

# ***Experiment in Compiler Construction*** ***PHÂN TÍCH CÚ PHÁP***

**Nguyen Huu Duc**

Department of information systems  
Faculty of information technology  
Hanoi university of technology

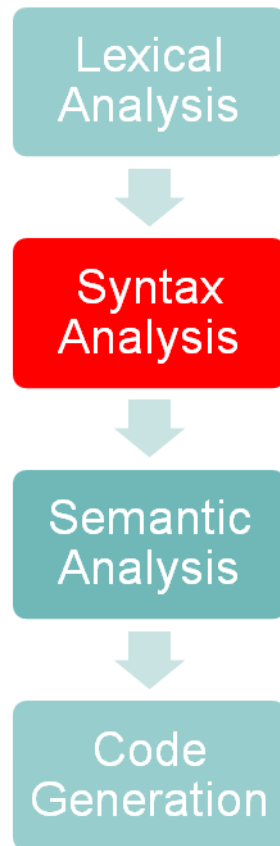
# Nội dung

---

- Tổng quan về phân tích cú pháp
- Văn phạm KPL
- Xây dựng bộ phân tích cú pháp (parser)

# Nhiệm vụ của một parser

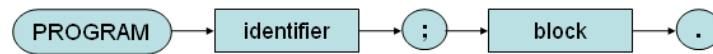
---



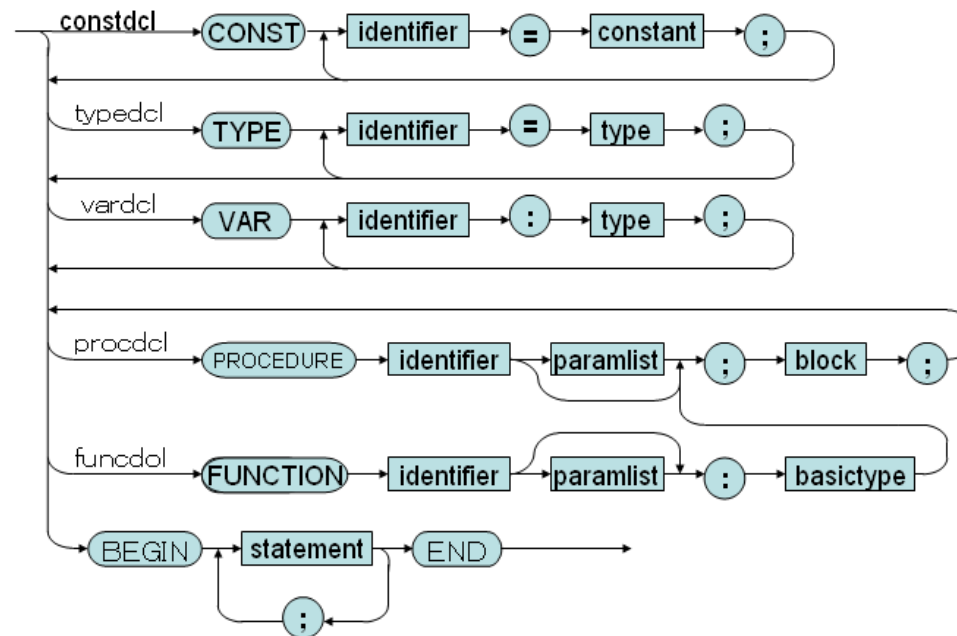
- Kiểm tra cấu trúc ngữ pháp của một chương trình
- Kích hoạt các bộ phân tích ngữ nghĩa và sinh mã

