

“Geometrica”

Система построение геометрических чертежей со
встроенным языком программирования и
возможностью удаленного программного управления

Студент: Чубий Савва Андреевич
БПИ 233

Научный руководитель: Куренков Владимир Вячеславович
старший преподаватель департамента
больших данных и информационного
поиска

Терминология Rust

Термин	Примерный Аналог
Крейт	Пакет
Трейт	Интерфейс
enum	sealed class typesafe union std::variant

Краткое описание

-

Краткое описание

- Построение и изменение геометрических чертежей
-

Краткое описание

- Построение и изменение геометрических чертежей
- Встроенный язык программирования (далее Язык)
-

Краткое описание

- Построение и изменение геометрических чертежей
- Встроенный язык программирования (далее Язык)
- Локальный сервер + 3 клиента:
 - Графический (gui)
 - Командной строки (cli)
 - Библиотека (lib) для ЯП Rust

Применимость

-

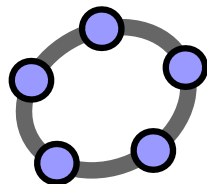
Применимость

- Целевая аудитория:
 - Школьники/ школьные учителя (gui)
 - Студенты/ преподаватели ВУЗов (cli, gui, lib)
-

Применимость

- Целевая аудитория:
 - Школьники/ школьные учителя (gui)
 - Студенты/ преподаватели ВУЗов (cli, gui, lib)
- Примеры использования:
 - Наглядная демонстрация теорем (см. “Описание языка”, гл. 4)
 - Совместное решение задач в классе
 - Проведение проверочных работ
 - Выполнение домашних заданий
 - Самостоятельное решение задач
 - Отладка программ

Аналоги

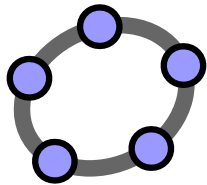


+

[https://www.geogebra.org/
geometry](https://www.geogebra.org/geometry)

от Markus Hohenwarter

-



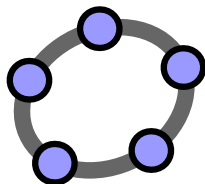
+

- Бесплатно
- Есть оффлайн версия
- Есть библиотека для сущ. ЯП
- Есть стили

[https://www.geogebra.org/
geometry](https://www.geogebra.org/geometry)

от Markus Hohenwarter

-



+

- Бесплатно
- Есть оффлайн версия
- Есть библиотека для сущ. ЯП
- Есть стили

[https://www.geogebra.org/
geometry](https://www.geogebra.org/geometry)

от Markus Hohenwarter

-

-
- Нет макросов
 - Ограниченный встроенный ЯП
 - Нет REST API
 - Нельзя работать из терминала



+

[https://www.desmos.com/
geometry](https://www.desmos.com/geometry)

-

от Desmos Studio PBC



[https://www.desmos.com/
geometry](https://www.desmos.com/geometry)

от Desmos Studio PBC

+

- Бесплатно
 - Есть библиотека для суц. ЯП
 - Есть стили
-

-



[https://www.desmos.com/
geometry](https://www.desmos.com/geometry)

от Desmos Studio PBC

- +**
 - Бесплатно
 - Есть библиотека для сущ. ЯП
 - Есть стили

- - Нет оффлайн версии
 - Нет макросов
 - Ограниченный встроенный ЯП
 - Нет REST API
 - Нельзя работать из терминала



+

[https://www.int-edu.ru/
content/rusticus-0](https://www.int-edu.ru/content/rusticus-0)

от Учреждение ДПО
“ИНТ”

-



+

- Есть оффлайн версия
 - Есть макросы
 - Есть стили
-

[https://www.int-edu.ru/
content/rusticus-0](https://www.int-edu.ru/content/rusticus-0)

от Учреждение ДПО
“ИНТ”

-



[https://www.int-edu.ru/
content/rusticus-0](https://www.int-edu.ru/content/rusticus-0)

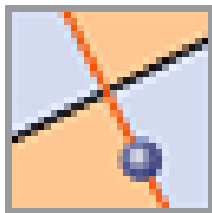
от Учреждение ДПО
“ИНТ”

+

- Есть оффлайн версия
- Есть макросы
- Есть стили

-

- Платно: домашняя — 2400 руб, базовая — 6120 руб
- Нет встроенного ЯП
- Нет библиотеки для сущ. ЯП
- Нет REST API
- Нельзя работать из терминала

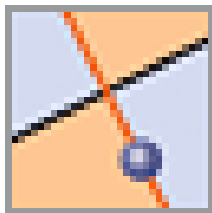


+

<https://obr.1c.ru/mathkit/>

от ООО “Виртуальная
лаборатория”

–



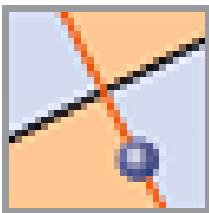
+

- Бесплатно
- Есть оффлайн версия
- Есть макросы
- Есть стили

<https://obr.1c.ru/mathkit/>

от ООО “Виртуальная
лаборатория”

–



+

- Бесплатно
- Есть оффлайн версия
- Есть макросы
- Есть стили

<https://obr.1c.ru/mathkit/>

от ООО “Виртуальная
лаборатория”

-

-
- Нет встроенного ЯП
 - Нет библиотеки для сущ. ЯП
 - Нет REST API
 - Нельзя работать из терминала

