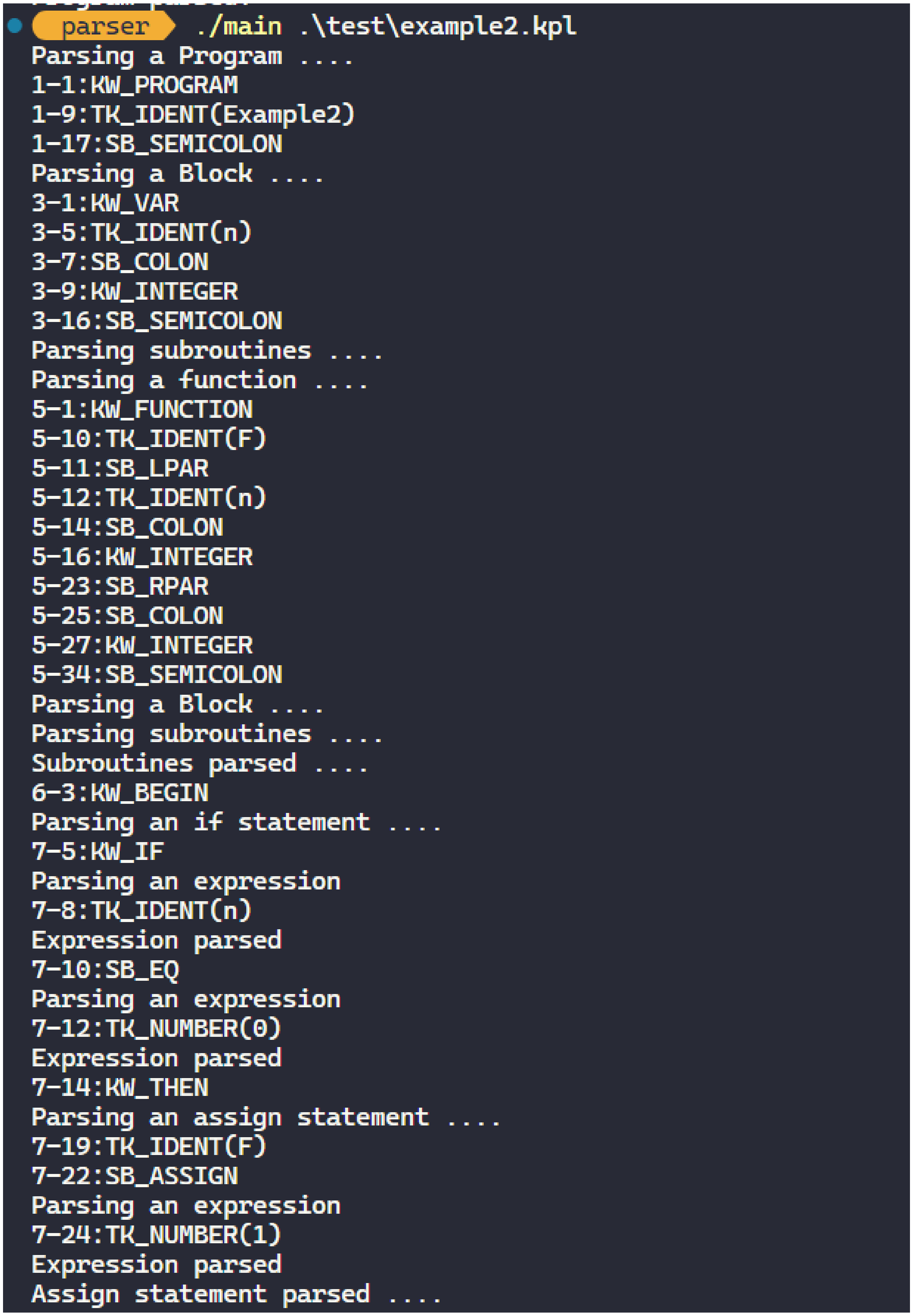
Lab 2 Report | Parser

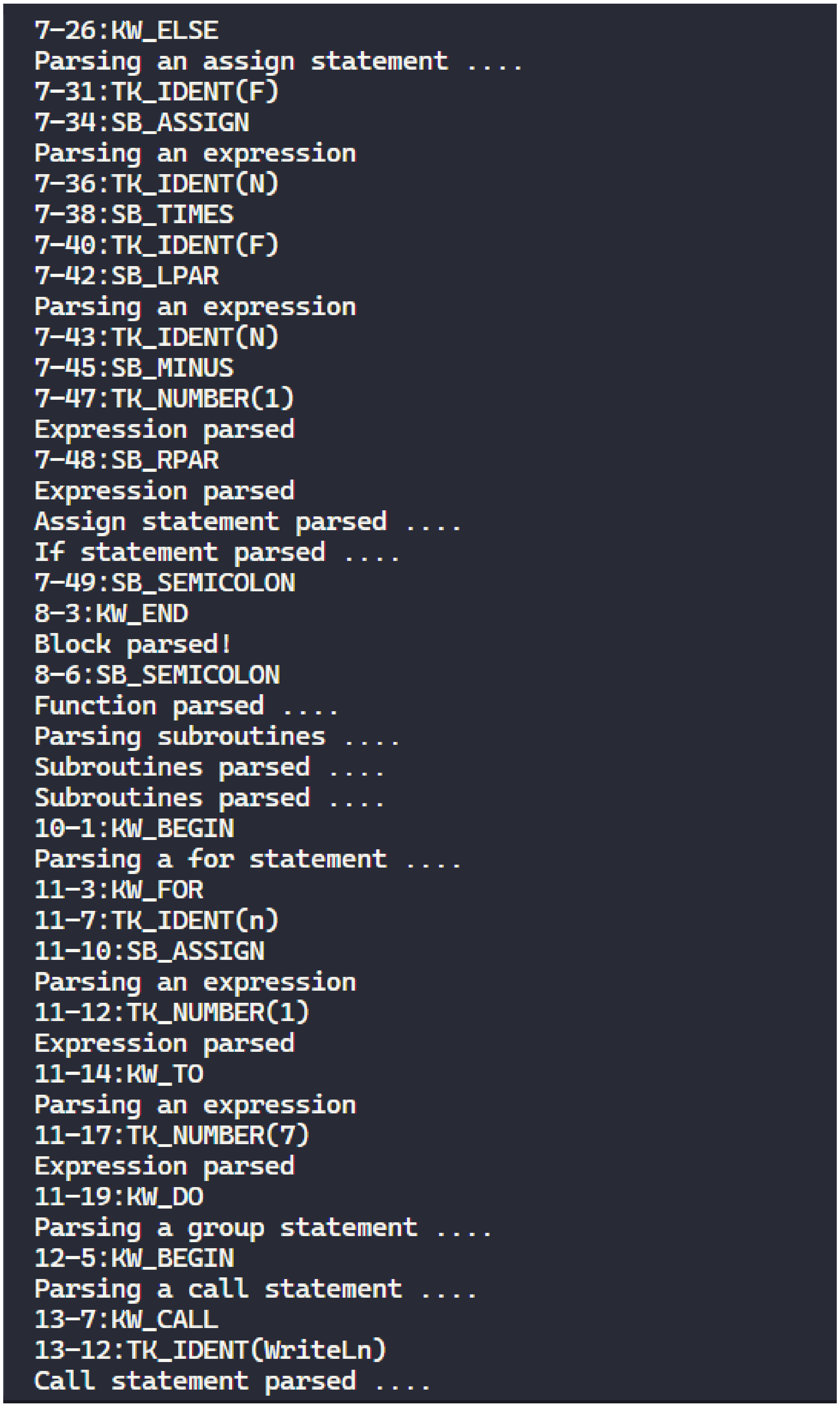
Nguyễn Tiểu Phương 20210692

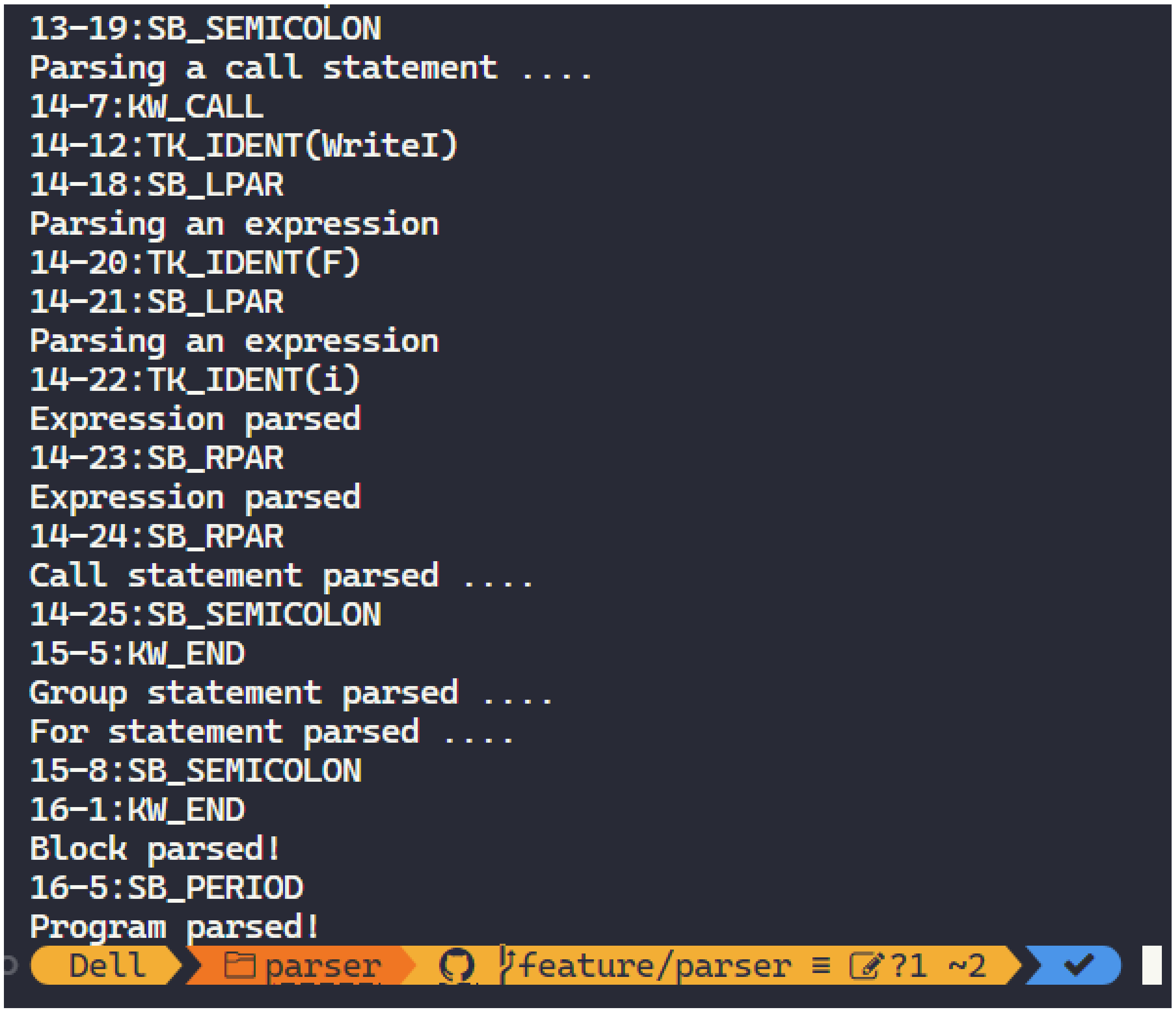
As the instruction only requires the reporting of only one example, I choose **Example 2** – neither too long nor too short for demonstration purposes.

