

## 付録 D 構 文

(出典)。Pascal, ウィルト他,  
培風館(1981)。

拡張バッカス・ナウア記法(Extended Backus-Naur Form, EBNF)による、プログラミング言語の構文の仕様は、その言語における文の構成の仕方を記述した規則あるいは生成規則の集まりからなる。それらの集まりを、一般に“文法”とよんでいる。各生成規則は非終端記号と EBNF 式とからなり、それらの間は等号で区切って、最後はピリオドで終わる。この非終端記号は“超越名(日本語名で表わす構文上の定数)”であり、EBNF 式はその定義である。

EBNF 式は空または一つ以上の記号や非終端記号の他に、以下の表に示す超記号からなる。

超記号	意味
=	定義
	別の選択
.	生成規則の終り
[X]	空または X
{X}	空または 1 回以上の X の繰返し
(X   Y)	グループ化、X または Y のどちらか
“XYZ”	終端記号 XYZ
超記号	非終端記号 超記号

たとえば、EBNF は自分自身の構文を使うと次のように定義できる。

構文	= { 生成規則 } .
生成規則	= 非終端記号 “=” 式 “.” .
式	= 項 { “ ” 項 } .
項	= 因子 { 因子 } .
因子	= 非終端記号   終端記号   “( 式 ” )”   “[ 式 ” ]”   “{ 式 ” }” .
終端記号	= “““ 文字 { 文字 } “””” .
非終端記号	= 英字 { 英字   数字 } .

## 注：

1. 終端記号(リテラル)は必ず前後を引用符(“と”)でくくる。ただし、引用符自身をくくるときは、それを2回続けて書く。次に示す Pascal の EBNFにおいて、“[”と“]”は Pascal プログラムにおける左右の角かっこを表わし、[と]は EBNF 式の中の超記号で、それらでくくられたものの0回または1回の表現を表わす。
2. どのような構文にも、その言語のすべての文を生成させ、どの EBNF 式の中でも使用していない超越名、すなわち開始記号がある。Pascal の開始記号はプログラムである。
3. 超越名には、本文中の EBNF の中で使用していても、ここに載せていないものがある(たとえば、符号つき数)。

## EBNF による Pascal の構文定義

プログラム	= プログラム頭部 “;” ブロック “.” .
プログラム頭部	= “program” 名前 [ プログラムパラメタ並び ] .
プログラムパラメタ並び	= “(” 名前並び “)” .
ブロック	= ラベル宣言部 定数定義部 型定義部 変数宣言部 手続き・関数宣言部 実行部.
ラベル宣言部	= [“label” 数字列 {“,” 数字列} “;” ] .
定数定義部	= [“const” 定数定義 “;” {定数定義 “;”} ] .
型定義部	= [“type” 型定義 “;” {型定義 “;”} ] .
変数宣言部	= [“var” 変数宣言 “;” {変数宣言 “;”} ] .
手続き・関数宣言部	= { (手続き宣言   関数宣言) “;” } .
実行部	= 複合文 .
定数定義	= 名前 “=” 定数 .
型定義	= 名前 “=” 型 .
変数宣言	= 名前並び “:” 型 .
手続き宣言	= 手続き頭部 “;” ブロック   手続き頭部 “;” 指令   手続き標示 “;” ブロック .
関数宣言	= 関数頭部 “;” ブロック   関数頭部 “;” 指令   関数標示 “;” ブロック .

手続き頭部	= “procedure” 名前 [ 仮パラメタ並び ] .
手続き標示	= “procedure” 手続き名 .
関数頭部	= “function” 名前 [ 仮パラメタ並び ] “:” 結果型 .
関数標示	= “function” 関数名 .
仮パラメタ並び	= “(” 仮パラメタ節 {“;” 仮パラメタ節} “)” .
仮パラメタ節	= 値/パラメタ仕様   変数/パラメタ仕様   手続き/パラメタ仕様   関数/パラメタ仕様 .
値/パラメタ仕様	= 名前並び “:” (型名   整合配列形式) .
変数/パラメタ仕様	= “var” 名前並び “:” (型名   整合配列形式) .
手続き/パラメタ仕様	= 手続き頭部 .
関数/パラメタ仕様	= 関数頭部 .
整合配列形式	= パック状整合配列形式   アンパック状整合配列形式 .
/パック状整合配列形式	= “packed” “array” “[ 添字型仕様 ”] “of” 型名 .
アンパック状整合配列形式	= “array” “[ 添字型仕様 {“;” 添字型仕様 } ”] “of” (型名   整合配列形式) .
添字型仕様	= 名前 “..” 名前 “:” 順序型名 .
複合文	= “begin” 文並び “end” .
文並び	= 文 {“;” 文} .
文	= [ラベル “:”] (単純文   構造文) .
単純文	= 空文   代入文   手続き呼出し文   goto 文 .
構造文	= 複合文   条件文   繰返し文   with 文 .
条件文	= if 文   case 文 .
繰返し文	= while 文   repeat 文   for 文 .
空文	= .
代入文	= (変数   関数名) “:=” 式 .
手続き呼出し文	= 手続き名 [ 実パラメタ並び   書出しパラメタ並び ] .
goto 文	= “goto” ラベル .
if 文	= “if” 論理式 “then” 文 [ “else” 文 ] .
case 文	= “case” 選択式 “of” 選択肢 {“;” 選択肢} [“;”] “end” .
repeat 文	= “repeat” 文並び “until” 論理式 .
while 文	= “while” 論理式 “do” 文 .
for 文	= “for” 制御変数 “:=” 初期値 (“to”   “downto”) 終値 “do” 文 .

with文	= “with” レコード変数並び “do” 文 .
レコード変数並び	= レコード変数 {“,” レコード変数} .
選択式	= 順序式 .
選択肢	= 定数 {“,” 定数} “:” 文 .
制御変数	= 変数名 .
初期値	= 順序式 .
終値	= 順序式 .
型	= 単純型   構造型   ポインタ型 .
単純型	= 順序型   実数型名 .
構造型	= [“packed”] アンパック状構造型   構造型名 .
ポインタ型	= “↑” 対象型   ポインタ型名 .
順序型	= 列挙型   部分範囲型   順序型名 .
アンパック状構造型	= 配列型   レコード型   集合型   ファイル型 .
対象型	= 型名 .
列挙型	= (“名前並び”) .
部分範囲型	= 定数 “..” 定数 .
配列型	= “array” “[” 添字型 {“,” 添字型} “]” “of” 要素型 .
レコード型	= “record” 欄並び “end” .
集合型	= “set” “of” 基底型 .
ファイル型	= “file” “of” 要素型 .
添字型	= 順序型 .
要素型	= 型 .
基底型	= 順序型 .
結果型	= 順序型名   実数型名   ポインタ型名 .
欄並び	= [(固定部 [“;” 可変部]   可変部) [“;”]] .
固定部	= レコード節 {“;” レコード節} .
可変部	= “case” 可変要素選択子 “of” 可変要素 {“;” 可変要素} .
レコード節	= 名前並び “:” 型 .
可変要素選択子	= [タグ名 “:”] タグ型 .
可変要素	= 定数 {“,” 定数} “:” (“欄並び”) .
タグ型	= 順序型名 .
タグ名	= 名前 .
定数	= [符号] (符号なし数   定数名)   文字列 .

