“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:
  - Командной строки (cli)
  - Графический (gui)
  - 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:
  -

Применимость

- Целевая аудитория:
  - Школьники/ школьные учителя (`gui`)
  - Студенты/ преподаватели ВУЗов (`cli`, `gui`, `lib`)
- Примеры использования:
  - Наглядная демонстрация теорем (см. “Описание языка”, гл. 4)
  -

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
  -

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
  -

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
  -

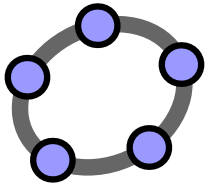
Применимость

- 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
  -

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



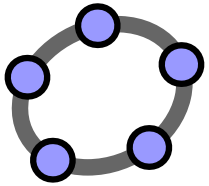
+

https://www.geogebra.org/geometry

от Markus Hohenwarter

-

GeoGebra



https://www.geogebra.org/geometry

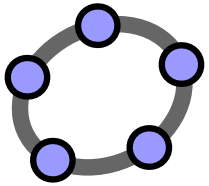
от Markus Hohenwarter

\+

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

\-

GeoGebra



https://www.geogebra.org/geometry

от Markus Hohenwarter

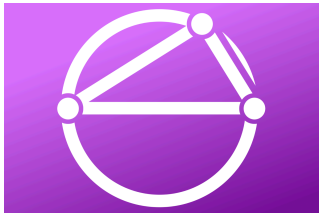
\+

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

\-

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

Desmos



https://www.desmos.com/geometry

от Desmos Studio PBC

+

-

Desmos

https://www.desmos.com/geometry

от Desmos Studio PBC

\+

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

\-

Desmos

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

Живая Математика

https://www.int-edu.ru/content/rusticus-0

от Учреждение ДПО “ИНТ”

\+

\-

Живая Математика

https://www.int-edu.ru/content/rusticus-0

от Учреждение ДПО “ИНТ”

\+

- Есть оффлайн версия
- Есть макросы
- Есть стили

\-

Живая Математика

https://www.int-edu.ru/content/rusticus-0

от Учреждение ДПО “ИНТ”

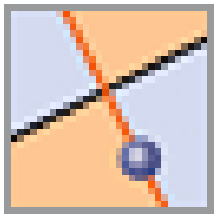
\+

- 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

Математический Конструктор (MathKit)



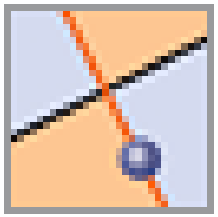
https://obr.1c.ru/mathkit/

от ООО “Виртуальная лаборатория”

\+

\-

Математический Конструктор (MathKit)



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

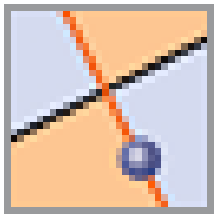
от ООО “Виртуальная лаборатория”

\+

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

\-

Математический Конструктор (MathKit)



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

от ООО “Виртуальная лаборатория”

\+

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

\-

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

Функционал

Функционал

-

Функционал

- Создать новый объект

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

  Виды: свободный/ зависимый

-

Функционал

- Создать новый объект

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

  Виды: свободный/ зависимый

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

-

Функционал

- 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
-

Функционал

- 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
-

Функционал

- 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
-

Функционал

- 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
-

Функционал

- 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

-

```
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
-

```
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:
  -

```
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
  -

```
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!
```

Конструкции

Скрипт

-

Конструкции

Скрипт

- Выражение (Statement)
  -

Конструкции

Скрипт

- Выражение (Statement)
  - Вызов команды
  -

Конструкции

Скрипт

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

Конструкции

Скрипт

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

Конструкции

Скрипт

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

Конструкции

Команды

-

Конструкции

Команды

- Изменения:

```
clear!
rm! x y z
set! x (10 * 2 + 1)
```

Конструкции

Команды

- Изменения:

```
clear!
rm! x y z
set! x (10 * 2 + 1)
```

-

- Служебные:

```
list_cmd!
list_func!
```

Конструкции

Команды

- Изменения:

```
clear!
rm! x y z
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 y z
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 y z
get_all!
```

Конструкции

Объявления значений

-

Конструкции

Объявления значений

- Независмые

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

// ошибка: real не int
t:int = 10.0

// none
x = none line
```

