UIT Report Vol. 15. No. 2. pp. 309~327. 1984 연구논문집 제15권 제 2 호 pp. 309~327. 1984

A Realization of the Documented Pascal-S System

Lee, Myung-Joon
Department of Computer Science U. I. T.
(Received April 30, 1984)

〈**Abstract**〉

This paper presents a generalized approach to a documented system in interactive computer systems. We decribe the software system, named UDPS (U. I. T. Documented Pascal-S System), which is a powerful tool for educating the structure of a compiler and providing environments to research problems in compiler area.

UDPS internally consists of three parts: monitor, documents, and Pascal-S system. This system is currently implemented on PRIME 750 system and will be referenced usefully for the the compiler-course.

문서화 된 Pascal-S 컴파일러의 실현

이 명 준
전 자 계 산 학 과
(1984. 4. 30 접수)

〈요 약〉

본 논문은 대화식 컴퓨터 시스템에서의 문서화 된 소프트웨어 시스템에 대한 일반적인 접근 방법을 통하여 개발된 UDPS(U. I. T. Documented Pascal-S System)의 구조를 보여주고 있다. UDPS는 컴파일러의 구조에 대한 교육과 컴파일러 분야에 있어서의 여러 연구에 도움을 줄 수 있도록 설계되었다. 본 시스템은 크게 모니터, 문서, Pascal-S 시스템의 세 부분으로 이루어져 있으며, 현재 PRIME 750 시스템에 구현되어 있다. 본 시스템은 앞으로 컴파일러의 교육에 유용하게 사용될 예정이다.

I. Introduction

An important facility in interactive computer systems is the ability for the user to learn a system through interactive processes. The term 'documented system' will be used in a generalized sense to denote a program supporting this process. Since a compiler system has a complex structure and many implementation-oriented features, it needs to be a documented system for the purpose of education. But most compilers are not documented systems since they are designed for commercial purposes. This fact motivated the author to realize a documented compiler system.

For this reason, UDPS (U. I. T. Documented Pascal-S System) is implemented on PRIME 750 system. This system internally consists of three parts: monitor, documents, and Pascal-S system. The external structure of this system is a tree structure. This paper will present the description of this system's external structure, and then present the description of this

system's internal structure.

II. Structure of UDPS

The external structure of the system is the structure of the system viewed from user's standpoint, and the internal structure of the system is the structure of system viewed from system designer's standpoint. The behavior of this system can be viewed as Fig-1.

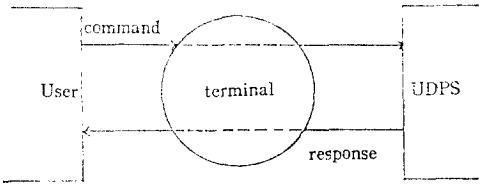


Fig. 1

The external structure and the internal structure of UDPS are as follows.

1. External Structure of UDPS

The external structure of UDPS is a tree structure. Fig-2 represents this tree structure. The tree has three types of nodes: tree nodes, document nodes and execution nodes. A document node is a sequence of an arbitrary number of characters which presents useful documents to the user. A tree node is represented as a list containing a head and a sequence of zero or more references to other nodes, its subnodes. An execution node is a special node which make it possible to use Pascal-S compiler-interpreter system.

UDPS
- 1. HELP
- 2. GLOBAL DOCUMENT
  - 1. COMPILER
    - 1. SYNTAX
    - 2. MODULE HIERARCHY
    - 3. PARSING METHOD
    - 4. CODEGEN RULES
    - 5. TABLE INFORMATION
    - 6. ERROR CODES
  - 2. INTERPRETER
    - 1. STACK MACHINE STRUCTURE
    - 2. POST-MORTTEM DUMP
    - 3. CODE INTERPRETATION
- 3. MODULE DOCUMENT
  - 1. PROGRAM LEVEL
    - 1. MAIN ROUTINE
    - 2. ERROR MESSAGE PROCEDURES
    - 3. TABLE MANAGEMENT PROCEDURES
    - 4. CODE GENERATION PROCEDURES
    - 5. LEXICAL ANALYSYS PROCEDURES
    - 6. DECLARATION AND INITIALIZATION
  - 2. BLOCK LEVEL
    - 1. BLOCK ROUTINE
    - 2. PROCEDURES FOR DECLARATIONS
    - 3. TABLE MANAGEMENT AND ERROR RECOVERY
    - 4. TYPE PROCESSING PROCEDURES
    - 5. PARAMETER PROCESSING PROCEDURES
    - 6. SPECIAL PROCEDURES.
  - 3. STATEMENT LEVEL
    - 1. STATEMENT ROUTINE
    - 2. ASSIGNMENT, COMPOUND, IF STATEMENT
    - 3. CASE STATEMENT
    - 4. REPEAT, WHILE, FOR STATEMENT
    - 5. STANDARD PROCEDURES
    - 6. SPECIAL PROCEDURES
  - 4. EXPRESSION LEVEL
    - 1. EXPRESSION
    - 2. SIMPLE EXPRESSION
    - 3. TERM
    - 4. FACTOR
- 4. EXECUTION ENVIRONMENT
  - 1. EXECUTION
  - 2. POSTEXECUTION
    - 1. SOURCE FILE
    - 2. INPUT FILE
    - 3. OUTPUT FILE
    - 4. LISTING FILE