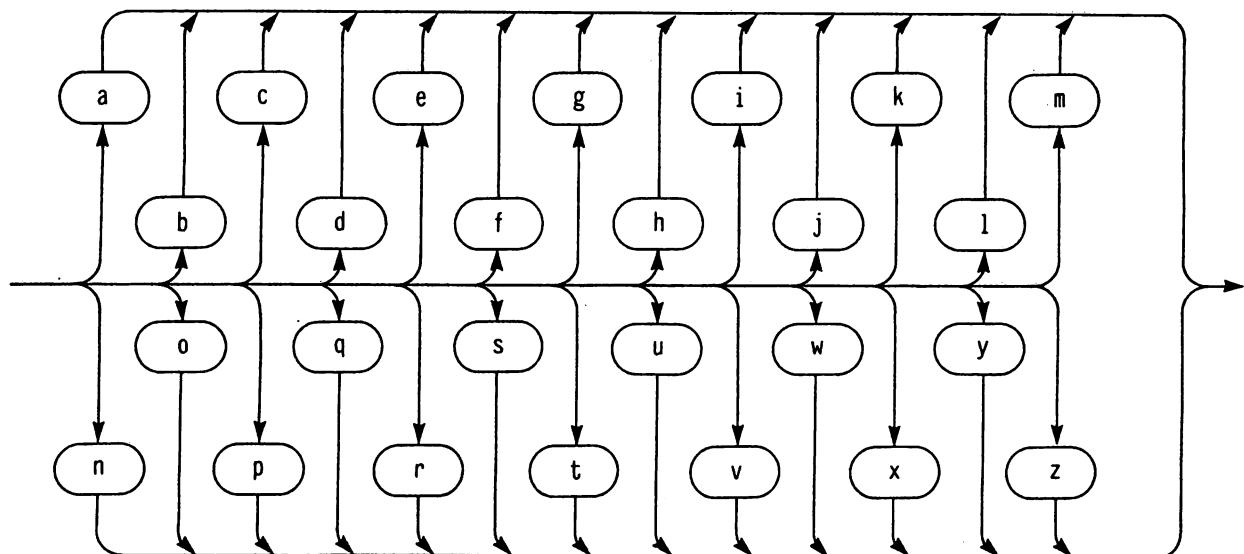
式	= 単純式 [ 関係演算子 単純式 ] .
単純式	= [ 符号 ] 項 { 加減演算子 項 } .
項	= 因子 { 乗除演算子 因子 } .
因子	= 符号なし定数   限界名   変数   集合構成子   関数呼出し   "not" 因子   "(" 式 ")" .
関係演算子	= "="   "<>"   "<"   "<="   ">"   ">="   "in" .
加減演算子	= "+"   "-"   "or" .
乗除演算子	= "*"   "/"   "div"   "mod"   "and" .
符号なし定数	= 符号なし数   文字列   定数名   "nil" .
関数呼出し	= 関数名 [ 実パラメタ並び ] .
変数	= 純変数   要素変数   対象変数   バッファ変数 .
純変数	= 変数名 .
要素変数	= 添字つき変数   欄指定子 .
対象変数	= ポインタ変数 "↑" .
バッファ変数	= ファイル変数 "↑" .
添字つき変数	= 配列変数 "[" 添字 { " , " 添字 } "]" .
欄指定子	= [ レコード変数 ":" ] 欄名 .
集合構成子	= "[" [ 要素記述 { " , " 要素記述 } ] "]" .
要素記述	= 順序式 [ ".." 順序式 ] .
実パラメタ並び	= "(" 実パラメタ { " , " 実パラメタ } ")" .
実パラメタ	= 式   変数   手書き名   関数名 .
書出しパラメタ並び	= "(" ( ファイル変数   書出しパラメタ ) { " , " 書出しパラメタ } ")" .
書出しパラメタ	= 式 [ ":" 整数式 [ ":" 整数式 ] ] .
配列変数	= 変数 .
レコード変数	= 変数 .
ファイル変数	= 変数 .
ポインタ変数	= 変数 .
整数式	= 順序式 .
論理式	= 順序式 .
添字	= 順序式 .
順序式	= 式 .
ポインタ型名	= 型名 .
構造型名	= 型名 .
順序型名	= 型名 .
実数型名	= 型名 .
定数名	= 型名 .

型名	= 名前 .
変数名	= 名前 .
欄名	= 名前 .
手書き名	= 名前 .
関数名	= 名前 .
限界名	= 名前 .
符号なし数	= 符号なし整数   符号なし実数 .
名前並び	= 名前 { “,” 名前 } .
名前	= 英字 { 英字   数字 } .
指令	= 英字 { 英字   数字 } .
ラベル	= 数字列 .
符号なし整数	= 数字列 .
符号なし実数	= 数字列 “.” 数字列 [ “e” 指数部 ]   数字列 “e” 指数部 .
指数部	= [ 符号 ] 数字列 .
符号	= “+”   “-” .
文字列	= “!” 文字列要素 “!” .
数字列	= 数字 { 数字 } .
英字	= “a”   “b”   “c”   “d”   “e”   “f”   “g”   “h”   “i”   “j”   “k”   “l”   “m”   “n”   “o”   “p”   “q”   “r”   “s”   “t”   “u”   “v”   “w”   “x”   “y”   “z” .
数字	= “0”   “1”   “2”   “3”   “4”   “5”   “6”   “7”   “8”   “9” .
文字列要素	= “ ”   アポストロフィ以外の任意の文字 .

