BÁO CÁO THỰC HÀNH XÂY DỰNG CHƯƠNG TRÌNH DỊCH

BÀI 2: PHÂN TÍCH CÚ PHÁP

Họ và tên: Vũ Văn Minh

Mssv: 20225747

I. Giải thích code

1. Cơ chế Hoạt động Cơ bản

a. Hàm scan() – Đọc token tiếp theo
Chức năng: Di chuyển cửa sổ đọc token sang phải một vị trí.
Cơ chế:

1. Lưu currentToken vào biến tạm (để giải phóng sau).
2. Gán giá trị của lookAhead cho currentToken.
3. Gọi Scanner để đọc token mới từ mã nguồn và gán vào lookAhead.
4. Giải phóng bộ nhớ của token cũ.

b. Hàm eat(tokenType) – Kiểm tra và tiêu thụ token
Chức năng: Xác nhận token hiện tại có đúng loại mong đợi không.
Quy trình:

1. Kiểm tra: lookAhead có khớp với tokenType truyền vào không?
2. Nếu khớp: In token ra console (debug) -> Gọi scan() để đi tiếp.
3. Nếu không khớp: Báo lỗi missingToken().

2. Cấu trúc Chương trình (Program Structure)

a. Cấu trúc tổng quát của KPL
Một chương trình KPL chuẩn bao gồm:

- PROGRAM + <Tên chương trình> + ;
- Khai báo CONST (tùy chọn)
- Khai báo TYPE (tùy chọn)
- Khai báo VAR (tùy chọn)
- Khai báo FUNCTION / PROCEDURE (tùy chọn)
- Thân chương trình: BEGIN ... END.

b. Hàm compileProgram()
Xử lý toàn bộ chương trình theo trình tự nghiêm ngặt:

1. In thông báo bắt đầu parse.
2. eat(KW_PROGRAM): Bắt buộc có từ khóa PROGRAM.

3. eat(TK_IDENT): Bắt buộc có tên chương trình.
4. eat(SB_SEMICOLON): Bắt buộc có dấu chấm phẩy.
5. Gọi compileBlock(): Xử lý toàn bộ phần thân.
6. eat(SB_PERIOD): Bắt buộc kết thúc bằng dấu chấm (.).
7. In thông báo hoàn thành.

3. Xử lý Khối lệnh (Block Parsing)

Block được xử lý tuần tự qua 5 giai đoạn (tương ứng 5 hàm con):

Hàm	Nhiệm vụ	Logic xử lý
compileBlock()	Xử lý CONST	Kiểm tra lookAhead. Nếu là KW_CONST -> eat -> gọi compileConstDecl -> sang bước 2. Ngược lại nhảy thẳng sang bước 2.
compileBlock2()	Xử lý TYPE	Kiểm tra KW_TYPE. Nếu có -> xử lý -> sang bước 3. Ngược lại nhảy sang bước 3.
compileBlock3()	Xử lý VAR	Kiểm tra KW_VAR. Nếu có -> xử lý -> sang bước 4. Ngược lại nhảy sang bước 4.
compileBlock4()	Xử lý Subroutines	Dùng vòng lặp kiểm tra KW_FUNCTION hoặc KW_PROCEDURE. Gọi hàm xử lý tương ứng cho đến khi hết.
compileBlock5()	Xử lý Thân BEGIN..END	Bắt buộc eat(KW_BEGIN) -> gọi compileStatements() -> bắt buộc eat(KW_END).

4. Chi tiết Xử lý Khai báo (Declarations)

a. Khai báo Hằng (CONST)

- compileConstDecls(): Dùng vòng lặp while để xử lý danh sách hằng liên tiếp. Dừng khi không gặp TK_IDENT.
- compileConstDecl(): Xử lý một hằng theo cú pháp: Tên = Giá_trị ;

  Logic: eat(IDENT) -> eat(EQ) -> compileConstant() -> eat(SEMICOLON).

b. Khai báo Kiểu (TYPE)

- compileTypeDecls(): Tương tự hằng, lặp qua các định nghĩa kiểu.
- compileTypeDecl(): Xử lý theo cú pháp: Tên_kiểu = Định_nghĩa ;

Logic: eat(IDENT) -> eat(EQ) -> compileType() -> eat(SEMICOLON).