Results



```
parser  ./main .\test\example2.kpl
Parsing a Program ....
1-1:KW_PROGRAM
1-9:TK_IDENT(Example2)
1-17:SB_SEMICOLON
Parsing a Block ....
3-1:KW_VAR
3-5:TK_IDENT(n)
3-7:SB_COLON
3-9:KW_INTEGER
3-16:SB_SEMICOLON
Parsing subroutines ....
Parsing a function ....
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
Parsing a Block ....
Parsing subroutines ....
Subroutines parsed ....
6-3:KW_BEGIN
Parsing an if statement ....
7-5:KW_IF
Parsing an expression
7-8:TK_IDENT(n)
Expression parsed
7-10:SB_EQ
Parsing an expression
7-12:TK_NUMBER(0)
Expression parsed
7-14:KW_THEN
Parsing an assign statement ....
7-19:TK_IDENT(F)
7-22:SB_ASSIGN
Parsing an expression
7-24:TK_NUMBER(1)
Expression parsed
Assign statement parsed ....
```


```
7-26:KW_ELSE
Parsing an assign statement ....
7-31:TK_IDENT(F)
7-34:SB_ASSIGN
Parsing an expression
7-36:TK_IDENT(N)
7-38:SB_TIMES
7-40:TK_IDENT(F)
7-42:SB_LPAR
Parsing an expression
7-43:TK_IDENT(N)
7-45:SB_MINUS
7-47:TK_NUMBER(1)
Expression parsed
7-48:SB_RPAR
Expression parsed
Assign statement parsed ....
If statement parsed ....
7-49:SB_SEMICOLON
8-3:KW_END
Block parsed!
8-6:SB_SEMICOLON
Function parsed ....
Parsing subroutines ....
Subroutines parsed ....
Subroutines parsed ....
10-1:KW_BEGIN
Parsing a for statement ....
11-3:KW_FOR
11-7:TK_IDENT(n)
11-10:SB_ASSIGN
Parsing an expression
11-12:TK_NUMBER(1)
Expression parsed
11-14:KW_TO
Parsing an expression
11-17:TK_NUMBER(7)
Expression parsed
11-19:KW_DO
Parsing a group statement ....
12-5:KW_BEGIN
Parsing a call statement ....
13-7:KW_CALL
13-12:TK_IDENT(WriteLn)
Call statement parsed ....
```



```
13-19:SB_SEMICOLON
Parsing a call statement ....
14-7:KW_CALL
14-12:TK_IDENT(WriteI)
14-18:SB_LPAR
Parsing an expression
14-20:TK_IDENT(F)
14-21:SB_LPAR
Parsing an expression
14-22:TK_IDENT(i)
Expression parsed
14-23:SB_RPAR
Expression parsed
14-24:SB_RPAR
Call statement parsed ....
14-25:SB_SEMICOLON
15-5:KW_END
Group statement parsed ....
For statement parsed ....
15-8:SB_SEMICOLON
16-1:KW_END
Block parsed!
16-5:SB_PERIOD
Program parsed!
 Dell  parser  feature/parser ≡ ?1 ~2  ✓
```

Errors

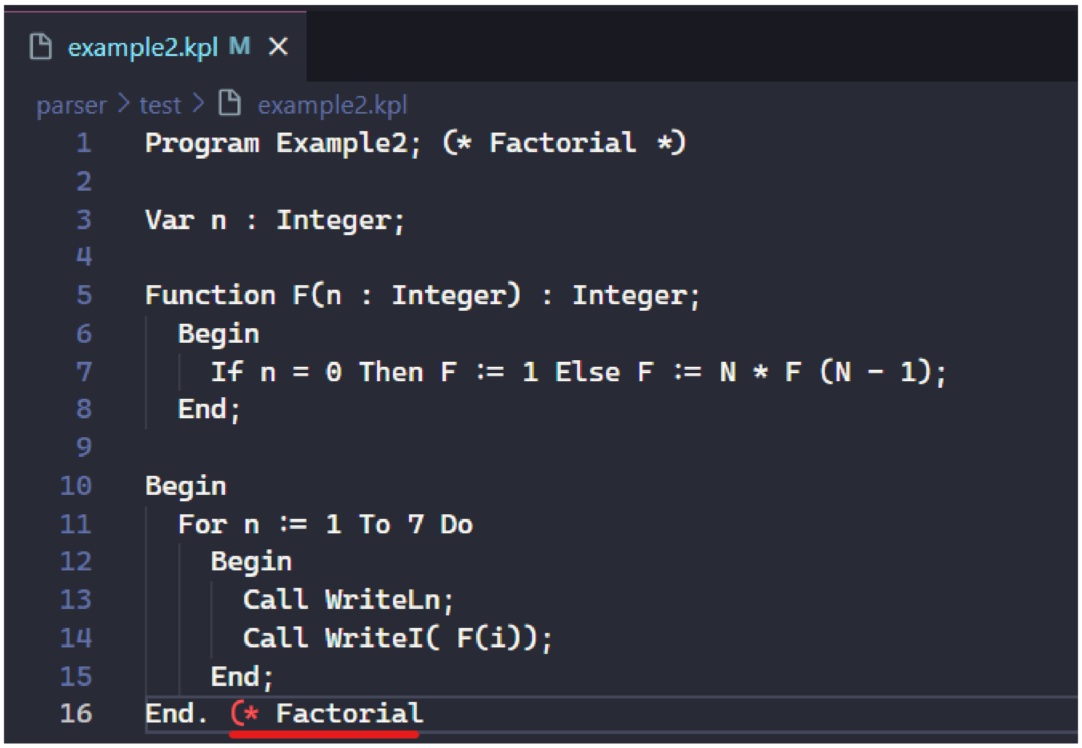
Similar to Lab 1 report, we can invoke the errors by making certain changes to the source program.

The following errors are demonstrated each by 2 screenshots:

- one screenshot of the change in the source program,
- one screenshot of the corresponding output produced by the parser.

I underlined the changes and the errors in red.

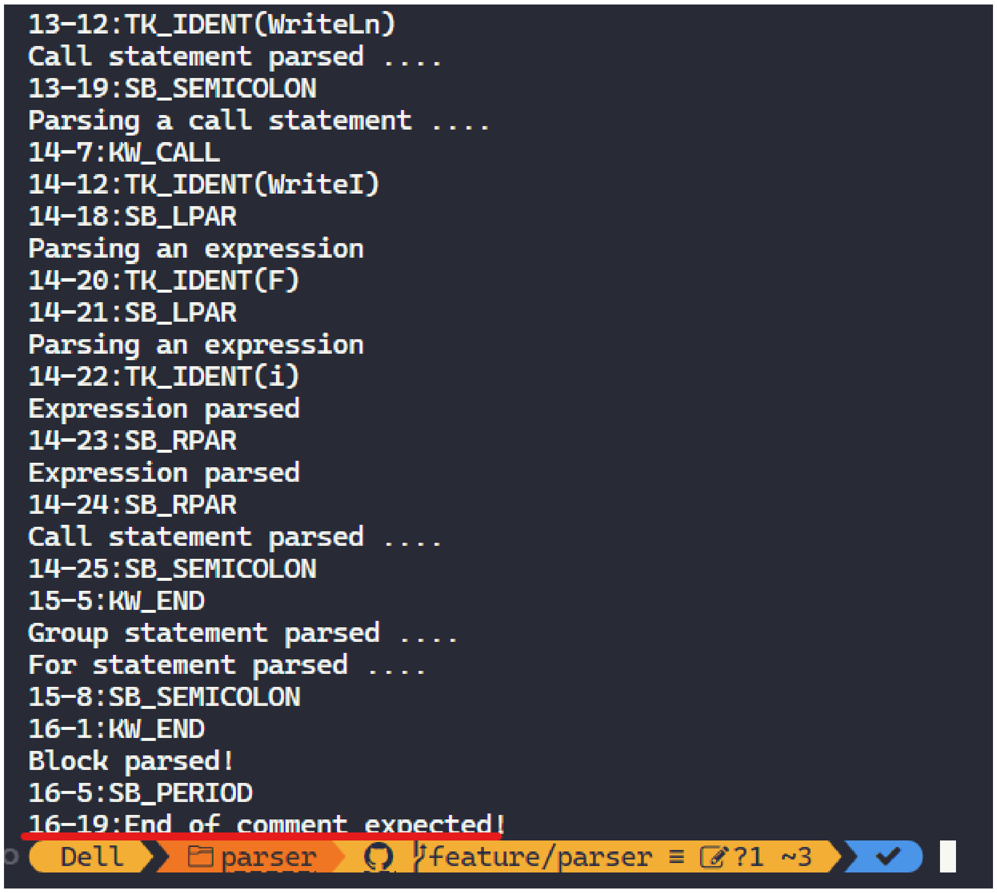
ERM_ENDOFCOMMENT "End of comment expected!"