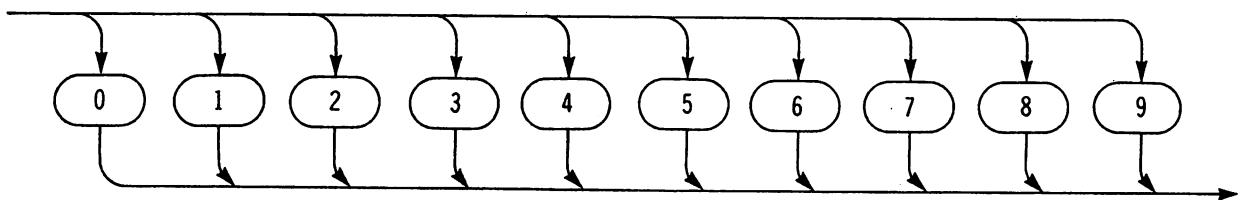
## 構文図

英字、数字、名前、指令、符号なし整数、符号なし数および文字列の構文図は、文字からの基本記号の作り方を表わす。その他の構文図は記号からの構文要素の作り方を表わす。

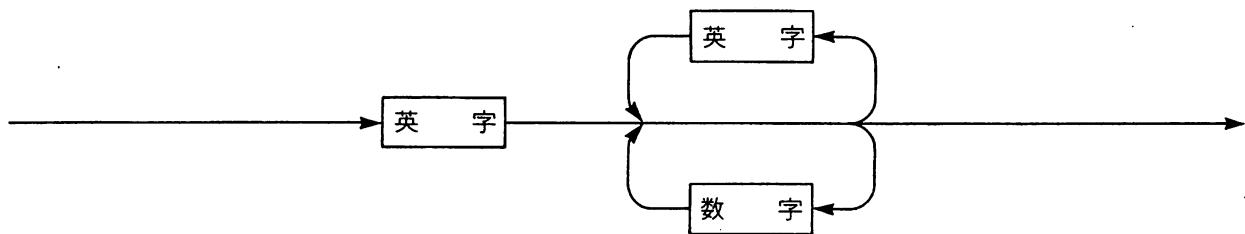
### 英 字



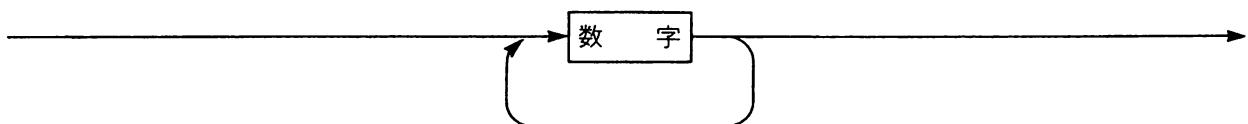
