MINI-PASCAL COMPILER

MINI-PASCAL COMPILER

INTRODUCTION

The MINI-PASCAL COMPILER is a set of two programs. These programs can be used to compile and interprete pascal programs.The compiler was orginally written on a TRS-80 MODEL 1 .The compiler as it stands can be used on the TRS-80 MODEL 1, the TRS-80 MODEL 3, and the TRS-80 MODEL 4 (in model 3 mode). These computers are reasonably old , so when designing the compiler, I decided to make it as portable as possible. The compiler is written in standard MICROSOFT BASIC and has only a few machine specific commands. These consist mainly of input and output and are descibed in the INSTALLATION section. This means the compiler can be implemented on most computers that support BASIC, with only minor modifications. The compiler requires at least one disk drive, but could be modified to run on a cassette system only (if enough memory is available), and at least 48K of memory. The compiler does not generate machine code for a specific computer, but instead generates a pseudo-code (or intermediate code). This allows the program to be run on most computers, as the code generated by the compiler is interpreted, rather than executed directly by the microprocessor. The two programs that make up the compiling system are the compiler and the interpreter. An ascii editor is also required, to generate the input for the compiler. The compiler is a one-pass recursive descent compiler and supports all the important features of pascal, and could be extended if a programmer wished.

INSTALLATION

The program is provided in two forms, a listing and a disk. The disk contains 3 programs, the compiler, the interpreter, and a test program. The disk does not contain a disk operating system, and requires one to be used. The DOS's that can be used are NEWDOS, MULTIDOS, DOSPLUS, and LDOS. The programs were developed using NEWDOS.

To load a program you first have to enter the particular DOS's BASIC. This is usually accomplished by typing BASIC (or LBASIC for LDOS) at DOS ready. Once in basic the files can be loaded by typing LOAD followed by the filename of the program enclosed in quotes. The filenames of the programs are :-

PASCAL/BAS - the pascal compiler
INTERPRT/BAS - the p-code interpreter
TEST/DAT - the test program

The input and output of the compiler system consists of two files. The compiler reads one line of text from one file (the input file) at a time, and outputs p-code to another. The input file must be in ascii, as produced by most editors. For modification purposes the following BASIC commands are described.

OPEN mode, buf no., filename - This command causes BASIC to initiate a disk file. The mode is a one-character string, "I" for input ,"O" for output. The buffer no. is a numeric parameter specifying the file number.The filename consists of an 8 letter name followed by a slash, and then an extension. e.g. TESTFILE/BAS is a file called TESTFILE with extension /BAS for BASIC.
examples of the open command are :-

```
OPEN "I",1,"TESTFILE/BAS"
OPEN "I",1,F$
OPEN "O",2,F1$
```

The filename may be replaced by a string variable, in the above examples - F$ and F1$.

CLOSE buf no.,buf no..... - CLOSE "closes" a disk file or files. The buf no. parameters specify which files are to be closed. If no number(s) are given then ALL files are closed

INPUT #1, L$ - This inputs a line of text from file number 1 into variable L$.

INPUT #1, OP, P1, P2 - This inputs 3 numbers into the variables OP, P1, and P2 from file number 1.

PRINT #2, OP, P1, P2 - This outputs 3 numbers contained in variables OP, P1 and P2 to disk file number 2.

The compiler only compiles one line of text at a time and so never has the complete ascii file in memory at one time, this means that the size of program able to be compiled is only limited by the size of the disk. The output file is also never fully in memory at one time and so is only limited by disk size. The files are 'spooled' back and forth from disk to gain space for the compiler.

OPERATION

The compiler is easy to use, the following steps outline the general method of compiling a pascal program :-

1. Create ascii file of pascal program with editor.
   (ascii file must be in upper case).
2. Save ascii file to disk.
3. Load the pascal compiler
   (enter BASIC and use the command LOAD "PASCAL/BAS")
4. Execute the compiler (use the command RUN)
5. The compiler will ask for the source filename (excluding disk drive no.), this is the filename of the ascii file.
6. The compiler will then ask for the disk drive no. of the source file.
7. The compiler will then ask for the destination filename, this is the filename of the file to which code will be written (excluding disk drive no.).
8. The compiler will then ask for the disk drive no. of the destination file.
9. The compiler will then compile the ascii file and output p-code to the destination file.
10. During compilation the compiler will generate an on screen listing (although it can be printed on the printer instead, see compiler listing). The listing shows the user how far the compiler has progressed.
11. Errors may be reported, if so they will be below the line to which they refer. The compiler will only report 10 errors at maximum, further errors are suppressed. The compiler will report the error number, the message associated with the number can be found in the ERRORS section.
12. When compilation is complete, the compiler will report the total number of errors.

These steps will hopefully produce an error free p-code program in the destination file. If an error is found, then the user should correct the error and repeat steps 3 - 8, until the program is error free.

After obtaining a p-code program, it can be interpreted using the interpreter. The following steps show how to interprete a p-code program :-

1. Load the p-code interpreter.
   (enter BASIC and use the command LOAD "INTERPRT/BAS")
2. Execute the interpreter (use the BASIC command RUN)
3. The interpreter will then ask for the source filename, this is the filename of the p-code program generated by the compiler.
4. The interpreter will then ask for the disk drive no. of the source file.
5. The interpreter will now load the p-code program into memory
6. Before starting to interprete, the interpreter will ask if you are ready for it to begin. Press the ENTER (or RETURN) key to start.
7. The pascal program will now start to run.
8. If there is a divide by zero error within the program then the error will occur in the BASIC interpreter or compiler. This is the only detectable runtime error.

Your pascal program should now be running, it will not run at a fast rate, because the p-code is being interpreted. The p-code can be converted to native machine code, but this part of a compiler system is machine specific. The p-code runs on a hypothetical stack based machine. The language requires some form of stack to allow block structuring to be possible.