c. Khai báo Biến (VAR)

- compileVarDecls(): Lặp qua các khai báo biến.
- compileVarDecl(): Xử lý theo cú pháp: Tên_biến : Kiểu ;

Logic: eat(IDENT) -> eat(COLON) -> compileType() -> eat(SEMICOLON).

d. Xử lý Giá trị Hằng & Kiểu dữ liệu

- Các loại hằng:
  - Không dấu (compileUnsignedConstant): Số nguyên, Tên hằng, Ký tự (char).
  - Có dấu (compileConstant): Xử lý dấu + hoặc - trước số nguyên.
- Các loại kiểu dữ liệu (compileType):

1. INTEGER: Số nguyên.
2. CHAR: Ký tự.
3. ARRAY: Mảng (Cú pháp: ARRAY[.size.] OF Type). Có thể đệ quy.
4. IDENT: Kiểu tự định nghĩa.

5. Xử lý Hàm và Thủ tục (Subroutines)

a. Hàm compileFuncDecl() (Function)

Đặc điểm: Có giá trị trả về (INTEGER hoặc CHAR).
Cú pháp: FUNCTION Tên(Tham_số) : Kiểu_trả_về ; Block ;

1. eat(KW_FUNCTION) -> eat(IDENT).
2. compileParams(): Xử lý tham số.
3. eat(SB_COLON) -> compileBasicType() (Lấy kiểu trả về).
4. eat(SB_SEMICOLON).
5. compileBlock(): Đệ quy xử lý thân hàm.
6. eat(SB_SEMICOLON).

b. Hàm compileProcDecl() (Procedure)

Đặc điểm: Không có giá trị trả về.
Cú pháp: PROCEDURE Tên(Tham_số) ; Block ;

Quy trình: Tương tự Function nhưng KHÔNG có bước kiểm tra kiểu trả về (không có dấu : và Type).

c. Xử lý Tham số (compileParams)

- Tham trị (Pass by value): x : INTEGER (Thay đổi trong hàm không ảnh hưởng bên ngoài).
- Tham chiếu (Pass by reference): VAR y : CHAR (Thay đổi trong hàm ảnh hưởng bên ngoài).

6. Xử lý Câu lệnh (Statements)

Hàm compileStatement() phân loại và gọi hàm xử lý tương ứng dựa trên token đầu tiên:

Loại câu lệnh	Token bắt đầu	Hàm xử lý	Cú pháp ví dụ
Gán	TK_IDENT	compileAssignSt	x := 5 hoặc a[i] := 1
Gọi Procedure	KW_CALL	compileCallSt	CALL WriteLn
Khối lệnh	KW_BEGIN	compileGroupSt	BEGIN ... END
Điều kiện	KW_IF	compileIfSt	IF x > 0 THEN ... [ELSE ...]
Vòng lặp While	KW_WHILE	compileWhileSt	WHILE i < 10 DO ...
Vòng lặp For	KW_FOR	compileForSt	FOR i:=1 TO 10 DO ...
Rỗng	; hoặc END	(Không làm gì)	;;

7. Xử lý Biểu thức (Expression Logic)

Cấu trúc phân cấp: Expression -> Term -> Factor

a. compileExpression() - Xử lý phép cộng/trừ

- Logic:

  1. Kiểm tra tính hợp lệ của token đầu tiên.
  2. Gọi compileTerm() (xử lý *, /).
  3. Gọi compileExpression3(): Vòng lặp xử lý +, -.

- Follow Set: Dừng parse khi gặp TO, DO,),], ,, THEN, ELSE...

b. compileTerm() - Xử lý phép nhân/chia

- Logic: Gọi compileFactor() -> Gọi compileTerm2() (xử lý *, /).

c. compileFactor() - Thành phần nhỏ nhất

- Xử lý: TK_NUMBER, TK_CHAR, TK_IDENT (biến/mảng/hàm), SB_LPAR (biểu thức trong ngoặc).

d. Xử lý Điều kiện (compileCondition)