Fig. 2

```
**********************************
*   WELCOME TO UDPS !            *
*      LEE MYUNG JOON            *
*      U.I.T DEPT. OF COMPUTER SCIENCE
*        SELECT YOUR OPTION.     *
**********************************
  1. HELP
  2. GLOBAL DOCUMENT
  3. MODULE DOCUMENT
  4. EXECUTION ENVIRONMENT
.....TYPE YOUR OPTION NUMBER !
     IF YOU DO NOT HAVE ANY INFORMATION ON THIS SYSTEM,
     SELECT OPTION-1. ..................................
>1
```

Fig. 3

```
.....................................................................
*** 1. HELP
.....................................................................
..... THIS UDPS (U.I.T. DOCUMENTED PASCAL-S SYSTEM) IS
        THE SYSTEM FOR TEACHING COMPILER-INTERPRETER SYSTEM.
..... YOU CAN HAVE THE OVERALL DOCUMENTATION OF THE SYSTEM.
        ALSO YOU YOU CAN GET THE SOURCE PROGRAM FOR EACH MODULE.
  MODEL COMPILER: PASCAL-S
  SYSTEM COMMANDS ARE
        N: SELECTED OPTION NUMBER
        F: DISPLAY NEXT PAGE OF THE SPECIFIED DOCUMENT
        B: DISPLAY PREVIOUS PAGE OF THE SPECIFIED DOCUMENT
        T: DISPLAY TOP PAGE OF THE SPECIFIED DOCUMENT
        R: RETURN TO UPPER SYSTEM LEVEL
        Q: EXIT TO PRIMOS
**********************************************************************
.....!!! TYPE COMMAND ('R' OR 'Q').
>R
```

Fig 4

The terminal screen is used to display the structure of a tree or the docment in a node. If the current node is a tree node, the head lines and the name of each subnode is showed. The subnods are numbered and these numbers can be used as commands. If current node is a document node, the first page of the specified document is displayed. If the user wants more pages, they can be displayed by specifying appropriate commands. Fig-3 shows an example of screen layout and Fig-4 shows the screen layout when a subnode is selected.

Tree traversal is done by menu selection. An integer as a command selects the corresponding subnode as current node. When a document node has many pages, it is not possible to show all the pages. In order to see the next page, the system command 'F' is used. Only a few commands are needed to make it easy for the user to walk around in a tree and get a general view of a structured document. There are commands to select the nth subnode (integer n>0), go one level up (R), go the next page (F), the previous page

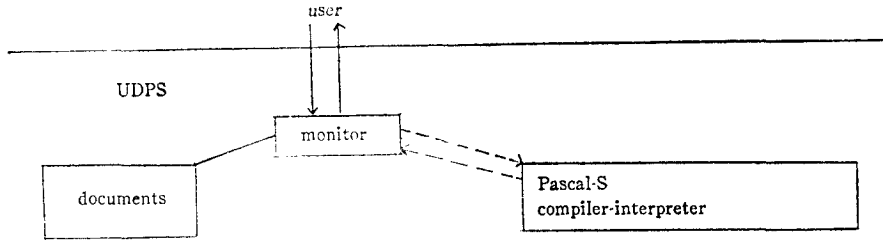


Fig. 5

(B) and the top page(T) of the specified document, and exit to operating system(Q).

2. Internal Structure of UDPS

Internally UDPS consists of three parts: monitor, documents, and Pascal-S compiler-interpreter system. Fig-5 shows the internal structure of UDPS.

The monitor is a program which reads a command from keyboard and process this command repeatedly until the exit command encounters. This is the main control routine of UDPS. This monitor can be defined by a finite tranducer which is obtained by taking a finite automaton and permitting the machine to emit a string of output symbols on each move. This finite transducer is defined by $M=(Q, \Sigma, \Delta, \delta, q_0, F)$, where (1) Q is a finite set of states which is defined by Monitor-status-words. (2) Σ is a finite input alphabet which consists of characters from keyboard. (3) Δ is a finite output alphabet which defines the character set of monitor's response. (4) δ is a mapping from $Q\times(\Sigma\cup\{\Delta\})$ to finite subsets of $Q\times\Delta^*$ which defines the monitor's progress. (5) $q_0\subseteq Q$ is the initial state of the monitor. (6) $F\subseteq Q$ is the set of final states.