ERRORS

The compiler will generate errors with the following numbers :-

1	-	Integer too large
2	-	Incomplete character string
3	-	expected symbol was identifier
4	-	expected symbol was an integer
5	-	expected symbol was a character string
6,7,8	-	expected symbol was logical operator
9	-	expected symbol was multiplication operator
10	-	expected symbol was division operator
11	-	expected symbol was addition operator
12	-	expected symbol was subtraction operator
13,14,15	-	expected symbol was boolean relation
16,17,18	-	expected symbol was boolean relation
21,22	-	missing bracket
23	-	missing comma
24	-	missing semicolon
25	-	missing period
26	-	missing colon
27	-	missing := (becomes) symbol
28	-	missing .. (subrange) symbol
29	-	expected symbol was PROGRAM
30	-	expected symbol was VAR
31	-	expected symbol was PROCEDURE
32	-	expected symbol was ARRAY
33	-	expected symbol was OF
34	-	expected symbol was BEGIN
35	-	expected symbol was END
36	-	expected symbol was IF
37	-	expected symbol was THEN
38	-	expected symbol was ELSE
39	-	expected symbol was WHILE
40	-	expected symbol was DO
41	-	expected symbol was READ
42	-	expected symbol was WRITE
43	-	UNEXPECTED SYMBOL
44	-	expected symbol was FUNCTION
45	-	expected symbol was REPEAT
46	-	expected symbol was UNTIL
47	-	identifier declared twice
48	-	identifier not declared
49	-	illegal class
50	-	indexed variable not of array type
51	-	illegal index expression
52	-	expected operand was of type boolean
53	-	expected operand was of type integer
54	-	illegal types
55	-	expression is different type to variable
56	-	illegal input variable
57	-	illegal output variable
58	-	expression not boolean
59	-	incorrect number of parameters

LANGUAGE DEFINITION

IDENTIFIERS

Identifiers are constructed from the 26 letters of the alphabet and the 10 digits. Only the first 8 characters are used to distinguish between identifiers. The first character of an identifier MUST be a letter.

e.g. CURSOR

CONSTANTS

Constants are integer numbers in the range -32767 to +32767, any number outside this range will generate an error.

e.g. -17245

CHARACTER STRINGS

Character strings are of one character in length, and are surrounded by quote marks. Any letter character of the character set of the computer may be used.

e.g 'A'

BOOLEANS

Booleans are integers with only two states, TRUE or FALSE
TRUE and FALSE are built in identifiers

e.g BOOLEAN := TRUE

EXPRESSIONS

Expressions can be formed from the following components :-

character strings
constants
identifiers
operators
brackets

The expressions are exactly as in standard pascal except the MOD command is not provided.

TYPES

Data may take one of the following types :-

INTEGER
BOOLEAN
CHAR
or ARRAY

The array type supports only one dimensional arrays, with subscripts starting at 1 and finishing at a positive number.

VAR statements

To define variables the following statement is used :-

```
VAR id,id,id.... : type ;
    id,id,id.... : type ;
        .            .
        .            .
        .            .
```

The number of identifiers declared may be one or more than one per definition. Variables of just one type may be declared, or more than one type.

The type part of the VAR statement is as follows :-

INTEGER - for integers
CHAR - for character strings
BOOLEAN - for boolean's

for arrays :-

ARRAY (1 .. no.) OF type - where type is not an array type and no. is positive.

ASSIGNMENTS

These are of the form :-

id := expression

e.g TIME := 359

BLOCK CONSTRUCTS

PROCEDURES

Procedures are provided as in standard pascal except for the passing of parameters. Parameters can only be passed by value.

FUNCTIONS

Functions are provided as in standard pascal exactly. A return value is provided.

examples of block constructs :-

```
PROCEDURE ADDONE ;
    I := I + 1 ;

FUNCTION ADDONE ( IDENT : INTEGER ) : INTEGER ;
    ADDONE := IDENT + 1 ;
```

The notation for parameters for procedures is :-

PROCEDURE id (id,id,... : type ; id,id,... : type ; ...) ;

No parameters may be used or just parameters of one type, or parameters of more than one type

The notation for parameters for functions is :-

FUNCTION id (id,id,... : type ; id,id... : type ; ...) : type;

where the parameters passed to the function are exactly like procedures, but a final type is specified outside the brackets. This is for the return value , which is the name of the function.

COMPOUND STATEMENTS

These are exactly as in standard pascal

e.g.
```
    BEGIN
      TIME := 359 ;
      OLDTIME := TIME - 1;
      WRITE (OLDTIME)
    END ;
```

This construct allows more statements where only one statement is usually accepted. The last statement in the group above has no semicolon following it, this rule must be adhered to.

REPETITION CONSTRUCTS

There are two repetition constructs provided, WHILE DO and REPEAT UNTIL.

WHILE DO

This is exactly as in standard pascal. The following illustrates a WHILE DO construct

e.g.
```
  WHILE TIME > 200 DO
     BEGIN
       TIME := TIME + 1 ;
       WRITE (TIME)
     END ;
```

REPEAT UNTIL

This is exactly as in standard pascal. The following illustrates a REPEAT UNTIL construct

e.g.
```
    REPEAT
      TIME := TIME + 1 ;
      WRITE (TIME)
    UNTIL TIME > 200
```

Notice that no begin end structure is required.

IF STATEMENTS

These are exactly as in standard pascal
e.g.

```
    IF TIME <= 354 THEN WRITE (TIME-1) ;
                   ELSE WRITE (TIME+1) ;
```

The statement can be used with or without the else clause.

FORMAL DEFINITIONS

Variable declarations :-

```
   VAR id,id.. : type ; id,id.. : type ; .. id,id.. : type ;
```

Assignment statement :-

```
   identifier := expression ;
```

Compound statement :-

```
   BEGIN
     statement ;
     statement ;
        .
        .
        .
     statement
   END
```

Procedure declaration :-

```
   PROCEDURE identifier ;
```

or with parameters :-

```
   PROCEDURE id (id,id..:type ; id,id..:type ; ..) ;
```

Function declaration :-

```
   FUNCTION identifier : type ;
```

or with parameters :-

```
   FUNCTION id (id,id..:type ; id,id..:type ; ..) : type ;
```

If statement :-

```
   IF expression THEN statement
```

or with an else clause :-

```
   IF expression THEN statement
                 ELSE statement
```

While statement :-

```
   WHILE expression DO statement
```

Repeat statement :-

```
   REPEAT statement ; statement ; .. statement UNTIL expression
```

These are the only constructs allowed, the case statement can be simulated by using an IF statement.
e.g.

```
CASE TIME OF
     1 : WRITE(LASTTIME) ;
     2 : WRITE(TIME) ;
     3 : WRITE(NEWTIME) ;
   END
```

can be simulated by :-

```
    IF TIME=1 THEN WRITE(LASTTIME) ; ELSE
         IF TIME=2 THEN WRITE(TIME) ; ELSE
              IF TIME=3 THEN WRITE(NEWTIME) ;
```

BLOCKS

Each procedure, function or program consists of three main parts :-

The variable declaration part
The procedure and function declaration parts
The statement part

The statement part must begin with a compound statement. The scope of identifiers is as in standard pascal. Only one restriction is placed on procedures and functions, that is they must have been declared before being called.

Recursion is allowed in only one form, where the procedure name is used within the same procedure.

PROGRAMS

There are standard identifiers for use within the program as a whole, these are :-

INTEGER
BOOLEAN
CHAR
FALSE
TRUE

They can of course be redeclared in a block for another purpose.

