



# Bài 3. Văn phạm sản sinh



# Lý thuyết ngôn ngữ

- Mô hình cho tất cả các ngôn ngữ
- Ngôn ngữ là tập các xâu (sentence, string) trên một bảng chữ nào đó
- Ví dụ về xâu
  - Dãy các bit
  - Số thực
  - Chương trình C
  - Câu tiếng Việt



# Vấn đề biểu diễn ngôn ngữ

- Thực chất là biểu diễn cú pháp của ngôn ngữ
- Biểu diễn phải hữu hạn
- Công cụ sản sinh: văn phạm
- Công cụ đoán nhận: ô tômat

# Phân cấp Chomsky

Lớp ngôn ngữ	Công cụ sản sinh	Công cụ đoán nhận	Ghi chú
Đệ quy kể được	Văn phạm loại 0 (ngữ cấu)	Máy Turing	Các bài toán tổng quát
Cảm ngữ cảnh	Văn phạm cảm ngữ cảnh	Ôtômat tuyến tính giới nội	Ngôn ngữ tự nhiên
Phi ngữ cảnh	Văn phạm phi ngữ cảnh	Ôtômat đẩy xuống	Ngôn ngữ lập trình, phần chính của ngôn ngữ tự nhiên
Chính quy	Văn phạm chính quy Công cụ biểu diễn: Biểu thức chính quy	Ôtômat hữu hạn	Từ vựng của ngôn ngữ tự nhiên, ngôn ngữ lập trình

# Văn phạm xuất phát từ ngôn ngữ tự nhiên

<câu>::=<chủ ngữ> <vị ngữ>

<chủ ngữ>::= <danh ngữ>

<danh ngữ>::= <danh từ> <tính từ>

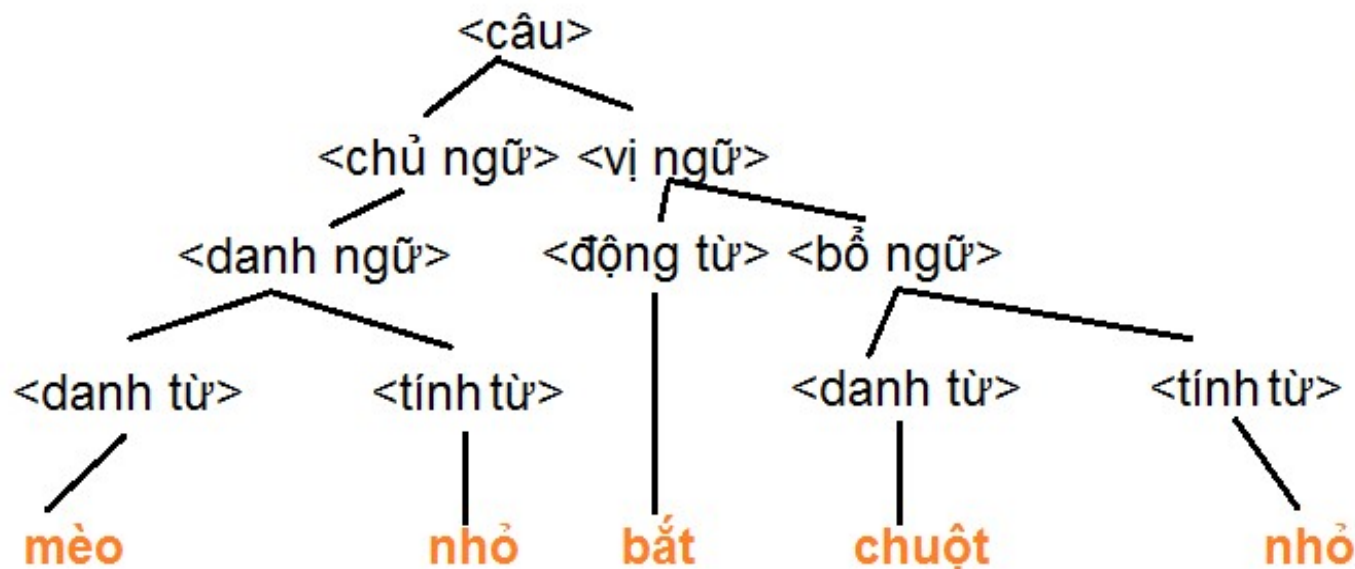
<vị ngữ>::= <động từ> <bổ ngữ>

<bổ ngữ>::= <danh từ> <tính từ>

<động từ> ::= **bắt**

<danh từ>::= **mèo** | **chuột**

<tính từ>::= **nhỏ**






# Văn phạm sản sinh các số thực

$\langle \text{Số} \rangle ::= -\langle \text{Số thập phân} \rangle | \langle \text{số thập phân} \rangle$

$\langle \text{Số thập phân} \rangle ::= \langle \text{Dãy chữ số} \rangle | \langle \text{Dãy chữ số} \rangle . \langle \text{Dãy chữ số} \rangle$

$\langle \text{Dãy chữ số} \rangle ::= \langle \text{Chữ số} \rangle | \langle \text{Chữ số} \rangle \langle \text{Dãy chữ số} \rangle$

$\langle \text{Chữ số} \rangle ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9$



# Làm thế nào để sản sinh ra các xâu ?

Văn phạm phi ngữ cảnh có thể dùng để sản sinh ra các xâu thuộc ngôn ngữ như sau:

*$X$  = Ký hiệu đầu*

***While*** còn ký hiệu không kết thúc  *$Y$*  trong  *$X$*  ***do***

*Áp dụng một trong các sản xuất của, văn  
phạm chẳng hạn  $Y \rightarrow w$*

Khi  *$X$*  chỉ chứa ký hiệu kết thúc, nó là xâu được sản sinh bởi văn phạm.