# Sơ đồ cú pháp của ngôn ngữ KPL

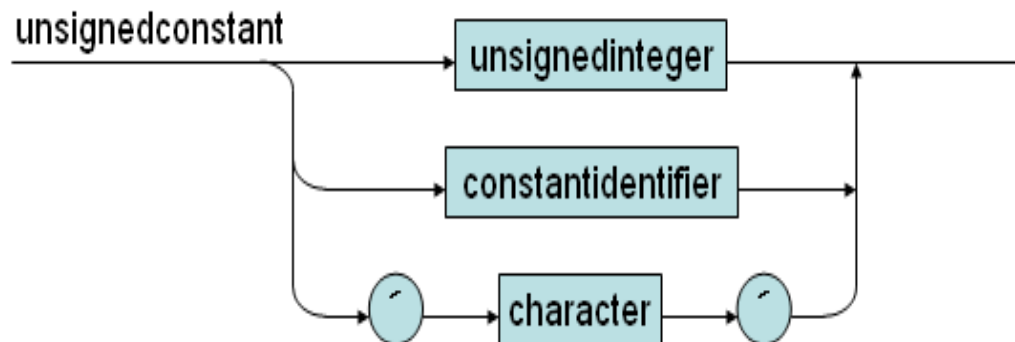
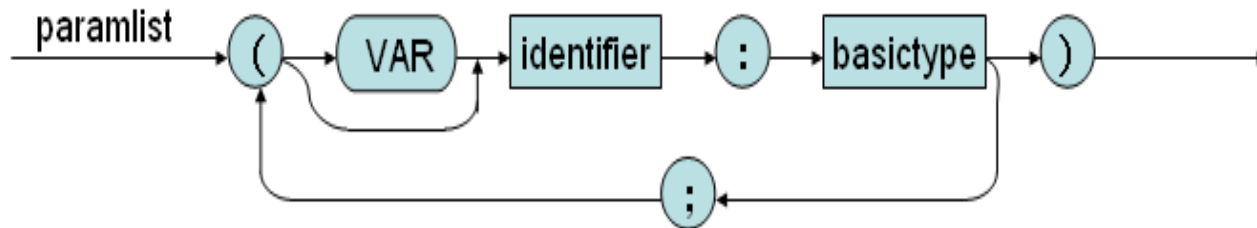
program