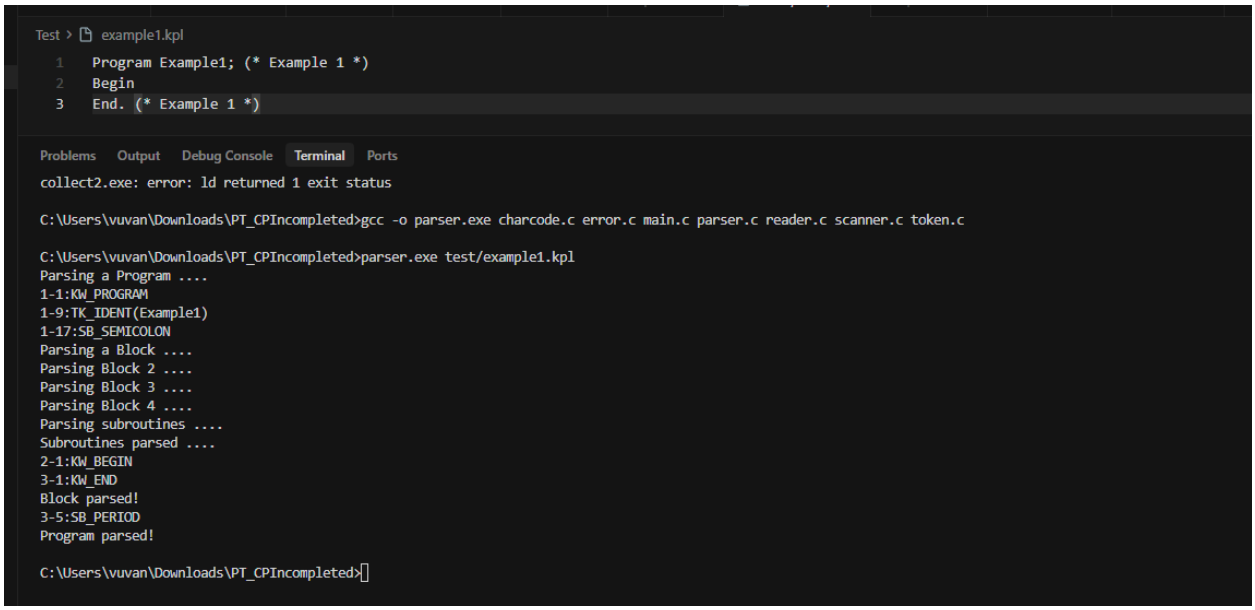
Cú pháp: Biểu_thức_1 Toán_tử Biểu_thức_2
Toán tử so sánh: =, !=, <, <=, >, >=.

8. Hàm Main - Entry Point

Hàm compile(fileName) thực hiện các bước:

1. Mở file: openInputStream(fileName).
2. Khởi tạo: lookAhead = Token đầu tiên.
3. Thực thi: Gọi compileProgram().
4. Dọn dẹp: Giải phóng bộ nhớ (free), đóng file stream.
5. Kết thúc: Trả về IO_SUCCESS hoặc IO_ERROR.

II. Kết quả thực hiện với example1.kpl (không lỗi)



III. Kết quả thực hiện với các trường hợp lỗi

a. ERR_END_OF_COMMENT

Test > error1.kpl

```
1  PROGRAM Test1;
2  BEGIN
3      (* comment
4      x := 5
5  END
```

Problems Output Debug Console Terminal Ports

```
C:\Users\vuvan\Downloads\PT_CPIncompleted>parser.exe test/error1.kpl
Parsing a Program ....
1-1:KW_PROGRAM
1-9:TK_IDENT(Test1)
1-14:SB_SEMICOLON
Parsing a Block ....
Parsing Block 2 ....
Parsing Block 3 ....
Parsing Block 4 ....
Parsing subroutines ....
Subroutines parsed ....
2-1:KW_BEGIN
5-4:End of comment expected!
```

i. Dòng 3: Mở comment bằng (* nhưng không có đóng
ii. Scanner đọc hết file trong trạng thái comment
iii. Kết quả: 6-1:End of comment expected!

b. ERR_IDENT_TOO_LONG

Test > error2.kpl

```
1  PROGRAM Test2;
2  VAR
3      jsdfkdsflasdlsadloasepasfoxkczxkawernasdasjkroasdsadfasKDsnczxnmcznaskdzjhkczkfdkasjdasdkjacll : INTEGER
4  BEGIN
5  END
```