### 数 字



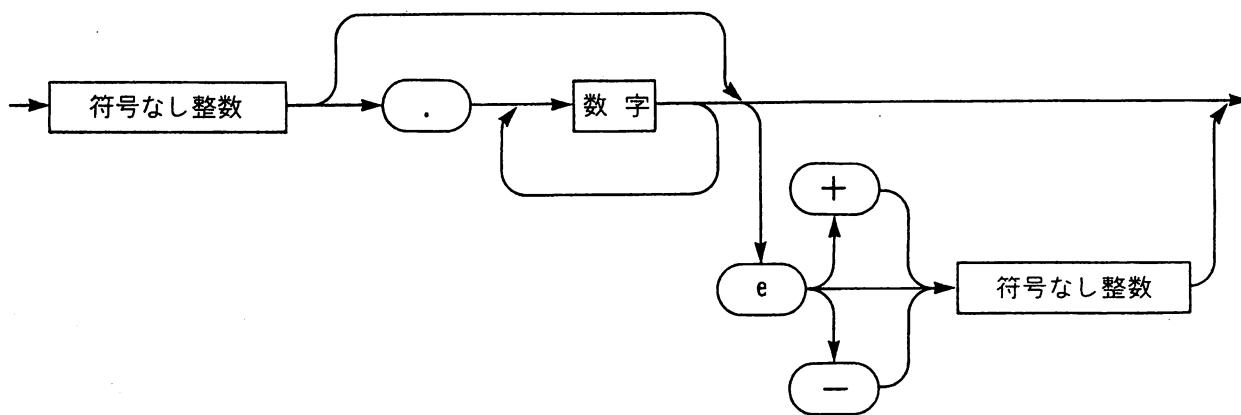
### 名前 および 指令



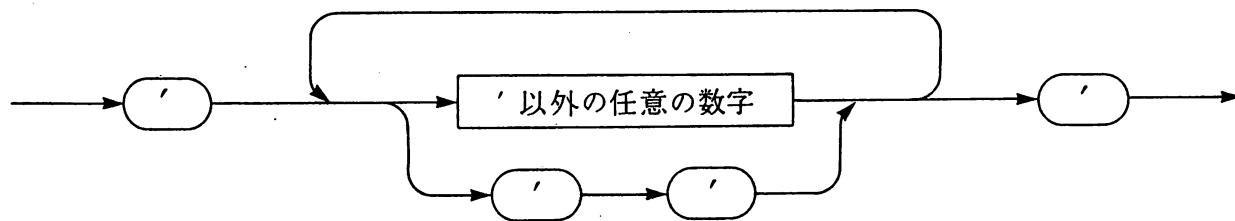
### 符号なし整数



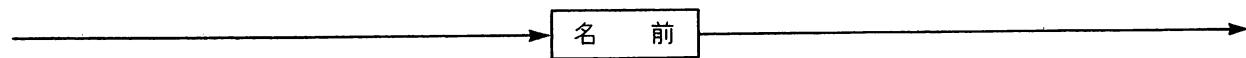
## 符号なし数



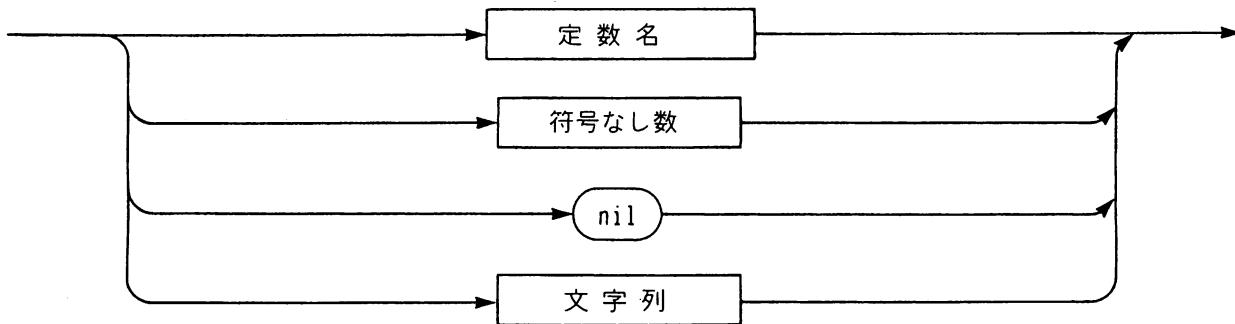
