# Recitation 5: The Parser

COP3402 FALL 2015 – ARYA POURTABATABAIE

FROM EURIPIDES MONTAGNE, FALL 2014

# The Parsing Problem

Take a **string of symbols** in a language (tokens) and a **grammar** for that language to construct the ***parse tree*** or report that the sentence is syntactically ***incorrect***.

Two ways to do this:

- Top-Down (recursive descending parser).
- ~~Bottom-Up~~. (We don't focus on this).

The 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.

# PL/0 Grammar

<program> ::= <block> .

| | |
|---|---|
| <block> | ::= <const-decl> <var-decl> <proc-decl> <statement> |
| <const-decl> | ::= const <const-assignment-list> ; \| e |
| <const-assignment-list> | ::= <ident> = <number><br>\| <const-assignment-list> , <ident> = <number> |
| <var-decl> | ::= var <ident-list> ; \| e |
| <ident-list> | ::= <ident> \| <ident-list> , <ident> |
| <proc-decl> | ::= <proc-decl> procedure <ident> ; <block> ; \| e |
| <statement> | ::= <ident> := <expression> \| call <ident><br>\| begin <statement-list> end \| if <condition> then <statement><br>\| while <condition> do <statement> \| e |
| <statement-list> | ::= <statement> \| <statement-list> ; <statement> |
| <condition> | ::= odd <expression> \| <expression> <relation> <expression> |
| <relation> | ::= = \| <> \| < \| > \| <= \| >= |
| <expression> | ::= <term> \| <adding-operator> <term><br>\| <expression> <adding-operator> <term> |
| <adding-operator> | ::= + \| - |
| <term> | ::= <factor> \| <term> <multiplying-operator> <factor> |
| <multiplying-operator> | ::= * \| / |
| <factor> | ::= <ident> \| <number> \| ( <expression> ) |

This is a slightly different version of the PL-0 grammar than the one you have in your notes.

- It's intended to be clearer for purposes of going through these slides
- We use <tokens> and literals instead of tokens and "literals" for the same reason
- The equivalence should be obvious enough

# PL/0 Grammar

**Terminals**

const, var, procedure, call, begin, end, if, then, while, do, odd

<> < > <= >= + - * / =

, ; e

**Non-Terminals**

<program> <block> <const-decl> <var-decl> <proc-decl> <statement> <const-assignment-list> <ident> <number > <ident-list> <expression> <statement-list> <condition> <relation> <term> <adding-operator> <factor> <multiplying-operator>

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

<program> ::= <block> .

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

# <block> Procedure

<block> ::= <const-decl> <var-decl> <proc-decl> <statement>

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

# <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

<proc-decl> ::= <proc-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> Procedure

<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> Procedure

<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> Procedure

<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> Procedure

<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

<expression> ::= <term> | <adding-operator> <term>
| <expression> <adding-operator> <term>

```
procedure EXPRESSION;
begin
 if TOKEN = ADDING_OPERATOR then GET_TOKEN();
 TERM();
 while TOKEN = ADDING_OPERATOR do begin
        GET_TOKEN();
        TERM();
 end
end;
```

# <term> Procedure

<term> ::= <factor> | <term> <multiplying-operator> <factor>

```
procedure TERM;
begin
 FACTOR();
 while TOKEN = MULTIPLYING_OPERATOR do begin
        GET_TOKEN();
        FACTOR();
 end
end;
```

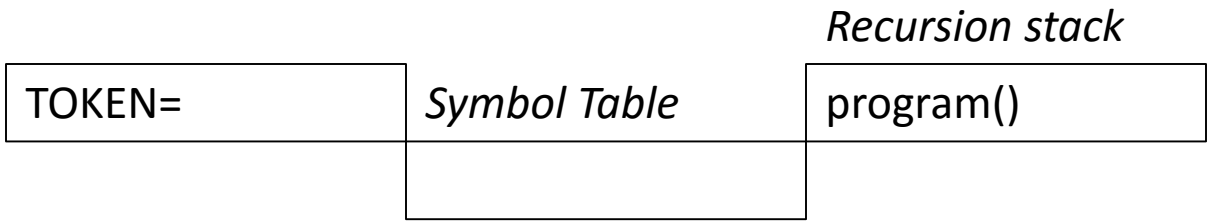
# <factor> Procedure

<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;
```

# Small Example

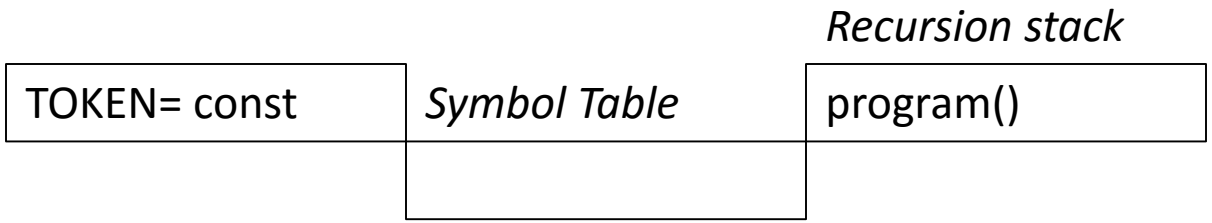
```
const m = 8;
var a, b, c;
procedure ratio;
var x, y;
begin
  x = a; y = b;
  if b > a then begin
                 x = b;
                 y = a;
  end
  c = x / y;
end;
begin
  a = m;
  b = 4;
  call ratio;
end.
```




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

# Small Example

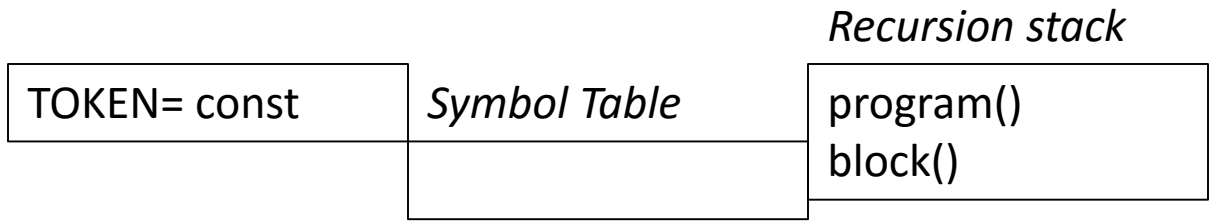
```
        m = 8;
var a, b, c;
procedure ratio;
var x, y;
begin
  x = a; y = b;
  if b > a then begin
                x = b;
                y = a;
  end
  c = x / y;
end;
begin
  a = m;
  b = 4;
  call ratio;
end.
```




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

# Small Example

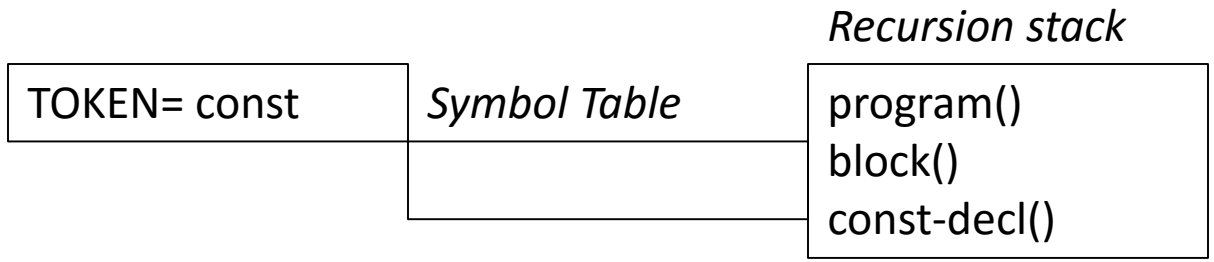
```
        m = 8;
var a, b, c;
procedure ratio;
var x, y;
begin
  x = a; y = b;
  if b > a then begin
                x = b;
                y = a;
  end
  c = x / y;
end;
begin
  a = m;
  b = 4;
  call ratio;
end.
```




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

# Small Example

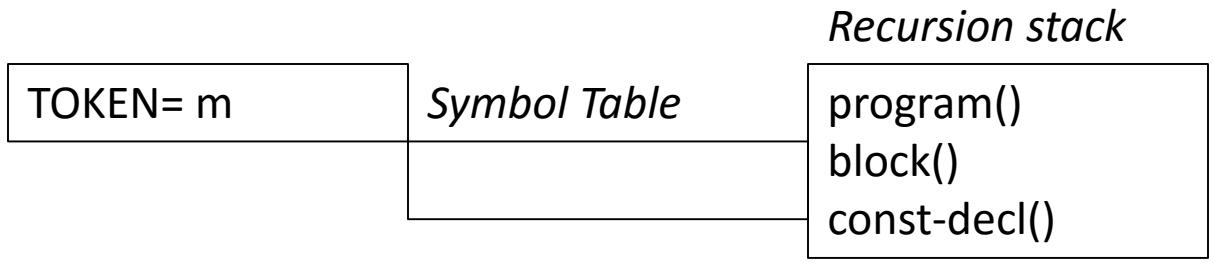
```
        m = 8;
var a, b, c;
procedure ratio;
var x, y;
begin
  x = a; y = b;
  if b > a then begin
                x = b;
                y = a;
  end
  c = x / y;
end;
begin
  a = m;
  b = 4;
  call ratio;
end.
```




```
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;
```

# Small Example

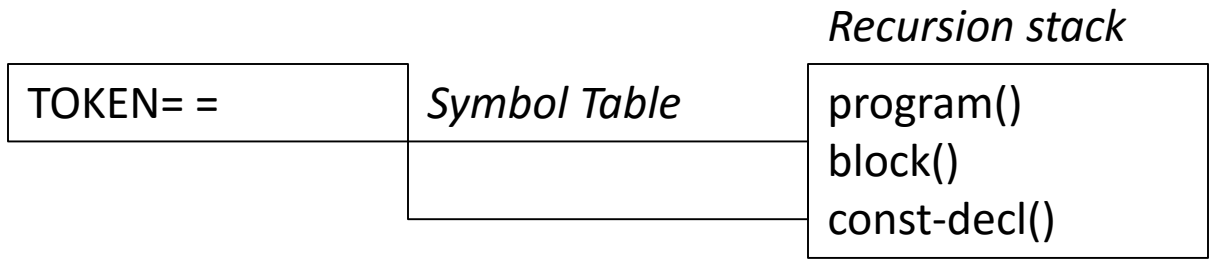
```
            = 8;
var a, b, c;
procedure ratio;
var x, y;
begin
  x = a; y = b;
  if b > a then begin
                x = b;
                y = a;
  end
  c = x / y;
end;
begin
  a = m;
  b = 4;
  call ratio;
end.
```




```
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;
```

# Small Example

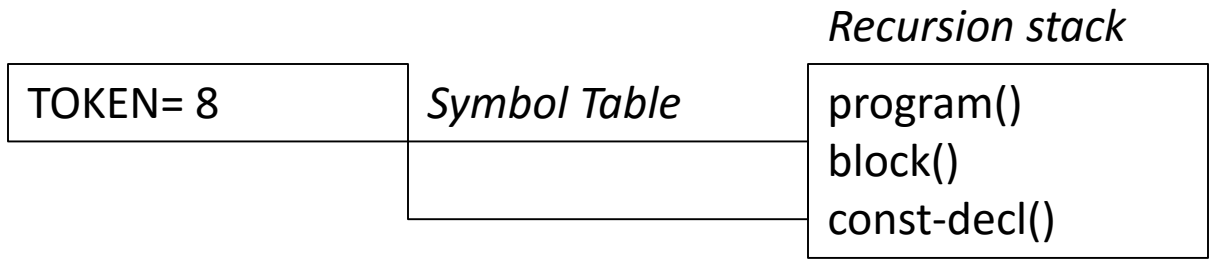
```
                8;
var a, b, c;
procedure ratio;
var x, y;
begin
  x = a; y = b;
  if b > a then begin
                x = b;
                y = a;
  end
  c = x / y;
end;
begin
  a = m;
  b = 4;
  call ratio;
end.
```




```
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;
```

# Small Example

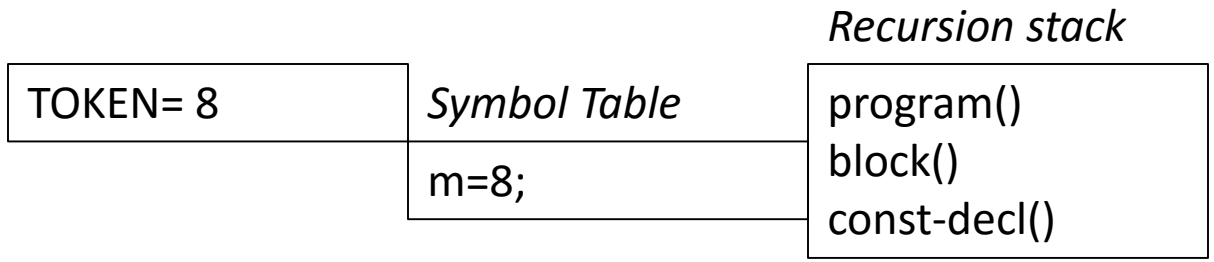
```
                ;
var a, b, c;
procedure ratio;
var x, y;
begin
  x = a; y = b;
  if b > a then begin
                x = b;
                y = a;
  end
  c = x / y;
end;
begin
  a = m;
  b = 4;
  call ratio;
end.
```




```
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;
```

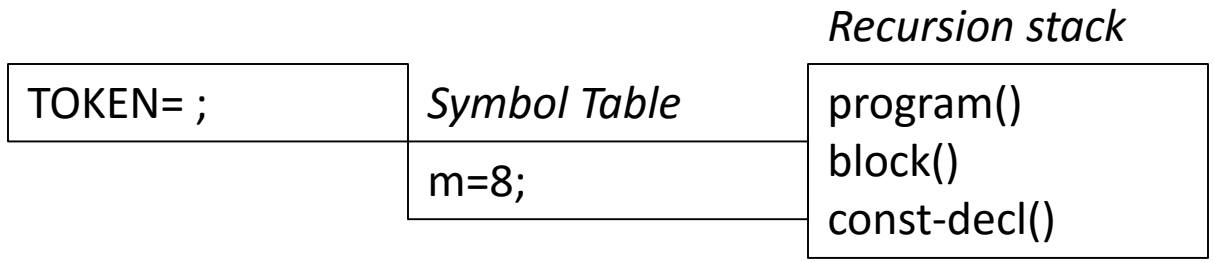
# Small Example

```
                ;
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;
```

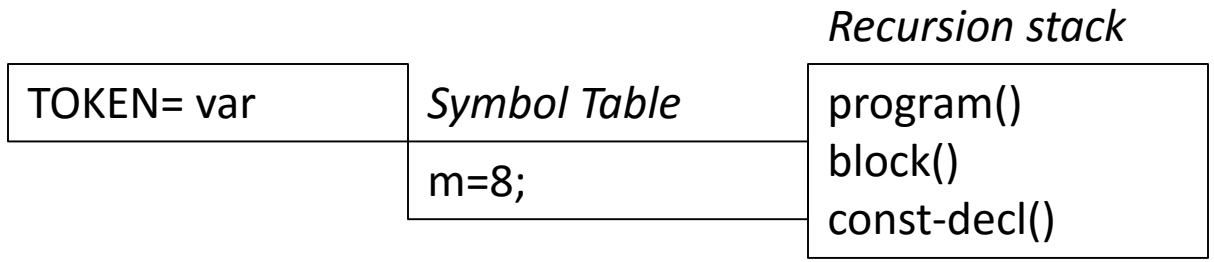
# Small Example




```
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;
```

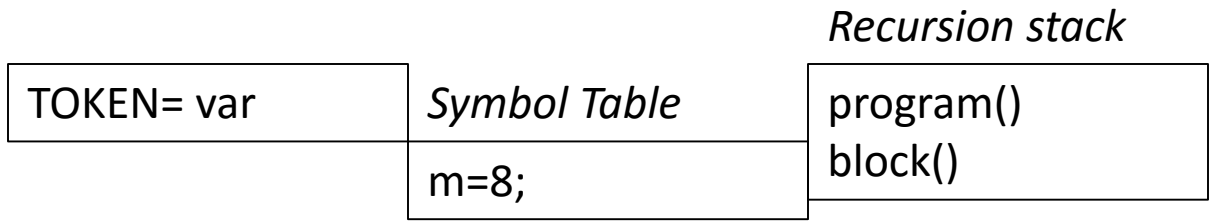
# Small Example




```
      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;
```

# Small Example

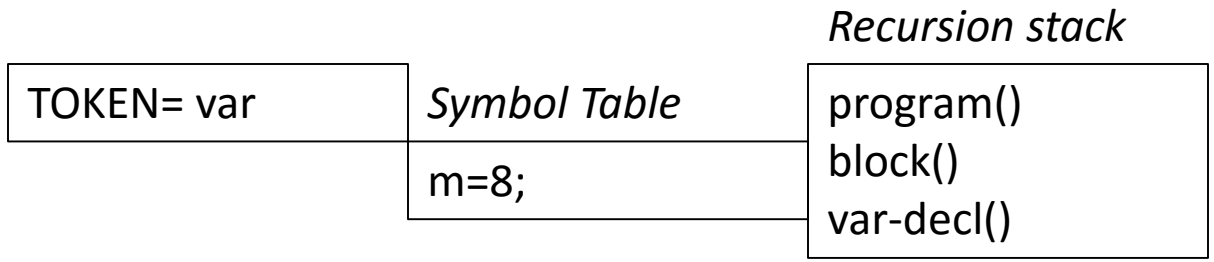



```
      a, b, c;
procedure ratio;
var x, y;
begin
  x = a; y = b;
  if b > a then begin
                x = b;
                y = a;
  end
  c = x / y;
end;
begin
  a = m;
  b = 4;
  call ratio;
end.
```

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

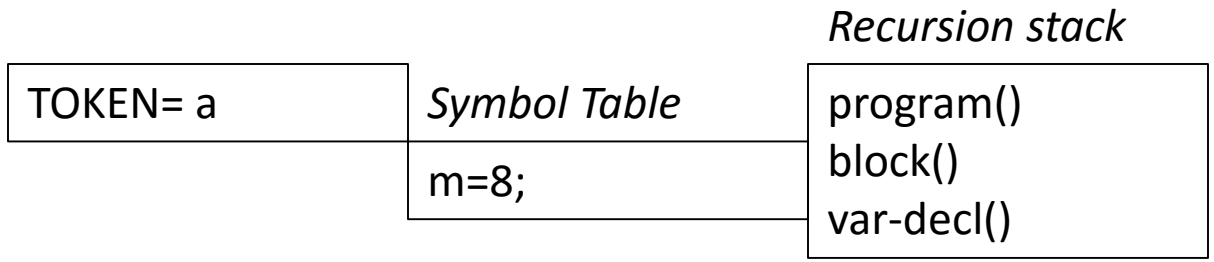
# Small Example

```
      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 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;
```

# Small Example

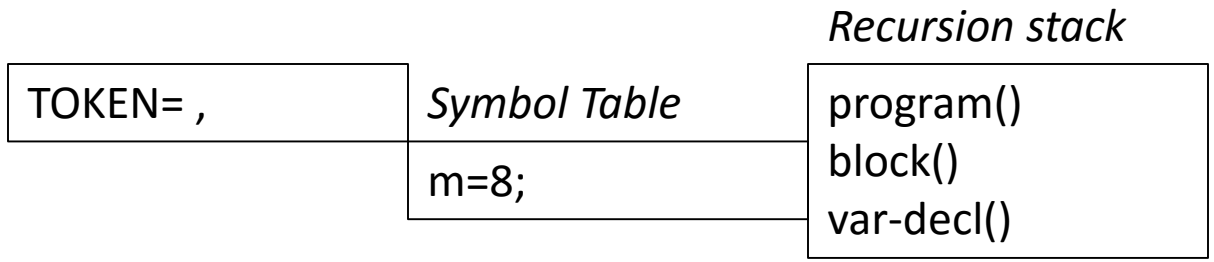



```
      , b, c;
procedure ratio;
var x, y;
begin
  x = a; y = b;
  if b > a then begin
                x = b;
                y = a;
  end
  c = x / y;
end;
begin
  a = m;
  b = 4;
  call ratio;
end.
```

```
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;
```

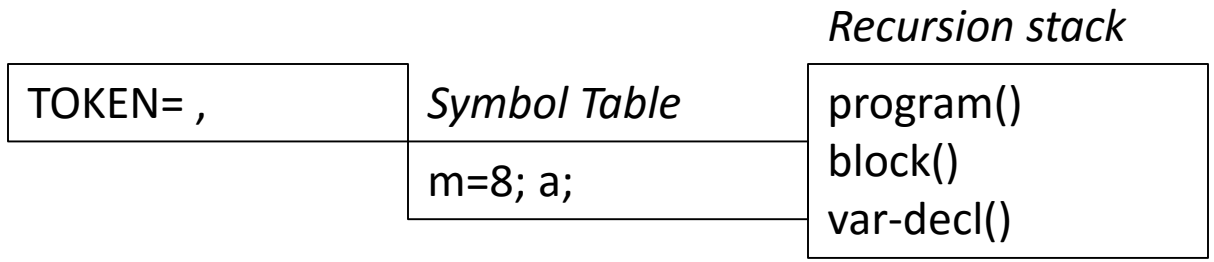
# Small Example

```
        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;
```

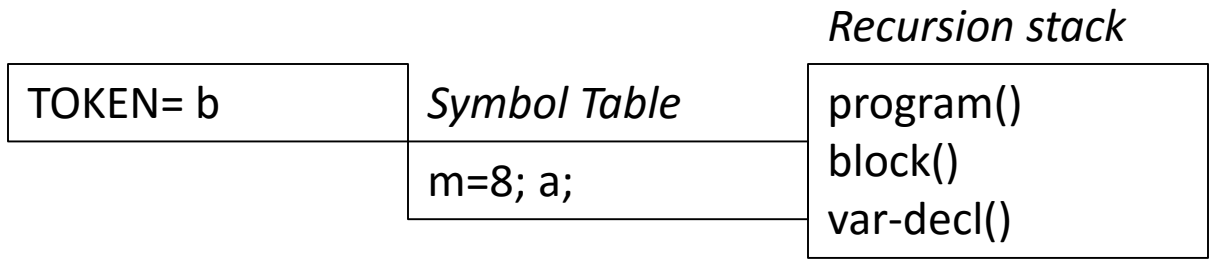
# Small Example




