Lab 1 Report

Scanner

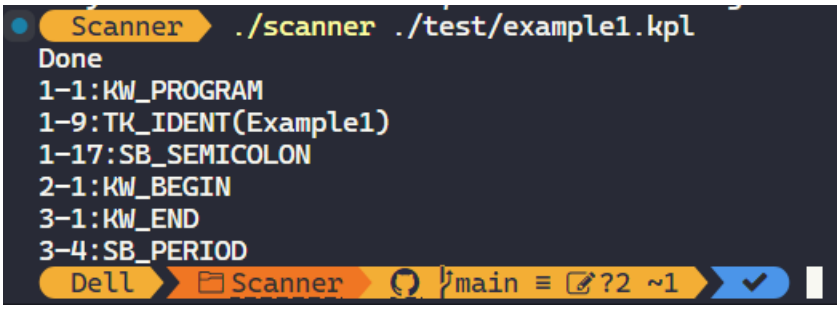
Compiler Construction Lab - 147826

Nguyễn Tiểu Phương 20210692

Results

The results are the same as the expected results given.

Example 1



Example 2

```
Scanner  ./scanner ./test/example2.kpl
Done
1-1:KW_PROGRAM
1-9:TK_IDENT(Example2)
1-17:SB_SEMICOLON
3-1:KW_VAR
3-5:TK_IDENT(n)
3-7:SB_COLON
3-9:KW_INTEGER
3-16:SB_SEMICOLON
5-1:KW_FUNCTION
5-10:TK_IDENT(F)
5-11:SB_LPAR
5-12:TK_IDENT(n)
5-14:SB_COLON
5-16:KW_INTEGER
5-23:SB_RPAR
5-25:SB_COLON
5-27:KW_INTEGER
5-34:SB_SEMICOLON
6-3:KW_BEGIN
7-5:KW_IF
7-8:TK_IDENT(n)
7-10:SB_EQ
7-12:TK_NUMBER(0)
7-14:KW_THEN
7-19:TK_IDENT(F)
7-21:SB_ASSIGN
7-24:TK_NUMBER(1)
7-26:KW_ELSE
7-31:TK_IDENT(F)
7-33:SB_ASSIGN
7-36:TK_IDENT(N)
7-38:SB_TIMES
7-40:TK_IDENT(F)
7-42:SB_LPAR
7-43:TK_IDENT(N)
7-45:SB_MINUS
```

```
7-47:TK_NUMBER(1)
7-48:SB_RPAR
7-49:SB_SEMICOLON
8-3:KW_END
8-6:SB_SEMICOLON
10-1:KW_BEGIN
11-3:KW_FOR
11-7:TK_IDENT(n)
11-9:SB_ASSIGN
11-12:TK_NUMBER(1)
11-14:KW_TO
11-17:TK_NUMBER(7)
11-19:KW_DO
12-5:KW_BEGIN
13-7:KW_CALL
13-12:TK_IDENT(WriteLn)
13-19:SB_SEMICOLON
14-7:KW_CALL
14-12:TK_IDENT(WriteI)
14-18:SB_LPAR
14-20:TK_IDENT(F)
14-21:SB_LPAR
14-22:TK_IDENT(i)
14-23:SB_RPAR
14-24:SB_RPAR
14-25:SB_SEMICOLON
15-5:KW_END
15-8:SB_SEMICOLON
16-1:KW_END
16-4:SB_PERIOD
Dell  Scanner  main ≡ ?2 ~1
```

