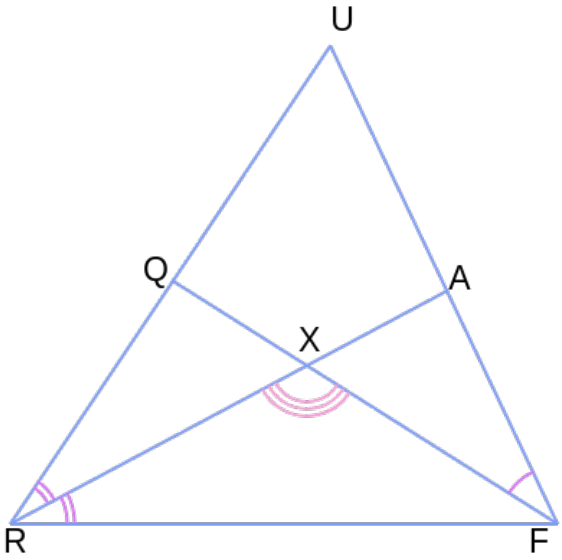
Ответ: 112,5



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»

```
(function() {
  retryWhileError(function() {

    let pyr1 = new RegularPyramid({
      height: sl(30, 70),
      baseSide: sl(20, 50),
      numberSide: 4
    });

    let pyr2 = new Pyramid({
      height: 0.5 * pyr1.height,
      baseArea: 0.5 * pyr1.baseArea,
    });

    pyr1.verticesOfFigure =
   coordinatesMiddleOfSegment3D(pyr1.verticesOfFigure[0],
   pyr1.verticesOfFigure[4]);

    let camera = {
      x: 0,
      y: 0,
      z: 0,
      scale: 5,

      rotationX: -Math.PI / 2 + Math.PI / 14,
      rotationY: 0,
      rotationZ: [1, 2].iz() * Math.PI / 3,
    };

    let point2DPyr = pyr1.verticesOfFigure.map((coord3D) =>
   project3DTo2D(coord3D, camera));

    autoScale(pyr1.verticesOfFigure, camera, point2DPyr, {
      startX: -180,
      finishX: 160,
      startY: -160,
      finishY: 160,
      maxScale: 50,
    });

    point2DPyr = pyr1.verticesOfFigure.map((coord3D) =>
   project3DTo2D(coord3D, camera));

    let letters = ['A', 'B', 'C', 'D', 'S', 'E'];
    let strok = [5, 4];

    let paint1 = function(ctx) {
      let h = 400;
      let w = 400;
      ctx.translate(h / 2, w / 2);
```

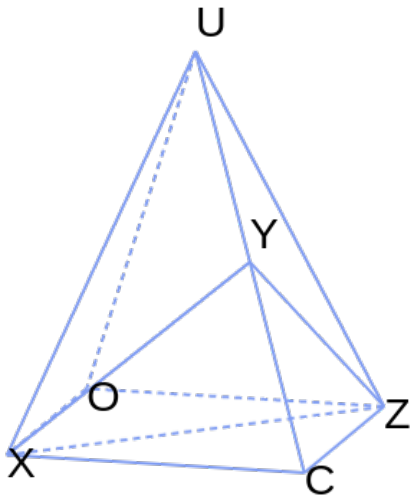
```
47        ctx.lineWidth = 2;
48        ctx.strokeStyle = om.secondaryBrandColors;
49        ctx.drawFigure(point2DPyr, [
50          [1],
51          [0, strok],
52          [1, strok, strok],
53          [1, 1, strok, 1, ],
54          [0, 1, 0, 1, 0]
55        ]);
56
57        ctx.font = "30px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

Листинг 7: 27114.js

Примеры генерируемых задач 27114.js

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

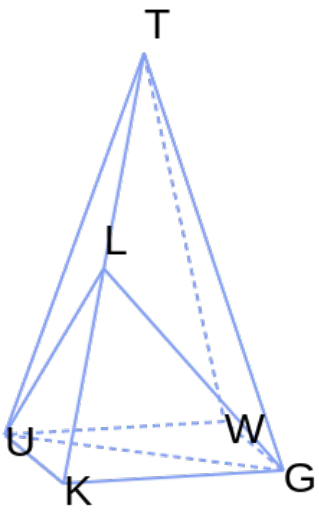
Ответ: 7791



Приложение. 16

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

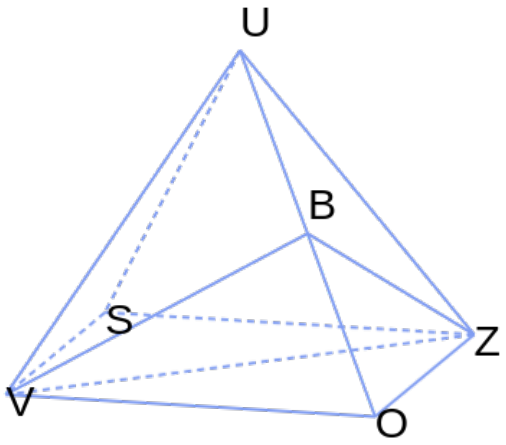
Ответ: 1200



Приложение. 17

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

Ответ: 6412,5



Приложение. 18