INSIDE THE COMPILER

The compiler is made of 4 distinct phases, lexical analysis, parsing, semantic analysis and code generation.

LEXICAL ANALYSIS

This part of the program deals with the input file replacing the words with numbers. This also passes any relevant information to the parser, such as the value of a constant. The compiler spends most of its time doing lexical analysis and so it has to be as efficient as possible. Lexical analysers can be automatically constructed from deterministic finite automaton. This is a complex subject and is best left to the reader to find out more if interested.

PARSING

The parser recognizes the syntactic structure of a language. There are several methods of parsing, top down parsing, and bottom up parsing. The method used in this compiler is recursive descent, a variant of top down parsing. Recursive descent works by having a small section of the compiler that recognizes certain structures of the language, using routines nested within it to recognize any other structures.

SEMANTIC ANALYSIS

This part of the compiler uses information about the variables, functions and procedures used in the program being compiled, and checks compatibility. The compatibility of types of variables is checked, the compatibility of the number of parameters of a procedure is also checked. The semantic rules that define this checking are usually implied and so are hard to define. The construction of semantic analysers is difficult, unlike parsing, and lexical analysis. This is one area in compilers that needs developing.

CODE GENERATION

This is the last section of the program, and it controls the generation of code. The code generator handles the output of code, and also uses the information in the symbol table to replace variable names with addresses. This part of a compiler can usually be written for a specific machine, but in this compiler it is written for a hypothetical machine for portability.

PASCAL INTERPRETER LISTING

```
 10 '   P-CODE INTERPRETER FOR MINI-PASCAL COMPILER MK 1
 20 '   WRITTEN BY C.TAYLOR 1988
 30 '   FOR MICROSOFT-BASIC COMPUTERS WITH AT LEAST ONE DISK DRIVE
 40 DIM C(1000,3):'   ARRAY TO HOLD P-CODE
 50 DIM S(1000):'   STACK
 60 DIM D(20):'   DISPLAY USED FOR ACCESSING DYNAMICALLY ALLOCATED VARIA
    BLES
 70 PRINT "P-CODE INTERPRETER MK 1"
 80 INPUT "NAME OF FILE TO BE INTERPRETED ";F$
 90 INPUT "NUMBER OF DISK DRIVE ON WHICH FILE RESIDES ";F
100 F$ = F$ + ":" + CHR$ (F + 48)
110 OPEN "I",1,F$:'   OPEN INPUT FILE
120 INPUT £1,OP,P1,P2:'   READ OPCODE, PARAMETER 1 AND PARAMETER 2 FROM
    INPUT FILE
130 IF   OP = 19 THEN C(P1,3) = P2: GOTO 120:'   BACKPATCH JUMPS
140 IF   OP = 20 THEN GOTO 160
150 C(PC,1) = OP:C(PC,2) = P1:C(PC,3) = P2:PC = PC + 1: GOTO 120
160 C(PC,1) = 20:C(PC,2) = 0:C(PC,3) = 0: CLOSE 1:'   CLOSE INPUT FILE
170 INPUT "READY TO INTERPRET ";A$
180 PC = 0:T = 0:B = 1:S(1) = 1:S(2) = 0:S(3) = - 1:D(1) = 1
190 OP = C(PC,1):P1 = C(PC,2):P2 = C(PC,3)
200 PC = PC + 1:IX = 0
210 IF   OP > 8 THEN IX = 1:OP = OP - 10
220 IF   OP = 0 THEN T = T + 1:S(T) = P2: GOTO 190
230 IF   OP < > 1 THEN 440
240 IF   P2 = 0 THEN T = B - 1:B = S(T + 2):PC = S(T + 3):LV = S(T + 1):
    D(LV) = B: GOTO 190
250 IF   P2 = 1 THEN S(T) = - S(T): GOTO 190
260 IF   P2 > 5 THEN 300
270 T = T - 1
271 IF   P2 = 5 THEN S(T) = INT (S(T) / S(T + 1)): GOTO 280
272 S(T) = INT ( - ((P2 = 2) * (S(T) + S(T + 1)) + (P2 = 3) * (S(T) - S(
    T + 1)) + (P2 = 4) * (S(T) * S(T + 1))))
280 GOTO 190
300 IF   P2 = 6 THEN S(T) = S(T) AND 1: GOTO 190
310 IF   P2 > 13 THEN 390
320 T = T - 1
330 S(T) = ((P2 = 8) * (S(T) = S(T + 1)) + (P2 = 9) * (S(T) < > S(T + 1)
    ) + (P2 = 10) * (S(T) < S(T + 1)) + (P2 = 11) * (S(T) > = S(T + 1)) +
    (P2 = 12) * (S(T) > S(T + 1)) + (P2 = 13) * (S(T) < = S(T + 1)) + (P
    2 = 14) * - (S(T) OR S(T + 1)) + (P2 = 15) * - (S(T) AND S(T + 1)))
340 GOTO 190
390 IF   P2 = 16 THEN S(T) = NOT S(T): GOTO 190
400 IF   P2 > 20 THEN 420
410 T = - ((P2 = 19) * (T + 1) + (P2 = 20) * (T - 1)): GOTO 190
420 IF   P2 = 21 THEN T = T + 1:S(T) = S(T - 1): GOTO 190
440 IF   OP < > 2 THEN 470
450 P1 = P1 + (S(T) * - (IX = 1))
460 T = T + 1 - IX:S(T) = S(D(P2) + P1): GOTO 190
470 IF   OP < > 3 THEN 500
480 P1 = P1 + (S(T - 1) * - (IX = 1))
490 S(D(P2) + P1) = S(T):T = T - IX - 1: GOTO 190
500 IF   OP < > 4 THEN 520
510 S(T + 2) = B:S(T + 3) = PC:S(T + 1) = P1:LV = P1:B = T + 1:D(LV) = B
    :PC = P2: GOTO 190
520 IF   OP = 5 THEN T = T + P2: GOTO 190
530 IF   OP = 6 THEN PC = P2: GOTO 190
535 IF   OP = 10 THEN PRINT : PRINT "INTERPRETATION ENDED": STOP
540 IF   OP < > 7 THEN 570
550 IF   S(T) = P1 THEN PC = P2
```

```
560 T = T - 1: GOTO 190
570 IF   P2 = 0 THEN T = T + 1: INPUT A$:S(T) = ASC (A$): GOTO 190:'   A
    SC CONVERTS FIRST CHARACTER IN A$ TO ASCII NUMBER
580 IF   P2 = 1 THEN PRINT CHR$ (S(T)):T = T - 1: GOTO 190:'   CHR$ CONV
    ERTS ASCII NUMBER IN S(T) TO A STRING CHARACTER
590 IF   P2 = 2 THEN T = T + 1: INPUT A:S(T) = A * - (A < 32768 AND A >
    - 32767): GOTO 190
600 PRINT S(T):T = T - 1: GOTO 190
```