example2.kpl M ×

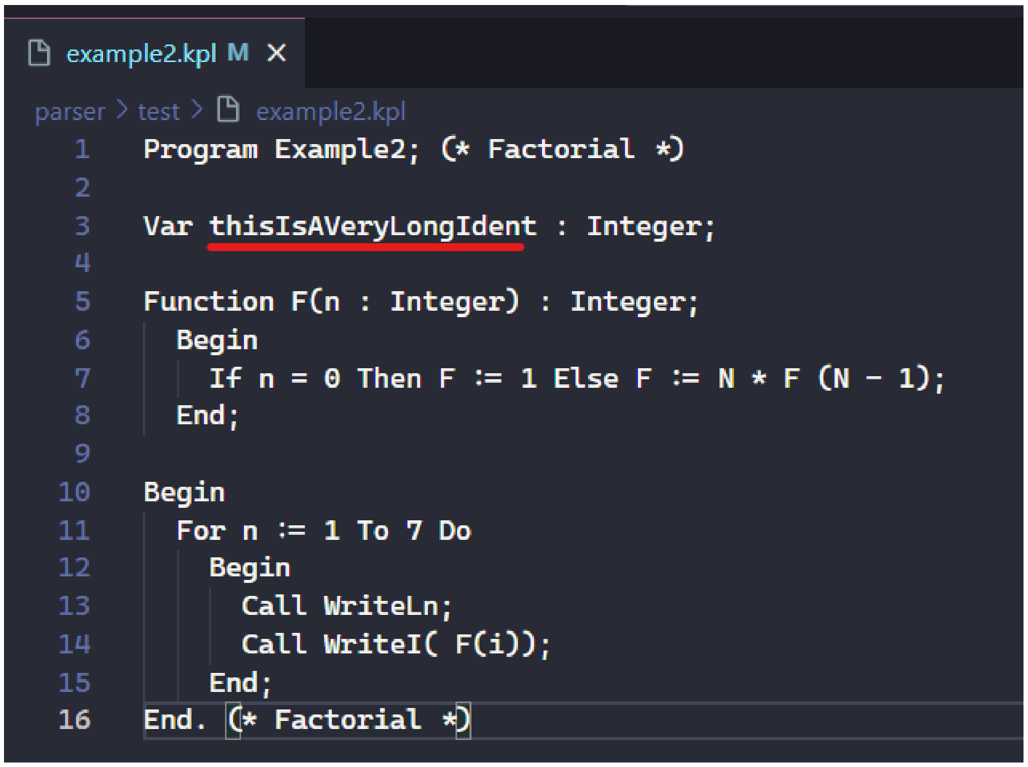
parser > test > example2.kpl

```
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;
      Call WriteI( F(i));
    End;
End. (* Factorial
```



```
13-12:TK_IDENT(WriteLn)
Call statement parsed ....
13-19:SB_SEMICOLON
Parsing a call statement ....
14-7:KW_CALL
14-12:TK_IDENT(WriteI)
14-18:SB_LPAR
Parsing an expression
14-20:TK_IDENT(F)
14-21:SB_LPAR
Parsing an expression
14-22:TK_IDENT(i)
Expression parsed
14-23:SB_RPAR
Expression parsed
14-24:SB_RPAR
Call statement parsed ....
14-25:SB_SEMICOLON
15-5:KW_END
Group statement parsed ....
For statement parsed ....
15-8:SB_SEMICOLON
16-1:KW_END
Block parsed!
16-5:SB_PERIOD
16-19:End of comment expected!
```

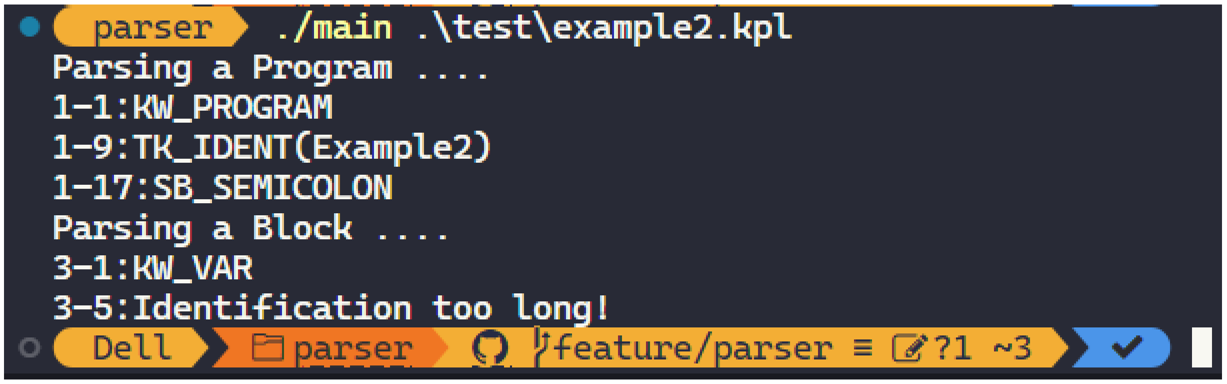
ERM_IDENTTOOLONG "Identification too long!"



example2.kpl M ×

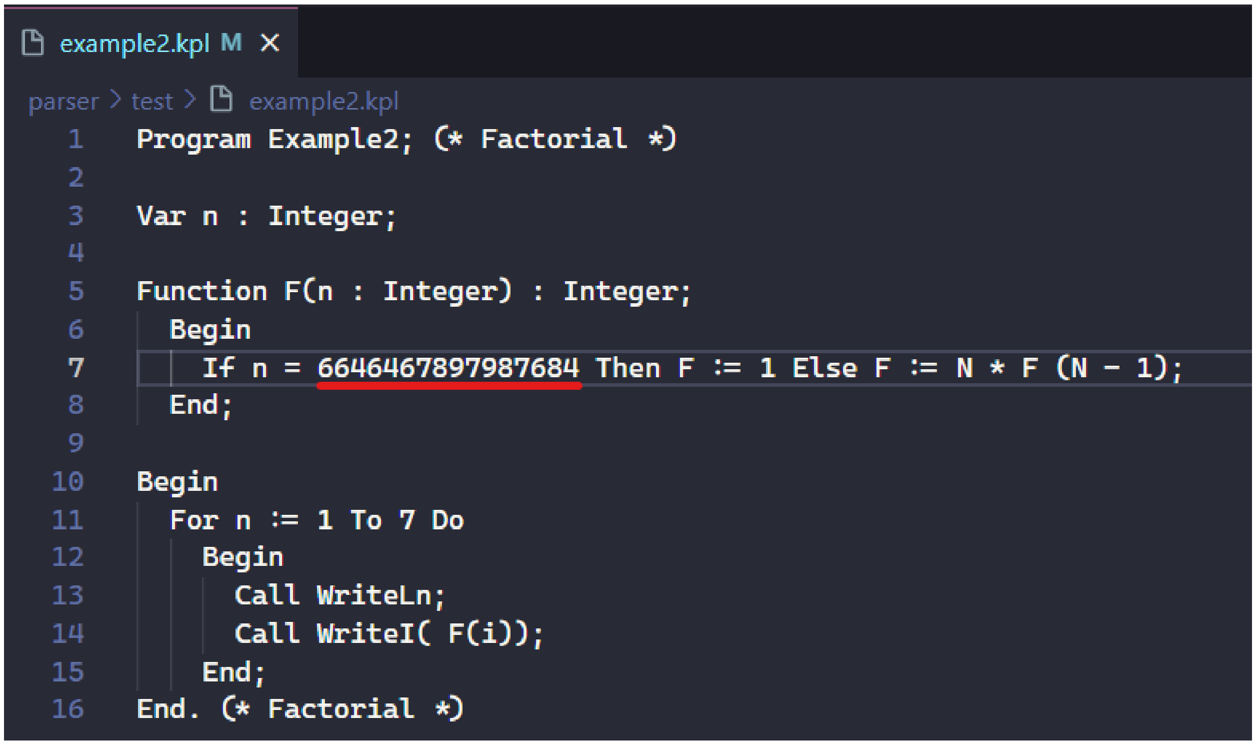
parser > test > example2.kpl

```
Program Example2; (* Factorial *)

Var thisIsAVeryLongIdent : 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;
      Call WriteI( F(i));
    End;
End. (* Factorial *)
```



```
parser  ./main .\test\example2.kpl
Parsing a Program ....
1-1:KW_PROGRAM
1-9:TK_IDENT(Example2)
1-17:SB_SEMICOLON
Parsing a Block ....
3-1:KW_VAR
3-5:Identification too long!
Dell  parser  feature/parser ≡ ?1 ~3
```

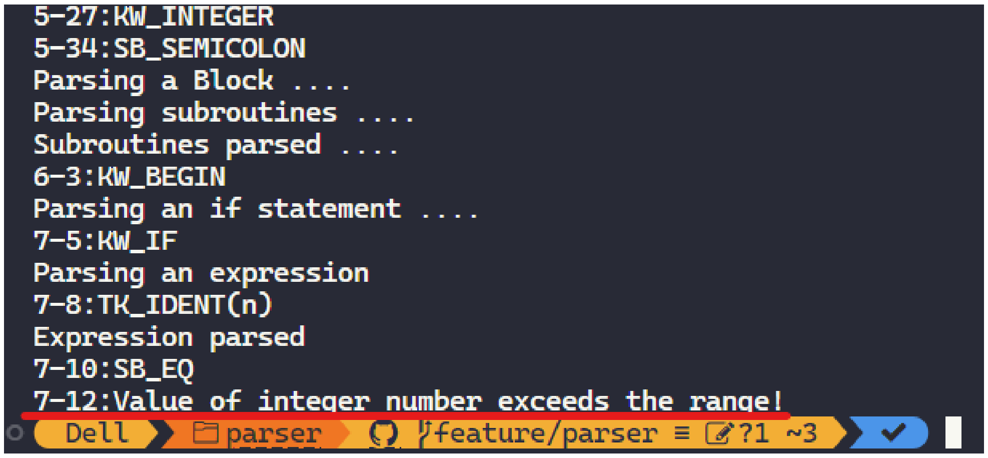
ERM_NUMBERTOOLONG "Value of integer number exceeds the range!"



example2.kpl M ×

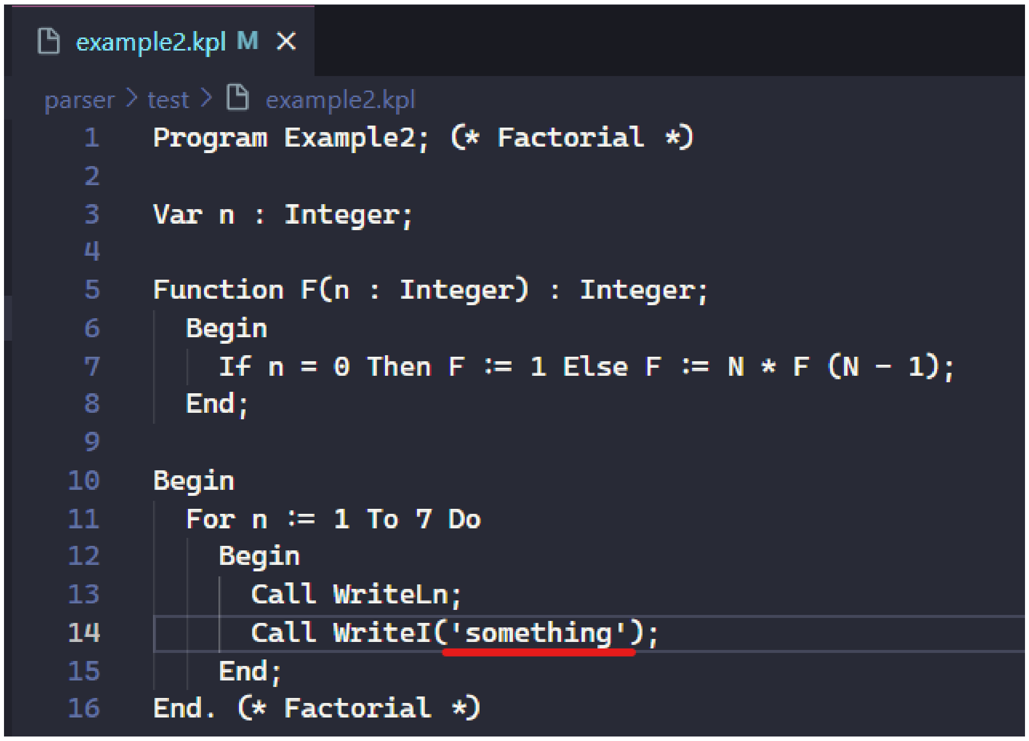
parser > test > example2.kpl

```
Program Example2; (* Factorial *)

Var n : Integer;

Function F(n : Integer) : Integer;
  Begin
    If n = 6646467897987684 Then F := 1 Else F := N * F (N - 1);
  End;

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



