とき，すでに現れた **then** の中で，まだどの **else** とも組み合わされてないもので，その **else** に最も近いものと組み合わせる，という規則が使われることが多い．この規則に従えば，上記の if 文の解析木は図 3.3 の左側のものとなる．

3.5 PL/0′ の 文 法

コンパイラの作り方を理解するには，簡単な言語でもいいから，1 つの言語のコンパイラのプログラムを完全に理解するのがよい．そこで，本書では，簡単なプログラム言語を 1 つ取り上げ，そのコンパイラの完全なリストまで載せることにする．

その言語としては，参考文献 [Wirth 76] にある PL/0 をもとにして，それを少し変更したものを使う．大きな変更点は，パラメータなしの手続き（procedure）をパラメータ付きの関数（function）にした点である．ここでは，その文法と例題を説明する．

PL/0′ の構文規則を図 3.4～図 3.7 に示す．各構文規則について以下に簡単に説明する．

プログラムは 1 つのブロックの形をしており，最後はピリオドで終わる．

ブロックはいくつかの宣言の後に 1 つの文（statement）がある形である．

定数宣言では，定数の名前とその値を宣言する．変数宣言では，変数名だけを宣言する．定数も変数もその型は整数型である．ここで，太字の「**const**」は，プログラム中にそのとおりの綴りが現れることを意味する．太字でない「ident」は識別子（identifier）の意味で，何かの識別子（名前）が現れることを意味する．この ident については，「先頭が英字で，その後ろに英字または数字が 0 個以上付いているもの」という構文規則があるのであるが，それは常識であるとしてここにはあげてない．ここで，ident を非終端記号でなく，終端記号としてあるのは，第 2 章で述べたように，構文解析の前に字句解析が行われ，その字句解析の段階で ident は 1 つのトークンとしてまとめられてしまうからである．すなわち，「先頭が英字で，その後ろに英字または数字が 0 個以上付いているもの」は字句に関する規則であるとして，構文規則とは考えない．この字句に関する規則の書き方は次章で述べる．number が終端記号になっているのも同様の理由による．ここでは，number は数字の列であり，整数を表すものとする．