Example 3

```
Scanner  gcc scanner.c charcode.c error.c reader.c token.c -o scanner
Scanner  ./scanner ./test/example3.kpl
Done
1-1:KW_PROGRAM
1-10:TK_IDENT(EXAMPLE3)
1-18:SB_SEMICOLON
2-1:KW_VAR
2-6:TK_IDENT(I)
2-7:SB_COLON
2-8:KW_INTEGER
2-15:SB_SEMICOLON
3-6:TK_IDENT(N)
3-7:SB_COLON
3-8:KW_INTEGER
3-15:SB_SEMICOLON
4-6:TK_IDENT(P)
4-7:SB_COLON
4-8:KW_INTEGER
4-15:SB_SEMICOLON
5-6:TK_IDENT(Q)
5-7:SB_COLON
5-8:KW_INTEGER
5-15:SB_SEMICOLON
6-6:TK_IDENT(C)
6-7:SB_COLON
6-8:KW_CHAR
6-12:SB_SEMICOLON
8-1:KW_PROCEDURE
8-12:TK_IDENT(HANOI)
8-17:SB_LPAR
8-18:TK_IDENT(N)
8-19:SB_COLON
8-20:KW_INTEGER
8-27:SB_SEMICOLON
8-30:TK_IDENT(S)
8-31:SB_COLON
8-32:KW_INTEGER
8-39:SB_SEMICOLON
8-42:TK_IDENT(Z)
8-43:SB_COLON
8-44:KW_INTEGER
8-51:SB_RPAR
8-52:SB_SEMICOLON
9-1:KW_BEGIN
10-3:KW_IF
```

```
10-7:TK_IDENT(N)
10-10:SB_NEQ
10-12:TK_NUMBER(0)
10-15:KW_THEN
11-5:KW_BEGIN
12-7:KW_CALL
12-13:TK_IDENT(HANOI)
12-18:SB_LPAR
12-19:TK_IDENT(N)
12-20:SB_MINUS
12-21:TK_NUMBER(1)
12-22:SB_COMMA
12-23:TK_IDENT(S)
12-24:SB_COMMA
12-25:TK_NUMBER(6)
12-26:SB_MINUS
12-27:TK_IDENT(S)
12-28:SB_MINUS
12-29:TK_IDENT(Z)
12-30:SB_RPAR
12-31:SB_SEMICOLON
13-7:TK_IDENT(I)
13-8:SB_ASSIGN
13-10:TK_IDENT(I)
13-11:SB_PLUS
13-12:TK_NUMBER(1)
13-13:SB_SEMICOLON
14-7:KW_CALL
14-13:TK_IDENT(WRITELN)
14-20:SB_SEMICOLON
15-7:KW_CALL
15-13:TK_IDENT(WRITEI)
15-19:SB_LPAR
15-20:TK_IDENT(I)
15-21:SB_RPAR
15-22:SB_SEMICOLON
16-7:KW_CALL
16-13:TK_IDENT(WRITEI)
16-19:SB_LPAR
16-20:TK_IDENT(N)
16-21:SB_RPAR
16-22:SB_SEMICOLON
17-7:KW_CALL
17-13:TK_IDENT(WRITEI)
17-19:SB_LPAR
```

```
17-20:TK_IDENT(S)
17-21:SB_RPAR
17-22:SB_SEMICOLON
18-7:KW_CALL
18-13:TK_IDENT(WRITEI)
18-19:SB_LPAR
18-20:TK_IDENT(Z)
18-21:SB_RPAR
18-22:SB_SEMICOLON
19-7:KW_CALL
19-13:TK_IDENT(HANOI)
19-18:SB_LPAR
19-19:TK_IDENT(N)
19-20:SB_MINUS
19-21:TK_NUMBER(1)
19-22:SB_COMMA
19-23:TK_NUMBER(6)
19-24:SB_MINUS
19-25:TK_IDENT(S)
19-26:SB_MINUS
19-27:TK_IDENT(Z)
19-28:SB_COMMA
19-29:TK_IDENT(Z)
19-30:SB_RPAR
20-5:KW_END
21-1:KW_END
21-4:SB_SEMICOLON
23-1:KW_BEGIN
24-3:KW_FOR
24-8:TK_IDENT(N)
24-10:SB_ASSIGN
24-13:TK_NUMBER(1)
24-16:KW_TO
24-20:TK_NUMBER(4)
24-23:KW_DO
25-5:KW_BEGIN
26-7:KW_FOR
26-12:TK_IDENT(I)
26-13:SB_ASSIGN
26-15:TK_NUMBER(1)
26-18:KW_TO
26-22:TK_NUMBER(4)
26-25:KW_DO
27-9:KW_CALL
27-15:TK_IDENT(WRITEC)
```

```
27-21:SB_LPAR
27-23:TK_CHAR(' ')
27-25:SB_RPAR
27-26:SB_SEMICOLON
28-7:KW_CALL
28-13:TK_IDENT(READC)
28-18:SB_LPAR
28-19:TK_IDENT(C)
28-20:SB_RPAR
28-21:SB_SEMICOLON
29-7:KW_CALL
29-13:TK_IDENT(WRITEC)
29-19:SB_LPAR
29-20:TK_IDENT(C)
29-21:SB_RPAR
30-5:KW_END
30-8:SB_SEMICOLON
31-3:TK_IDENT(P)
31-4:SB_ASSIGN
31-6:TK_NUMBER(1)
31-7:SB_SEMICOLON
32-3:TK_IDENT(Q)
32-4:SB_ASSIGN
32-6:TK_NUMBER(2)
32-7:SB_SEMICOLON
33-3:KW_FOR
33-8:TK_IDENT(N)
33-9:SB_ASSIGN
33-11:TK_NUMBER(2)
33-14:KW_TO
33-18:TK_NUMBER(4)
33-21:KW_DO
34-5:KW_BEGIN
35-7:TK_IDENT(I)
35-8:SB_ASSIGN
35-10:TK_NUMBER(0)
35-11:SB_SEMICOLON
36-7:KW_CALL
36-13:TK_IDENT(HANOI)
36-18:SB_LPAR
36-19:TK_IDENT(N)
36-20:SB_COMMA
36-21:TK_IDENT(P)
36-23:TK_IDENT(Q)
36-23:TK_IDENT(Q)
```

```
36-24:SB_RPAR
36-25:SB_SEMICOLON
37-7:KW_CALL
37-13:TK_IDENT(WRITELN)
38-5:KW_END
39-1:KW_END
39-4:SB_PERIOD
Dell  Scanner  main ≡ ?2 ~1
```