Функционал

-

- Создать новый объект

Типы: `bool`, `int`, `real`, `str`, `pt`, `line`, `circ`

Виды: свободный/ зависимый

-

- Создать новый объект

Типы: `bool`, `int`, `real`, `str`, `pt`, `line`, `circ`

Виды: свободный/ зависимый

- Изменить свободный объект и **пересчитать все зависимые**
-

- Создать новый объект

Типы: `bool`, `int`, `real`, `str`, `pt`, `line`, `circ`

Виды: свободный/ зависимый

- Изменить свободный объект и **пересчитать все зависимые**
- Удалить объект и все зависимые
-

- Создать новый объект

Типы: `bool`, `int`, `real`, `str`, `pt`, `line`, `circ`

Виды: свободный/ зависимый

- Изменить свободный объект и **пересчитать все зависимые**
- Удалить объект и все зависимые
- Получить значение объекта
-

- Создать новый объект

Типы: `bool`, `int`, `real`, `str`, `pt`, `line`, `circ`

Виды: свободный/ зависимый

- Изменить свободный объект и **пересчитать все зависимые**
- Удалить объект и все зависимые
- Получить значение объекта
- Выполнить код на Языке
-

- Создать новый объект

Типы: `bool`, `int`, `real`, `str`, `pt`, `line`, `circ`

Виды: свободный/ зависимый

- Изменить свободный объект и **пересчитать все зависимые**
- Удалить объект и все зависимые
- Получить значение объекта
- Выполнить код на Языке
- Экспортировать/ импортировать чертеж из файла
-

- Создать новый объект

Типы: `bool`, `int`, `real`, `str`, `pt`, `line`, `circ`

Виды: свободный/ зависимый

- Изменить свободный объект и **пересчитать все зависимые**
- Удалить объект и все зависимые
- Получить значение объекта
- Выполнить код на Языке
- Экспортировать/ импортировать чертеж из файла
- Экспортировать в SVG

Цель и задачи

Цель: разработать программный продукт “Geometrica”

Цель: разработать программный продукт “Geometrica”

Задачи:

- Определения функциональных требований
- Выбор стека технологий
- Написание “Технического Задания”
- Разработка архитектуры приложения
- Реализация программной системы “Geometrica”
- Тестирование программной системы “Geometrica”
- Написание итоговой документации
- Защита проекта

Описание Языка

-

Общая структура

- Типизация:
 - Сильная
 - Статическая
-

Общая структура

- Типизация:
 - Сильная
 - Статическая
- Конструкции:
 -

Общая структура

- Типизация:
 - Сильная
 - Статическая
- Конструкции:
 - Императивные:
 - объявления
 - команды
 -

```
fact n:int -> int = if
    n > 0 then n * (fact (n - 1)),
    n == 0 then 1
n = 5
t = fact n
set! n (1 + 1)
get_all!
```

Общая структура

- Типизация:
 - Сильная
 - Статическая
- Конструкции:
 - Императивные:
 - объявления
 - команды
 - Функциональные:
 - выражения

```
fact n:int -> int = if
    n > 0  then n * (fact (n - 1)),
    n == 0 then 1
n = 5
t = fact n
set! n (1 + 1)
get_all!
```


Скрипт

-

Скрипт

- Выражение (Statement)



Скрипт

- Выражение (Statement)
 - Вызов команды
 -

Скрипт

- Выражение (Statement)
 - Вызов команды
 - Объявление
 -

Конструкции

Скрипт

- Выражение (Statement)
 - Вызов команды
 - Объявление
 - Объявление функции
 -

Скрипт

- Выражение (Statement)
 - Вызов команды
 - Объявление
 - Объявление функции
 - Объявление значения

Команды

-

Команды

- Изменения:

-

clear!

rm! x

set! x (10 * 2 + 1)

Команды

- Изменения:

clear!

rm! x

set! x (10 * 2 + 1)

-

- Служебные:

list_cmd!

list_func!

Конструкции

Команды

- Изменения:

clear!

rm! x

set! x (10 * 2 + 1)

- Работа с файлами:

save! "file.geom"

load! "file.geom"

save_svg! "img.svg"

- Служебные:

list_cmd!

list_func!

-

Конструкции

Команды

- Изменения:

clear!

rm! x

set! x (10 * 2 + 1)

- Работа с файлами:

save! "file.geom"

load! "file.geom"

save_svg! "img.svg"

- Служебные:

list_cmd!

list_func!

- Вычисления:

eval! (x + 1)

get! x

get_all!

Объявления значений

-

Объявления значений

- Независимые

-

```
x:real = 42.0
```

```
y = "Hello,\nworld!"
```

```
p = 2.0 * (pt 10.0 20.0)
```

```
// ошибка: real не int
```

```
t:int = 10.0
```

```
// none
```

```
l = none line
```

Объявления значений

- Независимые

```
x:real = 42.0  
y = "Hello,\nworld!"  
p = 2.0 * (pt 10.0 20.0)
```

```
// ошибка: real не int  
t:int = 10.0
```

```
// none  
l = none line
```

- Зависимые

```
k:real = 2.0 * x  
l = (x + y) / 2.0
```

```
// ошибка: m опр. через m  
m = 2 * m
```