Ví dụ

$$S \rightarrow -A \mid A$$
$$A \rightarrow B.B \mid B$$
$$B \rightarrow BC \mid C$$
$$C \rightarrow 0 \mid 1 \mid 2 \mid \dots \mid 9$$



# Quá trình sản sinh sâu -3.14

## Quá trình thay thế

S

-A

-B.B

-B.BC

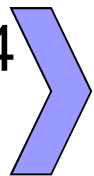
-C.BC

-C.CC

-3.CC

-3.1C

-3.14



Quá trình suy dẫn

## Sản xuất được sử dụng

$S \rightarrow -A$

$A \rightarrow B.B$

$B \rightarrow BC$

$B \rightarrow C$

$B \rightarrow C$

$C \rightarrow 3$

$C \rightarrow 1$

$C \rightarrow 4$



# Suy dẫn (Derivations)

- Mỗi lần thực hiện việc thay thế là một bước suy dẫn.
- Nếu mỗi dạng câu có nhiều ký hiệu không kết thúc để thay thế có thể sử dụng bất cứ sản xuất nào.

# Suy dẫn trái và suy dẫn phải

- Nếu giải thuật phân tích cú pháp chọn ký hiệu không kết thúc cực trái hay cực phải để thay thế, kết quả của nó là suy dẫn trái hoặc suy dẫn phải

Ví dụ suy dẫn trái:

$$\begin{aligned} S &\Rightarrow -A \Rightarrow -B.B \Rightarrow -C.B \Rightarrow -3.B \Rightarrow -3.BC \Rightarrow -3.CC \\ &\Rightarrow -3.1C \Rightarrow -3.14 \end{aligned}$$

Ví dụ suy dẫn phải:

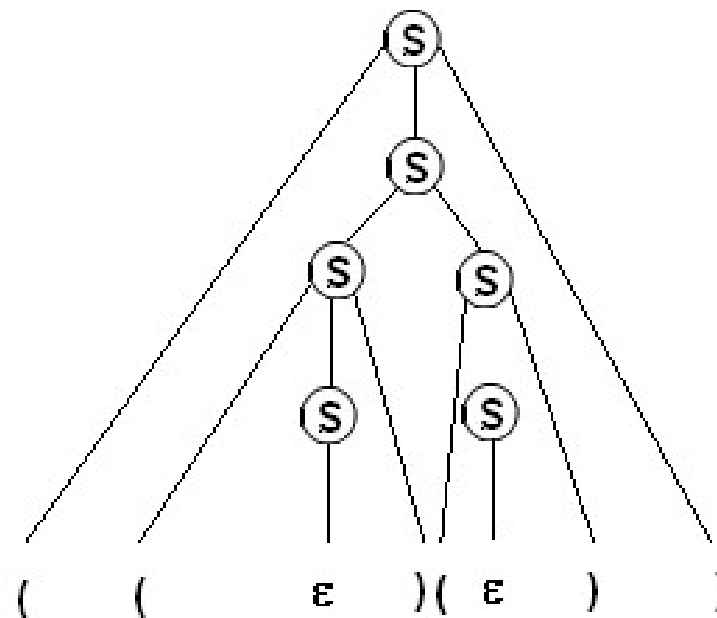
$$\begin{aligned} S &\Rightarrow -A \Rightarrow -B.B \Rightarrow -B.BC \Rightarrow -B.B4 \Rightarrow -B.C4 \Rightarrow -B.14 \\ &\Rightarrow -C.14 \Rightarrow -3.14 \end{aligned}$$

# Cây suy dẫn(Cây phân tích cú pháp)

Cây suy dẫn có những đặc điểm sau

- 1) Mỗi nút của cây có nhãn là ký hiệu kết thúc, ký hiệu không kết thúc hoặc  $\epsilon$  (xâu rỗng)
- 2) Nhãn của nút gốc là S (ký hiệu đầu)
- 3) Nút trong có nhãn là ký hiệu không kết thúc(nút lá có nhãn là ký hiệu kết thúc)
- 4) Nút A có các nút con từ trái qua phải là  $X_1, X_2, \dots, X_k$  thì có một sản xuất dạng  $A \rightarrow X_1 X_2 \dots X_k$
- 5) Nút lá có thể có nhãn  $\epsilon$  chỉ khi tồn tại sản xuất  $A \rightarrow \epsilon$  và nút cha của nút lá chỉ có một nút con duy nhất