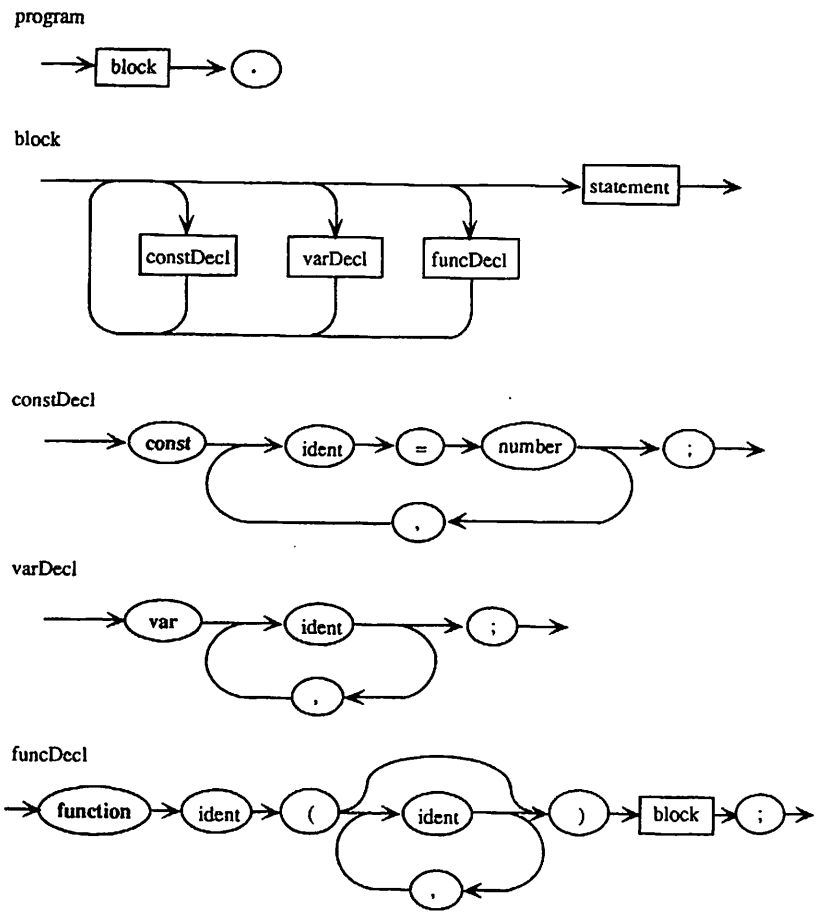

図 3.4 PL/0′ の構文規則 その(1) program と block

関数宣言では，関数名とパラメータ名を宣言し，関数本体はブロックの形で書く．パラメータがない場合も括弧を書く必要がある．関数は再帰的関数であってもよい．

文（statement）には，空文，代入文，文の列を begin と end で囲ったもの，if 文，while 文，return 文，write 文，writeln 文がある．return 文は関数の値を返す．write 文は式（expression）の値を出力するものであり，writeln 文では改行を出力する．

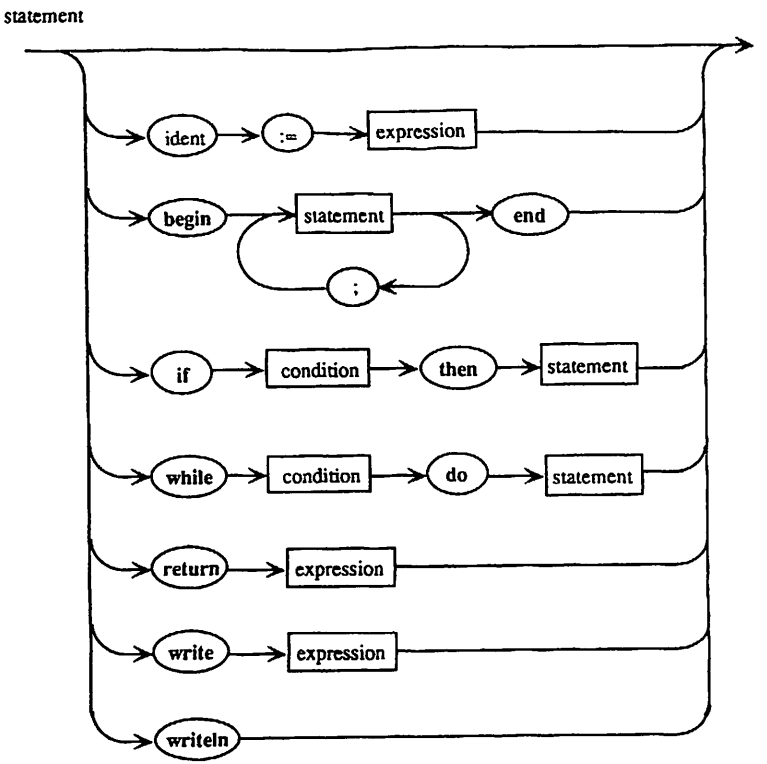

図 3.5 PL/0′ の構文規則 その(2) statement

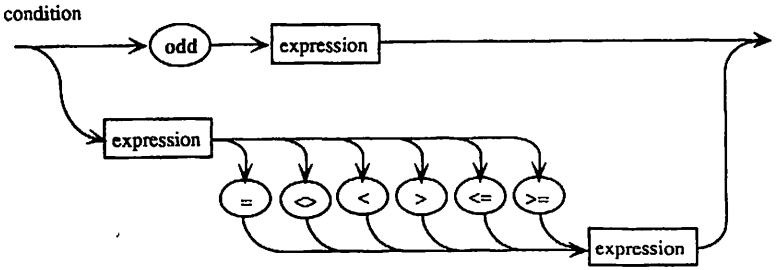

図 3.6 PL/0′ の構文規則 その(3) condition

条件 (condition) の中の「odd」は式の値が奇数であるときに真 (true) を返す.「<>」は不等号として使われる. factor の 3 番目にあるのは関数呼び出しの形である.