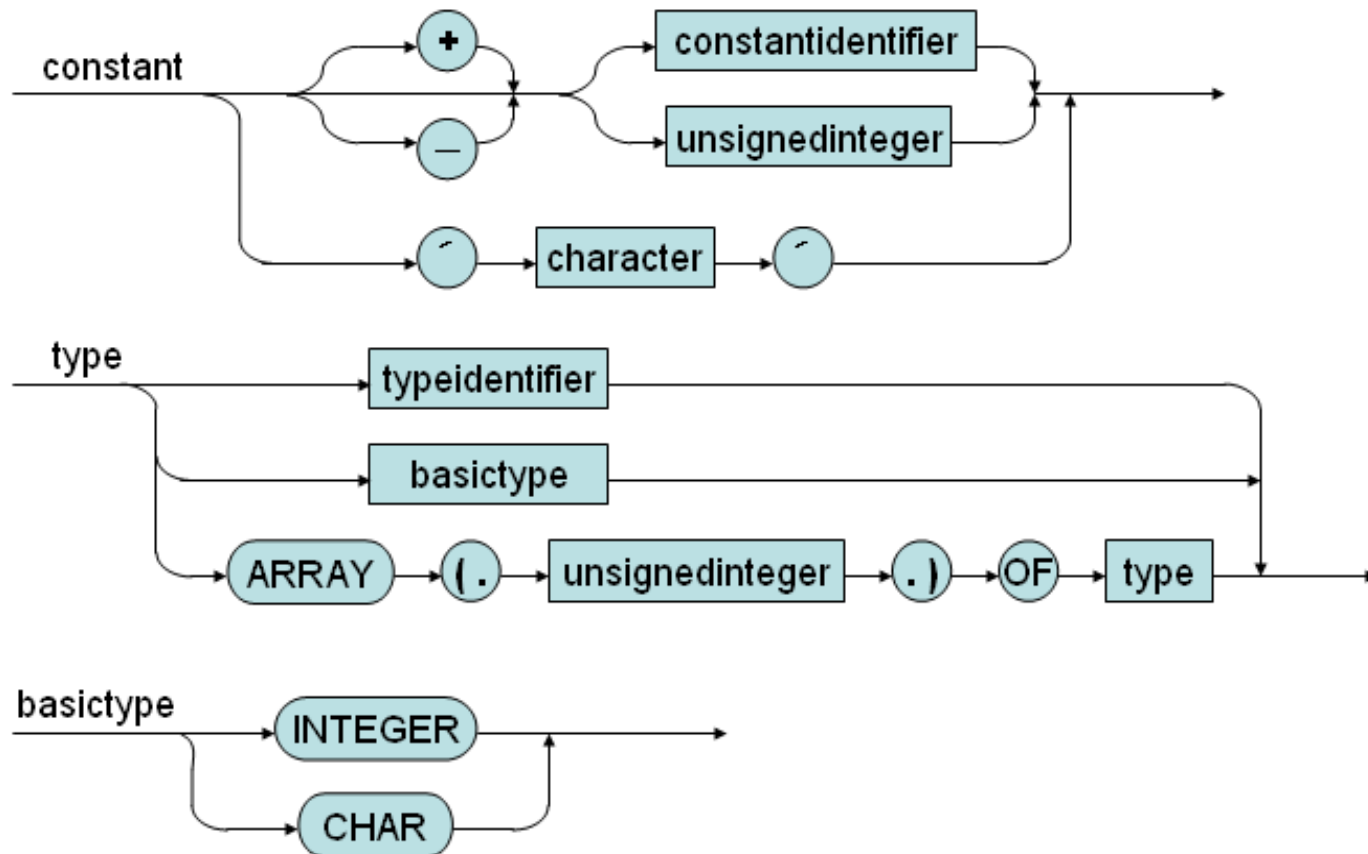
block



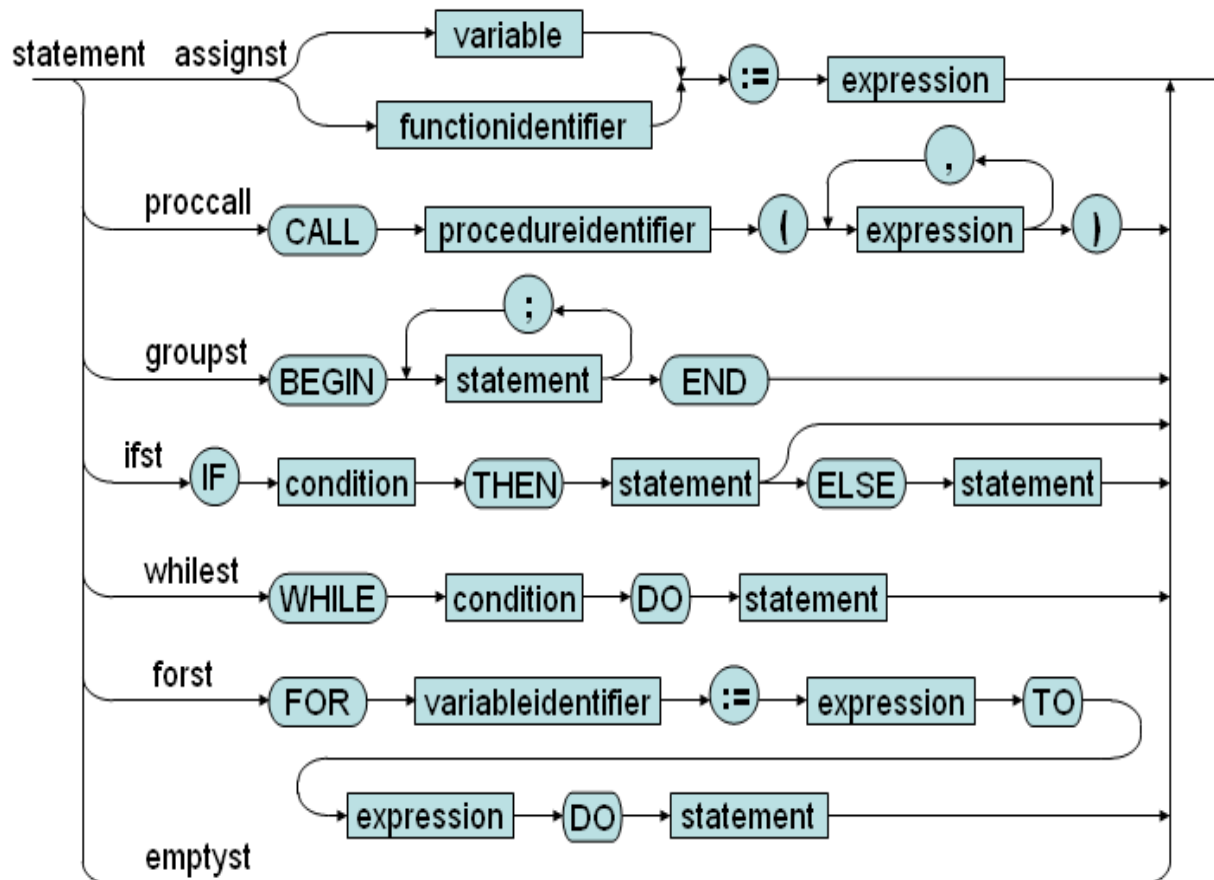
# Sơ đồ cú pháp của ngôn ngữ KPL



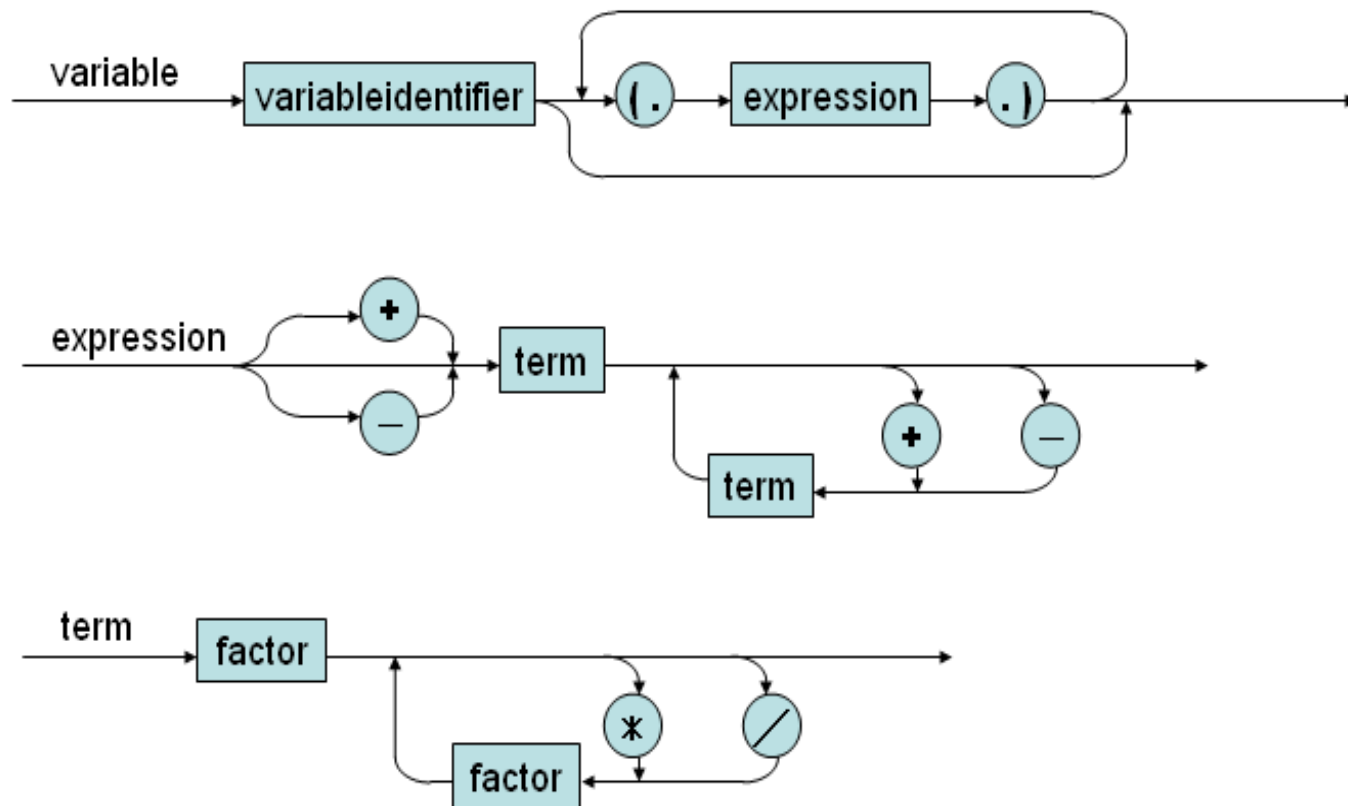
# Sơ đồ cú pháp của ngôn ngữ KPL



# Sơ đồ cú pháp của ngôn ngữ KPL

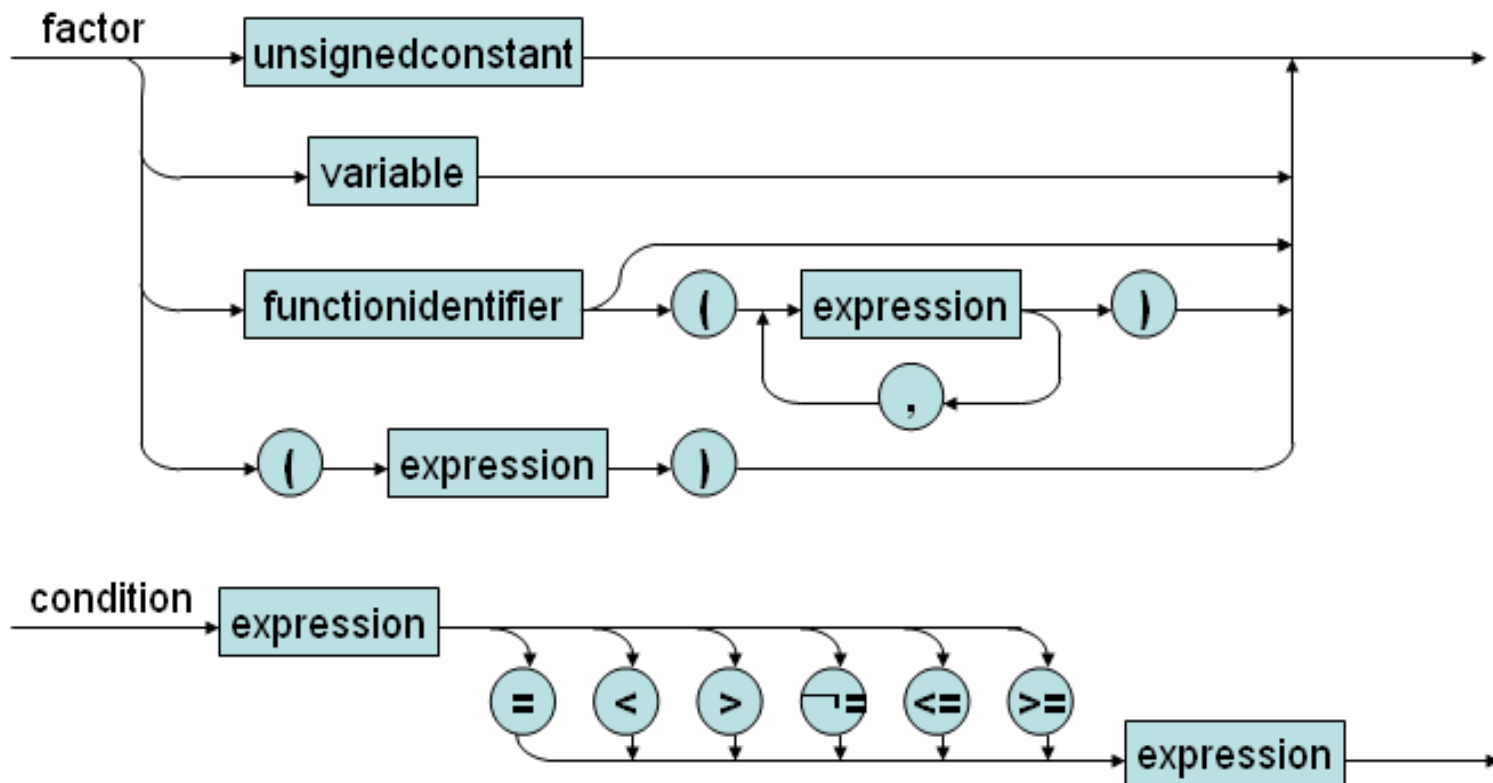


# Sơ đồ cú pháp của ngôn ngữ KPL

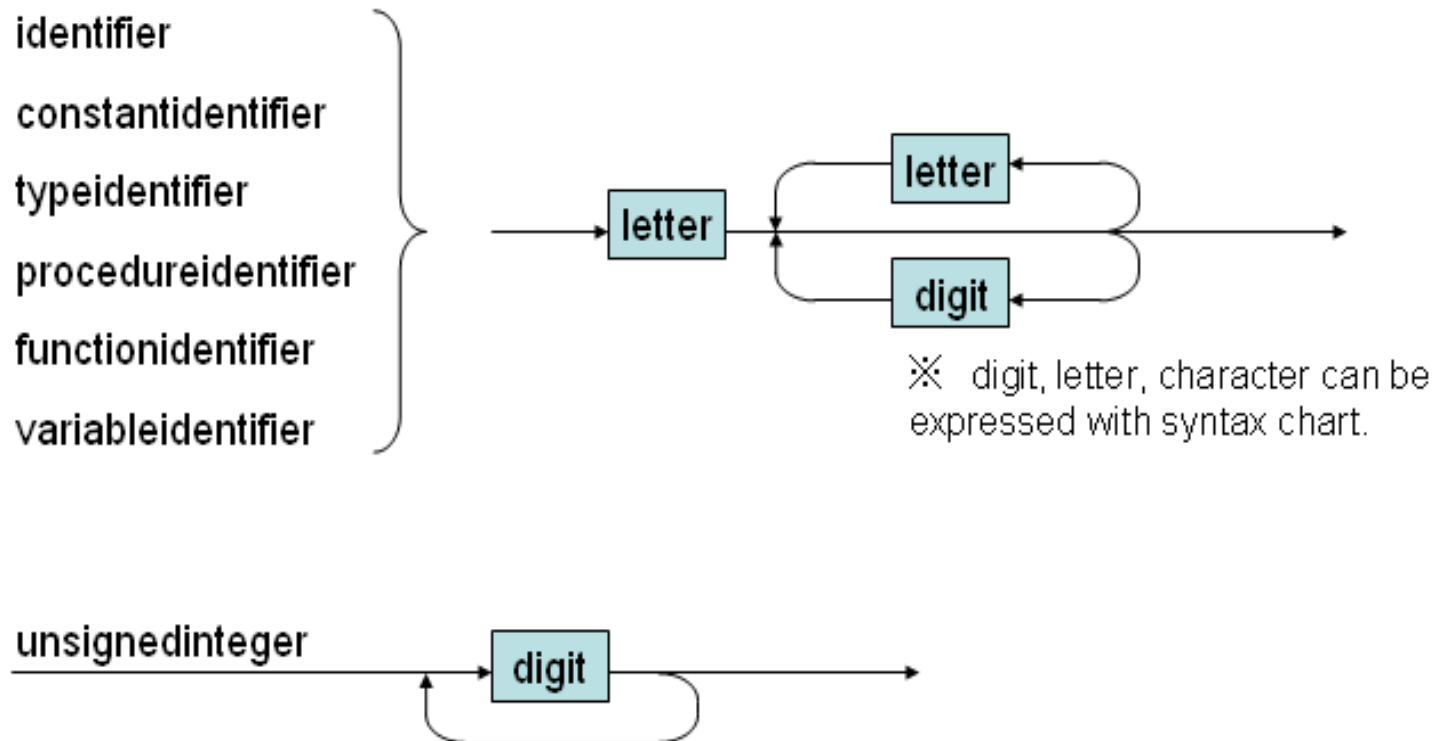




