III プログラム言語の文法(BNF)

1．言語記述構文の必要性

プログラムを記述した時、複数の人間、そしてコンピュータが理解できるために言語の構文記述には一定の規則を定義する必要がある。特定の言語のみでなく一般的な表現の場合メタ言語と呼ばれる。例えば英語では５構文に分類され、翻訳ソフトはこの構文を解析して日本語化している。一方、プログラム言語は BNF という表現で構文を定義している。

BNF とは Backus-Naur Form(バッカスナウアーフォーム)と呼ばれ、1965 年に Backs が ALGOL を簡潔に表現するために考案した記述法である。現在は言語記述の他 XML,HTML,データベースの定義にも利用されている。また、従来の BNF を拡張した ExtendedBNF が一般的であるがローカルルールも多い。本講義では、基本となる BNF について解説する。

2．BNF 構文

基本：

::=	左側に書かれた要素の構文を右側に定義する
<>	まぎらわしさを無くすための区切り
|	どちらかを選択する（選択）
[]	これで挟まれる要素は有ってもなくても良い（省略）
・・・	直前の要素を反復する
＊	ZERO 回以上の繰り返し
＋	１回以上の繰り返し
？	直前の要素があってもなくても良い
‘‘	文字や文字列そのものを表す

a)整数の表現

・１桁の正の整数数字 ERO 無し）

1～9 の数字を定義する

ZERO 無し数字 := ‘1’｜‘2’｜‘3’｜‘4’｜‘5’｜‘6’｜‘7’｜‘8’｜‘9’

・２桁の整数を表現

21,36,20 などの数字を定義する。まず 20 などを表現できるよう ZERO を含む数字を定義する。

数字 ::= ‘0’｜‘1’｜‘2’｜‘3’｜‘4’｜‘5’｜‘6’｜‘7’｜‘8’｜‘9’

これを利用して

２桁の整数 ::= ZERO 無し数字　数字

・正の整数

２桁の整数を利用すれば３桁あるいは４桁は

３桁の整数 ::= ZERO 無し数字　数字　数字

４桁の整数 ::= ZERO 無し数字　数字　数字　数字

で表現できるが.....桁数が増えると大変なので＊（ZERO 回以上の繰り返し）を利用する

正の整数 ::= ZERO 無し数字　数字＊

これで数字が省略できるので１桁も表現できる。

・符号付き整数

プラス記号’+’を付けた数字を表現する

正の整数 ::= ‘+’？ZERO 無し数字　数字＊

または

正の整数 ::= ＜ ‘+’　ZERO 無し数字　数字＊ ＞ ｜ ＜ ZERO 無し数字　数字＊ ＞

あるいは

正の整数 ::= [‘+’]　ZERO 無し数字　数字＊

このように、BNF での表現方法は複数存在する。

・負の整数

負の場合は必ず’ー‘が必要となる

負の整数 ::= ’ー‘　ZERO 無し数字　数字＊

・ZERO の定義

ZERO ::= ‘0’

・整数

最後に整数を定義する。

整数 ::= 正の整数　｜　ZERO　｜　負の整数

練習 1

C 言語における 16 進数整数を BNF で定義せよ。例えば　0x7f　など。

b)プログラム構造の表現

・if else 構造

if (条件式)

文

else

文

if else 構造 ::= ‘if(‘　条件式　‘)’　文 ‘else’　文

ただし、事前に 文 を定義しておく必要がある。

・if 構造

if (条件式)

文;

if 構造 ::= ‘if(‘　条件式　‘)’　文

・if 構文

一般的な if else と if 構造を定義する

if 構文 ::= ＜f(‘　条件式　‘)’　文　[‘else’　文]＞|＜‘if(‘　条件式　‘)’　文 ＞

または

if 構文 ::= ‘if(‘　条件式　‘)’　文　[‘else’　文]

となる。下の方が一般的。

練習 2

C 言語における while 構造および for 構造を BNF 法を用いて表現せよ

３．BNF 構文と構文木（解析木）

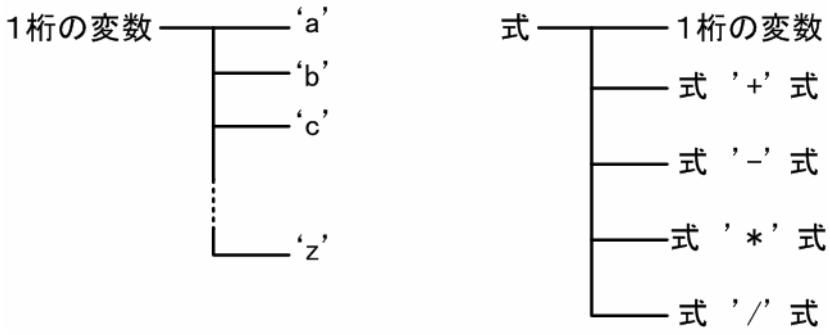
BNF の定義は構文木（ツリー構造）で表現できる。例えば、英文字や数字は以下のように示すことができる。

例えば１桁の変数名(小文字)と式は以下の BNF で表される。

１桁の変数 ::= ‘a’ | ‘b’ | ‘c’ | ‘d’ | ‘e’ | ‘f’ | ‘g’ | ‘h’ | ‘i’ | ‘j’ | 省略 | ‘y’ | ‘z’ |