G:  $S \rightarrow SS \mid (S) \mid \epsilon$      $w=((()))$



# Văn phạm nhập nhằng

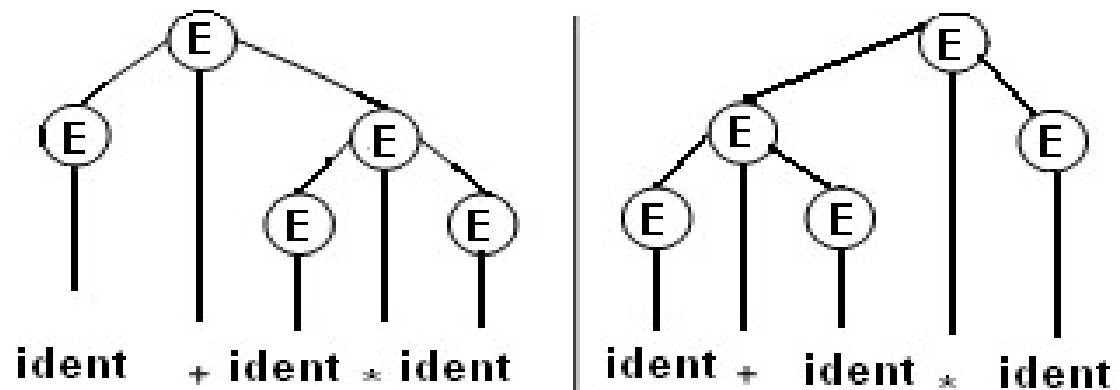
Văn phạm

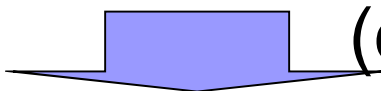
$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow ( E )$

$E \rightarrow \text{TK\_IDENT}$



Cho phép đưa ra hai suy dẫn khác nhau cho  
xâu  $\text{TK\_IDENT} + \text{TK\_IDENT} * \text{TK\_IDENT}$   
 (chẳng hạn  $x + y * z$ )

Văn phạm là nhập nhằng

# Khử nhập nhằng

$E \rightarrow E + T$

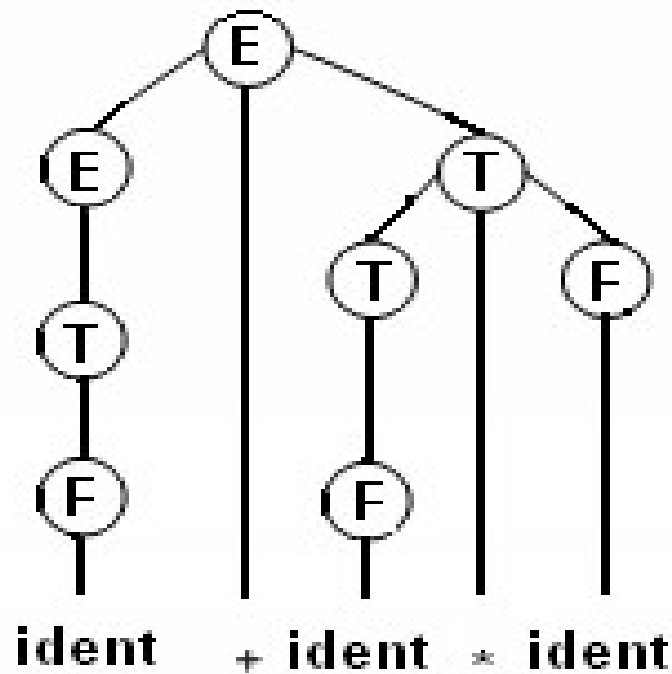
$E \rightarrow T$

$T \rightarrow T * F$

$T \rightarrow F$

$F \rightarrow ( E )$

$F \rightarrow \text{TK\_IDENT}$



(Bằng cách thêm các ký hiệu không kết thúc và các sản xuất để đảm bảo thứ tự ưu tiên)

# Đệ quy

- Một sản xuất là đệ quy nếu  $X \Rightarrow^* \omega_1 X \omega_2$
- Có thể dùng để biểu diễn các quá trình lặp hay cấu trúc lồng nhau

**Đệ quy trực tiếp**  $X \Rightarrow \omega_1 X \omega_2$

**Đệ quy trái**  $X \Rightarrow b \mid \textcolor{red}{X}a.$

$X \Rightarrow Xa \Rightarrow Xaa \Rightarrow Xaaa \Rightarrow baaaaa \dots$

**Đệ quy phải**  $X \Rightarrow b \mid a \textcolor{red}{X}.$

$X \Rightarrow aX \Rightarrow a aX \Rightarrow a a aX \Rightarrow \dots a a a a a b$

**Đệ quy giữa**  $X \Rightarrow b \mid ( \textcolor{red}{X} ).$

$X \Rightarrow (X) \Rightarrow ((X)) \Rightarrow (((X))) \Rightarrow (((\dots (b) \dots)))$

**Đệ quy gián tiếp**  $X \Rightarrow^* \omega_1 X \omega_2$



# Khử đệ quy trái

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow ( E ) \mid \text{TK\_IDENT}$$

Khử đệ quy trái bằng cách thêm ký hiệu không kết thúc và sản xuất mới

$$E \rightarrow T E'$$
$$E' \rightarrow + T E' \mid \varepsilon$$
$$T \rightarrow F T'$$
$$T' \rightarrow * F T' \mid \varepsilon$$
$$F \rightarrow ( E ) \mid \text{TK\_IDENT}$$





# Bài 4

## BNF và sơ đồ cú pháp




## Công thức siêu ngữ Backus và các biến thể

- **Siêu ngữ (metalanguage)**: Ngôn ngữ sử dụng các lệnh để mô tả ngôn ngữ khác
- **BNF (Backus Naur Form)** là dạng siêu cú pháp để mô tả các ngôn ngữ lập trình
- BNF được sử dụng rộng rãi để mô tả văn phạm của các ngôn ngữ lập trình, tập lệnh và các giao thức truyền thông.



