Формальные языки

домашнее задание до 23:59 01.04

1. Написать лексер для языка L (спецификация ниже), используя любимый инструмент (например, генератор лексеров из семейства Lex, ANTLR, парсер-комбинаторы, написать вручную...). (8 баллов за полностью выполненное задание)

   - Структуры данных для лексем должны однозначно их идентифицировать, а также содержать привязку к коду (в какой строке исходного кода и с какого по какой символ располагается лексема).
   - Составить набор тестов, демонстрирующий правильность работы полученного лексера (качество тестового покрытия важно!).
   - Сделать консольное приложение, принимающее на вход путь к файлу, содержащему программу на языке L, производящее лексический анализ и печатающее полученный поток лексем.
     - Результатом работы лексического анализатора должен быть поток лексем, который печатается при помощи *отдельной* процедуры печати.
   - Код должен быть размещен на гитхабе, собираться одним скриптом, содержать инструкцию по сборке и запуску собранного приложения, собираться на чистой Ubuntu 16.04 или Windows 10. Все зависимости, в случае их отсутствия в системе, должны доставляться скриптом.
     - Инструкция по запуску должна содержать информацию о том, где находится бинарник, как именно его полагается запускать, какой формат аргументов командной строки, куда пишется результат.

2. Предложить удобный конкретный синтаксис для языка L (абстрактный синтаксис приведен ниже, будьте внимательны, он немного отличается от того, что был на паре) — описать в pdf. (2 балла)

   - Удобность тут понятие относительное — какой вам синтаксис нравится, такой и используйте.
   - Какие есть варианты. Разделять ли операторы языка разделителями или сделать значимыми переносы строк и отсупы? Использовать скобки для группировки блоков кода, ключевые слова begin/end или маркеры конца блока, зависящие от того, в контексте какого оператора мы находимся? Использовать ли скобки для аргументов функций? Может вы хотите предоставить синтаксический сахар для облегчения синтаксиса?

3. *Это задание на через пару, но его полезно учитывать при разработке лексера и конкретного синтаксиса.* Написать парсер для предложенного конкретного синтаксиса языка L (описание языка ниже), используя любимый способ писать парсеры. Не забыть про тесты (8 баллов)

   - Можно реализовать любой понравившийся вам алгоритм синтаксического анализа.
   - Можно пользоваться парсер-комбинаторами, можно даже реализовать свою библиотеку.
   - Можно использовать генераторы синтаксических анализаторов (yacc, bison, antlr, любой другой).
   - Желательно, чтобы ваш синтаксический анализатор принимал на вход то, что выдает ваш лексер — те самые токены с позициями во входной строке; если для этого нужно править лексер — правьте. Если технологии не сочетаются, реализовать функциональность лексического анализа, как требуется в вашем случае.
   - Создайте консольное приложение для запуска синтаксического анализа.
     - Консольное приложение обязательно должно принимать адрес файла со входной программой
     - Программа может быть многострочной
     - Результатом синтаксического анализа должно быть *абстрактное синтаксическое дерево*, напечатанное в человекочитаемом формате; можно в файл, но название файла должно быть связано с названием входного файла (можно в dot).
     - Лексемы в дереве вывода должны отображаться со всей необходимой информацией (тип лексемы, значение и привязка к коду).
     - Код должен быть размещен на гитхабе, собираться одним скриптом, содержать инструкцию по сборке и запуску собранного приложения, собираться на чистой Ubuntu 16.04 или Windows 10. Все зависимости, в случае их отсутствия в системе, должны доставляться скриптом.

Лексическая спецификация языка L

Программы на языке L записываются символами ASCII. Ограничителями строк являются ASCII-символы CR, LF или 2 подряд идущих символа: CR LF. Пробельными символами являются символы-ограничители строк и символы пробела (SP), табуляции (HT) и перевода страницы (FF). Пробельные символы не имеют значения, для них не должно генерироваться лексем. Все буквы строчные. Программы могут быть многострочными.

Комментарии пока только однострочные. Комментарием считается суффикс строки до символа-ограничителя строк, начинающийся с "//".

Идентификаторы в языке — произвольная последовательность символов латинского алфавита, цифр и символа подчеркивания (_) (до первого пробельного символа, разделителя или оператора), начинающаяся либо с буквы, либо символа подчеркивания, которая не является ключевым словом или литералом.

Ключевые слова языка: `if`, `then`, `else`, `while`, `do`, `read`, `write`

Литералы языка: числа с плавающей точкой, `true`, `false`.

Числа с плавающей точкой должны соответствовать спецификации языка Java (по ссылке): обратите внимание на подчеркивания. В лексеме для числа должно присутствовать его значение — численного типа, не строковое представление.

Операторы языка: $+, -, *, /, \%, ==, !=, >, >=, <, <=, \&\&, ||$

Разделители языка: (,), ;

Пример программы (в каком-то конкретном синтаксисе):

$$\texttt{read}\ x;\ \texttt{if}\ y + 1 == x\ \texttt{then}\ \texttt{write}\ y\ \texttt{else}\ \texttt{write}\ x$$

Результат лексического анализа должен быть в духе:

```
KW_Read(0, 0, 3); Ident("x", 0, 5, 5); Colon(0, 6, 6); KW_If(0, 8, 9); Ident("y", 0, 11, 11);
Op(Plus, 0, 13, 13); Num(1, 0, 15, 15); Op(Eq, 0, 17, 18); Ident("x", 0, 20, 20); KW_Then(0, 22, 25);
KW_Write(0, 27, 31); Ident("y", 0, 33, 33); KW_Else(0, 35, 38); KW_Write(0, 40, 44); Ident("x", 0, 46, 47);
```

Абстрактный синтаксис языка L

X — счетно-бесконечное множество идентификаторов

$$\otimes = \{+, -, *, /, \%, ==, !=, >, >=, <, <=, \&\&, ||\}$$

- Определения (функций): $\mathcal{D} = \mathcal{X}_{name}\ \mathcal{X}_0 \dots \mathcal{X}_k \leftarrow \mathcal{S}$. $\mathcal{X}_{name}$ — имя функции; $\mathcal{X}_0 \dots \mathcal{X}_k$ — ее аргументы; $\mathcal{S}$ — тело.
- Вызовы функций: $\mathcal{C} = \mathcal{X}_{name}\ \mathcal{E}_0 \dots \mathcal{E}_k$. *Аргументами могут быть произвольные выражения.*
- Выражения: $\mathcal{E} = \mathcal{C} \cup X \cup \mathbb{N} \cup (\mathcal{E} \otimes \mathcal{E})$. *Вызовы функций могут быть использованы в выражениях.* В выражениях могут использоваться круглые скобки.
- Операторы:

$$\begin{array}{lll} \mathcal{S} = & \mathcal{X} := \mathcal{E} & \cup \\ & \mathcal{C} & \cup \\ & \texttt{write}\ \mathcal{E} & \cup \\ & \texttt{read}\ \mathcal{X} & \cup \\ & \texttt{while}\ \mathcal{E}\ \texttt{do}\ \mathcal{S} & \cup \\ & \texttt{if}\ \mathcal{E}\ \texttt{then}\ \mathcal{S}\ \texttt{else}\ \mathcal{S} & \cup \\ & \mathcal{S}^* \textit{ (Последовательность операторов)} & \end{array}$$

- Программы: $\mathcal{P} = (\mathcal{D}^*, \mathcal{S})$ — несколько определений, за которыми следует текст самой программы