```
        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;
```

# Small Example

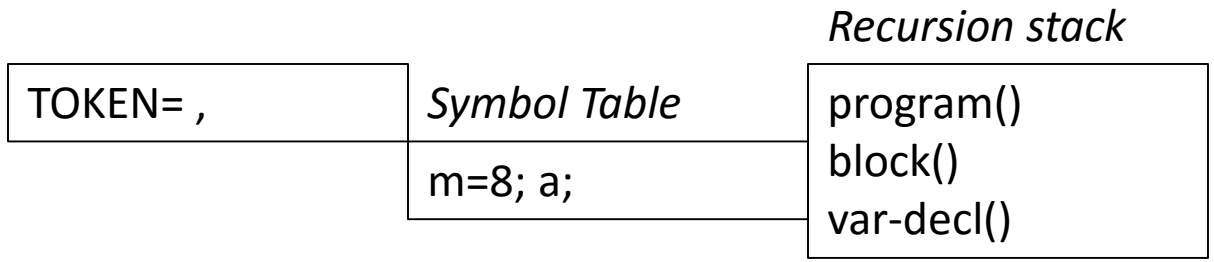



```
          , c;
procedure ratio;
var x, y;
begin
  x = a; y = b;
  if b > a then begin
                x = b;
                y = a;
  end
  c = x / y;
end;
begin
  a = m;
  b = 4;
  call ratio;
end.
```

```
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;
```

# Small Example

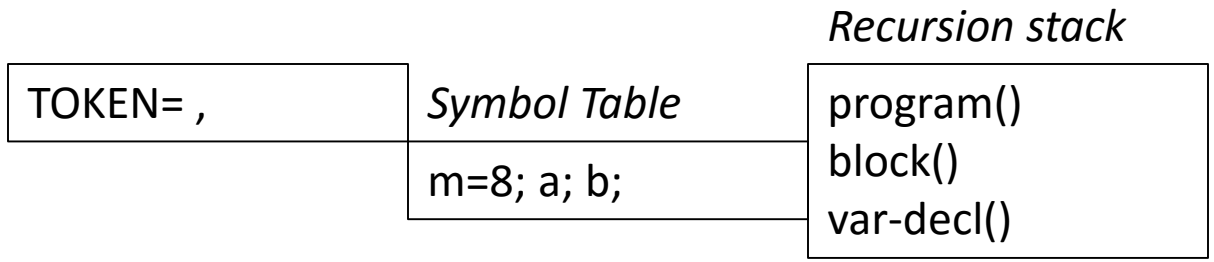
```
            c;
procedure ratio;
var x, y;
begin
  x = a; y = b;
  if b > a then begin
                x = b;
                y = a;
  end
  c = x / y;
end;
begin
  a = m;
  b = 4;
  call ratio;
end.
```




```
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;
```

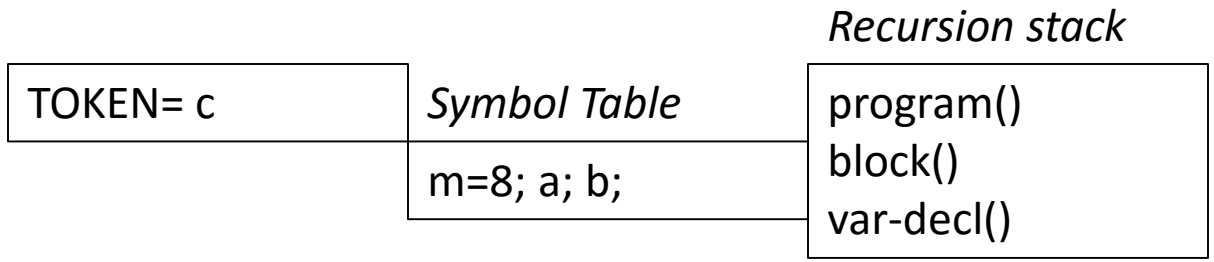
# Small Example

```
            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;
```

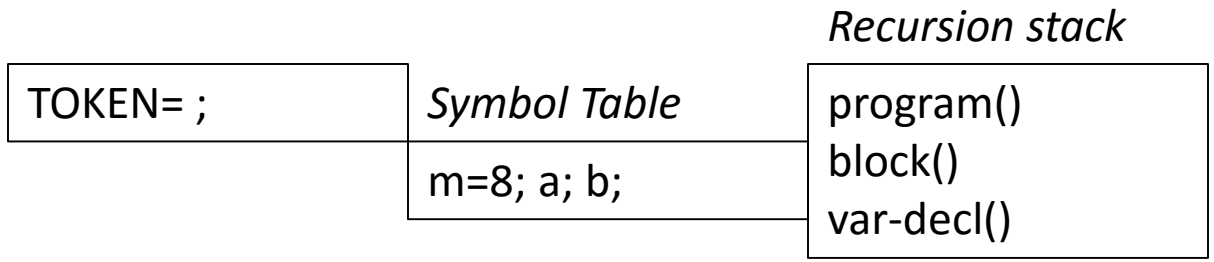
# Small Example




```
                ;
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;
```

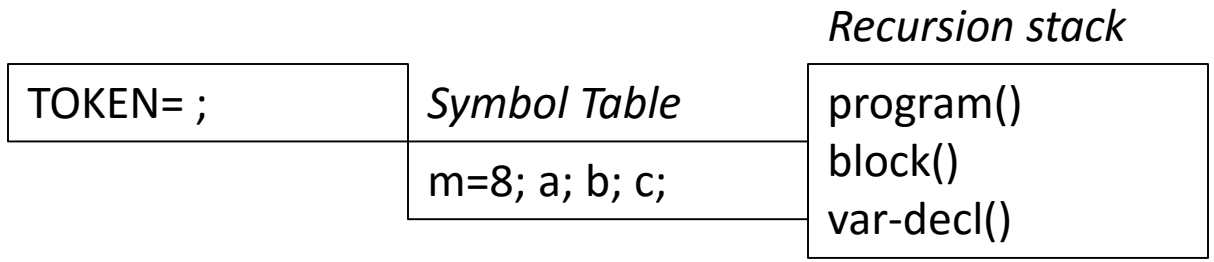
# Small Example




```
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;
```

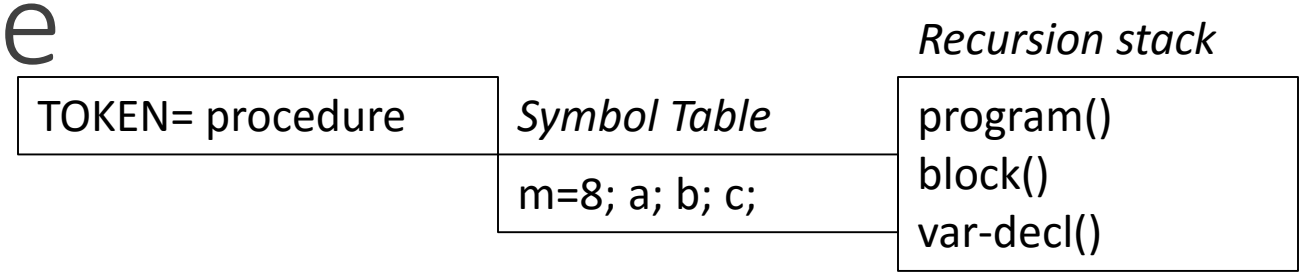
# Small Example




```
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;
```

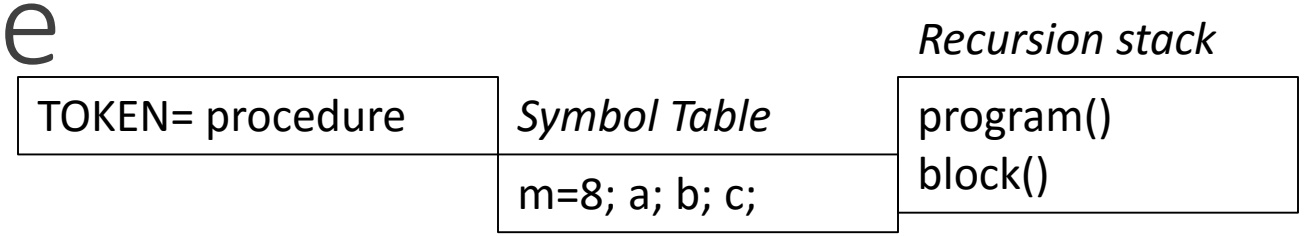
# Small Example




```
                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;
```

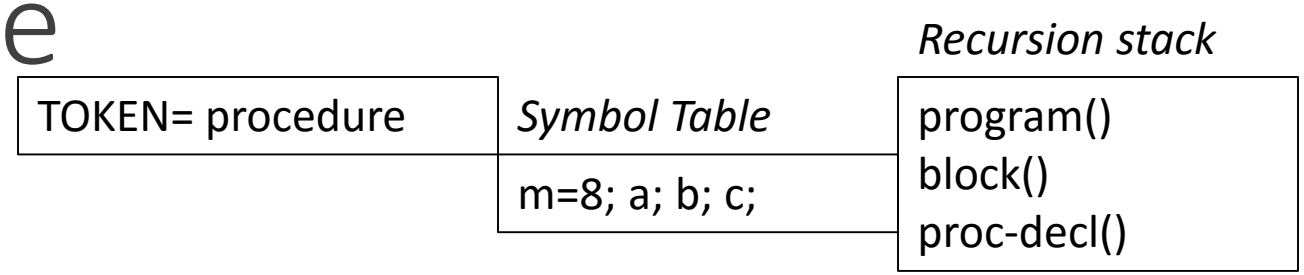
# Small Example




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

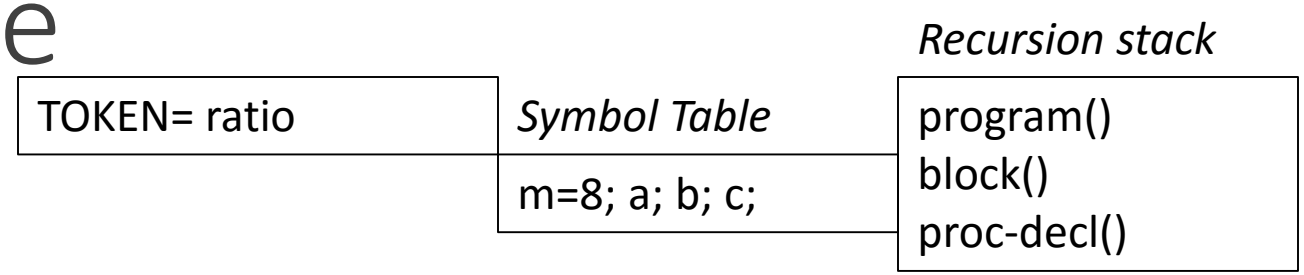
# Small Example




```
                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;
```

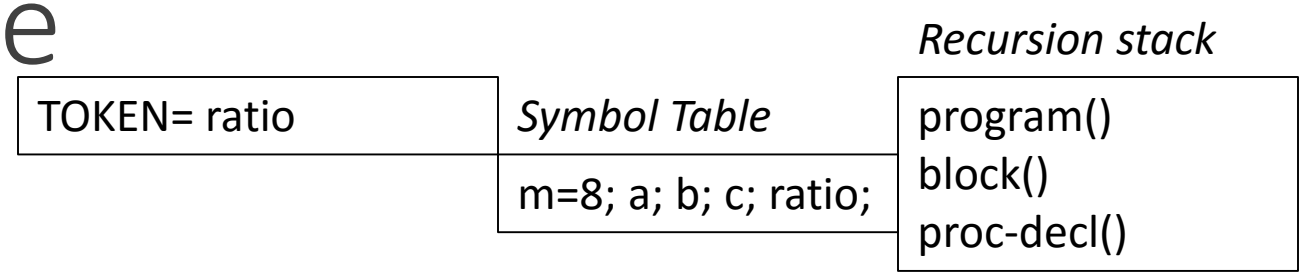
# Small Example




```
                    ;
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;
```

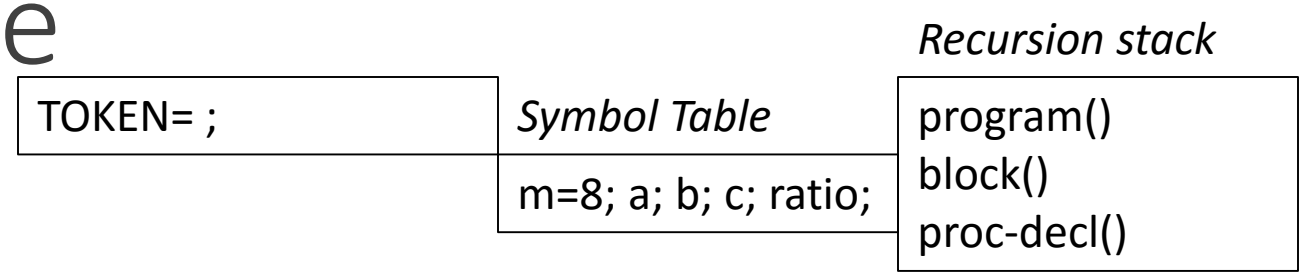
# Small Example




```
                    ;
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;
```

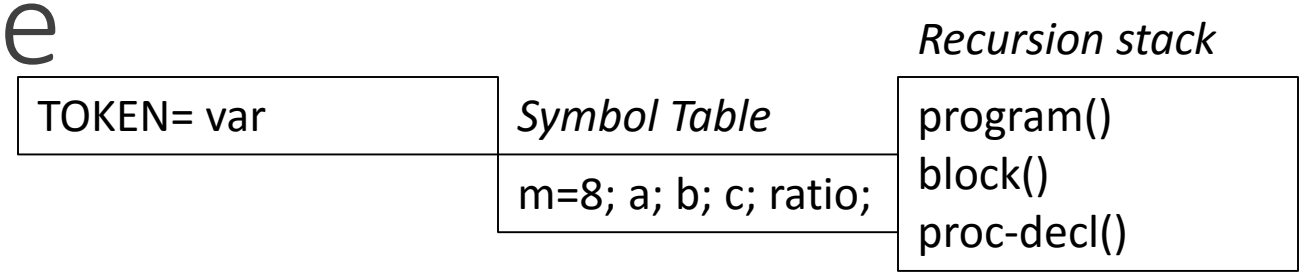
# Small Example




```
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;
```

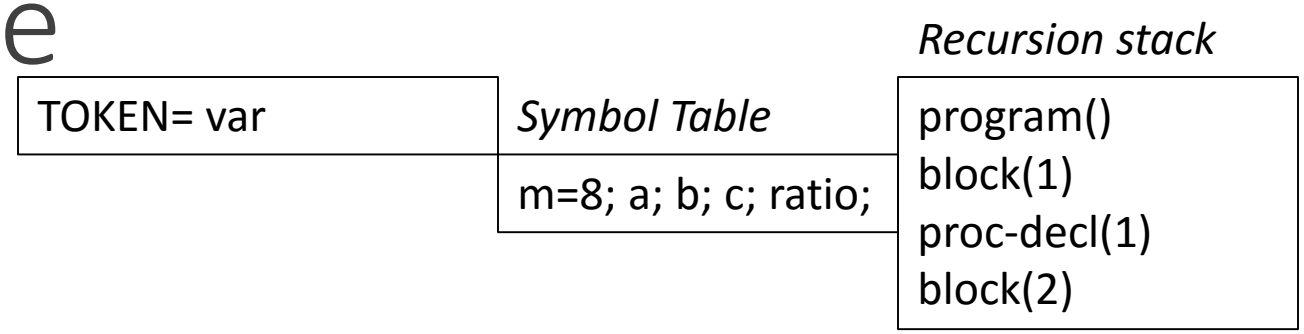
# Small Example




```
    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;
```

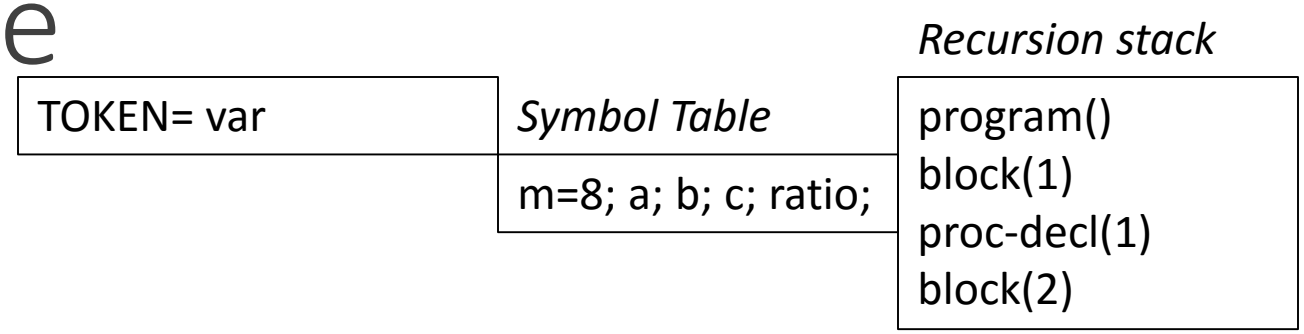
# Small Example




```
    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;
```

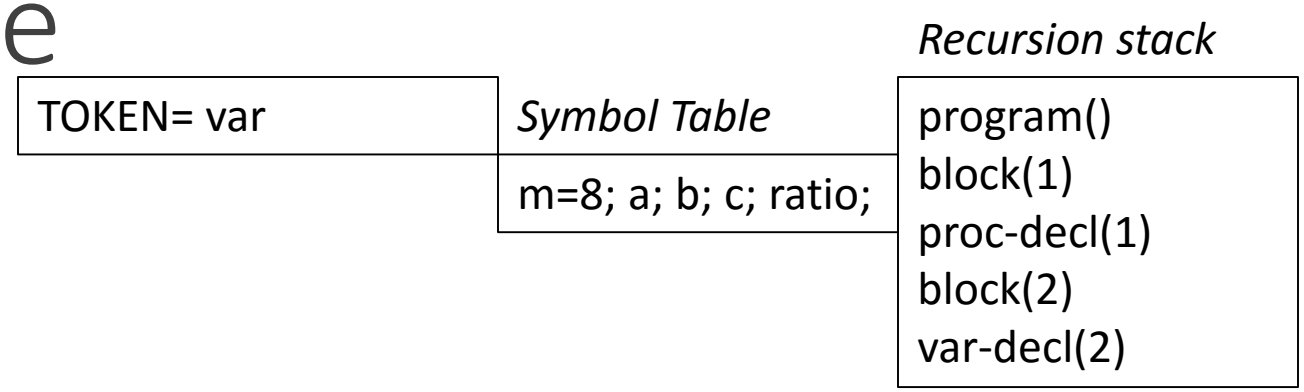
# Small Example




```
    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;
```

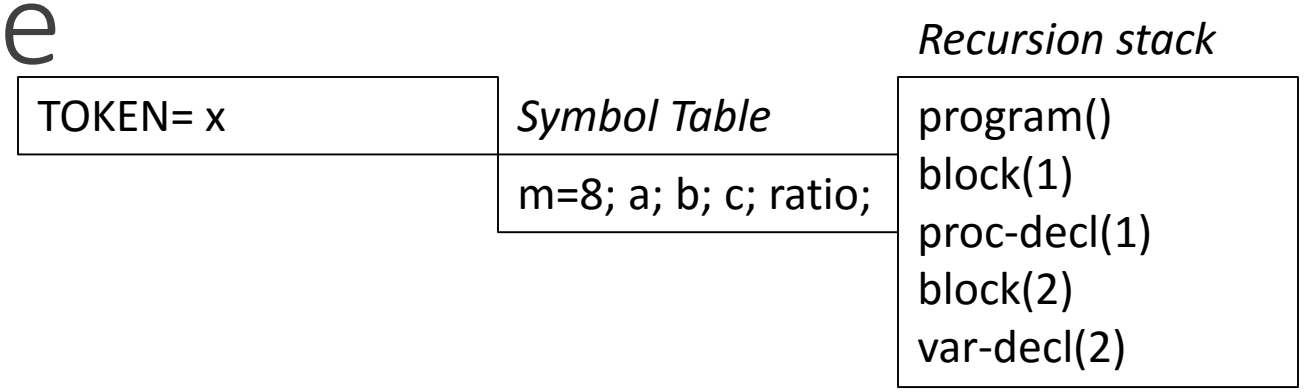
# Small Example




```
    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;
```

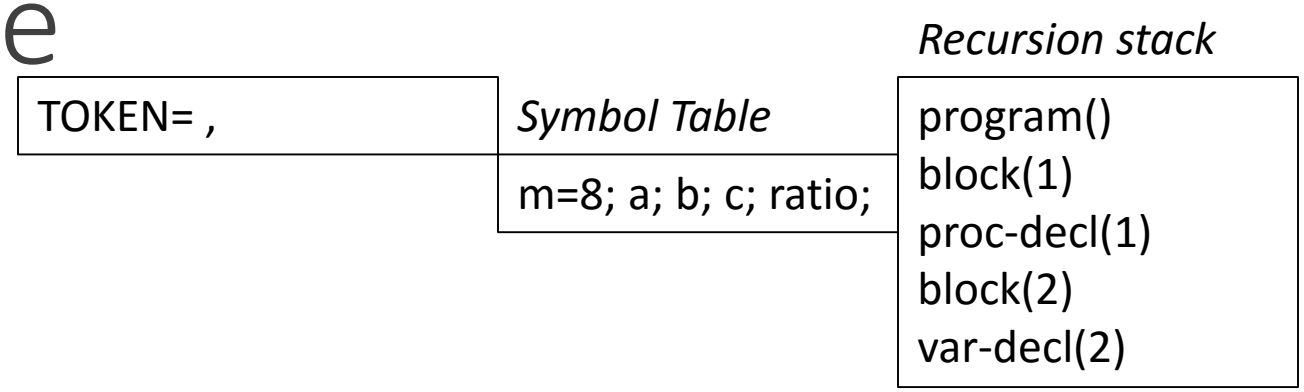
# Small Example




```
      , 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;
```

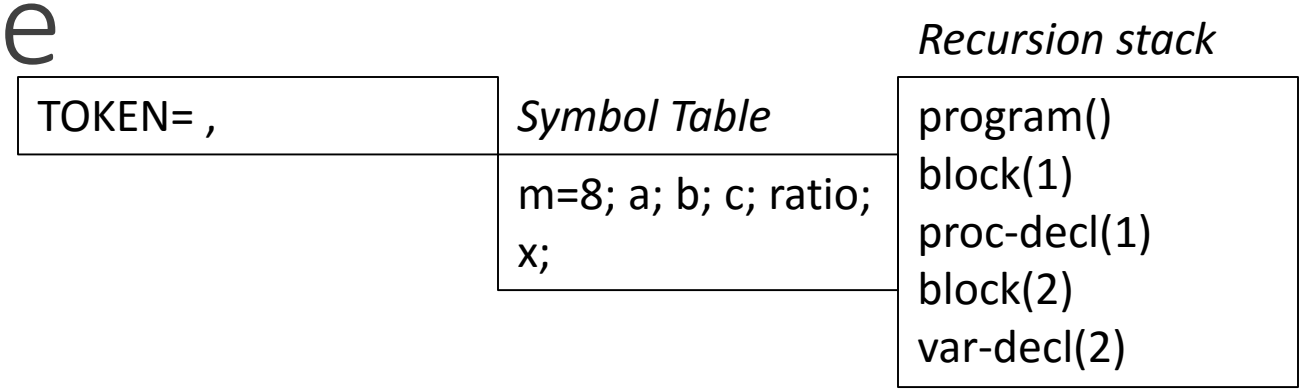
# Small Example