式 ::= １桁の変数 ｜ 式 ‘×’ 式 ｜ 式 ‘＋’ 式 ｜ 式 ‘／’ 式 ｜ 式 ‘－’ 式

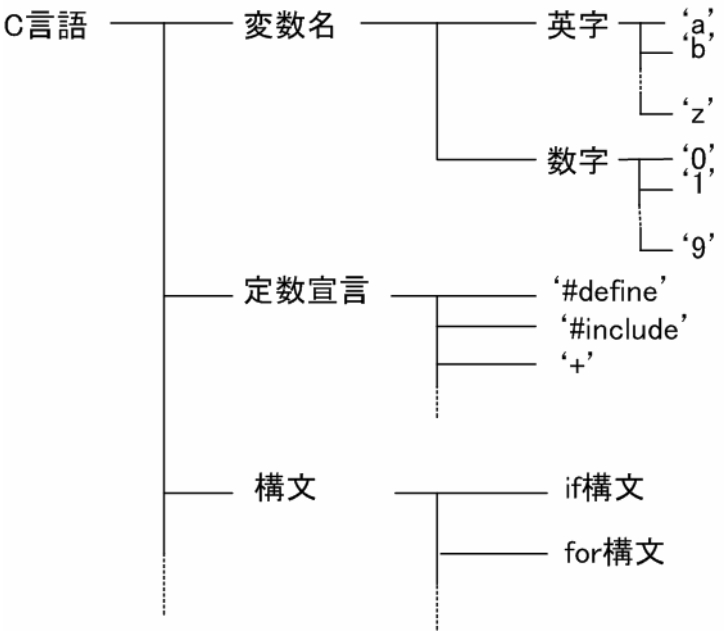
これは、構文木では



で表される。このことより、BNF 構文は構文木と互換があることがわかる。

ここで、BNF で表された式は、定義内に式が現れている。このことより、BNF の中で自信の定義が使えること、つまり再帰できることがわかる。

C 言語の構造で考えると



となる。繰り返し等は表現できないが、BNF を表現できることがわかる。ただし、ここに示した構文図は概念図であり、正式な C 言語の構文木ではない。

この図において、ツリーの末端に位置する記号を 終端記号（トークン:token）とよび、途中の記号を 非終端記号と呼ぶ。

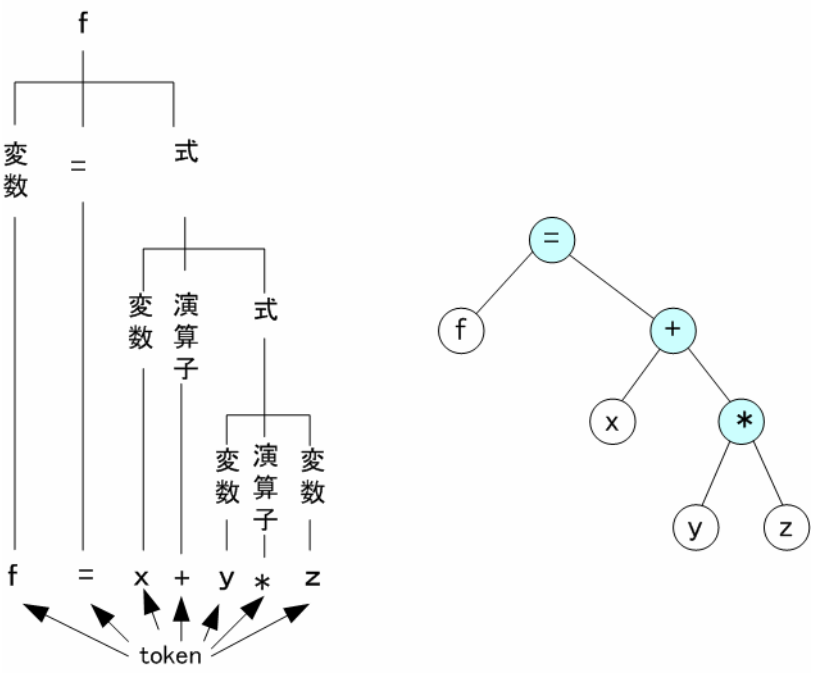
１桁の変数で表される式ｆ＝ｘ＊ｙ＋ｚを構文木で表すと

１桁の変数 ::= ‘a’ | ‘b’ | ‘c’ | ‘d’ | ‘e’ | ‘f’ | ‘g’ | ‘h’ | ‘i’ | ‘j’ | 省略 | ‘y’ | ‘z’

式 ::= １桁の変数 ｜ 式 ‘×’ 式 ｜ 式 ‘＋’ 式 ｜ 式 ‘／’ 式 ｜ 式 ‘－’ 式

計算式 ::= １桁の変数 ‘=’ 式

を用いて



構文木(解析木)

となる。

練習３

基本情報処理試験で出された関連問題を解け

問１正規表現 [A－Z] ＋ [0－9] ＊ が表現する文字列の集合の要素となるものはどれか。14S10Ｉ-1（エ）

ここで,

[A－Z] ：英字１文字を表す。

[0－9] ：数字１文字を表す。

＊： 直前の正規表現の０回以上の繰返しを表す。

＋： 直前の正規表現の１回以上の繰返しを表す。

ア 456789 イ ABC99＊ ウ ABC＋99 エ ABCDEF

問２ 数値に関する構文が次のとおり定義されているとき,<数値> として扱われるものはどれか。15S11Ｉ-1(イ)

<数値> ::＝<数字列>|<数字列>E<数字列>|<数字列>E<符号><数字列>

<数字列> ::＝<数字>|<数字列><数字>

<数字> ::＝0|1|2|3|4|5|6|7|8|9

<符号> ::＝＋|－

ア －12 イ 12E－10 ウ ＋12E－10 エ ＋12E10

参考

技術士１次試験（JABEE コース卒業時は免除）では以下のような問題が出されている。

プログラム言語の構文を形式的に記述するために用いられる BNF(Backus-Naur Forms)について、特定の言語での例を交えながら説明せよ。（平成１４年情報部門）