Государственное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет
имени Н.Э. Баумана»

Отчет
По лабораторной работе
По курсу «Конструирование компиляторов»
На тему
«Синтаксический разбор с использованием метода рекурсивного спуска»

Студент: Горин Д.И.
Группа: ИУ7-23М
Вариант: 3
Преподаватель: Ступников А.А.

Москва, 2020

Оглавление

1 Цель и задачи работы

Цель работы: Дополнить грамматику блоком, состоящим из последовательности операторов присваивания. Для реализации предлагаются два варианта расширенной грамматики.

2 Листинг

2.1 main.py

```
1  from grammar import Grammar
2  from ll import build_tree
3
4
5  if __name__ == '__main__':
6      g = Grammar.init_from_json_file('grammar_g3_no_lrec_c.json')
7      print(g, end='\n\n')
8
9      initial = g.initial_nterm
10
11     expr = '{x=2==2}'
12     print(expr)
13     is_ok, _ = build_tree(grammar=g, current_symbol=initial,
           string_to_read=expr)
14     print(f'{is_ok}, должно быть True\n')
15
16     expr = '{x=2*3>=2/4}'
17     print(expr)
18     is_ok, _ = build_tree(grammar=g, current_symbol=initial,
           string_to_read=expr)
19     print(f'{is_ok}, должно быть True\n')
20
21     expr = '{x=(2+3)<2}'
22     print(expr)
23     is_ok, _ = build_tree(grammar=g, current_symbol=initial,
           string_to_read=expr)
24     print(f'{is_ok}, должно быть True\n')
25
26     expr = '{x=2===2}'
27     print(expr)
28     is_ok, _ = build_tree(grammar=g, current_symbol=initial,
           string_to_read=expr)
29     print(f'{is_ok}, должно быть False\n')
30
31     expr = '{x=2=3/4==2}'
32     print(expr)
33     is_ok, _ = build_tree(grammar=g, current_symbol=initial,
           string_to_read=expr)
34     print(f'{is_ok}, должно быть False\n')
```

2.2 ll.py

```
1  from grammar import Grammar, NoTermSymbol, TermSymbol
2
3
4  def build_tree(grammar: Grammar, current_symbol, string_to_read) ->
       (bool, str):
5      flag = False
6      new_str = string_to_read
7      for production in grammar.rules[current_symbol]:
8          for prod_sym in production:
9              if isinstance(prod_sym, NoTermSymbol):
10                 flag, new_str = build_tree(grammar, prod_sym,
                       new_str)
11             else:
12                 cur_term_sym = prod_sym.symbol
13                 if cur_term_sym == grammar.eps_terminal.symbol:
14                     flag = True
15                 elif cur_term_sym == new_str[:len(cur_term_sym)]:
16                     flag = True
17                     new_str = new_str[len(cur_term_sym):]
18                 else:
19                     flag = False
20             if not flag:
21                 break
22         if flag:
23             break
24     return flag, new_str
```

3 Тесты

1. $\{x = 2 == 2\} == True$

2. $\{x = 2 * 3 >= 2/4\} == True$

3. $\{x = (2 + 3) < 2\} == True$

4. $\{x = 2 === 2\} == False$

5. $\{x = 2 = 3/4 == 2\} == False$

4 Выводы

По результатам проведенной работы студент приобрел практические навыки в реализации алгоритма синтаксического разбора с использованием рекурсивного спуска

5 Список литературы

1. БЕЛОУСОВ А.И., ТКАЧЕВ С.Б. Дискретная математика: Учеб. Для вузов / Под ред. В.С. Зарубина, А.П. Крищенко. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2001.

2. АХО А., УЛЬМАН Дж. Теория синтаксического анализа, перевода и компиляции: В 2-х томах. Т.1.: Синтаксичечкий анализ. - М.: Мир, 1978.

3. АХО А.В, ЛАМ М.С., СЕТИ Р., УЛЬМАН Дж.Д. Компиляторы: принципы, технологии и инструменты. – М.: Вильямс, 2008.