```
        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;
```

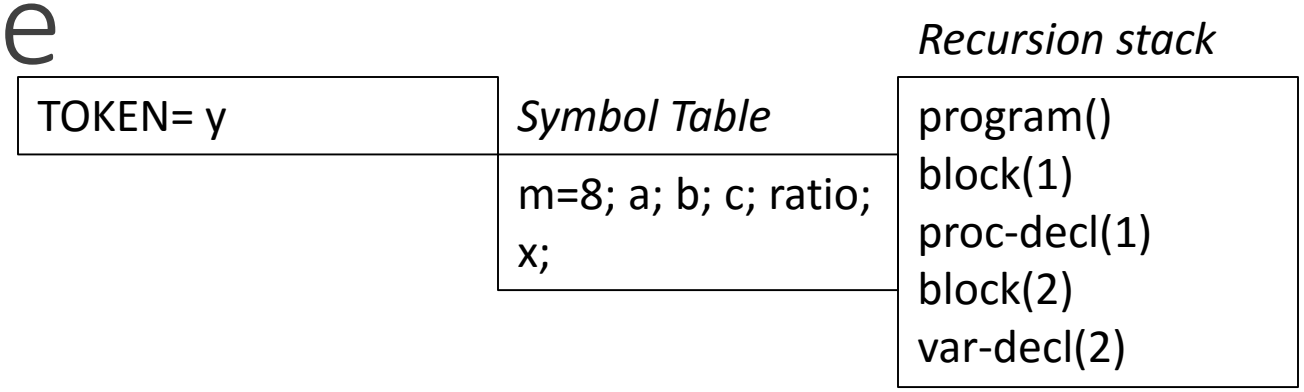
# Small Example




```
        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;
```

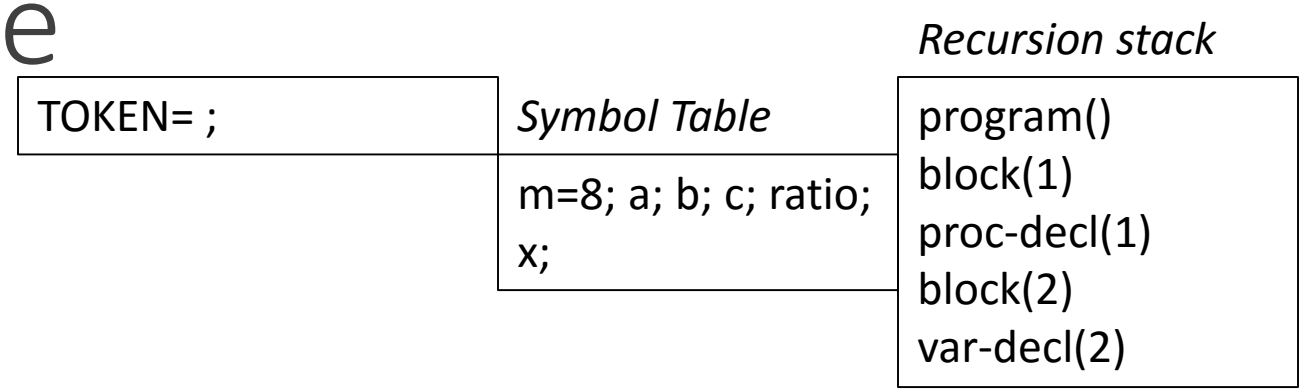
# Small Example




```
          ;
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;
```

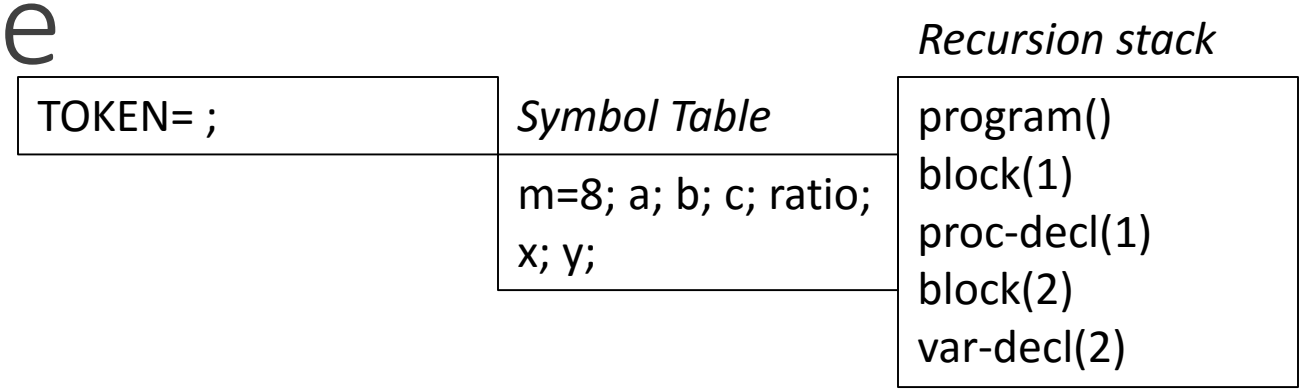
# Small Example




```
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;
```

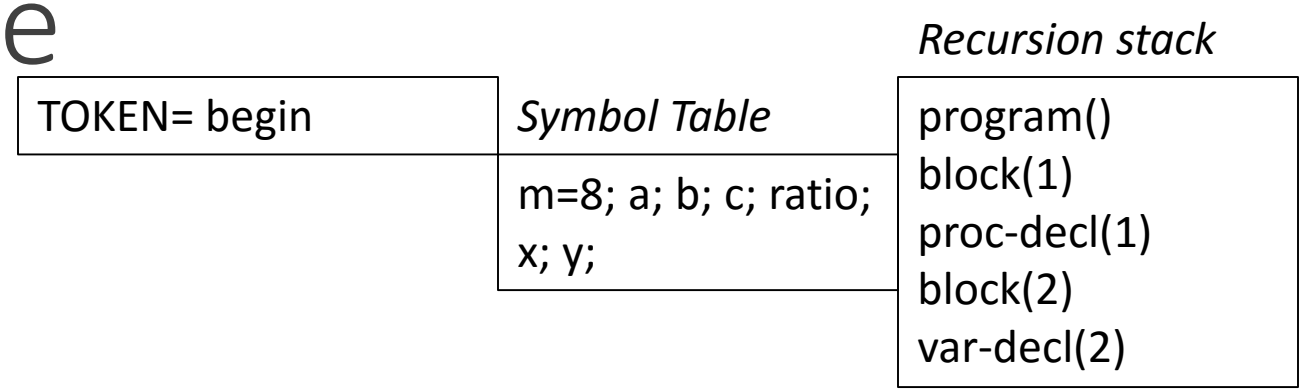
# Small Example




```
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;
```

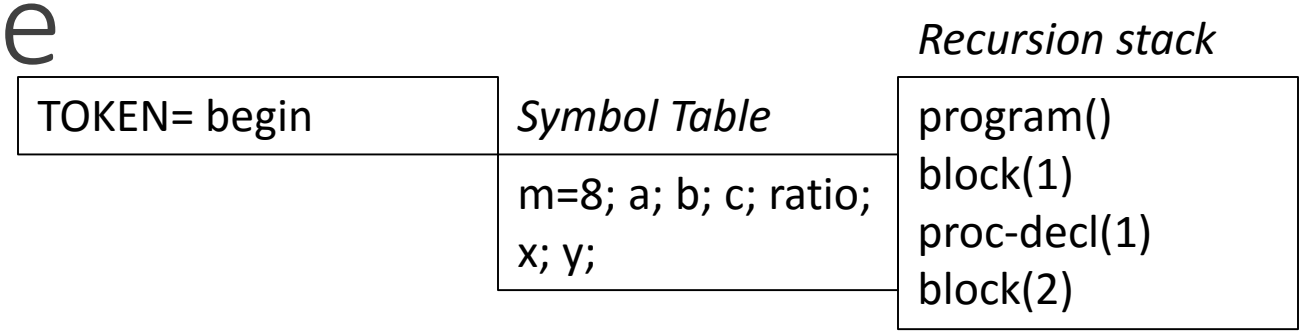
# Small Example




```
 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;
```

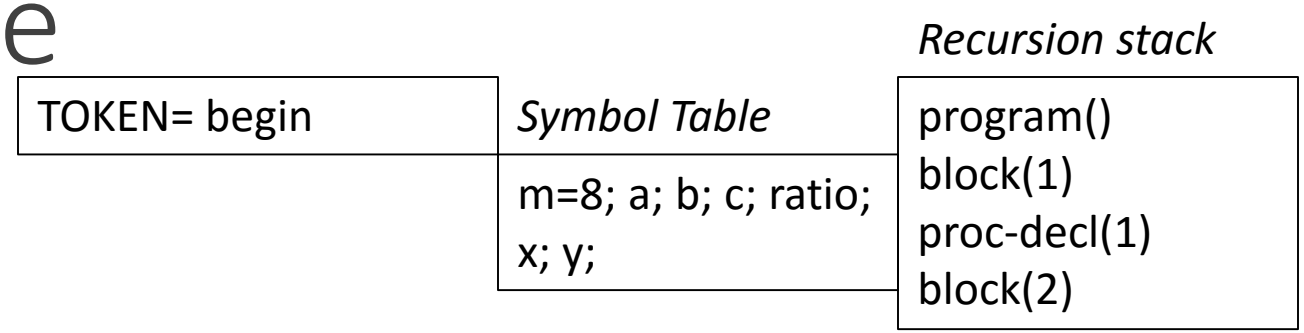
# Small Example




```
 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;
```

# Small Example




```
 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;
```

# Small Example

TOKEN= begin

*Symbol Table*

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

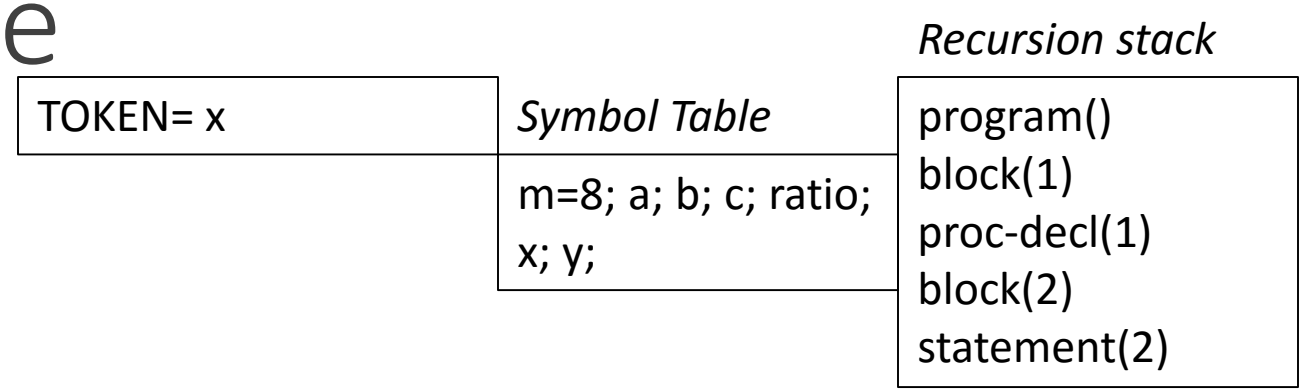
*Recursion stack*

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.
```

```
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;
…
```

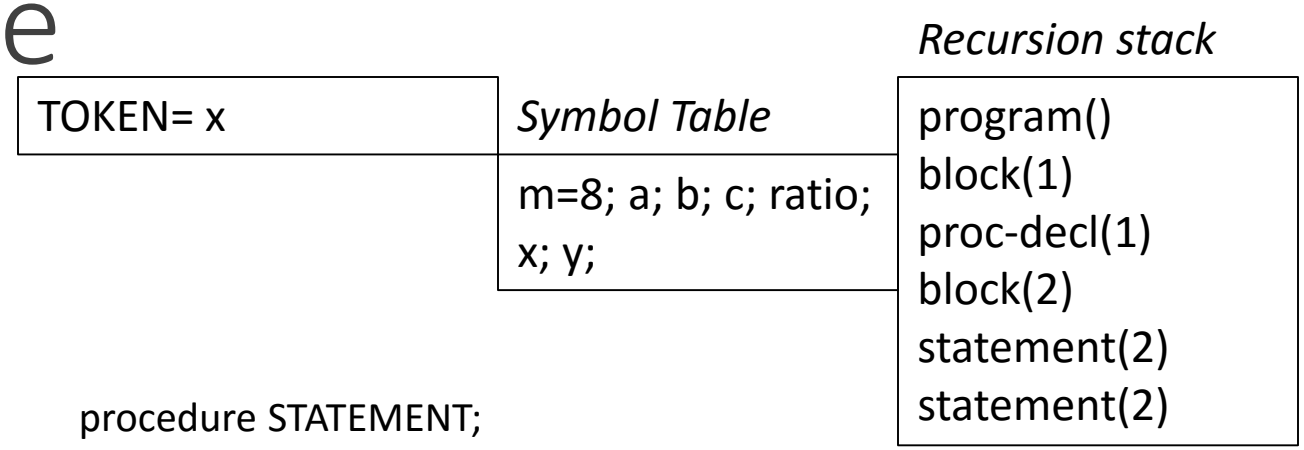
# Small Example




```
   = 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 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;
…
```

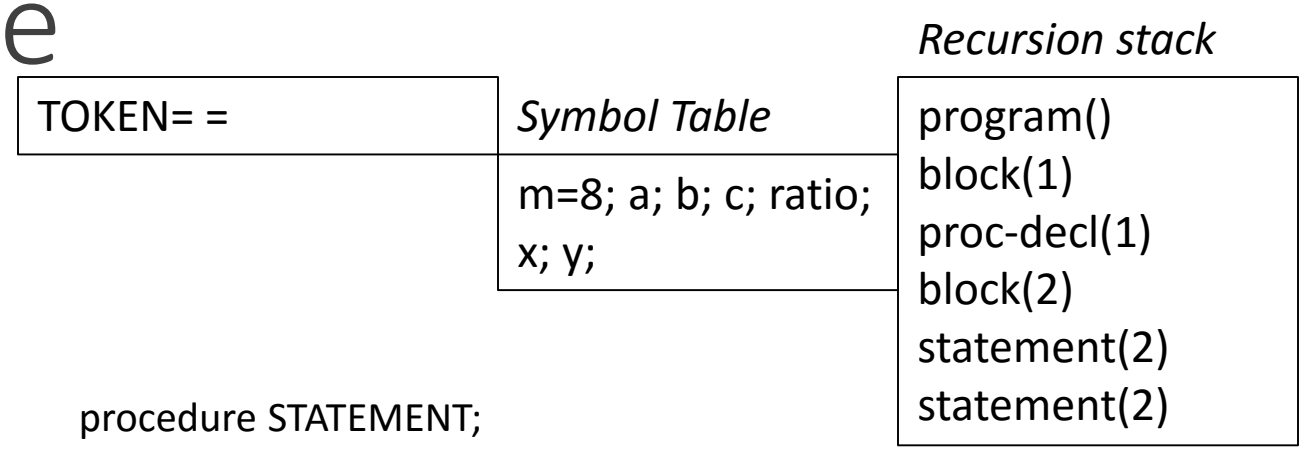
# Small Example




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

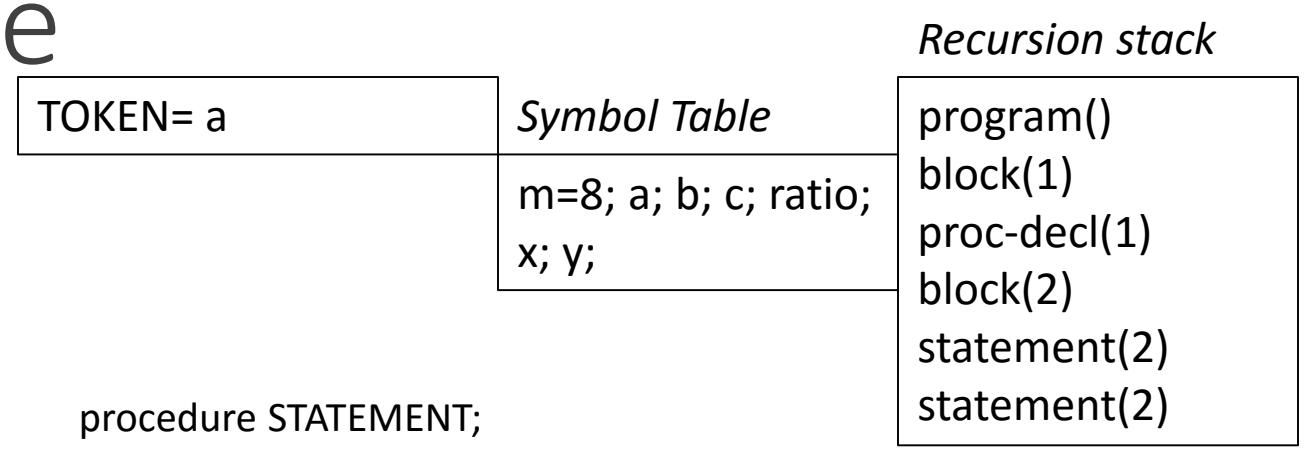
# Small Example




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

# Small Example




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

# Small Example

TOKEN= a

*Symbol Table*

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

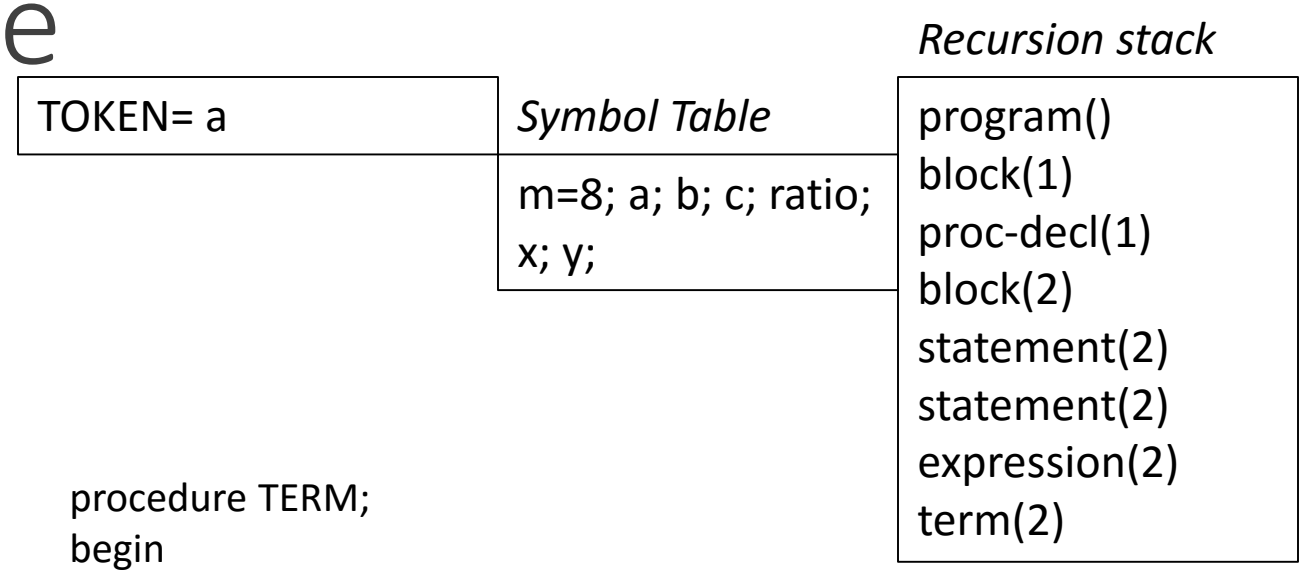
*Recursion stack*

program()
block(1)
proc-decl(1)
block(2)
statement(2)
statement(2)
expression(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 EXPRESSION;
begin
        if TOKEN = ADDING_OPERATOR then GET_TOKEN();
        TERM();
        while TOKEN = ADDING_OPERATOR do begin
                GET_TOKEN();
                TERM();
        end
end;
```

# Small Example