Errors

The goal is to evaluate whether the scanner can detect and announce all the errors encountered.

Example 1

For this example, since the KPL program is too simple (having nothing inside between **`begin`** and **`end`**) we only invoke the following error:

Error: Comment is not closed correctly

```
ERM_ENDOFCOMMENT
```

```
example1.kpl U ×
KPL-Compiler > Scanner > test > example1.kpl
1   Program Example1; (* Example 1 *)
2   Begin
3   End. (* Example 1
```

We invoke this error by deleting the closing bracket of a comment in Line 3 here.

```
Scanner  ./scanner ./test/example1.kpl
Done
1-1:KW_PROGRAM
1-9:TK_IDENT(Example1)
1-17:SB_SEMICOLON
2-1:KW_BEGIN
3-1:KW_END
3-4:SB_PERIOD
3-19:End of comment expected!
3-6:SB_LPAR
Dell  Scanner  main ≡ ?2 ~1
```

The scanner throws the error message.

Example 2

Error: Identification too long

```
example1.kpl U    example2.kpl U ×
KPL-Compiler > Scanner > test > example2.kpl
1   Program Example2; (* Factorial *)
2
3   Var nnnnnnnnnnnnnnnnnnnnnnnn : Integer;
4
5   Function F(n : Integer) : Integer;
6     Begin
7       If n = 0 Then F := 1 Else F := N * F (N - 1);
8     End;
9
10  Begin
11    For n := 1 To 7 Do
12      Begin
13        Call WriteLn;
14        Call WriteI( F(i));
15      End;
16  End. (* Factorial *)
```

This error can be raised by trying to create a variable with a very long name.

```
Scanner  ./scanner ./test/example2.kpl
Done
1-1:KW_PROGRAM
1-9:TK_IDENT(Example2)
1-17:SB_SEMICOLON
3-1:KW_VAR
3-5:Identification too long!
3-5:TK_NONE
3-30:SB_COLON
3-32:KW_INTEGER
```

The scanner announces the message and continues.

Error: Number outside of range (too long)

example1.kpl U example2.kpl U

KPL-Compiler > Scanner > test > example2.kpl

```
Program Example2; (* Factorial *)

Var n : Integer;

Function F(n : Integer) : Integer;
  Begin
    If n = 0 Then F := 1 Else F := N * F (N - 1);
  End;

Begin
  For n := 1 To 99999999999999999999 Do
    Begin
      Call WriteLn;
      Call WriteI( F(i));
    End;
End. (* Factorial *)
```

Again, we can try to create a very big number to raise the error.

```
10-1:KW_BEGIN
11-3:KW_FOR
11-7:TK_IDENT(n)
11-9:SB_ASSIGN
11-12:TK_NUMBER(1)
11-14:KW_TO
11-17:Value of integer number exceeds the range!
11-17:TK_NONE
11-38:KW_DO
12-5:KW_BEGIN
13-7:KW_CALL
13-12:TK_IDENT(WriteLn)
13-19:SB_SEMICOLON
```

The corresponding error message.