```
(function() {
  retryWhileError(function() {

    let pyr1 = new RegularPyramid({
      height: sl(10, 30)*(3).sqrt(),
      baseSide: sl(20, 50),
      numberSide: 3
    });

    let pyr2 = new RegularPyramid({
      height: pyr1.height,
      baseSide: 0.5 * pyr1.baseSide,
      numberSide: 3
    });


   pyr1.verticesOfFigure.push(coordinatesMiddleOfSegment3D(pyr1.verticesOfFigure
   pyr1.verticesOfFigure[1]));

   pyr1.verticesOfFigure.push(coordinatesMiddleOfSegment3D(pyr1.verticesOfFigure
   pyr1.verticesOfFigure[2]));

    let strok = [5, 4];

    let camera = {
      x: 0,
      y: 0,
      z: 0,
      scale: 5,

      rotationX: -Math.PI / 2 + Math.PI / 14,
      rotationY: 0,
      rotationZ: sl(1,2)* Math.PI / 8,
```

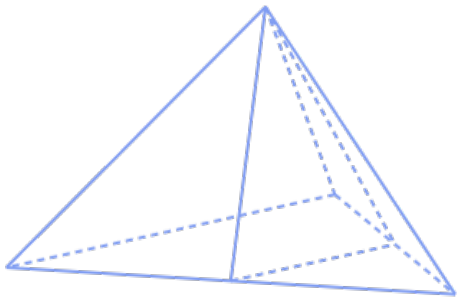
```
    };

    let point2DPyr = pyr1.verticesOfFigure.map((coord3D) =>
  project3DTo2D(coord3D, camera));

    autoScale(pyr1.verticesOfFigure, camera, point2DPyr, {
      startX: -180,
      finishX: 160,
      startY: -160,
      finishY: 160,
      maxScale: 50,
    });

    point2DPyr = pyr1.verticesOfFigure.map((coord3D) =>
  project3DTo2D(coord3D, camera));

    let paint1 = function(ctx) {
      let h = 400;
      let w = 400;
      ctx.translate(h / 2, w / 2);
      ctx.lineWidth = 2;
      ctx.strokeStyle = om.secondaryBrandColors;
      ctx.drawFigure(point2DPyr, [
        [1],
        [strok, strok],
        [1, 1, strok, 0, 1, strok],
        [0, 0, 0, 0, 0, strok]
      ]);
    };