```
        ; 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 TERM;
begin
      FACTOR();
      while TOKEN = MULTIPLYING_OPERATOR do begin
            GET_TOKEN();
            FACTOR();
      end
end;
```

# Small Example

| TOKEN= a |
| --- |

*Symbol Table*

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

*Recursion stack*

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

# Small Example

| TOKEN= ; | *Symbol Table* |
| --- | --- |
| | m=8; a; b; c; ratio;<br>x; y; |

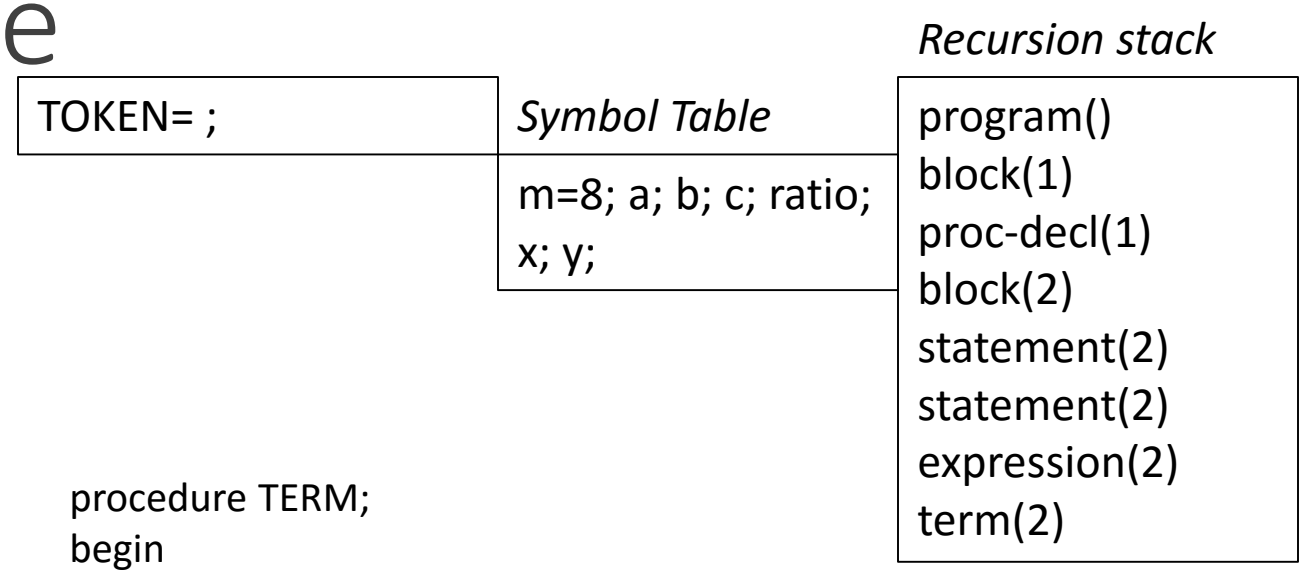
*Recursion stack*

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

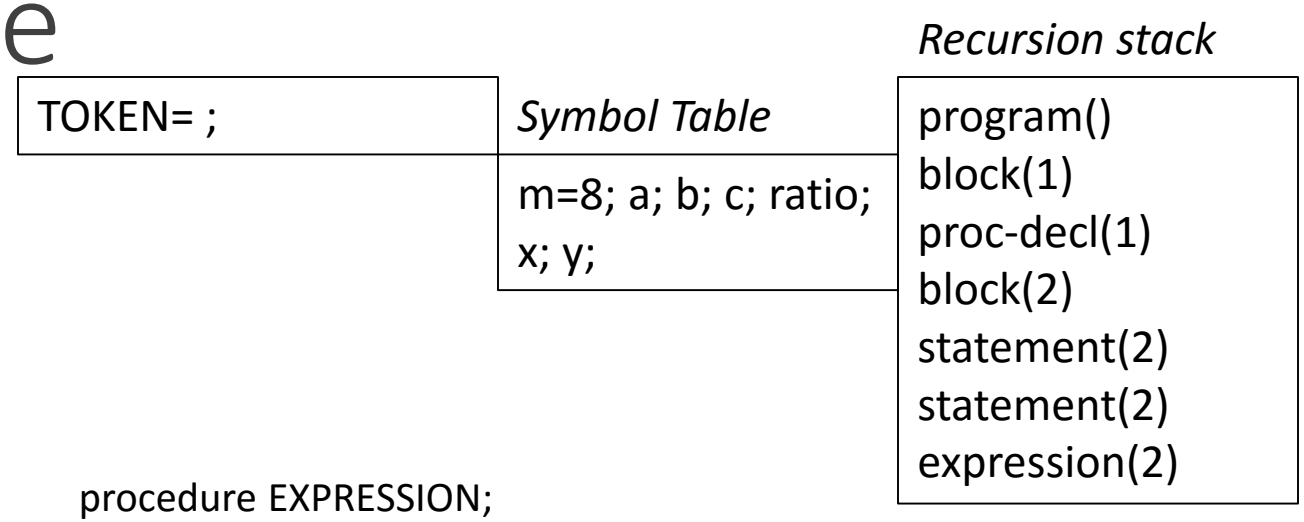
# Small Example




```
        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 TERM;
begin
    FACTOR();
    while TOKEN = MULTIPLYING_OPERATOR do begin
        GET_TOKEN();
        FACTOR();
    end
end;
```

# Small Example




```
        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 EXPRESSION;
begin
    if TOKEN = ADDING_OPERATOR then GET_TOKEN();
    TERM();
    while TOKEN = ADDING_OPERATOR do begin
        GET_TOKEN();
        TERM();
    end
end;
```

# Small Example

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

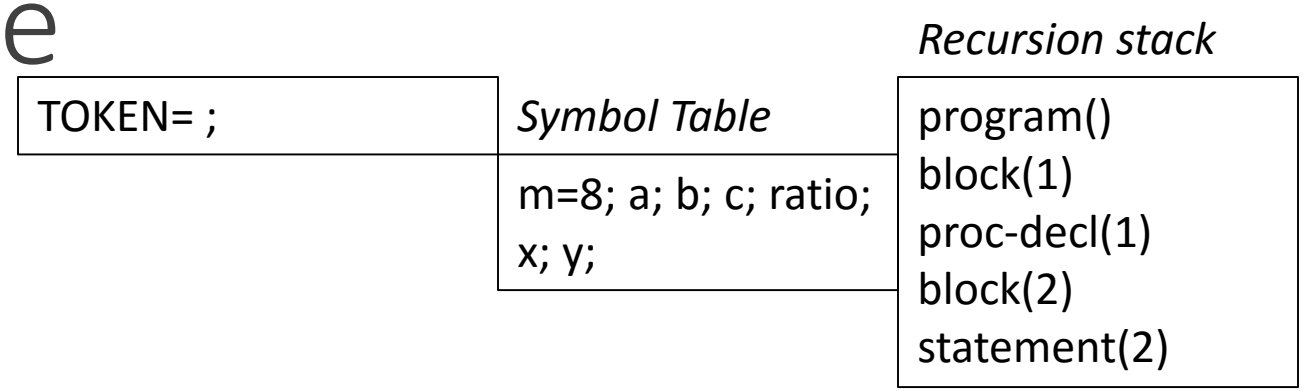
| TOKEN= ; | *Symbol Table* |
|---|---|
| | m=8; a; b; c; ratio;<br>x; y; |

*Recursion stack*

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

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

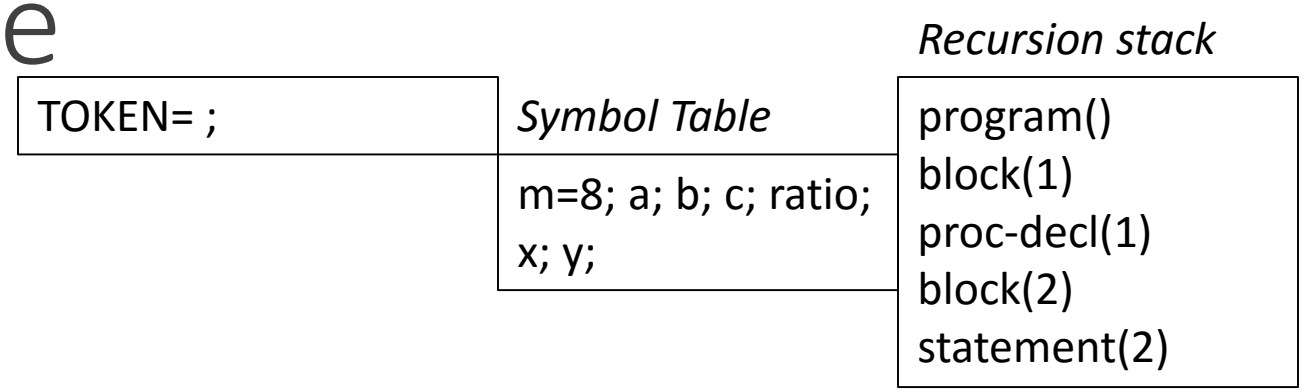
# Small Example




```
        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;
…
```

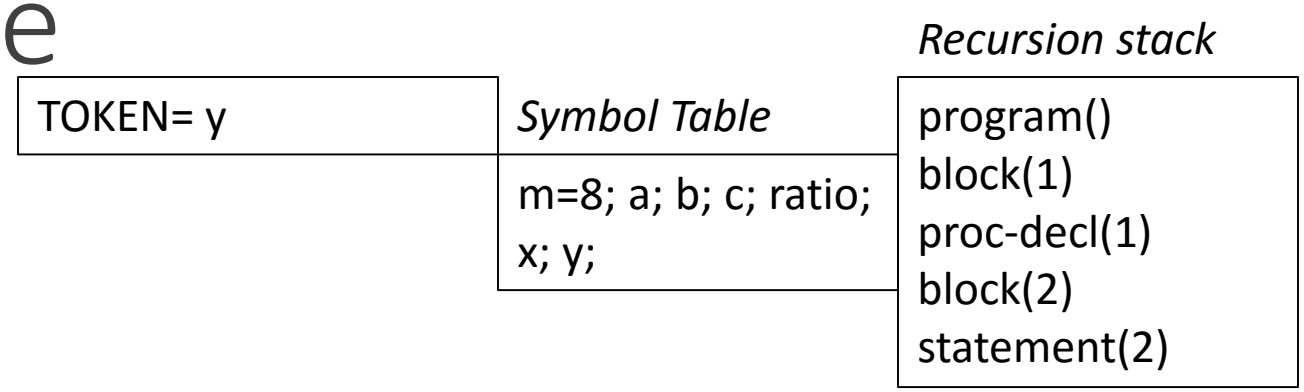
# Small Example




```
        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;
…
```

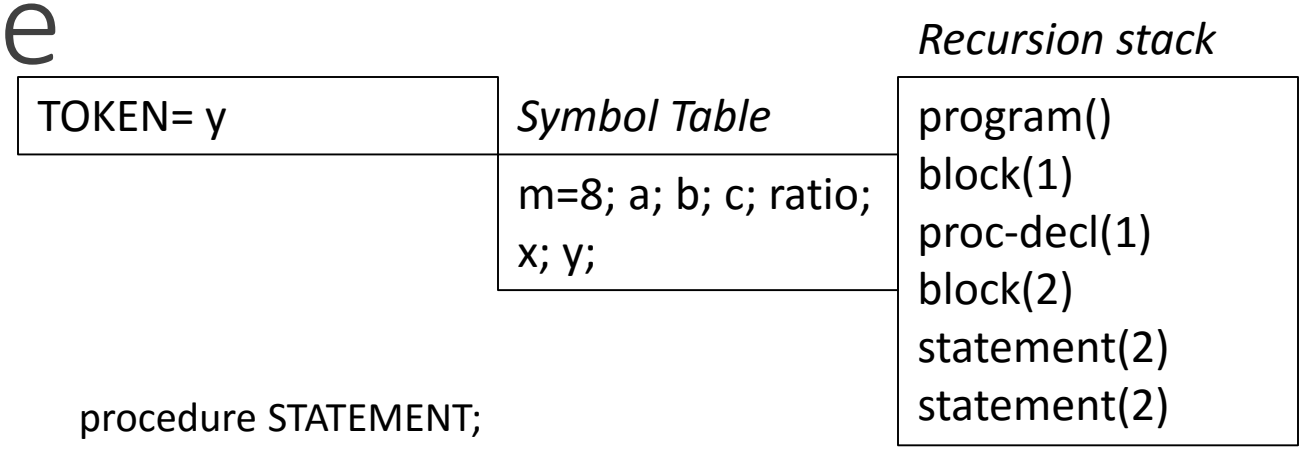
# Small Example




```
            = 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;
…
```

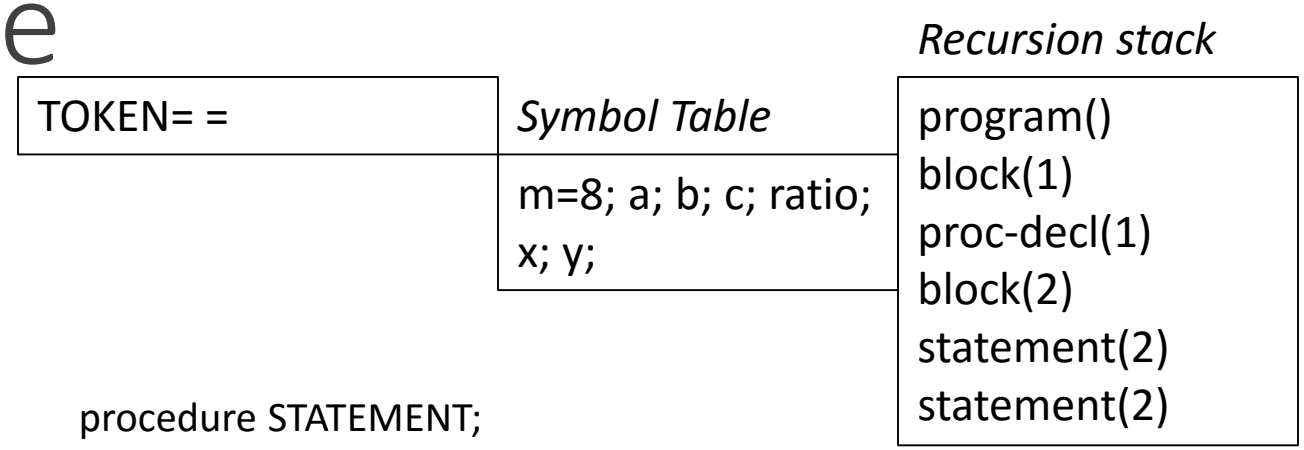
# Small Example




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

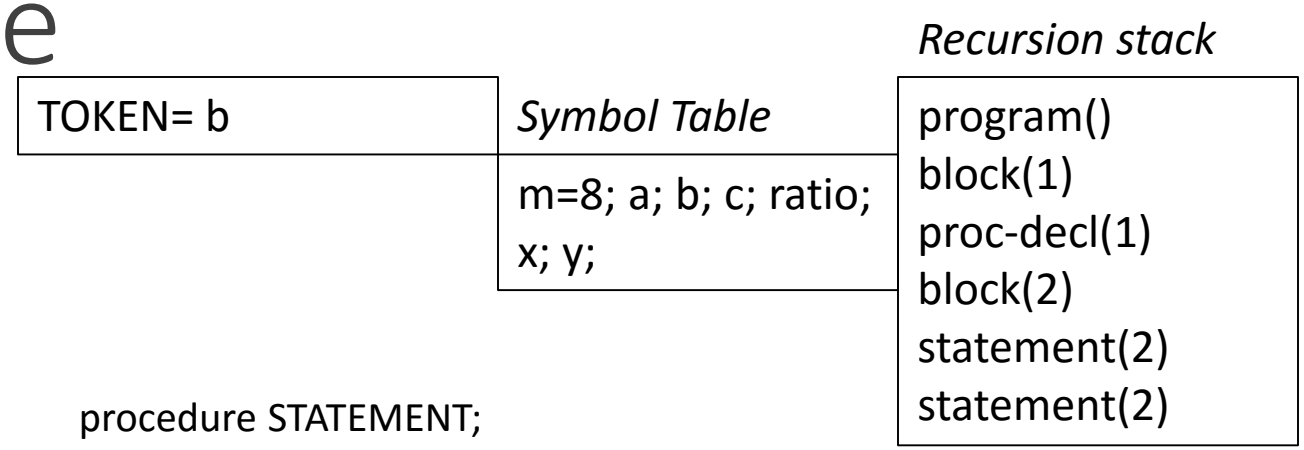
# Small Example




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

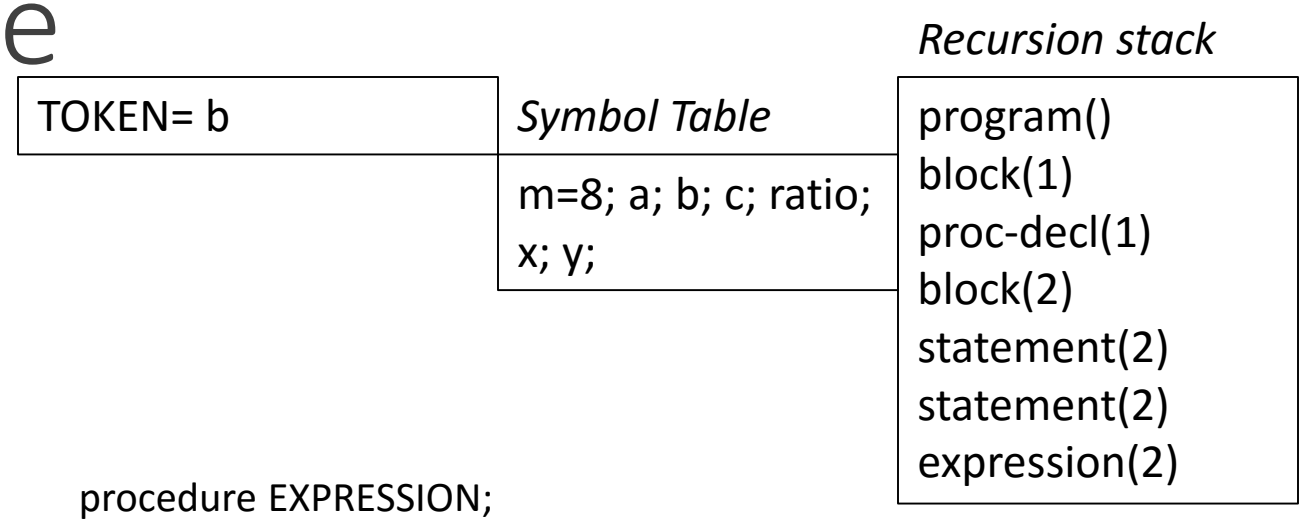
# Small Example




```
                ;
 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
        …
```

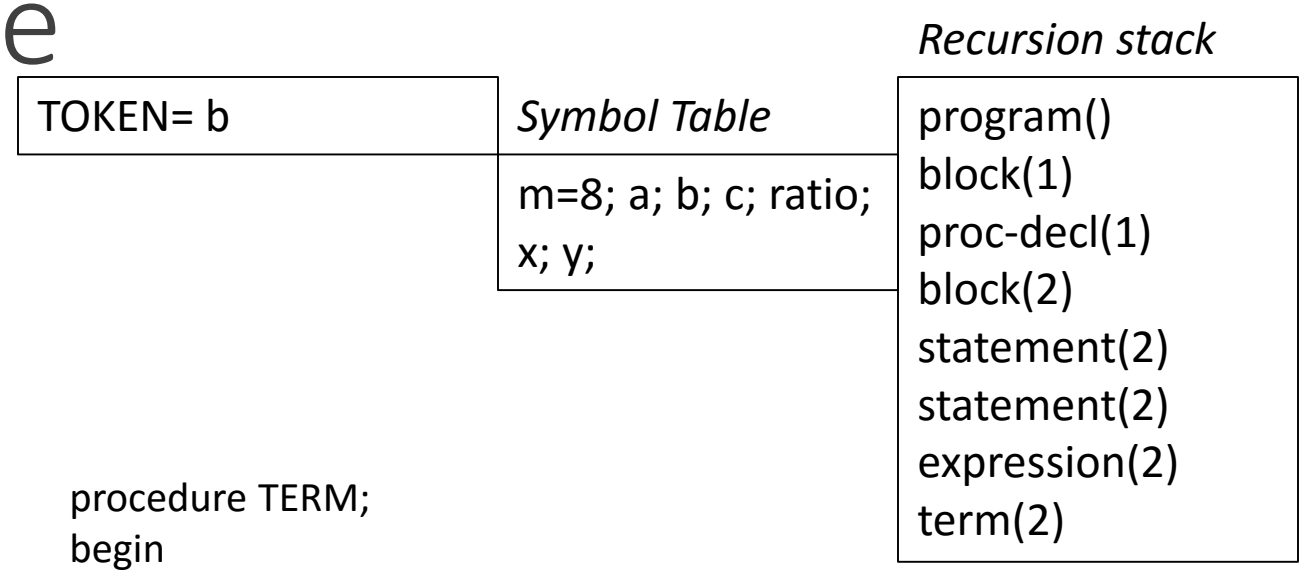
# Small Example




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

```
 procedure EXPRESSION;
begin
        if TOKEN = ADDING_OPERATOR then GET_TOKEN();
        TERM();
        while TOKEN = ADDING_OPERATOR do begin
                GET_TOKEN();
                TERM();
        end
end;
```

# Small Example




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

```
procedure TERM;
begin
    FACTOR();
    while TOKEN = MULTIPLYING_OPERATOR do begin
        GET_TOKEN();
        FACTOR();
    end
end;
```

# Small Example

| TOKEN= b |
| --- |

*Symbol Table*

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

*Recursion stack*

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

```
                ;
 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;
```

# Small Example

TOKEN= ;

*Symbol Table*

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

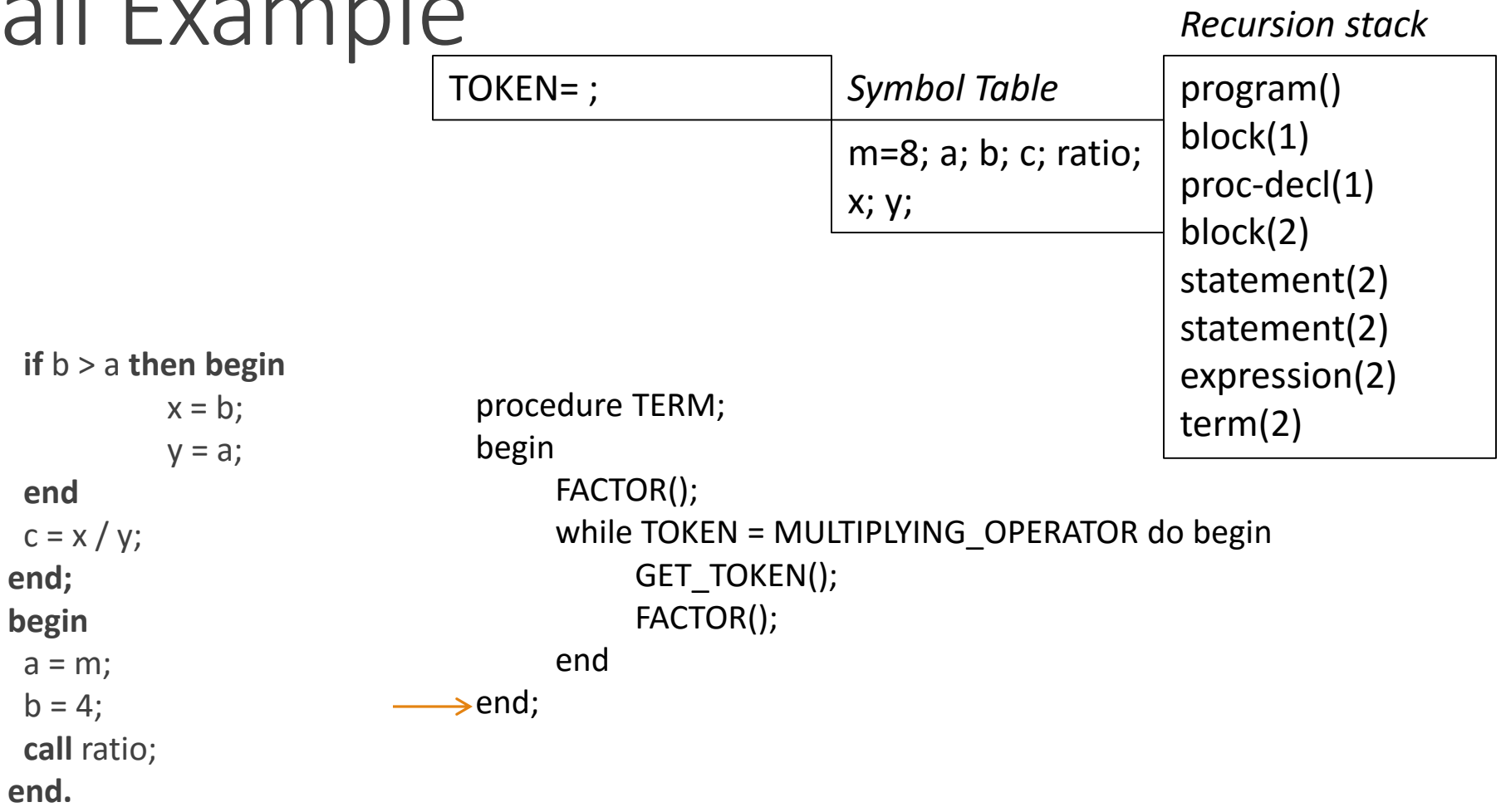
*Recursion stack*

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

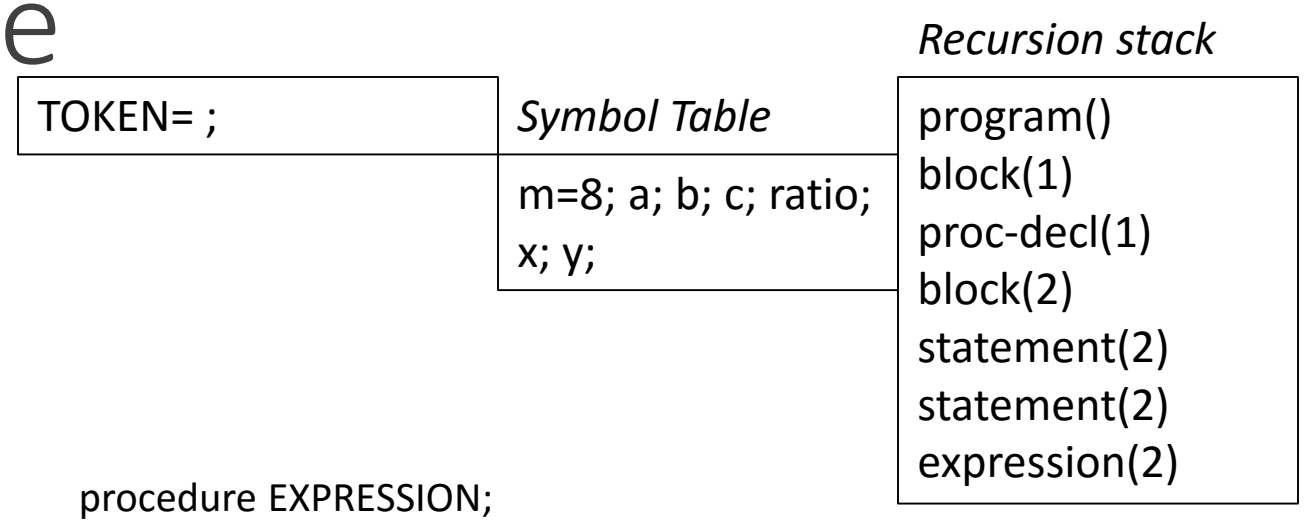
```
 if b > a then begin
              x = b;
              y = a;
 end
 c = x / y;
end;
begin
 a = m;
 b = 4;
 call ratio;
end.
```

