

## The Parsing Problem

- Given a string of symbols in a language
   (tokens) and a grammar for that language =>
   construct the parse tree or report that the
   sentence is syntactically incorrect.
- Two ways to do this:
  - Top-Down (recursive descending parser).
  - Buttom-Up. (We don't focus on this).

## Top-Down Approach

- Uses recursive procedures to model the parse tree.
- Beginning with the start symbol, for every non-terminal (syntactic class) a procedure which parses that syntactic class is created.

```
oprogram> ::= <block> "." .
<block> ::= <const-decl> <var-decl>  proc-decl> <statement> .
<const-decl> ::= const <const-assignment-list> ";"
<const-assignment-list> ::= <ident> = <number>
                        <const-assignment-list> "," <ident> "=" <number> .
<var-decl> ::= "var" <ident-list> ";"
<ident-list> ::= <ident> | <ident-list> "," <ident> .
<statement> ::= <ident> ":=" <expression>
               "call" <ident>
              "begin" <statement-list> "end"
              "if" <condition> "then" <statement>
              "while" <condition> "do" <statement>
               ε.
<statement-list> ::= <statement> | <statement-list> ";" <statement> .
```

```
<condition> ::= "odd" <expression> | <expression> <relation> <expression> .
<relation> ::= "=" | "<>" | "<" | ">" | "<=" | ">=" .
<expression> ::= <term> | <adding-operator> <term>
             <expression> <adding-operator> <term>
<adding-operator> ::= "+" | "-"
                 ::= <factor> | <term> <multiplying-operator> <factor>
<term>
<multiplying-operator> ::= "*" | "/"
                       ::= <ident> | <number> | ( <expression> )
<factor>
```



# PL/0 Grammar

#### **Terminals**

```
const, var,
procedure, call,
begin, end, if,
then, while, do,
odd
<>> <> > <= >= +
- * / =
, ; e
```

#### **Non-Terminals**

We must implement a procedure for each one of this non-terminals.



## In this parser we use:

- **TOKEN** —a global variable that stores the current token to analyze.
- **GET\_TOKEN()** a procedure that takes the next token in the string and stores it in TOKEN.
- ENTER(type, name, params) a procedure that stores a new symbol into the Symbol Table.
- **ERROR()** a procedure that stops parsing, and shows an error message.



## program> Procedure



end;

### <blook> Procedure



### <const-decl> Procedure

```
<const-decl> ::= const <const-assignment-list> ; | e
<const-assignment-list> ::= <ident> = <number>
                              | <const-assignment-list> , <ident> = <number>
procedure CONST-DECL;
begin
   repeat
    GET TOKEN;
     if TOKEN <> IDENT then ERROR (missing identifier);
    GET TOKEN;
     if TOKEN <> "=" then ERROR (identifier should be followed by =);
    GET TOKEN;
     if TOKEN <> NUMBER then ERROR (= should be followed by number);
     ENTER(constant, ident, number);
     GET TOKEN;
   until TOKEN <> ",";
   if TOKEN <> ";" then ERROR (declaration must end with ;);
   GET TOKEN;
end;
```



### <var-decl> Procedure

```
<var-decl> ::= var <ident-list> ; | e
<ident-list> ::= <ident> | <ident-list> , <ident>
procedure VAR-DECL;
begin
   repeat
     GET TOKEN;
     if TOKEN <> IDENT then ERROR (missing identifier);
     GET TOKEN;
     ENTER(variable, ident, level);
    until TOKEN <> ",";
    if TOKEN <> ";" then ERROR (declaration must end with ;);
    GET TOKEN;
end;
```



## proc-decl> Procedure

```
coc-decl> ::= coc-decl> procedure <ident> ; <block> ; | e
procedure PROC-DECL;
begin
while TOKEN = "procedure" do begin
   GET TOKEN;
   if TOKEN <> IDENT then ERROR (missing procedure declaration);
   ENTER(procedure, ident);
   GET TOKEN;
   if TOKEN <> ";" then ERROR (procedure declaration must end with ;);
   GET TOKEN;
   BLOCK(level+1);
   if TOKEN <> ";" then ERROR (no ; at the end of block);
   GET TOKEN;
end;
end;
```



```
<statement> ::= <ident> := <expression> | call <ident>
        | begin <statement-list> end | if <condition> then <statement>
        | while <condition> do <statement> | e
<statement-list> ::= <statement> | <statement-list> ; <statement>
procedure STATEMENT;
begin
   if TOKEN = IDENT then begin
    GET TOKEN();
    If TOKEN <> ":=" then ERROR (:= missing in statement);
    GET TOKEN();
    EXPRESSION();
   end
   else if TOKEN = "call" then begin
    GET TOKEN();
    if TOKEN <> IDENT then ERROR (missing identifier);
    GET TOKEN();
   end
```



```
<statement> ::= <ident> := <expression> | call <ident>
        | begin <statement-list> end | if <condition> then <statement>
        | while <condition> do <statement> | e
<statement-list> ::= <statement> | <statement-list> ; <statement>
procedure STATEMENT;
   else if TOKEN = "begin" then begin
    GET TOKEN();
    STATEMENT();
    while TOKEN = ";" do begin
         GET TOKEN();
         STATEMENT();
    end;
    if TOKEN <> "end" then ERROR (begin must be closed with end);
    GET TOKEN();
   end;
```



```
<statement> ::= <ident> := <expression> | call <ident>
        | begin <statement-list> end | if <condition> then <statement>
        | while <condition> do <statement> | e
<statement-list> ::= <statement> | <statement-list> ; <statement>
procedure STATEMENT;
   else if TOKEN = "if" then begin
    GET TOKEN();
    CONDITION();
    if TOKEN <> "then" then ERROR (if condition must be followed by then);
    GET TOKEN();
    STATEMENT();
   end;
```



```
<statement> ::= <ident> := <expression> | call <ident>
        | begin <statement-list> end | if <condition> then <statement>
        | while <condition> do <statement> | e
<statement-list> ::= <statement> | <statement-list> ; <statement>
procedure STATEMENT;
   else if TOKEN = "while" then begin
    GET TOKEN();
    CONDITION();
    if TOKEN <> "do" then ERROR (while condition must be followed by do);
    GET TOKEN();
    STATEMENT();
   end
end;
```



### <condition> Procedure

```
<condition> ::= odd <expression> | <expression> <relation> <expression>
procedure CONDITION;
begin
   if TOKEN = "odd" then begin
     GET_TOKEN();
    EXPRESSION();
   else begin
     EXPRESSION();
     if TOKEN <> RELATION then ERROR (relational operator missing in conditional statement);
     GET_TOKEN();
     EXPRESSION();
   end
end;
```



# <expression> Procedure



## <term> Procedure

```
<term> ::= <factor> | <term> <multiplying-operator> <factor>
procedure TERM;
begin
    FACTOR();
    while TOKEN = MULTIPLYING_OPERATOR do begin
    GET_TOKEN();
    FACTOR();
    end
end;
```



## <factor> Procedure

```
<factor> ::= <ident> | <number> | ( <expression> )
procedure FACTOR;
begin
   if TOKEN = IDENTIFIER then
     GET TOKEN();
   else if TOKEN = NUMBER then
     GET TOKEN();
   else if TOKEN = "(" then begin
     GET TOKEN();
     EXPRESSION();
     if TOKEN <> ")" then ERROR( left ( has not been closed );
     GET TOKEN();
   end
   else ERROR (identifier, ( or number expected);
end;
```



```
const m = 8;
var a, b, c;
```

```
var a, b, c;
procedure ratio;
var x, y;
begin
    x = a; y = b;
    if b > a then begin
     x = b;
     y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
procedure PROGRAM;
begin

GET_TOKEN();
BLOCK();
if TOKEN <> "." then ERROR (No
Period at end of file)
end;
```



begin

end;

begin

end.

x = a; y = b;

x = b;y = a;

c = x / y;

a = m;

b = 4;

call ratio;

end

if b > a then begin

# Small Example

```
m = 8;

var a, b, c;

procedure ratio;

var x, y;

TOKEN= const Symbol Table program()

program()
```

```
procedure PROGRAM;
begin
    GET_TOKEN();
    BLOCK();
    if TOKEN <> "." then ERROR (No
Period at end of file)
end;
```



Recursion stack

```
m = 8;
var a, b, c;
procedure ratio;
var x, y;
begin
    x = a; y = b;
    if b > a then begin
     x = b;
     y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
```

end.

```
TOKEN= const Symbol Table program() block()
```

```
procedure BLOCK;
begin

if TOKEN = "const" then CONST-DECL();
if TOKEN = "var" then VAR-DECL();
if TOKEN = "procedure" then PROC-DECL();
STATEMENT;
end;
```