```
5-27:KW_INTEGER
5-34:SB_SEMICOLON
Parsing a Block ....
Parsing subroutines ....
Subroutines parsed ....
6-3:KW_BEGIN
Parsing an if statement ....
7-5:KW_IF
Parsing an expression
7-8:TK_IDENT(n)
Expression parsed
7-10:SB_EQ
7-12:Value of integer number exceeds the range!
```

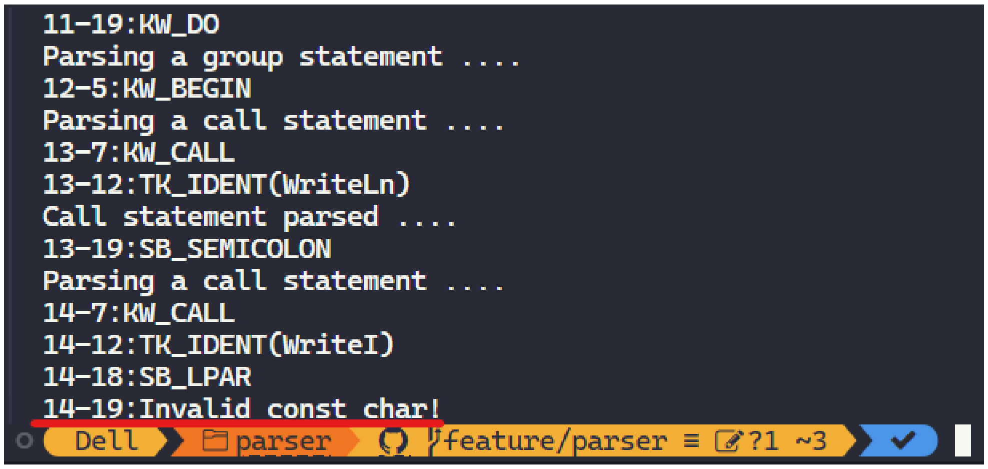
ERM_INVALIDCHARCONSTANT "Invalid const char!"



example2.kpl M ×

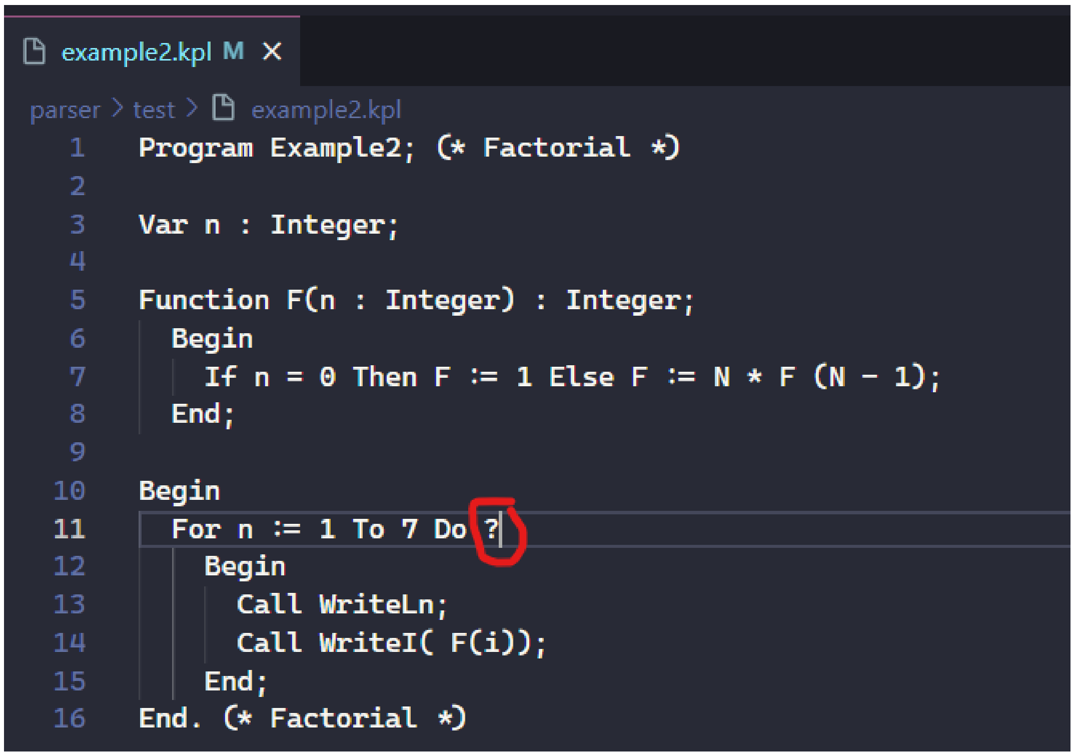
parser > test > example2.kpl

```
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;
      Call WriteI('something');
    End;
End. (* Factorial *)
```



```
11-19:KW_DO
Parsing a group statement ....
12-5:KW_BEGIN
Parsing a call statement ....
13-7:KW_CALL
13-12:TK_IDENT(WriteLn)
Call statement parsed ....
13-19:SB_SEMICOLON
Parsing a call statement ....
14-7:KW_CALL
14-12:TK_IDENT(WriteI)
14-18:SB_LPAR
14-19:Invalid const char!
```

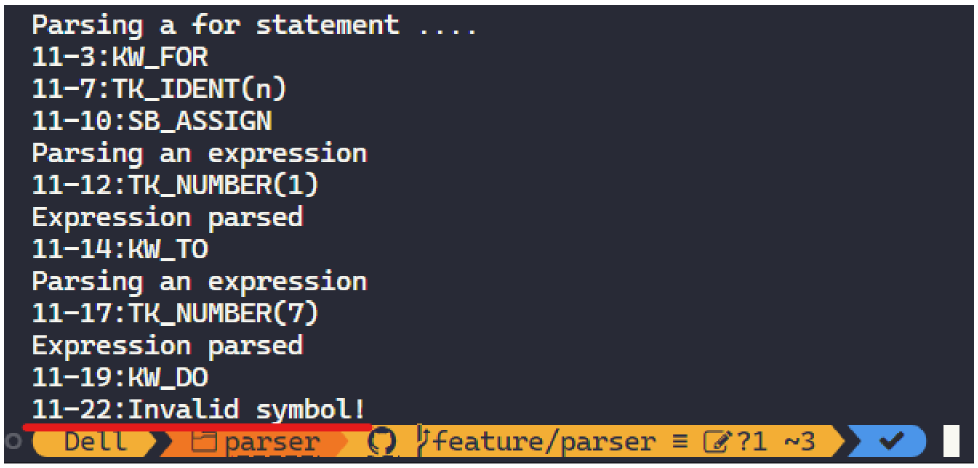
ERM_INVALIDSYMBOL "Invalid symbol!"



example2.kpl M ×

parser > test > example2.kpl

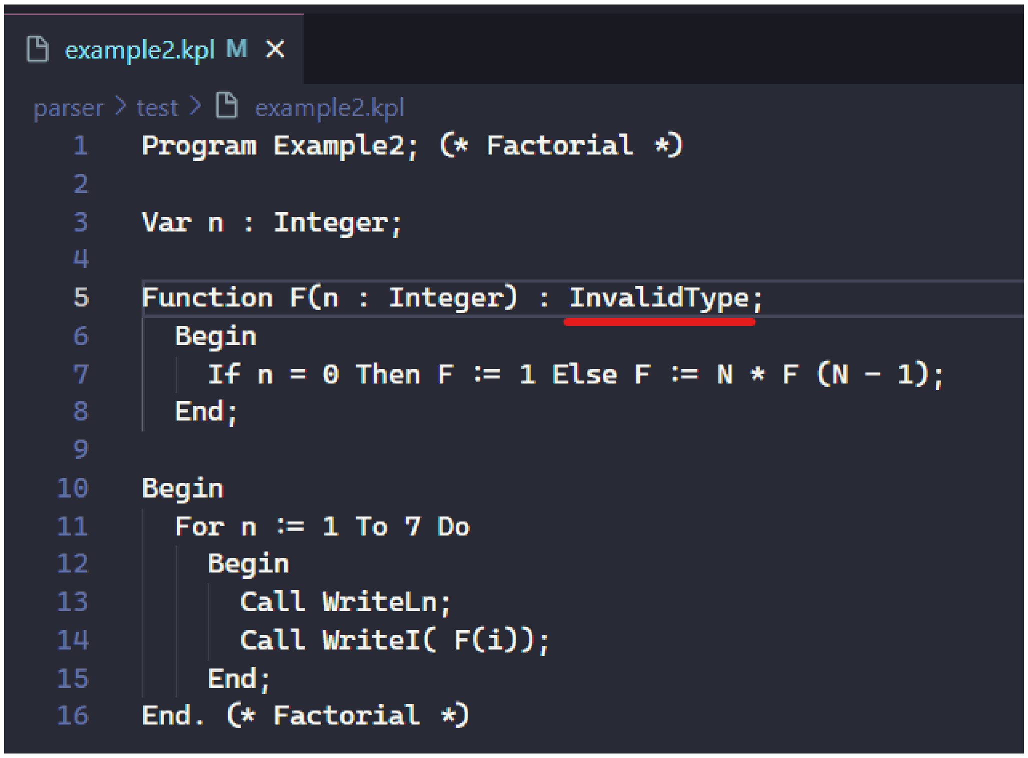
```
Program Example2; (* Factorial *)

