Retro Basic Conmplier

**Grammar :**

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 | empty  
print := PRINT id  
goto := GOTO line\_num  
stop := STOP

**First set, Follow set :**

|  |  |  |
| --- | --- | --- |
| **Non-terminal** | **First set** | **Follow set** |
| pgm | line\_num, EOF | EOF |
| line | line\_num | line\_num, EOF |
| stmt | id, IF, PRINT, GOTO, STOP | line\_num, EOF |
| asgmnt | id | line\_num, EOF |
| exp | id, const | line\_num, EOF |
| exp’ | +, -, empty | line\_num, EOF |
| term | id, const | line\_num, EOF |
| if | IF | line\_num, EOF |
| cond | id, const | line\_num |
| cond’ | <, = | line\_num |
| print | PRINT | line\_num, EOF |
| goto | GOTO | line\_num, EOF |
| stop | STOP | line\_num, EOF |

**Parsing table :**

1. pgm := line pgm 9. asgmnt := id = exp 17. cond := term cond’

2. pgm := EOF 10. exp := term exp’ 18. cond’ := < term

3. line := line\_num stmt 11. exp’ := + term 19. cond’ := = term

4. stmt := asgmnt 12. exp’ := - term 20. print := PRINT id

5. stmt := if 13. exp’ := empty 21. goto := GOTO line\_num

6. stmt := print 14. term := id 22. stop := STOP

7. stmt := goto 15. term := const

8. stmt := stop 16. If := IF cond line\_num

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 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 |  |  |  |  |  |

1. Scanner

tokens = []

file = input file

for line in file:

i=0

tmp = line.strip().split()

while(i<len(tmp)):

if(tmp[i].isdigit() and int(tmp[i])<100): tokens+=[["num<100",int(tmp[i])]]

elif(tmp[i].isdigit() and int(tmp[i])<1000): tokens+=[["num<1000",int(tmp[i])]]

elif(tmp[i] in alphabet\_uppercase and len(tmp[i]) == 1): tokens+=[["id",ord(tmp[i]) - ord('A') + 1]]

elif(tmp[i]=="IF"): tokens+=[["if",0]]

elif(tmp[i]=="GOTO"): tokens+=[["goto",int(tmp[i+1])]]

i+=1

elif(tmp[i]=="PRINT"): tokens+=[["print",0]]

elif(tmp[i]=="STOP"): tokens+=[["stop",0]]

elif(tmp[i]=="+"): tokens+=[["+",1]]

elif(tmp[i]=="-"): tokens+=[["-",2]]

elif(tmp[i]=="<"): tokens+=[["<",3]]

elif(tmp[i]=="="): tokens+=[["=",4]]

else: print("ERROR") ; exit(0)

i+=1

for i in range(0,len(tokens)):

if(tokens[i][0]=="if"):

tokens[i+4][0]="goto"

tokens.append(["EOF"])

2. Parser

parsing\_table = parsing table that derive from grammar

terminal = terminal symbols in grammar

nonterminal = non terminal symbols in grammar

stack = [] ; point=0 ; output=[]

stack.push("pgm")

while(1):

top=stack[len(stack)-1]

if(top in terminal):

if(tokens[point][0]==top):

output.append(stack.pop())

output.append(tokens[point][1])

point+=1

elif(top=="line\_num" and tokens[point][0]=="goto"):

stack.pop()

output.append("GOTO")

output.append(tokens[point][1])

point+=1

elif(top=="line\_num" and tokens[point][0] in ["num<100","num<1000"]):

tokens[point][0]="line\_num"

elif(top=="const" and tokens[point][0] == "num<100"): tokens[point][0]="const"

elif(top in nonterminal):

if(top in ["exp","term","cond"] and tokens[point][0]=="num<100"): tokens[point][0]="const"

elif(top in ["pgm","line","exp'"] and tokens[point][0] in ["num<100","num<1000"]):

tokens[point][0]="line\_num"

elif(tokens[point][0]=="if"): tokens[point][0]="IF"

elif(tokens[point][0]=="print"): tokens[point][0]="PRINT"

elif(tokens[point][0]=="goto"): tokens[point][0]="GOTO"

elif(tokens[point][0]=="stop"): tokens[point][0]="STOP"

key=(top,tokens[point][0])

if(key in parsing\_table):

if(parsing\_table[key]=="empty"): stack.pop()

else: stack.pop()

for i in range(len(parsing\_table[key])-1,-1,-1): stack.append(parsing\_table[key][i])

else: output="ERROR" ; break

elif(top=="EOF"): print("ACCEPT") ; break

else: output="ERROR" ; break

3. B-code

for i in range(0,len(output),1):

if(output[i] in terminal):

output[i]=terminal[output[i]]

output.append(0)

file = output file

file.write(" ".join([str(e) for e in output]))

4. Working Code

https://github.com/sunsingsong/Retro-Basic