



ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

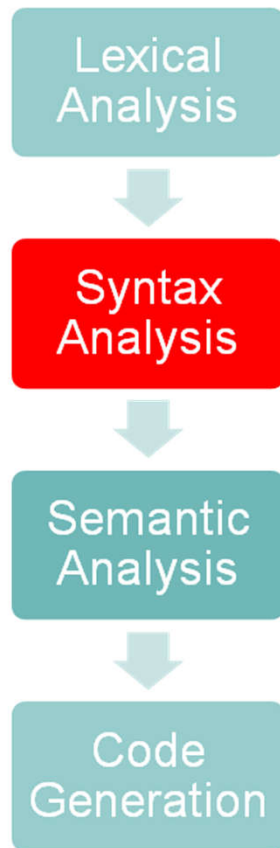
Bài thực hành 2

PHÂN TÍCH CÚ PHÁP

Nội dung

- Tổng quan về phân tích cú pháp
- Cú pháp KPL qua văn phạm và sơ đồ cú pháp
- Xây dựng bộ phân tích cú pháp (parser) cho ngôn ngữ KPL

Nhiệm vụ của bộ phân tích cú pháp



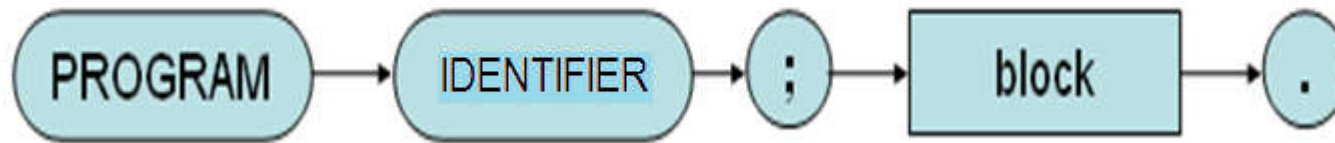
- Kiểm tra cấu trúc ngữ pháp của một chương trình
- Nếu chương trình đúng cú pháp, xây dựng được cây phân tích cú pháp (biểu diễn?)
- Các xử lý của bộ phân tích ngữ nghĩa và sinh mã là dựa trên bộ phân tích cú pháp