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;
      Call WriteI( F(i));
    End;
End. (* Factorial *)
```



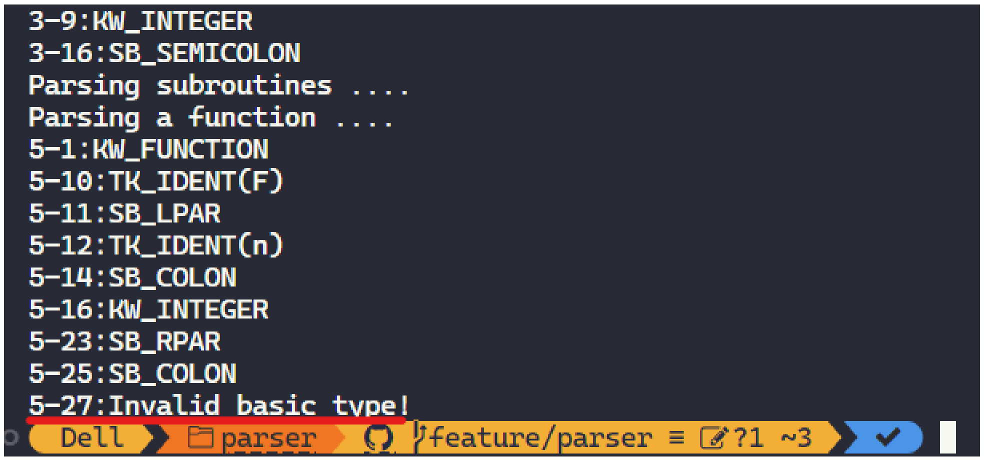
```
Parsing a for statement ....
11-3:KW_FOR
11-7:TK_IDENT(n)
11-10:SB_ASSIGN
Parsing an expression
11-12:TK_NUMBER(1)
Expression parsed
11-14:KW_TO
Parsing an expression
11-17:TK_NUMBER(7)
Expression parsed
11-19:KW_DO
11-22:Invalid symbol!
```

Dell parser feature/parser ≡ ?1 ~3

ERM_INVALIDBASICTYPE "Invalid basic type!"

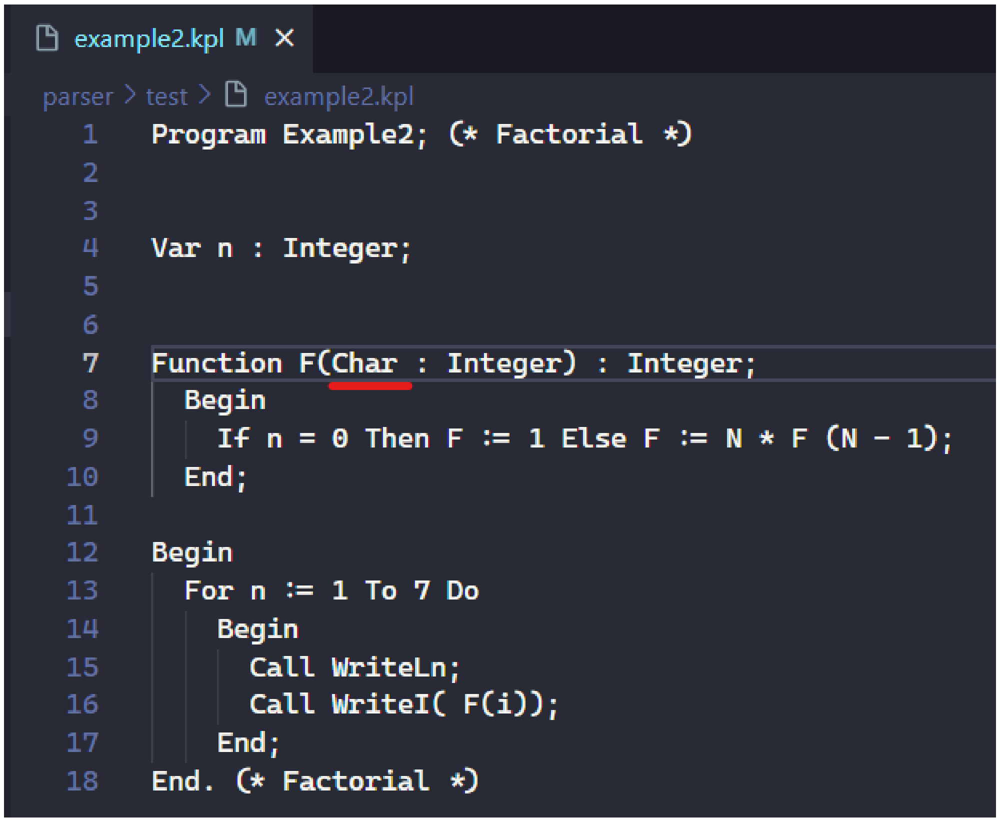


```
example2.kpl M ×
parser > test > example2.kpl
Program Example2; (* Factorial *)

Var n : Integer;

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

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

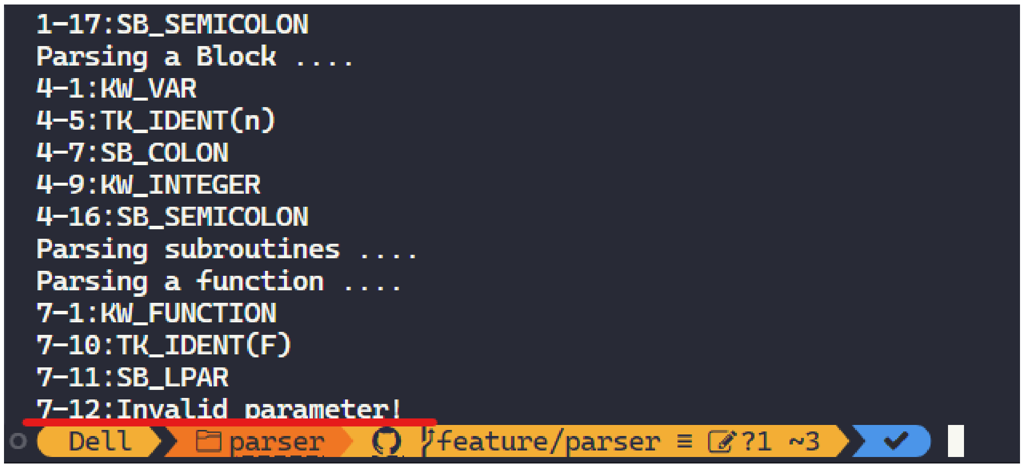


```
3-9:KW_INTEGER
3-16:SB_SEMICOLON
Parsing subroutines ....
Parsing a function ....
5-1:KW_FUNCTION
5-10:TK_IDENT(F)
5-11:SB_LPAR
5-12:TK_IDENT(n)
5-14:SB_COLON
5-16:KW_INTEGER
5-23:SB_RPAR
5-25:SB_COLON
5-27:Invalid basic type!
 Dell  parser   feature/parser ≡ ?1 ~3 ✓
```

ERM_INVALIDPARAM "Invalid parameter!"

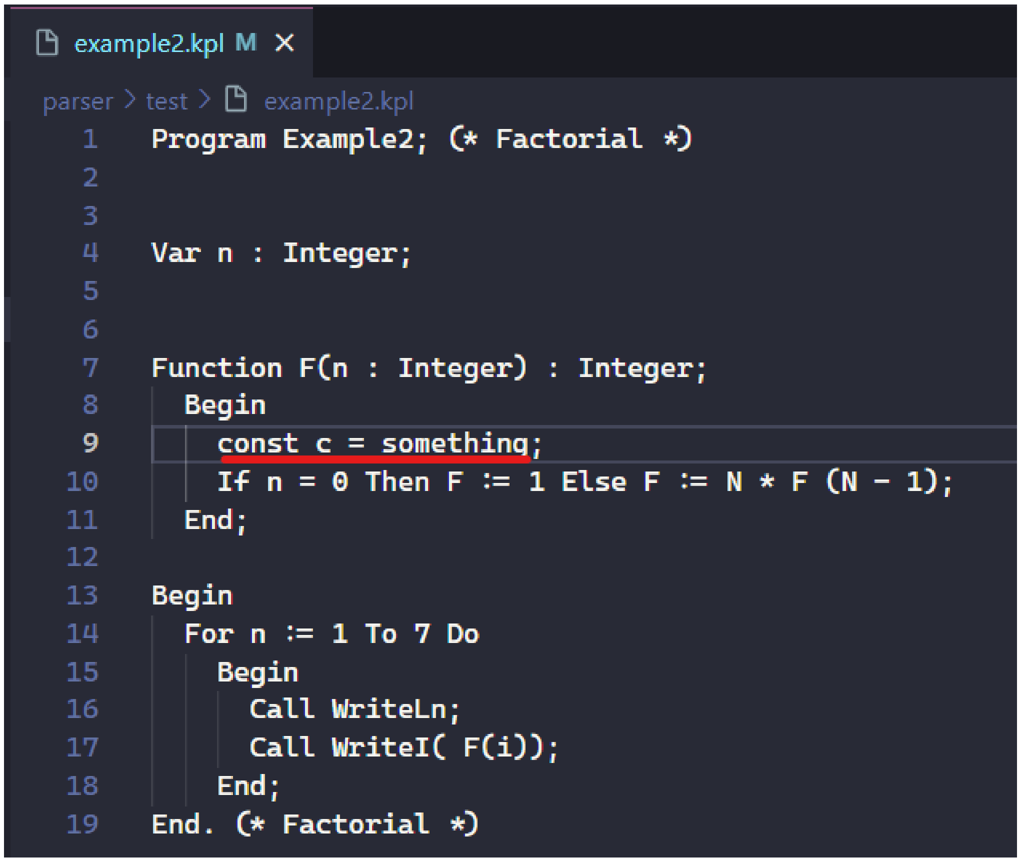


```
Program Example2; (* Factorial *)


Var n : Integer;


Function F(Char : 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;
      Call WriteI( F(i));
    End;
End. (* Factorial *)
```



```
1-17:SB_SEMICOLON
Parsing a Block ....
4-1:KW_VAR
4-5:TK_IDENT(n)
4-7:SB_COLON
4-9:KW_INTEGER
4-16:SB_SEMICOLON
Parsing subroutines ....
Parsing a function ....
7-1:KW_FUNCTION
7-10:TK_IDENT(F)
7-11:SB_LPAR
7-12:Invalid parameter!
```