Problems Output Debug Console Terminal Ports

```
C:\Users\vuvan\Downloads\PT_CPIncompleted>parser.exe test/error2.kpl
Parsing a Program ....
1-1:KW_PROGRAM
1-9:TK_IDENT(Test2)
1-14:SB_SEMICOLON
Parsing a Block ....
Parsing Block 2 ....
Parsing Block 3 ....
2-1:KW_VAR
3-5:Identification too long!
```

i. Dòng 3: Tên biến quá dài, vượt quá giới hạn MAX_IDENT_LEN = 15
ii. Kết quả: 3-5:Identification too long!

c. ERR_NUMBER_TOO_LONG

Test > error3.kpl

```
1  PROGRAM Test;
2  VAR
3      x : INTEGER;
4  BEGIN
5      x := 99999999999999999999999999999;
6  END
```

Problems Output Debug Console Terminal Ports

```
C:\Users\vuvan\Downloads\PT_CPIncompleted>parser.exe test/error3.kpl
Parsing a Program ....
Parsing a Program ....
1-1:KW_PROGRAM
1-9:TK_IDENT(Test)
1-13:SB_SEMICOLON
Parsing a Block ....
Parsing Block 2 ....
Parsing Block 3 ....
2-1:KW_VAR
3-5:TK_IDENT(x)
3-7:SB_COLON
3-9:KW_INTEGER
3-16:SB_SEMICOLON
Parsing Block 4 ....
Parsing subroutines ....
Subroutines parsed ....
4-1:KW_BEGIN
Parsing an assign statement ....
5-5:TK_IDENT(x)
5-8:SB_ASSIGN
5-10:Value of integer number exceeds the range!
```

i. Dòng 5: Số lớn hơn 10 chữ số, INT_MAX

ii. Kết quả: 5-10: Value of integer number exceeds the range!

d. ERR_INVALIDCHARCONSTANT

Test > error4.kpl

```
1  PROGRAM TestChar;
2  CONST
3      letter = 'ab';
4  BEGIN
5  END.
```

Problems Output Debug Console Terminal Ports

```
C:\Users\vuvan\Downloads\PT_CPIncompleted>parser.exe test/error4.kpl
Parsing a Program ....
1-1:KW_PROGRAM
1-9:TK_IDENT(TestChar)
1-17:SB_SEMICOLON
Parsing a Block ....
2-1:KW_CONST
3-5:TK_IDENT(letter)
3-12:SB_EQ
3-15:Invalid const char!
```

i. Dòng 3: Bị lỗi do char ‘ab’ chứa 2 kí tự

ii. Kết quả: 3-15:Invalid const char!

e. ERR_INVALID_SYMBOL

Test > error5.kpl

```
1  PROGRAM TestSymbol;
2  VAR
3      x : INTEGER;
4  BEGIN
5      x := 10 @ 5;
6  END.
```

Problems Output Debug Console Terminal Ports

```
C:\Users\vuvan\Downloads\PT_CPIncompleted>parser.exe test/error5.kpl
Parsing a Program ....
1-1:KW_PROGRAM
1-9:TK_IDENT(TestSymbol)
1-19:SB_SEMICOLON
Parsing a Block ....
Parsing Block 2 ....
Parsing Block 3 ....
2-1:KW_VAR
3-5:TK_IDENT(x)
3-7:SB_COLON
3-9:KW_INTEGER
3-16:SB_SEMICOLON
Parsing Block 4 ....
Parsing subroutines ....
Subroutines parsed ....
4-1:KW_BEGIN
Parsing an assign statement ....
5-5:TK_IDENT(x)
5-8:SB_ASSIGN
Parsing an expression
5-10:TK_NUMBER(10)
5-13:Invalid symbol!
```

i. Dòng 5: chứa kí tự @ không thuộc bảng kí tự hợp lệ kpl

ii. Kết quả: 5-13:Invalid symbol!