# Công thức siêu ngữ Backus

- Ký pháp BNF là một tập các luật ,về trái của mỗi luật là một cấu trúc cú pháp.
- Tên của cấu trúc cú pháp được gọi là ký hiệu không kết thúc.
- Các ký hiệu không kết thúc thường được bao trong cặp  $\langle \rangle$ .
- Các ký hiệu kết thúc thường được phân cách bằng cặp nháy đơn hoặc nháy kép



# Công thức siêu ngữ Backus và các biến thể

- Mỗi ký hiệu không kết thúc được định nghĩa bằng một hay nhiều luật.
- Các luật có dạng

$$N ::= s$$

*(N là ký hiệu không kết thúc, s là một xâu gồm 0 hay nhiều ký hiệu kết thúc và không kết thúc. Các luật có chung vế trái được phân cách bằng | )*

## Ví dụ về BNF : văn phạm sản sinh các số thực

$\langle \text{số thực} \rangle ::= \langle \text{dấu} \rangle \langle \text{số tự nhiên} \rangle \mid$

$\langle \text{dấu} \rangle \langle \text{số tự nhiên} \rangle '.' \langle \text{dãy chữ số} \rangle \mid$

$\langle \text{dấu} \rangle '.' \langle \text{chữ số} \rangle \langle \text{dãy chữ số} \rangle \mid$

$\langle \text{số thực} \rangle 'e' \langle \text{số tự nhiên} \rangle$

$\langle \text{dấu} \rangle ::= \varepsilon \mid '+' \mid '-'$

$\langle \text{số tự nhiên} \rangle ::= '0' \mid \langle \text{chữ số khác } 0 \rangle \langle \text{dãy chữ số} \rangle$

$\langle \text{chữ số khác } 0 \rangle ::= '1' \mid '2' \mid '3' \mid '4' \mid '5' \mid '6' \mid '7' \mid '8' \mid '9'$

$\langle \text{dãy chữ số} \rangle ::= \varepsilon \mid \langle \text{chữ số} \rangle \langle \text{dãy chữ số} \rangle$

$\langle \text{chữ số} \rangle ::= '0' \mid '1' \mid '2' \mid '3' \mid '4' \mid '5' \mid '6' \mid '7' \mid '8' \mid '9'$

Văn phạm này được viết bằng BNF, một công cụ rất phổ biến để biểu diễn cú pháp ngôn ngữ lập trình



# EBNF

- EBNF (Extended BNF ) được phát triển từ ký pháp BNF. EBNF có ký pháp tương tự BNF nhưng được đơn giản hoá bằng cách sử dụng một số ký hiệu đặc biệt :

[] phần này là tùy chọn(có hoặc không)

{ } phần này có thể lặp lại một số lần tùy ý hoặc không xuất hiện lần nào (Nếu lặp lại m hay n lần , dùng n hay m là chỉ số trên hoặc dưới)

Không cần dùng “ cho ký hiệu kết thúc

# So sánh BNF và EBNF

## Ví dụ

- Trong EBNF

*<Lệnh if> ::= IF <Biểu thức> THEN <Lệnh>  
[ELSE <Lệnh>]*

- Trong BNF

*<Lệnh if> ::= 'IF' <Biểu thức> 'THEN'  
<Lệnh> | 'IF' <Biểu thức> THEN <Lệnh>  
'ELSE' <Lệnh>*

## Ví dụ: Một đoạn văn phạm Python trên EBNF

compound\_stmt: if\_stmt | while\_stmt | for\_stmt | try\_stmt |  
with\_stmt | funcdef | classdef | decorated | async\_stmt

async\_stmt: 'async' (funcdef | with\_stmt | for\_stmt)

if\_stmt: 'if' test ':' suite ('elif' test ':' suite)\* ['else' ':' suite]

while\_stmt: 'while' test ':' suite ['else' ':' suite]

for\_stmt: 'for' exprlist 'in' testlist ':' suite ['else' ':' suite]

try\_stmt: ('try' ':' suite  
          ((except\_clause ':' suite)+  
          ['else' ':' suite]  
          ['finally' ':' suite] |  
          'finally' ':' suite))

with\_stmt: 'with' with\_item (',' with\_item)\* ':' suite

with\_item: test ['as' expr]





# 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`



- 26

# Văn phạm KPL viết bằng BNF

34) `<BasicType> ::= KW_INTEGER`

35) `<BasicType> ::= KW_CHAR`

36) `<UnsignedConstant> ::= TK_NUMBER`

37) `<UnsignedConstant> ::= TK_IDENT`

38) `<UnsignedConstant> ::= TK_CHAR`

40) `<Constant> ::= SB_PLUS <Constant2>`

41) `<Constant> ::= SB_MINUS <Constant2>`

42) `<Constant> ::= <Constant2>`

43) `<Constant> ::= TK_CHAR`

44) `<Constant2> ::= TK_IDENT`

45) `<Constant2> ::= TK_NUMBER`

46) `<Statements> ::= <Statement> <Statements2>`

47) `<Statements2> ::= SB_SEMICOLON <Statement> <Statements2>`

48) `<Statements2> ::= ε`

# 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> ::=  $\epsilon$
- 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> ::=  $\epsilon$
  
- 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) `<Arguments> ::= SB_LPAR <Expression> <Arguments2> SB_RPAR`