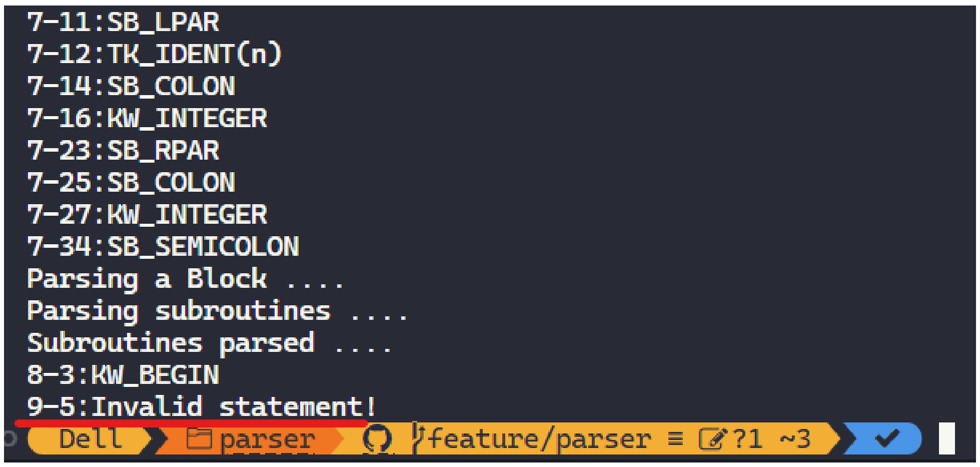
ERM_INVALIDSTATEMENT "Invalid statement!"



example2.kpl M ×

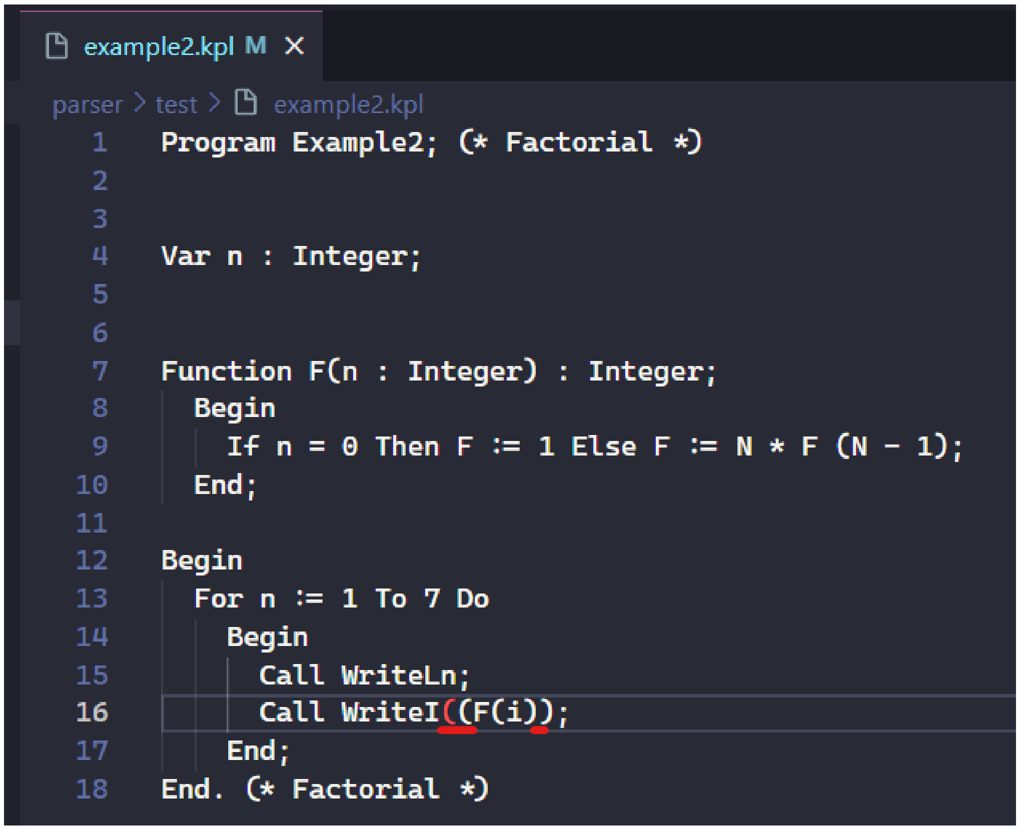
parser > test > example2.kpl

```
Program Example2; (* Factorial *)


Var n : Integer;


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

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



```
7-11:SB_LPAR
7-12:TK_IDENT(n)
7-14:SB_COLON
7-16:KW_INTEGER
7-23:SB_RPAR
7-25:SB_COLON
7-27:KW_INTEGER
7-34:SB_SEMICOLON
Parsing a Block ....
Parsing subroutines ....
Subroutines parsed ....
8-3:KW_BEGIN
9-5:Invalid statement!
```

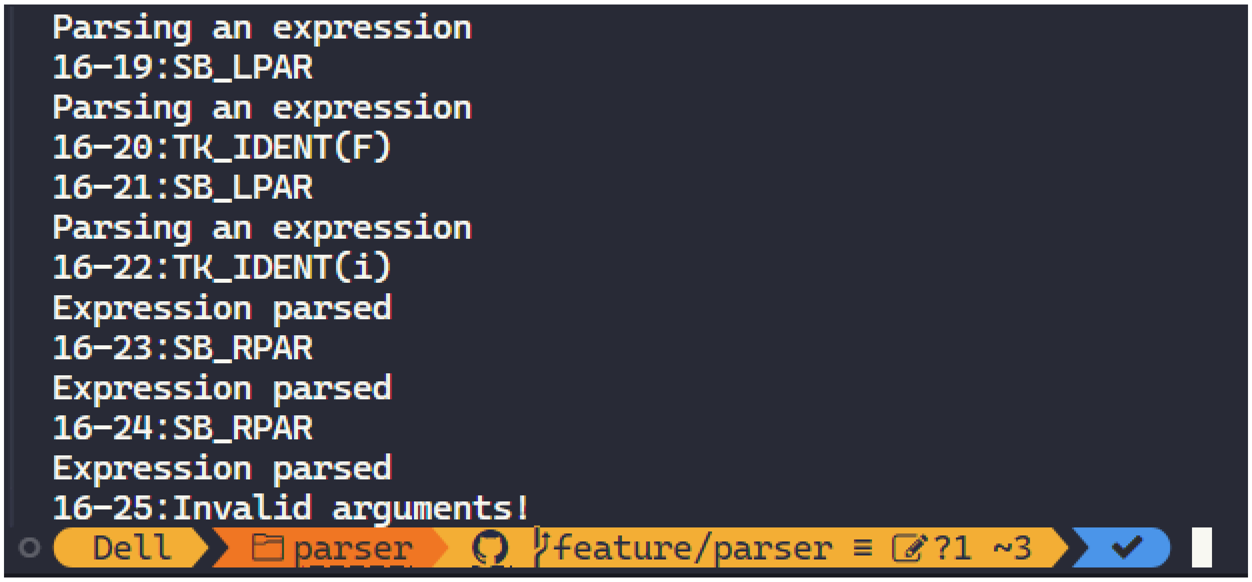
ERM_INVALIDARGUMENTS "Invalid arguments!"



example2.kpl M ×

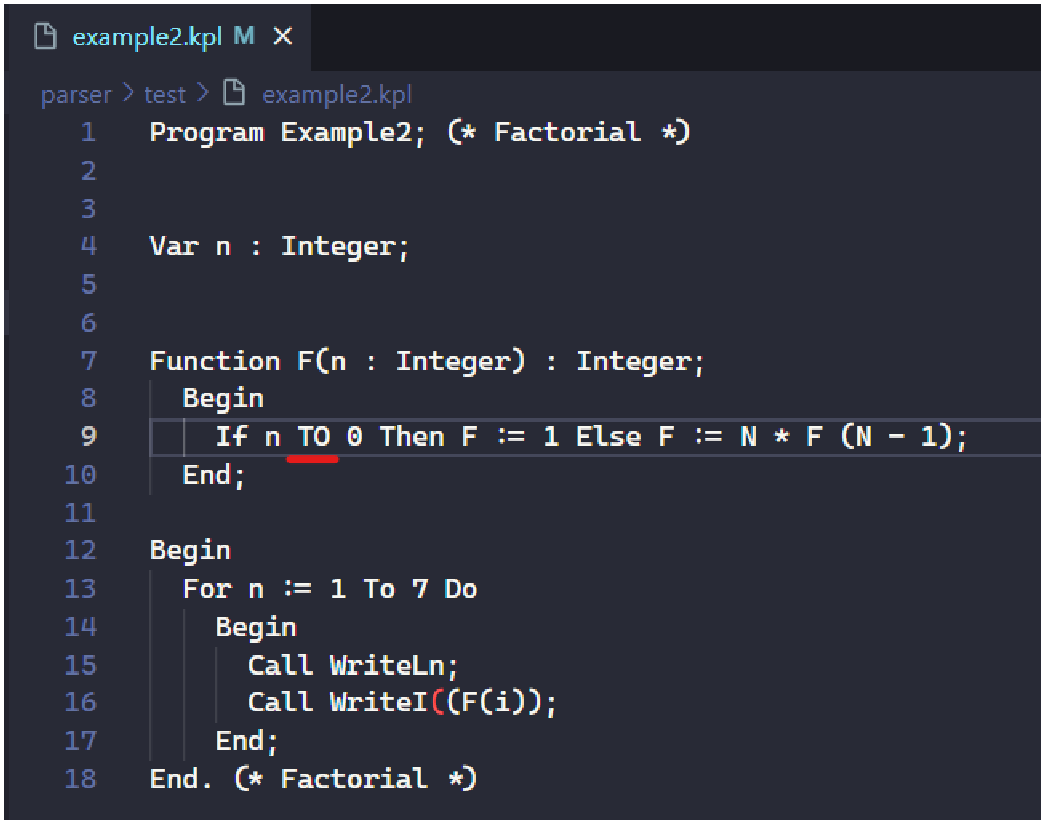
parser > test > example2.kpl

```
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;
      Call WriteI((F(i));
    End;
End. (* Factorial *)
```

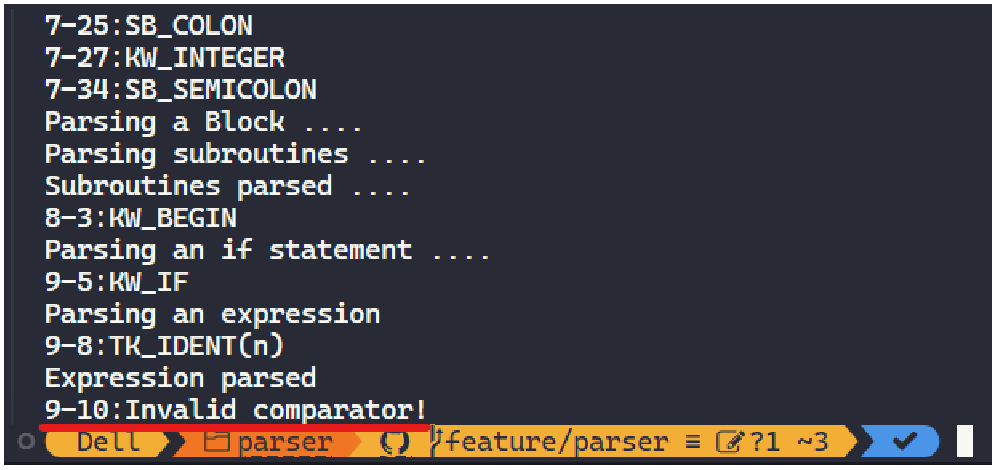