PASCAL COMPILER LISTING

```
  5 '   INTEGER PASCAL COMPILER , PRODUCES P-CODE TO BE INTERPRETED
  6 '   WRITTEN BY C.TAYLOR 1988
  7 '   FOR MICROSOFT-BASIC COMPUTERS WITH AT LEAST ONE DISK DRIVE
  8 '   FOR MORE INFORMATION ON COMPILERS SEE PRINCIPLES OF COMPILER DES
    IGN BY A.V.AHO AND J.D.ULLMAN
  9 '   THE DESIGN OF THIS COMPILER WAS BASED UPON THE COMPILER DESCRIBE
    D IN STRUCTURED SYSTEM PROGRAMMING BY WELSH AND MCKEAG (1980)
 10 CLEAR 500'    ( NOT NECESSARY FOR COMPUTERS WITH DYNAMIC STRING STOR
    AGE )
 20 PRINT "MINI PASCAL COMPILER MK. 1 VERSION 2"
 30 INPUT "NAME OF FILE TO BE COMPILED ";F$
 35 INPUT "NUMBER OF DISK DRIVE ON WHICH FILE RESIDES ";F
 36 F$ = F$ + ":" + CHR$ (F + 48)
 40 OPEN "I",1,F$:'   OPEN INPUT FILE
 50 INPUT "NAME OF OUTPUT FILE ";F1$
 55 INPUT "NUMBER OF DISK DRIVE TO WHICH FILE WILL BE WRITTEN ";F1
 56 F1$ = F1$ + ":" + CHR$ (F1 + 48)
 60 OPEN "O",2,F1$:'   OPEN OUTPUT FILE
 70 DIM WS$(29),WS(29),LL(29):'   RESERVED WORD LOOK-UP TABLES
 80 FOR  I = 1 TO 29: READ WS$(I),WS(I),LL(I): NEXT I:'   READ RESERVED
    WORDS INTO TABLE
 89 '   ************ RESERVED WORD DATA ************
 90 DATA " ",1,1,"IF",34,6,"DO",38,12,"OF",31,16
100 DATA "OR",6,22," ",1,24,"END",33,26,"VAR",28,29
110 DATA "DIV",8,0,"AND",5,0,"NOT",4,0," ",1,0
120 DATA "THEN",35,0,"ELSE",36,0,"READ",39,0
130 DATA " ",1,0,"BEGIN",32,0,"WHILE",37,0
140 DATA "ARRAY",30,0,"WRITE",40,0,"UNTIL",44,0
150 DATA " ",1,0,"REPEAT",43,0," ",1,0,"PROGRAM",27,0
160 DATA " ",1,0,"PROCEDUR",29,0,"FUNCTION",42,0," ",1,0
170 CH$ = "":'   CLEAR INPUT VARIABLE
171 LL = 1:'   SET LAST IN LINE VARIABLE TO FIRST POSITION IN LINE
172 CP = 1:'   SET CURRENT POSITION TO FIRST POSITION IN LINE
173 EOL = - 1:'   MAKE END OF FILE FALSE
174 MA = 32767:'   SET MAXIMUM INTEGER ALLOWED
175 EX = 0:'   SET LINE ERROR COUNT TO ZERO
176 DIM EL(10):'   DEFINE ERROR ARRAY
180 DIM N$(100),NA(100,8),VL(50),BI(50):'   SYMBOL AND LABEL TABLES
190 TA = 1
200 NF = 1:'   SET NEXT FREE ENTRY IN SYMBOL TABLE TO ONE
210 DIM OP(50),TY(100),DI(20),FR(20),BO(50),LS(2,12,50):'   DEFINE SCOPE
     , TEMPORARY STACKS AND LABEL STACK ARRAYS
220 PRINT "LISTING PRODUCED BY MINI-PASCAL COMPILER MK 1": PRINT : PRINT

230 GOSUB 580:'   GET NEXT CHARACTER
240 GOSUB 3790:'   PROGRAM
250 PRINT
260 PRINT "COMPILATION COMPLETED :"
270 IF   EC = 0 THEN PRINT " NO": ELSE PRINT " ";EC
280 PRINT "ERRORS REPORTED"
290 CLOSE :'   CLOSE ALL FILES
291 STOP
299 '   ************ READ NEXT LINE **********
300 IF   EX < = 0 THEN 410
309 '   PRINT ERRORS
310 PRINT "ERRORS - ";:EC = EC + EX
330 FOR  K = 1 TO EX
340 CR = EL(K)
370 PRINT CR;" ";
390 NEXT K
```

```
400 PRINT :EX = 0
409 '   READ NEXT LINE
410 CP = 2
415 EOL = EOF (1):'   SET END OF FILE FOR INPUT FILE
420 IF   EOL THEN LL = 100: RETURN :'   IF END OF FILE THEN EXIT
430 LINE INPUT £1,L$:'   READ NEXT ASCII LINE FROM INPUT FILE
440 L$ = " " + L$ + " ":LL = LEN (L$) + 1:CH$ = LEFT$ (L$,1): RETURN
449 '   ************ ERROR ***********
450 IF   EX > = 9 THEN EL(10) = 255:EX = 10: GOTO 480
460 EX = EX + 1
470 EL(EX) = FE
480 RETURN
489 '   ************ GET NEXT CHARACTER ************
490 IF   CP < > LL THEN 520
500 PRINT
510 GOSUB 300: RETURN
520 IF   EOL THEN 570:'   IF END OF FILE
530 CH$ = MID$ (L$,CP,1):'   PUT CHARACTER IN POSITION CP IN L$ INTO CH$

540 PRINT CH$;:'   OUTPUT CH$ TO SCREEN
541 '   IF OUTPUT WANTED ON A LINE PRINTER THEN SUBSTITUTE LPRINT FOR PR
    INT IN LINES 540 , 500 , 310 , 370 , 570 AND 400
550 CP = CP + 1
560 RETURN
569 '   END OF FILE
570 PRINT "   *** EOF ENCOUNTERED": CLOSE : STOP
579 '   ************ LEXICAL ANALYSIS *************
580 IF   CH$ = "" OR CH$ = " " THEN GOSUB 490: GOTO 580
590 IF   CH$ < "A" OR CH$ > "Z" THEN 700
599 '   ************* IDENTIFIER OR RESERVED WORD ************
600 KQ = 0:ID$ = ""
610 IF   KQ < 8 THEN KQ = KQ + 1:ID$ = ID$ + CH$
620 GOSUB 490