66) `<Arguments> ::= ε`

67) `<Arguments2> ::= SB_COMMA <Expression> <Arguments2>`

68) `<Arguments2> ::= ε`

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 KPL viết bằng BNF

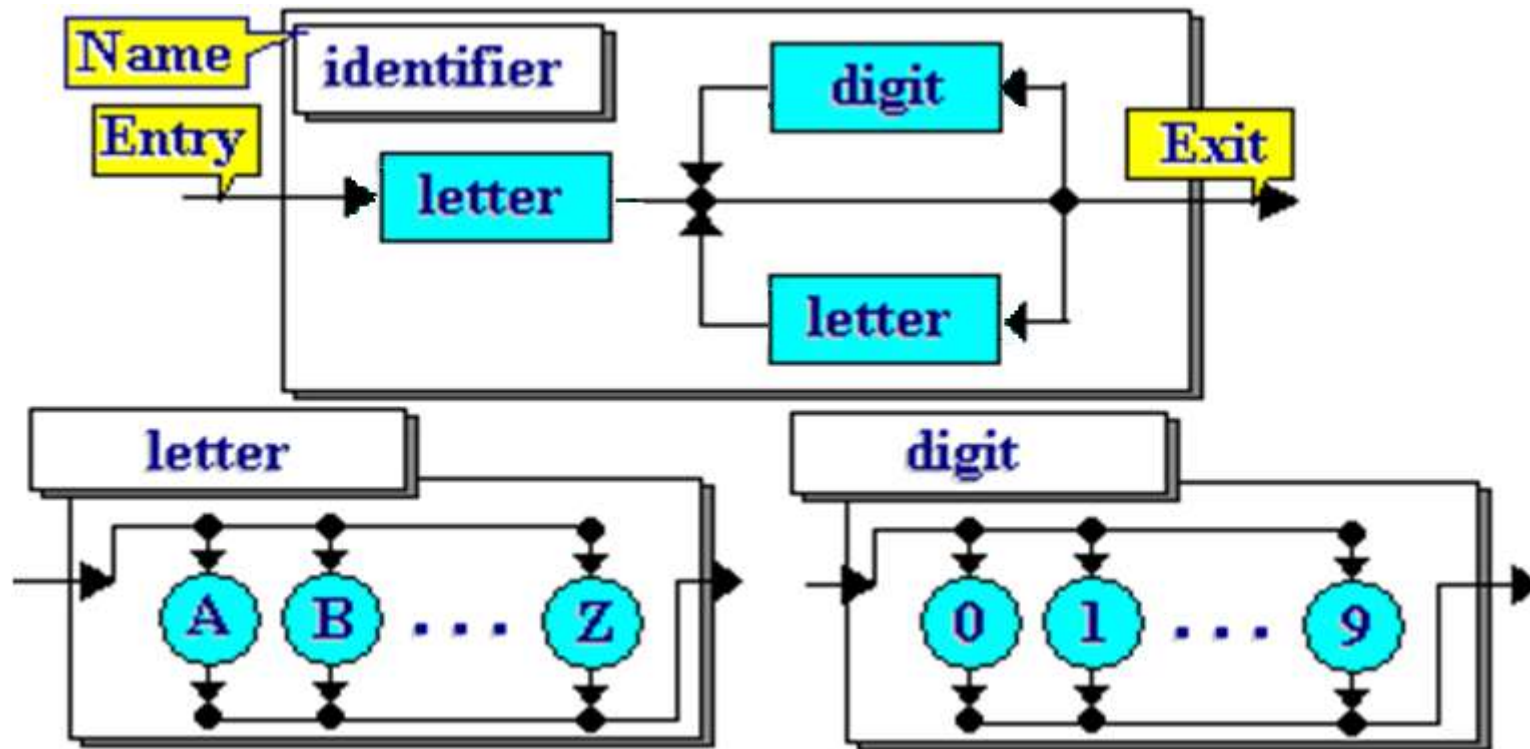
- 75) `<Expression> ::= SB_PLUS <Expression2>`
- 76) `<Expression> ::= SB_MINUS <Expression2>`
- 77) `<Expression> ::= <Expression2>`
  
- 78) `<Expression2> ::= <Term> <Expression3>`
  
- 79) `<Expression3> ::= SB_PLUS <Term> <Expression3>`
- 80) `<Expression3> ::= SB_MINUS <Term> <Expression3>`
- 81) `<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`



# Sơ đồ cú pháp

- Là công cụ để mô tả cú pháp của ngôn ngữ lập trình dưới dạng đồ thị
- Mỗi sơ đồ cú pháp là một đồ thị định hướng với lối vào và lối ra xác định.
- Mỗi sơ đồ cú pháp có một tên duy nhất