```
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;
```

# Small Example



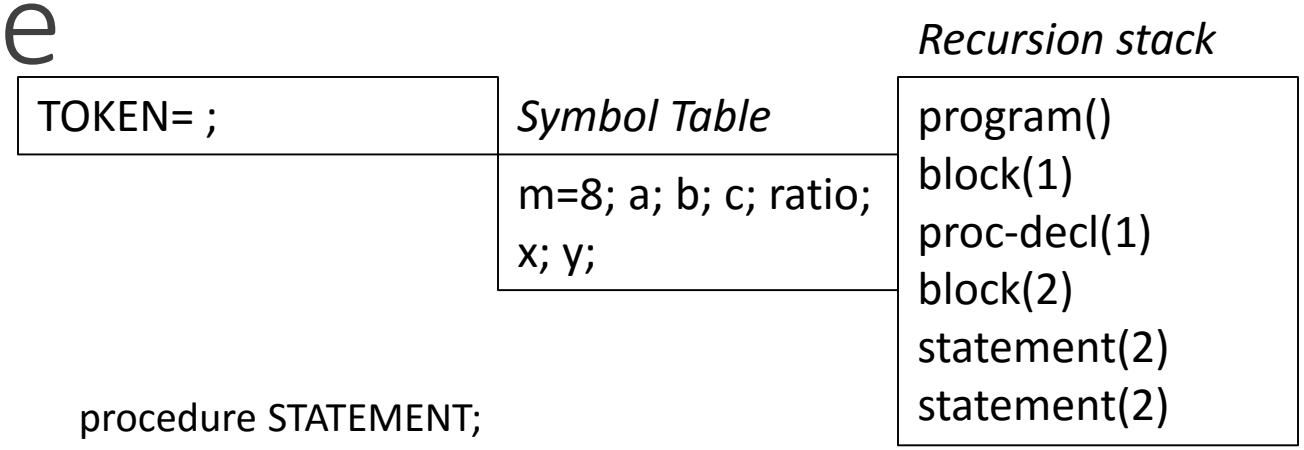
# Small Example




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

```
 procedure EXPRESSION;
begin
      if TOKEN = ADDING_OPERATOR then GET_TOKEN();
      TERM();
      while TOKEN = ADDING_OPERATOR do begin
            GET_TOKEN();
            TERM();
      end
end;
```

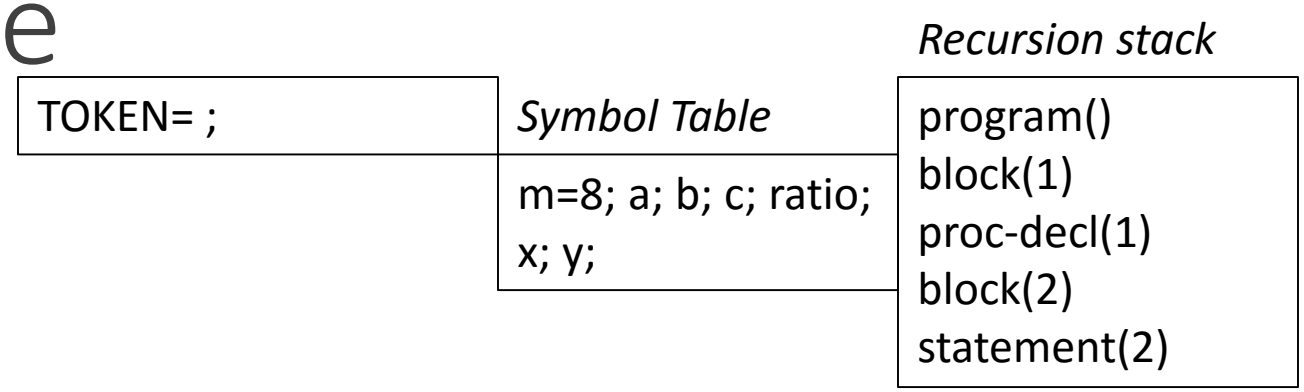
# Small Example




```
 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
      …
```

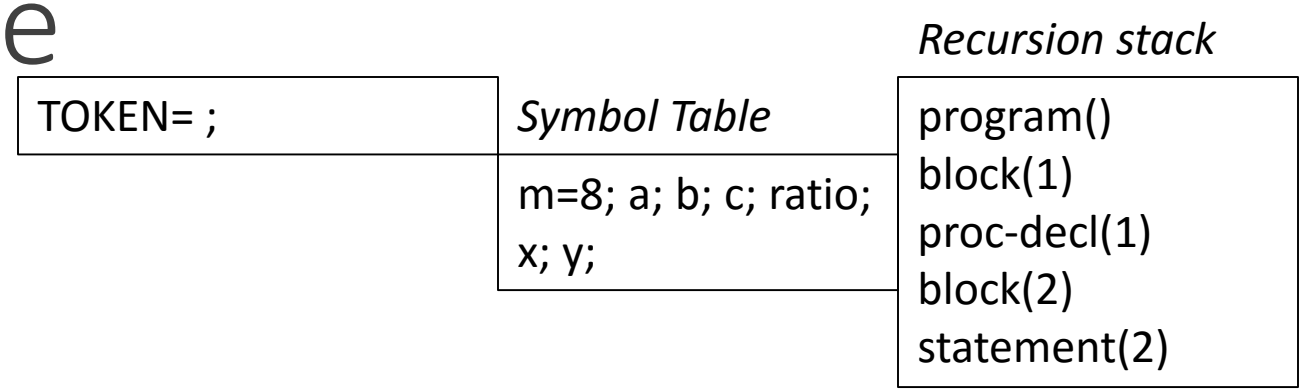
# Small Example




```
 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;
…
```

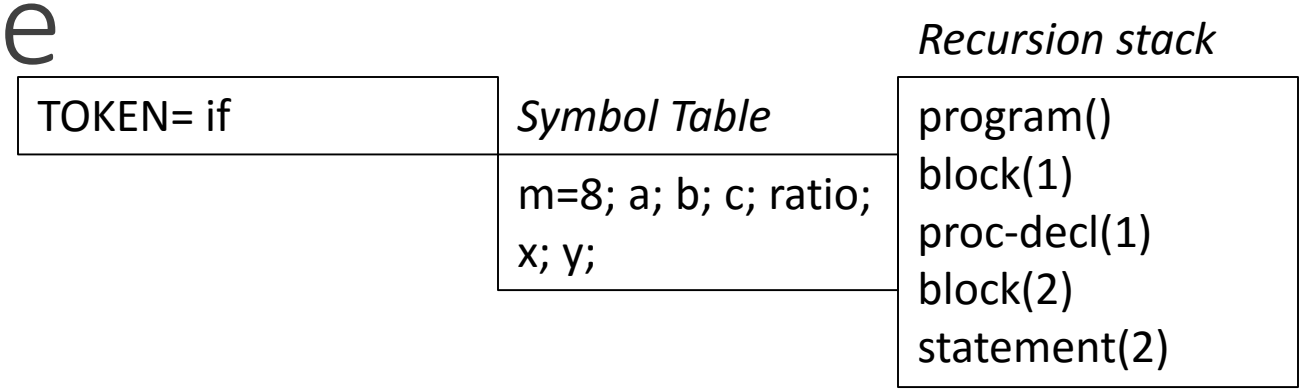
# Small Example




```
 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;
…
```

# Small Example




```
    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;
…
```

# Small Example

| TOKEN= if |
| --- |

*Symbol Table*

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

*Recursion stack*

| program() |
| --- |
| block(1) |
| proc-decl(1) |
| block(2) |
| statement(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 = "if" then begin
        ⟶GET_TOKEN();
                CONDITION();
                if TOKEN <> "then" then ERROR ();
                GET_TOKEN();
                STATEMENT();
        end;
…
```

# Small Example

| TOKEN= b |
| --- |

*Symbol Table*

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

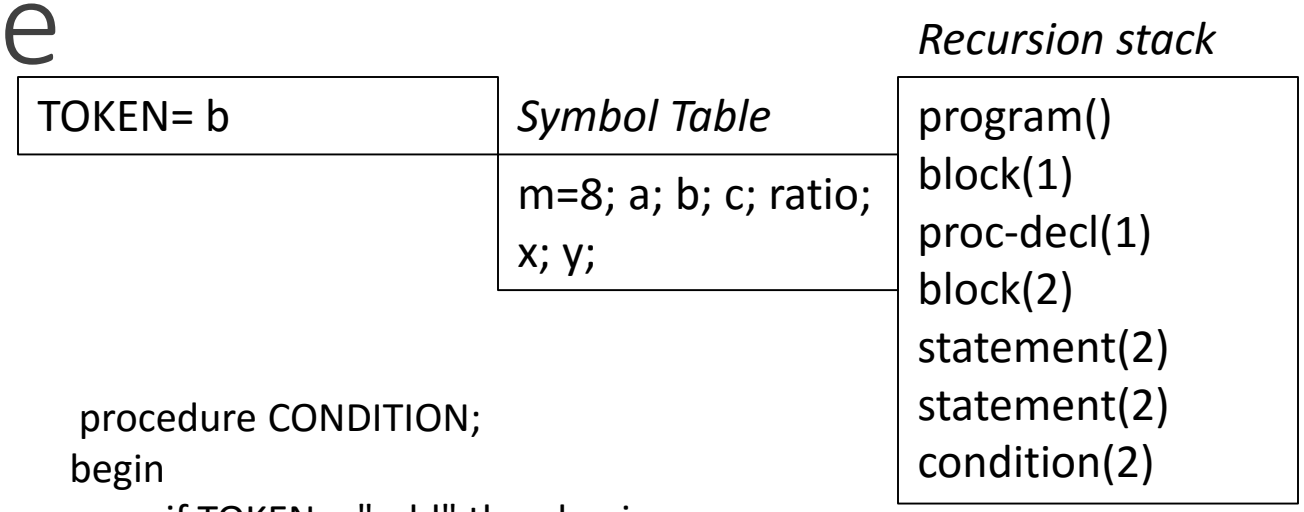
*Recursion stack*

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

```
      > 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 = "if" then begin
            GET_TOKEN();
            CONDITION();
            if TOKEN <> "then" then ERROR ();
            GET_TOKEN();
            STATEMENT();
      end;
…
```

# Small Example




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

```
 procedure CONDITION;
begin
    if TOKEN = "odd" then begin
            GET_TOKEN();
            EXPRESSION();
        else begin
            EXPRESSION();
            if TOKEN <> RELATION then ERROR ();
            GET_TOKEN();
            EXPRESSION();
        end
end;
```

# Small Example

| TOKEN= b | *Symbol Table* |
| --- | --- |
| | m=8; a; b; c; ratio;<br>x; y; |

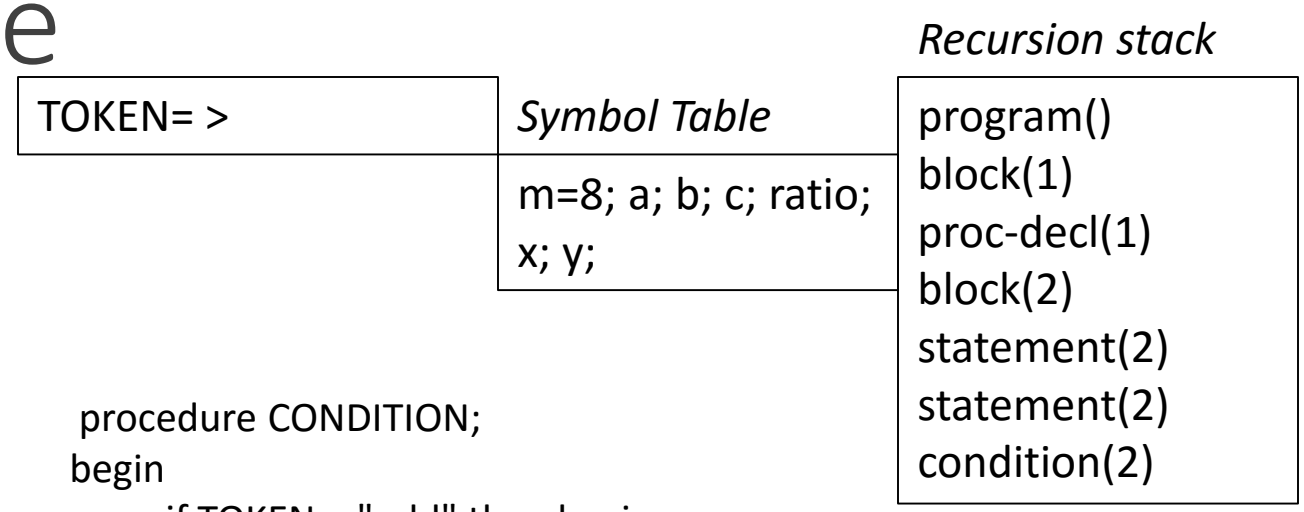
*Recursion stack*

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

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

```
 procedure CONDITION;
begin
        if TOKEN = "odd" then begin
                GET_TOKEN();
                EXPRESSION();
        else begin
        ⟶      EXPRESSION();
                if TOKEN <> RELATION then ERROR ();
                GET_TOKEN();
                EXPRESSION();
        end
end;
```

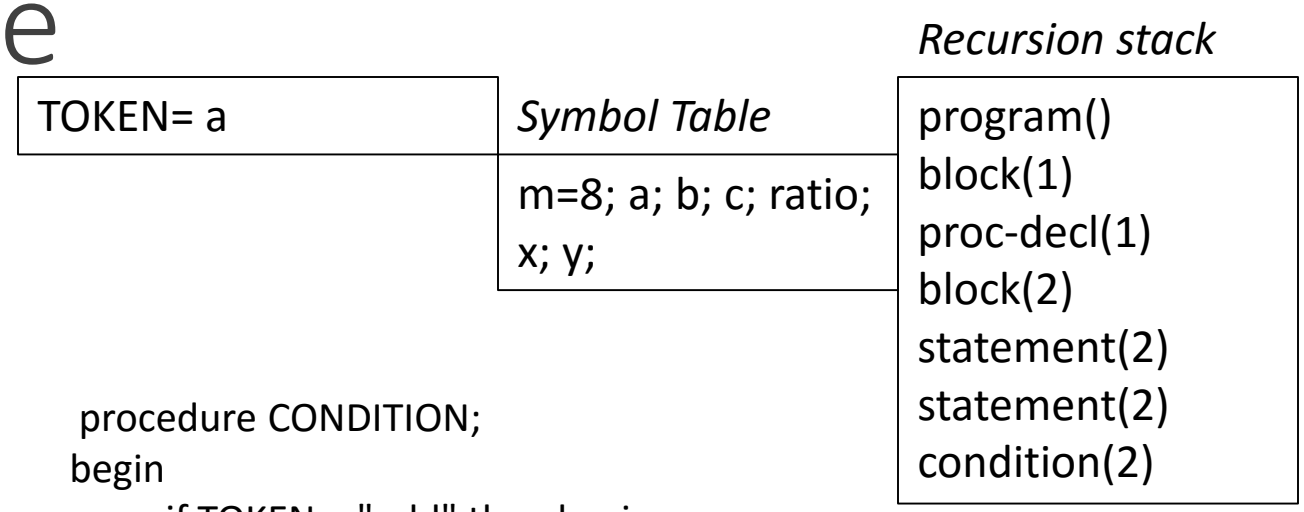
# Small Example




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

```
 procedure CONDITION;
begin
        if TOKEN = "odd" then begin
                GET_TOKEN();
                EXPRESSION();
        else begin
                EXPRESSION();
                if TOKEN <> RELATION then ERROR ();
        ⟶      GET_TOKEN();
                EXPRESSION();
        end
end;
```

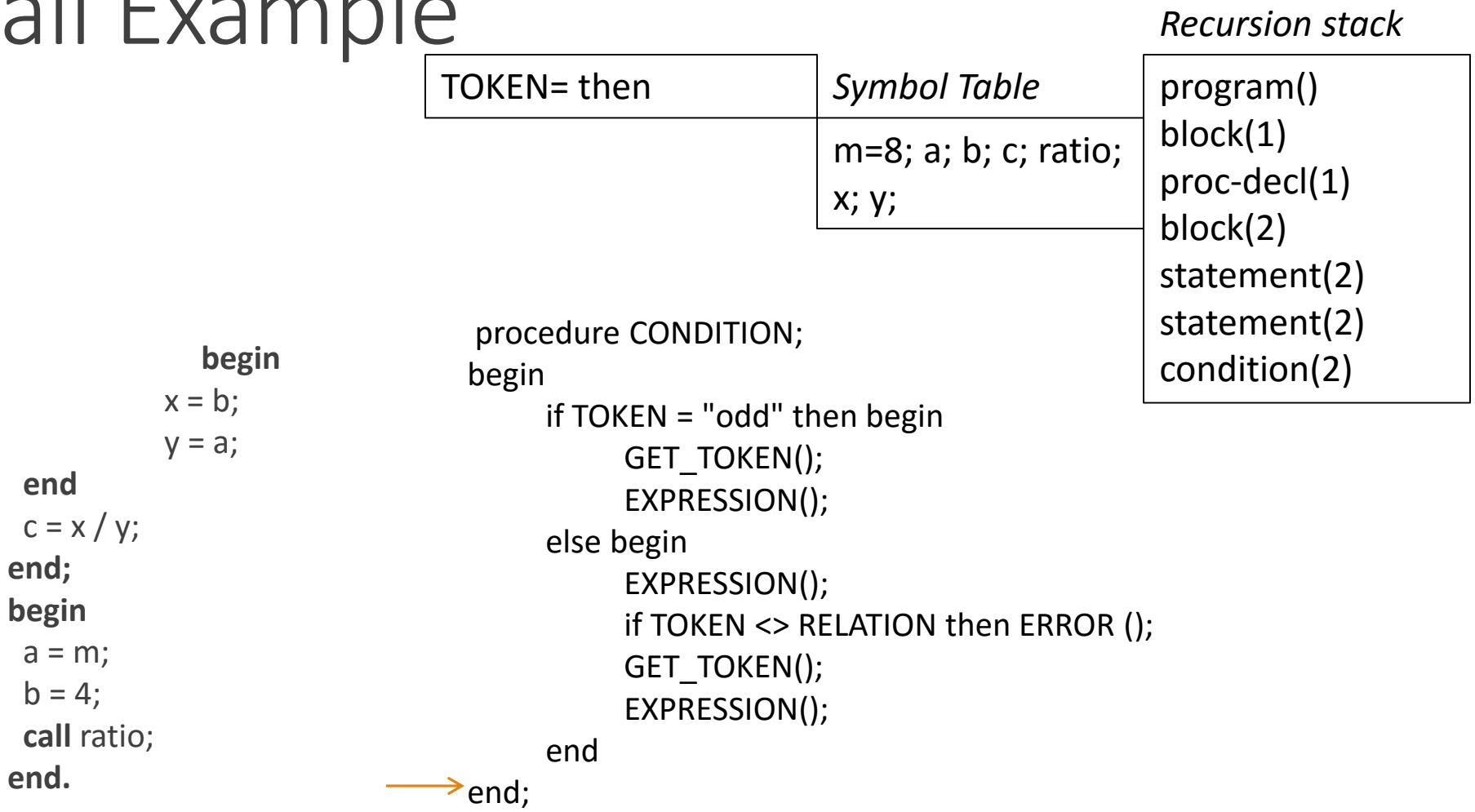
# Small Example



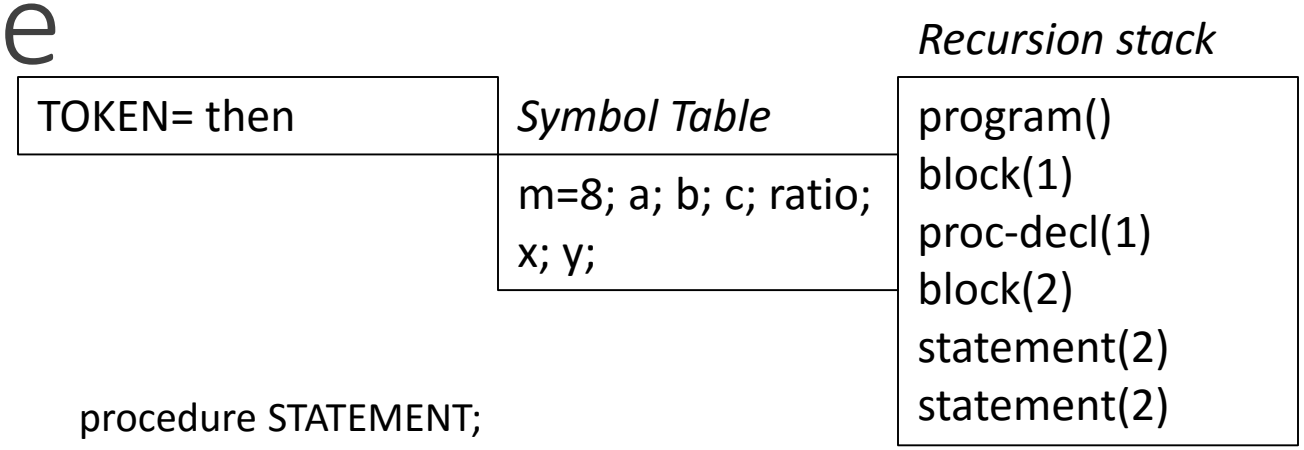

```
        then begin
            x = b;
            y = a;
  end
  c = x / y;
end;
begin
  a = m;
  b = 4;
  call ratio;
end.
```

```
procedure CONDITION;
begin
        if TOKEN = "odd" then begin
                GET_TOKEN();
                EXPRESSION();
        else begin
                EXPRESSION();
                if TOKEN <> RELATION then ERROR ();
                GET_TOKEN();
        ⟶      EXPRESSION();
        end
end;
```