The principal algorithm of the monitor is

```
procedure monitor;
begin initialize MSW and auxiliary variables
      while MSW [0] <> exit do
      begin readcommand;
            processcommand;
      end
end;
```

3. Pascal-S System

The Pascal-S system consists of two main parts: compiler and interpreter. The compiler part translates Pascal-S programs into codes for a hypothetical stack computer especially designed for this purpose. This computer is itself defined as an algorithm, called the interpreter of the compiled code. The interpreter describes a straightforward stack computer, consisting of a store S organized as a stack, two index resiters T and B which control the stack, a program counter PC, an instruction resister IR, a program status register PS and a DISPLAY used to speed up the addressing mechanism.

The principal algorithm is

```
procedure interpret;
      begin initialize registers and auxialiary
      variables
        repeat IR:=code[PC]; PC:=PC+1;
              interpret IR
      until ps<>run;
      if ps<>fin then postmortemdump;
end;
```

The Pascal-S system is introduced by ref [3]. In order to present more useful information to the user, this Pascal-S system is implemented with modifications and extensions on PRIME 750 system. The syntax of Pascal-S is defined in APPENDIX-2. Fig-6 shows the example use of Pascal-S system.

```
.....
**** 4. EXECUTION ENVIRONMENT ****
.....

..... YOU CAN EXECUTE
          PASCAL-S COMPILER-INTERPRETER SYSTEM.
    1. EXECUTION
    2. POST-EXECUTION
**********************************************************
..... TYPE OPTION NUMBERS!
>1
.....
**** 4.1 EXECUTION *****
.....

    YOU CAN GET COMPILE-TIME INFORMATION AND
       RUN-TIME INFORMATION BY SPECIFYING
OPTION-1: SYSTEM-RESERVED TABLE
OPTION-2: IDENTIFIER TABLE
OTTION-3: BLOCK TABLE
OPTION-4: ARRAY TABLE
OPTION-5: STRING TABLE
OPTION-6: REAL CONSTANT TABLE
OPTION-7: CODES FOR EACH STATEMEMT
OPTION-8: CODES FOR TOTAL PROGRAM
OPTION-9: CODE EXECUTION SEQUENCE
**********************************************************
.....SELECT YOUR OPTIONS!
          TYPE 'Y' OR 'N'.
>DO YOU SELECT OPTION-1: N
>DO YOU SELECT OPTION-2: N
>DO YOU SELECT OPTION-3: N
>DO YOU SELECT OPTION-4: N
>DO YOU SELECT OPTION-5: N
>DO YOU SELECT OPTION-6: N
>DO YOU SELECT OPTION-7: N
>DO YOU SELECT OPTION-8: N
>DO YOU SELECT OPTION-9: N
>SOURCE FILE NAME: SAMPLE
>INPUT DATA FILE NAME: D-SAMPLE
>OUTPUT DATA FILE NAME: O-SAMPLE
>LISTING FILE NAME: L-SAMPLE
.....! EXECECUTION BEGINS...
.....! EXECUTION ENDS     ...
```

Fig. 6

Ⅲ. Conclusion

UDPS was designed as a prototype for a documented system providing environments for educating the structure of a compiler and for researching problems in compiler area. The significance of this paper lies in the fact that the system developed is directly applicable to teaching the structure of a compiler and that many problems of the compiler area can be researched in virtue of the presence of a

compiler source. But Pascal-S, which is the model language of UDPS, has several restrictions relative to Pascal. Thus extension to the model language will be a further work.

An example behavior of UDPS and the syntax definition of Pascal-S are presented in APPENDIX.

References

1. Aho, A. V., and Ullman, J. D. 'The Theory of Parsing, Translation and Compiling' Vol. 1: parsing, Prentice-Hall, 1972.
2. N. Wirth 'Pascal-S': A subset and its implemention' in 'PASCAL-the language and its implemention' edited by D. W. Barron, John Wiley & Sons, 1981.
3. O. Stromfors and L. Jonesjo 'The implemention and experiences of a structured-oriented text editor' ACM, 1981.
4. PRIME Compuetr, Inc. 'The pascal Reference Guide' pascal language manual, 1980.