Biểu diễn cú pháp

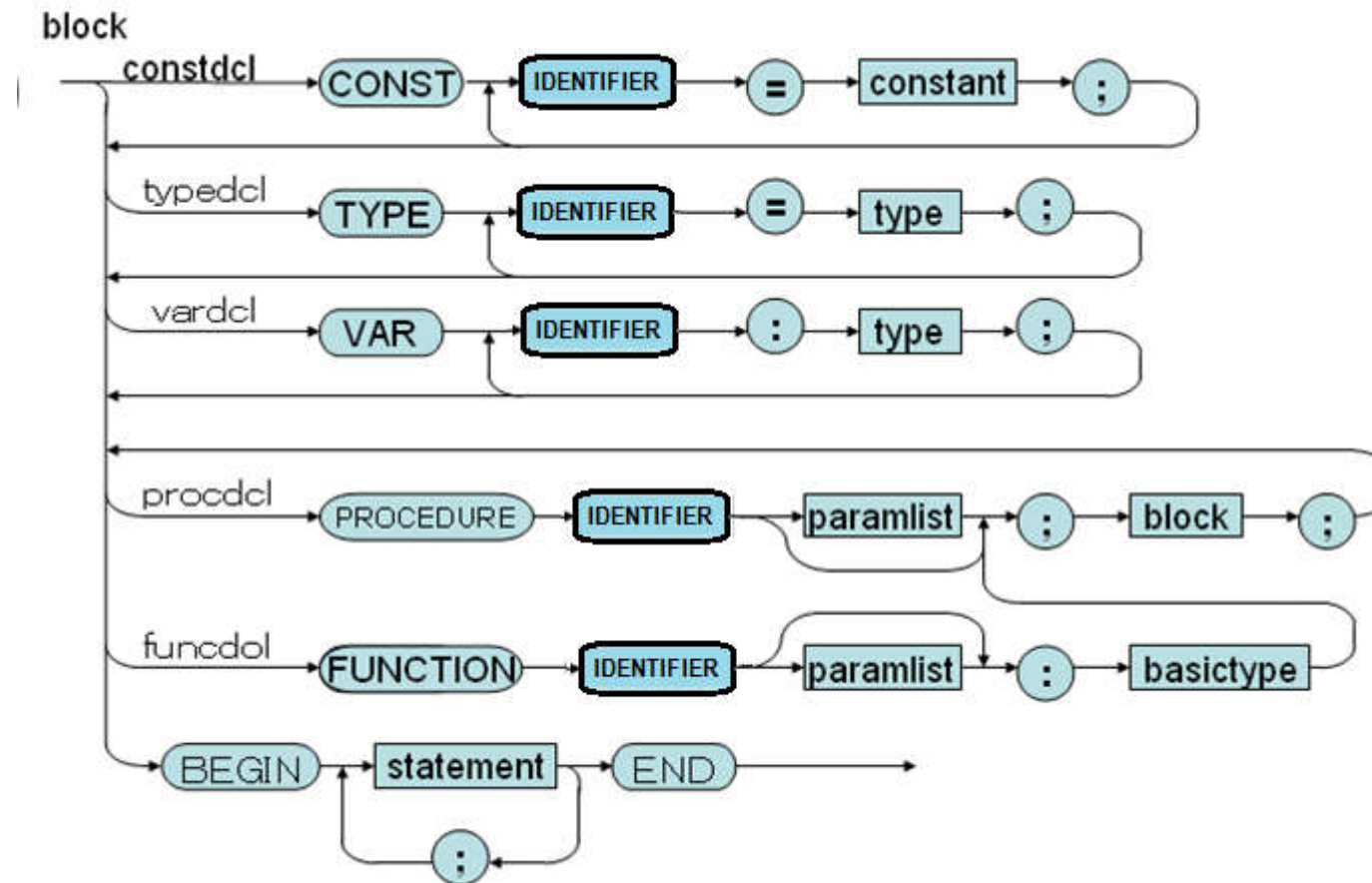
- Dạng chuẩn BNF
- Dạng trực quan: sơ đồ cú pháp
- Project xây dựng dựa trên BNF
- Cần tra cứu quy tắc cú pháp, có thể tham khảo sơ đồ cú pháp

Sơ đồ cú pháp của KPL (Tổng thể CT)

program

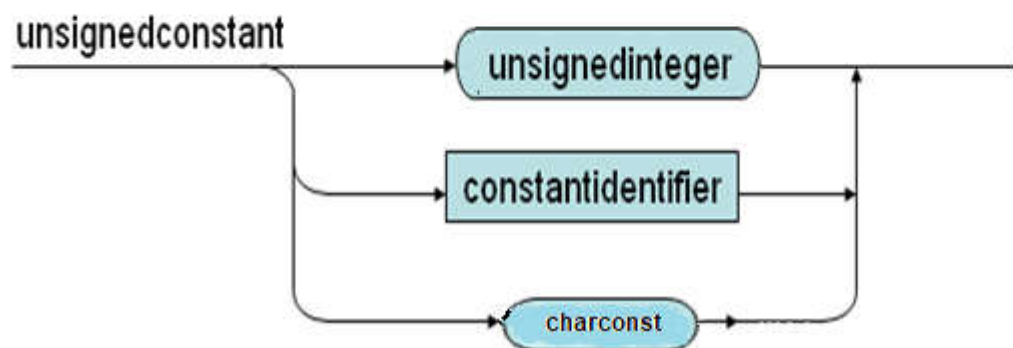
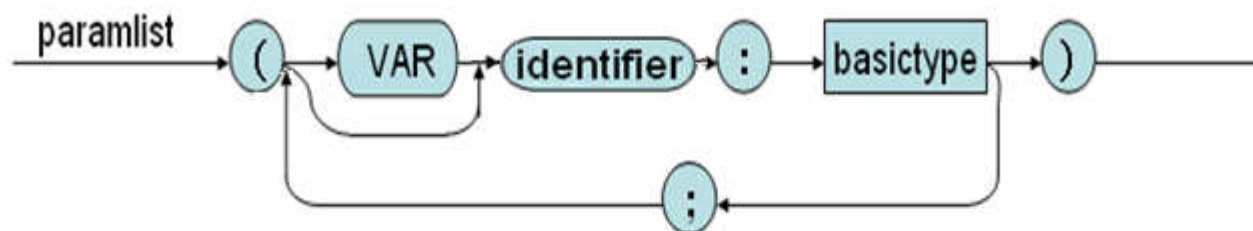