630 IF   (CH$ > = "0" AND CH$ < = "9") OR (CH$ > = "A" AND CH$ < = "Z") THEN
    610
640 WS$(LL(KQ)) = ID$
650 I = LL(KQ - 1) + 1
660 IF   WS$(I) = ID$ THEN 680
670 I = I + 1: GOTO 660
680 SY = WS(I)
690 RETURN
700 IF   CH$ < "0" OR CH$ > "9" THEN 770
709 '   ************ CONSTANT ************
710 CO = 0
720 AE = ASC (CH$) - 48
730 IF   (CO < MA / 10) OR (CO = MA / 10) AND (AE < = (MA / 10 - INT (MA
    / 10))) THEN CO = 10 * CO + AE: ELSE FE = 1: GOSUB 450:CO = 0
740 GOSUB 490
750 IF   NOT (CH$ < "0" OR CH$ > "9") THEN 720
760 SY = 2: RETURN
770 IF   CH$ < > "'" THEN 850
779 '   ************ CHAR SYMBOL ************
780 GOSUB 490
790 IF   CH$ = "'" THEN GOSUB 490: IF   CH$ < > "'" THEN FE = 2: GOSUB 4
    50
800 CO = ASC (CH$)
810 GOSUB 490
820 IF   CH$ < > "'" THEN FE = 2: GOSUB 450
830 GOSUB 490
840 SY = 3: RETURN
850 IF   CH$ < > ":" THEN 880
```

```
 859 '   ************* COLON AND BECOMES SYMBOLS ************
 860 GOSUB 490: IF   CH$ = "=" THEN SY = 25: GOSUB 490: RETURN
 870 SY = 24: RETURN
 880 IF   CH$ < > "." THEN 910
 889 '   ************* THRU AND PERIOD SYMBOLS ************
 890 GOSUB 490: IF   CH$ = "." THEN SY = 26: GOSUB 490: RETURN
 900 SY = 23: RETURN
 910 IF   CH$ < > "<" THEN 960
 919 '   ************* LESS THAN, LESS THAN OR EQUAL TO , AND NOT EQUAL TO
      SYMBOLS ************
 920 GOSUB 490
 930 IF   CH$ = "=" THEN SY = 12: GOSUB 490: RETURN
 940 IF   CH$ = ">" THEN SY = 15: GOSUB 490: RETURN
 950 SY = 11: RETURN
 960 IF   CH$ < > ">" THEN 1000
 969 '   ************* GREATER THAN, AND GREATER THAN OR EQUAL TO SYMBOLS
     ************
 970 GOSUB 490
 980 IF   CH$ = "=" THEN SY = 13: GOSUB 490: RETURN
 990 SY = 14: RETURN
 999 '   ************* SPECIAL SYMBOLS ************
1000 IF   CH$ = "+" THEN SY = 9: GOSUB 490: RETURN
1010 IF   CH$ = "-" THEN SY = 10: GOSUB 490: RETURN
1020 IF   CH$ = "*" THEN SY = 7: GOSUB 490: RETURN
1030 IF   CH$ = "=" THEN SY = 16: GOSUB 490: RETURN
1040 IF   CH$ < > "(" THEN 1110
1049 '   ************* BRACKETS AND COMMENTS SYMBOLS *************
1050 GOSUB 490
1060 IF   CH$ < > "*" THEN SY = 18: RETURN
1061 '   IF COMMENT THEN COMMENT SKIPPED
1070 GOSUB 490
1080 IF   CH$ < > "*" THEN 1070
1090 GOSUB 490: IF   CH$ < > ")" THEN 1070
1100 GOSUB 490: GOTO 580
1109 '   ************* MORE SPECIAL SYMBOLS ************
1110 IF   CH$ = ")" THEN SY = 17: GOSUB 490: RETURN
1120 IF   CH$ = "," THEN SY = 21: GOSUB 490: RETURN
1130 IF   CH$ = ";" THEN SY = 22: GOSUB 490: RETURN
1139 '   ************* ILLEGAL SYMBOLS ************
1140 IF   CH$ = "£" OR CH$ = "/" OR CH$ = "!" OR CH$ = "[" OR CH$ = "&" OR
     CH$ = "@" OR CH$ = "$" OR CH$ = "%" OR CH$ = "?" OR CH$ = CHR$ (34) THEN
     SY = 41
1150 GOSUB 490: RETURN
1159 '   ************* SYNTAX ERROR ************
1160 FE = ES + 9: GOSUB 450: RETURN
1169 '   ************* ACCEPT A SYMBOL *************
1170 IF   SY = SE THEN GOSUB 580: ELSE ES = SE: GOSUB 1160: GOSUB 580
1180 RETURN
1190 '   ************* SEMANTIC ERROR *************
1200 FE = ES: GOSUB 450: RETURN
1209 '   ************* SIMPLE TYPE ************
1210 IF   ID$ = "INTEGER" THEN TB = 1
1220 IF   ID$ = "CHAR" THEN TB = 4
1230 IF   ID$ = "BOOLEAN" THEN TB = 2
1240 SE = 1: GOSUB 1170
1250 RETURN
1259 '   ************ INDEX RANGE TYPE *************
1260 IN = CO
1270 SE = 2: GOSUB 1170
1280 SE = 26: GOSUB 1170
```

```
1290 IX = CO
1300 SE = 2: GOSUB 1170
1310 RETURN
1319 '   ************ COMPLEX TYPE ************
1320 IF   SY = 1 THEN GOSUB 1210:BT = TB: GOTO 1410
1330 SE = 30: GOSUB 1170
1340 SE = 18: GOSUB 1170
1350 GOSUB 1260
1360 SE = 17: GOSUB 1170
1370 SE = 31: GOSUB 1170
1380 GOSUB 1210
1390 AE = TB
1400 BT = 8
1410 RETURN
1419 '   ************ VARIABLE DECLARATION ************
1420 GOSUB 4420
1430 SE = 1: GOSUB 1170
1440 IF   NOT (SY = 21) THEN 1490
1450 SE = 21: GOSUB 1170
1460 GOSUB 4420
1470 SE = 1: GOSUB 1170
1480 GOTO 1440
1490 SE = 24: GOSUB 1170
1500 GOSUB 1320
