SMAKY NEWS

No 12 30 octobre 1980

N A D I A : un "Tiny PASCAL"

INTRODUCTION

Le langage NADIA est structuré de la même manière qu'un programme PASCAL, mais ne fonctionne pas comme un subset de PASCAL.

Seuls les nombres entiers sont manipulés (16 bits). NADIA manipule les caractères ainsi que les chaînes de caractères et permet l'utilisation de variables.

Les paragraphes suivants présentent les procédures de base du langage NADIA. Les statements sont présentés par des diagrammes syntaxiques.

DIAGRAMMES SYNTAXIQUES

Les règles syntaxiques du langage NADIA sont définies sur les 4 planches annexes.

Un diagramme syntaxique se compose de 3 éléments principaux:

1) des rectangles indiquant les symboles 'non-terminaux', pour lesquels il existe un diagramme syntaxique
2) des cercles ou 'rectangles arrondis' indiquant les symboles 'terminaux', pour lesquels il n'existe pas de diagramme syntaxique
3) des flèches indiquant l'enchaînement des éléments, et ainsi la construction d'une phrase.

Exemple:

Tout programme Nadia commence par le diagramme syntaxique 'program' défini ci-dessous:

(PROG)---->[ident]---->(;)---->[bloc]---->(.)

Ce diagramme syntaxique appelle d'autres diagrammes, ce sont les diagrammes de 'ident' et 'bloc'. Puis le diagramme 'bloc' appellera le diagramme 'statement',...

Une phrase correcte serait:

```
PROG DEMO ; BEGIN WRITELN ( 'exemple de phrase' ) END.
```

MOTS RESERVES ET SYMBOLES SPECIAUX

Les 'mots réservés' de NADIA ne peuvent pas être utilisés comme identificateurs. Les identificateurs standards de NADIA sont confondus avec les mots réservés de NADIA.

```
and     array   assert  begin   call    case    close
const   div     do      downto  else    end     eof
eoln    exit    false   for     forward func    if
integer input   keyboard        loop    maxint  mem
mod     not     of      on      or      other   output
proc    prog    read    readln  repeat  reset   rewrite
shl     shr     size    text    then    time    to
true    type    until   var     while   write   writeln
```

Les 'symboles spéciaux' sont définis ci-dessous:

```
+     -     *
=     <>    <     <=    >=    >
(     )     [     ]     {     }
:=    .     ,     ;     :     '
--
```

IDENTIFICATEURS STANDARDS

Les 'identificateurs standards' de NADIA ne peuvent pas être redéfinis par le programmeur.

```
CONST
  MAXINT  = 32767 ;le plus grand entier positif
  FALSE   = 0     ;les tests booléens retournent
  TRUE    = 1     ;l'une ou l'autre de ces 2 valeurs

TYPE
  INTEGER = ens. des nbres entiers de -MAXINT à +MAXINT
  TEXT    = fichier de car. ;accès séquentiel par byte

VAR
  INPUT    : TEXT  ;entrée clavier avec écho
  OUTPUT   : TEXT  ;sortie display
  KEYBOARD : TEXT  ;entrée clavier sans écho
  MEM      : mémoire SMAKY, chaque byte est un él. du champ
             ; de 0 à (65535)-1
```