Appendix-1

An example behavior of UDPS

```
OK,

OK,
SEG =UDPS

*********************************
*                               *
*   WELCOME TO UDPS !           *
*                               *
*      LEE MYUNG JOON           *
*      U.I.T DEPT. OF COMPUTER SCIENCE
*                               *
*       SELECT  YOUR  OPTION.   *
*                               *
*********************************

      1. HELP
      2. GLOBAL DOCUMENT
      3. MODULE DOCUMENT
      4. EXECUTION ENVIRONMENT

.....TYPE YOUR OPTION NUMBER !
       IF YOU DO NOT HAVE ANY INFORMATION ON THIS SYSTEM,
       SELECT OPTION-1. .....
>1

.....
***  1. HELP
.....

..... THIS UDPS(U.I.T. DOCUMENTEE PASCAL-S SYSTEM) IS
          THE SYSTEM FOR TEACHING COMPILER-INTERPRETER SYSTEM.

..... YOU CAN HAVE THE OVERALL DOCUMENTATION OF THE SYSTEM.
          ALSO YOU CAN GET THE SOURCE PROGRAM FOR EACH MODULE.

  MODEL COMPILER: PASCAL-S

  SYSTEM COMMANDS ARE

          N: SELECTED OPTION NUMBER
          F: DISPLAY NENT PAGE OF THE SPECIFIED DOCUMENT
          B: DISPLAY PREVIOUS PAGE OF THE SPECIFIED DOCUMENT
          T: DISPLAY TOP PAGE OF THE SPECIFIED DOCUMENT
          R: RETURN TO UPPER SYSTEM LEVEL
          Q: EXIT TO PRIMOS
**********************************************************

.....!!! TYPE COMMAND ('R' OR 'Q').
>R
```

```
**********************************
*                                *
*   WELCOME TO UDPS !            *
*                                *
*       LEE MYUNG JOON           *
*       U.I.T DEPT. OF COMPUTER SCIENCE
*                                *
*         SELECT  YOUR  OPTION.  *
*                                *
**********************************
```

1. HELP
2. GLOBAL DOCUMENT
3. MODULE DOCUMENT
4. EXECUTION ENVIRONMENT

.....TYPE YOUR OPTION NUMBER !
IF YOU DO NOT HAVE ANY INFORMATION ON THIS SYSTEM,
SELECT OPTION-1.....

>3

.....

**** 3. MODULE DOCUMENT ****

.....

****

..... YOU CAN GET THE MODULE DOCUMENT FOR
THIS COMPILER-INTERPRETER SYSTEM.

1. PROGRAM LEVEL
2. BLOCK LEVEL
3. STATEMENT LEVEL
4. EXPRESSION LEVEL

****

..... TYPE OPTION NUMBER !

>1

.....

**** 3. 1 PROGRAM LEVEL

.....

..... YOU CAN GET FOLLOWING INFORMATION.

1. MAIN ROUTINE
2. ERROR-MESSAGE PROCEDURES
3. TABLE-MANAGEMENT PROCEDURES
4. CODE-GENERATION PROCEDURES
5. LEXICAL ANALYSYS PROCEDURES
6. DECLARATION AND INITIALIZATION

****

....!! TYPE OPTION NUMBER.

>1

*** 3.1.1 MAIN ROUTINE ****

THIS DOCUMENT CONTAINS
MAIN COMPILER ROUTINE AND INTERPRETER ROUTINE

1. DECLARATION
   ..... REFERENCE TO 3.1.6
2. INITIALIZATION
   ..... REFERENCE TO 3.1.6
3. COMPILING
4. EXECUTING

```
PROGRAM PASCALS(INPUT, OUTPUT);

 (DECLARATION)

BEGIN

 (INITIALIZATION)

>R
```

**** 3.1 PROGRAM LEVEL

..... YOU CAN GET FOLLOWING INFORMATION.

1. MAIN ROUTINE
2. ERROR-MESSAGE PROCEDURES
3. TABLE-MANAGEMENT PROCEDURES
4. CODE-GENERATION PROCEDURES
5. LEXICAL ANALYSYS PROCEDURES
9. DECLARATION AND INITIALIZATION

****

.....!! TYPE OPTION NUMBER.

```
>R
```

**** 3. MODULE DOCUMENT ****

****

..... YON CAN GET THE MODULE DOCUMENT FOR
THIS COMPILER-INTERPRETER SYSTEM.

