РАЗРАБОТКА ЯЗЫКА ПРОГРАММИРОВАНИЯ

Мищенко С. Д, научный руководитель - Ахтямов П. И.

Цель работы

Цель работы - разработка языка программирования, хорошо подходящего для проектов средней сложности.

Требования

- Easy to maintain / read code
- Fast project setup, easy to deal with packages
- Powerful abstractions
- Compact code
- Easy interloop with native code

Решения при проектировании

- **Поддержка кода:** Элементы синтаксиса, характерные для функциональных языков
- **Абстракции:** Параметрические типы, алгебраические типы, классы типов / trait-ы
- **Компактность кода:** Простой и консистентный синтаксис, удобные языковые конструкции
- **Интеграция с платформой и производительность:** Планируется сделать язык компилируемым, в основном императивный стиль программирования

Примеры

Примеры

type class / trait:

```
typeclass (Ord : #Eq) =
  & var ( < ) : Ord → Bool
  & var ( ≥ ) : Ord → Bool
  & var ( > ) : Ord → Bool
  & var ( ≤ ) : Ord → Bool

namespace var Ord {
  def ( ≥ ) : x = not: (self < x)
  def ( > ) : x = x < self
  def ( ≤ ) : x = not: (x < self)
}
```

Примеры

tuple type definition and tuples:

```
struct ThreeTuple = & String & String & String

decl scan_three_t : -> ThreeTuple
def scan_three_t = $ThreeTuple & IO.scan: & IO.scan: & IO.scan:

decl scan_three : -> (& String & String & String)
def scan_three = & IO.scan: & IO.scan: & IO.scan:

exec main {
  var & a & b & c = scan_three_t:
  ; IO.print: b
  var & d & e & f = scan_three:
  ; IO.print: e
}
```

Примеры

parametric functions:

```
decl scan_anything ('A : #Read) : -> 'A
def scan_anything = 'A.read: (IO.scan:)

exec main {
  var n = scan_anything Int:
  var a = for _ in 0--n do scan_int:
  ; print_anything Int: n
}
```

Примеры

Возможность "замедления" операторов

```
(((x + 3) * y) == 10) || ((a < 34) && (b > 33))
```

со стандартными приоритетами:

```
((x + 3) * y == 10) || (a < 34 && b > 33)
```

с "замедлением":

```
(x + 3 *. y ==.. 10) ||.. a < 34 &&. b > 32
```

Примеры

Возврат значений из блока в произвольном месте

```
{
  if a < 32 then bring 456
  bring 12
}
```

Примеры

Or-expressions (in development)

```
type Fruit =
  | Apple
  | Orange
  | Banana

decl some_fruit : → Fruit

exec main {
  var | maybe_apple | maybe_orange | maybe_banana = some_fruit:
  var fruit = | maybe_apple | maybe_orange | maybe_banana
}
```

Итог разработки

- Проведён анализ особенностей некоторых языков программирования
- Разработана грамматика языка (tree-sitter)
- Написана проверка корректности кода (не требующая исполнения кода) (C++)
- В основном написан интерпретатор (C++)

Дальнейшая разработка

- improved interpreter
- C / Wasm / IL (LLVM, QBE) code generation
- modules, module import, package manager
- stdlib, musl / libc integration
- some advanced language features (parametric type classes / traits, concurrency, memory management)