#### Recursion stack

```
m = 8;
var a, b, c;
procedure ratio;
var x, y;
begin
    x = a; y = b;
    if b > a then begin
     x = b;
     y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
```

end.

```
procedure CONST-DECL;
begin
     repeat
     → GET TOKEN();
          if TOKEN <> IDENT then ERROR ();
          GET TOKEN();
          if TOKEN <> "=" then ERROR ();
          GET TOKEN();
          if TOKEN <> NUMBER then ERROR);
          ENTER(constant, ident, number);
          GET TOKEN();
     until TOKEN <> ",";
     if TOKEN <> ";" then ERROR ();
     GET TOKEN;
end;
```



```
= 8:
var a, b, c;
procedure ratio;
var x, y;
begin
    x = a; y = b;
    if b > a then begin
     x = b;
     y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4:
    call ratio;
end.
```

```
TOKEN= m Symbol Table program() block() const-decl()
```

```
procedure CONST-DECL;
begin
     repeat
          GET TOKEN();
          if TOKEN <> IDENT then ERROR ();
       →GET TOKEN();
          if TOKEN <> "=" then ERROR ();
          GET TOKEN();
          if TOKEN <> NUMBER then ERROR);
          ENTER(constant, ident, number);
          GET TOKEN();
     until TOKEN <> ",";
     if TOKEN <> ";" then ERROR ();
     GET TOKEN;
end;
```



```
8;
var a, b, c;
procedure ratio;
var x, y;
begin
    x = a; y = b;
    if b > a then begin
     x = b;
     y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4:
    call ratio;
end.
```

```
TOKEN= Symbol Table program() block() const-decl()
```

```
procedure CONST-DECL;
begin
     repeat
          GET TOKEN();
          if TOKEN <> IDENT then ERROR ();
          GET TOKEN();
          if TOKEN <> "=" then ERROR ();
        →GET TOKEN();
          if TOKEN <> NUMBER then ERROR);
          ENTER(constant, ident, number);
          GET TOKEN();
     until TOKEN <> ",";
     if TOKEN <> ";" then ERROR ();
     GET TOKEN;
end;
```



```
var a, b, c;
procedure ratio;
var x, y;
begin
    x = a; y = b;
    if b > a then begin
     x = b;
     y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
TOKEN= 8 Symbol Table program() block() const-decl()
```

```
procedure CONST-DECL;
begin
     repeat
          GET TOKEN();
          if TOKEN <> IDENT then ERROR ();
          GET TOKEN();
          if TOKEN <> "=" then ERROR ();
          GET TOKEN();
          if TOKEN <> NUMBER then ERROR);
        ENTER(constant, ident, number);
          GET TOKEN();
     until TOKEN <> ",";
     if TOKEN <> ";" then ERROR ();
     GET TOKEN;
end;
```



```
var a, b, c;
procedure ratio;
var x, y;
begin
    x = a; y = b;
    if b > a then begin
     x = b;
     y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
TOKEN= 8 Symbol Table program()
m=8; block()
const-decl()
```

```
procedure CONST-DECL;
begin
     repeat
          GET TOKEN();
          if TOKEN <> IDENT then ERROR ();
          GET TOKEN();
          if TOKEN <> "=" then ERROR ();
          GET TOKEN();
          if TOKEN <> NUMBER then ERROR);
          ENTER(constant, ident, number);
        →GET TOKEN();
     until TOKEN <> ",";
     if TOKEN <> ";" then ERROR ();
     GET TOKEN;
end;
```



```
TOKEN=; Symbol Table program()
m=8; block()
const-decl()
```

```
var a, b, c;
procedure ratio;
var x, y;
begin
    x = a; y = b;
    if b > a then begin
     x = b;
     y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4:
    call ratio;
end.
```

```
procedure CONST-DECL;
begin
     repeat
          GET TOKEN();
          if TOKEN <> IDENT then ERROR ();
          GET TOKEN();
          if TOKEN <> "=" then ERROR ();
          GET TOKEN();
          if TOKEN <> NUMBER then ERROR);
          ENTER(constant, ident, number);
          GET TOKEN();
     until TOKEN <> ",";
     if TOKEN <> ";" then ERROR ();
     GET TOKEN();
end;
```



```
a, b, c;
procedure ratio;

TOKEN= var

Symbol Table
program()
block()
const-decl()
```

```
var x, y;
begin
    x = a; y = b;
                               procedure CONST-DECL;
    if b > a then begin
                               begin
     x = b;
                                    repeat
     y = a;
                                         GET TOKEN();
    end
                                         if TOKEN <> IDENT then ERROR ();
    c = x / y;
                                         GET TOKEN();
end;
                                         if TOKEN <> "=" then ERROR ();
begin
                                         GET TOKEN();
    a = m;
                                         if TOKEN <> NUMBER then ERROR);
    b = 4;
                                         ENTER(constant, ident, number);
    call ratio;
                                         GET TOKEN();
end.
                                    until TOKEN <> ",";
                                    if TOKEN <> ";" then ERROR ();
                                    GET TOKEN();
                              end;
```



```
TOKEN= var Symbol Table program() block()
```

```
a, b, c;
procedure ratio;
var x, y;
begin
    x = a; y = b;
                               procedure BLOCK;
    if b > a then begin
                               begin
     x = b;
                                    if TOKEN = "const" then CONST-DECL();
     y = a;
                                   if TOKEN = "var" then VAR-DECL();
    end
    c = x / y;
                                    if TOKEN = "procedure" then PROC-DECL();
end;
                                   STATEMENT;
begin
                               end;
    a = m;
    b = 4;
    call ratio;
end.
```



```
TOKEN= var

Symbol Table
program()

block()
var x, y;

procedure ratio;

procedure ratio;
```

```
var x, y;
begin
    x = a; y = b;
                               procedure VAR-DECL;
    if b > a then begin
                               begin
     x = b;
                                     repeat
     y = a;
                                    → GET TOKEN();
    end
                                          if TOKEN <> IDENT then ERROR ();
    c = x / y;
                                          GET TOKEN();
end;
                                          ENTER(variable, ident, level);
begin
                                     until TOKEN <> ",";
    a = m;
                                     if TOKEN <> ";" then ERROR ();
    b = 4;
                                     GET TOKEN();
    call ratio;
                               end;
end.
```



```
var x, y;
begin
    x = a; y = b;
                               procedure VAR-DECL;
    if b > a then begin
                               begin
     x = b;
                                     repeat
     y = a;
                                          GET TOKEN();
    end
                                          if TOKEN <> IDENT then ERROR ();
    c = x / y;
                                     → GET TOKEN();
end;
                                          ENTER(variable, ident, level);
begin
                                     until TOKEN <> ",";
    a = m;
                                     if TOKEN <> ";" then ERROR ();
    b = 4;
                                     GET TOKEN();
    call ratio;
                               end;
end.
```



```
TOKEN=, Symbol Table program()
m=8; block()
var-decl()
```

```
b, c;
procedure ratio;
var x, y;
begin
    x = a; y = b;
                               procedure VAR-DECL;
    if b > a then begin
                               begin
     x = b;
                                     repeat
     y = a;
                                           GET TOKEN();
    end
                                           if TOKEN <> IDENT then ERROR ();
    c = x / y;
                                           GET TOKEN();
end;
                                         ENTER(variable, ident, level);
begin
                                     until TOKEN <> ",";
    a = m;
                                     if TOKEN <> ";" then ERROR ();
    b = 4;
                                     GET TOKEN();
    call ratio;
                               end;
end.
```



```
TOKEN=, Symbol Table program()
m=8; a; block()
var-decl()
```

```
b, c;
procedure ratio;
var x, y;
begin
    x = a; y = b;
    if b > a then begin
     x = b;
     y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
procedure VAR-DECL;
begin
    repeat
    GET_TOKEN();
    if TOKEN <> IDENT then ERROR ();
    GET_TOKEN();
    ENTER(variable, ident, level);
    until TOKEN <> ",";
    if TOKEN <> ";" then ERROR ();
    GET_TOKEN();
end;
```



```
TOKEN= b

Symbol Table

program()

block()

var-decl()
```

```
procedure ratio;
var x, y;
begin
    x = a; y = b;
                               procedure VAR-DECL;
    if b > a then begin
                               begin
     x = b;
                                     repeat
     y = a;
                                          GET TOKEN();
    end
                                          if TOKEN <> IDENT then ERROR ();
    c = x / y;
                                     → GET TOKEN();
end;
                                          ENTER(variable, ident, level);
begin
                                     until TOKEN <> ",";
    a = m;
                                     if TOKEN <> ";" then ERROR ();
    b = 4;
                                     GET TOKEN();
    call ratio;
                               end;
end.
```



```
TOKEN=, Symbol Table program()
m=8; a; block()
var-decl()
```

```
С;
procedure ratio;
var x, y;
begin
    x = a; y = b;
                               procedure VAR-DECL;
    if b > a then begin
                               begin
     x = b;
                                     repeat
     y = a;
                                           GET TOKEN();
    end
                                           if TOKEN <> IDENT then ERROR ();
    c = x / y;
                                           GET TOKEN();
end;
                                         ENTER(variable, ident, level);
begin
                                     until TOKEN <> ",";
    a = m;
                                     if TOKEN <> ";" then ERROR ();
    b = 4;
                                     GET TOKEN();
    call ratio;
                               end;
end.
```



```
TOKEN=, Symbol Table program()
m=8; a; b;
block()
var-decl()
```

```
С;
procedure ratio;
var x, y;
begin
    x = a; y = b;
    if b > a then begin
     x = b;
     y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
procedure VAR-DECL;
begin
    repeat
    GET_TOKEN();
    if TOKEN <> IDENT then ERROR ();
    GET_TOKEN();
    ENTER(variable, ident, level);
    until TOKEN <> ",";
    if TOKEN <> ";" then ERROR ();
    GET_TOKEN();
end;
```



```
;
procedure ratio;

TOKEN= c

Symbol Table
program()
block()
var-decl()

var x, y;
```

```
procedure ratio;
var x, y;
begin
    x = a; y = b;
                               procedure VAR-DECL;
    if b > a then begin
                               begin
     x = b;
                                     repeat
     y = a;
                                          GET TOKEN();
    end
                                          if TOKEN <> IDENT then ERROR ();
    c = x / y;
                                     → GET TOKEN();
end;
                                          ENTER(variable, ident, level);
begin
                                     until TOKEN <> ",";
    a = m;
                                     if TOKEN <> ";" then ERROR ();
    b = 4;
                                     GET TOKEN();
    call ratio;
                               end;
end.
```



```
TOKEN=;

Symbol Table

m=8; a; b;

program()
block()
var-decl()
```

```
procedure ratio;
var x, y;
begin
    x = a; y = b;
                               procedure VAR-DECL;
    if b > a then begin
                               begin
     x = b;
                                     repeat
     y = a;
                                          GET TOKEN();
    end
                                          if TOKEN <> IDENT then ERROR ();
    c = x / y;
                                          GET TOKEN();
end;
                                         ENTER(variable, ident, level);
begin
                                     until TOKEN <> ",";
    a = m;
                                     if TOKEN <> ";" then ERROR ();
    b = 4;
                                     GET TOKEN();
    call ratio;
                               end;
end.
```



```
TOKEN=;

Symbol Table

m=8; a; b; c;

program()
block()
var-decl()
```

```
procedure ratio;
var x, y;
begin
    x = a; y = b;
                               procedure VAR-DECL;
    if b > a then begin
                               begin
     x = b;
                                     repeat
     y = a;
                                           GET TOKEN();
    end
                                           if TOKEN <> IDENT then ERROR ();
    c = x / y;
                                           GET TOKEN();
end;
                                           ENTER(variable, ident, level);
begin
                                     until TOKEN <> ",";
    a = m;
                                     if TOKEN <> ";" then ERROR ();
    b = 4;
                                    GET TOKEN();
    call ratio;
                               end;
end.
```



```
TOKEN= procedure Symbol Table program()
m=8; a; b; c; program()
block()
var-decl()
```

```
ratio;
var x, y;
begin
    x = a; y = b;
                                procedure VAR-DECL;
    if b > a then begin
                               begin
     x = b;
                                     repeat
     y = a;
                                           GET TOKEN();
    end
                                           if TOKEN <> IDENT then ERROR ();
    c = x / y;
                                           GET TOKEN();
end;
                                           ENTER(variable, ident, level);
begin
                                     until TOKEN <> ",";
    a = m;
                                     if TOKEN <> ";" then ERROR ();
    b = 4;
                                     GET TOKEN();
    call ratio;
                              ≥end;
end.
```