## 文字列



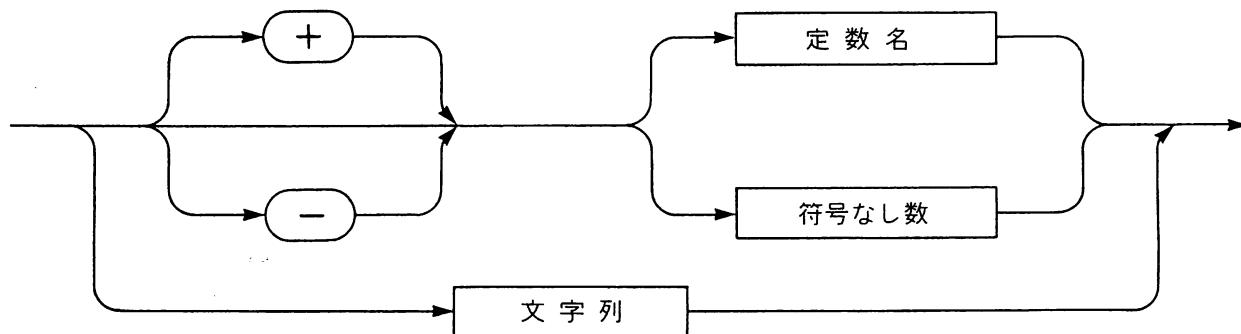
定数名, 変数名, 欄名, 限界名, 型名, 手続き名, および 関数名



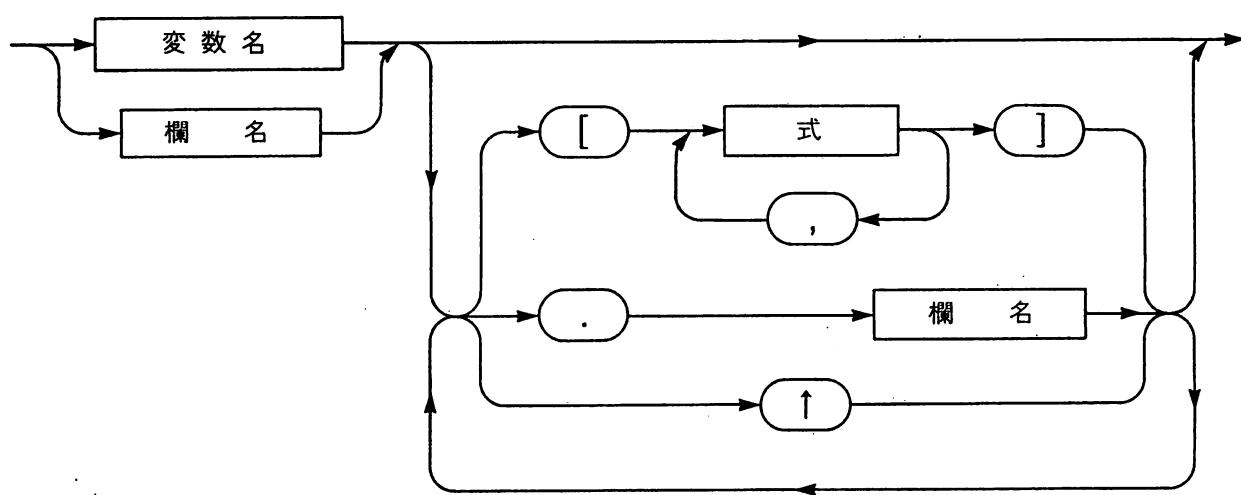
## 符号なし定数



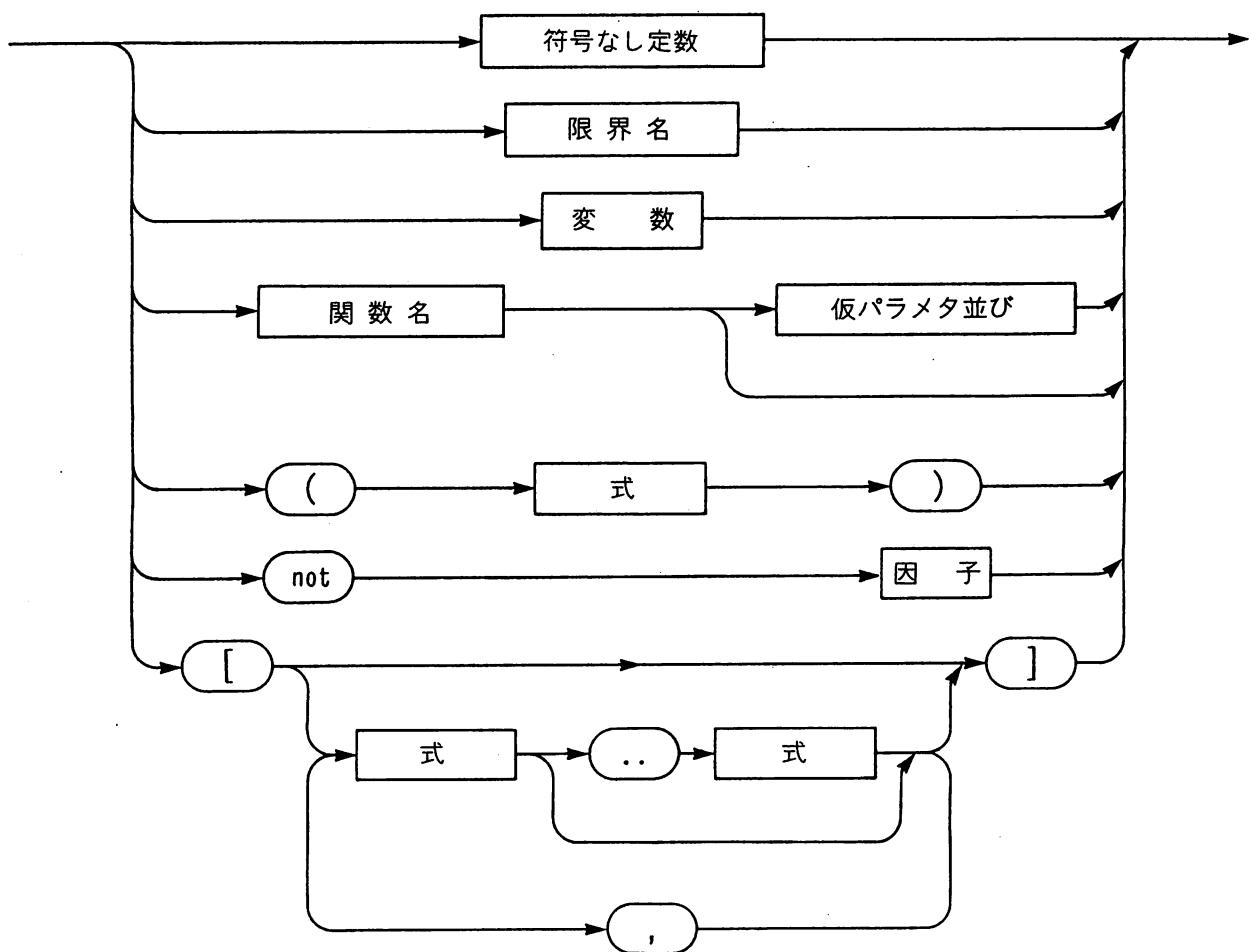
## 定 数



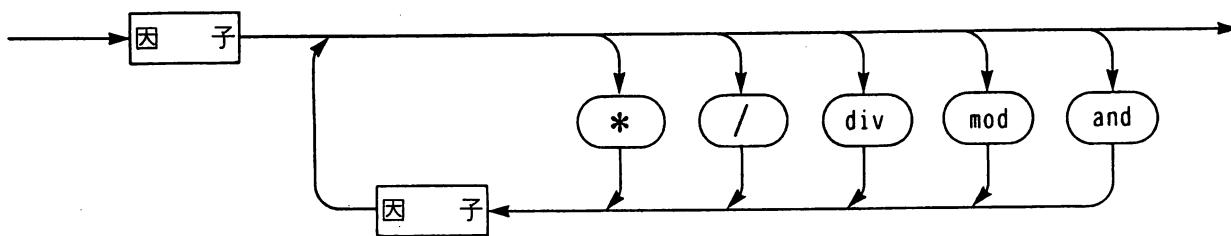