1. PROGRAM LEVEL
2. BLOCK LEVEL
3. STATEMENT LEVEL
4. EXPRESSION LEVEL

```
**********************************************************************

------ TYPE OPTION NUMBER !

? R

                    **********************************
                    *                                *
                    *   WELCOME TO UDPS !            *
                    *                                *
                    *      LEE MYUNG JOON            *
                    *      U.I.T DEPT. OF COMPUTER SCIENCE
                    *                                *
                    *        SELECT YOUR OPTION.     *
                    *                                *
                    **********************************

        1. HELP
        2. GLOBAL DOCUMENT
        3. MODULE DOCUMENT
        4. EXECUTION ENVIRONMENT

------ TYPE YOUR OPTION NUMBER !
       IF YOU DO NOT HAVE ANY INFORMATION ON THIS SYSTEM,
       SELECT OPTION-1.
? 4

**********  4. EXECUTION ENVIRONMENT  **********

------ YOU CAN EXECUTE
            PASCAL-S COMPILER-INTERPRETER SYSTEM

    1. EXECUTION
    2. POST-EXECUTION

**********************************************************************

------ TYPE OPTION NUMBER !
? 1

**********  4.1 EXECUTION  **********

    YOU CAN GET COMPILE-TIME INFORMATION AND
       RUN-TIME INFORMATION BY SPECIFYING

OPTION-1: SYSTEM-RESERVED TABLE
OPTION-2: IDENTIFIER TABLE
OPTION-3: BLOCK TABLE
OPTION-4: ARRAY TABLE
OPTION-5: STRING TABLE
OPTION-6: REAL CONSTANT TABLE
OPTION-7: CODES FOR EACH STATEMENT
OPTION-8: CODES FOR TOTAL PROGRAM
OPTION-9: CODE EXECUTION SEQUENCE
```

```
**************************************************

..... SELECT YOUR OPTIONS !
          'TYPE 'Y' OR 'N'.
>DO YOU SELECT OPTION-1: Y
>DO YOU SELECT OPTION-2: Y
>DO YOU SELECT OPTION-3: Y
>DO YOU SELECT OPTION-4: Y
>DO YOU SELECT OPTION-5: Y
>DO YOU SELECT OPTION-6: Y
>DO YOU SELECT OPTION-7: Y
>DO YOU SELECT OPTION-8: Y
>DO YOU SELECT OPTION-9: Y
>SOURCE FILE NAME: SAMPLE
>INPUT DATA FILE NAME: D-SAMPLE
>OUTPUT DATA FILE NAME: O-SAMPLE
>LISTING FILE  NAME: L-SAMPLE
.....! EXECECUTION BEGINS ...
.....! EXECUTION ENDS      ...

..................................................
****  4. EXECUTION ENVIRONMENT                ****
..................................................

..... YOU CAN EXECUTE
          PASCAL-S COMPILER-INTERPRETER SYSTEM.

    1. EXECUTION
    2. POST-EXECUTION

**************************************************

..... TYPE OPTION NUMBER !
>2

..................................................
****  4.2 POST-EXECUTION
..................................................

..... YOU CAN GET FOLLOWING INFORMATION.

    1. SOURCE FILE
    2. INPUT DATA
    3. OUTPUT DATA
    4. LISTING

**************************************************

.....!! TYPE OPTION NUMBER.

>4

    1  PROGRAM HANOI (INPUT, OUTPUT);
    2  VAR N: INTEGER;
    3      FFROM, TTO, AUX: CHAR;
    4  PROCEDURE TOWERS (N: INTEGER; FFROM, TTO, AUX: CHAR);
```

```
    5    BEGIN
    6        IF(N=1) THEN

***CODE***      1:      1       2       5
***CODE***      2:     24               1
***CODE***      3:     45
       7          WRITELN('MOVE', ,N,' FROM PEG', FFROM', TTO)
***CODE***      4:     11
***CODE***      5:     24               9
***CODE***      6:     28               0
***CODE***      7:      1       2       5
***CODE***      8:     29               1
***CODE***      9:     24              10
***CODE***     10:     28               9
***CODE***     11:      1       2       6
***CODE***     12:     29               4
***CODE***     13:     24               8
***CODE***     14:     28              19
>F

       6        ELSE BEGIN
***CODE***     15:      1       2       7
***CODE***     16:     29               4
***CODE***     17:     63
       9          TOWERS(N-1, FFROM, AUX, TTO);
***CODE***     18:     10
***CODE***     19:     18              33
***CODE***     20:      1       2       5
***CODE***     21:     24               1
***OCDE***     22:     53
***CODE***     23:      1       2       6
***CODE***     24:      1       2       8
***CODE***     25:      1       2       7
      10          WRITELN( MOVE DISK', N', FROM PEG', FFROM,' TO PEG', TTO)
***CODE***     26:     19               8
***CODE***     27:      3       1       2
***CODE***     28:     24               9
***CODE***     29:     28              27
***CODE***     30:      1       2       5
***CODE***     31:     29               1
***CODE***     32:     24              10
***CODE***     33:     28              36
***CODE***     34:      1       2       6
>F

***CODE***     35:     29               4
***CODE***     36:     24               8
***CODE***     37:     28              46
***CODE***     38:      1       2       7
***CODE***     39:     29               4
      11          TOMERS(N-1, AUX, TTO, FFROM);
***CODE***     44:     53
***CODE***     45:      1       2       8
***CODE***     46:      1       2       7
```

```
***CODE***      47:    1    2    6
      12          END;
***CODE***      48:   19         8
***CODE***      49:    3    1    2
      13        END;
      14
      15        BEGIN
      16          READ(N);
***CODE***      50:   32
***CODE***      51:    0    1    5
>F

***CODE***      52:   27         1
      17        FFROM:='A';
***CODE***      53:    0    1    6
***CODE***      54:   24       193
      18        TTO:='C';
***CODE***      55:   38
***CODE***      56:    0    1    7
***CODE***      57:   24       195
      19        AUX:='B';
***CODE***      58:   38
***CODE***      59:    0    1    8
***CODE***      60:   24       194
      20        TOWERS(N, FFROM, TTO, AUX);
***CODE***      61:   38
***CODE***      62:   18        33
***CODE***      63:    1    1    5
***CODE***      64:    1    1    6
***CODE***      65:    1    1    7
***CODE***      66:    1    1    8
      21        END.
***CODE***      67:   19         8
***CODE***      68:   31
>F
```