# Sơ đồ cú pháp của ngôn ngữ KPL



# Sơ đồ cú pháp của ngôn ngữ KPL



# Văn phạm BNF

---

- Thực hiện loại bỏ đệ quy trái
- Thực hiện nhân tử trái

# Văn phạm BNF

---

01) Prog ::= KW\_PROGRAM Ident SB\_SEMICOLON Block SB\_PERIOD

02) Block ::= KW\_CONST ConstDecl ConstDecls Block2

03) Block ::= Block2

04) Block2 ::= KW\_TYPE TypeDecl TypeDecls Block3

05) Block2 ::= Block3

06) Block3 ::= KW\_VAR VarDecl VarDecls Block4

07) Block3 ::= Block4

08) Block4 ::= SubDecls Block5

09) Block5 ::= KW\_BEGIN Statements KW\_END

# Văn phạm BNF

---

- 10) `ConstDecls ::= ConstDecl ConstDecls`
- 11) `ConstDecls ::=  $\epsilon$`  epsilon: không có gì cả
- 12) `ConstDecl ::= Ident SB_EQUAL Constant SB_SEMICOLON`
- 13) `TypeDecls ::= TypeDecl TypeDecls`
- 14) `TypeDecls ::=  $\epsilon$`
- 15) `TypeDecl ::= Ident SB_EQUAL Type SB_SEMICOLON`
- 16) `VarDecls ::= VarDecl VarDecls`
- 17) `VarDecls ::=  $\epsilon$`
- 18) `VarDecl ::= Ident SB_COLON Type SB_SEMICOLON`
- 19) `SubDecls ::= FunDecl SubDecls`
- 20) `SubDecls ::= ProcDecl SubDecls`
- 21) `SubDecls ::=  $\epsilon$`

# Văn phạm BNF

---

- 22) FunDecl ::= KW\_FUNCTION Ident Params SB\_COLON BasicType SB\_SEMICOLON  
Block SB\_SEMICOLON
  