# Small Example



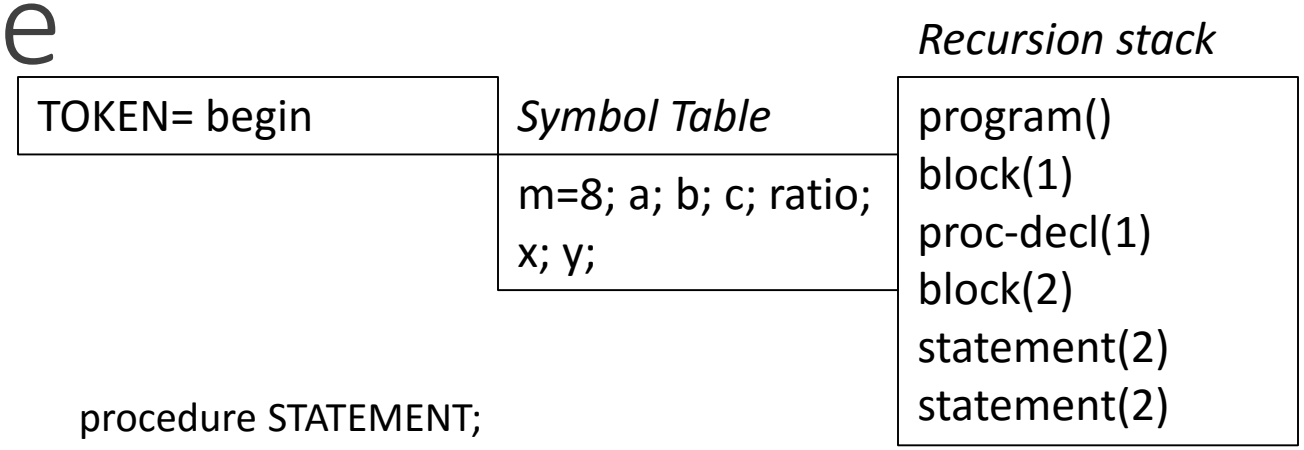
# Small Example




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

```
procedure STATEMENT;
…
      else if TOKEN = "if" then begin
            GET_TOKEN();
            CONDITION();
            if TOKEN <> "then" then ERROR ();
            GET_TOKEN();
            STATEMENT();
      end;
…
```

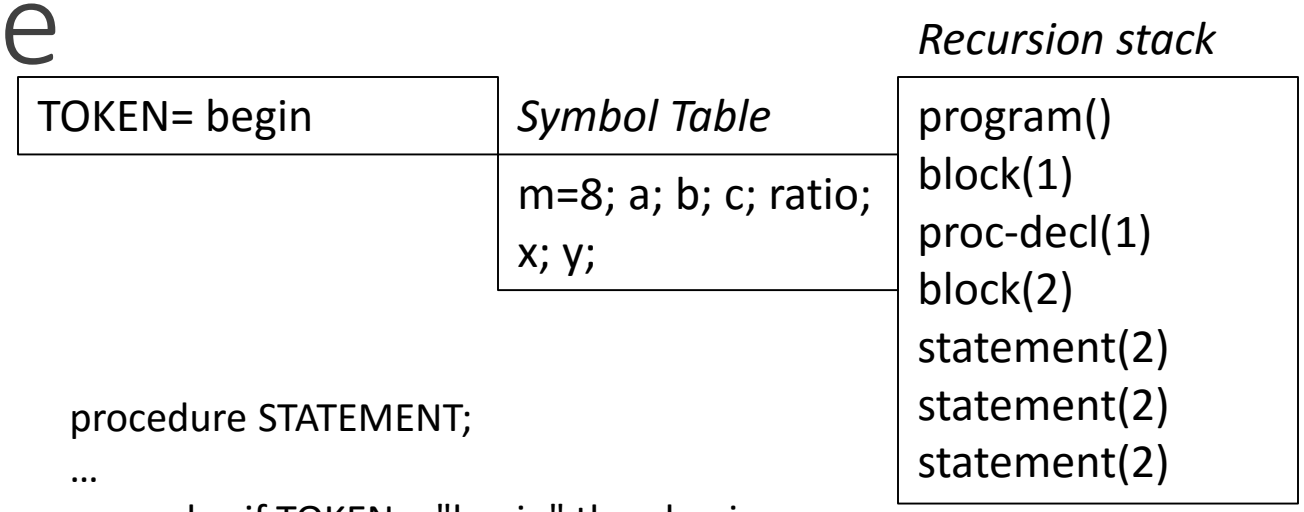
# Small Example




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

```
procedure STATEMENT;
…
        else if TOKEN = "if" then begin
                GET_TOKEN();
                CONDITION();
                if TOKEN <> "then" then ERROR ();
                GET_TOKEN();
                STATEMENT();
        end;
…
```

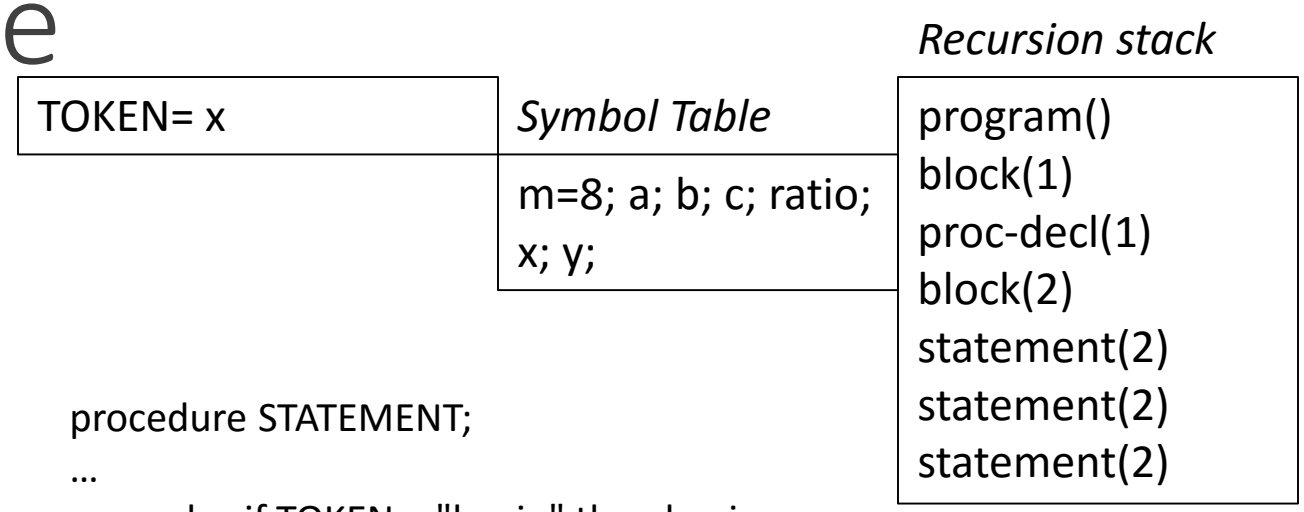
# Small Example




```
            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;
…
```

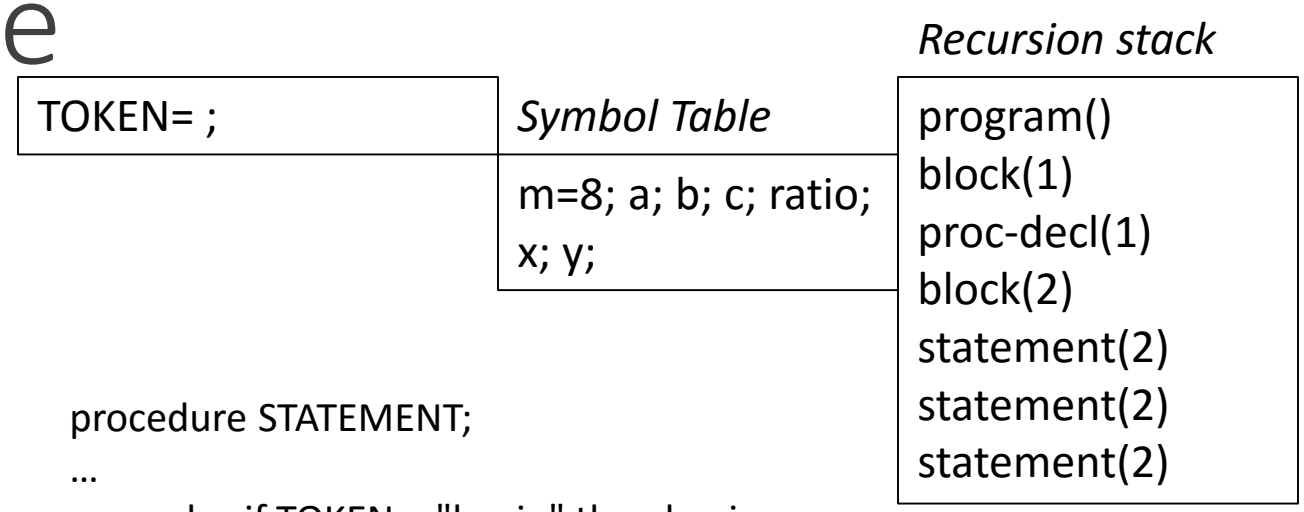
# Small Example




```
                = 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;
…
```

# Small Example




```
            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;
…
```

# Small Example

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

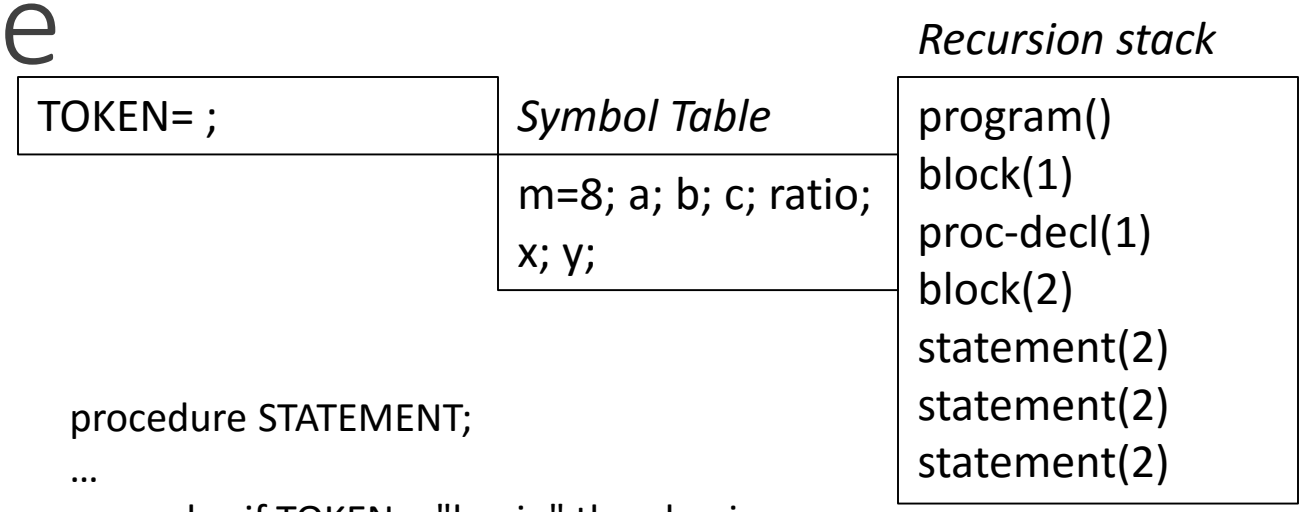
| TOKEN= y | *Symbol Table* |
|---|---|
| | m=8; a; b; c; ratio;<br>x; y; |

*Recursion stack*

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

```
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;
…
```

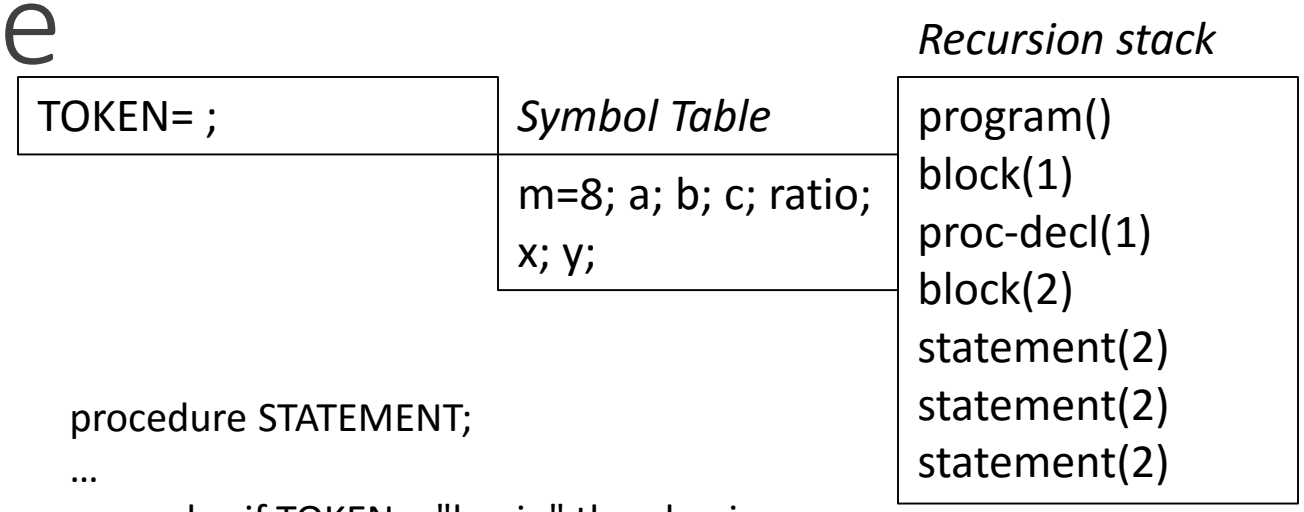
# Small Example




```
  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;
…
```

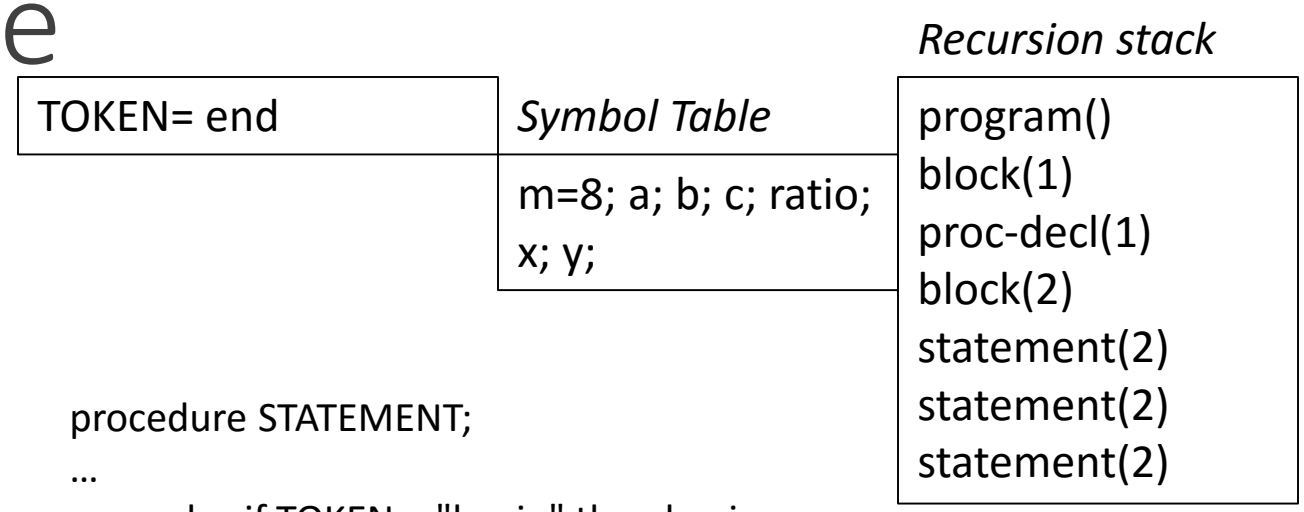
# Small Example




```
  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;
…
```

# Small Example




```
  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;
…
```

# Small Example

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

| TOKEN= end |
|---|

| *Symbol Table* |
|---|
| m=8; a; b; c; ratio; x; y; |

*Recursion stack*

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

```
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;
…
```

# Small Example

TOKEN= ;

*Symbol Table*

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

*Recursion stack*

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.
```

```
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;
…
```

# Small Example

| TOKEN= ; | *Symbol Table* |
| --- | --- |
| | m=8; a; b; c; ratio;<br>x; y; |

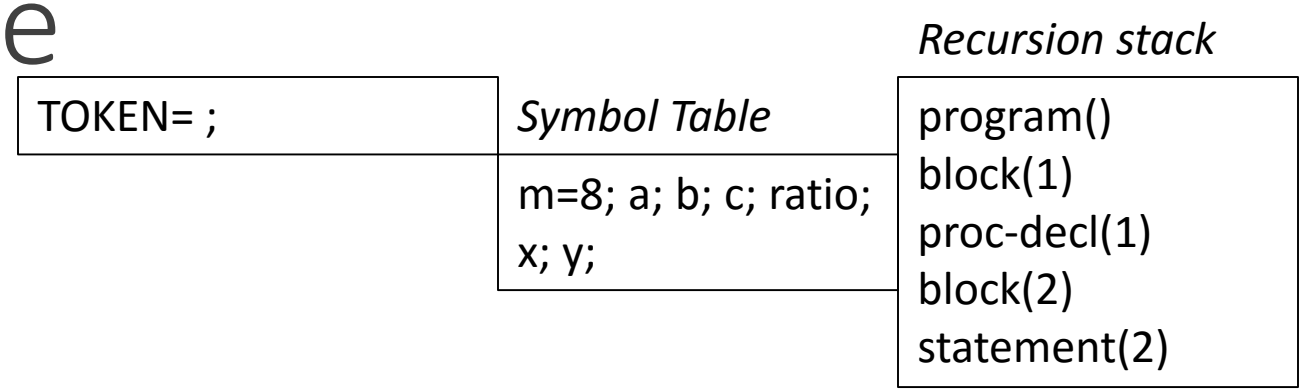
*Recursion stack*

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 = "if" then begin
                GET_TOKEN();
                CONDITION();
                if TOKEN <> "then" then ERROR ();
                GET_TOKEN();
                STATEMENT();
        end;
…
```

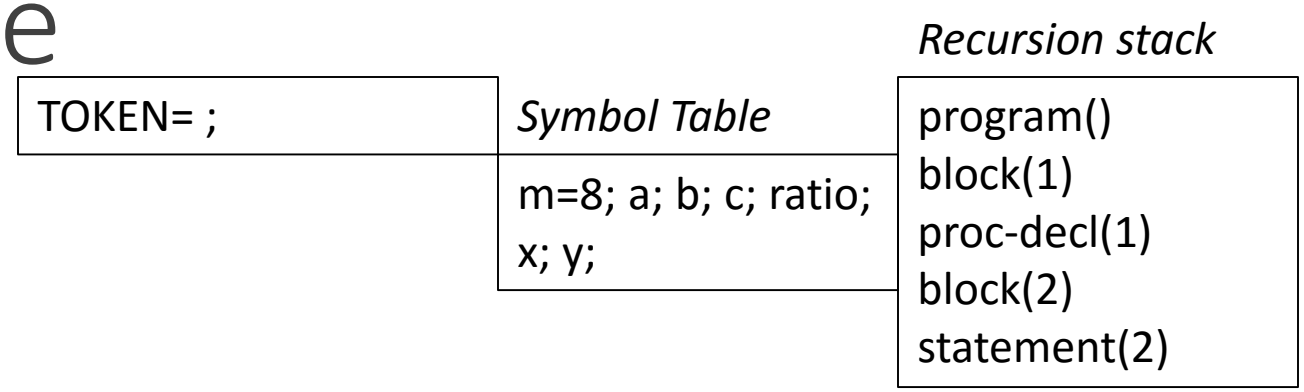
# Small Example




```
  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;
…
```

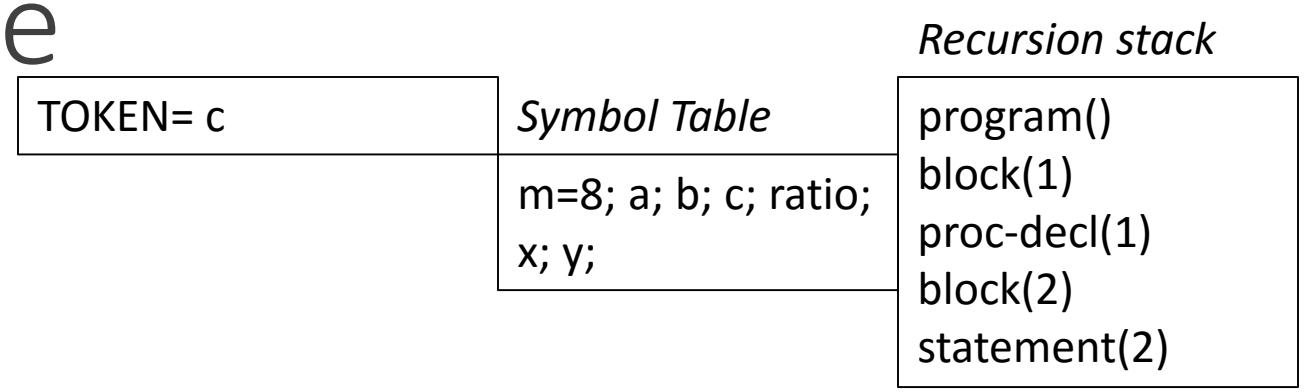
# Small Example




```
  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;
…
```