f. ERR_INVALID_CONSTANT

Test > error6.kpl

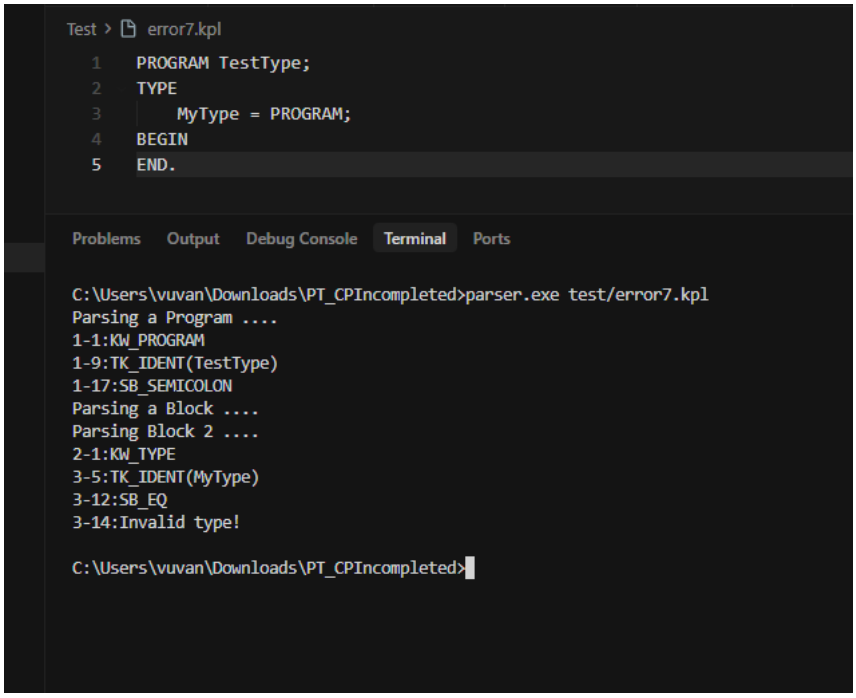
```
1  PROGRAM TestConstant;
2  CONST
3      c = END;
4  BEGIN
5  END.
```

Problems Output Debug Console Terminal Ports

```
C:\Users\vuvan\Downloads\PT_CPIncompleted>parser.exe test/error6.kpl
Parsing a Program ....
1-1:KW_PROGRAM
1-9:TK_IDENT(TestConstant)
1-21:SB_SEMICOLON
Parsing a Block ....
2-1:KW_CONST
3-5:TK_IDENT(c)
3-7:SB_EQ
3-9:Invalid constant!
```

i. Dòng 3 sử dụng từ end làm giá trị constant
ii. Gây lỗi 3-9:Invalid constant!

g. ERR_INVALIDTYPE



i. Dòng 3 sử dụng từ program làm định nghĩa cho type
ii. Parser chỉ chấp nhận: INTEGER, CHAR, ARRAY, hoặc identifier
iii. Gây lỗi 3-14:invalid type

h. ERR_INVALIDBASICTYPE

Test > error8.kpl

```
1   PROGRAM TestBasicType;
2   VAR
3       x : INTEGER;
4
5   FUNCTION Calculate(a: INTEGER): BOOLEAN;
6   BEGIN
7       Calculate := a
8   END;
9
10  BEGIN
11      x := Calculate(5);
12      CALL WriteI(x)
13  END.
```

Problems Output Debug Console **Terminal** Ports

```
C:\Users\vuvan\Downloads\PT_CPIncompleted>parser.exe test/error8.kpl
Parsing a Program ....
1-1:KW_PROGRAM
1-9:TK_IDENT(TestBasicType)
1-22:SB_SEMICOLON
Parsing a Block ....
Parsing Block 2 ....
Parsing Block 3 ....
2-1:KW_VAR
3-5:TK_IDENT(x)
3-7:SB_COLON
3-9:KW_INTEGER
3-16:SB_SEMICOLON
Parsing Block 4 ....
Parsing subroutines ....
Parsing a function ....
5-1:KW_FUNCTION
5-10:TK_IDENT(Calculate)
5-19:SB_LPAR
5-20:TK_IDENT(a)
5-21:SB_COLON
5-23:KW_INTEGER
5-30:SB_RPAR
5-31:SB_COLON
5-33:Invalid basic type!
```

   i. Dòng 5 Kiểu trả về BOOLEAN không phải basic type
   ii. Gây lỗi 5-33:Invalid basic type!

i. ERR_INVALIDPARAM

Test > error9.kpl

```
1   PROGRAM TestParam;
2   VAR
3       x : INTEGER;
4
5   PROCEDURE Process(123 : INTEGER);
6   BEGIN
7       CALL WriteI(123)
8   END;
9
10  BEGIN
11      CALL Process(x)
12  END.
```