Sơ đồ cú pháp của KPL (Khối)

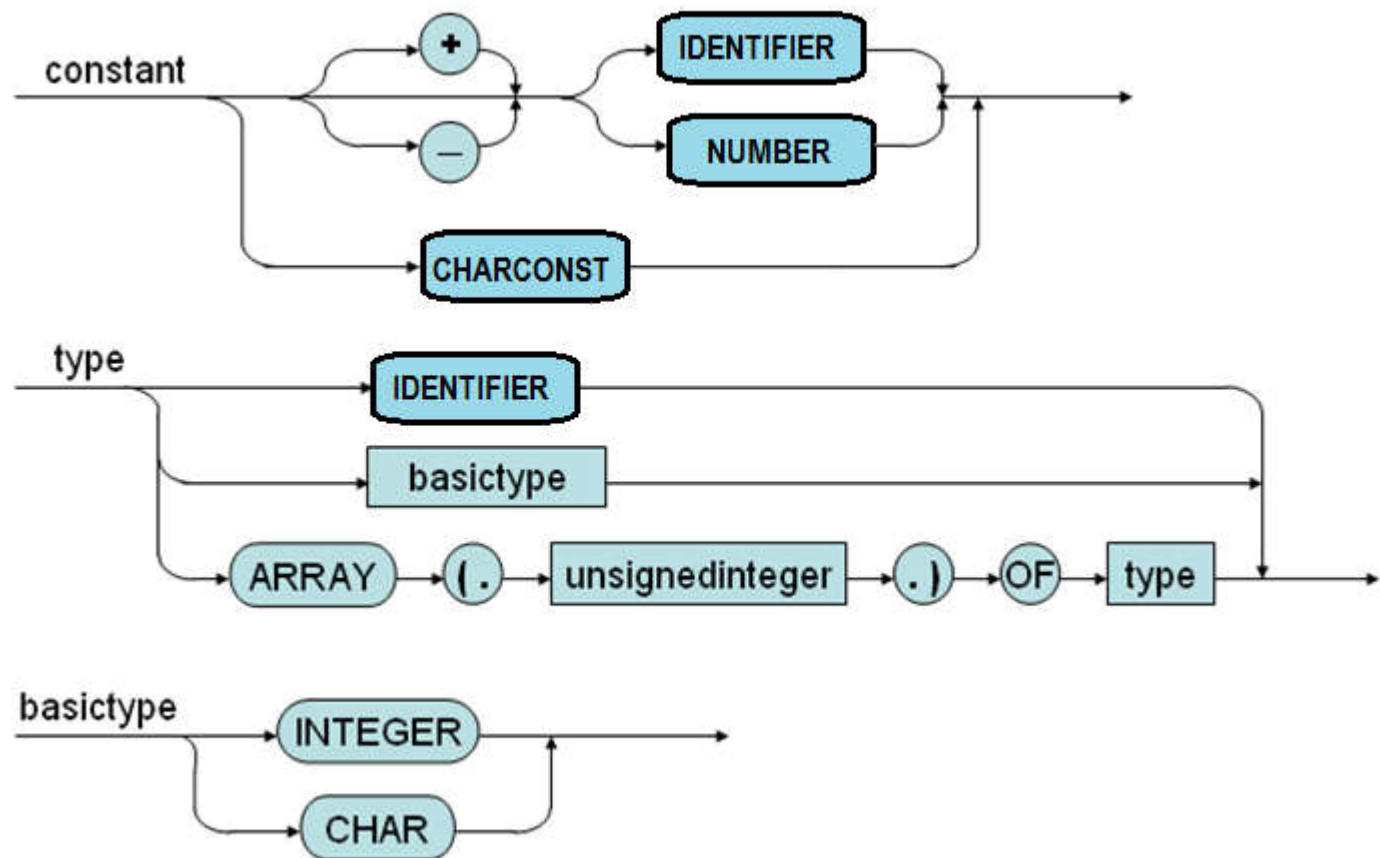


Sơ đồ cú pháp của KPL (tham số, hằng không dấu)