PROCEDURES ET FONCTIONS

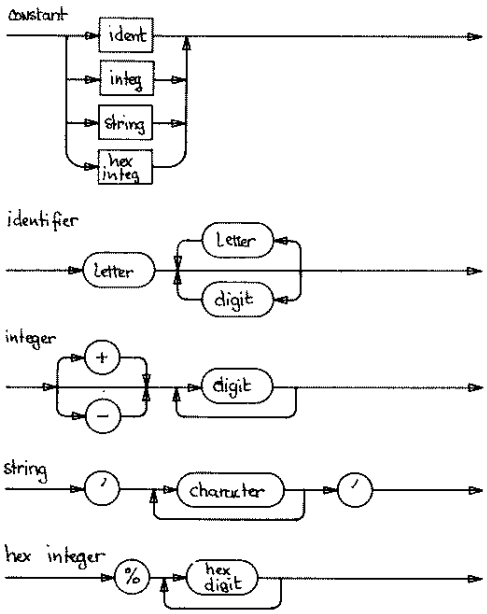
Les procédures

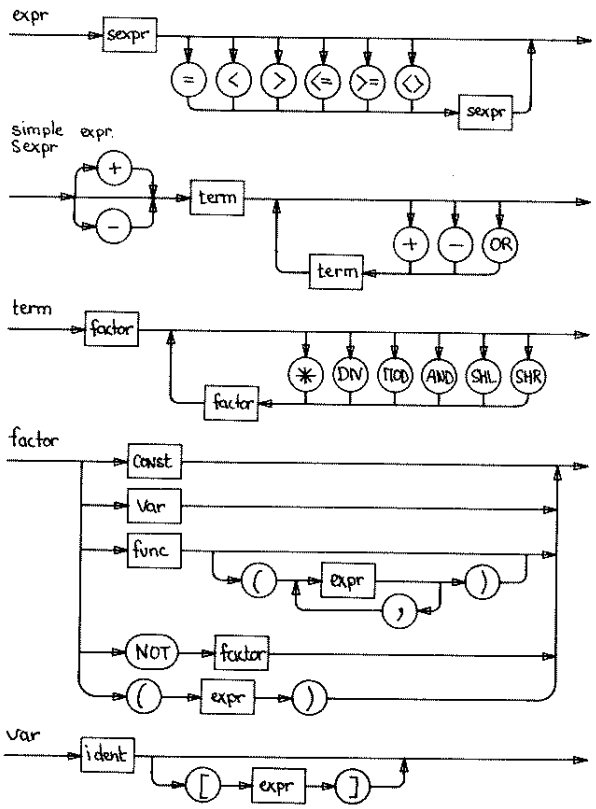
```
exit    call    reset   rewrite
close   read    write   concat
```

et les fonctions

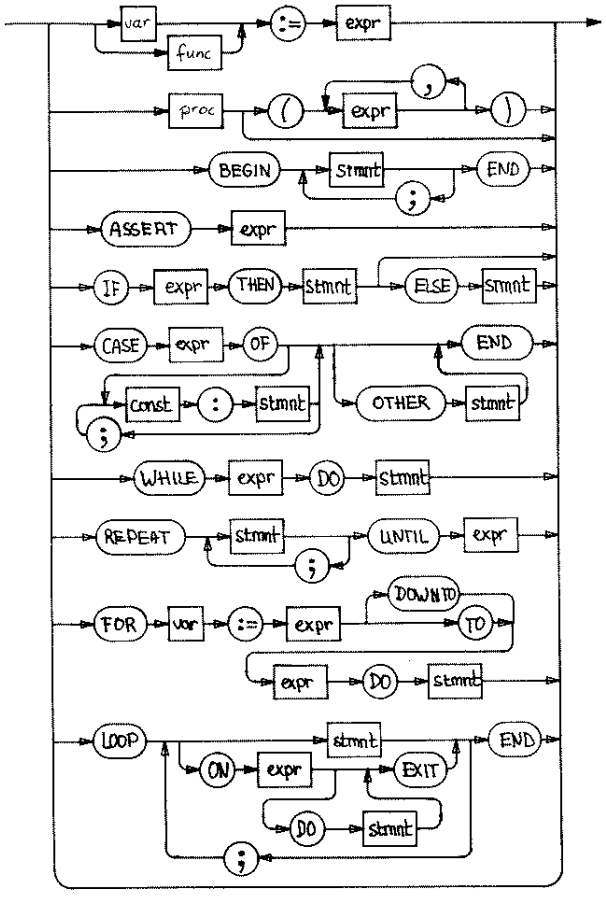
```
size    time    eof     eoln    search
```

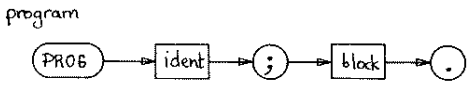
sont décrites dans la notice complète de NADIA.



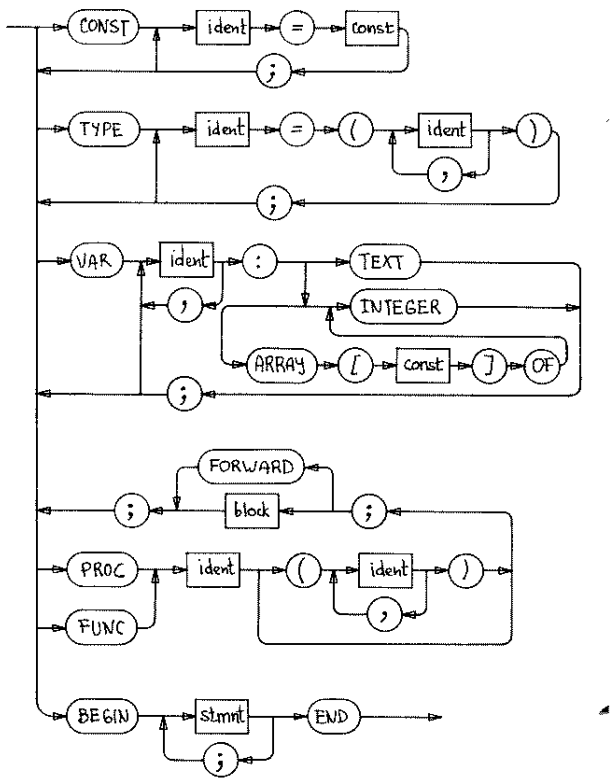


statement





block



NADIA: diagrammes syntaxiques.