## 変 数



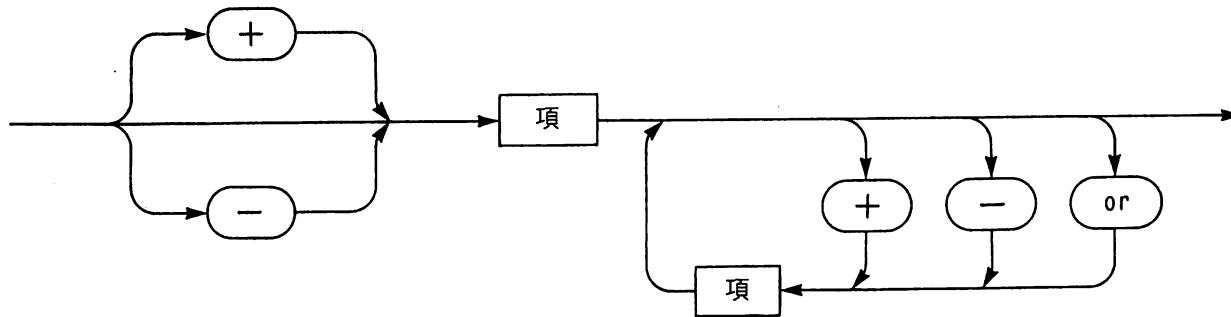
## 因 子



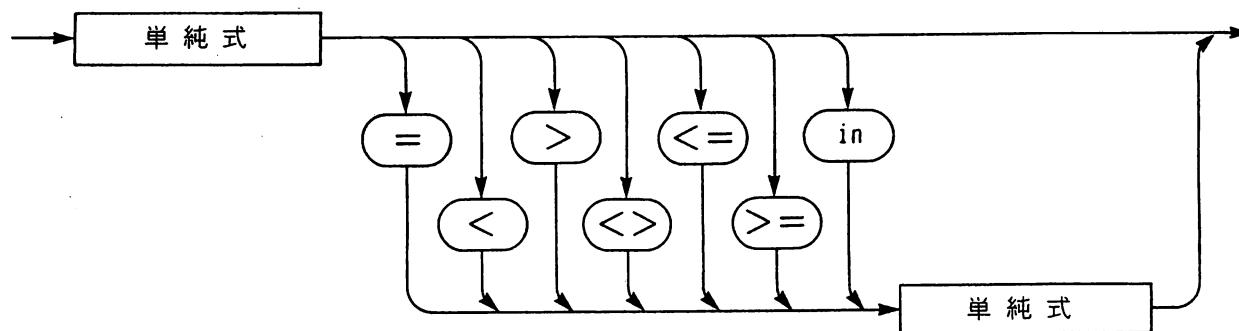
項



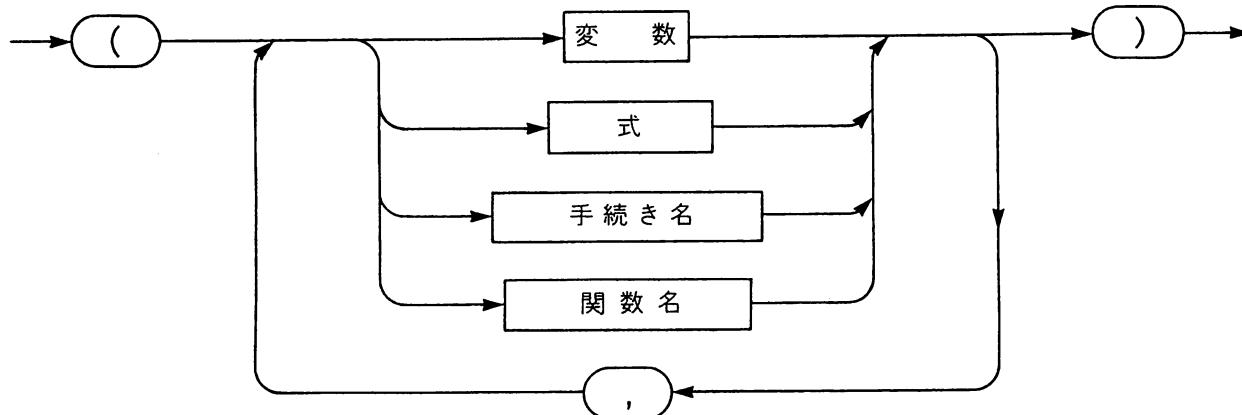
単純式



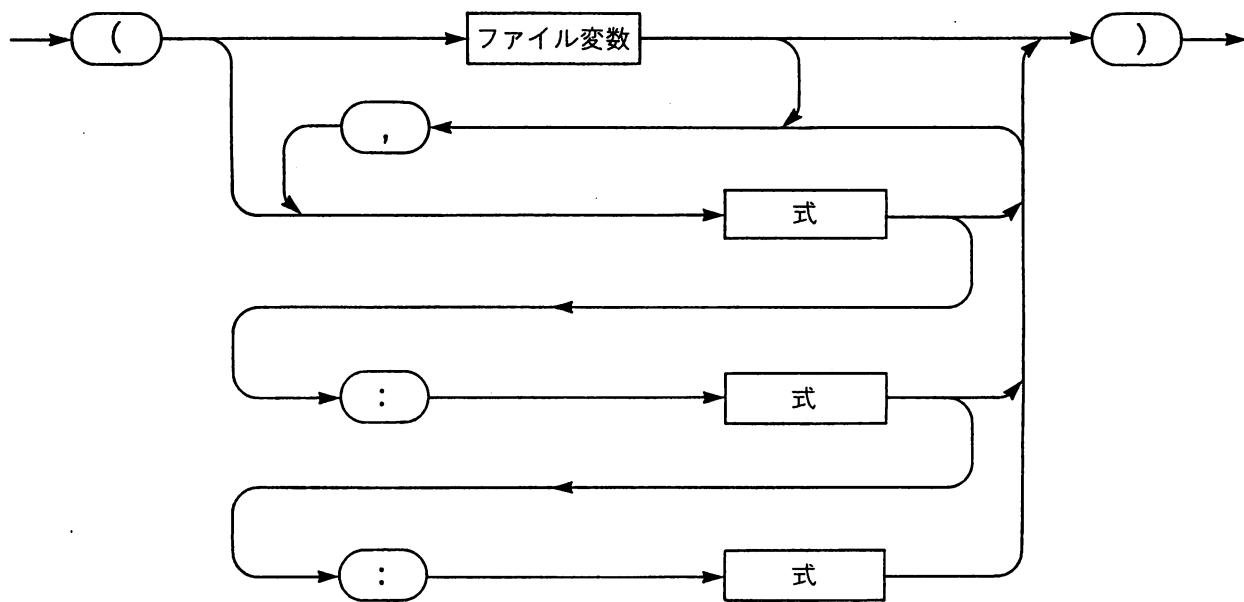
式



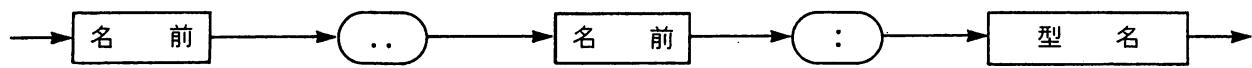
