Compiler Design Course Project

Yasmeen Yasser Mohamed
Yousef Mohamed Azmi
Hager Mounir
Yousef Magdy Assad
Nizar Wailed

Project 2.1

Implement a functioning parser for a simple programming language (having at least 2 statements). The parser has to draw the output parse tree (s) in a suitable visual form (As described in lectures).

Selected statements

- While loop statement.
- If-else statement.

programming language

- Python

project description!

Aim:

We have designed predictive parser LL (1)
That will parse any if-else statement or while statement or both according to the following CFG, then draw the parse tree of this statements.

Input:

the input of the project will enter in a source file (txt file)

Output:

the output from the project will be the parse tree if the parsing prosses done successfully, but if not the program will give you error message.

The CFG(context free grammar):

stmts→ stmt stmts | ε

stmt → assign_stmt | while_stmt|if_else_stmt|**{**stmts**}**

assign_stmt → **id =** exp **;**

while_stmt → **while (** cond **) {** stmts **}**

if_else_stmt → **if (** cond **)** stmt opt_stmt

opt_stmt → **else** stmt | ε

cond → **id op** exp

op→> | < | >= | <= | =|…….

exp → term R

R → **+** term R | **-** term R | ε

term→ factor R1

R1 → ***** factor R1 | **/** factor R1 | ε

factor → **id** | digits | **(** exp **)**

digits → **digit** digits | ε

id → a|……|z|A|……|Z

digit → 0|1|…..|9

The first of each statement in the CFG:

Statement	First
stmts	{‘id’ , ‘while’ , ‘if’ , ‘{‘ , ε }
stmt	{‘ id’ , ‘while’ , ‘if’ , ‘{‘ }
assign_stmt	{‘ id’}
while_stmt	{‘while’}
if_else_stmt	{‘if’}
opt_stmt	{‘else’ , ε }
cond	{‘ id’}
exp	{‘ id’ , ‘digit’ , ‘(’}
R	{‘+’ , ‘-’ , ε }
term	{‘ id’ , ‘digit’ , ‘(’}
R1	{‘*’ , ‘/’ , ε }
factor	{‘ id’ , ‘digit’ , ‘(’}
digits	{‘ digit’}

Predictive Parser Code:

```
procedure match (t:token);
begin
if (lookahead == t) then
Lookahead := nextToken();
else
error();
end;
```

```
procedure error();
begin
Printing message syntax error;
exit;
end;
```

```
procedure stmt ();
begin
if (lookahead == ‘id’) then
assign_stmt();
else if (lookahead == ‘if’) then
if_else_stmt ();
else if (lookahead == ‘while’) then
while_stmt();
else if (lookahead == ‘{’) then
match(‘{’);  stmts(); match(‘}’);
else
error();
end;
```

```
procedure stmts();
begin
if(lookahead == ‘id’|’while’|’if’|’{’)then
stmt(); stmts();
```

```
else
return;
end;
```

```
procedure assign_stmt ();
begin
if(lookahead == ‘id’)then
match(‘id’); match (‘=‘); exp(); match(‘;’);
else
error();
end;
```

```
procedure while_stmt ();
begin
if(lookahead == ‘while’)then
match(‘while’); match (‘(‘); cond(); match(‘)’); match(‘{’);
stmts(); match(‘}’);
else
error();
end;
```

```
procedure **if_else_stmt**();

begin

if(lookahead == ‘if’)then

match(‘if’); match (‘(‘); cond(); match(‘)’);stmt();
opt_stmt();

else

error();

end;
```

```
procedure **opt_stmt**();

begin

if(lookahead == ‘else’)then

match(‘else’);stmt();

else

return;

end;
```

```
procedure cond ();
begin
if (lookahead = ='id') then
match('id'); op(); exp();
else
error();
end;
```

```
procedure op ();
begin
if (lookahead == 'relation') then
match('relation');
else
error();
end;
```

```
procedure exp();
begin
if (lookahead = ='id'|'digit'|'(') then
term();R();
```

```
else
error();
end;
```

```
procedure R();
begin
if (lookahead == '+') then
match('+'); term(); R();
else if (lookahead == '-') then
match('-'); term(); R();
else
return;
end;
```

```
procedure term();
begin
if (lookahead = ='id'|'digit'|'(') then
factor();R1();
else
error();end;
```

```
procedure R1();
begin
if (lookahead == ‘*’) then
match(‘*’); factor(); R1();
else if (lookahead == ‘/’) then
match(‘/’); factor(); R1();
else
return;
end;
```

```
procedure factor();
begin
if (lookahead == ‘digit’) then
digits();
else if (lookahead == ‘id’) then
match(‘id);
else if (lookahead == ‘(’) then
match(‘(’); exp(); match(‘)’);
```

```
else
error();
end;
```

```
procedure digits();
begain
if(lookahead == ‘digit’)then
match(‘digit’); digits();
else
return;
end;
```