- 23) ProcDecl ::= KW\_PROCEDURE Ident Params SB\_SEMICOLON Block SB\_SEMICOLON
  
- 24) Params ::= SB\_LPAR Param Params2 SB\_RPAR
- 25) Params ::=  $\epsilon$
  
- 26) Params2 ::= SB\_SEMICOLON Param Params2
- 27) Params2 ::=  $\epsilon$
  
- 28) Param ::= Ident SB\_COLON BasicType
- 29) Param ::= KW\_VAR Ident SB\_COLON BasicType

# Văn phạm BNF

---

- 30) `Type ::= KW_INTEGER`
- 31) `Type ::= KW_CHAR`
- 32) `Type ::= TypeIdent`
- 33) `Type ::= KW_ARRAY SB_LSEL Number SB_RSEL KW_OF Type`
  
- 34) `BasicType ::= KW_INTEGER`
- 35) `BasicType ::= KW_CHAR`
  
- 36) `UnsignedConstant ::= Number`
- 37) `UnsignedConstant ::= ConstIdent`
- 38) `UnsignedConstant ::= ConstChar`
  
- 40) `Constant ::= SB_PLUS Constant2`
- 41) `Constant ::= SB_MINUS Constant2`
- 42) `Constant ::= Constant2`
- 43) `Constant ::= ConstChar`
  
- 44) `Constant2 ::= ConstIdent`
- 45) `Constant2 ::= Number`

# Văn phạm BNF

---

46) `Statements ::= Statement Statements2`

47) `Statements2 ::= KW_SEMICOLON Statement Statement2`

48) `Statements2 ::=  $\epsilon$`

49) `Statement ::= AssignSt`

50) `Statement ::= CallSt`

51) `Statement ::= GroupSt`

52) `Statement ::= IfSt`

53) `Statement ::= WhileSt`

54) `Statement ::= ForSt`

55) `Statement ::=  $\epsilon$`



# Văn phạm BNF

---

- 56) `AssignSt ::= Variable SB_ASSIGN Expression`
- 57) `AssignSt ::= FunctionIdent SB_ASSIGN Expression`
  
- 58) `CallSt ::= KW_CALL ProcedureIdent Arguments`
  
- 59) `GroupSt ::= KW_BEGIN Statements KW_END`
  
- 60) `IfSt ::= KW_IF Condition KW_THEN Statement ElseSt`
  
- 61) `ElseSt ::= KW_ELSE statement`
- 62) `ElseSt ::= ε`
  
- 63) `WhileSt ::= KW_WHILE Condition KW_DO Statement`
- 64) `ForSt ::= KW_FOR VariableIdent SB_ASSIGN Expression KW_TO  
Expression KW_DO Statement`

# Văn phạm BNF

---

65) `Arguments ::= SB_LPAR Expression Arguments2 SB_RLAR`

66) `Arguments ::=  $\epsilon$`

67) `Arguments2 ::= SB_COMMA Expression Arguments2`

68) `Arguments2 ::=  $\epsilon$`

68) `Condition ::= Expression Condition2`

69) `Condition2 ::= SB_EQ Expression`

70) `Condition2 ::= SB_NEQ Expression`

71) `Condition2 ::= SB_LE Expression`

72) `Condition2 ::= SB_LT Expression`

73) `Condition2 ::= SB_GE Expression`

74) `Condition2 ::= SB_GT Expression`

# Văn phạm BNF

---

- 75) `Expression ::= SB_PLUS Expression2`
- 76) `Expression ::= SB_MINUS Expression2`
- 77) `Expression ::= Expression2`
  
- 78) `Expression2 ::= Term Expression3`
  
- 79) `Expression3 ::= SB_PLUS Term Expression3`
- 80) `Expression3 ::= SB_MINUS Term Expression3`
- 81) `Expression3 ::=  $\epsilon$`
  
- 82) `Term ::= Factor Term2`
  
- 83) `Term2 ::= SB_TIMES Factor Term2`
- 84) `Term2 ::= SB_SLASH Factor Term2`
- 85) `Term2 ::=  $\epsilon$`

# Văn phạm BNF

---

- 86) `Factor ::= UnsignedConstant`
- 87) `Factor ::= Variable`
- 88) `Factor ::= FunctionApptication`
- 89) `Factor ::= SB_LPAR Expression SB_RPAR`
  
- 90) `Variable ::= VariableIdent Indexes`
- 91) `FunctionApplication ::= FunctionIdent Arguments`
  
- 92) `Indexes ::= SB_LSEL Expression SB_RSEL Indexes`
- 93) `Indexes ::=  $\epsilon$`