```
Parsing an expression
16-19:SB_LPAR
Parsing an expression
16-20:TK_IDENT(F)
16-21:SB_LPAR
Parsing an expression
16-22:TK_IDENT(i)
Expression parsed
16-23:SB_RPAR
Expression parsed
16-24:SB_RPAR
Expression parsed
16-25:Invalid arguments!
```

ERM_INVALIDCOMPARATOR "Invalid comparator!"

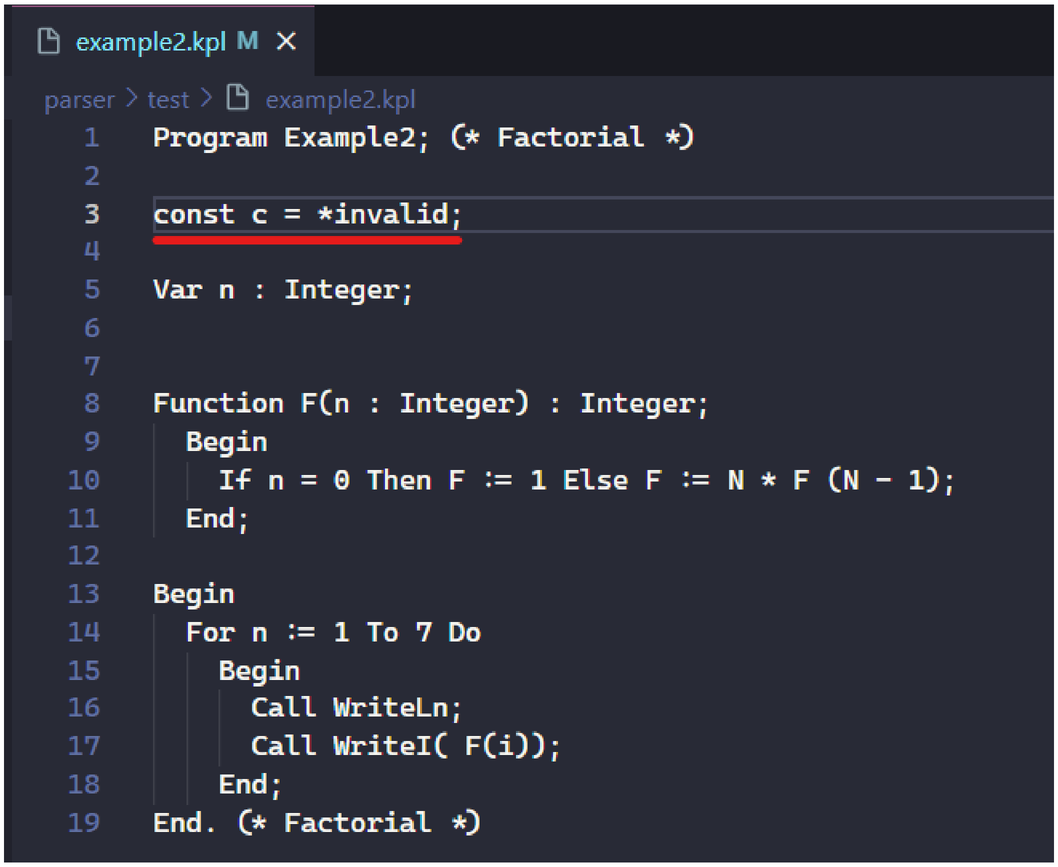


```
Program Example2; (* Factorial *)


Var n : Integer;


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

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

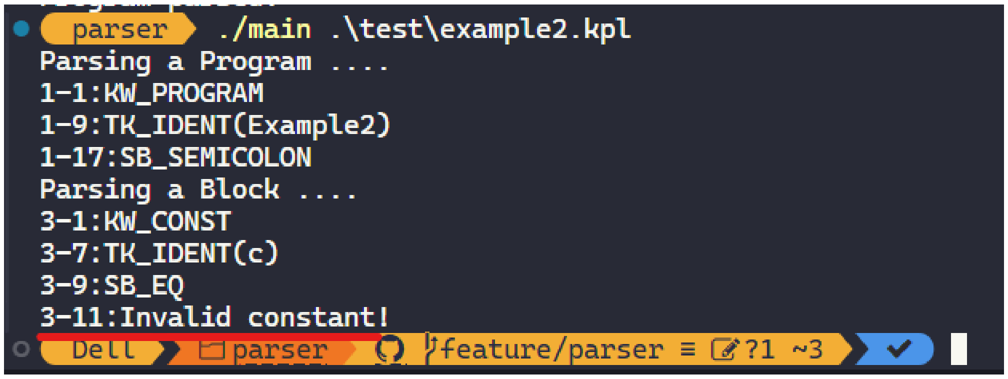


```
7-25:SB_COLON
7-27:KW_INTEGER
7-34:SB_SEMICOLON
Parsing a Block ....
Parsing subroutines ....
Subroutines parsed ....
8-3:KW_BEGIN
Parsing an if statement ....
9-5:KW_IF
Parsing an expression
9-8:TK_IDENT(n)
Expression parsed
9-10:Invalid comparator!
```

ERM_INVALIDCONSTANT "Invalid constant!"

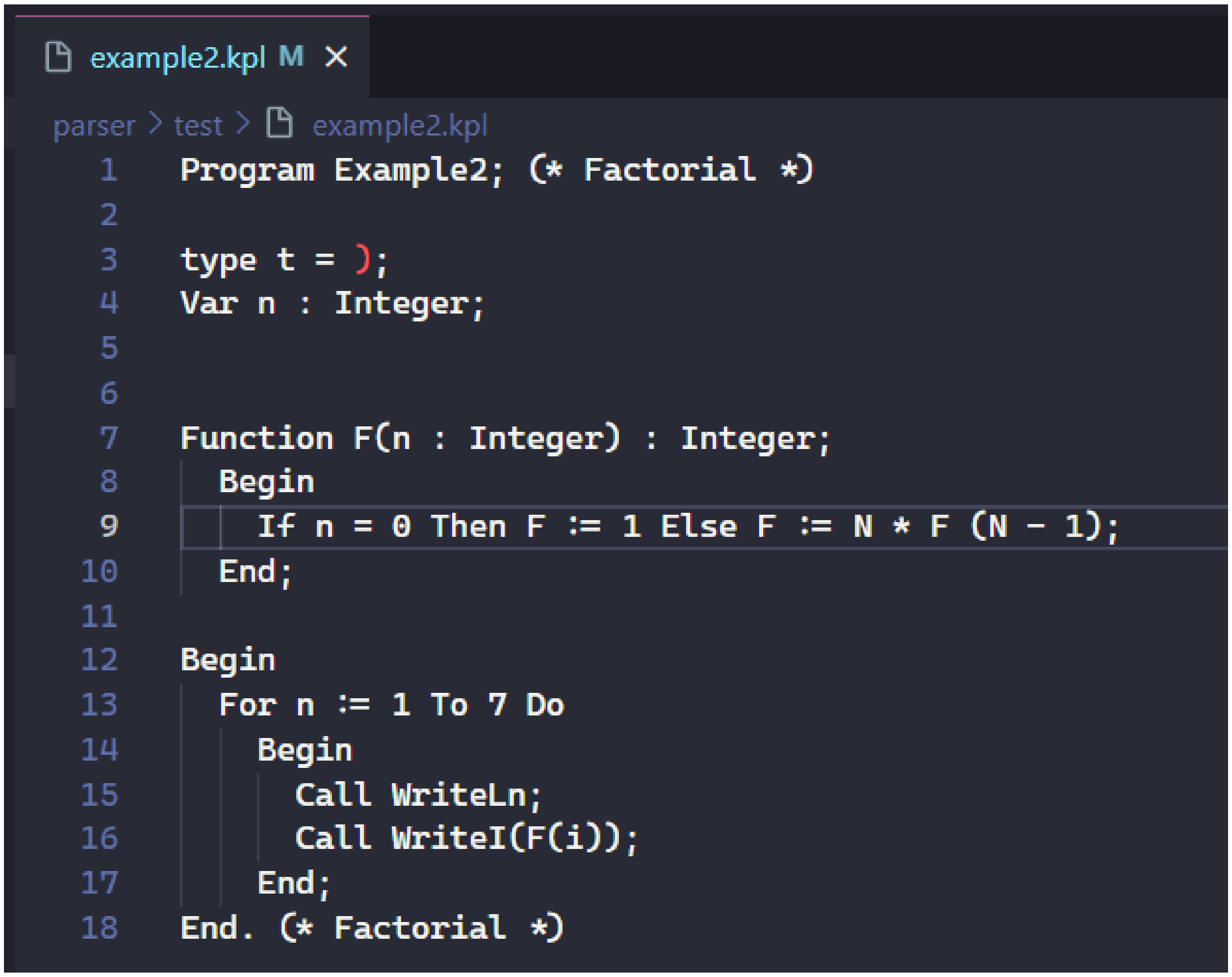


```
Program Example2; (* Factorial *)

const c = *invalid;

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;
      Call WriteI( F(i));
    End;
End. (* Factorial *)
```



```
parser  ./main .\test\example2.kpl
Parsing a Program ....
1-1:KW_PROGRAM
1-9:TK_IDENT(Example2)
1-17:SB_SEMICOLON
Parsing a Block ....
3-1:KW_CONST
3-7:TK_IDENT(c)
3-9:SB_EQ
3-11:Invalid constant!
```

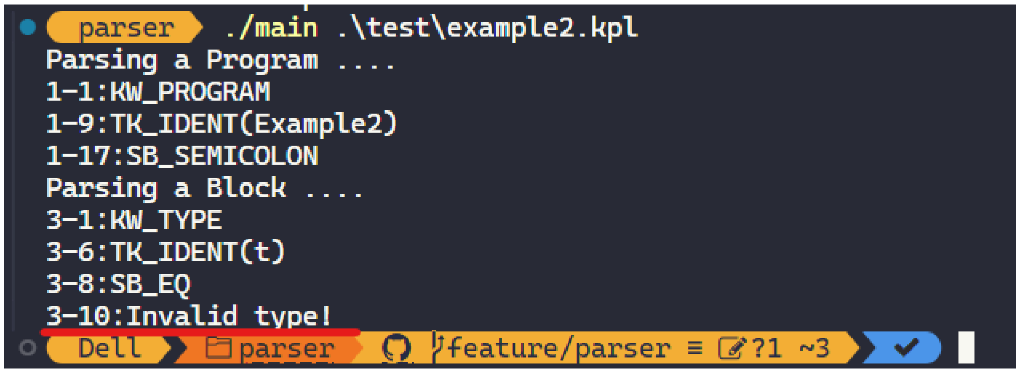
ERM_INVALIDTYPE "Invalid type!"



