Calcus

Функциональный язык программирования

Семантика

Язык фокусируется на лямбда-исчислении, поэтому главной единицей является лямбда-выражение.

На данной стадии в языке существуют только лямбда-выражения, целые числа, дробные числа и булевы значения. Используется утиная типизация, что пока что не должно создавать проблем, так как в языке нет никаких комплексных типов, методов, и он не компилируем.

Синтаксис

Функции

Определение

```
func_name arg1 arg2 arg3... = <expr> where [
    local_func1 arg1 arg2 arg3... = <expr>;
    local_func2 arg1 arg2 arg3... = <expr>;
    local_func3 arg1 arg2 arg3... = <expr>;
];
```

where блок определяет локальные ленивые значения.

Точки с запятой используются для разделения функций, в том числе локальных функций.

Вызов

```
func_name arg1 arg2 arg3...
```

Вызов функции имеет больший приоритет, чем бинарные операторы, что будет далее описано в разделе “Операторы”.

Если функция возвращает лямбда-выражение, то после конца вызова основных аргументов можно сразу же вызвать это лямбда-выражение.

Лямбда-выражения

Определение

```
lambda arg1 arg2 arg3... -> <expr>
```

Лямбда-выражения могут быть вложены друг в друга. Функции могут возвращать лямбда-выражения.

Литералы

Пока что кроме чисел и булевых значений, никаких литералов нет.

Для упрощения все числа являются дробными, потом это может быть изменено.

Числа

```
1234567890
1234567890.0987654321
6.022140857e23
1.8e-14
```

Булевые значения

```
true
false
```

Условные выражения

```
case <expr> of
    <pattern> -> <expr>;
    <pattern> -> <expr>;
    ...
end
```

Пока что в `<pattern>` входят только значения. Они могут быть являться лениво вычисленными, но обязательно должны быть сравнимыми. В паттерн нельзя передавать лямбда-выражения.

Операторы

Унарные операторы:

- `-`
- `!`

Унарные операторы имеют более высокий приоритет, чем вызов функции, и, соотвественно, более высокий приоритет, чем бинарные операторы.

Бинарные операторы

1 уровень приоритета:

- `|>` - pipe operator. Он передает результат левой части следующим аргументом функции правой части.

2 уровень приоритета:

- `*`
- `/`

3 уровень приоритета:

- `+`
- `-`

4 уровень приоритета:

- `==`
- `!=`
- `<`
- `>`
- `<=`
- `>=`

5 уровень приоритета:

- `and`

6 уровень приоритета:

- `or`

7 уровень приоритета:

- `xor`

8 уровень приоритета:

- `,` - создает кортеж

Итоговый приоритет:

- Унарные операторы
- Вызов функции
- Бинарные операторы