ENTREE ET SORTIE SUR FICHIERS

Les entrées/sorties sont effectuées par le moyen de fichiers de type 'TEXT'. Le fichier TEXT est un fichier de caractères groupés en lignes. Chaque ligne est terminée par un caractère de fin de ligne. Ce caractère est <CR> dans le SMAKY.

Lorsque le pointeur d'un fichier 'TEXT' est positionné sur un caractère de fin de ligne, EOLN prend la valeur TRUE. NADIA diffère du PASCAL dans le sens que le caractère sous le pointeur du fichier n'est pas changé en <SP> mais reste <CR>. Cette différence permet à NADIA de lire un fichier de data quelconque par le biais d'un fichier TEXT.

Les fichiers que l'utilisateur désire manipuler ne peuvent pas être transmis comme paramètres de programme. Ils sont toujours locaux au programme. Trois fichiers globaux sont offerts au programmeur, ce sont:

- INPUT
- OUTPUT
- KEYBOARD.

Ces 3 fichiers sont ouverts pour l'exécution de chaque programme. Le fichier 'KEYBOARD' est identique au fichier 'INPUT' car il utilise le clavier mais il ne génère pas l'écho sur le display.

Les fichiers 'INPUT' et 'KEYBOARD' sont normalement interactifs car il y a un opérateur humain. Le fichier 'TEXT' n'est pas intéractif (déf. PASCAL). NADIA contourne la difficulté de la manière suivante:

1) la lecture de 1 caractère ne provoque pas la recherche du suivant. Le caractère est retourné sur le display selon le type de fichier
2) l'appel de la fonction EOLN demande la lecture de 1 caractère afin que la fonction puisse retourner une valeur en accord avec la déf. PASCAL. Le caractère lu n'est pas retourné sur le display mais mis en mémoire. Il sera ensuite lu et retourné sur le display ou testé
3) l'appel de la fonction EOF provoque un mécanisme identique à la fonction EOLN. Le 'end-of-file' du clavier est généré par la touche <DEFINE>

READ,READLN,WRITE,WRITELN

Dans ce paragraphe, t indique un fichier TEXT, et v, v1, v2, ... indiquent des variables.

READ :

READ (v1,v2,...) signifie READ (INPUT,v1,v2,...)

READ (t,v1,v2,...) signifie BEGIN READ (t,v1) , READ (t,v2) ... END.

Si un nombre est lu, les blancs et fins de ligne sont sautés jusqu'à ce qu'un nombre (peut-être signé) soit reconnu.

READLN:

READLN (v1,v2,...) signifie READLN (INPUT,v1,v2,...)

READLN (t,v1,v2,...) signifie
BEGIN READ (t,v1) ; READ (t,v2) ... ; READLN (t) END

L'effet de READLN (t) est d'avancer le pointeur du fichier t sur le début d'une nouvelle ligne.

Dans le peragraphe suivant, p1, p2,... indiquent des parametres d'écriture. Un paramètre d'écriture a une des 2 formes suivantes:

e e:w

avec e qui est un nombre, un caractère ou un string (voir chapitre suivant) et w une expression. L'effet de w est d'écrire la représentation du paramètre précédée de blancs de façon à assurer que w caractères soient écrits.

WRITE:

WRITE(p1,p2,...) signifie WRITE(OUTPUT,p1,p2,...)

WRITE(t,p1,p2,...) signifie
BEGIN WRITE(t,p1);WRITE(t,p2) ... END.

WRITELN:

WRITELN(p1,p2,...) signifie WRITELN(OUTPUT,p1,p2,...).

WRITELN(t,p1,p2,...) signifie
BEGIN WRITELN(t,p1);WRITELN(t,p2) ... ;WRITELN(t) END.

L'effet de WRITELN(t) est d'écrire une fin de ligne sur t.

NOMBRES, CARACTERES ET STRINGS

Les types de variables de calcul sont très limitées dans NADIA. Le type 'CHAR' (déf. PASCAL) n'existe pas ainsi que le type 'STRING' (déf. PASCAL-UCSD). Le type 'ARRAY [x..y] OF z' est restreint à 'ARRAY [0..n] OF INTEGER'.

La lecture ou l'écriture d'un nombre entier est en accord avec PASCAL.

Exemple:

```
VAR I:INTEGER;
BEGIN
  READ ( I ); WRITELN ( 'I+I=' , I+I )
END.
```

La lecture ou l'écriture d'un caractère est obtenue en faisant suivre une variable ou une expression du caractère '$'.