Конструкции

Объявления значений

- Независмые

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

// ошибка: real не int
t:int = 10.0

// none
x = 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)

-

Конструкции

Выражения (Expr)

- Литерал
-

Конструкции

Выражения (Expr)

- Литерал
- Переменная
-

Конструкции

Выражения (Expr)

- Литерал
- Переменная
- Приведение типов `as`
-

Конструкции

Выражения (Expr)

- Литерал
- Переменная
- Приведение типов `as`
- Условное выражение `if`
-

Конструкции

Выражения (Expr)

- Литерал
- Переменная
- Приведение типов `as`
- Условное выражение `if`
- Вызов функции
-

Конструкции

Выражения (Expr)

- Литерал
- Переменная
- Приведение типов `as`
- Условное выражение `if`
- Вызов функции
- `Dot`-нотация
-

Конструкции

Выражения (Expr)

- Литерал
- Переменная
- Приведение типов `as`
- Условное выражение `if`
- Вызов функции
- `Dot`-нотация
- Применение бинарного оператора
-

Конструкции

Выражения (Expr)

- Литерал
- Переменная
- Приведение типов `as`
- Условное выражение `if`
- Вызов функции
- `Dot`-нотация
- Применение бинарного оператора
- Применение унарного оператора
-

Конструкции

Выражения (Expr)

- Литерал
- Переменная
- Приведение типов `as`
- Условное выражение `if`
- Вызов функции
- `Dot`-нотация
- Применение бинарного оператора
- Применение унарного оператора
- Выражение `let`

Конструкции

Выражения. Приведение типов as

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

Конструкции

Выражения. Условное выражение `if`

```
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 200.0 200.0)
l_p2_y = y (p2 l)
```

Конструкции

Выражения. Dot-нотация

```
l_p2_x = l.p2.x
```

Конструкции

Выражения. Унарный оператор

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

Конструкции

Выражения. Бинарный оператор

```
mid = (p1 + p2) / 2
```

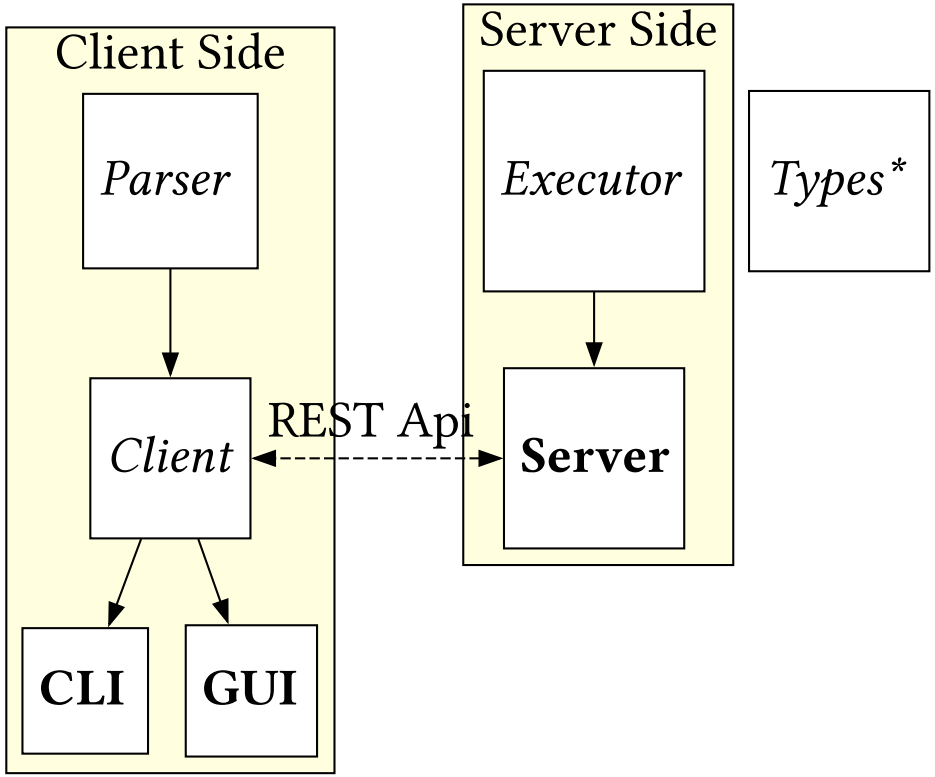
Конструкции

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

Взаимодействие между крейтами



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

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

Крейт Types

-

Крейт Types

- Содержит:
  -

Крейт Types

- Содержит:
  - Типы Языка (`Value`, `FunctionSignature`, ...)
  -

Крейт Types

- Содержит:
  - Типы Языка (`Value`, `FunctionSignature`, ...)
  - Конструкции Языка (`Expr`, ...)
  -