Конструкции

Объявления функций

```
sum x:int y:int -> int = x + y
```

```
// перегрузка
```

```
sum x:real y:real -> real = x + y
```

```
// рекурсия
```

```
fact n:int -> int = if  
    n > 0 then n * (fact (n - 1))  
    n == 0 then 1
```

```
// ошибка: x - НЕ аргумент ф-ции
```

```
add_x t:int -> int = t + x
```

Конструкции

Выражения (Expr)

- Литерал
- Переменная
- Выражение as (приведение типов)
- Условное выражение if
- Вызов функции
- Dot-нотация
- Применение бинарного оператора
- Применение унарного оператора
- Выражение let

Выражения. Выражение as

```
x = 10  
y = x as real // y = 10.0
```

Выражения. Условное выражение `if`

```
// Все ветки должны быть одного типа
cmp x:int y:int -> str = if
  x > y  then "x is greater",
  x < y  then "y is greater",
  x == y then "x and y are the same",
  else      "just how?" // else можно не писать
```

Конструкции

Выражения. Вызов функции

```
p1 = pt 100.0 100.0
```

```
l = line p1 (pt 200.0 300.0)
```

```
l_p2_y = y (p2 l)
```

Выражения. Dot-нотация

$l_{p2_x} = l.p2.x$

// то же, что и:

$l_{p2_x} = x \ (p2 \ l)$

Выражения. Унарный оператор

```
y = -x  
cond2 = !cond1
```

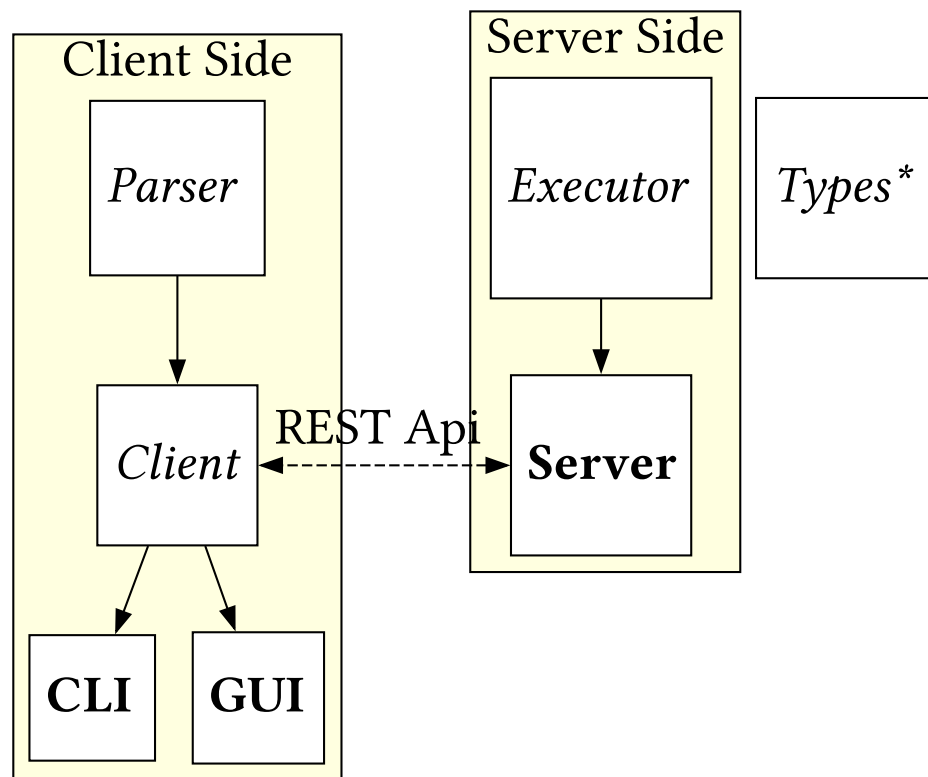
Выражения. Бинарный оператор

```
mid = (p1 + p2) / 2.0
```

Выражения. Выражение `let`

```
dist p1:pt p2:pt -> real = let
    delta = p1 - p2,
    x = delta.x,
    y = delta.y,
in
    (x^2.0 + y^2.0)^0.5
```

Реализация



Бинарные крейты (bin) выделены жирным, крейты-библиотеки (lib) выделены курсивом. Обычными стрелками показаны библиотечные зависимости, пунктирными — зависимости других типов.

- **Types** — общие объявления
- **Executor** — основные вычисления
- **Server** — сервер
- **Parser** — парсер Языка
- **Client** — клиент-библиотека
- **GUI** — графический клиент
- **CLI** — клиент командной строки

-

- Легкий, использует условную компиляцию
-

- Легкий, использует условную компиляцию
- Содержит:
 - ▶

- Легкий, использует условную компиляцию
- Содержит:
 - Типы Языка (`Value`, `FunctionSignature`, ...)
 -

- Легкий, использует условную компиляцию
- Содержит:
 - Типы Языка (`Value`, `FunctionSignature`, ...)
 - Конструкции Языка (`Expr`, ...)
 -

- Легкий, использует условную компиляцию
- Содержит:
 - Типы Языка (`Value`, `FunctionSignature`, ...)
 - Конструкции Языка (`Expr`, ...)
 - `api (api :: json :: dump :: {Request, Response, ROUTE}, ...)`