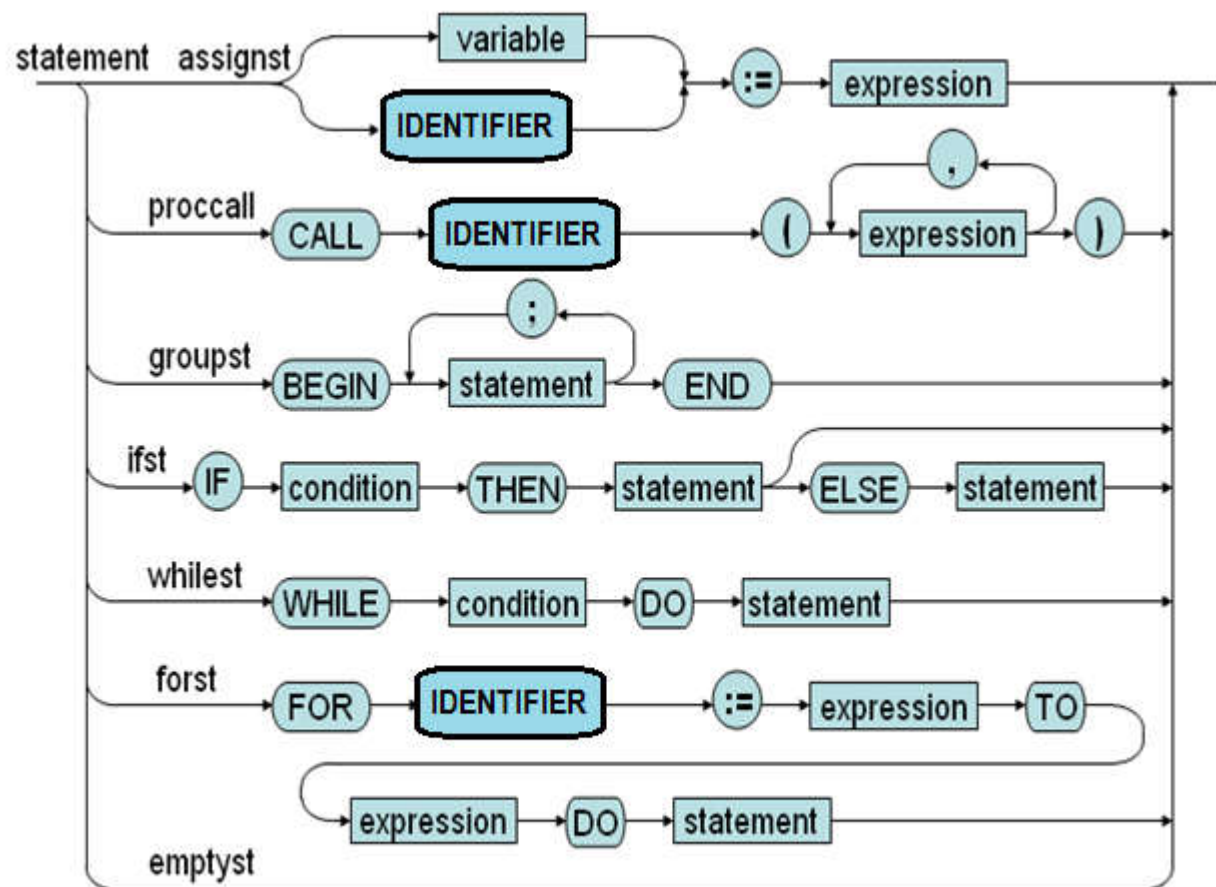
Sơ đồ cú pháp của KPL(tham số, hằng không dấu)