Крейт Types

- Содержит:
  - Типы Языка (`Value`, `FunctionSignature`, ...)
  - Конструкции Языка (`Expr`, ...)
  - api (`api::json::dump::{Request, Respone, ROUTE}`, ...)
-

Крейт Types

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

Крейт Types

Про api

-
-
-

Крейт Types

Про api

- Только `POST`
-
-

Крейт Types

Про api

- Только `POST`
- Есть обработка ошибок
-

Крейт Types

Про api

- Только `POST`
- Есть обработка ошибок
- Пример объявления:

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

Крейт Executor

-

Крейт Executor

- Поддерживает состояние чертежа через `Node`
-

Крейт Executor

- Поддерживает состояние чертежа через `Node`
- Компилирует `Expr` в `CExpr`
-

Крейт Executor

- Поддерживает состояние чертежа через `Node`
- Компилирует `Expr` в `CExpr`
- Выполняет все вычисления

Крейт Server

-

Крейт Server

- Фасад над **Executor**
-

Крейт Server

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

Крейт Client

-

Крейт Client

- Используется в **GUI** и **CLI**
-

Крейт Client

- Используется в **GUI** и **CLI**
- Главная структура — `Client`
-

Крейт Client

- Используется в **GUI** и **CLI**
- Главная структура — `Client`
- При создании `Client` может запустить сервер
-

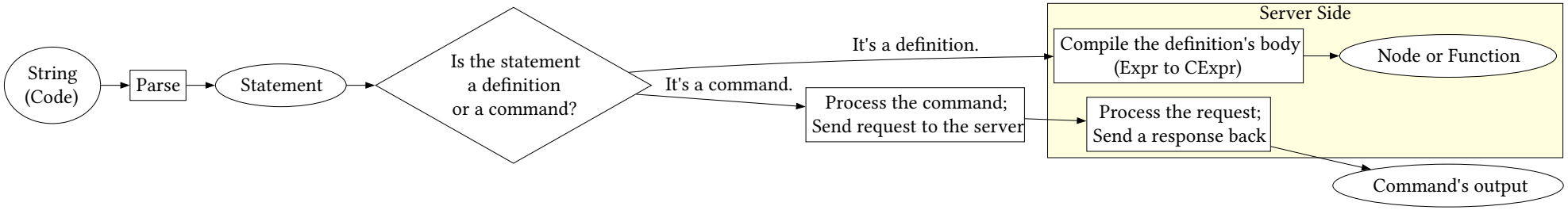
Крейт Client

- Используется в **GUI** и **CLI**
- Главная структура — `Client`
- При создании `Client` может запустить сервер
- Всё делается через методы `Client`:
  - `Client::eval`
  - `Client::get_all_items`
  - `Cilent::command`
  - ...
-

Крейт Client

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

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.

Крейт CLI

Режимы работы:

-
-
-

Крейт CLI

Режимы работы:

- **Скриптовый**

  Запуск: `cli script.geom`

-

-

Крейт CLI

Режимы работы:

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

  Запуск: `cat script.geom | cli`

-

Крейт 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

Статистика

Строк кода:	~6500
Крейтов:	7
Модулей:	~60
Структур и перечислений:	~130
Функций и методов:	~500
Покрытие тестами (`parser`, `executor`, `client`):	~60%

Стек технологий

-

Стек технологий

- **Общие**:
  -

Стек технологий

- **Общие**:
  - Rust
  -

Стек технологий

- **Общие**:
  - Rust
  - Cargo
  -

Стек технологий

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

Стек технологий

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

Стек технологий

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

Стек технологий

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

Стек технологий

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

Стек технологий

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

Стек технологий

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

Стек технологий

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

Стек технологий

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

Стек технологий

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

Стек технологий

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

Стек технологий

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

Стек технологий

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

Стек технологий

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

Стек технологий

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

Стек технологий

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

Итог

Выводы

-

Выводы

- Цель достигнута
-

Выводы

- Цель достигнута
- Все поставленные задачи выполнены

Сравнение с аналогами

	Geometrica	GeoGebra	Desmos	Жив. Мат.	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.