TOKEN= procedure	Symbol Table	program()
	m=8; a; b; c;	block()

```
ratio;
var x, y;
begin
    x = a; y = b;
                               procedure BLOCK;
    if b > a then begin
                               begin
     x = b;
                                   if TOKEN = "const" then CONST-DECL();
     y = a;
                                   if TOKEN = "var" then VAR-DECL();
    end
    c = x / y;
                                   if TOKEN = "procedure" then PROC-DECL();
end;
                                   STATEMENT;
begin
                               end;
    a = m;
    b = 4;
    call ratio;
end.
```



```
TOKEN= procedure Symbol Table program()

m=8; a; b; c; block()
proc-decl()
```

```
ratio;
var x, y;
begin
    x = a; y = b;
    if b > a then begin
     x = b;
      y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
procedure PROC-DECL;
begin
 while TOKEN = "procedure" do begin
→ GET TOKEN();
     if TOKEN <> IDENT then ERROR ();
      ENTER(procedure, ident);
     GET TOKEN();
     if TOKEN <> ";" then ERROR ();
     GET TOKEN();
     BLOCK(level+1);
     if TOKEN <> ";" then ERROR ();
     GET_TOKEN();
 end;
end;
```



```
TOKEN= ratio

Symbol Table
program()
block()
proc-decl()
```

```
var x, y;
begin
    x = a; y = b;
    if b > a then begin
     x = b;
      y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
procedure PROC-DECL;
begin
while TOKEN = "procedure" do begin
     GET TOKEN();
     if TOKEN <> IDENT then ERROR ();
  ENTER(procedure, ident);
     GET TOKEN();
     if TOKEN <> ";" then ERROR ();
     GET TOKEN();
     BLOCK(level+1);
     if TOKEN <> ";" then ERROR ();
     GET_TOKEN();
end;
end;
```



```
TOKEN= ratio

Symbol Table

m=8; a; b; c;
ratio;

program()
block()
proc-decl()
```

```
var x, y;
begin
    x = a; y = b;
    if b > a then begin
     x = b;
      y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
procedure PROC-DECL;
begin
while TOKEN = "procedure" do begin
     GET TOKEN();
     if TOKEN <> IDENT then ERROR ();
     ENTER(procedure, ident);
  GET TOKEN();
     if TOKEN <> ";" then ERROR ();
     GET TOKEN();
     BLOCK(level+1);
     if TOKEN <> ";" then ERROR ();
     GET_TOKEN();
end;
end;
```



```
TOKEN=;

Symbol Table

m=8; a; b; c;
ratio;

program()
block()
proc-decl()
```

```
var x, y;
begin
    x = a; y = b;
    if b > a then begin
    x = b;
    y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
procedure PROC-DECL;
begin
while TOKEN = "procedure" do begin
     GET TOKEN();
     if TOKEN <> IDENT then ERROR ();
     ENTER(procedure, ident);
     GET TOKEN();
     if TOKEN <> ";" then ERROR ();
    GET TOKEN();
     BLOCK(level+1);
     if TOKEN <> ";" then ERROR ();
     GET_TOKEN();
end;
end;
```