    NAtask.setTask({
      text: ' Объём треугольной пирамиды равен $' + pyr1.volume + '$. ' +
        'Через вершину пирамиды и среднюю линию её основания проведена плоскость
  см(. рисунок). ' +
        'Найдите объём отсечённой треугольной пирамиды.',
      answers: pyr2.volume,
      author: ['Суматохина Александра'],
    });
    NAtask.modifiers.multiplyAnswerBySqrt(13);
    NAtask.modifiers.allDecimalsToStandard(true);
    NAtask.modifiers.assertSaneDecimals();
    NAtask.modifiers.addCanvasIllustration({
      width: 400,
      height: 400,
      paint: paint1,
    });
  }, 1000);
})();
//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
```

Листинг 8: 27115.js

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.

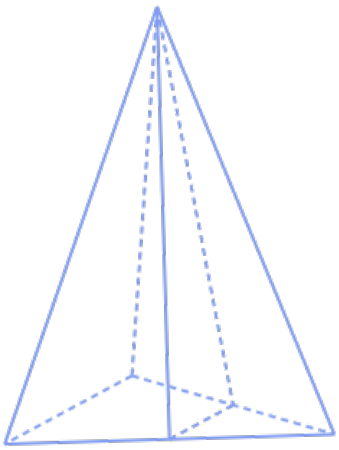
Ответ: 640



Приложение. 19

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

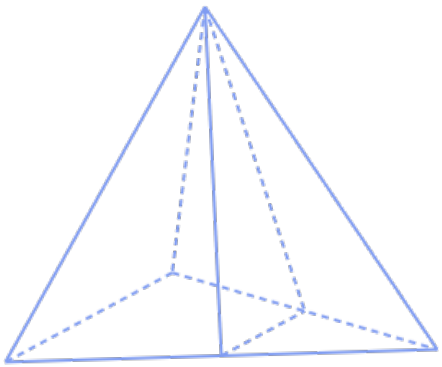
Ответ: 1237,5



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

```
(function() {
  retryWhileError(function() {
      NAinfo.requireApiVersion(0, 2);

      let stroke = [4, 5];

      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],
      };

      let par1 = new Parallelepiped({
        depth: sl(10, 20),
        height: sl(10, 20),
        width: sl(10, 20),
      });

      let par2 = new Parallelepiped({
        depth: slKrome(par1.depth, 5, par1.depth - 3),
        height: slKrome(par1.height, 5, par1.height - 3),
        width: slKrome(par1.width, 5, par1.width - 5),
      });

      let vertex3D =
   par1.verticesOfFigure.concat(par2.verticesOfFigure.map((elem) =>
   shiftCoordinate3D(elem, {
        x: -0.5 * (par1.width - par2.width),
        y: -0.5 * (par1.depth - par2.depth),
```

```
      z: -0.5 * (par1.height - par2.height),
    })));

    let camera = {
      x: 0,
      y: 0,
      z: 0,
      scale: 1,

      rotationX: -Math.PI / 2 + Math.PI / 14,
      rotationY: 0,
      rotationZ: Math.PI / sl(10, 14),
    };

    let point2D = vertex3D.map((coord3D) => project3DTo2D(coord3D,
camera));

    autoScale(vertex3D, camera, point2D, {
      startX: -180,
      finishX: 160,
      startY: -160,
      finishY: 160,
      maxScale: 100,
    });

    point2D = vertex3D.map((coord3D) => project3DTo2D(coord3D, camera));
    genAssert((point2D[3].y - point2D[11].y).abs() > 20);
    genAssert((point2D[3].y - point2D[8].y).abs() > 20);
    genAssert((point2D[13].x - point2D[14].x).abs() > 20);

    let rand = sl1();

    let paint1 = function(ctx) {
      let h = 400;
      let w = 400;
      ctx.translate(w / 2, h / 2);
      ctx.lineWidth = 2;
      ctx.strokeStyle = om.secondaryBrandColors;
      ctx.drawFigureVer2(point2D, matrixConnections);