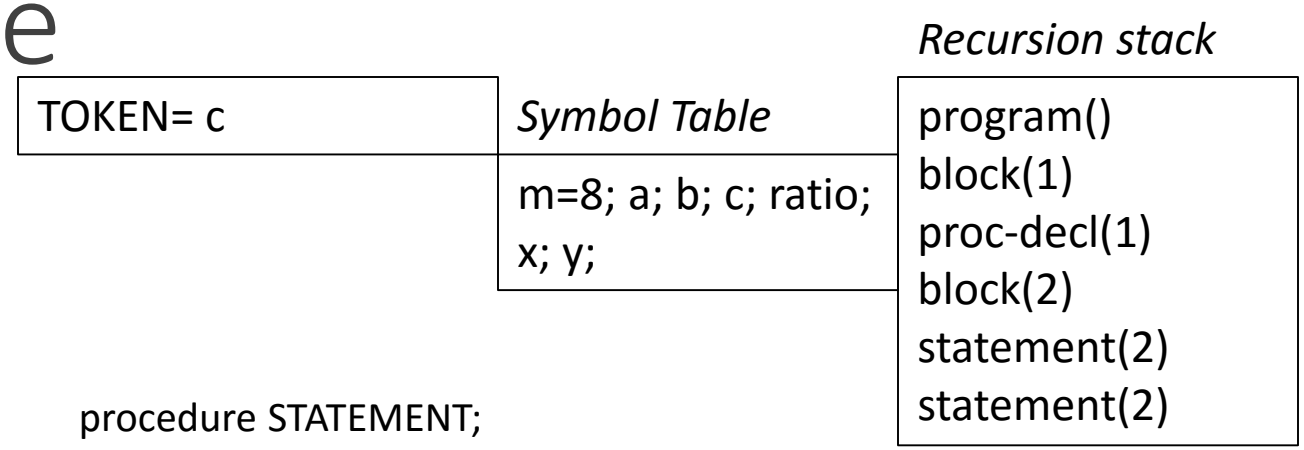
# Small Example




```
    = 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;
…
```

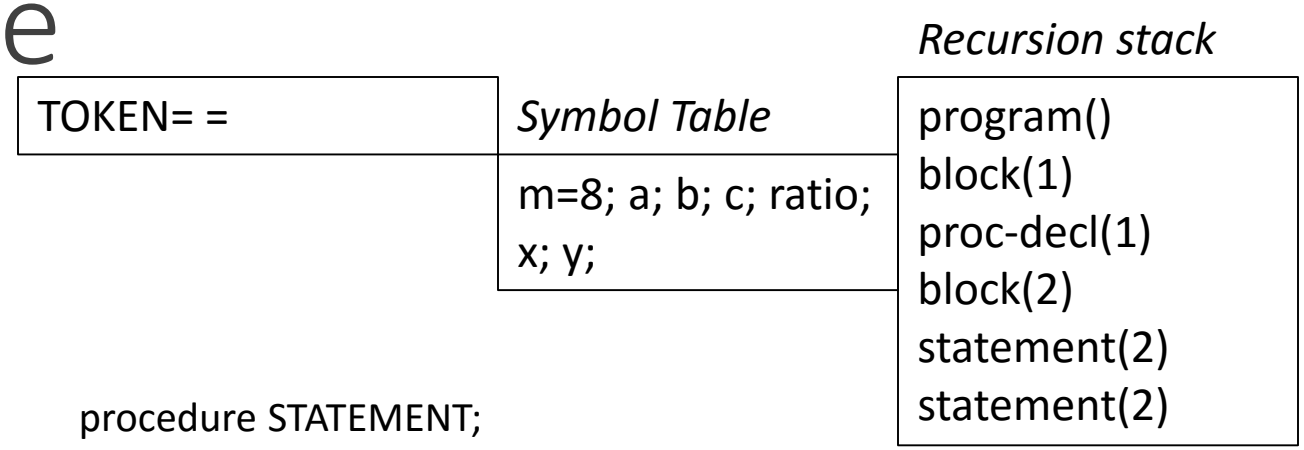
# Small Example




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

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

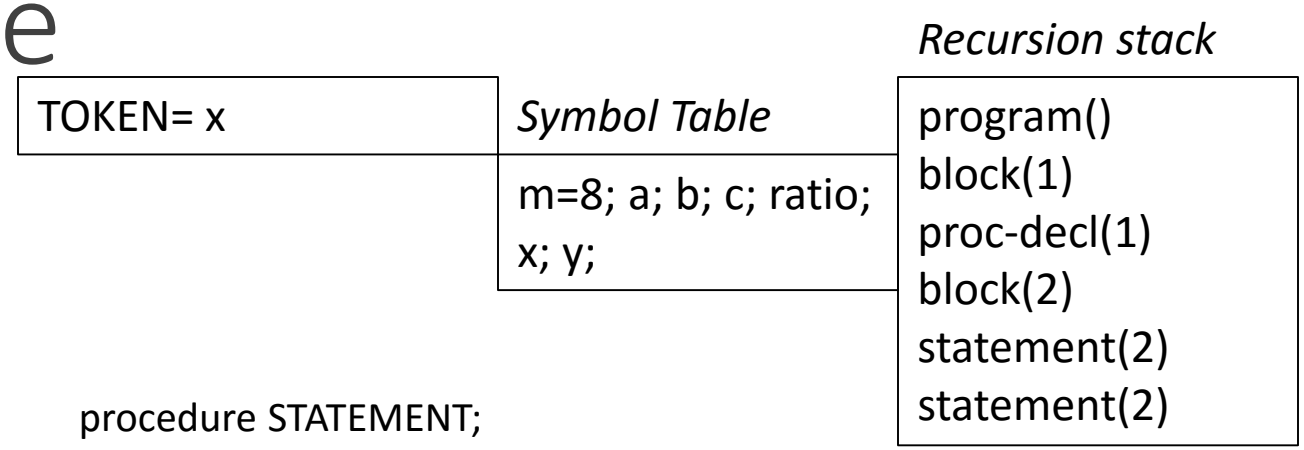
# Small Example




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

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

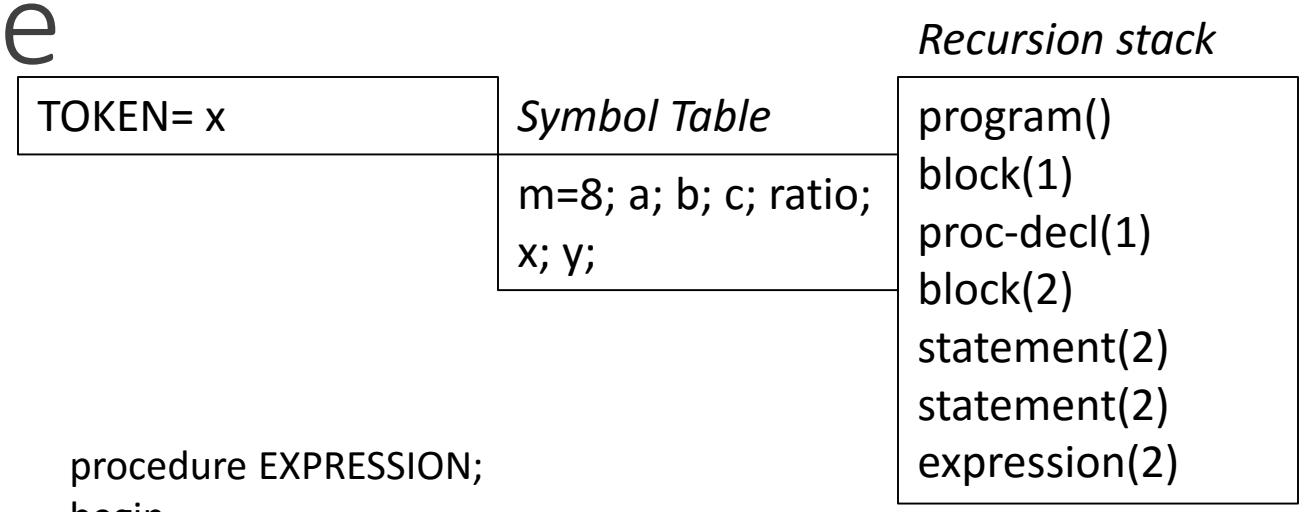
# Small Example




```
        / 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
```

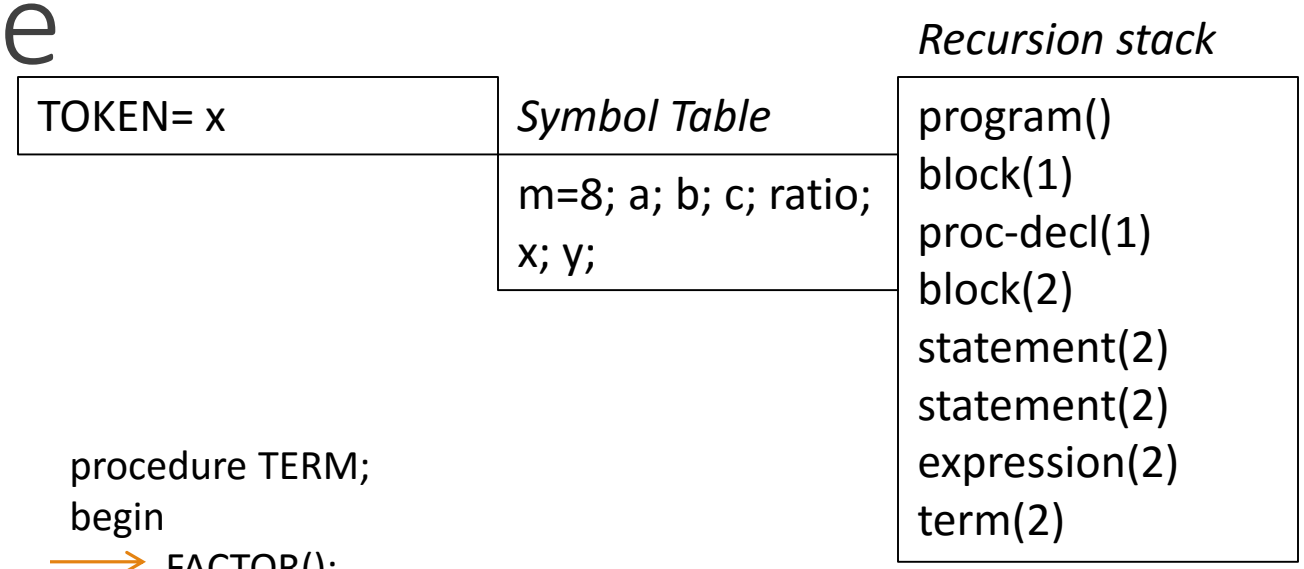
# Small Example




```
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.
```

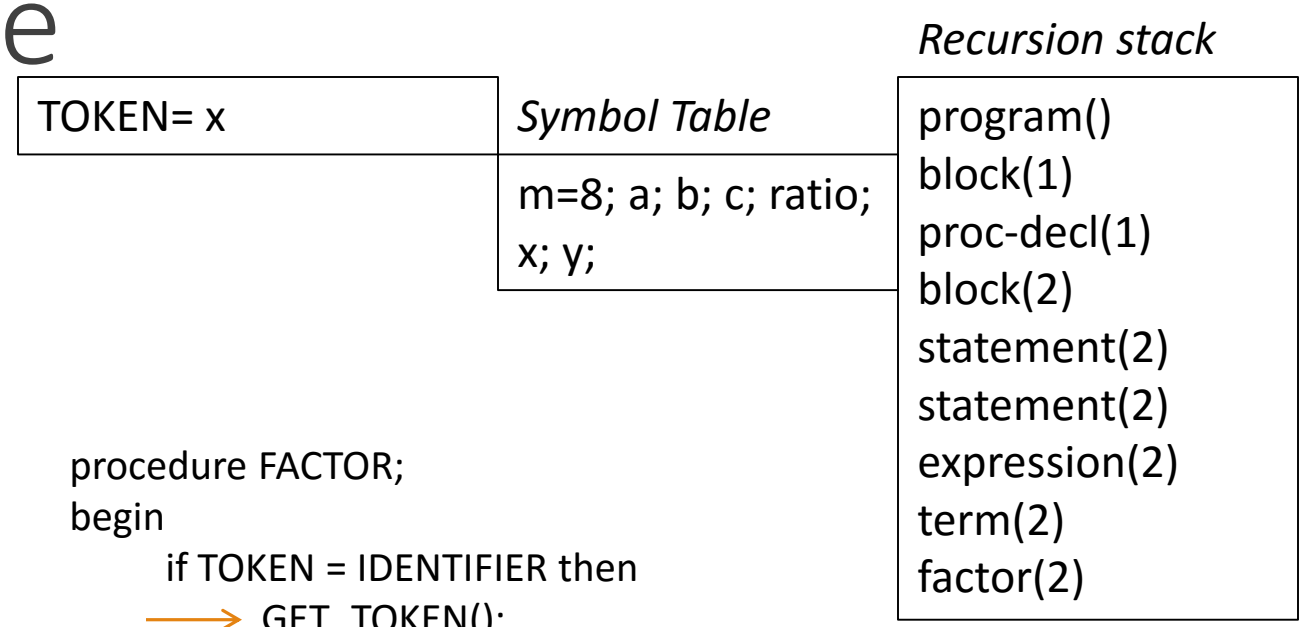
# Small Example




```
procedure TERM;
begin
    FACTOR();
    while TOKEN = MULTIPLYING_OPERATOR do begin
        GET_TOKEN();
        FACTOR();
    end
end;
```

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

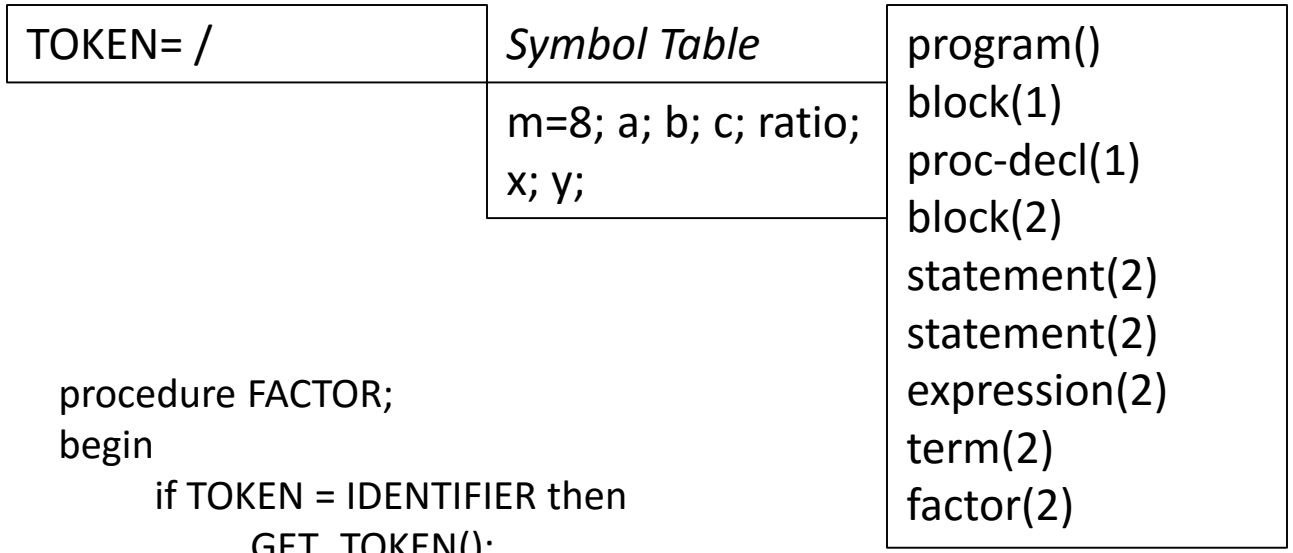
# Small Example




```
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;
```

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

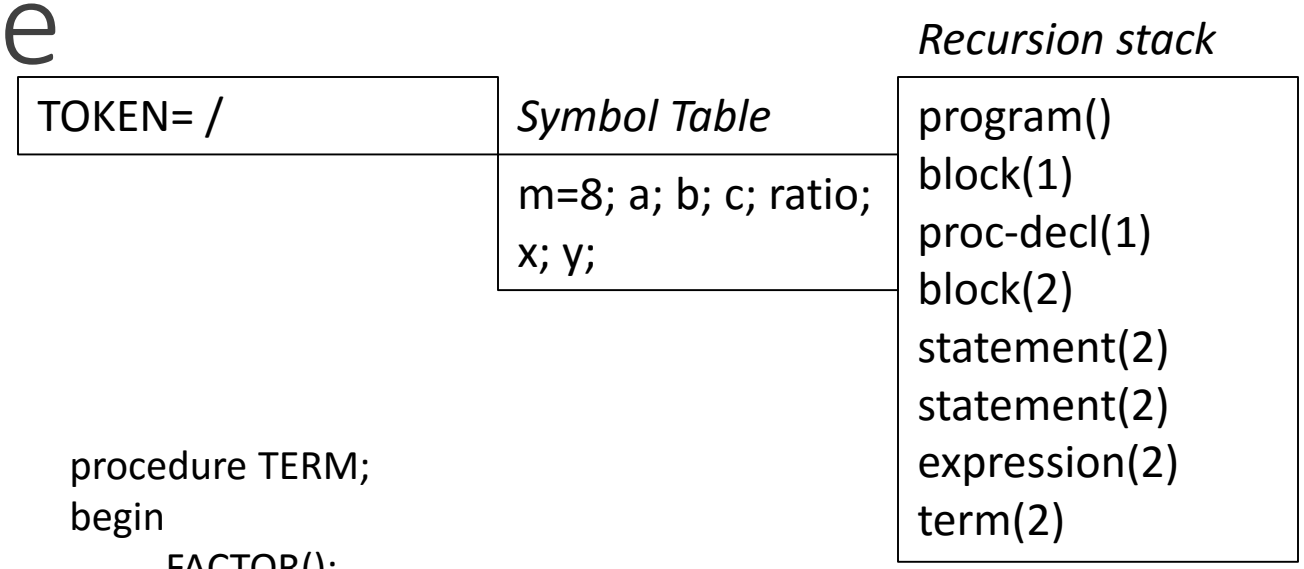
# Small Example




```
        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;
```

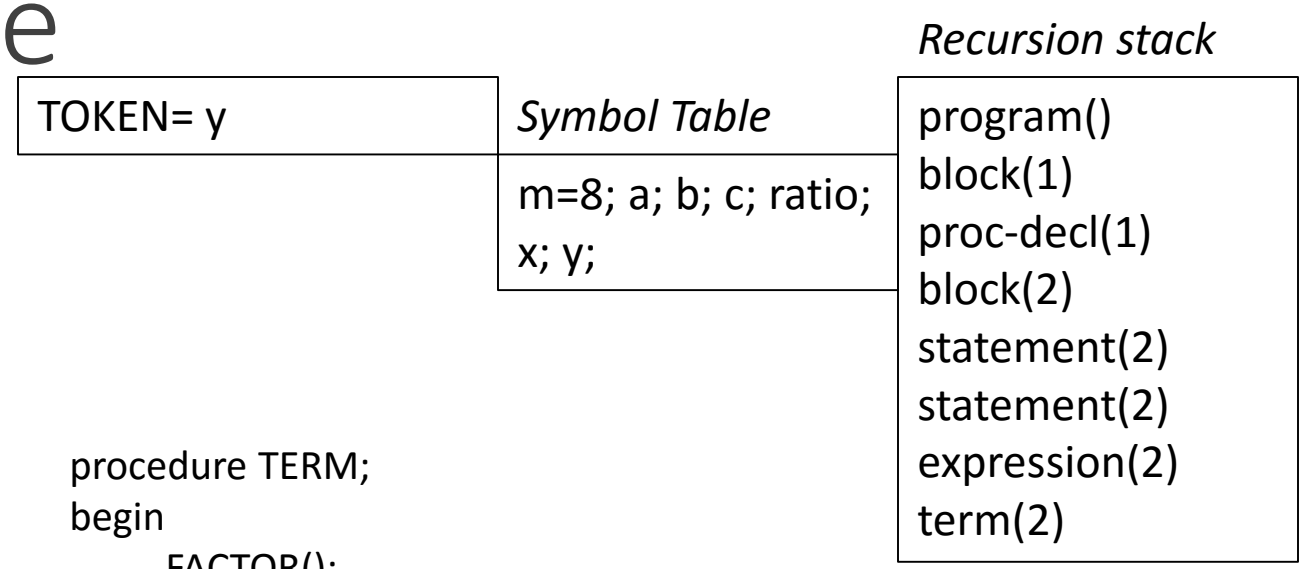
# Small Example




```
procedure TERM;
begin
        FACTOR();
        while TOKEN = MULTIPLYING_OPERATOR do begin
        ⟶      GET_TOKEN();
                FACTOR();
        end
end;
```

