Студент: Александр Кузнецов
Группа: SE
Дата: 12 мая 2018 г.

Конкретный синтаксис языка L

Конкретный синтаксис следующий:

1. Выражения отделяются друг от друга разделителем — точкой с запятой;
2. Предикаты в конструкциях `if-then-else` и `while-do` заключены в круглые скобки: `if (x == 5) then ...`;
3. Области видимости заключены в фигурные скобки: `if (x == 5) then { y := 1; } else { y := x - 3; }`;
4. Аргументы в `read` и `write` заключены в круглые скобки, отделяются друг от друга запятыми: `read(a)`, `write(a, a+b)`;
5. Аргументы функции указаны в скобках и отделены друг от друга запятыми: `foo(a, b, c)`;
6. Объявление функции начинается с ключевого слова `fun`, за которым следует идентификатор-имя функции, а затем аргументы функции. Возвращаемое значение функции следует за ключевым словом `return`: `fun bar(a, b) { return a + b; }`
7. В лексическом анализаторе предусмотрено наличие оператора `<-`, он используется в однострочных функциях, возвращающих значение. Например, вместо `fun bar(a, b) { return a + b; }` можно записать просто `fun bar(a, b) <- a + b;`
8. Многострочные комментарии – это конструкции вида `/* ... */`. При этом тело комментария заканчивается ровно в момент первой встречи конструкции `*/` (внутри тела может быть сколько угодно конструкций `/*`). Например, строка `/* /* /* /* text */` – это один комментарий, а строка `/* /* /* /* text */ */` будет разбита на лексемы: **многострочный комментарий**, **умножение**, **деление**.

Данный синтаксис подразумевает, что пробельные символы значимы только при объявлении функций и при возвращении значения в функции. (У меня есть подозрение, что любой корректный код без функций тогда можно записать в одну строку и без пробелов).

Впоследствии можно будет попробовать избавиться от скобок в выражениях с операторами `read` и `write` (раз уж это зарезервированные языком слова, то можно отделить их внешне от пользовательских функций). Можно было бы также избавиться от запятых, а также убрать скобки в определении и вызове функций, как в функциональных языках, но это, на мой взгляд, ненужное усложнение, так как удобства использования языка в данном случае это не добавит.

Еще есть вариант с избавлением от фигурных скобок, точки с запятой и использованием переноса строки как разделителя, но это добавит дополнительные сложности, например, придется учитывать индентацию, чтобы отделять одну область видимости от другой. Более того, исчезнет возможность записи в одну строку нескольких каких-нибудь тривиальных выражений, и в такой ситуации было бы неплохо ввести какой-нибудь синтаксический сахар. Все вместе получается чем-то совсем сложным. Замена фигурных скобок на ключевые слова типа `begin` и `end`, как мне кажется, необоснована: во-первых, никаких особых преимуществ эти слова не несут, а код становится длиннее; во-вторых, придется и здесь учитывать пробельные символы.

Синтаксический сахар

1. Функции, тело которых состоит из единственной строчки, возвращающей значение, можно записать, используя оператор <-. Например, конструкции

```
fun foo(x, y) {
    return 7 ** 8;
}
```

и

```
fun foo(x, y) <- 7 ** 8;
```

имеют одно и то же AST.

2. с if-ами можно работать как с выражениями, если их тела представляют собой выражения. Например, такой if можно присвоить переменной:

```
x := if (true != false) then 1 else 2;
```

3. Можно присваивать одно и то же значение сразу нескольким переменным:

```
x := y := z := 3;
```