      ctx.font = "20px liberation_sans";
      ctx.signSegmentInMiddle(point2D[4].x, point2D[4].y, point2D[7].x,
point2D[7].y, par1.height, -10, 20);
      ctx.signSegmentInMiddle(point2D[10].x, point2D[10].y,
point2D[15].x, point2D[15].y, par2.height, -20, 20);
      ctx.signSegmentInMiddle(point2D[12].x, point2D[12].y,
point2D[15].x, point2D[15].y, par2.width, 20, 20);
      ctx.signSegmentInMiddle(point2D[0].x, point2D[0].y, point2D[1].x,
point2D[1].y, par1.width, 18, 20);
      ctx.signSegmentInMiddle(point2D[1].x, point2D[1].y, point2D[2].x,
point2D[2].y, par1.depth, 18, 20);
      ctx.signSegmentInMiddle(point2D[9].x, point2D[9].y, point2D[10].x,
```

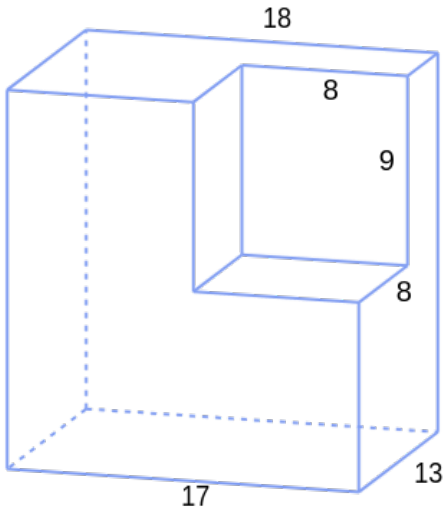
```
   point2D[10].y, par2.depth, 15, 20);
78       };
79       NAtask.setTask({
80         text: 'Найдите ' + ['площадь поверхности', 'объём'][rand] +
81           ' многогранника, изображённого на рисунке все( двугранные углы -
   прямые).',
82         answers: [par1.surfaceArea, par1.volume - par2.volume][rand],
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 9: 12.js

Примеры генерируемых задач 12.js

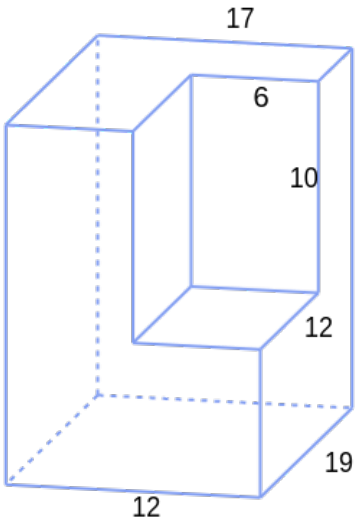
Найдите объём многогранника, изображённого на рисунке (все двугранные углы – прямые).

Ответ: 3402



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

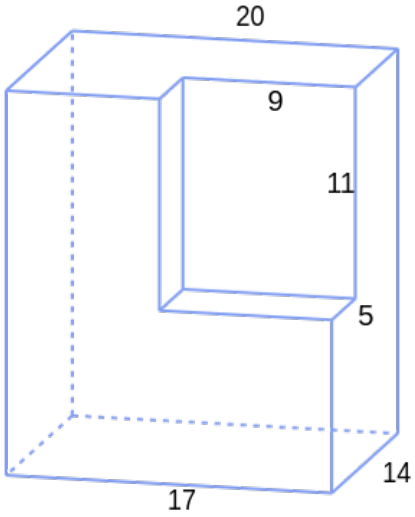
Ответ: 3156



Приложение. 23

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

Ответ: 4265



Приложение. 24

```
(function() {
  retryWhileError(function() {
      NAinfo.requireApiVersion(0, 2);

      let stroke = [4, 2];

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

