

第七次编译上机作业

运行截图

```
[linxuan@linxuan-5390 Python Code]$ cd "/run/media/linxuan/Data/Python Code/Python Code" ; /usr/bin/env /bin/python /home/linxuan/.vscode/extensions/ms-python.python-2021.11.1422169775/pythonFiles/lib/python/debugpy/launcher 36001 -- "/run/media/linxuan/Data/Python Code/Python Code/src/task_CompilationPrinciple/task_6_.py"
请输入计算的表达式(直接敲击回车默认使用 4**2+4):
移进('id', 4)
按照F -> ['id']进行归约
按照T -> ['F']进行归约
移进('*', '')
error happend: drop ('*', '')
移进('id', 2)
按照F -> ['id']进行归约
按照T -> ['T', '*', 'F']进行归约
按照E -> ['T']进行归约
移进('+', '')
移进('id', 4)
按照F -> ['id']进行归约
按照T -> ['F']进行归约
按照E -> ['E', '+', 'T']进行归约
accept, anylasis success!
表达式的值为12
```

运行说明:

直接运行文件, 按照提示输入计算表达式即可

代码:

```
1  '''
2  Author: LinXuan
3  Date: 2021-11-09 20:32:16
4  Description: 改造上一次的LR分析, 使其支持翻译方案
5  LastEditors: LinXuan
6  LastEditTime: 2021-12-06 22:46:13
7  FilePath: /Python Code/src/task_CompilationPrinciple/task_6_.py
8  '''
9  import re
10 import json
11
12 # 构造分析表
13
14
15 def get_table():
16     talbe_json = '''{
17         "I0": { "E": "1", "T": "2", "F": "3", "(" : "s4", "id":
18             "s5" },
19         "I1": { "$": "acc", "+": "s6", "-": "s7" },
```

```
19      "I2": { "$": "R3", "-": "R3", "+": "R3", "*": "S8",
20      "/", "S9" },
21      "I3": { "/": "R6", "*": "R6", "+": "R6", "-": "R6",
22      "$": "R6" },
23      "I4": { "E": "10", "T": "11", "F": "12", "(" : "S13",
24      "id": "S14" },
25      "I5": { "/": "R8", "*": "R8", "+": "R8", "$": "R8", "-
26      ": "R8" },
27      "I6": { "T": "15", "F": "3", "(" : "S4", "id": "S5" },
28      "I7": { "T": "16", "F": "3", "(" : "S4", "id": "S5" },
29      "I8": { "F": "17", "(" : "S4", "id": "S5" },
30      "I9": { "F": "18", "(" : "S4", "id": "S5" },
31      "I10": { ")" : "S19", "+" : "S20", "-" : "S21" },
32      "I11": { ")" : "R3", "-" : "R3", "+" : "R3", "*": "S22",
33      "/", "S23" },
34      "I12": { "/": "R6", "*": "R6", "+" : "R6", ")" : "R6", "-
35      ": "R6" },
36      "I13": { "E": "24", "T": "11", "F": "12", "(" : "S13",
37      "id": "S14" },
38      "I14": { "/": "R8", "*": "R8", "+" : "R8", ")" : "R8", "-
39      ": "R8" },
40      "I15": { "$": "R1", "-": "R1", "+" : "R1", "*": "S8",
41      "/", "S9" },
42      "I16": { "$": "R2", "-": "R2", "+" : "R2", "*": "S8",
43      "/", "S9" },
44      "I17": { "/": "R4", "*": "R4", "+" : "R4", "-": "R4",
45      "$": "R4" },
46      "I18": { "/": "R5", "*": "R5", "+" : "R5", "-": "R5",
47      "$": "R5" },
48      "I19": { "/": "R7", "*": "R7", "+" : "R7", "$": "R7", "-
49      ": "R7" },
50      "I20": { "T": "25", "F": "12", "(" : "S13", "id": "S14"
51      },
52      "I21": { "T": "26", "F": "12", "(" : "S13", "id": "S14"
53      },
54      "I22": { "F": "27", "(" : "S13", "id": "S14" },
55      "I23": { "F": "28", "(" : "S13", "id": "S14" },
56      "I24": { ")" : "S29", "+" : "S20", "-" : "S21" },
57      "I25": { ")" : "R1", "-" : "R1", "+" : "R1", "*": "S22",
58      "/", "S23" },
59      "I26": { ")" : "R2", "-" : "R2", "+" : "R2", "*": "S22",
60      "/", "S23" },
```

```

44         "I27": { "/": "R4", "*": "R4", "+": "R4", ")": "R4", "-": "R4" },
45         "I28": { "/": "R5", "*": "R5", "+": "R5", ")": "R5", "-": "R5" },
46         "I29": { "/": "R7", "*": "R7", "+": "R7", ")": "R7", "-": "R7" }
47     }
48     '''
49     return json.loads(talbe_json)
50
51
52 def lexical_sentence(sentence: str):
53     # 分隔输入，只支持算数表达式
54     input: list = list(re.findall(r"[0-9]+|[\+\-\*/\(\)]$", sentence))
55     lexical_words = []
56     for item in input:
57         item: str
58         if item.isdigit():
59             lexical_words.append(("id", int(item)))
60         else:
61             lexical_words.append((item, ''))
62     return lexical_words
63
64
65 def LR_analysis(lexical_words: list, table: dict, grammar="", begin="E"):
66     lexical_words.append("$", '') # 放入终结符
67     lexical_words.reverse() # 倒置为栈
68     # 初始化栈
69     stack = ['0'] # 状态栈
70     val_stack = [''] # 计算值的栈
71
72     while True:
73         status = 'I' + stack[-1]
74         item = lexical_words[-1]
75
76         # 错误处理
77         if item[0] not in table[status]:
78             lexical_words.pop()
79             print(f"error happend: drop {item}")
80             continue
81