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

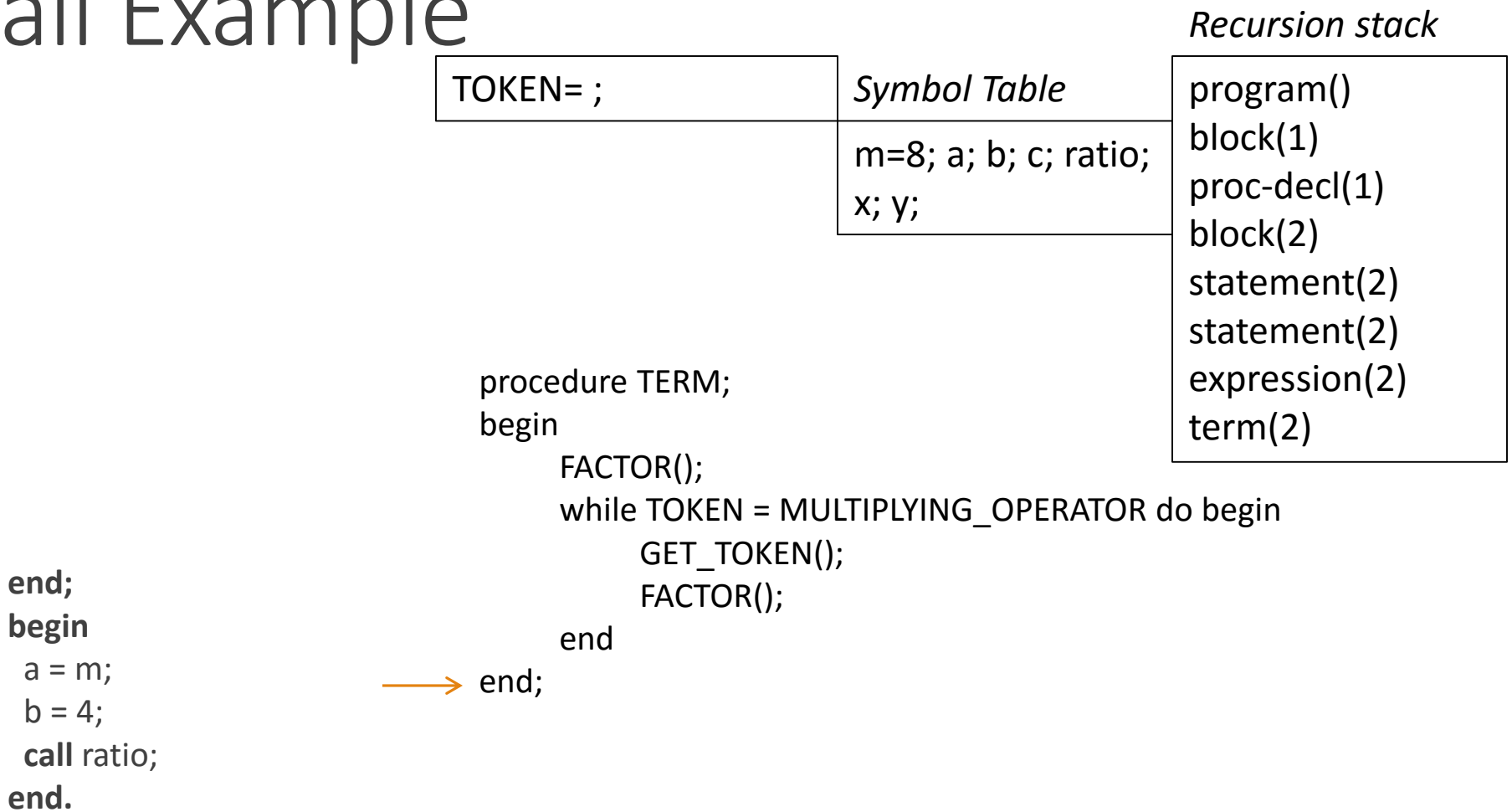
# Small Example



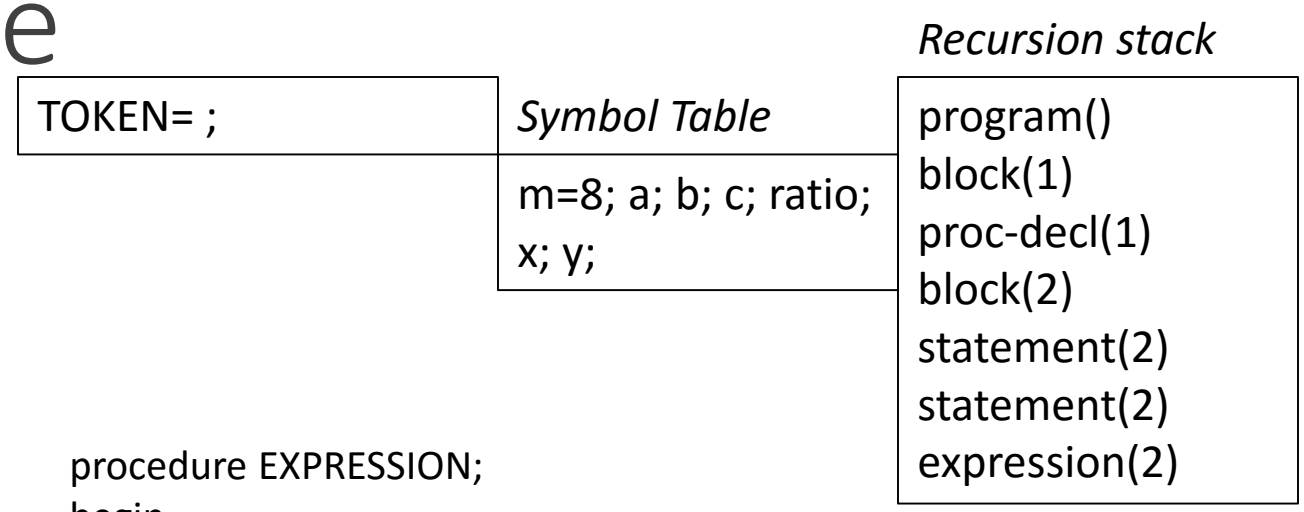

```
procedure TERM;
begin
        FACTOR();
        while TOKEN = MULTIPLYING_OPERATOR do begin
                GET_TOKEN();
                FACTOR();
        end
end;
```

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

# Small Example



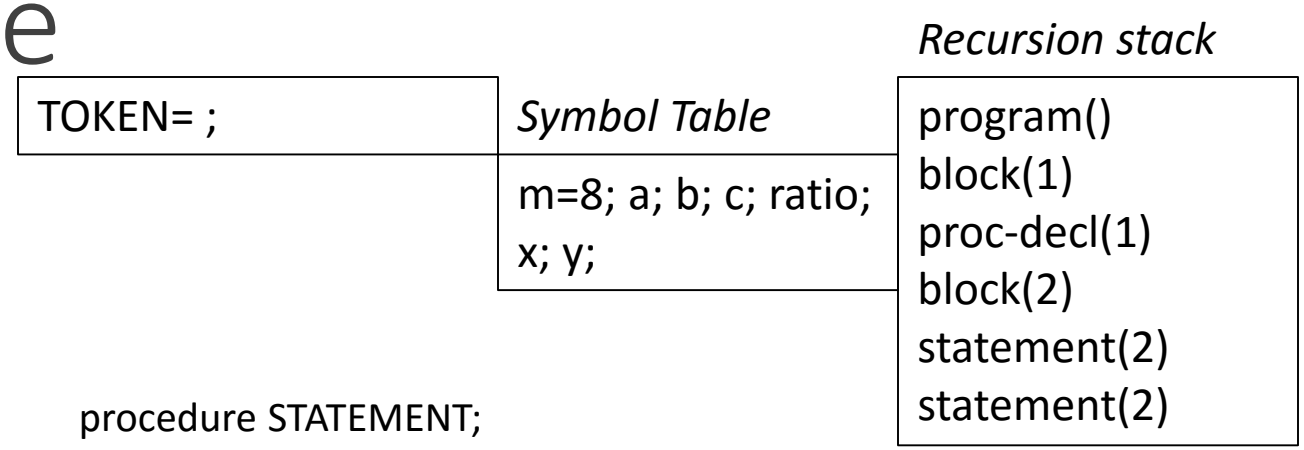
# Small Example




```
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.
```

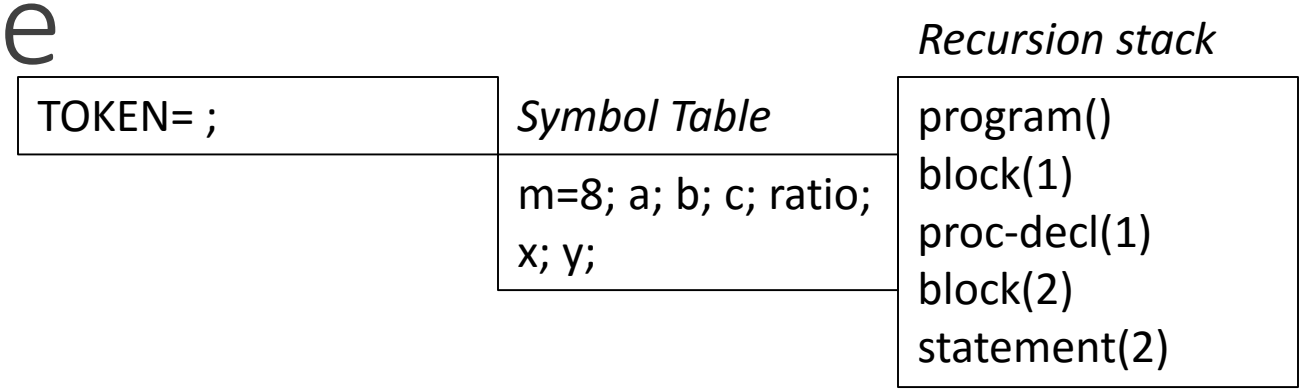
# Small Example




```
 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.
```

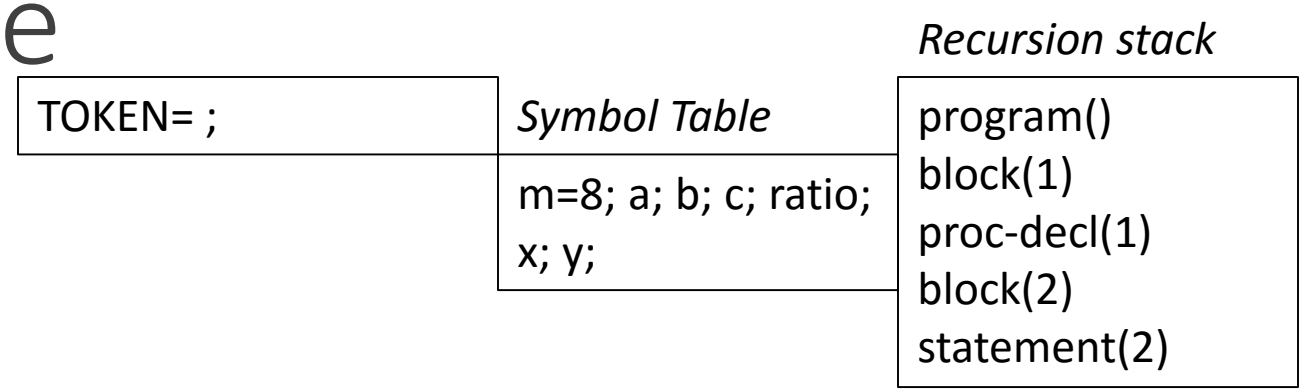
# Small Example




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

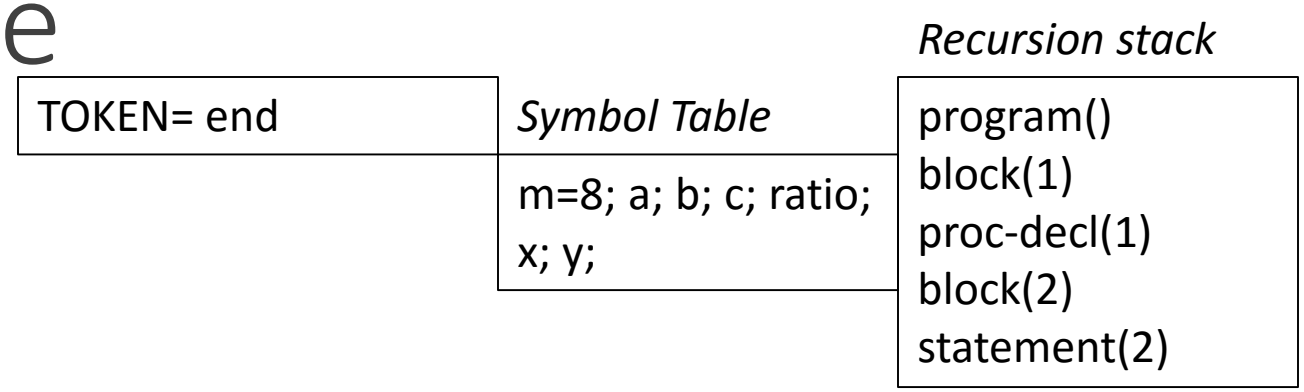
# Small Example




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

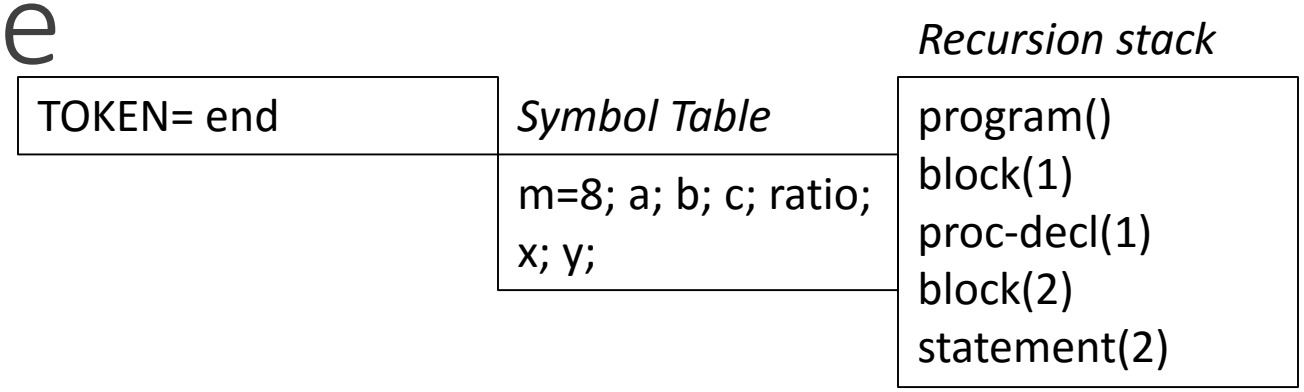
# Small Example




```
      ;
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;
…
```

# Small Example




```
      ;
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;
…
```

# Small Example

TOKEN= ;

*Symbol Table*

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

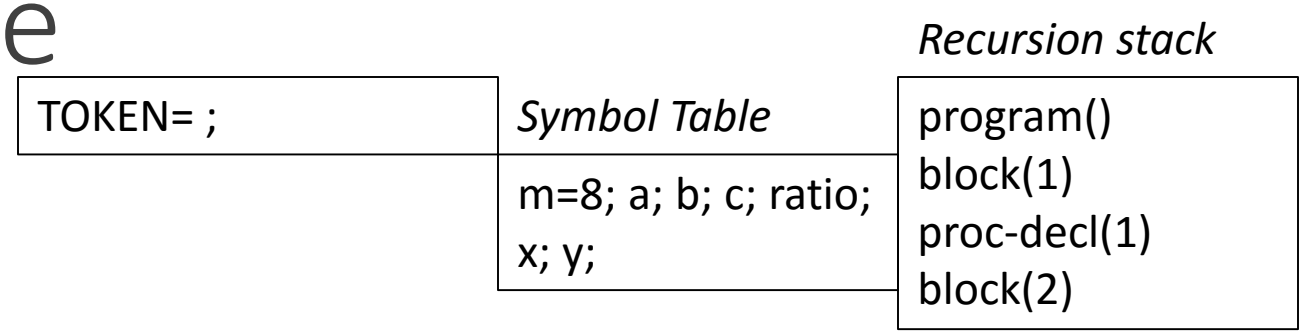
*Recursion stack*

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

```
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;
…
```

# Small Example




```
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.
```

# Small Example

*Recursion stack*

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

```
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;
```

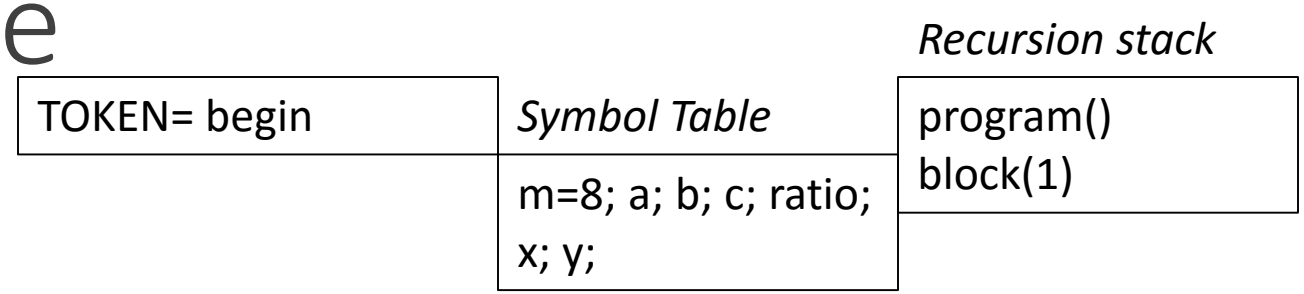
# Small Example

| TOKEN= begin | *Symbol Table* | *Recursion stack* |
| --- | --- | --- |
| | m=8; a; b; c; ratio;<br>x; y; | program()<br>block(1)<br>proc-decl(1) |

```
  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;
```

# Small Example




```
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.
```

# Small Example

| TOKEN= begin | *Symbol Table* | *Recursion stack* |
| --- | --- | --- |
| | m=8; a; b; c; ratio;<br>x; y; | program()<br>block(1)<br>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;
…
```

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

# Small Example

| TOKEN= a | *Symbol Table* | *Recursion stack* |
| --- | --- | --- |
| | m=8; a; b; c; ratio; x; y; | program()<br>block(1)<br>statement(1) |

```
    = 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;
…
```

# Small Example

| TOKEN= ; | *Symbol Table* | *Recursion stack* |
| --- | --- | --- |
| | m=8; a; b; c; ratio; x; y; | program()<br>block(1)<br>statement(1) |

```
  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;
…
```

# Small Example

| TOKEN= ; | *Symbol Table* | *Recursion stack* |
| --- | --- | --- |
| | m=8; a; b; c; ratio; x; y; | program()<br>block(1)<br>statement(1) |

```
  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;
…
```

# Small Example

| TOKEN= b | *Symbol Table* | *Recursion stack* |
| --- | --- | --- |
| | m=8; a; b; c; ratio; x; y; | program()<br>block(1)<br>statement(1) |

```
    = 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;
…
```

# Small Example

*Recursion stack*

| TOKEN= ; | *Symbol Table* | program()<br>block(1)<br>statement(1) |
|---|---|---|
| | m=8; a; b; c; ratio;<br>x; y; | |

```
  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;
…
```

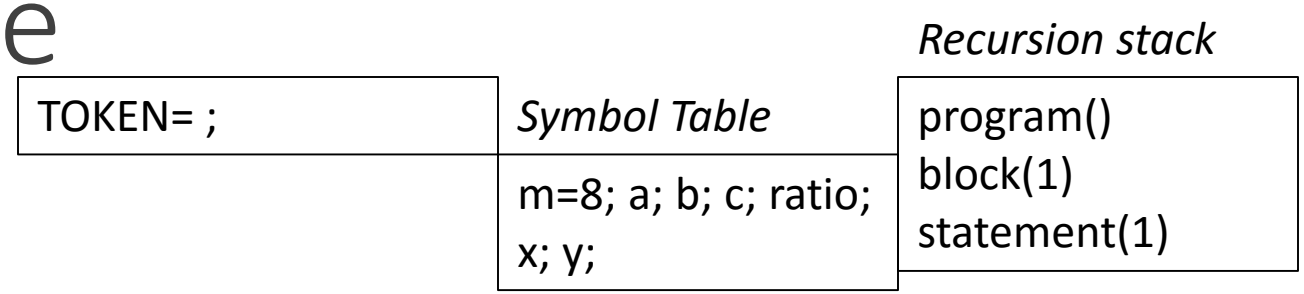
# Small Example

| TOKEN= call | *Symbol Table* | *Recursion stack* |
| --- | --- | --- |
| | m=8; a; b; c; ratio; x; y; | program()<br>block(1)<br>statement(1) |

```
        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;
…
```

# Small Example




```
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.**

# Small Example

| TOKEN= end | *Symbol Table* | *Recursion stack* |
| --- | --- | --- |
| | m=8; a; b; c; ratio; x; y; | program()<br>block(1)<br>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;
…
```

# Small Example

| TOKEN= end | *Symbol Table* | *Recursion stack* |
| --- | --- | --- |
| | m=8; a; b; c; ratio; x; y; | program()<br>block(1)<br>statement(1) |

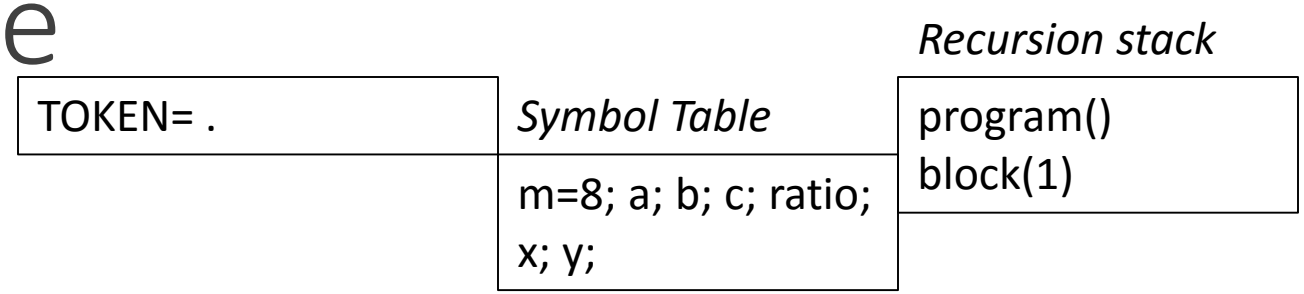
```
procedure STATEMENT;
…
        else if TOKEN = "begin" then begin
                GET TOKEN();
                STATEMENT();
                while TOKEN = ";" do begin
                        GET_TOKEN();
                        STATEMENT();
                end;
                if TOKEN <> "end" then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

# Small Example

| TOKEN= . | *Symbol Table* | *Recursion stack* |
| --- | --- | --- |
| | m=8; a; b; c; ratio; x; y; | program()<br>block(1)<br>statement(1) |

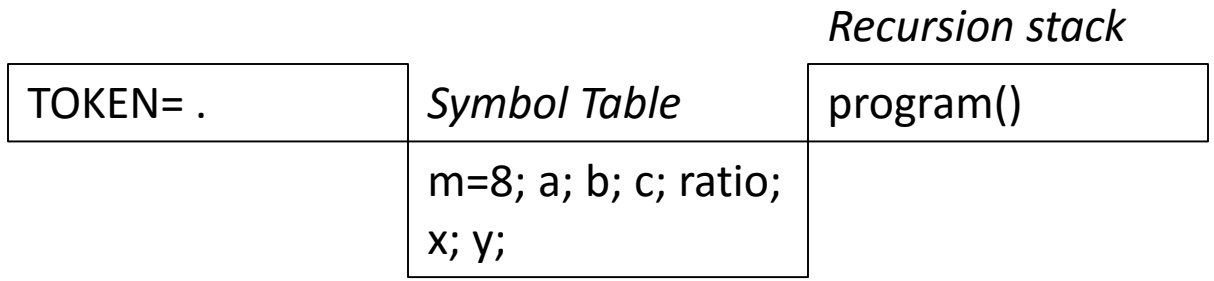
```
procedure STATEMENT;
…
        else if TOKEN = "begin" then begin
                GET TOKEN();
                STATEMENT();
                while TOKEN = ";" do begin
                        GET_TOKEN();
                        STATEMENT();
                end;
                if TOKEN <> "end" then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

# Small Example




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

# Small Example




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

# Questions?