Формальные языки

домашнее задание до 23:59 29.04

1. Изучить описание синтаксиса вашего второго самого любимого языка программирования. Найти пару особенностей синтаксиса, о которых вы раньше не знали. В отчете указать, где и какую спецификацию читали и привести примеры программ с найденными особенностями. (2 балла)

2. Провести синтаксический анализ при помощи алгоритма CYK: вручную или реализовать и запустить код. Язык — корректные арифметические выражения над цифрами с операциями $-$ и $*$ с естественными приоритетом и ассоциативностью. Проанализировать хотя бы одну корректную цепочку длины не меньше 7 и хотя бы одну некорректную цепочку длины не меньше 7. Результатом должна быть заполненная таблица и дерево вывода (в случае ошибки дерева не должно быть). (3 балла)

3. Реализовать парсер для предложенного вами 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. Комментарии перед синтаксическим анализом имеет смысл отфильтровать
   - 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.

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:

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

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