Направления дальнейшей работы

-

Направления дальнейшей работы

- Стили
-

Направления дальнейшей работы

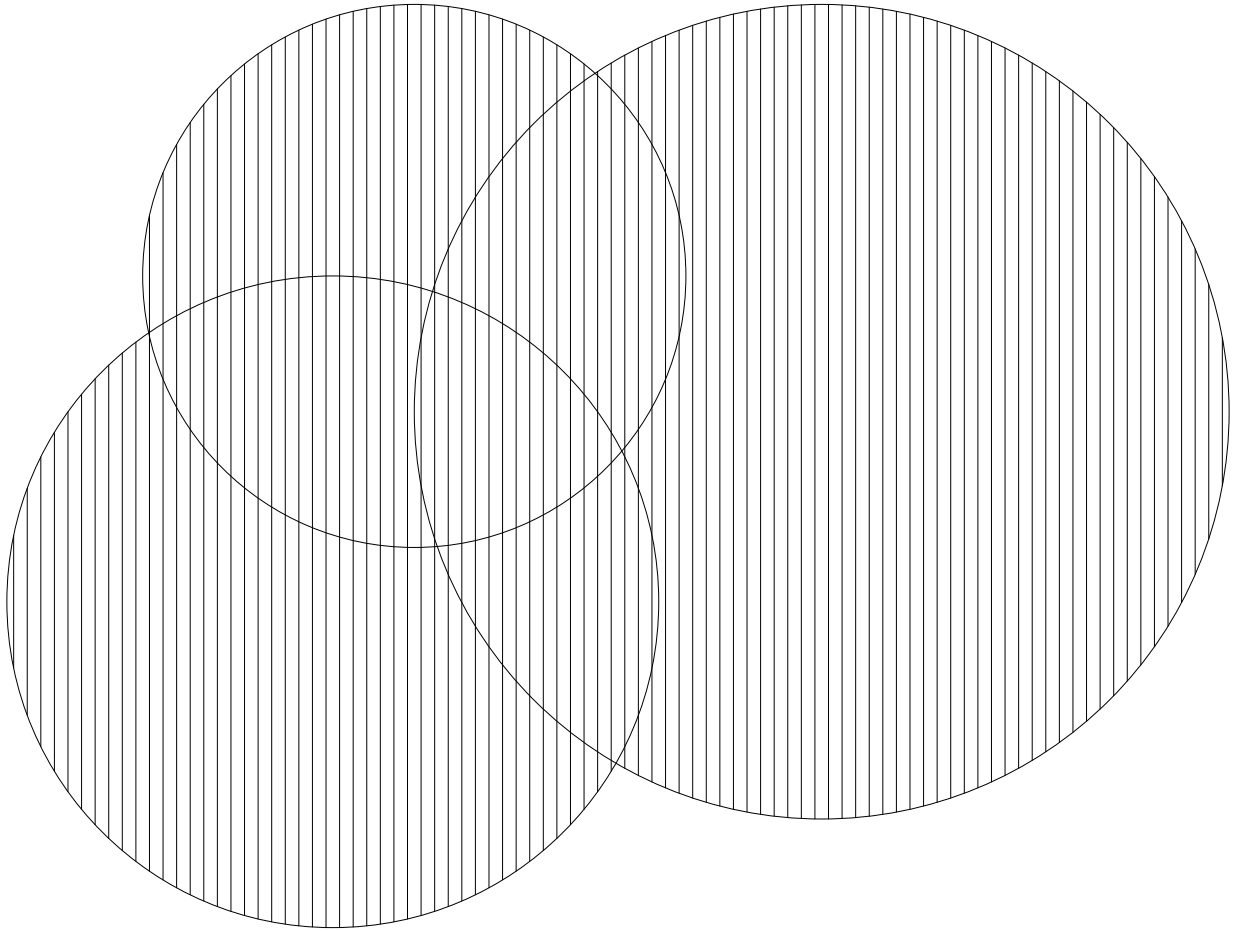
- Стили
- Больше фигур
-

Направления дальнейшей работы

- Стили
- Больше фигур
- Больше платформ