Problems Output Debug Console Terminal Ports

```
C:\Users\vuvan\Downloads\PT_CPIncompleted>parser.exe test/error9.kpl
Parsing a Program ....
1-1:KW_PROGRAM
1-9:TK_IDENT(TestParam)
1-18:SB_SEMICOLON
Parsing a Block ....
Parsing Block 2 ....
Parsing Block 3 ....
2-1:KW_VAR
3-5:TK_IDENT(x)
3-7:SB_COLON
3-9:KW_INTEGER
3-16:SB_SEMICOLON
Parsing Block 4 ....
Parsing subroutines ....
Parsing a procedure ....
5-1:KW_PROCEDURE
5-11:TK_IDENT(Process)
5-18:SB_LPAR
5-19:Invalid parameter!
```

i. Dòng 5 Parameter 123 không phải identifier
ii. Gây lỗi 5-19:Invalid parameter!

j. ERR_INVALIDSTATEMENT

Test › error10.kpl

```
1   PROGRAM TestStatement;
2   VAR
3       x : INTEGER;
4   BEGIN
5       x := 5;
6       TYPE newType = INTEGER;
7       CALL WriteI(x)
8   END.
```

Problems Output Debug Console Terminal Ports

```
C:\Users\vuvan\Downloads\PT_CPIncompleted>parser.exe test/error10.kpl
Parsing a Program ....
1-1:KW_PROGRAM
1-9:TK_IDENT(TestStatement)
1-22:SB_SEMICOLON
Parsing a Block ....
Parsing Block 2 ....
Parsing Block 3 ....
2-1:KW_VAR
3-5:TK_IDENT(x)
3-7:SB_COLON
3-9:KW_INTEGER
3-16:SB_SEMICOLON
Parsing Block 4 ....
Parsing subroutines ....
Subroutines parsed ....
4-1:KW_BEGIN
Parsing an assign statement ....
5-5:TK_IDENT(x)
5-8:SB_ASSIGN
Parsing an expression
5-10:TK_NUMBER(5)
Expression parsed
Assign statement parsed ....
5-11:SB_SEMICOLON
6-5:Invalid statement!
```

k. ERR_INVALIDARGUMENTS

test > error11.kpl

```
1   PROGRAM TestArguments;
2   VAR
3       x : INTEGER;
4
5   FUNCTION Add(a: INTEGER; b: INTEGER): INTEGER;
6   BEGIN
7       Add := a + b
8   END;
9
```

Problems Output Debug Console Terminal Ports

```
5-29:KW_INTEGER
5-36:SB_RPAR
5-37:SB_COLON
5-39:KW_INTEGER
5-46:SB_SEMICOLON
Parsing a Block ....
Parsing Block 2 ....
Parsing Block 3 ....
Parsing Block 4 ....
Parsing subroutines ....
Subroutines parsed ....
6-1:KW_BEGIN
Parsing an assign statement ....
7-5:TK_IDENT(Add)
7-10:SB_ASSIGN
Parsing an expression
7-12:TK_IDENT(a)
7-14:SB_PLUS
7-16:TK_IDENT(b)
Expression parsed
Assign statement parsed ....
8-1:KW_END
Block parsed!
8-4:SB_SEMICOLON
Function parsed ....
Subroutines parsed ....
10-1:KW_BEGIN
Parsing an assign statement ....
11-5:TK_IDENT(x)
11-8:SB_ASSIGN
Parsing an expression
11-10:TK_IDENT(Add)
11-13:SB_LPAR
Parsing an expression
11-14:TK_NUMBER(5)
Expression parsed
11-15:SB_COMMA
11-17:Invalid arguments!
```

l. ERR_INVALIDCOMPARATOR

```
1   PROGRAM TestComparator;
2   VAR
3       x : INTEGER;
4       y : INTEGER;
5   BEGIN
6       x := 5;
7       y := 10;
8       IF x + y THEN
9           CALL WriteI(x)
```