Про арі

-
-
-
-

Про арі

- Только POST
-
-
-

Про арі

- Только POST
- Формат — JSON
-
-

Про арі

- Только POST
- Формат — JSON
- Есть обработка ошибок
-

Про api

- Только POST
- Формат — JSON
- Есть обработка ошибок
- Пример объявления:

```
pub mod items {  
  pub mod get {  
    route! {  
      ROUTE "/items/get"  
      REQUEST {  
        name: Ident  
      }  
      RESPONSE {  
        value: Value  
      }  
    }  
  }  
}
```

-

- Поддерживает состояние чертежа в виде древовидной структуры
-

- Поддерживает состояние чертежа в виде древовидной структуры
- Компилирует Expr в CExpr (от Compiled Expr)
-

- Поддерживает состояние чертежа в виде древовидной структуры
- Компилирует Expr в CExpr (от Compiled Expr)
- Выполняет все вычисления (пересчёт значений)

-

- Фасад над **Executor**
-

- Фасад над **Executor**
- Реализует api из **Types**

-

- Главная структура — Client
-

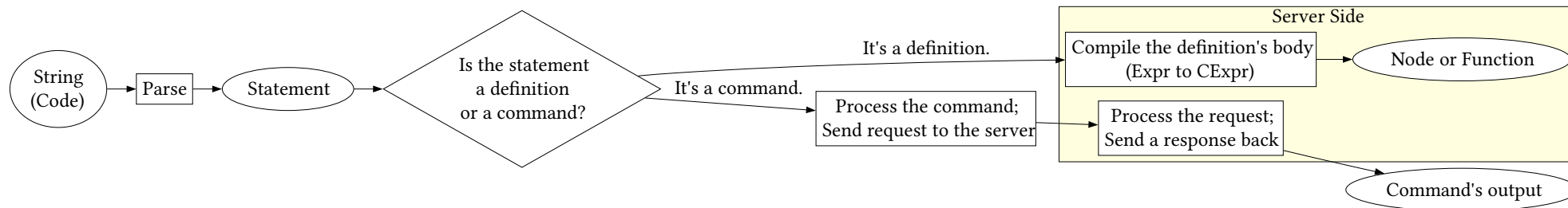
- Главная структура — Client
- При создании Client может запустить сервер
-

- Главная структура — `Client`
- При создании `Client` может запустить сервер
- Всё делается через методы `Client`:
 - `Client::eval`
 - `Client::get_all_items`
 - `Client::command`
 - ...
-

- Главная структура — `Client`
- При создании `Client` может запустить сервер
- Всё делается через методы `Client`:
 - `Client::eval`
 - `Client::get_all_items`
 - `Client::command`
 - ...
- Некоторые методы просто посылают запрос, другие имеют более сложную логику
-

- Главная структура — `Client`
- При создании `Client` может запустить сервер
- Всё делается через методы `Client`:
 - `Client::eval`
 - `Client::get_all_items`
 - `Client::command`
 - ...
- Некоторые методы просто посылают запрос, другие имеют более сложную логику
- Используется в **GUI** и **CLI**

Процесс исполнение кода на Языке



В кругах обозначены состояния, в прямоугольниках — действия, в ромбах — условия. Действия в желтом прямоугольнике происходят на стороне сервера, остальные — на стороне клиента.

Режимы работы:

-
-
-

Режимы работы:

- **Скриптовый**

Запуск: `cli script.geom`

-
-

Режимы работы:

- **Скриптовый**
- **Стандартного ввода**

Запуск: `cat script.geom | cli`

-

Режимы работы:

- **Скриптовый**
- **Стандартного ввода**
- **Интерактивный**

Запуск: `cli`

Пример сессии:

```
Welcome to Geometrica Cli!
```

```
Enter list_cmd! to see all available commands.
```

```
> x = 1.0
```

```
> y = 2.0
```

```
> z = (x + y) / 2.0
```

```
> get! z
```

Name	Value
z	1.500

```
> set! x 4.0
```

```
> get! z
```

Name	Value
z	3.000

File Transformation Server

Variables

new_var = 1.0

A pt -0.747 0.359

B pt 0.169 -0.806

C pt 0.692 0.058

alt1 line (pt 0.692 0.058) (pt -0.051

alt2 line (pt -0.747 0.359) (pt 0.440

circ1 circ (pt -0.027 0.209) 0.735

dist1 0.735

interpoint1 pt 0.283 -0.264

line1 line (pt -0.747 0.359) (pt 0.169

line2 line (pt 0.169 -0.806) (pt 0.692 0.058)

line3 line (pt 0.692 0.058) (pt -0.747 0.359)