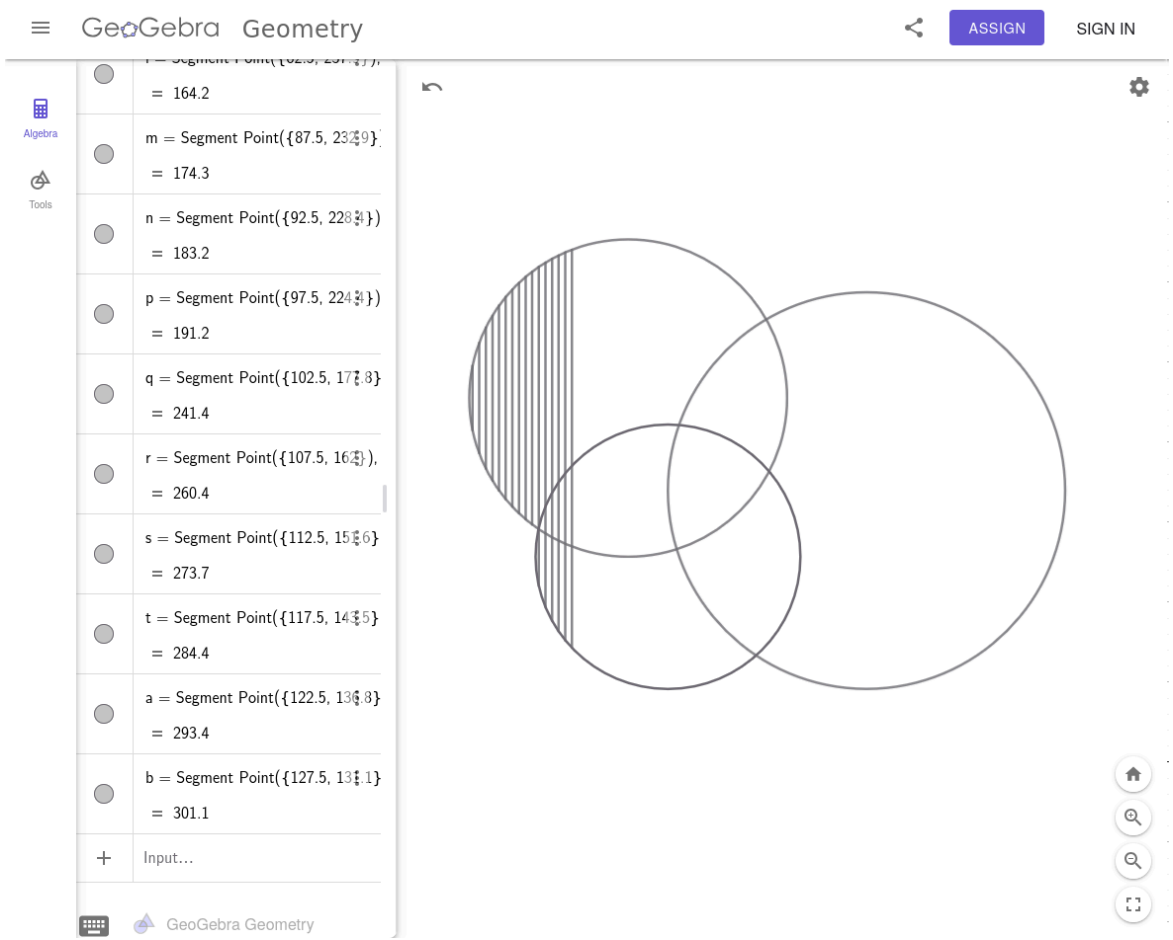
Практический пример

Постановка задачи



Практический пример

Решение с GeoGebra



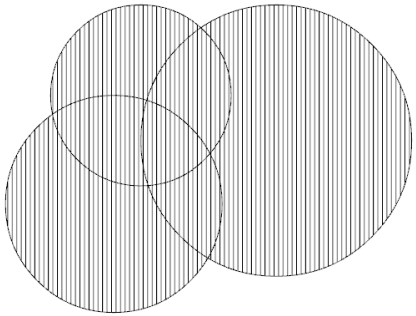
Практический пример

Решение с Geometrica

Варианты решения:

-
-
-

```rust
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;
```



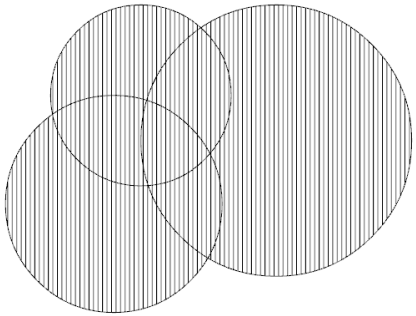
Практический пример

Решение с Geometrica

Варианты решения:

- Через lib-клиент
-
-

```rust
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;
```



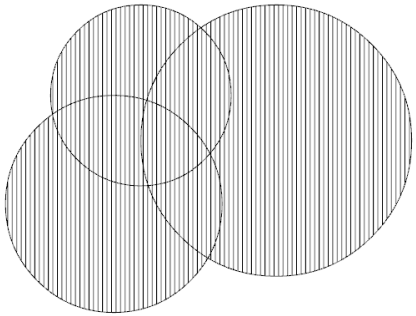
Практический пример

Решение с Geometrica

Варианты решения:

- Через lib-клиент
- Через api
-

```rust
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;
```



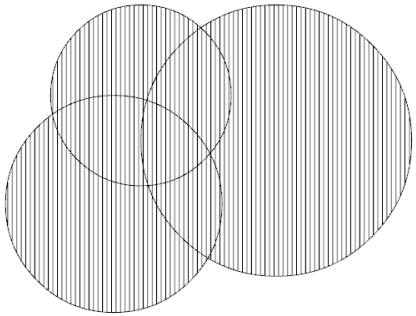
Практический пример

Решение с Geometrica

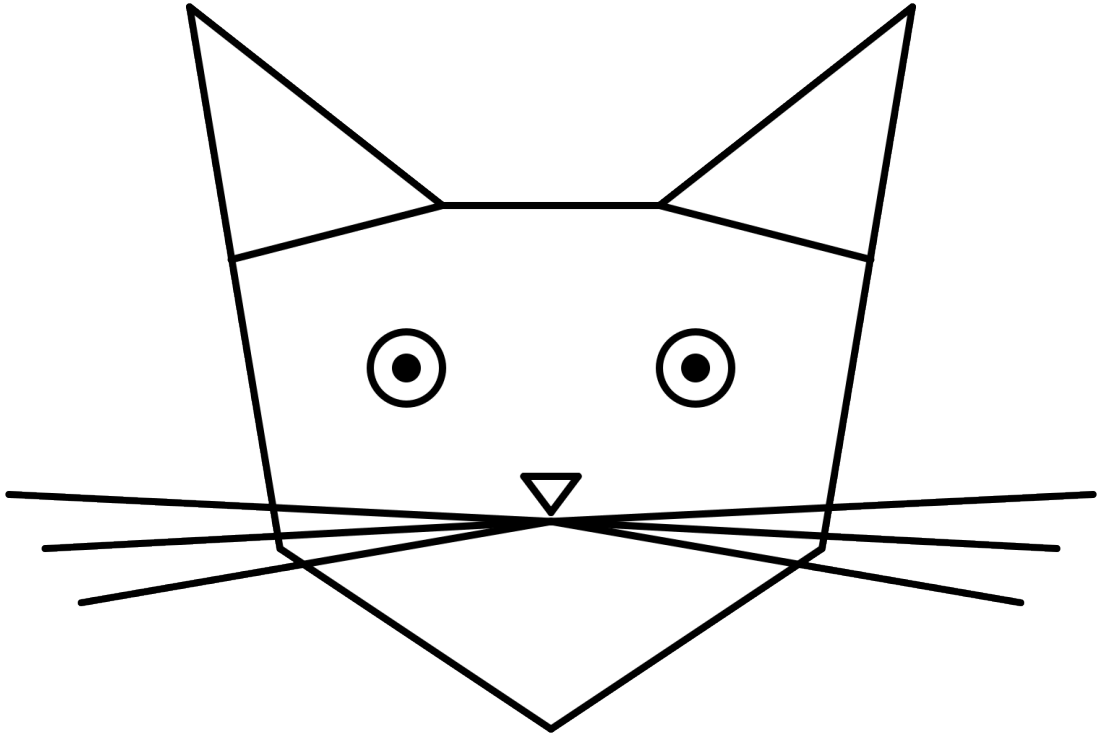
Варианты решения:

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

```rust
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;
```



Конец



Котик нарисован при помощи Geometrica