TOKEN= var	Symbol Table	program()
	m=8; a; b; c; ratio;	block() proc-decl()

```
x, y;

begin

x = a; y = b;

if b > a then begin

x = b;

y = a;

end

c = x / y;

end;

begin

a = m;

b = 4;

call ratio;

end.
```

```
procedure PROC-DECL;
begin
while TOKEN = "procedure" do begin
     GET TOKEN();
     if TOKEN <> IDENT then ERROR ();
     ENTER(procedure, ident);
     GET TOKEN();
     if TOKEN <> ";" then ERROR ();
     GET TOKEN();
  BLOCK(level+1);
     if TOKEN <> ";" then ERROR ();
     GET_TOKEN();
end;
end;
```



```
TOKEN= var

Symbol Table

m=8; a; b; c;
ratio;

program()
block(1)
proc-decl(1)
block(2)
```

```
x, y;
begin

x = a; y = b;
if b > a then begin

x = b;
y = a;
end
c = x / y;
end;
begin

a = m;
b = 4;
call ratio;
end.
```

```
procedure BLOCK;
begin

if TOKEN = "const" then CONST-DECL();
   if TOKEN = "var" then VAR-DECL();
   if TOKEN = "procedure" then PROC-DECL();
   STATEMENT;
end;
```



```
TOKEN= var

Symbol Table

m=8; a; b; c;
ratio;

program()
block(1)
proc-decl(1)
block(2)
```

```
x, y;
begin
    x = a; y = b;
    if b > a then begin
    x = b;
    y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
procedure BLOCK;
begin
    if TOKEN = "const" then CONST-DECL();
    if TOKEN = "var" then VAR-DECL();
    if TOKEN = "procedure" then PROC-DECL();
    STATEMENT;
end;
```



```
TOKEN= var

Symbol Table

m=8; a; b; c;
ratio;

program()
block(1)
proc-decl(1)
block(2)
var-decl(2)
```

```
x, y;
begin
    x = a; y = b;
    if b > a then begin
    x = b;
    y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
procedure VAR-DECL;
begin
    repeat
    GET_TOKEN;
    if TOKEN <> IDENT then ERROR ();
    GET_TOKEN;
    ENTER(variable, ident, level);
    until TOKEN <> ",";
    if TOKEN <> ";" then ERROR ();
    GET_TOKEN;
end;
```



```
TOKEN= x

Symbol Table

m=8; a; b; c;
ratio;

program()
block(1)
proc-decl(1)
block(2)
var-decl(2)
```

```
, y;
begin
    x = a; y = b;
    if b > a then begin
     x = b;
     y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
procedure VAR-DECL;
begin

repeat

GET_TOKEN;

if TOKEN <> IDENT then ERROR ();

GET_TOKEN;

ENTER(variable, ident, level);

until TOKEN <> ",";

if TOKEN <> ";" then ERROR ();

GET_TOKEN;

end;
```



```
TOKEN=,

Symbol Table

m=8; a; b; c;
ratio;

program()
block(1)
proc-decl(1)
block(2)
var-decl(2)
```

```
у;
begin
    x = a; y = b;
    if b > a then begin
     x = b;
     y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
procedure VAR-DECL;
begin

repeat

GET_TOKEN;

if TOKEN <> IDENT then ERROR ();

GET_TOKEN;

ENTER(variable, ident, level);

until TOKEN <> ",";

if TOKEN <> ";" then ERROR ();

GET_TOKEN;

end;
```



```
TOKEN=,

Symbol Table

m=8; a; b; c;
ratio; x;

program()
block(1)
proc-decl(1)
block(2)
var-decl(2)
```

```
у;
begin
    x = a; y = b;
    if b > a then begin
     x = b;
     y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
procedure VAR-DECL;
begin
    repeat
    GET_TOKEN;
    if TOKEN <> IDENT then ERROR ();
    GET_TOKEN;
    ENTER(variable, ident, level);
    until TOKEN <> ",";
    if TOKEN <> ";" then ERROR ();
    GET_TOKEN;
end;
```



```
TOKEN= y

Symbol Table

m=8; a; b; c;
ratio; x;

program()
block(1)
proc-decl(1)
block(2)
var-decl(2)
```

```
begin
    x = a; y = b;
                               procedure VAR-DECL;
    if b > a then begin
                               begin
     x = b;
                                    repeat
     y = a;
                                          GET TOKEN;
    end
                                          if TOKEN <> IDENT then ERROR ();
    c = x / y;
                                    → GET TOKEN;
end;
                                          ENTER(variable, ident, level);
begin
                                    until TOKEN <> ",";
    a = m;
                                    if TOKEN <> ";" then ERROR ();
    b = 4;
                                    GET TOKEN;
    call ratio;
                               end;
end.
```



```
TOKEN=;

Symbol Table

m=8; a; b; c;
ratio; x;

program()
block(1)
proc-decl(1)
block(2)
var-decl(2)
```

```
begin
    x = a; y = b;
    if b > a then begin
    x = b;
    y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
procedure VAR-DECL;
begin

repeat

GET_TOKEN;

if TOKEN <> IDENT then ERROR ();

GET_TOKEN;

ENTER(variable, ident, level);

until TOKEN <> ",";

if TOKEN <> ";" then ERROR ();

GET_TOKEN;

end;
```



```
TOKEN=;

Symbol Table

m=8; a; b; c;
ratio; x; y;

program()
block(1)
proc-decl(1)
block(2)
var-decl(2)
```

```
begin
    x = a; y = b;
    if b > a then begin
    x = b;
    y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
procedure VAR-DECL;
begin

repeat

GET_TOKEN;

if TOKEN <> IDENT then ERROR ();

GET_TOKEN;

ENTER(variable, ident, level);

until TOKEN <> ",";

if TOKEN <> ";" then ERROR ();

GET_TOKEN;

end;
```



```
TOKEN= begin

Symbol Table

m=8; a; b; c;
ratio; x; y;

program()
block(1)
proc-decl(1)
block(2)
var-decl(2)
```

```
x = a; y = b;
                               procedure VAR-DECL;
    if b > a then begin
                               begin
     x = b;
                                    repeat
     y = a;
                                          GET TOKEN;
    end
                                          if TOKEN <> IDENT then ERROR ();
    c = x / y;
                                          GET TOKEN;
end;
                                          ENTER(variable, ident, level);
begin
                                    until TOKEN <> ",";
    a = m;
                                    if TOKEN <> ";" then ERROR ();
    b = 4;
                                    GET TOKEN;
    call ratio;
                              end:
end.
```



```
TOKEN= begin

Symbol Table

m=8; a; b; c;
ratio; x; y;

program()
block(1)
proc-decl(1)
block(2)
```

```
x = a; y = b;
if b > a then begin
x = b;
y = a;
end
c = x / y;
end;
begin
a = m;
b = 4;
call ratio;
end.
```

```
procedure BLOCK;
begin
    if TOKEN = "const" then CONST-DECL();
    if TOKEN = "var" then VAR-DECL();
    if TOKEN = "procedure" then PROC-DECL();
        STATEMENT;
end;
```



TOKEN= begin	Symbol Table	program()
	m=8; a; b; c; ratio; x; y;	block(1) proc-decl(1) block(2)

```
x = a; y = b;
if b > a then begin
x = b;
y = a;
end
c = x / y;
end;
begin
a = m;
b = 4;
call ratio;
end.
```

```
procedure BLOCK;
begin
    if TOKEN = "const" then CONST-DECL();
    if TOKEN = "var" then VAR-DECL();
    if TOKEN = "procedure" then PROC-DECL();

>STATEMENT;
end;
```



```
TOKEN= begin

Symbol Table

m=8; a; b; c;
ratio; x; y;

program()
block(1)
proc-decl(1)
block(2)
statement(2)
```

```
x = a; y = b;
if b > a then begin
x = b;
y = a;
end
c = x / y;
end;
begin
a = m;
b = 4;
call ratio;
end.
```



### Recursion stack

```
TOKEN= x

Symbol Table

m=8; a; b; c;
ratio; x; y;

program()
block(1)
proc-decl(1)
block(2)
statement(2)
```

```
= a; y = b;

if b > a then begin

x = b;
y = a;
end
c = x / y;
end;
begin
a = m;
b = 4;
call ratio;
end.
```

•••



```
Symbol Table
TOKEN= x
                                          program()
                                          block(1)
                       m=8; a; b; c;
                                          proc-decl(1)
                       ratio; x; y;
                                          block(2)
                                          statement(2)
                                          statement(2)
 procedure STATEMENT;
```

```
= a; y = b;
    if b > a then begin
     x = b;
     y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
begin
     if TOKEN = IDENT then begin
    → GET TOKEN();
          If TOKEN <> ":=" then ERROR ();
          GET_TOKEN();
          EXPRESSION();
     end
```



```
TOKEN==
                      Symbol Table
                                         program()
                                         block(1)
                      m=8; a; b; c;
                                         proc-decl(1)
                      ratio; x; y;
                                         block(2)
                                         statement(2)
                                         statement(2)
 procedure STATEMENT;
```

```
a; y = b;
    if b > a then begin
     x = b;
     y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
begin
     if TOKEN = IDENT then begin
          GET TOKEN();
          If TOKEN <> ":=" then ERROR ();
        →GET_TOKEN();
          EXPRESSION();
     end
```



```
TOKEN= a
                       Symbol Table
                                          program()
                                          block(1)
                       m=8; a; b; c;
                                          proc-decl(1)
                       ratio; x; y;
                                          block(2)
                                          statement(2)
                                          statement(2)
  procedure STATEMENT;
 begin
```

```
; y = b;
    if b > a then begin
     x = b;
     y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
if TOKEN = IDENT then begin
     GET TOKEN();
     If TOKEN <> ":=" then ERROR ();
     GET_TOKEN();
    EXPRESSION();
end
```



```
Symbol Table
TOKEN= a
                                           program()
                                           block(1)
                       m=8; a; b; c;
                                           proc-decl(1)
                       ratio; x; y;
                                           block(2)
                                           statement(2)
                                           statement(2)
                                           expression(2)
  procedure EXPRESSION;
 begin
      if TOKEN = ADDING OPERATOR then GET TOKEN();
  →TERM();
      while TOKEN = ADDING OPERATOR do begin
          GET_TOKEN();
          TERM();
      end
 end;
```

```
; y = b;

if b > a then begin

x = b;

y = a;

end

c = x / y;

end;

begin

a = m;

b = 4;

call ratio;

end.
```



end.

# Small Example

```
Symbol Table
                          TOKEN= a
                                                                        program()
                                                                        block(1)
                                                   m=8; a; b; c;
                                                                        proc-decl(1)
                                                   ratio; x; y;
                                                                        block(2)
                                                                        statement(2)
                                                                        statement(2)
       ; y = b;
   if b > a then begin
                                                                        expression(2)
                            procedure TERM;
    x = b;
                                                                        term(2)
                            begin
    y = a;
                               FACTOR();
   end
                                 while TOKEN = MULTIPLYING_OPERATOR do begin
   c = x / y;
                                      GET TOKEN();
end;
                                      FACTOR();
begin
                                 end
   a = m;
                            end;
   b = 4;
   call ratio;
```

end;



# Small Example

```
Symbol Table
TOKEN= a
                      m=8; a; b; c;
                      ratio; x; y;
```

```
; y = b;
    if b > a then begin
     x = b;
      y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
procedure FACTOR;
begin
     if TOKEN = IDENTIFIER then
    → GET TOKEN();
     else if TOKEN = NUMBER then
          GET TOKEN();
     else if TOKEN = "(" then begin
          GET TOKEN();
          EXPRESSION();
          if TOKEN <> ")" then ERROR );
          GET TOKEN();
     end
     else ERROR ();
```

```
program()
block(1)
proc-decl(1)
block(2)
statement(2)
statement(2)
expression(2)
term(2)
factor(2)
```



```
TOKEN=;

Symbol Table

m=8; a; b; c;
ratio; x; y;

block(
proc-c)
block(
staten
```

```
y = b;
if b > a then begin
x = b;
y = a;
end
c = x / y;
end;
begin
a = m;
b = 4;
call ratio;
end.
```

```
procedure FACTOR;
begin
     if TOKEN = IDENTIFIER then
          GET TOKEN();
     else if TOKEN = NUMBER then
          GET TOKEN();
     else if TOKEN = "(" then begin
          GET TOKEN();
          EXPRESSION();
          if TOKEN <> ")" then ERROR );
          GET TOKEN();
     end
     else ERROR ();
```