実パラメタ並び



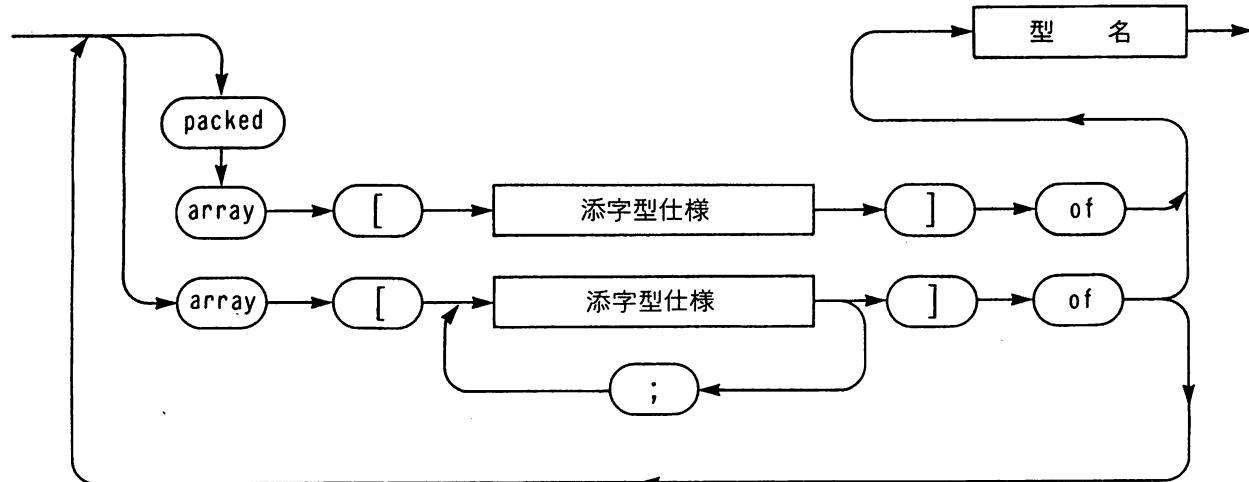
## 書出し/パラメタ並び



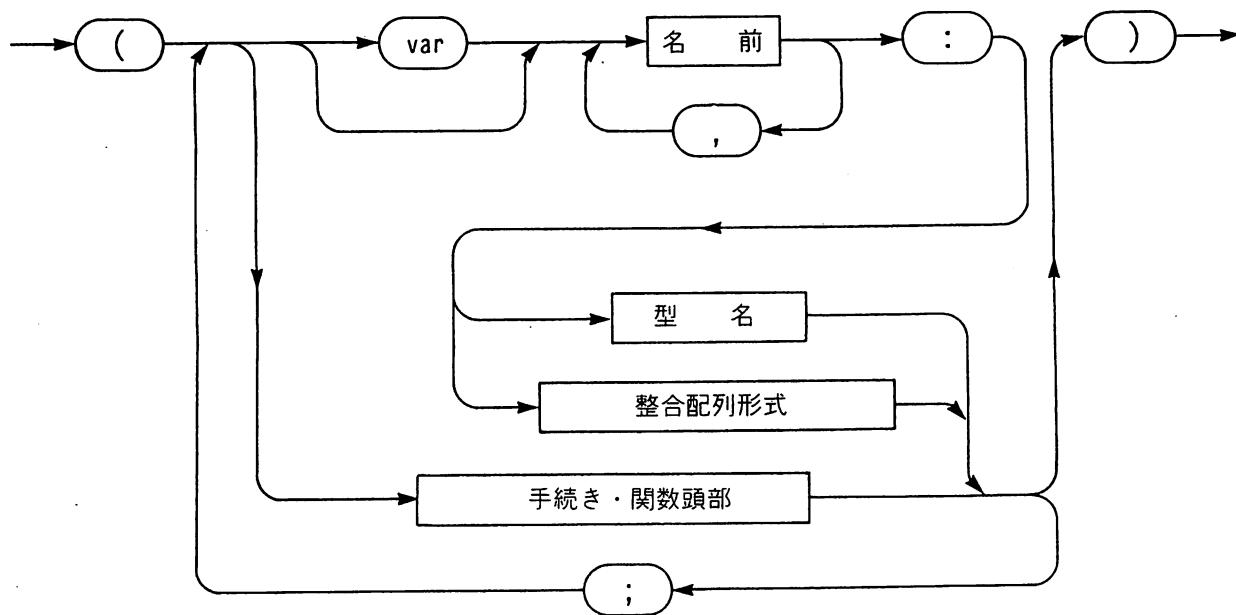
## 添字型仕様



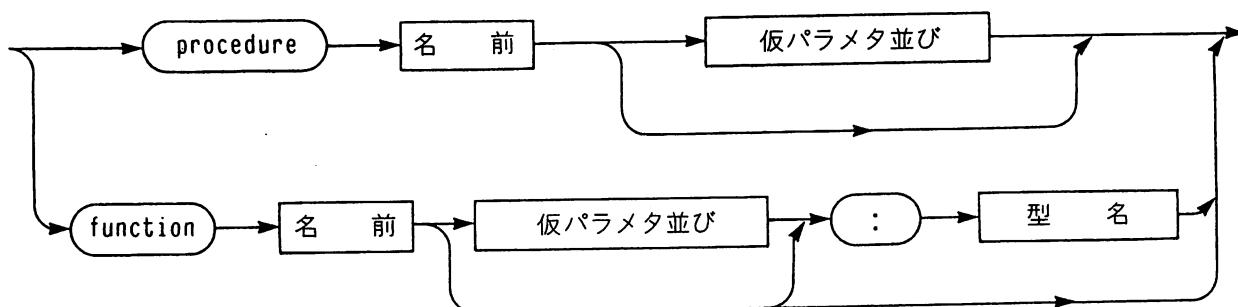
## 整合配列形式



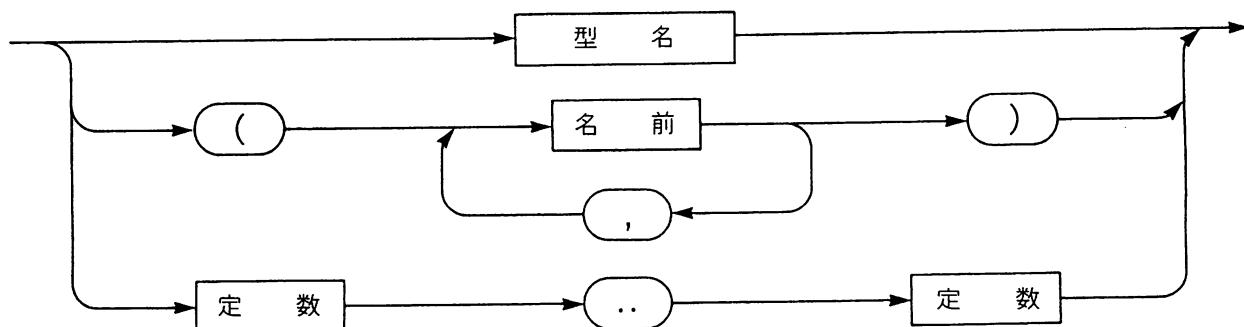
