5-й элемент

Опыт разработки и реализации языка Тривиль

```
вывод.ф("Тривиль-1 компилятор (трик): v$;\n", версия)
основа.старт()

выбор действие {
когда тестировать:
    компилятор.тестировать-модуль(аргумент)
другое
    если основа.отладка лексера() {
        компилятор.лексический анализ(аргумент)
    } иначе {
        компилятор.компилировать(аргумент)
    }
}
```

Алексей Недоря, август 2023

Расшифровка

5-й язык программирования:

1. Вир/a0
2. Вир/a1
3. Экспериментальный L2 (работа)
4. ArkTS (работа, open source)
5. Тривиль

Языки выходного дня

6. ....
7. ....
8. ...

План

- ❏ Зачем
- ❏ История разработки
- ❏ Статистика
- ❏ Принципы
- ❏ Находки и неудобства

Зачем делаю

Интенсивное программирование

Языки выходного дня:
L1: динамический язык
L2: статика + сборка мусора
+ компоненты
L3: статика + ARC (Swift)
L4: статика + системный (Rust)

Зачем рассказываю

Полигон для студентов:
- простота языка
- современный вид
- современные типы
- простота компилятора
- открытая лицензия

Тривиль

- язык для разработки компилятора и экосистемы
- современный, надежный, удобный
- простой и понятный
- русскоязычный
- минимально достаточный

История разработки

20.11.2022	Начало разработки, грамматика	http://алексейнедоря.рф/?p=422 http://алексейнедоря.рф/?p=427
02.12.2022	Компилятор на Го - первый коммит в репозиторий	
03.02.2023	Основные конструкции языка компилируются и работают (60 дней)	http://алексейнедоря.рф/?p=518
19.02.2023	Черновик описания языка (latex, pdf)	http://алексейнедоря.рф/?p=534
16.03.2023	Репозиторий сделан публичным	https://gitflic.ru/project/alekseinedoria/trivil-0
16.03.2023	Компилятор на Тривиле - первый коммит	
14.04.2023	Рассказ о Тривиле на семинаре	STEP-2023
25.06.2023	Тривиль компилятор на Тривиле компилирует сам себя (100 дней)	http://алексейнедоря.рф/?p=569

Статистика

Описание языка

- 50 страниц (PDF)

Компиляторы
(классическая схема)

- AST
- Лексер
- Парсер
- Семантика
  - разрешение имен
  - контроль типов
- Генерация (C99)

Компилятор на Го	Компилятор на Тривиле
11,200 строк, 10 пакетов, 57 файлов	10,500 строк, 21 модуль, 76 файлов

Runtime (Си)	1800 строк
Библиотеки (Тривиль)	2300 строк, 13 модулей

• вывод
• комстрока
• *контейнеры*
  ○ словарь
  ○ стек
• платформа
• строки
	• *тестирование*
	  ○ тест-основа
	  ○ тестер
	• файлы
	• *хеши*
	  ○ fnv
	• юникод
	• utf8

Общие принципы дизайна

Принципы:

- читабельность
- определение требований в начале работы и следование им

- модульность
- безопасные ссылки
- отсутствие явных указателей
- ООП, да не КЛОП

\+ конкретные требования!

Четко заданные требования приводят к упрощению и ускорению разработки языка и повышению качества языка.

Тривиль

- Находки
  - Обязательная инициализация полей и переменных
  - Выбор изменяемый/ неизменный
  - Безопасные ссылки
  - Оператор надо
  - Оператор выбор по типу
  - Оператор цикл
  - Обобщенные модули
  - Синтаксис и ключевые слова

- Неудобства
  - Отсутствие анонимных типов
  - Обобщенные модули

Последовательность

Обязательная инициализация, Выбор изменяемый/ неизменный	Пример “ини” стд::строки/формат.Разборщик
Безопасные ссылки	Пример “может” асд.выражения
ООП, да не КЛОП: АСД - сравнение с Го	Слайд + парсер.описание-переменной
Оператор “надо”	Пример “оп-надо” кон-типы.эквивалентные типы
Оператор “выбор”	Пример “оп-выбор”, “оп-выбор-т” кон-выражения
Оператор “цикл”	Пример “оп-цикл”
Обобщенные модули	стд::контейнеры/словарь

AST/АСД

```
type Node interface {
    GetPos() int
}

type Type interface {
    Node
    TypeNode()
}

type Decl interface {
    Node
    DeclNode()
    GetName() string
    GetType() Type
    GetHost() *Module //
    SetHost(host *Module)
    IsExported() bool
}

type Expr interface {
    Node
    ExprNode()
    GetType() Type
    SetType(t Type)
    IsReadOnly() bool
}

type Statement interface
    Node
```

```
type DeclBase struct {
    Pos      int
    Name     string
    Typ      Type
    Host     *Module
    Exported bool
}

func (n *DeclBase) DeclNode() {}

func (n *DeclBase) GetPos() int {
    return n.Pos
}

func (n *DeclBase) GetName() string {
    return n.Name
}

func (n *DeclBase) GetType() Type {
    return n.Typ
}

func (n *DeclBase) GetHost() *Module {
    return n.Host
}
```

```
тип Узел* = класс {
   поз*: Цел64 = позже // позиция в тексте
}

тип Тип* = класс (Узел) {}
тип Типы* = []Тип

тип Описание* = класс (Узел) {
   имя*: Строка := ""
   Т*: мб Тип := пусто
   владелец*: мб Модуль := пусто
   экспорт* := ложь
}
тип Описания* = []Описание

тип Выражение* = класс (Узел) {
   Т*: мб Тип := пусто
   только-чтение* := ложь
}
тип Выражения* = []Выражение

тип Оператор* = класс (Узел) {}
тип Операторы* = []Оператор
```

Вопросы и ответы