$c(x)$

Южный Федеральный Университет
Институт математики, механики и компьютерных наук имени Воровича

ЯЗЫК ПРОГРАММИРОВАНИЯ С ЗАВИСИМЫМИ УТОЧНЕННЫМИ ТИПАМИ

Докладчик: Квачев Всеволод Дмитриевич
Научный руководитель: Дубров Денис Владимирович

Мотивация

- Типы как средство повышения безопасности

Мотивация

- Типы как средство повышения безопасности
  - Refinement Types (пример на LiquidHaskell)

```
type Even = {v : Int | v mod 2 = 0}
weAreEven :: [Even]
weAreEven =  [10, 4, 0, 2, 666]
```

  - Dependent Types (пример на Idris)

```
app : Vect n a -> Vect m a -> Vect (n + m) a
app Nil       ys = ys
app (x :: xs) ys = x :: app xs ys
```

Мотивация

- Типы как средство повышения безопасности
- Типы как инструмент в решении задач

Мотивация

- Типы как средство повышения безопасности
- Типы как инструмент в решении задач

(пример на Scala)

```
trait Similarity {
  def isSimilar   (x: Any): Boolean
  def isNotSimilar(x: Any): Boolean =
      !isSimilar(x)
}
```

Мотивация

- Типы как средство повышения безопасности
- Типы как инструмент в решении задач
- Типы как предикаты
- Типы как множества

Мотивация

- Типы как средство повышения безопасности
- Типы как инструмент в решении задач
- Типы как предикаты
- Типы как множества

=>

- Облегчение разработки верифицированного ПО
- Повышение эффективности разработки

Примеры кода

```
// равенство, принадлежность типу
// и неравенство — утверждения на равном уровне
Int x = 0
Int y > 0

// объявление типа с произвольным ограничением
SortedPair(x, y) : if x > y
```

Примеры кода

```
// равенство, принадлежность типу
// и неравенство — утверждения на равном уровне
Int x = 0
Int y > 0

// объявление типа с произвольным ограничением
SortedPair(x, y) : if x > y

// пересечение типов
Positive = Int & (> 0)

// произвольный предикат в качестве типа
x : Char & isUppercase
```

Примеры кода

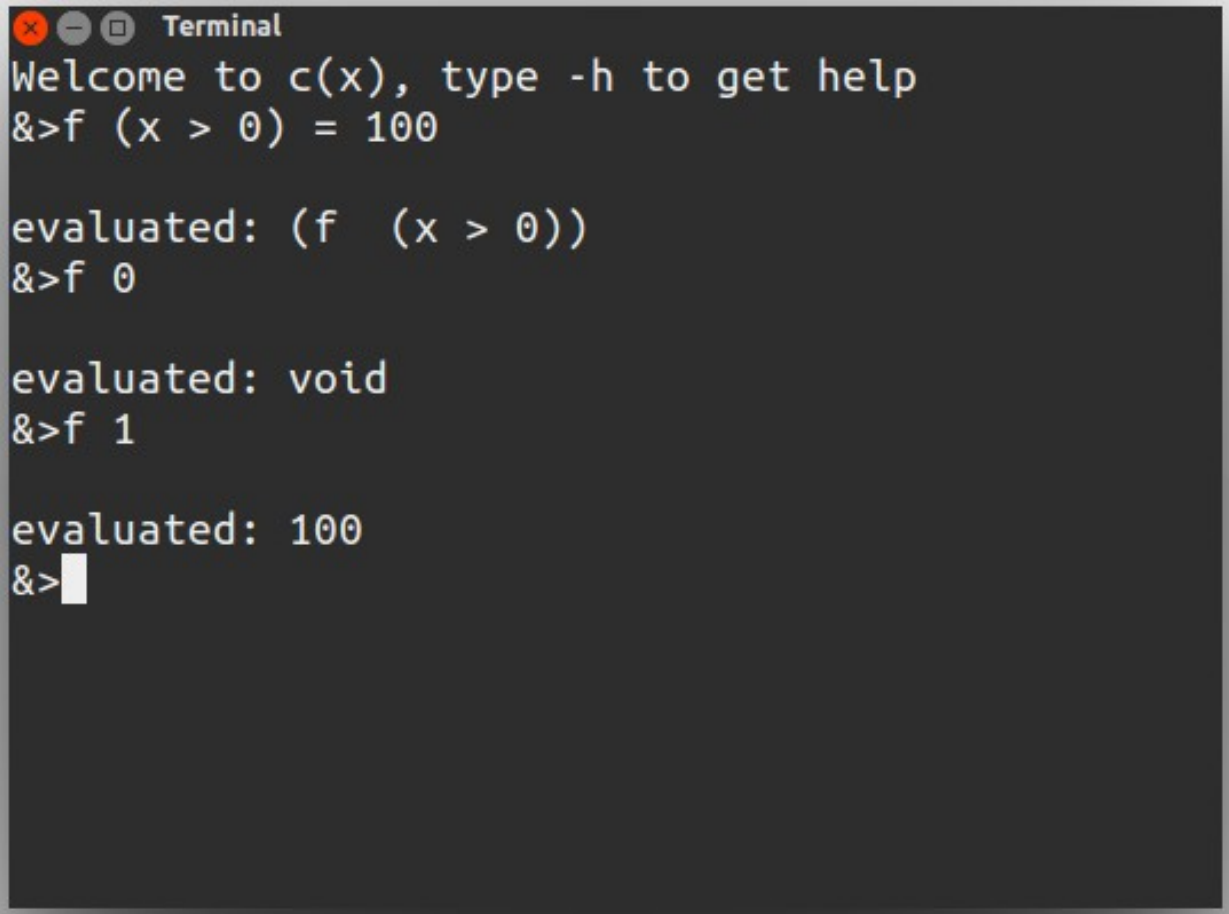
```
// объявление функции, индентационная зависимость
myFunction1 : Int → Int
              affects stdio
myFunction1 x =
    print x
    y = read Int
    return x + y
```