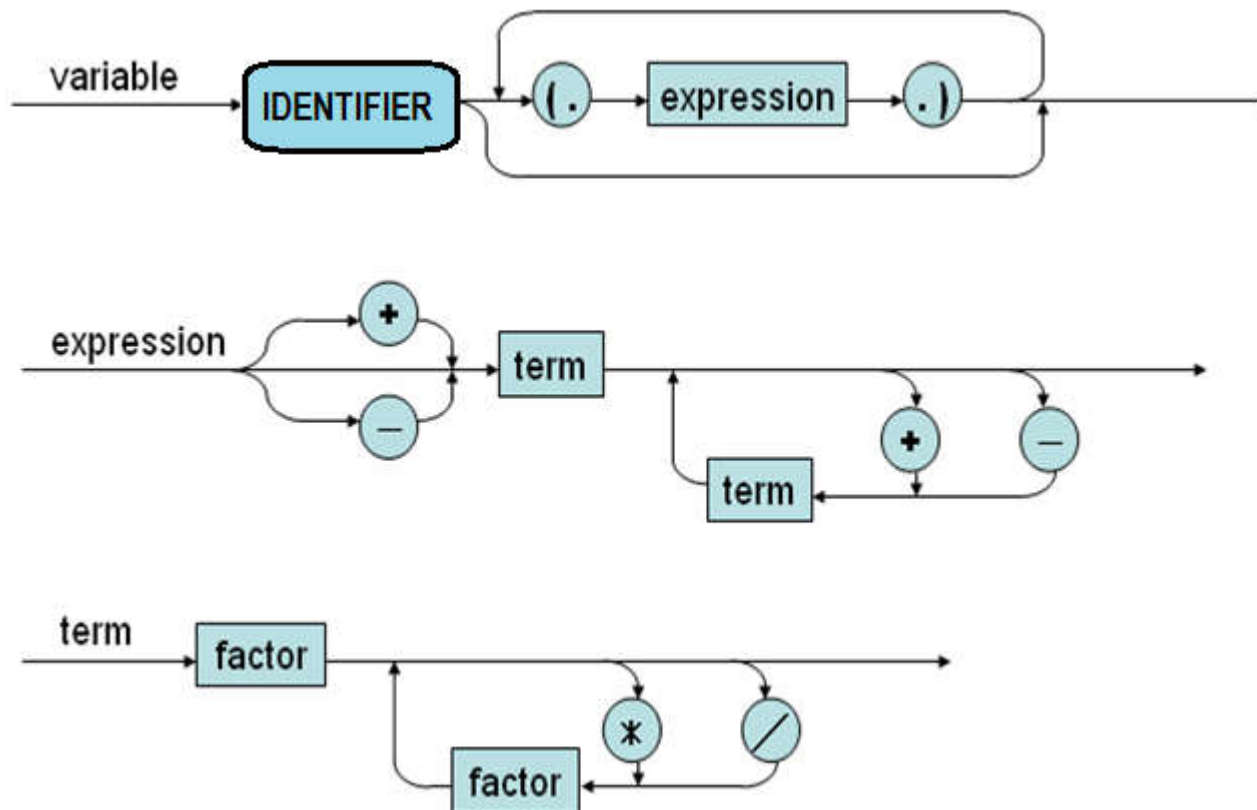
Sơ đồ cú pháp của KPL (Khái báo)