Error: Comment not closed

```
example1.kpl U    example2.kpl U ×
KPL-Compiler > Scanner > test > example2.kpl
1   Program Example2; (* Factorial *)
2
3   Var n : Integer;
4
5   Function F(n : Integer) : Integer;
6     Begin
7       If n = 0 Then F := 1 Else F := N * F (N - 1);
8     End;
9
10  Begin
11    For n := 1 To 7 Do
12      Begin
13        Call WriteLn;
14        Call WriteI( F(i));
15      End;
16  End. (* Factorial
```

Similar to Example 1. We try to delete some comment brackets.

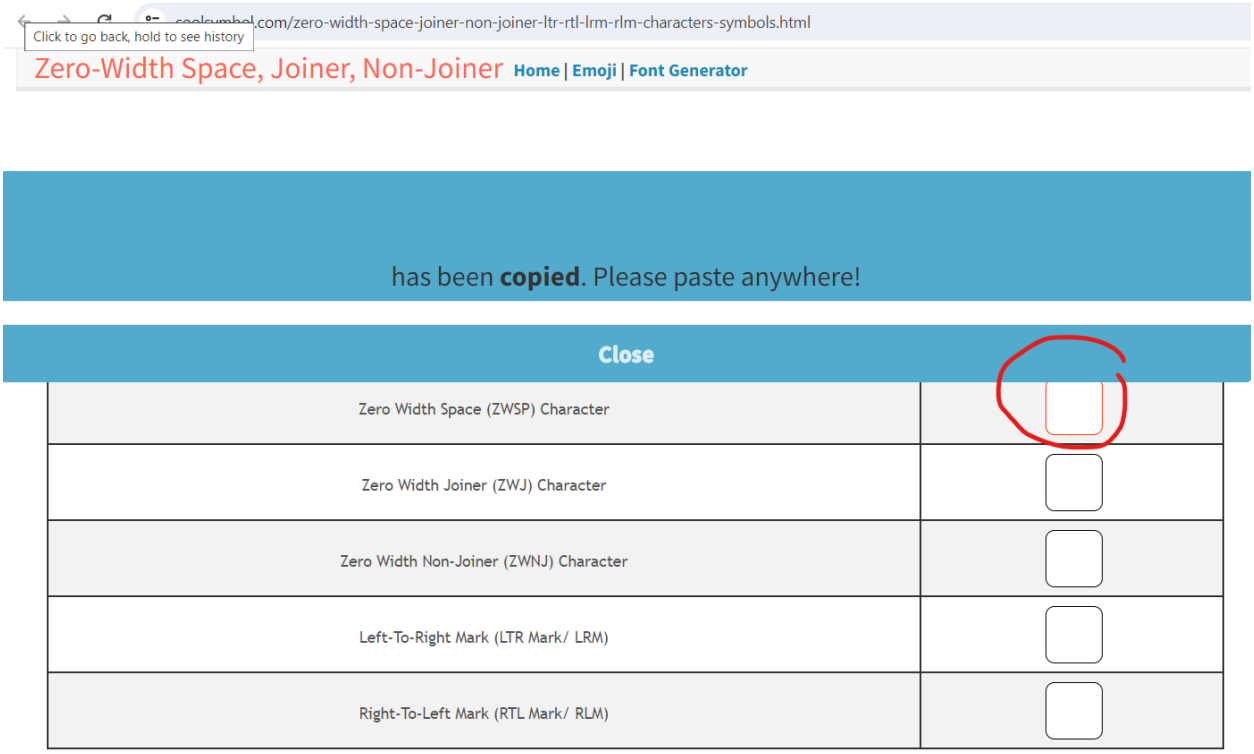
```
15-5:KW_END
15-8:SB_SEMICOLON
16-1:KW_END
16-4:SB_PERIOD
16-19:End of comment expected!
16-6:SB_LPAR
Dell  Scanner  main ≡ ?2 ~1
```