⇒end:

```
program()
block(1)
proc-decl(1)
block(2)
statement(2)
statement(2)
expression(2)
term(2)
factor(2)
```



```
TOKEN=;
                                                    Symbol Table
                                                                        program()
                                                                        block(1)
                                                   m=8; a; b; c;
                                                                        proc-decl(1)
                                                   ratio; x; y;
                                                                        block(2)
                                                                        statement(2)
                                                                        statement(2)
        y = b;
   if b > a then begin
                                                                        expression(2)
                            procedure TERM;
    x = b;
                                                                        term(2)
                            begin
    y = a;
                                 FACTOR();
   end
                                 while TOKEN = MULTIPLYING_OPERATOR do begin
   c = x / y;
                                      GET TOKEN();
end;
                                      FACTOR();
begin
                                 end
   a = m;
                           end;
   b = 4;
   call ratio;
end.
```



```
Symbol Table
                          TOKEN=;
                                                                        program()
                                                                        block(1)
                                                   m=8; a; b; c;
                                                                        proc-decl(1)
                                                   ratio; x; y;
                                                                        block(2)
                                                                        statement(2)
        y = b;
                                                                        statement(2)
   if b > a then begin
                                                                        expression(2)
                            procedure EXPRESSION;
    x = b;
                            begin
    y = a;
                                 if TOKEN = ADDING OPERATOR then GET TOKEN();
   end
                                 TERM();
   c = x / y;
                                 while TOKEN = ADDING OPERATOR do begin
end;
                                      GET_TOKEN();
begin
                                      TERM();
   a = m;
                                 end
   b = 4:
                           end;
   call ratio;
end.
```



```
TOKEN=;
                       Symbol Table
                                           program()
                                           block(1)
                       m=8; a; b; c;
                                           proc-decl(1)
                       ratio; x; y;
                                           block(2)
                                           statement(2)
                                           statement(2)
  procedure STATEMENT;
 begin
      if TOKEN = IDENT then begin
```

```
y = b;
    if b > a then begin
     x = b;
     y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
GET TOKEN();
     If TOKEN <> ":=" then ERROR ();
     GET_TOKEN();
     EXPRESSION();
end
```



```
TOKEN=;

Symbol Table

m=8; a; b; c;
ratio; x; y;

program()
block(1)
proc-decl(1)
block(2)
statement(2)
```

```
y = b;
if b > a then begin
x = b;
y = a;
end
c = x / y;
end;
begin
a = m;
b = 4;
call ratio;
end.
```



```
TOKEN=;

Symbol Table

m=8; a; b; c;
ratio; x; y;

program()
block(1)
proc-decl(1)
block(2)
statement(2)
```

```
y = b;
if b > a then begin
x = b;
y = a;
end
c = x / y;
end;
begin
a = m;
b = 4;
call ratio;
end.
```

```
procedure STATEMENT;
...

else if TOKEN = "begin" then begin
    GET TOKEN();
    STATEMENT();
    while TOKEN = ";" do begin
    GET_TOKEN();
    STATEMENT();
    end;
    if TOKEN <> "end" then ERROR ();
    GET_TOKEN();
    end;
```



#### Recursion stack

```
TOKEN= y

Symbol Table

m=8; a; b; c;
ratio; x; y;

program()
block(1)
proc-decl(1)
block(2)
statement(2)
```

```
= b;

if b > a then begin

x = b;

y = a;

end

c = x / y;

end;

begin

a = m;

b = 4;

call ratio;

end.
```

•••



```
TOKEN= y
                       Symbol Table
                                          program()
                                          block(1)
                       m=8; a; b; c;
                                          proc-decl(1)
                       ratio; x; y;
                                          block(2)
                                          statement(2)
                                          statement(2)
  procedure STATEMENT;
```

```
= b;
    if b > a then begin
     x = b;
     y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
begin
     if TOKEN = IDENT then begin
    → GET TOKEN();
          If TOKEN <> ":=" then ERROR ();
          GET_TOKEN();
          EXPRESSION();
     end
```



```
Symbol Table
TOKEN==
                                         program()
                                         block(1)
                      m=8; a; b; c;
                                         proc-decl(1)
                      ratio; x; y;
                                         block(2)
                                         statement(2)
                                         statement(2)
 procedure STATEMENT;
```

```
b;
    if b > a then begin
     x = b;
     y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
begin
     if TOKEN = IDENT then begin
          GET TOKEN();
          If TOKEN <> ":=" then ERROR ();
        →GET_TOKEN();
          EXPRESSION();
     end
```

if b > a then begin

x = b;

y = a;

c = x / y;

a = m; b = 4;

call ratio;

end

end;

begin

end.



## Small Example

### Recursion stack

program()

proc-decl(1)

statement(2)

statement(2)

block(1)

block(2)

```
Symbol Table
TOKEN= b
                        m=8; a; b; c;
                        ratio; x; y;
  procedure STATEMENT;
 begin
      if TOKEN = IDENT then begin
           GET TOKEN();
           If TOKEN <> ":=" then ERROR ();
           GET_TOKEN();
           EXPRESSION();
      end
```



```
TOKEN= b
                       Symbol Table
                                           program()
                                           block(1)
                       m=8; a; b; c;
                                           proc-decl(1)
                       ratio; x; y;
                                           block(2)
                                           statement(2)
                                           statement(2)
                                           expression(2)
  procedure EXPRESSION;
 begin
      if TOKEN = ADDING OPERATOR then GET TOKEN();
  →TERM();
      while TOKEN = ADDING OPERATOR do begin
           GET_TOKEN();
           TERM();
      end
 end;
```

```
;
    if b > a then begin
        x = b;
        y = a;
    end
        c = x / y;
end;
begin
        a = m;
        b = 4;
        call ratio;
end.
```



end;

begin

end.

## Small Example

```
Symbol Table
                      TOKEN= b
                                                                    program()
                                                                    block(1)
                                               m=8; a; b; c;
                                                                    proc-decl(1)
                                               ratio; x; y;
                                                                    block(2)
                                                                    statement(2)
                                                                    statement(2)
if b > a then begin
                                                                    expression(2)
                        procedure TERM;
 x = b;
                                                                    term(2)
                        begin
 y = a;
                           FACTOR();
end
                             while TOKEN = MULTIPLYING_OPERATOR do begin
c = x / y;
                                  GET TOKEN();
                                  FACTOR();
                             end
a = m;
                        end;
b = 4;
call ratio;
```



```
TOKEN= b

Symbol Table

m=8; a; b; c;
ratio; x; y;
```

```
if b > a then begin
                              procedure FACTOR;
     x = b;
                              begin
     y = a;
                                    if TOKEN = IDENTIFIER then
    end
                                   → GET TOKEN();
    c = x / y;
                                    else if TOKEN = NUMBER then
end;
                                         GET TOKEN();
begin
                                    else if TOKEN = "(" then begin
    a = m;
                                         GET TOKEN();
    b = 4;
                                         EXPRESSION();
    call ratio;
                                         if TOKEN <> ")" then ERROR );
end.
                                         GET TOKEN();
                                    end
                                    else ERROR ();
```

end;

```
program()
block(1)
proc-decl(1)
block(2)
statement(2)
statement(2)
expression(2)
term(2)
factor(2)
```



```
TOKEN=;

Symbol Table

m=8; a; b; c;
ratio; x; y;
```

```
if b > a then begin
                               procedure FACTOR;
     x = b;
                               begin
     y = a;
                                    if TOKEN = IDENTIFIER then
   end
                                         GET TOKEN();
    c = x / y;
                                    else if TOKEN = NUMBER then
end;
                                         GET TOKEN();
begin
                                    else if TOKEN = "(" then begin
    a = m;
                                         GET TOKEN();
    b = 4;
                                         EXPRESSION();
   call ratio;
                                          if TOKEN <> ")" then ERROR );
end.
                                         GET TOKEN();
```

end

⇒end:

else ERROR ();

```
program()
block(1)
proc-decl(1)
block(2)
statement(2)
statement(2)
expression(2)
term(2)
factor(2)
```



x = b;

y = a;

c = x / y;

a = m;

b = 4;

call ratio;

end

end;

begin

end.