## 仮パラメタ並び



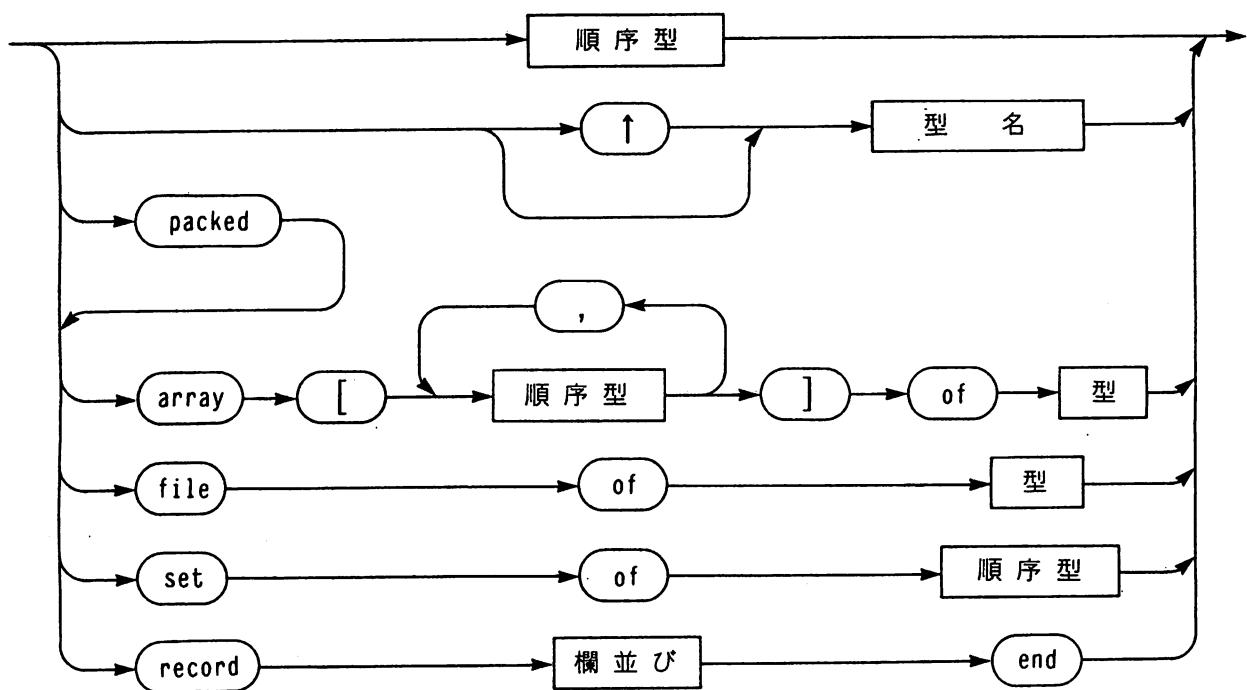
## 手続き・関数頭部



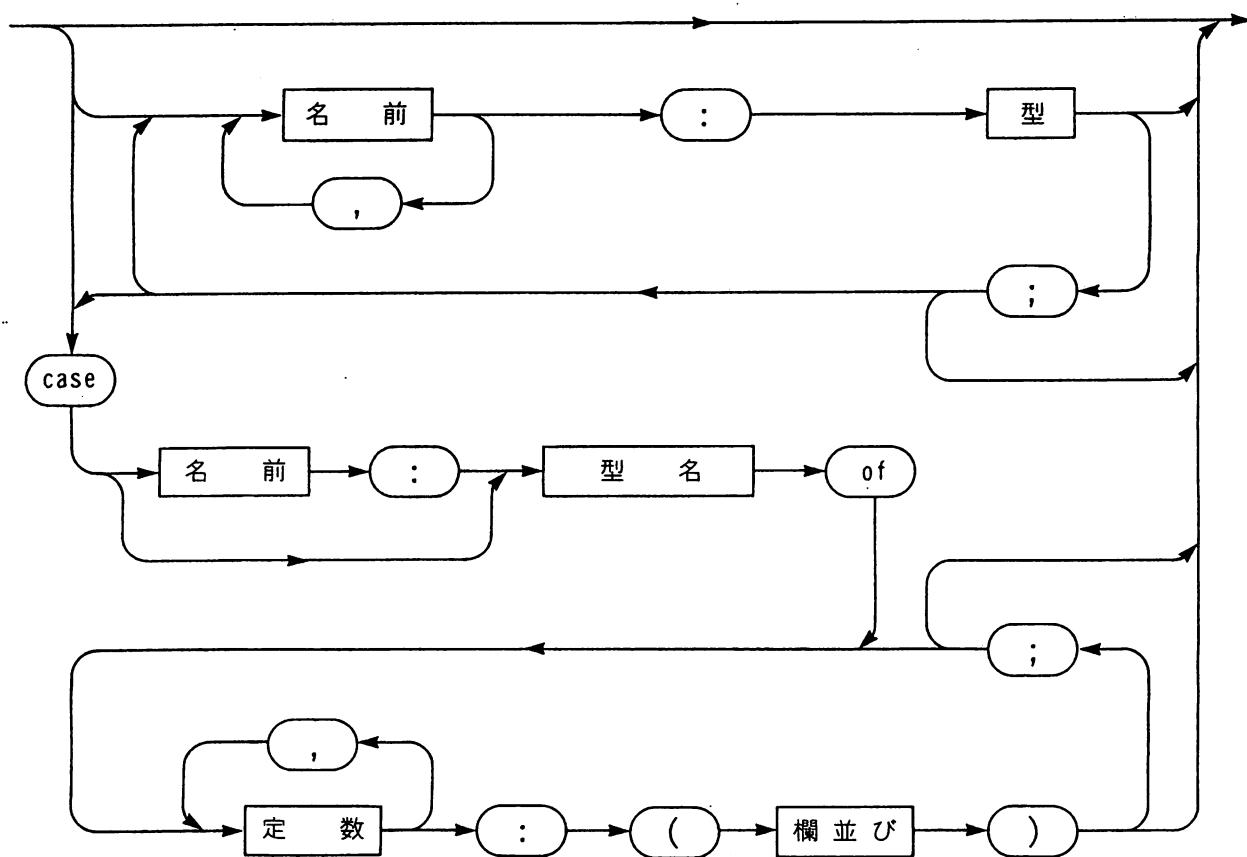
## 順序型



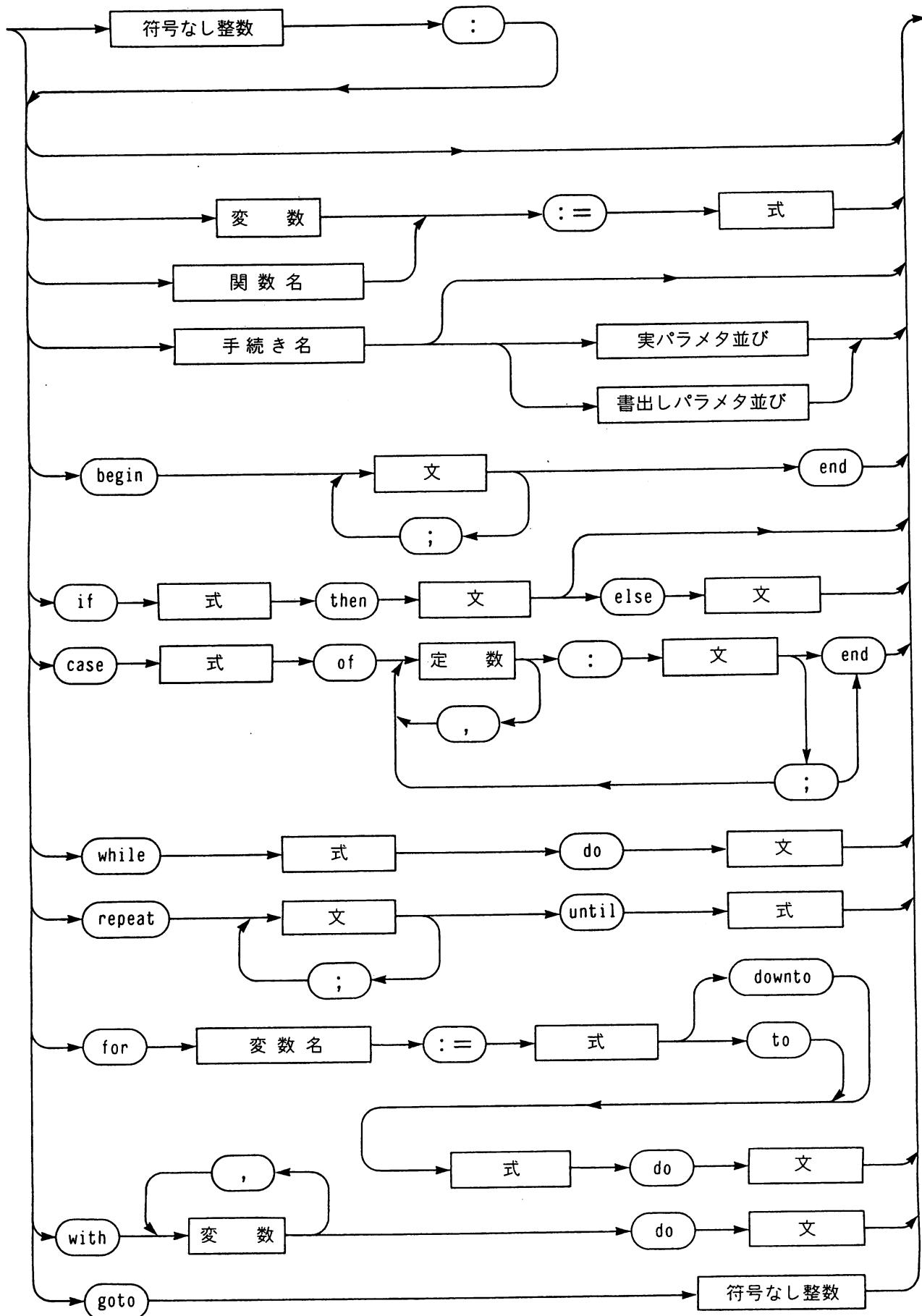
型



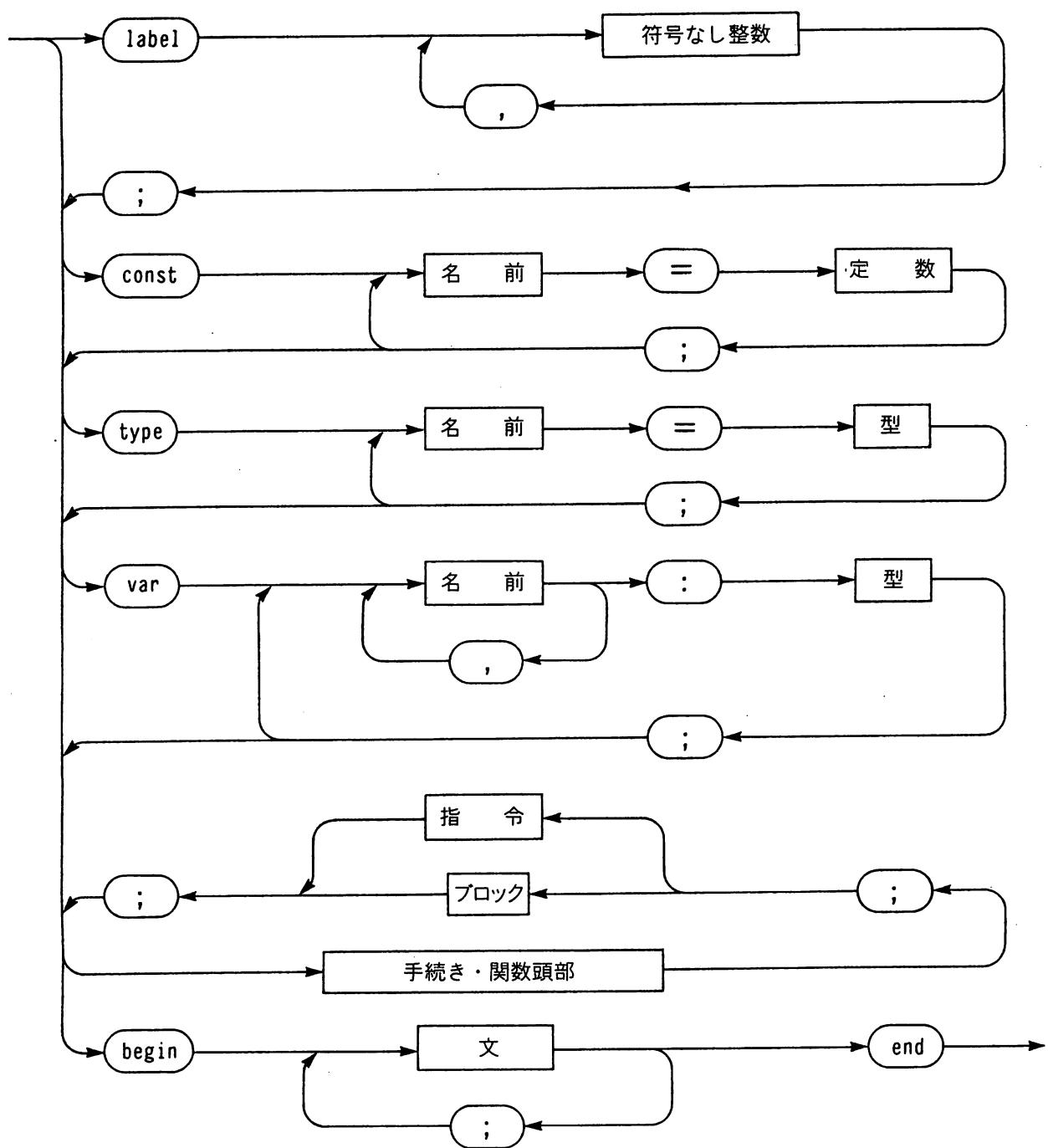
欄並び



文



## プロック



## プログラム