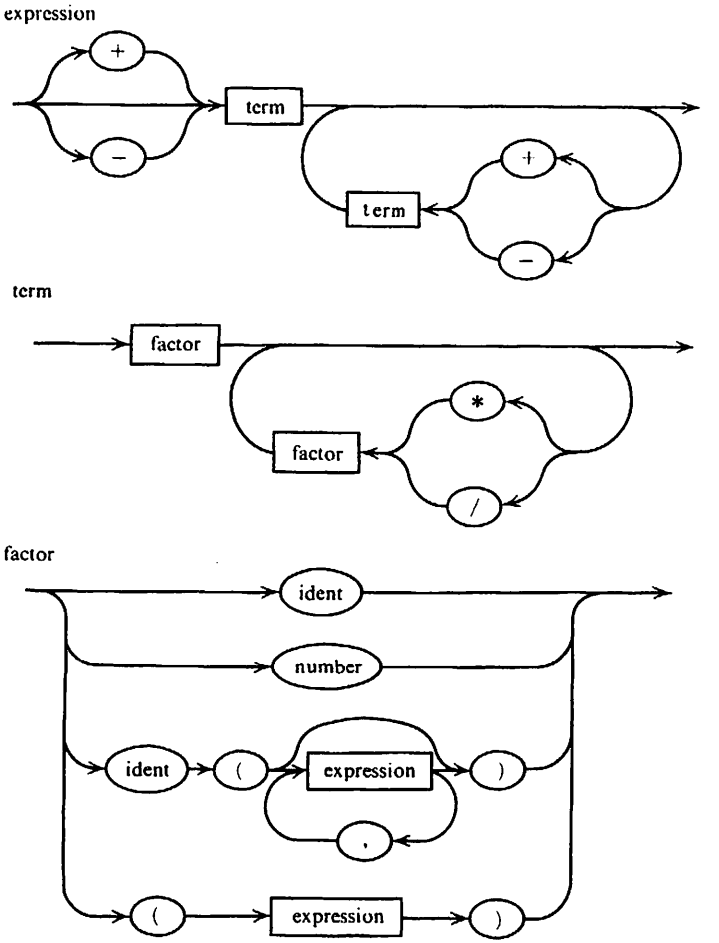

図 3.7 PL/0′ の構文規則 その(4) expression

PL/0′ 言語はブロック構造を持った言語である. すなわち, 宣言部分と実行部分を持ったブロックが入れ子構造になってもよい言語である (ブロックの中に変数宣言や関数宣言があり, 関数宣言の中にまたブロックがある). 一般にこのような構造を持つ言語では, ブロックの中で宣言された名前はそのブロックの中で有効であるが, それより内側のブロックで同じ名前がまた宣言された場合は, その後者の名前が有効な範囲では前者の名前は有効ではなくなる. PL/0′ 言語もその一般的な規則に従うものとする.

PL/0′ のプログラム例を図3.8 に示す．このプログラムは，参考文献 [Wirth 76] の PL/0 プログラムを PL/0′ に書き直してみたものである．関数 multiply と divide は，掛け算や割り算の命令を持たない計算機でそれをシフトと加減算で実行するアルゴリズムを模したものである．関数 gcd と gcd2 は，ユークリッドの互除法によって最大公約数を求めるものであり，gcd は再帰的関数である．このプログラムでは，変数の宣言などは，それを使う場所にできるだけ近いところに置くべきであると考えて，主ルーチンの変数宣言は関数宣言の後に置いてある．

```
function multiply(x,y)
   var a,b,c;
begin a := x; b := y; c := 0;
   while b > 0 do
   begin
      if odd b then c := c + a;
      a := 2*a; b := b/2
   end;
   return c
end ;

function divide(x,y)
   var r,q,w;
begin  r := x; q := 0; w := y;
   while w <= r do w := 2*w;
   while w > y do
      begin q := 2*q; w := w/2;
         if w <= r then
            begin r := r-w; q := q+1
            end
      end;
   return q
end;

function gcd(x,y)
begin
   if x <> y then
      begin if x<y then return gcd(x,y-x);
         return gcd(x-y,y)
      end;
   return x
end;

function gcd2(x,y)
begin
   while x <> y do
      begin if x<y then y:=y-x;
         if y<x then x:=x-y;
      end;
   return x
end;

const m = 7, n = 85;
var x,y;

begin
   x := m; y := n;
   write x; write y; write multiply(x,y); writeln;
   x := 84; y := 36;
   write x; write y; write gcd(x,y); write gcd2(x,y); writeln;
   write divide(x,y); write divide(x,gcd(x,y)); writeln
end.
```

図 3.8　PL/0′ のプログラム例

演 習 問 題

1．構文規則

S→(L)|a

L→S{,S}

について

（i） 各構文規則に対応する構文図式を書け．

（ii） S について 1 つにまとめた構文図式を書け．