## Small Example

```
TOKEN=;
                                             Symbol Table
                                                                 program()
                                                                 block(1)
                                             m=8; a; b; c;
                                                                 proc-decl(1)
                                             ratio; x; y;
                                                                 block(2)
                                                                 statement(2)
                                                                 statement(2)
if b > a then begin
                                                                 expression(2)
                       procedure TERM;
                                                                 term(2)
                       begin
                            FACTOR();
                            while TOKEN = MULTIPLYING_OPERATOR do begin
                                 GET TOKEN();
                                 FACTOR();
                            end
                      end;
```



```
TOKEN=;

Symbol Table

m=8; a; b; c;
ratio; x; y;

procedure EXPRESSION;

program()
block(1)
proc-decl(1)
block(2)
statement(2)
expression(2)
```

```
if b > a then begin
     x = b;
                              begin
     y = a;
                                   if TOKEN = ADDING OPERATOR then GET TOKEN();
   end
                                   TERM();
   c = x / y;
                                   while TOKEN = ADDING OPERATOR do begin
end;
                                        GET_TOKEN();
begin
                                        TERM();
   a = m;
                                   end
   b = 4:
                             end;
   call ratio;
end.
```



```
TOKEN=;
                      Symbol Table
                                         program()
                                         block(1)
                      m=8; a; b; c;
                                         proc-decl(1)
                      ratio; x; y;
                                         block(2)
                                         statement(2)
                                         statement(2)
```

```
if b > a then begin
     x = b;
     y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
procedure STATEMENT;
begin
     if TOKEN = IDENT then begin
          GET TOKEN();
          If TOKEN <> ":=" then ERROR ();
          GET_TOKEN();
          EXPRESSION();
    end
```



```
TOKEN=;

Symbol Table

m=8; a; b; c;
ratio; x; y;

program()
block(1)
proc-decl(1)
block(2)
statement(2)
```

```
if b > a then begin
    x = b;
    y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```



```
TOKEN=;

Symbol Table

m=8; a; b; c;
ratio; x; y;

program()
block(1)
proc-decl(1)
block(2)
statement(2)
```

```
if b > a then begin
    x = b;
    y = a;
    end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
procedure STATEMENT;
...

else if TOKEN = "begin" then begin
    GET TOKEN();
    STATEMENT();
    while TOKEN = ";" do begin
    GET_TOKEN();
    STATEMENT();
    end;
    if TOKEN <> "end" then ERROR ();
    GET_TOKEN();
    end;
end;
```



```
TOKEN= if

Symbol Table

m=8; a; b; c;
ratio; x; y;

program()
block(1)
proc-decl(1)
block(2)
statement(2)
```

```
b > a then begin
x = b;
y = a;
end
c = x / y;
end;
begin
a = m;
b = 4;
call ratio;
end.
```

```
procedure STATEMENT;
...

else if TOKEN = "begin" then begin
    GET TOKEN();
    STATEMENT();
    while TOKEN = ";" do begin
        GET_TOKEN();
    STATEMENT();
    end;
    if TOKEN <> "end" then ERROR ();
    GET_TOKEN();
    end;
end;
```



```
TOKEN= if

Symbol Table

m=8; a; b; c;
ratio; x; y;

procedure STATEMENT;

program()
block(1)
proc-decl(1)
block(2)
statement(2)
```

```
b > a then begin
x = b;
y = a;
end
c = x / y;
end;
begin
a = m;
b = 4;
call ratio;
end.
```



```
TOKEN= b

Symbol Table

m=8; a; b; c;
ratio; x; y;

procedure STATEMENT;

program()
block(1)
proc-decl(1)
block(2)
statement(2)
```

```
> a then begin

x = b;

y = a;

end

c = x / y;

end;

begin

a = m;

b = 4;

call ratio;

end.
```



```
Symbol Table
TOKEN= b
                                            program()
                                            block(1)
                        m=8; a; b; c;
                                            proc-decl(1)
                        ratio; x; y;
                                            block(2)
                                            statement(2)
                                            statement(2)
  procedure CONDITION;
                                            condition(2)
 begin
    if TOKEN = "odd" then begin
           GET TOKEN();
           EXPRESSION();
      else begin
           EXPRESSION();
           if TOKEN <> RELATION then ERROR ();
           GET TOKEN();
           EXPRESSION();
      end
 end;
```

```
> a then begin

x = b;

y = a;

end

c = x / y;

end;

begin

a = m;

b = 4;

call ratio;

end.
```



```
Symbol Table
TOKEN= b
                                            program()
                                            block(1)
                        m=8; a; b; c;
                                            proc-decl(1)
                        ratio; x; y;
                                            block(2)
                                            statement(2)
                                            statement(2)
  procedure CONDITION;
                                            condition(2)
 begin
      if TOKEN = "odd" then begin
           GET TOKEN();
           EXPRESSION();
      else begin
        EXPRESSION();
           if TOKEN <> RELATION then ERROR ();
           GET TOKEN();
           EXPRESSION();
      end
 end;
```

```
> a then begin

x = b;

y = a;

end

c = x / y;

end;

begin

a = m;

b = 4;

call ratio;

end.
```



```
Symbol Table
TOKEN=>
                                            program()
                                            block(1)
                        m=8; a; b; c;
                                            proc-decl(1)
                        ratio; x; y;
                                            block(2)
                                            statement(2)
                                            statement(2)
  procedure CONDITION;
                                            condition(2)
 begin
      if TOKEN = "odd" then begin
           GET TOKEN();
           EXPRESSION();
      else begin
           EXPRESSION();
           if TOKEN <> RELATION then ERROR ();
           GET_TOKEN();
           EXPRESSION();
      end
 end;
```

```
a then begin

x = b;

y = a;

end

c = x / y;

end;

begin

a = m;

b = 4;

call ratio;

end.
```

then begin

x = b;

y = a;

c = x / y;

a = m;

b = 4;

call ratio;

end

end;

begin

end.



## Small Example

```
Symbol Table
TOKEN= a
                                            program()
                                            block(1)
                        m=8; a; b; c;
                                            proc-decl(1)
                        ratio; x; y;
                                            block(2)
                                            statement(2)
                                            statement(2)
  procedure CONDITION;
                                            condition(2)
 begin
      if TOKEN = "odd" then begin
           GET TOKEN();
           EXPRESSION();
      else begin
           EXPRESSION();
           if TOKEN <> RELATION then ERROR ();
           GET TOKEN();
           EXPRESSION();
      end
 end;
```

begin

x = b;

y = a;

c = x / y;

a = m;

b = 4;

call ratio;

end

end;

begin

end.



## Small Example

```
Symbol Table
TOKEN= then
                                            program()
                                            block(1)
                        m=8; a; b; c;
                                            proc-decl(1)
                        ratio; x; y;
                                            block(2)
                                            statement(2)
                                            statement(2)
  procedure CONDITION;
                                            condition(2)
 begin
      if TOKEN = "odd" then begin
           GET TOKEN();
           EXPRESSION();
      else begin
           EXPRESSION();
           if TOKEN <> RELATION then ERROR ();
           GET TOKEN();
           EXPRESSION();
      end
end;
```



#### Recursion stack

#### begin

```
x = b;
y = a;
end
c = x / y;
end;
begin
a = m;
b = 4;
call ratio;
end.
```

```
else if TOKEN = "if" then begin

GET_TOKEN();

CONDITION();

if TOKEN <> "then" then ERROR ();

GET_TOKEN();

STATEMENT();

end;
```



```
TOKEN= begin

Symbol Table

m=8; a; b; c;
ratio; x; y;

procedure STATEMENT;

...

Symbol Table
program()
block(1)
proc-decl(1)
block(2)
statement(2)
statement(2)
```

```
x = b;
y = a;
end
c = x / y;
end;
begin
a = m;
b = 4;
call ratio;
end.
```

```
else if TOKEN = "if" then begin

GET_TOKEN();

CONDITION();

if TOKEN <> "then" then ERROR ();

GET_TOKEN();

STATEMENT();
end;
```



## TOKEN= begin Symbol Table m=8; a; b; c; ratio; x; y; procedure STATEMENT; ... procedure STATEMENT; ... procedure Statement(2) statement(2) statement(2)

```
x = b;
y = a;
end
c = x / y;
end;
begin
a = m;
b = 4;
call ratio;
end.
```

```
else if TOKEN = "begin" then begin

GET TOKEN();

STATEMENT();

while TOKEN = ";" do begin

GET_TOKEN();

STATEMENT();

end;

if TOKEN <> "end" then ERROR ();

GET_TOKEN();

end;
```



# TOKEN= x Symbol Table m=8; a; b; c; ratio; x; y; procedure STATEMENT; ... program() block(1) proc-decl(1) block(2) statement(2) statement(2) statement(2)

```
= b;
y = a;
end
c = x / y;
end;
begin
a = m;
b = 4;
call ratio;
end.
```

```
else if TOKEN = "begin" then begin

GET TOKEN();

STATEMENT();

while TOKEN = ";" do begin

GET_TOKEN();

STATEMENT();

end;

if TOKEN <> "end" then ERROR ();

GET_TOKEN();

end;
```



```
TOKEN=;

Symbol Table

m=8; a; b; c;
ratio; x; y;

procedure STATEMENT;
...

program()
block(1)
proc-decl(1)
block(2)
statement(2)
statement(2)
```

```
y = a;
end
c = x / y;
end;
begin
a = m;
b = 4;
call ratio;
end.
```

```
else if TOKEN = "begin" then begin

GET TOKEN();

STATEMENT();

while TOKEN = ";" do begin

GET_TOKEN();

STATEMENT();

end;

if TOKEN <> "end" then ERROR ();