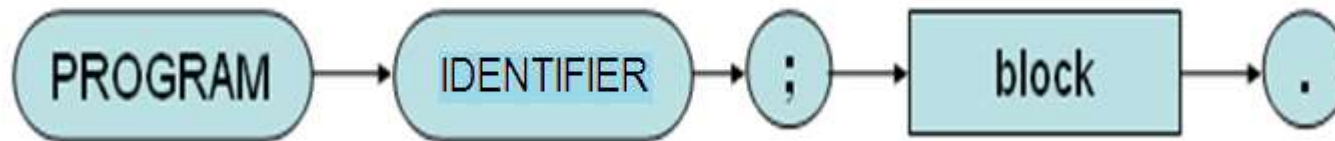
# Ví dụ một sơ đồ cú pháp



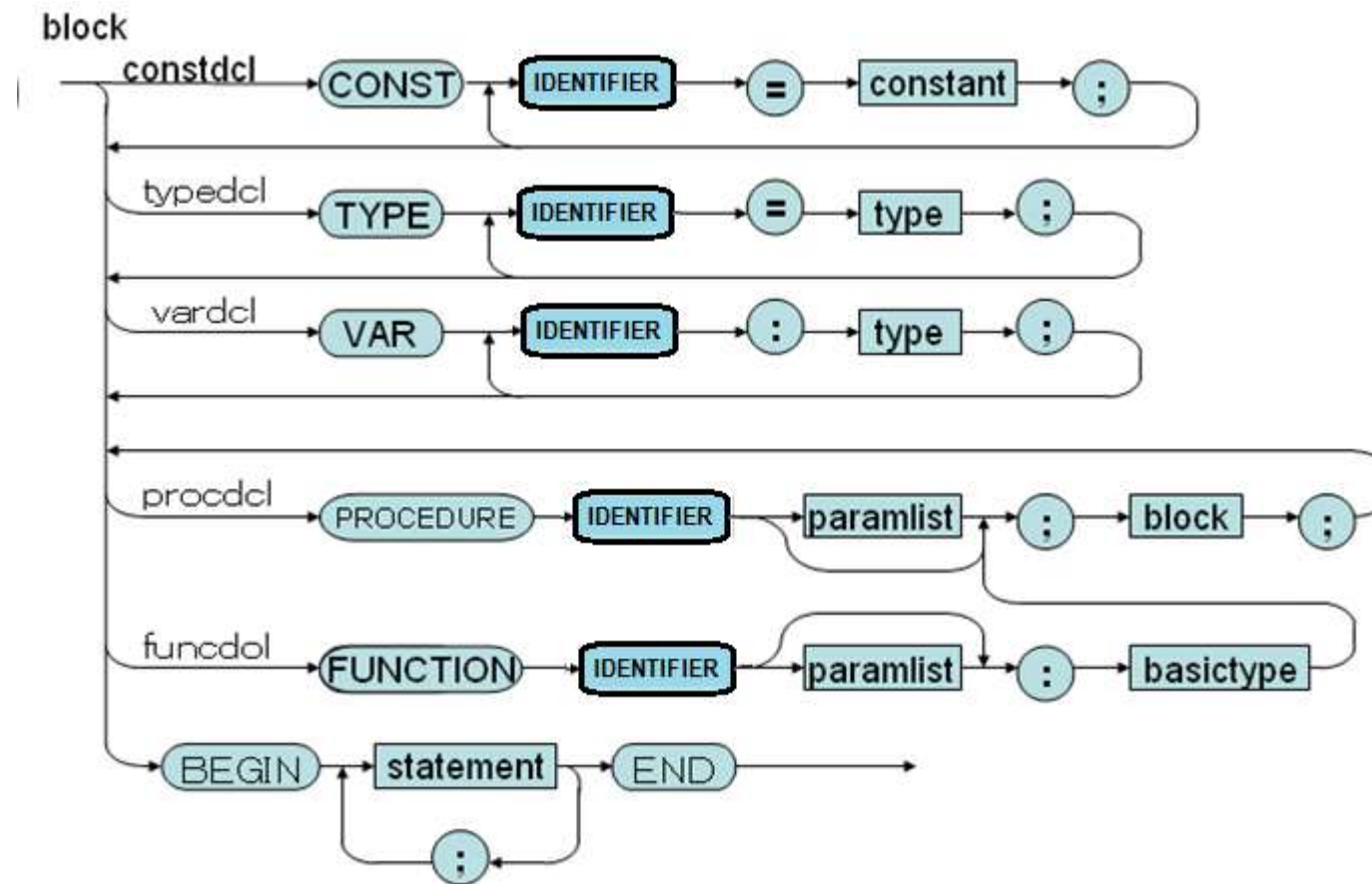


# Sơ đồ cú pháp của KPL (Tổng thể CT)

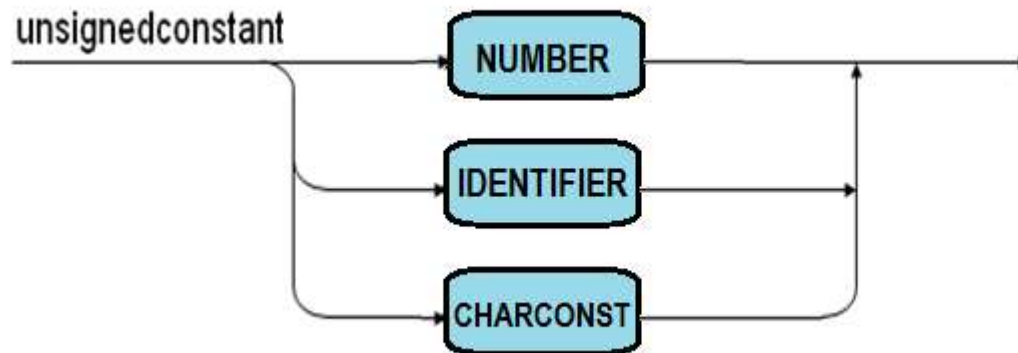
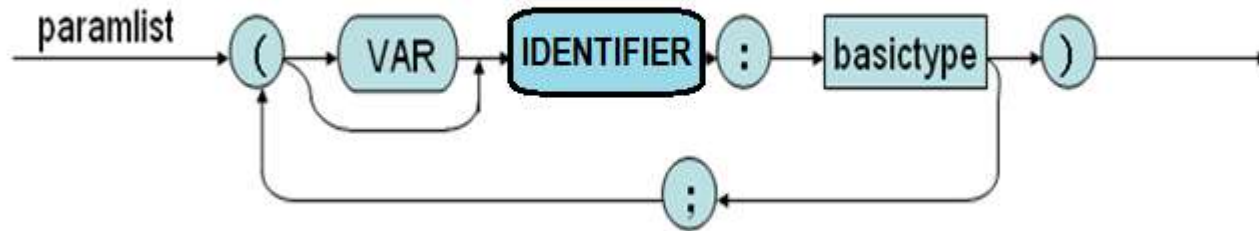
program