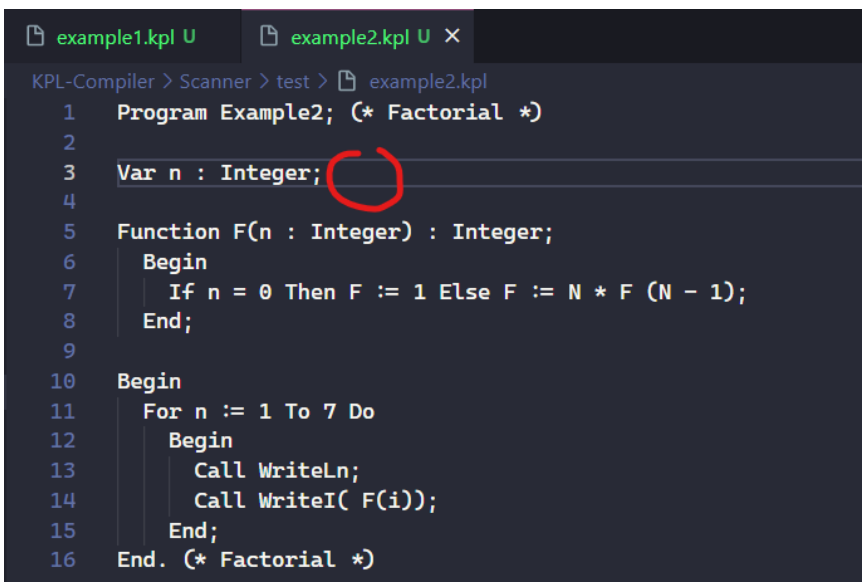
The scanner detects the error.

Error: Invalid symbol

One way to invoke this error is using non-printable character (`state` = 43). We can find those easily by Google search and copy the character from there.

I use the following site to copy a ***Zero-width space character (ZWSP)***:





I've pasted the character here, but since it is not printable, it does not show up in the editor.

```
Scanner  ./scanner ./test/example2.kpl
Done
1-1:KW_PROGRAM
1-9:TK_IDENT(Example2)
1-17:SB_SEMICOLON
3-1:KW_VAR
3-5:TK_IDENT(n)
3-7:SB_COLON
3-9:KW_INTEGER
3-16:SB_SEMICOLON
3-17:Invalid symbol!
3-17:TK_NONE
3-18:Invalid symbol!
3-18:TK_NONE
3-19:Invalid symbol!
3-19:TK_NONE
5-1:KW_FUNCTION
5-10:TK_IDENT(F)
5-11:SB_LPAR
```

The scanner recognizes the non-printables and outputs the errors.

Error: Invalid char constant

example1.kpl U | example2.kpl U | example3.kpl U

KPL-Compiler > Scanner > test > example2.kpl

```
Program Example2; (* Factorial *)

Var n : Integer;

Function F(n : Integer) : Integer;
  Begin
    If n = 0 Then F := 1 Else F := N * F (N - 1);
  End;

Begin
  For n := 1 To 7 Do
    Begin
      Call WriteLn('invalid char');
      Call WriteI( F(i));
    End;
End.
```

In KPL, the string is not a valid char constant, so we can raise the error as above.

```
13-7:KW_CALL
13-12:TK_IDENT(WriteLn)
13-19:SB_LPAR
13-20:Invalid const char!
13-22:SB_LPAR
13-23:TK_IDENT(valid)
13-29:KW_CHAR
13-33:Invalid const char!
13-35:SB_LPAR
14-7:KW_CALL
14-12:TK_IDENT(WriteI)
```

The scanner successfully detects the invalid char constant.

Example 3

Similar to previous examples, we can create the errors by the same method.

Error: Comment not closed

example1.kpl U | example2.kpl U | example3.kpl U ×

KPL-Compiler > Scanner > test > example3.kpl

```
BEGIN
  FOR  N := 1  TO  4  DO
  P:=1;
  Q:=2;
  FOR  N:=2  TO  4  DO
    BEGIN
      I:=0;
      CALL  HANOI(N,P,Q);
      CALL  WRITELN
    END
END.  (* TOWER OF HANOI *
```

```
37-13:TK_IDENT(WRITELN)
38-5:KW_END
39-1:KW_END
39-4:SB_PERIOD
40-1:End of comment expected!
39-7:SB_LPAR
```