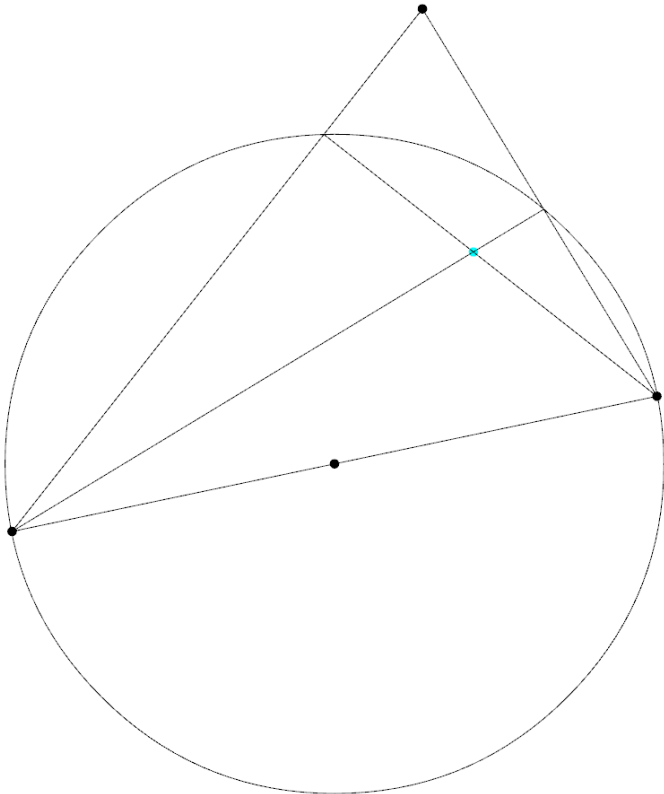
mid1 pt -0.027 0.209

Command Line

ME: list_cmd!

SERVER:	Name	Sign	Description
	clear!	-	clear all items and t
	eval!	expr+	evaluate some expr
	get!	ident+	get item's value by
	get_all!	-	get all items' values
	list_cmd!	-	list all commands
	list_func!	-	list all functions
	rm!	ident+	remove some items
	set!	ident expr	set item's value to e
	save!	expr	writes the state to a
			expr should evaluat
	load!	expr	reads state from file
			expr should evaluat
			writes svg image to

list_cmd! >



Mode Selector

Transform

Modify

Create Point

Delete

Builtins (B)

Function Name

a line (B)

b line (B)

c line (B)

circ pt real (B)

cross pt pt (B)

dot pt pt (B)

line pt pt (B)

o circ (B)

p1 line (B)

Arguments

Function: line

pt

pt

Строк кода:	~8800
Крейтов:	7
Модулей:	~75
Структур и перечислений:	~110
Функций и методов:	~550
Покрытие тестами (parser, executor, client):	~60%

-

- **Общие:**



- **Общие:**
 - Rust
 -

- **Общие:**

- Rust
- Cargo
-

- **Общие:**

- Rust
- Cargo
- Nix

-

- **Общие:**
 - Rust
 - Cargo
 - Nix
- **Документация:**
 -

- **Общие:**
 - Rust
 - Cargo
 - Nix
- **Документация:**
 - Typst
 -

- **Общие:**
 - Rust
 - Cargo
 - Nix
- **Документация:**
 - Typst
 - GraphViz
-

- **Общие:**
 - Rust
 - Cargo
 - Nix
- **Документация:**
 - Typst
 - GraphViz
- **Types:**
 -

- **Общие:**
 - Rust
 - Cargo
 - Nix
- **Документация:**
 - Typst
 - GraphViz
- **Types:**
 - serde

- **Общие:**
 - Rust
 - Cargo
 - Nix
- **Документация:**
 - Typst
 - GraphViz
- **Types:**
 - serde
- **Parser:**
 -

- **Общие:**
 - Rust
 - Cargo
 - Nix
- **Документация:**
 - Typst
 - GraphViz
- **Types:**
 - serde
- **Parser:**
 - peg
 -

- **Общие:**
 - Rust
 - Cargo
 - Nix
- **Документация:**
 - Typst
 - GraphViz
- **Types:**
 - serde
- **Parser:**
 - peg
- **Server:**
 -

- **Общие:**
 - Rust
 - Cargo
 - Nix
- **Документация:**
 - Typst
 - GraphViz
- **Types:**
 - serde
- **Parser:**
 - peg
- **Server:**
 - tokio
 -

- **Общие:**
 - Rust
 - Cargo
 - Nix
- **Документация:**
 - Typst
 - GraphViz
- **Types:**
 - serde
- **Parser:**
 - peg
- **Server:**
 - tokio
 - axum
-

- **Общие:**
 - Rust
 - Cargo
 - Nix
- **Документация:**
 - Typst
 - GraphViz
- **Types:**
 - serde
- **Parser:**
 - peg
- **Server:**
 - tokio
 - axum
- **Client:**
 -

- **Общие:**
 - Rust
 - Cargo
 - Nix
- **Документация:**
 - Typst
 - GraphViz
- **Types:**
 - serde
- **Parser:**
 - peg
- **Server:**
 - tokio
 - axum
- **Client:**
 - tokio
 -

- **Общие:**
 - Rust
 - Cargo
 - Nix
- **Документация:**
 - Typst
 - GraphViz
- **Types:**
 - serde
- **Parser:**
 - peg
- **Server:**
 - tokio
 - axum
- **Client:**
 - tokio
 - reqwest
-

- **Общие:**
 - Rust
 - Cargo
 - Nix
- **Документация:**
 - Typst
 - GraphViz
- **Types:**
 - serde
- **Parser:**
 - peg
- **Server:**
 - tokio
 - axum
- **Client:**
 - tokio
 - reqwest
- **GUI:**
 -

- **Общие:**
 - Rust
 - Cargo
 - Nix
- **Документация:**
 - Typst
 - GraphViz
- **Types:**
 - serde
- **Parser:**
 - peg
- **Server:**
 - tokio
 - axum
- **Client:**
 - tokio
 - reqwest
- **GUI:**
 - iced