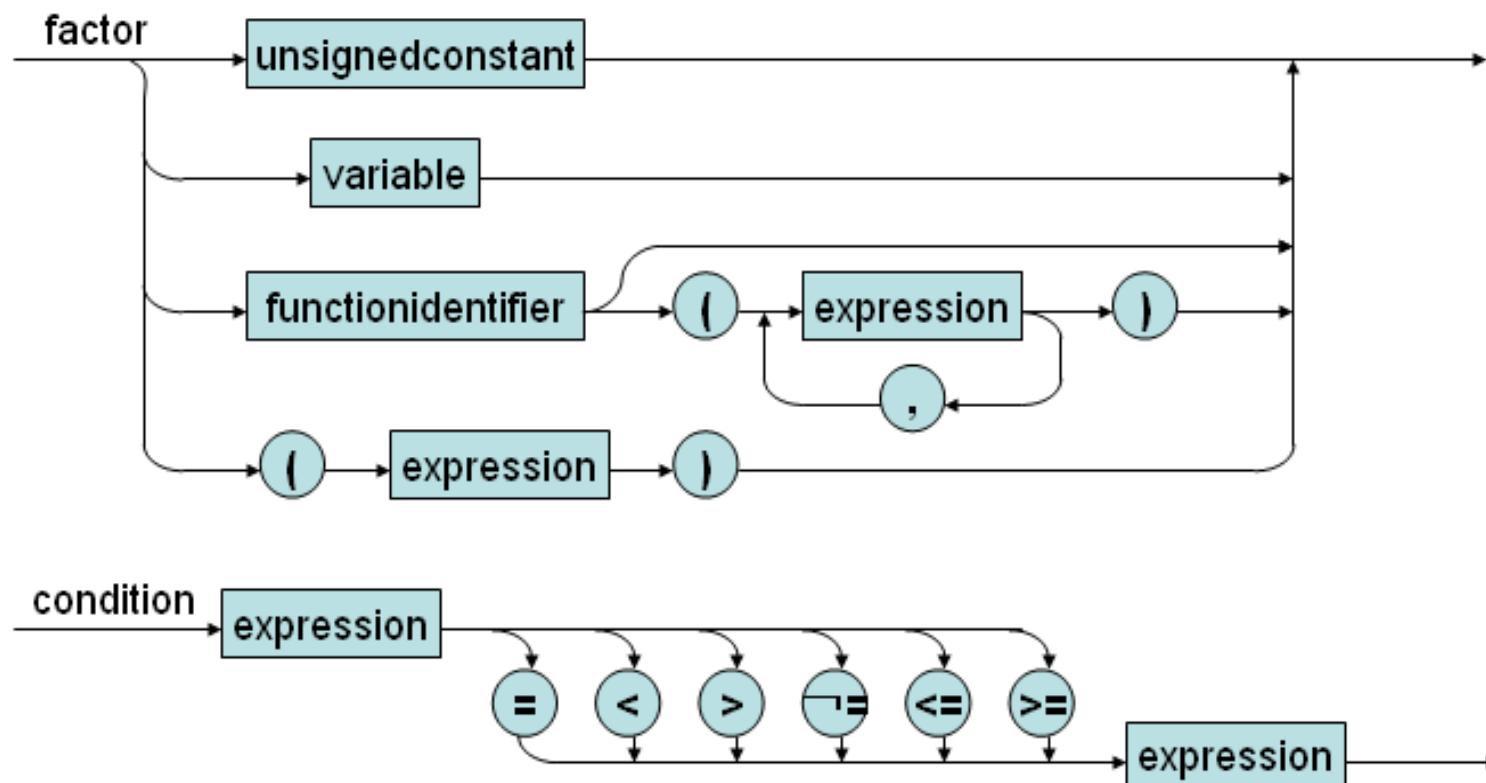
Sơ đồ cú pháp của KPL (lệnh)



Sơ đồ cú pháp của KPL (biểu thức)



Sơ đồ cú pháp của KPL (thừa số, điều kiện)



Văn phạm BNF

- Đã khử đệ quy trái
- Thực hiện nhân tử trái

60) $\langle \text{IfSt} \rangle ::= \text{KW_IF } \langle \text{Condition} \rangle$
 $\text{KW_THEN } \langle \text{Statement} \rangle \langle \text{ElseSt} \rangle$

61) $\langle \text{ElseSt} \rangle ::= \text{KW_ELSE } \langle \text{Statement} \rangle$