1510 GOSUB 4460
1520 RETURN
1529 '   ************ VARIABLE PART ************
1530 IF   SY < > 28 THEN 1580
1540 SE = 28: GOSUB 1170
1550 GOSUB 1420
1560 SE = 22: GOSUB 1170
1570 IF   SY = 1 THEN 1550
1580 RETURN
1589 '   ************ PROCEDURE AND FUNCTION DECLARATION ************
1590 BI(BR) = - 1
1600 IF   SY = 29 THEN SE = 29:CL = 2: ELSE SE = 42:CL = 6:FV = - 1
1610 GOSUB 1170
1620 IF   SY = 1 THEN N$ = ID$: ELSE N$ = "????????"
1630 GOSUB 4040
1640 NW = EN: GOSUB 5420
1650 SE = 1: GOSUB 1170
1660 GOSUB 4840
1670 GOSUB 3950:D9 = - 1
1680 IF   SY < > 18 THEN 1780
1690 SE = 18: GOSUB 1170
1700 GOSUB 1420
1710 IF   SY = 17 THEN 1730
1720 SE = 22: GOSUB 1170: GOTO 1700
1730 OC = 5:P1 = 0:P2 = 5: GOSUB 5440: FOR  I = NF - 1 TO DI(DP) STEP - 1

1740 OC = 2:P1 = D9:P2 = LV: GOSUB 5440
1750 OC = 3:P1 = NA(I,7):P2 = NA(I,8): GOSUB 5440
1760 D9 = D9 - 1
1770 NEXT I:OC = 5:P1 = 0:P2 = - 5: GOSUB 5440:SE = 17: GOSUB 1170:NA(NW,
     2) = NF - DI(DP): GOSUB 5600
1780 SE = 22: GOSUB 1170:NA(NW,7) = - (NA(NW,2) + ( - 1 * FV)):NA(NW,8) =
     LV
1790 GOSUB 3720
1800 GOSUB 3980
1810 GOSUB 4850
1820 RETURN
```

```
1829 '   ************ PROCEDURE AND FUNCTION PART ***********
1830 IF   (SY < > 29 AND SY < > 42) THEN 1870
1840 GOSUB 1590
1850 SE = 22: GOSUB 1170
1860 GOTO 1830
1870 RETURN
1879 '   ************ VARIABLE ************
1880 N$ = ID$:CL = 3: GOSUB 4150
1890 VT = EN
1900 VA = VT: GOSUB 4930
1910 SE = 1: GOSUB 1170
1920 IF   SY < > 18 THEN 2040
1930 IF   NA(VT,0) = 6 THEN 2040
1940 IF   NA(VT,1) < > 8 THEN ES = 61: GOSUB 1190
1950 SE = 18: GOSUB 1170
1960 Q1 = VT
1970 GOSUB 2530
1980 VT = Q1: GOSUB 4570:I1 = TY:I2 = 1: GOSUB 4480: IF   CT = 0 THEN ES =
     62: GOSUB 1190
1990 IF   NA(VT,4) = 1 THEN VT = 1
2000 IF   NA(VT,4) = 2 THEN VT = 3
2010 IF   NA(VT,4) = 4 THEN VT = 2
2020 GOSUB 4940
2030 SE = 17: GOSUB 1170
2040 RETURN
2049 '   ************ FACTOR ************
2050 IF   SY < > 1 AND SY < > 2 AND SY < > 3 AND SY < > 4 AND SY < > 18 THEN
     2270
2060 IF   SY < > 1 THEN 2130
2070 N$ = ID$:CL = 15: GOSUB 4150
2080 IF   NA(EN,0) = 5 THEN TY = EN: GOSUB 4550:SC = NA(EN,7): GOSUB 5010
     :SE = 1: GOSUB 1170: GOTO 2280
2090 IF   NA(EN,0) = 6 THEN GOSUB 5470: GOTO 2280
2100 GOSUB 1880:TY = VT: GOSUB 4550
2110 GOSUB 4950
2120 GOTO 2280
2130 IF   SY = 2 THEN TY = 1: GOSUB 4550:SC = CO: GOSUB 5010:SE = 2: GOSUB
     1170: GOTO 2280
2140 IF   SY = 3 THEN TY = 2: GOSUB 4550:SC = CO: GOSUB 5010:SE = 3: GOSUB
     1170: GOTO 2280
2150 IF   SY < > 18 THEN 2200
2160 SE = 18: GOSUB 1170
2170 GOSUB 2530
2180 SE = 17: GOSUB 1170
2190 GOTO 2280
2200 IF   SY < > 4 THEN 2270
2210 SE = 4: GOSUB 1170
2220 GOSUB 2050
2230 TY = 3: GOSUB 4550:OP = 2: GOSUB 4540
2240 GOSUB 4580
2250 GOSUB 5080
2260 GOTO 2280
2270 ES = 41: GOSUB 1160
2280 RETURN
2289 '   ************ TERM ************
2290 GOSUB 2050
2300 IF   SY < > 7 AND SY < > 8 AND SY < > 5 THEN 2380
2310 IF   SY = 7 OR SY = 8 THEN OP = 1: ELSE OP = 2
2320 BO(B1) = SY:B1 = B1 + 1: GOSUB 4540
2330 GOSUB 580
2340 GOSUB 2050
```

```
2350 GOSUB 4580
2360 B1 = B1 - 1:BO = BO(B1): IF   BO = 5 THEN GOSUB 5100: ELSE GOSUB 505
     0
2370 GOTO 2300
2380 RETURN
2389 '   ********** SIMPLE EXPRESSION **********
2390 SI = 0
2400 IF   SY = 9 OR SY = 10 THEN SI = - 1: GOSUB 580
2410 GOSUB 2290: IF   SI = 0 THEN 2440
2420 OP = 1: GOSUB 4540:TY = 1: GOSUB 4550
2430 GOSUB 4580: GOSUB 5030
2440 IF   SY < > 9 AND SY < > 10 AND SY < > 6 THEN 2520
2450 IF   SY = 9 OR SY = 10 THEN OP = 1: ELSE OP = 2
2460 BO(B1) = SY:B1 = B1 + 1: GOSUB 4540
2470 GOSUB 580
2480 GOSUB 2290
2490 GOSUB 4580
2500 B1 = B1 - 1:BO = BO(B1): IF   BO = 6 THEN GOSUB 5100: ELSE GOSUB 505
     0
2510 GOTO 2440
2520 RETURN
2529 '   ********** EXPRESSION **********
2530 GOSUB 2390
2540 IF   SY < 11 OR SY > 16 THEN 2600
2550 BO(B1) = SY:B1 = B1 + 1:OP = 4: GOSUB 4540
2560 GOSUB 580
2570 GOSUB 2390
2580 GOSUB 4580