example2.kpl M ×

parser > test > example2.kpl

```
Program Example2; (* Factorial *)

type t = );
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;
      Call WriteI(F(i));
    End;
End. (* Factorial *)
```

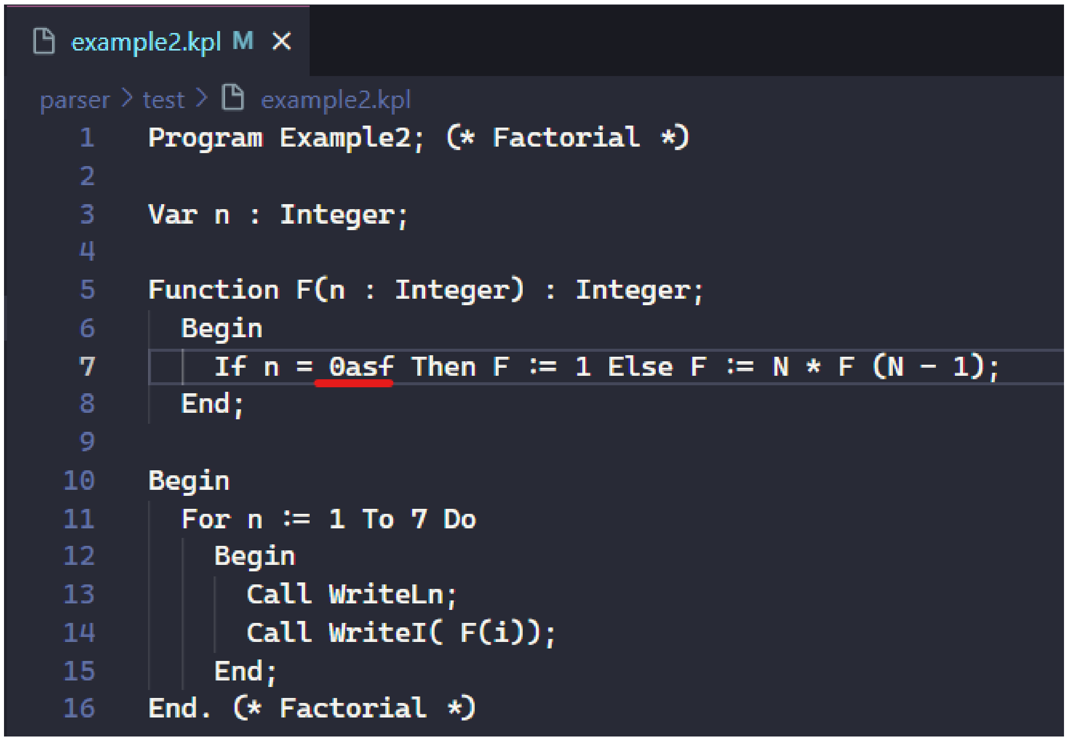


```
parser  ./main .\test\example2.kpl
Parsing a Program ....
1-1:KW_PROGRAM
1-9:TK_IDENT(Example2)
1-17:SB_SEMICOLON
Parsing a Block ....
3-1:KW_TYPE
3-6:TK_IDENT(t)
3-8:SB_EQ
3-10:Invalid type!
Dell  parser  feature/parser ≡ ?1 ~3 ✓
```

ERM_INVALIDEXPRESSION "Invalid expression!"

This is not required in the assignment instructions. To do this, we would need to compute FIRST and FOLLOW sets of all symbols – which is quite a demanding task itself.

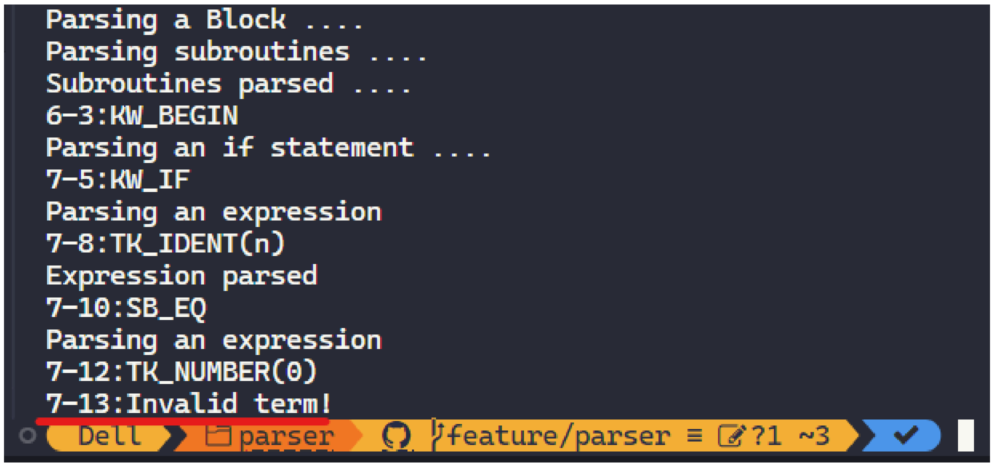
ERM_INVALIDTERM "Invalid term!"



example2.kpl M ×

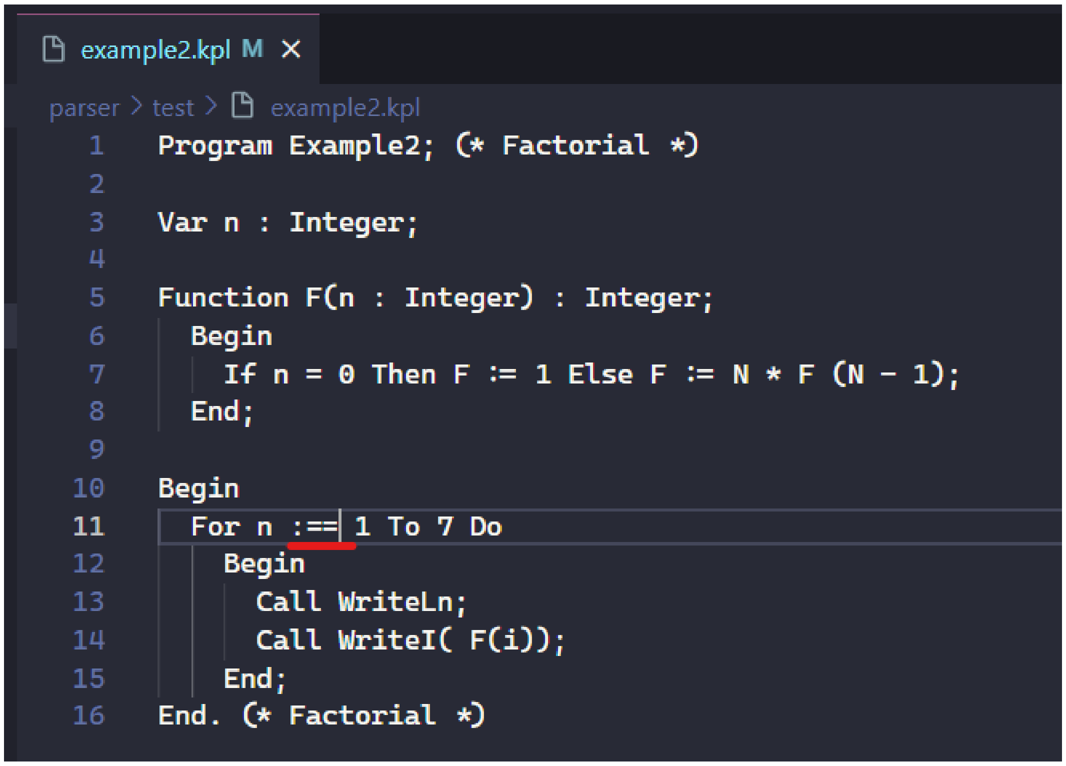
parser > test > example2.kpl

```
Program Example2; (* Factorial *)

Var n : Integer;

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

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



```
Parsing a Block ....
Parsing subroutines ....
Subroutines parsed ....
6-3:KW_BEGIN
Parsing an if statement ....
7-5:KW_IF
Parsing an expression
7-8:TK_IDENT(n)
Expression parsed
7-10:SB_EQ
Parsing an expression
7-12:TK_NUMBER(0)
7-13:Invalid term!
```

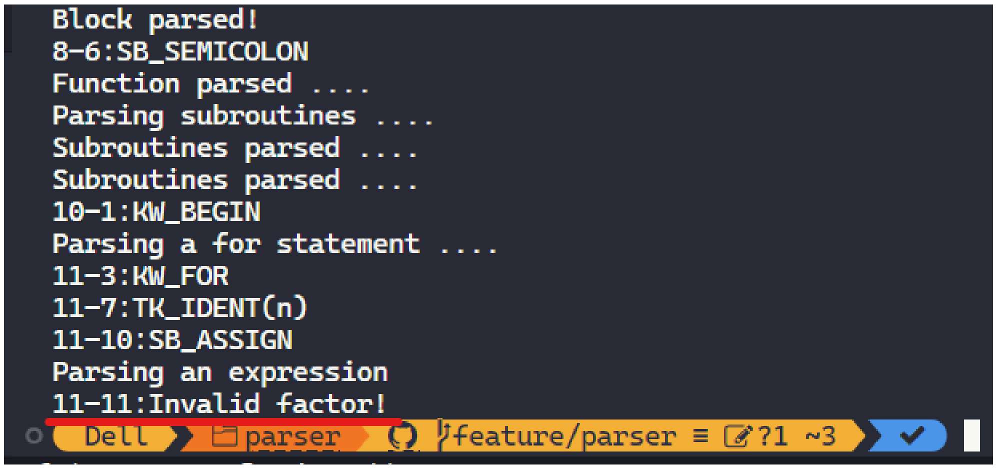
ERM_INVALIDFACTOR "Invalid factor!"



example2.kpl M ×

parser > 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;
      Call WriteI( F(i));
    End;
End. (* Factorial *)
```



```
Block parsed!
8-6:SB_SEMICOLON
Function parsed ....
Parsing subroutines ....
Subroutines parsed ....
Subroutines parsed ....
10-1:KW_BEGIN
Parsing a for statement ....
11-3:KW_FOR
11-7:TK_IDENT(n)
11-10:SB_ASSIGN
Parsing an expression
11-11:Invalid factor!
```

THE END