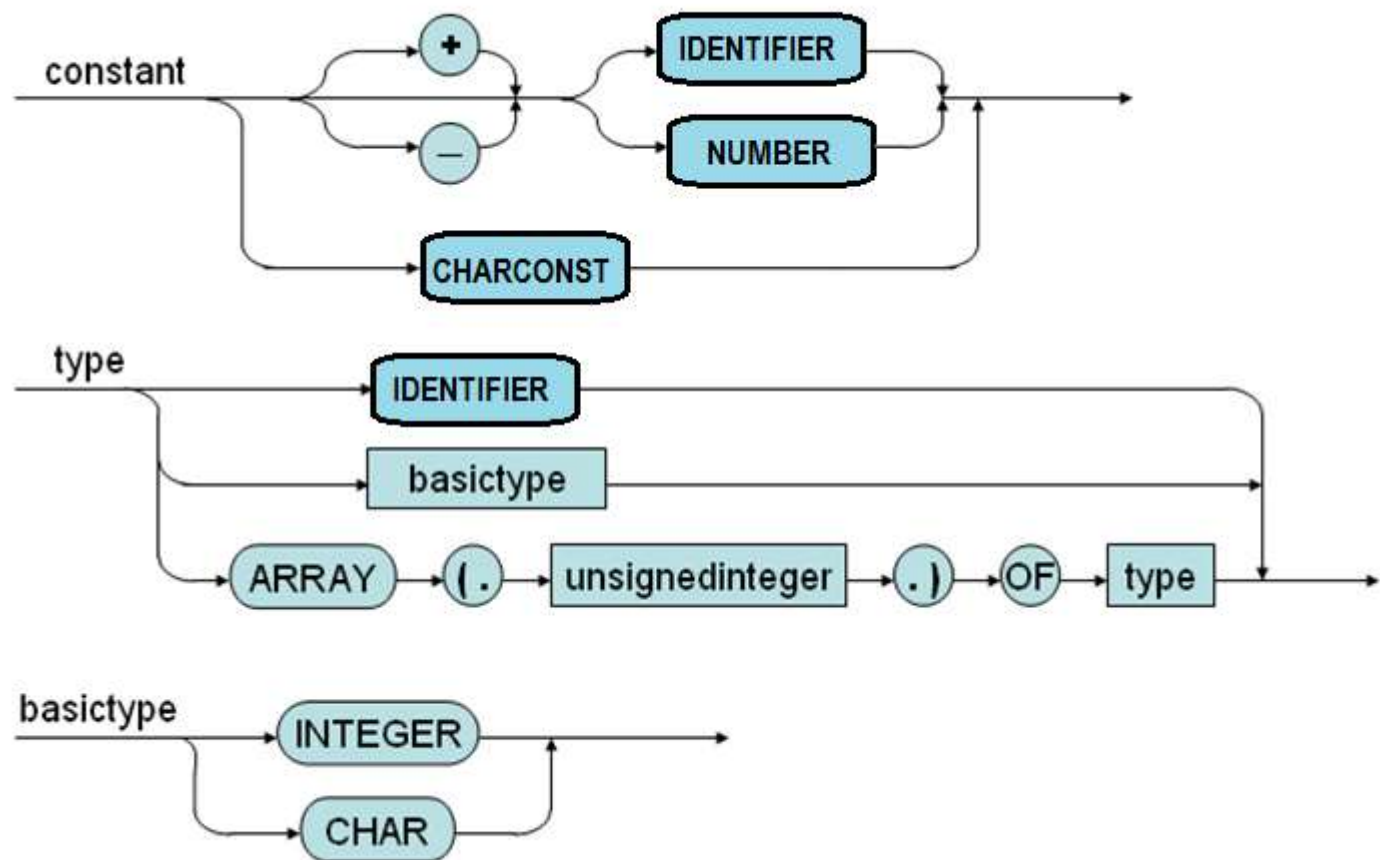
# Sơ đồ cú pháp của KPL (Khối)