2590 B1 = B1 - 1:BO = BO(B1): GOSUB 5120
2600 RETURN
2609 '   ********** ASSIGNMENT STATEMENT **********
2610 GOSUB 1880
2620 AV = VT
2630 SE = 25: GOSUB 1170
2640 GOSUB 2530
2650 GOSUB 4570:I1 = AV:I2 = TY: GOSUB 4480: IF   CT = 0 THEN ES = 66: GOSUB
     1190
2660 GOSUB 5130
2670 RETURN
2679 '   ********** INPUT VARIABLE **********
2680 GOSUB 1880
2690 I1 = 2:I2 = VT: GOSUB 4480: IF   CT < > 0 THEN IM = 4: ELSE I1 = 1: GOSUB
     4480: IF   CT < > 0 THEN IM = 1: ELSE ES = 67: GOSUB 1190
2700 GOSUB 5180: GOSUB 5130
2710 RETURN
2719 '   ********** READ STATEMENT **********
2720 SE = 39: GOSUB 1170
2730 SE = 18: GOSUB 1170
2740 GOSUB 2680
2750 IF   SY < > 21 THEN 2790
2760 SE = 21: GOSUB 1170
2770 GOSUB 2680
2780 GOTO 2750
2790 SE = 17: GOSUB 1170
2800 RETURN
2809 '   ********** REPEAT STATEMENT **********
2810 GOSUB 5210
2820 NW = 1: GOSUB 5230
2830 SE = 43: GOSUB 1170
2840 GOSUB 3360
2850 IF   SY < > 22 THEN 2870
```

```
2860 SE = 22: GOSUB 1170: GOTO 2840
2870 SE = 44: GOSUB 1170
2880 GOSUB 2530
2890 GOSUB 4570:I1 = TY:I2 = 3: GOSUB 4480: IF   CT = 0 THEN ES = 69: GOSUB
     1190
2900 NW = 1: GOSUB 5410
2910 GOSUB 5220
2920 RETURN
2929 '   ************ OUTPUT VALUE ************
2930 GOSUB 2530
2940 I1 = 2:I2 = VT: GOSUB 4480: IF   CT < > 0 THEN IM = 4: ELSE I1 = 1: GOSUB

     4480: IF   CT < > 0 THEN IM = 1: ELSE ES = 68: GOSUB 1190
2950 GOSUB 5190
2960 RETURN
2969 '   ************ WRITE STATEMENT ************
2970 SE = 40: GOSUB 1170
2980 SE = 18: GOSUB 1170
2990 GOSUB 2930
3000 IF   SY < > 21 THEN 3040
3010 SE = 21: GOSUB 1170
3020 GOSUB 2930
3030 GOTO 3000
3040 SE = 17: GOSUB 1170
3050 RETURN
3060 GOSUB 5210
3069 '   ************ IF STATEMENT ************
3070 SE = 34: GOSUB 1170
3080 GOSUB 2530
3090 GOSUB 4570:I1 = TY:I2 = 3: GOSUB 4480: IF   CT = 0 THEN ES = 69: GOSUB
     1190
3100 NW = 1: GOSUB 5260
3110 NW = 1: GOSUB 5410
3120 SE = 35: GOSUB 1170
3130 GOSUB 3360
3140 IF   SY < > 36 THEN NW = 1: GOSUB 5280: GOTO 3220
3150 NW = 2: GOSUB 5260
3160 NW = 2: GOSUB 5350
3170 NW = 1: GOSUB 5280
3180 SE = 36: GOSUB 1170
3190 GOSUB 3360
3200 NW = 2: GOSUB 5280
3210 GOSUB 5220
3220 RETURN
3229 '   ************ WHILE STATEMENT ************
3230 GOSUB 5210
3240 NW = 1: GOSUB 5230
3250 NW = 2: GOSUB 5260
3260 SE = 37: GOSUB 1170
3270 GOSUB 2530
3280 GOSUB 4570:I1 = TY:I2 = 3: GOSUB 4480: IF   CT = 0 THEN ES = 69: GOSUB
     1190
3290 NW = 2: GOSUB 5410
3300 SE = 38: GOSUB 1170
3310 GOSUB 3360
3320 NW = 1: GOSUB 5350
3330 NW = 2: GOSUB 5280
3340 GOSUB 5220
```

```
3350 RETURN
3359 '   ************ STATEMENT ************
3360 IF   SY < > 1 AND SY < > 32 AND SY < > 39 AND SY < > 40 AND SY < > 3
     4 AND SY < > 37 AND SY < > 43 THEN 3580
3370 IF   SY < > 1 THEN 3500
3380 N$ = ID$:CL = 6: GOSUB 4150
3390 IF   NA(EN,0) = 6 OR NA(EN,0) = 3 THEN GOSUB 2610: GOTO 3490
3400 NW = EN:SE = 1: GOSUB 1170:ZP = 0
3410 IF   SY < > 18 THEN 3470: ELSE SE = 18: GOSUB 1170
3420 ZP = ZP + 1: GOSUB 2530: GOSUB 4570
3430 IF   SY < > 21 THEN 3460
3440 SE = 21: GOSUB 1170
3450 GOTO 3420
3460 SE = 17: GOSUB 1170: IF   ZP < > NA(NW,2) THEN ES = 102: GOSUB 1200
3470 GOSUB 5430
3480 OC = 5:P1 = 0:P2 = - NA(NW,2): GOSUB 5440
3490 GOTO 3580
3500 IF   SY = 34 THEN GOSUB 3060: GOTO 3580
3510 IF   SY = 37 THEN GOSUB 3230: GOTO 3580
3520 IF   SY = 39 THEN GOSUB 2720: GOTO 3580
3530 IF   SY = 40 THEN GOSUB 2970: GOTO 3580
3540 IF   SY = 32 THEN GOSUB 3590: GOTO 3580
3550 IF   SY = 43 THEN GOSUB 2810
3560 GOTO 3580
3570 ES = 41: GOSUB 1160
3580 RETURN
3589 '   ************ COMPOUND STATEMENT ************
3590 SE = 32: GOSUB 1170
3600 GOSUB 3360
3610 IF   SY < > 22 THEN 3650
3620 SE = 22: GOSUB 1170
3630 GOSUB 3360
3640 GOTO 3610
3650 SE = 33: GOSUB 1170
3660 RETURN
3670 '   ************ STATEMENT PART ************
3680 IF   BI(BR - 1) = 0 THEN OC = 19:P1 = 0:P2 = PC - 1: GOSUB 5440
3690 GOSUB 3590
3700 IF   BI(BR - 1) = - 1 THEN OC = 1:P1 = 0:P2 = 0: GOSUB 5440: ELSE OC
