2110316  
(2018/1)  
Programming Languages Principles  
 Compiler  
ชยพล ปัญจะเทวคุปต์ 5931011221

Part Language Analysis

จาก grammar พบว่ามี left recursion การทำ LL(1) parser จะมีจุดที่ต้องแก้ไข คือ ที่ exp และ cond (ใช้ exp\* และ cond\*)

pgm := line pgm | EOF  
line := line\_num stmt  
stmt := asgmnt | if | print | goto | stop  
asgmnt := id = exp  
exp := term exp\*  
exp\* := + term | - term | EMPTY  
term := id | const  
if := IF cond line\_num  
cond := term cond\*  
cond\* := < term | = term  
print := PRINT id  
goto := GOTO line\_num  
stop := STOP  
  
id is {A..Z}  
const is {1..100}  
line\_num is {1..1000}

first set:

pgm = {line\_num, EOF}

line = {line\_num}

stmt = {id, IF, PRINT, GOTO, STOP}

asgmnt = {id}

exp = {id, const}

exp\* = {+, -, EMPTY}

term = {id, const}

if = {IF}

cond = {id, const}

cond\* = {<, =}

print = {PRINT}

goto = {GOTO}

stop = {STOP}

follow set:

pgm = {EOF}

line = {EOF, line\_num}

stmt = {EOF, line\_num}

asgmnt = {EOF, line\_num}

exp = {EOF, line\_num}

exp\* = {EOF, line\_num}

term = {+, -, EOF, line\_num}

term = {+, -, EOF, line\_num}

if = {EOF, line\_num}

cond = {line\_num}

cond\* = {line\_num}

print = {EOF, line\_num}

goto = {EOF, line\_num}

stop = {EOF, line\_num}

rules set:

1. pgm := line pgm

2. pgm := EOF

3. line := line\_num stmt

4. stmt := asgmnt

5. stmt := if

6. stmt := print

7. stmt := goto

8. stmt := stop

9. asgmnt := id = exp

10. exp := term exp\*

11. exp\* := + term

12. exp\* := - term

13. exp\* := EMPTY

14. term := id

15. term := const

16. if := IF cond line\_num

17. cond := term cond\*

18.cond\* := < term

19.cond\* := = term

20. print := PRINT id

21. goto := GOTO line\_num

22. stop := STOP

Parsing Table

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Line\_num | Id | const | IF | PRINT | GOTO | STOP | + | - | < | = | EOF |
| pgm | 1 |  |  |  |  |  |  |  |  |  |  | 2 |
| line | 3 |  |  |  |  |  |  |  |  |  |  |  |
| stmt |  | 4 |  | 5 | 6 | 7 | 8 |  |  |  |  |  |
| asgmnt |  | 9 |  |  |  |  |  |  |  |  |  |  |
| exp |  | 10 | 10 |  |  |  |  |  |  |  |  |  |
| exp\* | 13 |  |  |  |  |  |  | 11 | 12 |  |  | 13 |
| term |  | 14 | 15 |  |  |  |  |  |  |  |  |  |
| if |  |  |  | 16 |  |  |  |  |  |  |  |  |
| cond |  | 17 | 17 |  |  |  |  |  |  |  |  |  |
| cond\* |  |  |  |  |  |  |  |  |  | 18 | 19 |  |
| print |  |  |  |  | 20 |  |  |  |  |  |  |  |
| goto |  |  |  |  |  | 21 |  |  |  |  |  |  |
| stop |  |  |  |  |  |  | 22 |  |  |  |  |  |

Part Lexical Scanner

ใช้ภาษา python   
เปิดไฟล์ input  
อ่านทีละ line   
strip หัวท้ายของ line ออก  
แล้ว split ด้วย white space   
จะทำให้ได้ token แต่ละตัว

Part Parser

ทำ parsing table และ next set (follow set) ด้วย dict   
ถ้าไม่พบกฎใน parsing table จะมี exception ‘rule not defined’  
สร้าง set ของ alphabet(A-Z,a-z)  
สร้าง set ของ terminals(+,-,IF,<,=,PRINT,GOTO,STOP,EOF)  
generate bcode จาก list ของ token  
ถ้า token ไม่ใช่ terminal ของ top ของ stack จะไปทำการหา rule ที่เหมาะสมและ push ลง stack  
ถ้า token match จะทำการ pop ออก

https://github.com/mawinw/Compiler