62) $\langle \text{ElseSt} \rangle ::= \varepsilon$

Văn phạm KPL viết bằng BNF

- 01) <Prog> ::= KW_PROGRAM TK_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>|<Block5>
- 09) <Block5> ::= KW_BEGIN <Statements> KW_END
- 10) <ConstDecls> ::= <ConstDecl> <ConstDecls>
- 11) <ConstDecls> ::= ε
- 12) <ConstDecl> ::= TK_IDENT SB_EQUAL <Constant> SB_SEMICOLON
- 13) <TypeDecls> ::= <TypeDecl> <TypeDecls>
- 14) <TypeDecls> ::= ε
- 15) <TypeDecl> ::= TK_IDENT SB_EQUAL <Type> SB_SEMICOLON
- 16) <VarDecls> ::= <VarDecl> <VarDecls>
- 17) <VarDecls> ::= ε
- 18) <VarDecl> ::= TK_IDENT SB_COLON <Type> SB_SEMICOLON

Văn phạm KPL viết bằng BNF

- 19) `<SubDecls> ::= <FunDecl> <SubDecls>`
- 20) `<SubDecls> ::= <ProcDecl> <SubDecls>`
- 21) `<SubDecls> ::= ε`

- 22) `<FunDecl> ::= KW_FUNCTION TK_IDENT <Params> SB_COLON <BasicType>`
 `SB_SEMICOLON`
 `<Block> SB_SEMICOLON`

- 23) `<ProcDecl> ::= KW_PROCEDURE TK_IDENT <Params> SB_SEMICOLON <Block>`
 `SB_SEMICOLON`

- 24) `<Params> ::= SB_LPAR <Param> <Params2> SB_RPAR`
- 25) `<Params> ::= ε`

- 26) `<Params2> ::= SB_SEMICOLON <Param> <Params2>`
- 27) `<Params2> ::= ε`

- 28) `<Param> ::= TK_IDENT SB_COLON <BasicType>`
- 29) `<Param> ::= KW_VAR TK_IDENT SB_COLON <BasicType>`

- 30) `<Type> ::= KW_INTEGER`
- 31) `<Type> ::= KW_CHAR`
- 32) `<Type> ::= TK_IDENT`
- 33) `<Type> ::= KW_ARRAY SB_LSEL TK_NUMBER SB_RSEL KW_OF <Type>`

Văn phạm KPL viết bằng BNF

34) $\langle \text{BasicType} \rangle ::= \text{KW_INTEGER}$

35) $\langle \text{BasicType} \rangle ::= \text{KW_CHAR}$

36) $\langle \text{UnsignedConstant} \rangle ::= \text{TK_NUMBER}$

37) $\langle \text{UnsignedConstant} \rangle ::= \text{TK_IDENT}$

38) $\langle \text{UnsignedConstant} \rangle ::= \text{TK_CHAR}$

40) $\langle \text{Constant} \rangle ::= \text{SB_PLUS } \langle \text{Constant2} \rangle$

41) $\langle \text{Constant} \rangle ::= \text{SB_MINUS } \langle \text{Constant2} \rangle$

42) $\langle \text{Constant} \rangle ::= \langle \text{Constant2} \rangle$

43) $\langle \text{Constant} \rangle ::= \text{TK_CHAR}$

44) $\langle \text{Constant2} \rangle ::= \text{TK_IDENT}$

45) $\langle \text{Constant2} \rangle ::= \text{TK_NUMBER}$

46) $\langle \text{Statements} \rangle ::= \langle \text{Statement} \rangle \langle \text{Statements2} \rangle$

47) $\langle \text{Statements2} \rangle ::= \text{SB_SEMICOLON } \langle \text{Statement} \rangle \langle \text{Statements2} \rangle$

48) $\langle \text{Statements2} \rangle ::= \varepsilon$

Văn phạm KPL viết bằng BNF

- 49) `<Statement> ::= <AssignSt>`
- 50) `<Statement> ::= <CallSt>`
- 51) `<Statement> ::= <GroupSt>`
- 52) `<Statement> ::= <IfSt>`
- 53) `<Statement> ::= <WhileSt>`
- 54) `<Statement> ::= <ForSt>`
- 55) `<Statement> ::= ε`
- 56) `<AssignSt> ::= <Variable> SB_ASSIGN <Expression>`
- 57) `<AssignSt> ::= TK_IDENT SB_ASSIGN <Expression>`