GET_TOKEN();

end;
```



```
TOKEN= y

Symbol Table

m=8; a; b; c;
ratio; x; y;

procedure STATEMENT;
...

program()
block(1)
proc-decl(1)
block(2)
statement(2)
statement(2)
```

```
= a;
end
c = x / y;
end;
begin
a = m;
b = 4;
call ratio;
end.
```

```
else if TOKEN = "begin" then begin

GET TOKEN();

STATEMENT();

while TOKEN = ";" do begin

GET_TOKEN();

STATEMENT();

end;

if TOKEN <> "end" then ERROR ();

GET_TOKEN();

end;
```



# TOKEN=; Symbol Table m=8; a; b; c; ratio; x; y; procedure STATEMENT; ... procedure STATEMENT; ... procedure Statement(2) statement(2) statement(2)

```
end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
else if TOKEN = "begin" then begin

GET TOKEN();

STATEMENT();

while TOKEN = ";" do begin

GET_TOKEN();

STATEMENT();

end;

if TOKEN <> "end" then ERROR ();

GET_TOKEN();

end;
```



## TOKEN=; Symbol Table m=8; a; b; c; ratio; x; y; procedure STATEMENT; ... program() block(1) proc-decl(1) block(2) statement(2) statement(2)

```
end
    c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
else if TOKEN = "begin" then begin

GET TOKEN();

STATEMENT();

while TOKEN = ";" do begin

GET_TOKEN();

STATEMENT();

end;

if TOKEN <> "end" then ERROR ();

GET_TOKEN();

end;
```



# TOKEN= end Symbol Table m=8; a; b; c; ratio; x; y; procedure STATEMENT; ... program() block(1) proc-decl(1) block(2) statement(2) statement(2)

```
c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
else if TOKEN = "begin" then begin
     GET TOKEN();
     STATEMENT();
     while TOKEN = ";" do begin
          GET TOKEN();
        → STATEMENT();
     end;
     if TOKEN <> "end" then ERROR ();
     GET TOKEN();
end;
```



# TOKEN= end Symbol Table m=8; a; b; c; ratio; x; y; procedure STATEMENT; ... program() block(1) proc-decl(1) block(2) statement(2) statement(2) statement(2)

```
;
c = x / y;
end;
begin
a = m;
b = 4;
call ratio;
end.
```

```
else if TOKEN = "begin" then begin

GET TOKEN();

STATEMENT();

while TOKEN = ";" do begin

GET_TOKEN();

STATEMENT();

end;

if TOKEN <> "end" then ERROR ();

GET_TOKEN();

end;

end;
```



```
TOKEN=;

Symbol Table

m=8; a; b; c;
ratio; x; y;

procedure STATEMENT;
...

procedure STATEMENT;
...

program()
block(1)
proc-decl(1)
block(2)
statement(2)
statement(2)
```

```
c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
procedure STATEMENT;
...

else if TOKEN = "begin" then begin
    GET TOKEN();
    STATEMENT();
    while TOKEN = ";" do begin
        GET_TOKEN();
        STATEMENT();
    end;
    if TOKEN <> "end" then ERROR ();
    GET_TOKEN();

end;
end;
```



```
TOKEN=;

Symbol Table

m=8; a; b; c;
ratio; x; y;

procedure STATEMENT;
...

else if TOKEN = "if" then begin
GET_TOKEN();
CONDITION();
```

```
c = x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
GET_TOKEN();
CONDITION();
if TOKEN <> "then" then ERROR ();
GET_TOKEN();
STATEMENT();

end;
```



#### Recursion stack

```
TOKEN=;

Symbol Table

m=8; a; b; c;
ratio; x; y;

program()
block(1)
proc-decl(1)
block(2)
statement(2)
```

```
else if TOKEN = "begin" then begin
                                         GET TOKEN();
                                         STATEMENT();
   c = x / y;
                                         while TOKEN = ";" do begin
end;
                                               GET TOKEN();
begin
                                               STATEMENT();
    a = m;
                                        →end;
    b = 4;
                                         if TOKEN <> "end" then ERROR ();
   call ratio;
                                         GET TOKEN();
end.
                                    end;
```

procedure STATEMENT;



#### Recursion stack

```
TOKEN=;

Symbol Table

m=8; a; b; c;
ratio; x; y;

program()
block(1)
proc-decl(1)
block(2)
statement(2)
```

```
else if TOKEN = "begin" then begin
                                         GET TOKEN();
                                         STATEMENT();
   c = x / y;
                                         while TOKEN = ";" do begin
end;
                                            → GET_TOKEN();
begin
                                               STATEMENT();
    a = m;
                                         end;
    b = 4;
                                         if TOKEN <> "end" then ERROR ();
   call ratio;
                                         GET TOKEN();
end.
                                    end;
```

procedure STATEMENT;



end;

begin

end.

# Small Example

### Recursion stack

```
TOKEN= c

Symbol Table

m=8; a; b; c;
ratio; x; y;

program()
block(1)
proc-decl(1)
block(2)
statement(2)
```

```
else if TOKEN = "begin" then begin

GET TOKEN();

STATEMENT();

while TOKEN = ";" do begin

GET_TOKEN();

in

a = m;

b = 4;
call ratio;

cell ratio;

else if TOKEN = "begin" then begin

GET_TOKEN();

while TOKEN = ";" do begin

GET_TOKEN();

GET_TOKEN();

end;

end;

GET_TOKEN();

end;
```

procedure STATEMENT;



#### Symbol Table TOKEN= c program() block(1) m=8; a; b; c; proc-decl(1) ratio; x; y; block(2) statement(2)

```
= x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
statement(2)
procedure STATEMENT;
begin
     if TOKEN = IDENT then begin
    → GET_TOKEN();
         If TOKEN <> ":=" then ERROR);
         GET_TOKEN();
         EXPRESSION();
     end
```



→ GET\_TOKEN();

end

EXPRESSION();

#### Symbol Table TOKEN== program() block(1) m=8; a; b; c; proc-decl(1) ratio; x; y; block(2) statement(2)

```
x / y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
statement(2)
procedure STATEMENT;
begin
     if TOKEN = IDENT then begin
          GET TOKEN();
          If TOKEN <> ":=" then ERROR);
```



GET\_TOKEN();

EXPRESSION();

end

#### Symbol Table TOKEN= x program() block(1) m=8; a; b; c; proc-decl(1) ratio; x; y; block(2) statement(2)

```
/ y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```

```
statement(2)
procedure STATEMENT;
begin
     if TOKEN = IDENT then begin
          GET TOKEN();
          If TOKEN <> ":=" then ERROR);
```



```
TOKEN= x
                       Symbol Table
                                           program()
                                           block(1)
                       m=8; a; b; c;
                                           proc-decl(1)
                       ratio; x; y;
                                           block(2)
                                           statement(2)
                                           statement(2)
                                           expression(2)
 procedure EXPRESSION;
 begin
      if TOKEN = ADDING_OPERATOR then GET_TOKEN();
  → TERM();
      while TOKEN = ADDING_OPERATOR do begin
           GET TOKEN();
           TERM();
      end
 end;
```

```
/ y;
end;
begin
a = m;
b = 4;
call ratio;
end.
```



end

end;

```
TOKEN= x
                       Symbol Table
                                          program()
                                          block(1)
                       m=8; a; b; c;
                                          proc-decl(1)
                       ratio; x; y;
                                          block(2)
                                          statement(2)
                                          statement(2)
                                          expression(2)
 procedure TERM;
 begin
                                          term(2)
   FACTOR();
      while TOKEN = MULTIPLYING OPERATOR do begin
          GET_TOKEN();
          FACTOR();
```

```
/ y;
end;
begin
a = m;
b = 4;
call ratio;
end.
```



GET TOKEN();

GET\_TOKEN();

EXPRESSION();

GET\_TOKEN();

end

end;

else ERROR ();

else if TOKEN = "(" then begin

if TOKEN <> ")" then ERROR ();

```
Symbol Table
TOKEN= x
                                           program()
                                           block(1)
                       m=8; a; b; c;
                                           proc-decl(1)
                       ratio; x; y;
                                           block(2)
                                           statement(2)
                                           statement(2)
                                           expression(2)
 procedure FACTOR;
 begin
                                           term(2)
      if TOKEN = IDENTIFIER then
                                           factor(2)
     → GET TOKEN();
      else if TOKEN = NUMBER then
```

```
/ y;
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```



```
Symbol Table
TOKEN=/
                                           program()
                                           block(1)
                       m=8; a; b; c;
                                           proc-decl(1)
                       ratio; x; y;
                                           block(2)
                                           statement(2)
                                           statement(2)
                                           expression(2)
 procedure FACTOR;
 begin
                                           term(2)
      if TOKEN = IDENTIFIER then
                                           factor(2)
           GET TOKEN();
```

Recursion stack

```
y;
end;
begin
a = m;
b = 4;
call ratio;
end.
```

else if TOKEN = NUMBER then

end;



# Small Example