```

```

82         # 结束条件
83         action = table[status][item[0]]
84         if action == "acc":
85             print("accept, anylasis success!")
86             eval(grammar['0'][2]) # 执行R0对应的翻译动作
87             break
88
89         # 移进
90         op, label = action[0], action[1:]
91         if op == "s":
92             stack.append(item[0])
93             stack.append(label)
94
95             val_stack.append(item[1])
96             val_stack.append('')
97             lexical_words.pop()
98             print(f"移进{item}")
99         # 归约
100        elif op == "R":
101            index = label
102            left, right = grammar[index][0], grammar[index][1]
103            value = eval(grammar[index][2]) # 执行翻译动作
104
105            # 弹出旧字符
106            for symbol in reversed(right):
107                while symbol is not stack[-1]:
108                    stack.pop()
109                    val_stack.pop()
110                    stack.pop()
111                    val_stack.pop()
112
113            status = 'I' + stack[-1]
114            goto = table[status][left]
115
116            stack.append(left)
117            stack.append(goto)
118            val_stack.append(value)
119            val_stack.append('')
120            print(f"按照{left} -> {right}进行归约")
121        pass
122
123
124    def main():

```

```

125     grammar = {
126         r'0': (r'L', [r'E'], r"print(f'表达式的值为
{val_stack[-2]}')"),
127         r'1': (r'E', [r'E', '+', 'T'], r"val_stack[-6] +
val_stack[-2]"),
128         r'2': (r'E', [r'E', '-', 'T'], r"val_stack[-6] -
val_stack[-2]"),
129         r'3': (r'E', [r'T'], r"val_stack[-2]"),
130         r'4': (r'T', [r'T', '*', 'F'], r"val_stack[-6] *
val_stack[-2]"),
131         r'5': (r'T', [r'T', '/', 'F'], r"val_stack[-6] /
val_stack[-2]"),
132         r'6': (r'T', [r'F'], r"val_stack[-2]"),
133         r'7': (r'F', [r'(', 'E', ')'], r"val_stack[-4]"),
134         r'8': (r'F', [r'id'], r"val_stack[-2]"),
135     }
136     sentence = "4**2+4" # 对应的表达式
137     print("请输入计算的表达式(直接敲击回车默认使用 4**2+4):", end='')
138     input_sentenct = input()
139     if input_sentenct != '':
140         sentence = input_sentenct
141         lexical_words = lexical_sentence(sentence=sentence)
142         table = get_table()
143         LR_analysis(lexical_words=lexical_words, table=table,
grammar=grammar, begin='L')
144
145     pass
146
147
148 if __name__ == "__main__":
149     main()
150

```