Error: Identifier too long

example1.kpl U | example2.kpl U | example3.kpl U ×

KPL-Compiler > Scanner > test > example3.kpl

```
PROGRAM  EXAMPLE3;  (* TOWER OF HANOI *)
VAR  ThisIsAVeryLongIdent:INTEGER;
     N:INTEGER;
     P:INTEGER;
     Q:INTEGER;
     C:CHAR;

PROCEDURE  HANOI(N:INTEGER;  S:INTEGER;  Z:INTEGER);
BEGIN
  IF  N ≠ 0  THEN
    BEGIN
      CALL  HANOI(N-1,S,6-S-Z);
      I:=I+1;
      CALL  WRITELN;
      CALL  WRITEI(I);
      CALL  WRITEI(N);
      CALL  WRITEI(S);
```

```
Scanner  ./scanner ./test/example3.kpl
Done
1-1:KW_PROGRAM
1-10:TK_IDENT(EXAMPLE3)
1-18:SB_SEMICOLON
2-1:KW_VAR
2-6:Identification too long!
2-6:TK_NONE
2-26:SB_COLON
2-27:KW_INTEGER
2-34:SB_SEMICOLON
```

Error: Number too long

example1.kpl U example2.kpl U example3.kpl U

KPL-Compiler > Scanner > test > example3.kpl

```
PROGRAM  EXAMPLE3;  (* TOWER OF HANOI *)
VAR  I:INTEGER;
     N:INTEGER;
     P:INTEGER;
     Q:INTEGER;
     C:CHAR;

PROCEDURE  HANOI(N:INTEGER;  S:INTEGER;  Z:INTEGER);
BEGIN
  IF  N ≠ 46468954564431321354  THEN
    BEGIN
      CALL  HANOI(N-1,S,6-S-Z);
      I:=I+1;
      CALL  WRITELN;
      CALL  WRITEI(I);
      CALL  WRITEI(N);
      CALL  WRITEI(S);
```

```
9-1:KW_BEGIN
10-3:KW_IF
10-7:TK_IDENT(N)
10-10:SB_NEQ
10-12:Value of integer number exceeds the range!
10-12:TK_NONE
10-34:KW_THEN
11-5:KW_BEGIN
12-7:KW_CALL
```

Error: Invalid char constant

example1.kpl U | example2.kpl U | example3.kpl U

KPL-Compiler > Scanner > test > example3.kpl

```
22
23  BEGIN
24    FOR  N := 1  TO  4  DO
25      BEGIN
26        FOR  I:=1  TO  4  DO
27          CALL  WRITEC('another invalid char');
28        CALL  READC(C);
29        CALL  WRITEC(C)
30      END;
31    P:=1;
32    Q:=2;
33    FOR  N:=2  TO  4  DO
34      BEGIN
35        I:=0;
36        CALL  HANOI(N,P,Q);
37        CALL  WRITELN
38      END
39  END.  (* TOWER OF HANOI *)
```

```
27-21:SB_LPAR
27-22:Invalid const char!
27-24:SB_LPAR
27-25:TK_IDENT(other)
27-31:TK_IDENT(invalid)
27-39:KW_CHAR
27-43:Invalid const char!
27-45:SB_LPAR
28-7:KW_CALL
28-13:TK_IDENT(READC)
```

Error: Invalid symbol

example1.kpl U | example2.kpl U | example3.kpl U

KPL-Compiler > Scanner > test > example3.kpl

```
1   PROGRAM  EXAMPLE3;  (* TOWER OF HANOI *)
2   VAR  I:INTEGER;
3        N:INTEGER;
4        P:INTEGER;
5        Q:INTEGER;
6        C:CHAR;
7
8   PROCEDURE  HANOI(N:INTEGER;  S:INTEGER;  Z:INTEGER);
9   BEGIN
10    IF  N !? 0  THEN
11      BEGIN
12        CALL  HANOI(N-1,S,6-S-Z);
13        I:=I+1;
14        CALL  WRITELN;
15        CALL  WRITEI(I);
16        CALL  WRITEI(N);
17        CALL  WRITEI(S);
18        CALL  WRITEI(Z);
```

```
8-44:KW_INTEGER
8-51:SB_RPAR
8-52:SB_SEMICOLON
9-1:KW_BEGIN
10-3:KW_IF
10-7:TK_IDENT(N)
10-9:Invalid symbol!
10-9:TK_NONE
10-10:Invalid symbol!
10-10:TK_NONE
10-12:TK_NUMBER(0)
10-15:KW_THEN
11-5:KW_BEGIN
```

THE END