```
Symbol Table
TOKEN=/
                                          program()
                                          block(1)
                       m=8; a; b; c;
                                          proc-decl(1)
                       ratio; x; y;
                                          block(2)
                                          statement(2)
                                          statement(2)
                                          expression(2)
 procedure TERM;
 begin
                                          term(2)
      FACTOR();
      while TOKEN = MULTIPLYING OPERATOR do begin
     → GET TOKEN();
          FACTOR();
      end
```

```
y;
end;
begin
a = m;
b = 4;
call ratio;
end.
```



end

end;

```
TOKEN= y
                       Symbol Table
                                          program()
                                          block(1)
                       m=8; a; b; c;
                                          proc-decl(1)
                       ratio; x; y;
                                          block(2)
                                          statement(2)
                                          statement(2)
                                          expression(2)
 procedure TERM;
 begin
                                          term(2)
      FACTOR();
      while TOKEN = MULTIPLYING OPERATOR do begin
          GET_TOKEN();
     → FACTOR();
```

```
;
end;
begin
a = m;
b = 4;
call ratio;
end.
```

end;



# Small Example

```
TOKEN=;
                       Symbol Table
                                           program()
                                           block(1)
                       m=8; a; b; c;
                                           proc-decl(1)
                       ratio; x; y;
                                           block(2)
                                           statement(2)
                                           statement(2)
                                           expression(2)
 procedure TERM;
 begin
                                           term(2)
      FACTOR();
      while TOKEN = MULTIPLYING OPERATOR do begin
           GET_TOKEN();
           FACTOR();
      end
```

```
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```



```
TOKEN=;
                       Symbol Table
                                           program()
                                           block(1)
                       m=8; a; b; c;
                                           proc-decl(1)
                       ratio; x; y;
                                           block(2)
                                           statement(2)
                                           statement(2)
                                           expression(2)
 procedure EXPRESSION;
 begin
      if TOKEN = ADDING_OPERATOR then GET_TOKEN();
      TERM();
      while TOKEN = ADDING OPERATOR do begin
           GET TOKEN();
           TERM();
      end
end;
```

```
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```



## Recursion stack

```
TOKEN=;
                       Symbol Table
                                           program()
                                           block(1)
                       m=8; a; b; c;
                                           proc-decl(1)
                       ratio; x; y;
                                           block(2)
                                           statement(2)
                                           statement(2)
  procedure STATEMENT;
 begin
      if TOKEN = IDENT then begin
           GET TOKEN();
```

If TOKEN <> ":=" then ERROR);

GET\_TOKEN();

EXPRESSION();

end

```
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```



### Recursion stack

```
TOKEN=;

Symbol Table

m=8; a; b; c;
ratio; x; y;

program()
block(1)
proc-decl(1)
block(2)
statement(2)
```

```
else if TOKEN = "begin" then begin

GET TOKEN();

STATEMENT();

while TOKEN = ";" do begin

GET_TOKEN();

STATEMENT();

end;

if TOKEN <> "end" then ERROR ();

GET_TOKEN();

end;
```

end;
begin
 a = m;
 b = 4;
 call ratio;
end.



```
TOKEN=;

Symbol Table

m=8; a; b; c;
ratio; x; y;

program()
block(1)
proc-decl(1)
block(2)
statement(2)
```

```
end;
begin
    a = m;
    b = 4;
    call ratio;
end.
```



### Recursion stack

```
TOKEN= end

Symbol Table

m=8; a; b; c;
ratio; x; y;

program()
block(1)
proc-decl(1)
block(2)
statement(2)
```

; begin a = m; b = 4; call ratio; end.



### Recursion stack

```
TOKEN= end

Symbol Table

m=8; a; b; c;
ratio; x; y;

program()
block(1)
proc-decl(1)
block(2)
statement(2)
```

```
else if TOKEN = "begin" then begin

GET TOKEN();

STATEMENT();

while TOKEN = ";" do begin

GET_TOKEN();

STATEMENT();

end;

if TOKEN <> "end" then ERROR ();

end;

end;

end;
```

begin a = m; b = 4; call ratio; end.

• • •



```
TOKEN=;

Symbol Table

m=8; a; b; c;
ratio; x; y;

program()
block(1)
proc-decl(1)
block(2)
statement(2)
```

```
begin

a = m;

b = 4;

call ratio;

end.
```



TOKEN=;	Symbol Table	program()
	m=8; a; b; c; ratio; x; y;	block(1) proc-decl(1) block(2)

```
procedure BLOCK;
begin
    if TOKEN = "const" then CONST-DECL();
    if TOKEN = "var" then VAR-DECL();
    if TOKEN = "procedure" then PROC-DECL();
    STATEMENT();
end;
```

```
begin
    a = m;
    b = 4;
    call ratio;
end.
```



#### Recursion stack

TOKEN=;	Symbol Table	program()
	m=8; a; b; c; ratio; x; y;	block(1) proc-decl(1)

```
procedure PROC-DECL;
begin
while TOKEN = "procedure" do begin
    GET_TOKEN();
    if TOKEN <> IDENT then ERROR ();
    ENTER(procedure, ident);
    GET_TOKEN();
    if TOKEN <> ";" then ERROR ();
    GET_TOKEN();
    BLOCK(level+1);
    if TOKEN <> ";" then ERROR ();
    GET_TOKEN();
end;
end;
```

```
begin
a = m;
b = 4;
call ratio;
```

end.



TOKEN= begin	Symbol Table	program()
	m=8; a; b; c; ratio; x; y;	block(1) proc-decl(1)

```
procedure PROC-DECL;
begin
while TOKEN = "procedure" do begin
    GET_TOKEN();
    if TOKEN <> IDENT then ERROR ();
    ENTER(procedure, ident);
    GET_TOKEN();
    if TOKEN <> ";" then ERROR ();
    GET_TOKEN();
    BLOCK(level+1);
    if TOKEN <> ";" then ERROR ();
    GET_TOKEN();
    end;
end;
```

```
a = m;
b = 4;
call ratio;
end.
```



TOKEN= begin	Symbol Table	program()
	m=8; a; b; c;	block(1)
	ratio; x; y;	

```
procedure BLOCK;
begin
    if TOKEN = "const" then CONST-DECL();
    if TOKEN = "var" then VAR-DECL();
    if TOKEN = "procedure" then PROC-DECL();

>>> STATEMENT;
end;
```

```
a = m;
b = 4;
call ratio;
end.
```



TOKEN= begin	Symbol Table	program()
	m=8; a; b; c; ratio; x; y;	block(1) statement(1)

```
a = m;
b = 4;
call ratio;
end.
```



TOKEN= a	Symbol Table	program()
	m=8; a; b; c; ratio; x; y;	block(1) statement(1)

```
= m;
b = 4;
call ratio;
end.
```



TOKEN=;	Symbol Table	program()
	m=8; a; b; c; ratio; x; y;	block(1) statement(1)

```
b = 4;
call ratio;
end.
```



#### Recursion stack

TOKEN=;	Symbol Table	program()
	m=8; a; b; c; ratio; x; y;	block(1) statement(1)

b = 4; call ratio; end.



#### Recursion stack

TOKEN= b	Symbol Table	program()
	m=8; a; b; c; ratio; x; y;	block(1) statement(1)

= 4; call ratio; end.



#### Recursion stack

TOKEN=;	Symbol Table	program()
	m=8; a; b; c; ratio; x; y;	block(1) statement(1)

```
procedure STATEMENT;
...

else if TOKEN = "begin" then begin
    GET TOKEN();
    STATEMENT();
    while TOKEN = ";" do begin
    GET_TOKEN();
    STATEMENT();
    end;
    if TOKEN <> "end" then ERROR ();
    GET_TOKEN();
    end;
```

call ratio; end.



### Recursion stack

TOKEN= call	Symbol Table	program()
	m=8; a; b; c; ratio; x; y;	block(1) statement(1)

ratio; end.



### Recursion stack

TOKEN=;	Symbol Table	program()
	m=8; a; b; c; ratio; x; y;	block(1) statement(1)

```
procedure STATEMENT;
...

else if TOKEN = "begin" then begin
    GET TOKEN();
    STATEMENT();
    while TOKEN = ";" do begin
    GET_TOKEN();
    STATEMENT();
    end;
    if TOKEN <> "end" then ERROR ();
    GET_TOKEN();
    end;
```

end.



### Recursion stack

TOKEN= end	Symbol Table	program()
	m=8; a; b; c; ratio; x; y;	block(1) statement(1)

•



### Recursion stack

TOKEN= end	Symbol Table	program()
	m=8; a; b; c; ratio; x; y;	block(1) statement(1)

•



TOKEN= .	Symbol Table	program()
	m=8; a; b; c; ratio; x; y;	block(1) statement(1)



TOKEN= .	Symbol Table	program()
	m=8; a; b; c;	block(1)
	ratio; x; y;	

```
procedure BLOCK;
begin
    if TOKEN = "const" then CONST-DECL();
    if TOKEN = "var" then VAR-DECL();
    if TOKEN = "procedure" then PROC-DECL();
    STATEMENT;

end;
```



TOKEN= .	Symbol Table	program()
	m=8; a; b; c; ratio; x; y;	

```
procedure PROGRAM;
begin
GET_TOKEN();
BLOCK();
if TOKEN <> "." then ERROR ()
→ end;
```