Примеры кода

```
// объявление функции, индентационная зависимость
myFunction1 : Int → Int
              affects stdio
myFunction1 x =
    print x
    y = read Int
    return x + y

// Pattern Matching
myFunction2(x > 0,
            ”str”,
            y - 10,
            isUppercase c) = ord c + y * x

// макро (функция, принимающая код)
printPlusOperands '(x + y) = print x
                             print y
                             return x + y
```

Практический пример

```
SortedList []
SortedList x::xt =
    if xt = y::yt then x >= y
    SortedList xt
    return x::xt

insertionSort : [Float] → SortedList
InsertionSort [] = []
insertionSort x::xt =
    insert y []     = [y]
    insert y z::zs  = if z >= y then y::z::zs
                                else z::(insert y zs)
    return insert x (insertionSort xs)
```


Terminal
Welcome to c(x), type -h to get help
&>f (x > 0) = 100
evaluated: (f (x > 0))
&>f 0
evaluated: void
&>f 1
evaluated: 100
&>

Выводы

- Разработан язык программирования, поддерживающий современные концепции
  - Refinement Types
  - Dependent Types,
  - Union & Intersection Types
  - etc
- Частично реализован прототип интерпретатора
- Проект представлен и вынесен на обсуждение на семинаре по языкам программировния и компиляторам

Планы на будущее

- Доработка прототипа интерпретатора
- Оценка и тестирование идей
- Изучение теории типов, компиляторостроения и теории категорий
- Выбор и продвижение отдельных идей

Литература

1. Chlipala A. Certified Programming with Dependent Types: A Pragmatic Introduction to the Coq Proof Assistant. — The MIT Press, 2013. — ISBN 0262026651, 9780262026659.

2. Knowles K., Flanagan C. Type Reconstruction for General Refinement Types // Proceedings of the 16th European Conference on Programming. — Braga, Portugal : Springer—Verlag, 2007. — Pp. 505–519. — (ESOP'07). — ISBN 978-3-540-71314-2. — URL: http://dl.acm.org/citation.cfm?id=1762174.1762220.

3. Rondon P. M., Kawaguci M., Jhala R. Liquid Types // SIGPLAN Not. — New York, NY, USA, 2008. — June. — Vol. 43, no. 6. — Pp. 159–169. — ISSN 0362-1340. — DOI: 10.1145/1379022.1375602. — URL: http://doi.acm.org/10.1145/1379022.1375602.

4. Vazou N., Bakst A., Jhala R. Bounded Refinement Types // SIGPLAN Not. — New York, NY, USA, 2015. — Aug. — Vol. 50, no. 9. — Pp. 48–61. — ISSN 0362-1340. — DOI: 10.1145/2858949.2784745. — URL: http://doi.acm.org/10.1145/2858949.2784745.

5. Vazou N., Seidel E. L., Jhala R. LiquidHaskell: Experience with Refinement Types in the Real World // SIGPLAN Not. — New York, NY, USA, 2014. — Sept. — Vol. 49, no. 12. — Pp. 39–51. — ISSN 0362-1340. — DOI: 10.1145/2775050.2633366. — URL: http://doi.acm.org/10.1145/2775050.2633366.

P. S.

- https://github.com/Rasie1/c-of-x
- rasielll@gmail.com
- Видеозапись семинара: http://youtube.com/8m0I0M5RHOk