Итог

-

- Цель достигнута
-

- Цель достигнута
- Все поставленные задачи выполнены

	Geometrica	GeoGebra	Desmos	Жив. Mat.	MathKit	
Бесплатно	+	+	+	-	+	●
Оффлайн версия	+	+	-	+	+	●
Макросы	?	-	-	+	+	●
Библиотека для сущ. ЯП	+	+	+	-	-	●
Встроенный ЯП	+	?	?	-	-	●
REST API	+	-	-	-	-	●
Работа из терминала	+	-	-	-	-	●
Стили	-	+	+	+	+	●

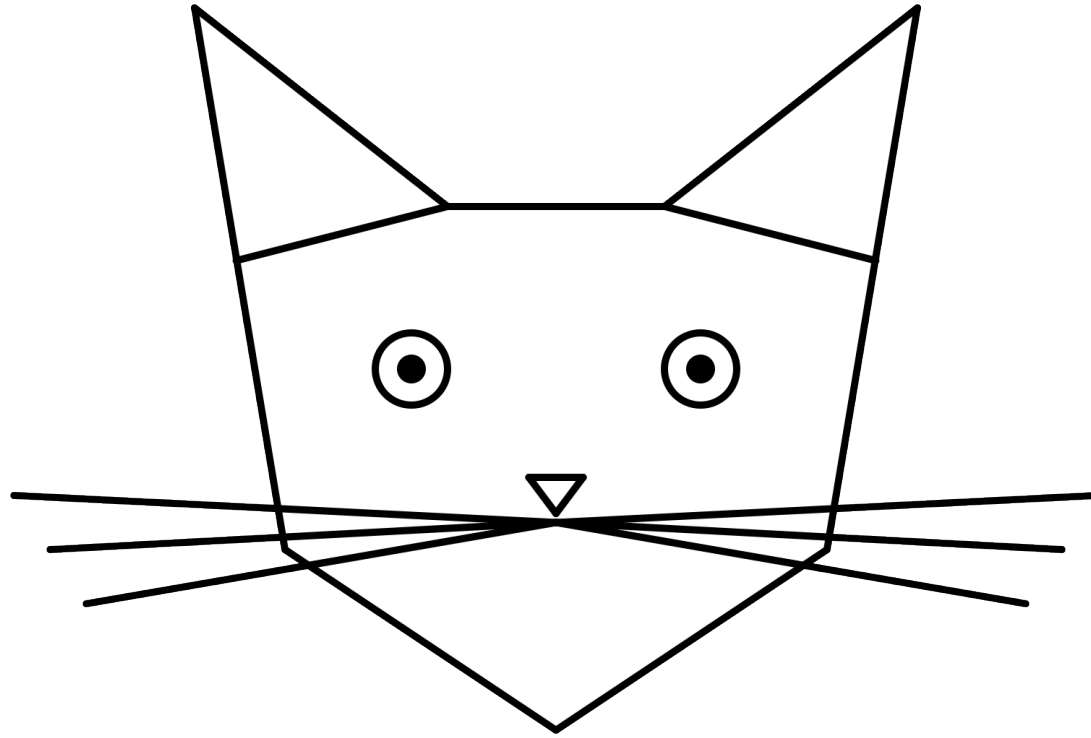
Полные названия аналогов приведены в секции “Аналоги”. “+” — функция имеется, “-” — функция отсутствует, “?” — функция частично присутствует/ имеются значительные ограничения. Зеленый — Geometrica превосходит большинство аналогов, желтый — Geometrica превосходит многие аналоги, красный — Geometrica проигрывает аналогам.

-

- Стили
-

- Стили
- Больше фигур
-

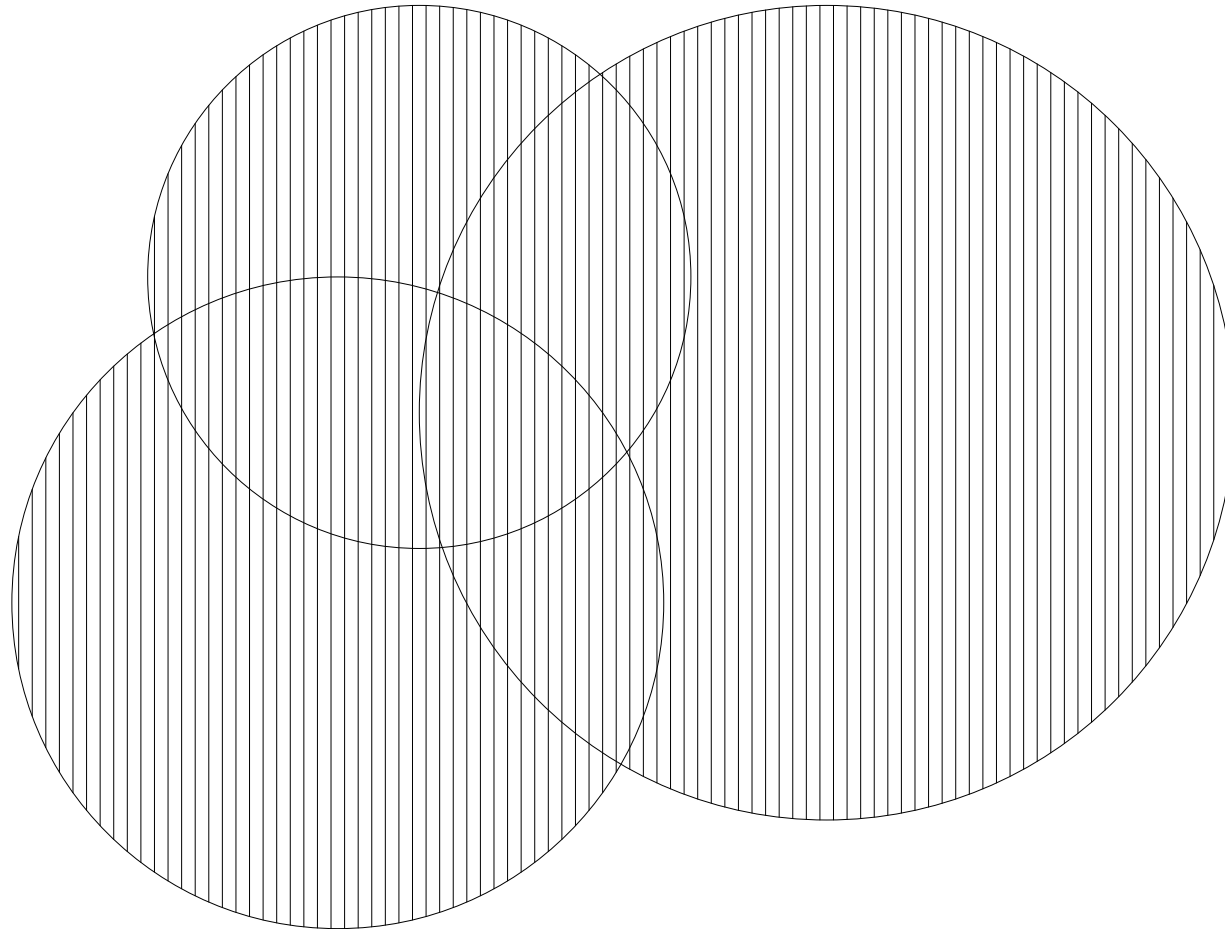
- Стили
- Больше фигур
- Больше платформ