***** SYSTEM-RESERVED-TABLE *****

IDENTIFIERS	LINK	OBJ	TYP	REF	NRM	LEV	ADR
0: AUX	−1	1	0	0	1	0	0
1: FALSE	0	0	3	0	1	0	0
2: TRUE	1	0	3	0	1	0	−32767
3: REAL	2	2	2	0	1	0	1
4: CHAR	3	2	4	0	1	0	1
5: BOOLEAN	4	2	3	0	1	0	1
6: INTEGER	5	2	1	0	1	0	1
7: ABS	6	4	2	0	1	0	0
8: SQR	7	4	2	0	1	0	2
9: ODD	8	4	3	0	1	0	4
10: CHR	9	4	4	0	1	0	5
11: ORD	10	4	1	0	1	0	6
12: SUCC	11	4	4	0	1	0	7
13: PRED	12	4	4	0	1	0	8
14: ROUND	13	4	1	0	1	0	9
15: TRUNC	14	4	1	0	1	0	10

```
    16: SIN          15      4      2      0      1      0     11
    17: COS          16      4      2      0      1      0     12
    18: EXP          17      4      2      0      1      0     13
    19: LN           18      4      2      0      1      0     14
>F

    20: SQRT         19      4      2      0      1      0     15
    21: ARCTAN       20      4      2      0      1      0     16
    22: EOF          21      4      3      0      1      0     17
    23: EOLN         22      4      3      0      1      0     18
    24: READ         23      3      0      0      1      0      1
    25: READLN       24      3      0      0      1      0      2
    26: WRITE        25      3      0      0      1      0      3
    27: WRITELN      26      3      0      0      1      0      4
    28:              27      3      0      2      1      0     50
****** IDENTIFIER-TABLE******
 IDENTIFIERS       LINK      OBJ      TYP      REF      NRM      LEV      ADR
    29: N             0        1        1        0        1        1        5
    30: FFROM        29        1        4        0        1        1        6
    31: TTO:         30        1        4        0        1        1        7
    32: AUX          31        1        4        0        1        1        8
    33: TOMERS       32        3        0        3        1        1        0
    34: N             0        1        1        0        1        2        5
    35: FFROM        34        1        4        0        1        2        6
    36: TTO          35        1        4        0        1        2        7
    37: AUX          36        1        4        0        1        2        8

****** BLOCK TABLE ******
BLOCKS      LAST      LPAR      PSZE      VSZE
>F
     1:       28         1         0         0
     2:       33        28         5         9
     3:       37        37         9         9

****** ARRAY TABLE ******
ARRAYS      XTYP ETYP EREF      LOW HIGH ELSZ SIZE

****** STRING TABLE ******
MOVE DISK FROM PEG    TO PEG MOVE DISK FROM PEG    TO PEG

****** REAL-CONSTANT TABLE ******

****** CODES FOR TOTAL PROGRAM ******
    0:    1 2  5,   24    1,   45    ,   11   18,   24    9,
    5:   28    0,    1 2  5,   29    1,   24   10,   28    9,
   10:    1 2  6,   29    4,   24    8,   28   19,    1 2  7,
   15:   29    4,   63    ,   10   49,   18   33,    1 2  5,
   20:   24    1,   53    ,    1 2  6,    1 2  8,    1 2  7,
   25:   19    8,    3 1  2,   24    9,   28   27,    1 2  5,
   30:   29    1,   24   10,   28   36,    1 2  6,   29    4,
>F

   35:   24    8,   28   46,    1 2  7,   29    4,   63    ,
   40:   18   33,    1 2  5,   24    1,   53    .    1 2  8,
   45:    1 2  7,    1 2  6,   19    8,    3 1  2,   32    ,
```

```
   50:      0  1    5,      27      1,      0  1    6,      24    193,      38      ,
   55:      0  1    7,      24    195,      38      ,       0  1    8,      24    194,
   60:     38      ,       18     33,       1  1    5,       1  1    6,       1  1    7,
   65:      1  1    8,      10      8,      31      , (EOR)
3
(EOF)

****** CODE EXECUTION SEQUENCE ******
     50·····>  51·····>  52·····>  53·····>  54·····>
     55·····>  56·····>  57·····>  58·····>  59·····>
     60·····>  61·····>  62·····>  63·····>  64·····>
     65·····>  66·····>   0·····>   1·····>   2·····>
      3·····>  18·····>  19·····>  20·····>  21·····>
     22·····>  23·····>  24·····>  25·····>   0·····>
      1·····>   2·····>   3·····>  18·····>  19·····>
     20·····>  21·····>  22·····>  23·····>  24·····>
     25·····>   0·····>   1·····>   2·····>   3·····>
      4·····>   5·····>   6·····>   7·····>   8·····>
      9·····>  10·····>  11·····>  12·····>  13·····>
     14·····>  15·····>  16·····>  17·····>  40·····>
>F

     26·····>  27·····>  28·····>  29·····>  30·····>
     31·····>  32·····>  33·····>  34·····>  35·····>
     36·····>  37·····>  38·····>  39·····>  40·····>
     41·····>  42·····>  43·····>  44·····>  45·····>
     46·····>  47·····>   0·····>   1·····>   2·····>
      3·····>   4·····>   5·····>   6·····>   7·····>
      8·····>   9·····>  10·····>  11·····>  12·····>
     13·····>  14·····>  15·····>  16·····>  17·····>
     40·····>  48·····>  49·····>  26·····>  27·····>
     28·····>  29·····>  30·····>  31·····>  32·····>
     33·····>  34·····>  35·····>  36·····>  37·····>
     38·····>  39·····>  40·····>  41·····>  42·····>
     43·····>  44·····>  45·····>  46·····>  47·····>
      0·····>   1·····>   2·····>   3·····>  18·····>
     19·····>  20·····>  21·····>  22·····>  23·····>
     24·····>  25·····>   0·····>   1·····>   2·····>
      3·····>   4·····>   5·····>   6·····>   7·····>
      8·····>   9·····>  10·····>  11·····>  12·····>
     13·····>  14·····>  15·····>  16·····>  17·····>
     49·····>  26·····>  27·····>  28·····>  29·····>
     30·····>  31·····>  32·····>  33·····>  34·····>
     35·····>  36·····>  37·····>  38·····>  39·····>
     40·····>  41·····>  42·····>  43·····>  44·····>
>F

     45·····>  46·····>  47·····>   0·····>   1·····>
      2·····>   3·····>   4·····>   5·····>   6·····>
      7·····>   8·····>   9·····>  10·····>  11·····>
     12·····>  13·····>  14·····>  15·····>  16·····>
     17·····>  49·····>  48·····>  49·····>  48·····>
     49·····>  67·····>
>T
```

```
    1   PROGRAM HANOI (INPUT, OUTPUT);
    2   VAR N: INTEGER;
    3         FFROM, TTO, AUX: CHAR;
    4   PROCEDURE TOWERS(N: INTEGER; FFROM, TTO, AUX: CHAR);
    5      BEGIN
    6         IF(N=1) THEN
***CODE***     1:       1       2       5
***CODE***     2:      24               1
***CODE***     3:      45
    7            WRITELN('MOVE DISK' ,N,' FROM PEG', FFROM,' TO PEG', TTO)
***CODE***     4:      11
***CODE***     5:      24               9
***CODE***     6:      28               0
***CODE***     7:       1       2       5
***CODE***     8:      29               1
***CODE***     9:      24              10
***CODE***    10:      28               9
***CODE***    11:       1       2       6
***CODE***    12:      29               4
***CODE***    13:      24               8
***CODE***    14:      28              19
>R
```