## Câu hỏi?

---

- Hãy tính các tập FIRST và FOLLOW cho mỗi ký hiệu không kết thúc
- Hãy xây dựng bảng duyệt

## Xây dựng parser

---

- Về cơ bản KPL là một ngôn ngữ LL(1)
- Thiết kế một parser đệ quy trên dưới
  - Token ***lookAhead***
  - Duyệt ký hiệu kết thúc
  - Duyệt ký hiệu không kết thúc

# Xây dựng parser – Cấu trúc

---

STT	Tên tệp	Nội dung
1	Makefile	Project
2	scanner.c, scanner.h	Đọc từng token
3	reader.h, reader.c	Đọc mã nguồn
4	charcode.h, charcode.c	Phân loại ký tự
5	token.h, token.c	Phân loại và nhận dạng token, từ khóa
6	error.h, error.c	Thông báo lỗi
7	parser.c, parser.h	Duyệt các cấu trúc chương trình
8	main.c	Chương trình chính

# lookAhead

---

- Xem trước nội dung một token

```
Token *currentToken;    // Token vừa đọc  
Token *lookAhead;      // Token xem trước
```

```
void scan(void) {  
    Token* tmp = currentToken;  
    currentToken = lookAhead;  
    lookAhead = getValidToken();  
    free(tmp);  
}
```



# Duyệt ký hiệu kết thúc

---

```
void eat(TokenType tokenType) {  
    if (lookAhead->tokenType == tokenType) {  
        printToken(lookAhead);  
        scan();  
    } else missingToken(tokenType, lookAhead->lineNo, lookAhead->colNo);  
}
```

# Duyệt ký hiệu không kết thúc

---

```
void compileProgram(void) {  
    assert("Parsing a Program ....");  
    eat(KW_PROGRAM);  
    eat(TK_IDENT);  
    eat(SB_SEMICOLON);  
    compileBlock();  
    eat(SB_PERIOD);  
    assert("Program parsed!");  
}
```

# Kích hoạt parser

---

```
int compile(char *fileName) {  
    if (openInputStream(fileName) == IO_ERROR)  
        return IO_ERROR;  
  
    currentToken = NULL;  
    lookAhead = getValidToken();  
  
    compileProgram();  
  
    free(currentToken);  
    free(lookAhead);  
    closeInputStream();  
    return IO_SUCCESS;  
}
```

# Ví dụ - duyệt statement

---

`FIRST(Statement) = {TK_IDENT, KW_CALL, KW_BEGIN, KW_IF, KW_WHILE,  
KW_FOR,  $\epsilon$ }`

`FOLLOW(Statement) = {SB_SEMICOLON, KW_END, KW_ELSE}`

`/* Predict parse table for Expression */`

Input	Production
-----	
TK_IDENT	49) Statement ::= AssignSt
KW_CALL	50) Statement ::= CallSt
KW_BEGIN	51) Statement ::= GroupSt
KW_IF	52) Statement ::= IfSt
KW_WHILE	53) Statement ::= WhileSt
KW_FOR	54) Statement ::= ForSt
-----	
SB_SEMICOLON	55) $\epsilon$
KW_END	55) $\epsilon$
KW_ELSE	55) $\epsilon$
-----	
Others	Error

# Ví dụ - duyệt statement

---

```
void compileStatement(void) {
    switch (lookAhead->tokenType)
    {
        case TK_IDENT:
            compileAssignSt();
            break;
        case KW_CALL:
            compileCallSt();
            break;
        case KW_BEGIN:
            compileGroupSt();
            break;
        case KW_IF:
            compileIfSt();
            break;
        case KW_WHILE:
            compileWhileSt();
            break;
        case KW_FOR:
            compileForSt();
            break;
            // check FOLLOW tokens
        case SB_SEMICOLON:
        case KW_END:
        case KW_ELSE:
            break;
            // Error occurs
        default:
            error(ERR_INVALIDSTATEMENT,
                lookAhead->lineNo, lookAhead->colNo);
            break;
    }
}
```

# Bài tập 1

---

- Dịch chương trình với
  - Khai báo hằng
  - Khai báo kiểu
  - Khai báo biến
  - Thân hàm rỗng

## Bài tập 2

---

- Dịch chương trình với
  - Khai báo hằng
  - Khai báo kiểu
  - Khai báo biến
  - Các lệnh

## Bài tập 3

---

- Dịch chương trình với đầy đủ sơ đồ cú pháp