```
        [7, stroke],
        [9, stroke],
        [15, stroke]
      ],
      12: [
        [11, stroke], 13, 15
      ],
      14: [9, 13, 15],
    };

    let par1 = new Parallelepiped({
      depth: sl(10, 20),
      height: sl(10, 20),
      width: sl(10, 20),
    });

    let par2 = new Parallelepiped({
      depth: par1.depth,
      height: sl(5, par1.height - 4),
      width: slKrome(par1.width, 5, par1.width - 5),
    });

    let deltaWidth = par1.width - par2.width;
    let diagonal = (0.25 * deltaWidth.pow(2) +
par2.height.pow(2)).sqrt();

    let vertex3D =
par1.verticesOfFigure.concat(par2.verticesOfFigure.map((elem) =>
shiftCoordinate3D(elem, {
      x: 0,
      y: 0,
      z: -0.5 * (par1.height - par2.height),
    })));

    let camera = {
      x: 0,
      y: 0,
      z: 0,
      scale: 1,

      rotationX: -Math.PI / 2 + Math.PI / 14,
      rotationY: 0,
      rotationZ: Math.PI / sl(12, 14),
    };

    let point2D = vertex3D.map((coord3D) => project3DTo2D(coord3D,
camera));

    autoScale(vertex3D, camera, point2D, {
      startX: -180,
      finishX: 160,
      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 +
```

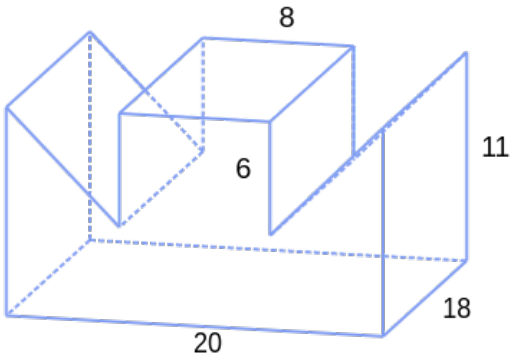
```
   par2.depth) + 2 * par2.height * par2.depth + 2 *
106          diagonal * par2.depth,
107          par1.volume - par2.height * (par1.width - par2.width) *
   par2.width
108        ][rand],
109        author: ['Суматохина Александра']
110      });
111      NAtask.modifiers.multiplyAnswerBySqrt(13);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

Листинг 10: 144526144.js

Примеры генерируемых задач 144526144.js

Найдите площадь поверхности многогранника, изображённого на рисунке.

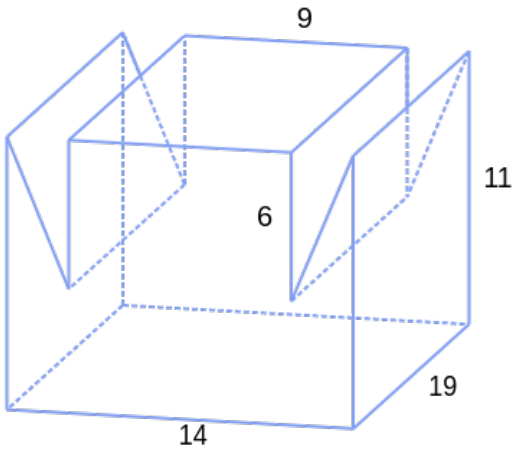
Ответ: 3384



Приложение. 25

Найдите площадь поверхности многогранника, изображённого на рисунке.

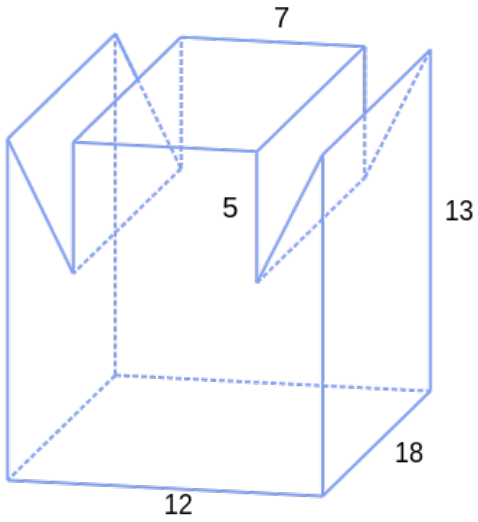
Ответ: 2656



Приложение. 26

Найдите площадь поверхности многогранника, изображённого на рисунке.

Ответ: 2633



Приложение. 27