- 58) `<CallSt> ::= KW_CALL TK_IDENT <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 TK_IDENT SB_ASSIGN <Expression> KW_TO
 <Expression> KW_DO <Statement>`

Văn phạm KPL viết bằng BNF

65) $\langle \text{Arguments} \rangle ::= \text{SB_LPAR } \langle \text{Expression} \rangle \langle \text{Arguments2} \rangle \text{SB_RPAR}$

66) $\langle \text{Arguments} \rangle ::= \epsilon$

67) $\langle \text{Arguments2} \rangle ::= \text{SB_COMMA } \langle \text{Expression} \rangle \langle \text{Arguments2} \rangle$

68) $\langle \text{Arguments2} \rangle ::= \epsilon$

68) $\langle \text{Condition} \rangle ::= \langle \text{Expression} \rangle \langle \text{Condition2} \rangle$

69) $\langle \text{Condition2} \rangle ::= \text{SB_EQ } \langle \text{Expression} \rangle$

70) $\langle \text{Condition2} \rangle ::= \text{SB_NEQ } \langle \text{Expression} \rangle$

71) $\langle \text{Condition2} \rangle ::= \text{SB_LE } \langle \text{Expression} \rangle$

72) $\langle \text{Condition2} \rangle ::= \text{SB_LT } \langle \text{Expression} \rangle$

73) $\langle \text{Condition2} \rangle ::= \text{SB_GE } \langle \text{Expression} \rangle$

74) $\langle \text{Condition2} \rangle ::= \text{SB_GT } \langle \text{Expression} \rangle$

Văn phạm KPL viết bằng 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> ::= ε`

- 82) `<Term> ::= <Factor> <Term2>`

- 83) `<Term2> ::= SB_TIMES <Factor> <Term2>`
- 84) `<Term2> ::= SB_SLASH <Factor> <Term2>`
- 85) `<Term2> ::= ε`
- 86) `<Factor> ::= <UnsignedConstant>`
- 87) `<Factor> ::= <Variable>`
- 88) `<Factor> ::= <FunctionApptication>`
- 89) `<Factor> ::= SB_LPAR <Expression> SB_RPAR`

- 90) `<Variable> ::= TK_IDENT <Indexes>`
- 91) `<FunctionApplication> ::= TK_IDENT <Arguments>`

- 92) `<Indexes> ::= SB_LSEL <Expression> SB_RSEL <Indexes>`
- 93) `<Indexes> ::= ε`

Câu hỏi?

- Hãy tính các tập FIRST và FOLLOW cho mỗi ký hiệu không kết thúc

- Ví dụ : với các luật cho $\langle \text{constant} \rangle$

40) $\langle \text{Constant} \rangle ::= \text{SB_PLUS } \langle \text{Constant2} \rangle \quad \alpha_1$

41) $\langle \text{Constant} \rangle ::= \text{SB_MINUS } \langle \text{Constant2} \rangle \quad \alpha_2$

42) $\langle \text{Constant} \rangle ::= \langle \text{Constant2} \rangle \quad \alpha_3$

43) $\langle \text{Constant} \rangle ::= \text{TK_CHAR} \quad \alpha_4$

$\text{FIRST}(\alpha_1) = \{\text{SB_PLUS}\}$

$\text{FIRST}(\alpha_2) = \{\text{SB_MINUS}\}$

$\text{FIRST}(\alpha_3) = \{\text{TK_IDENT}, \text{TK_NUMBER}\}$

$\text{FIRST}(\alpha_4) = \{\text{TK_CHAR}\}$

Giao của từng cặp đều là tập rỗng

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

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

$\text{FOLLOW}(\text{Statement}) = \{\text{SB_SEMICOLON}, \text{KW_END}, \text{KW_ELSE}\}$

/* 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) ε |
| KW_END | 55) ε |
| KW_ELSE | 55) ε |
| ----- | |
| 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