Exemple:

```
VAR C:INTEGER;
BEGIN
  READ ( C$ ); WRITE ( C$ ) { joue au
perroquet }
END.
```

La lecture ou l'écriture d'un 'string' sont obtenues en faisant suivre une variable par le caractère '$'. Le nombre de caractere du 'string' est rangé dans l'élément 0, la 1ere lettre dans l'élément 1 et ainsi de suite. La lecture d'un string est terminée par un caractère <CR>. Le caractère <BS> permet d'effacer le caractere précédent.

Exemple:

```
VAR S:ARRAY [10] OF INTEGER;
BEGIN
  READLN ( S$ ); le car. <CR> est absorbe
  WRITELN ( 'il y a ',S[0],'caractères dans
''' , S$ , ''' )
END.
```

INTERPRETEUR ET P-CODE

L'interpréteur P-Code est une 'machine' à utilisation de pile avec 4 registres et 2 zones de mémoire. La mémoire est répartie en mémoire de programme et en mémoire de donnée. La mémoire de programme contient le code du programme (P-Code), et demeure inchangée lors de l'exécution du programme. La mémoire de donnée contient les variables du programme et les valeurs temporaires (calcul,...).

Les 4 registres de la 'machine P' sont:

1) le compteur de programme, P, qui pointe la prochaine instruction P-Code exécutable
2) le registre d'instruction, I, qui pointe l'instruction P-Code exécutée
3) le pointeur de pile, T, qui pointe le sommet de la pile
4) le registre de base, B, qui contient l'adresse de la base courante des variables.

L'activation d'une procédure genere les informations du lien statique, du lien dynamique et de l'adresse de retour sur la pile. Un exemple est donné dans la notice détaillée.

REFERENCES

A "Tiny" Pascal Compiler, Kin-Man Chung & Herbert Yuen, tiré à part de la revue BYTE, 1979
PASCAL: An Introduction to Methodical Programming, W. Findlay & DA Watt, Pitman, 2nd edition 1979
Le langage PASCAL, N. Magnenat-Thalmann et al., Gaëtan Morin, 1979

EXEMPLES DE PROGRAMMES

L'édition se fait avec SMILE ou EPRO.

```
prog pgcd;      -- plus grand commun diviseur
var a,b,d,e,f:integer;

begin
  writeln('ce programme calcule le PGCD de 2 nombres A et B');
  writeln; writeln('svp A)B'); writeln;

  write('donner A '); read(a); writeln;
  write('donner B '); read(b); writeln;

  assert a)b;   -- vérifie a)b

  e:=a; f:=b;

  repeat
    d:=a-b;
    if d(b then begin a:=b; b:=d end else a:=d
  until a=b;

  writeln; writeln('le PGCD de ',e,' et ',f,' vaut ',a)
end.
```

```
prog horloge;
var t:integer;

begin
  writeln('ce programme affiche le temps pendant 10 sec');
  writeln;
  write('temps: 0.0'); t:=0;

  repeat
    while t=time do ;
    t:=time;
    write(8$,8$,8$,8$,t div 10:2,',',t mod 10)
  until t)=100;

  writeln; writeln('comptage terminé')
end.
```

MODE D'EMPLOI

1) Appeler l'interpréteur INADIA/
2) Appeler le compilateur CNADIA.PA/
3) Le programme demande *code (sinon texte)? O(oui N(*
4) Répondre O pour créer du code
5) Le programme demande le nom du fichier en entrée *in:*
6) Répondre par exemple PGCD.PS/
7) Le programme demande le nom du fichier en sortie *ou:*
8) Répondre par exemple PGCD.PA/
9) La compilation s'effectue en affichant des renseignements sur l'écran
10) Pour exécuter le programme, taper ensuite PGCD.PA/
11) Pour quitter NADIA, donner la commande ./

```
prog poème;

var i,j,k,n:integer;
    s:array [25] of integer;
    s1,s2,s3:array [200] of integer;

begin

  concat(s1$,
       'le baladin          ',
       'l''automne malade    ',
       'la rose              ',
       'une silhouette grise',
       'l''heure             ');
  concat(s2$,
       's''éloigne',
       'meurt    ',
       'fleurit  ',
       's''en va  ',
       'passe    ');
  concat(s3$,
       'le long des jardins ',
       'quand la neige tombe',
       'dans ma mémoire     ',
       'dans le brouillard  ',
       'si lentement        ');

  n:=0;

  loop
    i:=n mod 5; j:=n div 5 mod 5; k:=n div 25 mod 5;
    concat(s$,s1$:20*i+20:20); write(s$,' ');
    concat(s$,s2$:9 *j+9 :9 ); write(s$,' ');
    concat(s$,s3$:20*k+20:20); writeln(s$);
    n:=n+1;
    on n=125 exit
  end

end.
```

J.M. Paratte

SMAKY NEWS

ADRESSEZ VOS COMMUNICATIONS A:

EPSITEC-system sa
Chemin de la Mouette, CH - 1092 Belmont