

Язык программирования Тривиль

Алексей Недоря

12.02.2023

Содержание

| | | |
|----------|--|----------|
| 1 | Назначение | 3 |
| 2 | Обзор языка | 4 |
| 3 | Лексика | 5 |
| 3.1 | Комментарии | 5 |
| 3.2 | Разделители синтаксических конструкций | 5 |
| 3.3 | Идентификаторы | 6 |
| 3.4 | Ключевые слова | 7 |
| 3.5 | Знаки операций и знаки препинания | 7 |
| 3.6 | Целочисленные литералы | 7 |
| 3.7 | Вещественные литералы | 7 |
| 3.8 | Строковые литералы | 8 |
| 3.9 | Символьные литералы | 8 |

1. Назначение

Язык программирования Тривиль разработан в рамках работы на семейством языков программирования "Языки выходного дня"(ЯВД) [link](#) проекта Интенсивное программирования [link](#).

Тривиль является нулевым языком семейства ЯВД, предназначенным для реализации компиляторов и экосистемы других языков семейства. В рамках классификации языков, принятом в проекте Интенсивное программирования, это язык L2.

Основными требованиями к языку при разработке были поставлены

- Язык должен быть минимально достаточным для удобной разработки компиляторов. Требование это во многом субъективно, так как компиляторы можно писать существенно по разному.
- Язык должен быть русскоязычным и с синтаксисом минимизирующим переключение на латиницу в процессе разработки программ.

Название языка происходит от слова "тривиальный" что означает, что при разработке языка практически везде использовались решения, проверенные в других современных языках программирования, в первую очередь "донорами" являются Go, Swift, Kotlin и Oberon.

Несмотря на узкую направленность на разработку компиляторов, Тривиль является языком программирования общего назначения, пригодным для решения широкого круга задач.

Язык (и экосистема) обладает существенными предпосылками для использование его в качестве учебного языка для обучения студентов разработке компиляторов, библиотек, средств разработки, алгоритмов оптимизации и так далее, в первую очередь это:

- Простота языка
- Современный вид и набор конструкций языка
- Простота компилятора
- Открытая лицензия.

2. Обзор языка

Тривиль - это модульный язык с явным экспортом и импортом, автоматическим управлением памятью (сборка мусора), с поддержкой ООП.

Программа на языке Тривиль состоит из модулей (единиц компиляции), исходный текст каждого модуля расположен в одном или нескольких исходных файлах.

Пример программы:

```
1  модуль x
2
3  импорт "std/вывод"
4
5  вход {
6      вывод.ф("Привет! \n")
7  }
```

Для описания языка используется EBNF в формате ANTLR4 [link](#).

Программа
: Модуль+
;

3. Лексика

Исходный текст есть последовательность лексем: идентификаторов, ключевых слов, литералов, знаков операций и знаков препинания. Каждая лексема состоит из последовательности Unicode символов (unicode code point) в кодировке UTF-8.

Пробелы (U+0020), символы табуляции (U+0009) и символы завершения строки (U+000D, U+000A) разделяют лексемы, и, игнорируются, кроме следующих случаев:

- Символы завершения строк могут использоваться как разделители синтаксических конструкций (§3.2).
- Пробелы являются значащими символами в идентификаторах, состоящих из нескольких слов (§3.3).
- Пробелы являются значащими в строковых и символьных литералах (§3.3).

Несколько таких разделителей трактуются, как один.

Исходный текст может содержать *комментарии*.

3.1. Комментарии

Есть две формы комментариев:

- Строчный комментарий начинается с последовательности символов `'/'` и заканчивается в конце строки.
- Блочный комментарий начинается с последовательности символов `'/*'` и заканчивается последовательностью символов `'*/'`. Блочные комментарии могут быть вложенные.

Комментарий:

```
: '/' (любой символ, кроме завершения строки) *  
| '/*' (любой символ) * '*/'  
;
```

3.2. Разделители синтаксических конструкций

Некоторые синтаксические правила используют нетерминал *Разделитель* для разделения двух подряд идущих синтаксических конструкций, например:

Список-операторов

```
: Оператор (Разделитель Оператор) *  
;
```

В качестве разделителя может использоваться символ `';` или символ завершения строки.

Разделитель

```
: ';'
| символ-завершения-строки
;
```

Пример:

```
1  а := 1; б := 2
2  в := 1
```

В строке 1 операторы разделены символом ';', а оператор в строке 2 отделен от операторов строки 1 символом завершения строки.

Ошибка компиляции - нет разделителя:

```
1  а := 1 б := 2
```

3.3. Идентификаторы

Идентификатор - это последовательность *слов*, разделенных пробелами или символами дефис '-' с опционально завершающим знаком препинания:

Каждое слово состоит из *Букв* и цифр, и начинается с Буквы. Буквой считается любой Unicode символ, имеющий признак *Letter*, и, дополнительно, символы '№' и '_'.

Идентификатор

```
: Слово ((' ' | '-') Слово)* Знак-препинания?
;
```

Слово

```
: Буква (Буква | Цифра)*
;
```

Буква

```
: Unicode-letter
| '_'
| '№'
;
```

Знак-препинания

```
: '?'
| '!'
;
```

Цифра

```
: '0' .. '9'
;
```

Примеры идентификаторов:

```
1  буква
2  буква-или-цифра
3  №-символа
```

| | |
|---|------------------|
| 4 | Цифра? |
| 5 | Пора паниковать! |

3.4. Ключевые слова

Следующие ключевые слова зарезервированы и не могут быть использованы, как идентификаторы:

| | | | | |
|---------|--------|--------|-----------|----------|
| авария | есть | когда | надо | прервать |
| вернуть | иначе | конст | осторожно | пусть |
| вход | импорт | мб | пока | тип |
| если | класс | модуль | позже | фн |

3.5. Знаки операций и знаки препинания

Следующие последовательности символов представляют знаки операций и знаки препинания:

```

+   -   *   /   %
&   |   ~
=   #   <   <=  >   >=
:=  ++  --
(   )   [   ]   {   }
(:   .   ^   ,   :   ;

```

3.6. Целочисленные литералы

Целочисленный-литерал

```

: Цифра+
| '0x' Цифра16+
;

```

Цифра16

```

: '0'..'9'
| 'a'..'f'
| 'A'..'F'
;

```

3.7. Вещественные литералы

В текущей реализации есть только одна форма записи вещественных литералов, без экспоненты.

Вещественный-литерал

```

: Цифра+ '.' Цифра*
;

```

3.8. Строковые литералы

Строковый литерал - это последовательность символов, заключенные в двойные кавычки. Строковый литерал может содержать символы, закодированные с помощью escape-последовательности, которая начинается с символа '\ '.

Строковый литерал

```
: '"'
  (~('"' | '\\\' | '\n' | '\r' | '\t') | Escape)*
'"'
;

Escape
: '\\\'
( 'u' Цифра16 Цифра16 Цифра16 Цифра16
| 'n' | 'r' | 't'
| '"'
| "'"
)
;
```

3.9. Символьные литералы

Символьный литерал задает значение для Unicode code point, - это последовательность символов, заключенные в двойные кавычки. Он записывается как один или несколько символов, заключенных в одинарные кавычки. Символьный литерал может быть закодирован с помощью escape-последовательности, которая начинается с символа '\ '.

Символьный литерал

```
: "'"
  (~("'" | '\\\' | '\n' | '\r' | '\t') | escape_value)
"'"
;
```