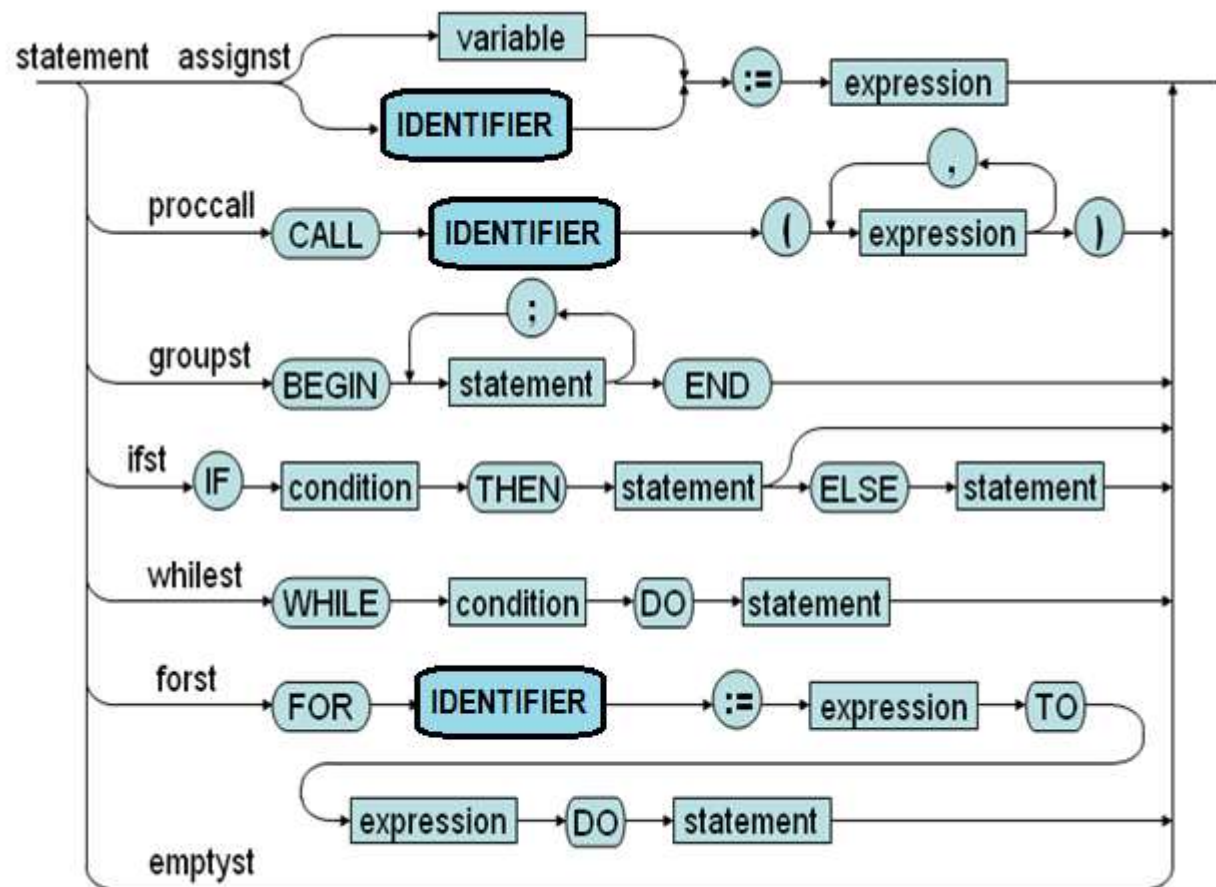
# Sơ đồ cú pháp của KPL (tham số, hằng không dấu)



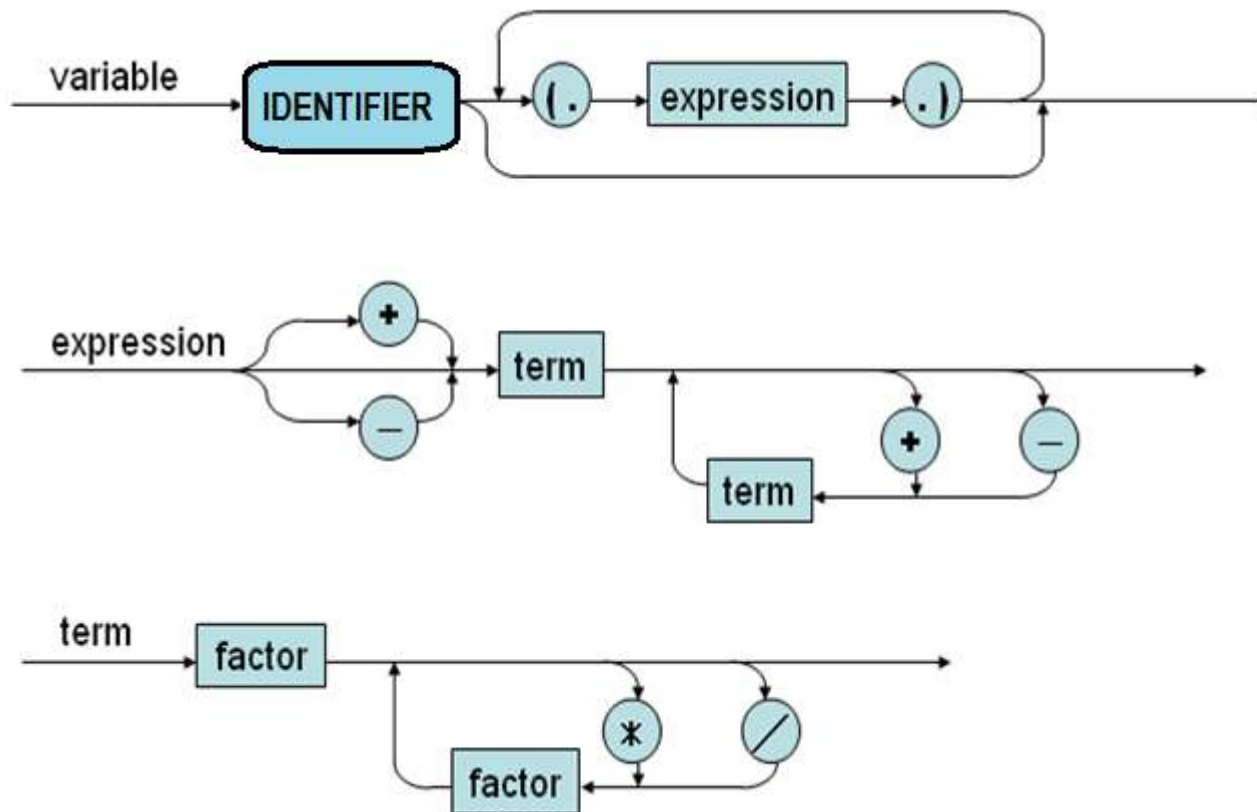
# Sơ đồ cú pháp của KPL (Khai báo)



# 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)