Котик нарисован при помощи Geometrica

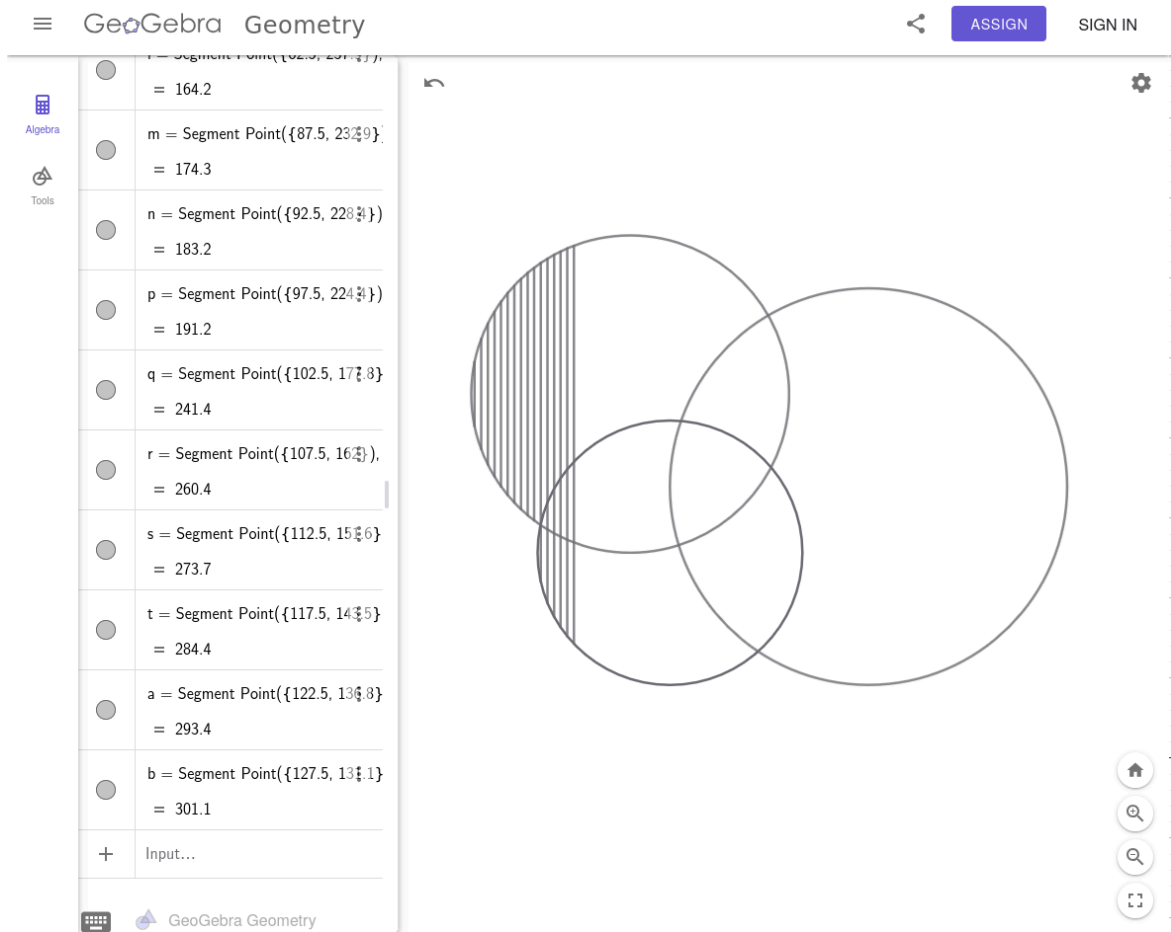
Приложение

Постановка задачи



Практический пример

Отладка с GeoGebra

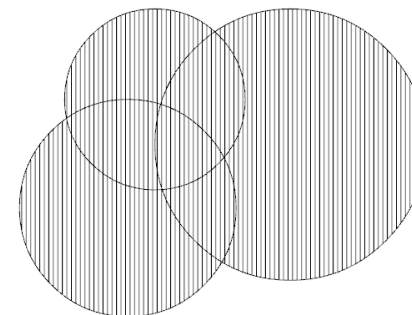


Отладка с Geometrica

Варианты решения:

-
-
-

```
use client::{  
    types::core::{Circ, Line, Pt},  
    Client,  
};  
  
#[tokio::main]  
async fn main() -> anyhow::Result<()> {  
    let client = Client::new().await?;  
    client.clear().await?;  
  
    let cs = [  
        Circ::new(Pt::new(200.0, 200.0), 100.0),  
        Circ::new(Pt::new(350.0, 250.0), 150.0),  
        Circ::new(Pt::new(170.0, 320.0), 120.0),  
    ];  
  
    let min_x = 0f64;  
    let max_x = 500f64;  
    let n = 100usize;  
    let h = (max_x - min_x) / n as f64;  
  
    let mut ls = vec![];  
    for i in 0..n {  
        let a = min_x + i as f64 * h;  
        let b = a + h;  
        let x = (b + a) / 2.0;
```



Отладка с Geometrica

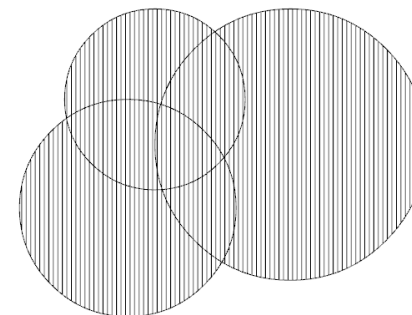
Варианты решения:

- Через lib-клиент

-

-

```
use client::{  
    types::core::{Circ, Line, Pt},  
    Client,  
};  
  
#[tokio::main]  
async fn main() -> anyhow::Result<()> {  
    let client = Client::new().await?  
    client.clear().await?  
  
    let cs = [  
        Circ::new(Pt::new(200.0, 200.0), 100.0),  
        Circ::new(Pt::new(350.0, 250.0), 150.0),  
        Circ::new(Pt::new(170.0, 320.0), 120.0),  
    ];  
  
    let min_x = 0f64;  
    let max_x = 500f64;  
    let n = 100usize;  
    let h = (max_x - min_x) / n as f64;  
  
    let mut ls = vec![];  
    for i in 0..n {  
        let a = min_x + i as f64 * h;  
        let b = a + h;  
        let x = (b + a) / 2.0;  
    }
```



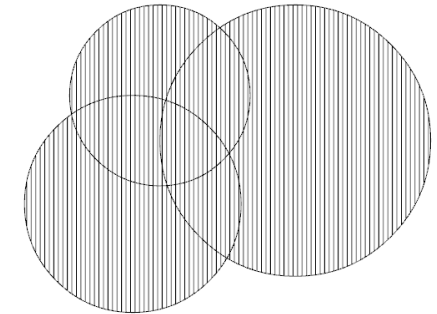
Практический пример

Отладка с Geometrica

Варианты решения:

- Через lib-клиент
- Через api
-

```
use client::{  
    types::core::{Circ, Line, Pt},  
    Client,  
};  
  
#[tokio::main]  
async fn main() -> anyhow::Result<()> {  
    let client = Client::new().await?  
    client.clear().await?  
  
    let cs = [  
        Circ::new(Pt::new(200.0, 200.0), 100.0),  
        Circ::new(Pt::new(350.0, 250.0), 150.0),  
        Circ::new(Pt::new(170.0, 320.0), 120.0),  
    ];  
  
    let min_x = 0f64;  
    let max_x = 500f64;  
    let n = 100usize;  
    let h = (max_x - min_x) / n as f64;  
  
    let mut ls = vec![];  
    for i in 0..n {  
        let a = min_x + i as f64 * h;  
        let b = a + h;  
        let x = (b + a) / 2.0;
```



Практический пример

Отладка с Geometrica

Варианты решения:

- Через lib-клиент
- Через api
- Через стандартный вывод,
Язык и cli-клиент

```
use client::{  
    types::core::{Circ, Line, Pt},  
    Client,  
};  
  
#[tokio::main]  
async fn main() -> anyhow::Result<()> {  
    let client = Client::new().await?;  
    client.clear().await?;  
  
    let cs = [  
        Circ::new(Pt::new(200.0, 200.0), 100.0),  
        Circ::new(Pt::new(350.0, 250.0), 150.0),  
        Circ::new(Pt::new(170.0, 320.0), 120.0),  
    ];  
  
    let min_x = 0f64;  
    let max_x = 500f64;  
    let n = 100usize;  
    let h = (max_x - min_x) / n as f64;  
  
    let mut ls = vec![];  
    for i in 0..n {  
        let a = min_x + i as f64 * h;  
        let b = a + h;  
        let x = (b + a) / 2.0;  
    }
```