.....

**** 4.2 POST-EXECUTION

.....

..... YOU CAN GET FOLLOWING INFORMATION.

1. SOURCE FILE
2. INPUT DATA
3. OUTPUT DATA
4. LISTING

****

.....!! TYPE OPTION NUMBER.

```
>3

MOVE DISK              1 FROM PEG A TO PEG C
MOVE DISK              2 FROM PEG A TO PEG B
MOVE DISK              1 FROM PEG C TO PEG B
MOVE DISK              3 FROM PEG A TO PEG C
MOVE DISK              1 FROM PEG B TO PEG A
MOVE DISK              2 FROM PEG B TO PEG C
MOVE DISK              1 FROM PEG A TO PEG C

        202 STEPS
>R
```

.....

**** 4.2 POST-EXECUTION

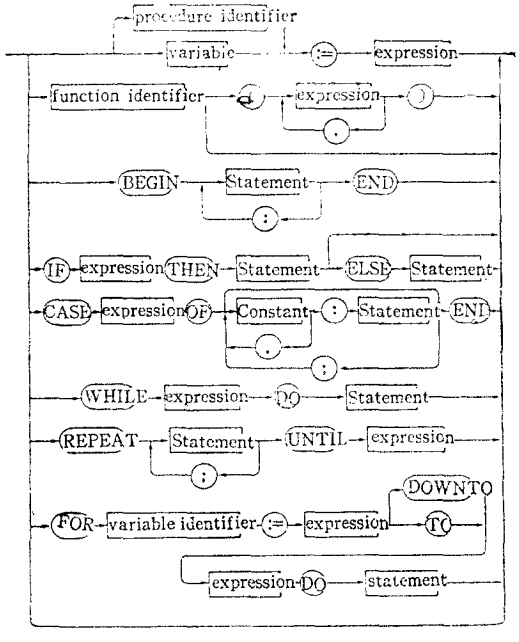
.....

..... YOU CAN GET FOLLOWING INFORMATION.