Problems Output Debug Console **Terminal** Ports

```
Parsing Block 3 ....
2-1:KW_VAR
3-5:TK_IDENT(x)
3-7:SB_COLON
3-9:KW_INTEGER
3-16:SB_SEMICOLON
4-5:TK_IDENT(y)
4-7:SB_COLON
4-9:KW_INTEGER
4-16:SB_SEMICOLON
Parsing Block 4 ....
Parsing subroutines ....
Subroutines parsed ....
5-1:KW_BEGIN
Parsing an assign statement ....
6-5:TK_IDENT(x)
6-8:SB_ASSIGN
Parsing an expression
6-10:TK_NUMBER(5)
Expression parsed
Assign statement parsed ....
6-11:SB_SEMICOLON
Parsing an assign statement ....
7-5:TK_IDENT(y)
7-8:SB_ASSIGN
Parsing an expression
7-10:TK_NUMBER(10)
Expression parsed
Assign statement parsed ....
7-12:SB_SEMICOLON
Parsing an if statement ....
8-5:KW_IF
Parsing an expression
8-8:TK_IDENT(x)
8-10:SB_PLUS
8-12:TK_IDENT(y)
Expression parsed
8-14:Invalid comparator!
```

m. ERR_INVALIDEXPRESSION

Test > error13.kpl

```
1   PROGRAM TestExpression;
2   VAR
3       x : INTEGER;
4   BEGIN
5       x := WHILE + 5;
6       CALL WriteI(x)
7   END.
```

Problems Output Debug Console Terminal Ports

```
C:\Users\vuvan\Downloads\PT_CPIncompleted>parser.exe test/error13.kpl
Parsing a Program ....
1-1:KW_PROGRAM
1-9:TK_IDENT(TestExpression)
1-23:SB_SEMICOLON
Parsing a Block ....
Parsing Block 2 ....
Parsing Block 3 ....
2-1:KW_VAR
3-5:TK_IDENT(x)
3-7:SB_COLON
3-9:KW_INTEGER
3-16:SB_SEMICOLON
Parsing Block 4 ....
Parsing subroutines ....
Subroutines parsed ....
4-1:KW_BEGIN
Parsing an assign statement ....
5-5:TK_IDENT(x)
5-8:SB_ASSIGN
Parsing an expression
5-10:Invalid expression!
```

n. ERR_INVALIDTERM

Test > error14.kpl

```
1   PROGRAM TestTerm;
2   VAR
3       x : INTEGER;
4   BEGIN
5       x := 10 * FOR;
6       CALL WriteI(x)
7   END.
```

Problems Output Debug Console Terminal Ports

```
C:\Users\vuvan\Downloads\PT_CPIncompleted>parser.exe test/error14.kpl
Parsing a Program ....
1-1:KW_PROGRAM
1-9:TK_IDENT(TestTerm)
1-17:SB_SEMICOLON
Parsing a Block ....
Parsing Block 2 ....
Parsing Block 3 ....
2-1:KW_VAR
3-5:TK_IDENT(x)
3-7:SB_COLON
3-9:KW_INTEGER
3-16:SB_SEMICOLON
Parsing Block 4 ....
Parsing subroutines ....
Subroutines parsed ....
4-1:KW_BEGIN
Parsing an assign statement ....
5-5:TK_IDENT(x)
5-8:SB_ASSIGN
Parsing an expression
5-10:TK_NUMBER(10)
5-13:SB_TIMES
5-15:Invalid term!
```

o. ERR_INVALIDFACTOR

Test › error15.kpl

```
1   PROGRAM TestFactor;
2   VAR x : INTEGER;
3   BEGIN
4       x := - WHILE + 5 ;
5   END.
```

Problems Output Debug Console **Terminal** Ports

```
C:\Users\vuvan\Downloads\PT_CPIncompleted>parser.exe test/error15.kpl
Parsing a Program ....
1-1:KW_PROGRAM
1-9:TK_IDENT(TestFactor)
1-19:SB_SEMICOLON
Parsing a Block ....
Parsing Block 2 ....
Parsing Block 3 ....
2-1:KW_VAR
2-5:TK_IDENT(x)
2-7:SB_COLON
2-9:KW_INTEGER
2-16:SB_SEMICOLON
Parsing Block 4 ....
Parsing subroutines ....
Subroutines parsed ....
3-1:KW_BEGIN
Parsing an assign statement ....
4-5:TK_IDENT(x)
4-8:SB_ASSIGN
Parsing an expression
4-10:SB_MINUS
4-12:Invalid factor!
```