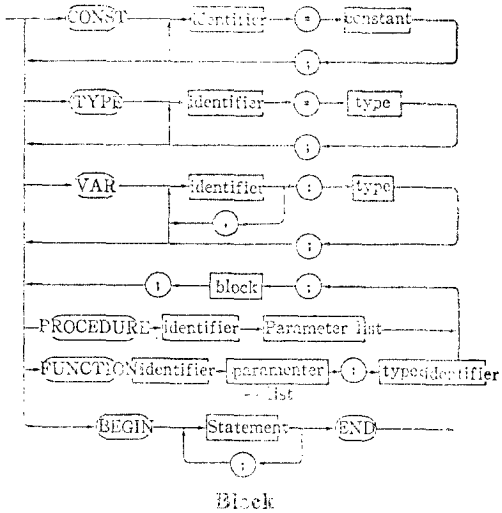
```
    1. SOURCE FILE
    2. INPUT DATA
    3. OUTPUT DATA
    4. LISTING

**********************************************************

.....!! TYPE OPTION NUMBER.

>Q
OK, COMO -E
```

Aprendix-2

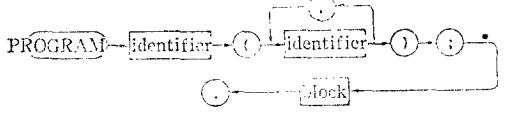
The syntax definiiton of Pascal-S



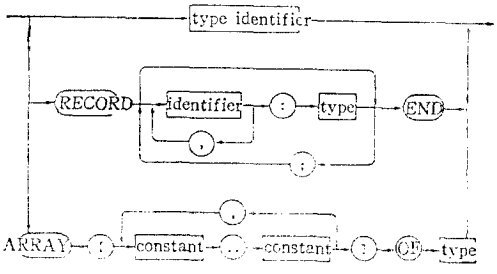
Statement



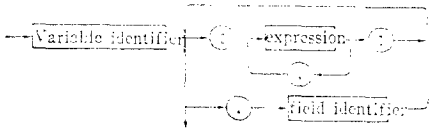
Block



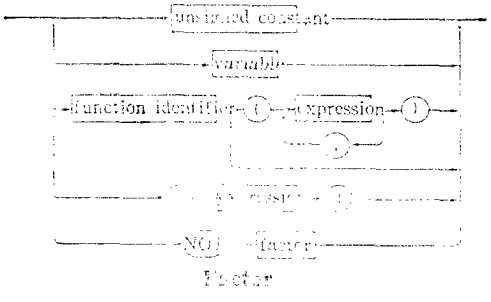
Program



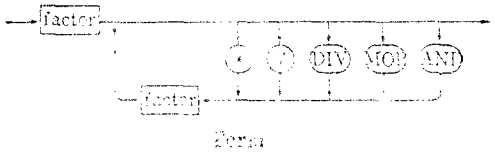
Type



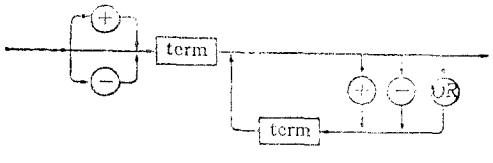
Variable



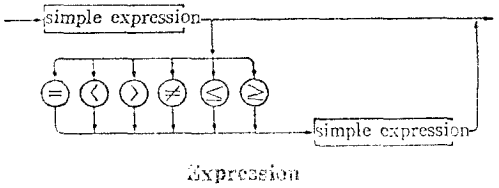
Factor



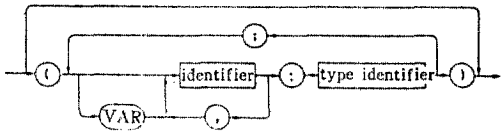
Term



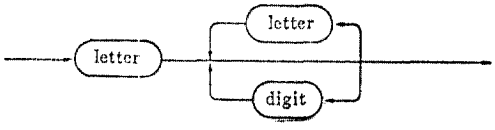
Simple expression



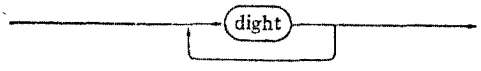
Expression



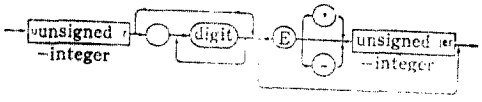
Parameter list



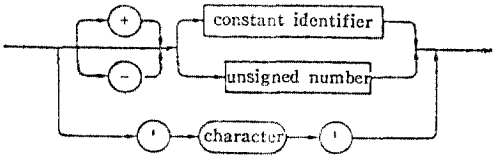
Identifier



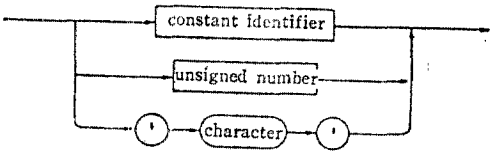
Unsigned integer



Unsigned number



Unsigned constant



Constant