     = 20:P1 = 0:P2 = 0: GOSUB 5440
3710 RETURN
3719 '   ************ BLOCK ************
3720 BR = BR + 1
3730 GOSUB 1530
3740 GOSUB 1830
3750 OC = 5:P1 = 0:P2 = NL: GOSUB 5440
3760 GOSUB 3670
3770 BR = BR - 1
3780 RETURN
3789 '   ************ PROGRAM ************
3790 GOSUB 3950
3800 N$ = "INTEGER":CL = 7: GOSUB 4040:NA(EN,1) = 1
3810 N$ = "CHAR":CL = 7: GOSUB 4040:NA(EN,1) = 4
3820 N$ = "BOOLEAN":CL = 7: GOSUB 4040:NA(EN,1) = 2
3830 N$ = "TRUE":CL = 5: GOSUB 4040:NA(EN,7) = 1
3840 N$ = "FALSE":CL = 5: GOSUB 4040:NA(EN,7) = 0
3850 SE = 27: GOSUB 1170
3860 SE = 1: GOSUB 1170
```

```
3870 SE = 22: GOSUB 1170
3880 BI(BR) = 0
3890 OC = 6:P1 = 0:P2 = 0: GOSUB 5440
3900 GOSUB 3950: GOSUB 4840
3910 GOSUB 3720
3920 GOSUB 3980: GOSUB 4850
3930 GOSUB 3980
3940 RETURN
3949 '   ************ OPEN SCOPE **********
3950 DP = DP + 1
3960 DI(DP) = NF
3970 RETURN
3979 '   ************ CLOSE SCOPE ***********
3980 FOR  I = DI(DP) TO NF
3990 N$(I) = "":NA(I,0) = 0
4000 NA(I,1) = 0:NA(I,2) = 0:NA(I,3) = 0
4010 NA(I,4) = 0:NA(I,5) = 0:NA(I,6) = 0:NA(I,7) = 0:NA(I,8) = 0
4020 NEXT I
4030 NF = DI(DP):DP = DP - 1: RETURN
4039 '   ************ ENTER NEW IDENTIFIER ************
4040 TE = DI(DP)
4050 IF   N$(TE) = "" THEN 4130
4060 LE = TE
4070 IF   N$(TE) > N$ THEN TE = NA(TE,5):LT = - 1: GOTO 4110
4080 IF   N$(TE) < N$ THEN TE = NA(TE,6):LT = 0: GOTO 4110
4090 ES = 51: GOSUB 1190
4100 TE = NA(TE,6):LT = 0
4110 IF   N$(TE) < > "" THEN 4060
4120 IF   LT THEN NA(LE,5) = NF: ELSE NA(LE,6) = NF
4130 N$(NF) = N$:NA(NF,0) = CL:NA(NF,1) = 0:NA(NF,2) = 0:NA(NF,3) = 0:NA(
     NF,4) = 0:NA(NF,7) = 0:NA(NF,8) = 0
4140 EN = NF:NF = NF + 1: RETURN
4149 '   ************ SEARCH FOR IDENTIFIER ************
4150 MU = 0:LS = DP
4160 TE = DI(LS)
4170 IF   N$(TE) = "" OR TE = 0 THEN 4250
4180 IF   N$(TE) > N$ THEN TE = NA(TE,5): GOTO 4240
4190 IF   N$(TE) < N$ THEN TE = NA(TE,6): GOTO 4240
4200 IF   INT (CL / NA(TE,0)) = CL / NA(TE,0) THEN 4400
4210 IF   NA(TE,0) = 6 THEN 4400
4220 MU = - 1
4230 TE = NA(TE,6)
4240 GOTO 4170
4250 IF   MU THEN ES = 53: GOSUB 1190:N$(TE) = "        ":NA(TE,1) = 0:NA
     (TE,2) = 0:NA(TE,3) = 0:NA(TE,4) = 0
4260 IF   MU = 0 THEN 4320
4270 IF   CL / 3 = INT (CL / 3) THEN CL = 3
4280 IF   CL / 2 = INT (CL / 2) THEN CL = 2
4290 IF   CL / 7 = INT (CL / 7) THEN CL = 7
4300 IF   CL / 5 = INT (CL / 5) THEN CL = 5
4310 NA(TE,0) = CL: GOTO 4400
4320 LS = LS - 1
4330 IF   LS > 0 THEN 4160
4340 ES = 52: GOSUB 1190
4350 IF   CL / 3 = INT (CL / 3) THEN CL = 3
4360 IF   CL / 2 = INT (CL / 2) THEN CL = 2
4370 IF   CL / 7 = INT (CL / 7) THEN CL = 7
4380 IF   CL / 5 = INT (CL / 5) THEN CL = 5
```

```
4390 GOSUB 4040:TE = EN
4400 EN = TE
4410 RETURN
4419 '   ********** NEW VARIABLE **********
4420 IF   SY < > 1 THEN 4450
4430 N$ = ID$:CL = 3: GOSUB 4040
4440 VL(TA) = EN:TA = TA + 1
4450 RETURN
4459 '   ********** ADD ATTRIBUTES **********
4460 TA = TA - 1: FOR  I = 1 TO TA:NA(VL(I),1) = BT:NA(VL(I),2) = IN:NA(V
     L(I),3) = IX:NA(VL(I),4) = AE:VA = VL(I): GOSUB 4860:VL(I) = 0: NEXT
     I
4470 TA = 1: RETURN
4479 '   ********** CHECK COMPATIBILITY **********
4480 CT = 0: IF   NA(I1,1) = 8 THEN 4520
4490 IF   NA(I1,1) = NA(I2,1) THEN CT = - 1
4500 IF   (NA(I1,1) = 0) OR (NA(I2,1) = 0) THEN CT = - 1
4510 GOTO 4530
4520 IF   (NA(I1,2) = NA(I2,2)) AND (NA(I1,3) = NA(I2,3)) AND (NA(I1,4) =
     NA(I2,4)) THEN CT = - 1
4530 RETURN
4540 OT = OT + 1:OP(OT) = OP: RETURN
4550 TP = TP + 1:TY(TP) = TY: RETURN
4560 OP = OP(OT):OT = OT - 1: RETURN
4570 TY = TY(TP):TP = TP - 1: RETURN
4579 '   ********** CHECK COMPATIBILITY OF EXPRESSION **********
4580 GOSUB 4570:TT = TY
4590 GOSUB 4570:NT = TY
4600 GOSUB 4560
4610 IF   OP < > 1 THEN 4660
4620 I2 = 1:I1 = TT: GOSUB 4480:C1 = CT
4630 I1 = NT: GOSUB 4480:C2 = CT
4640 IF   (C1 = 0) OR (C2 = 0) THEN ES = 64: GOSUB 1190
4650 TY = 1: GOSUB 4550: RETURN
4660 IF   OP < > 2 THEN 4710
4670 I2 = 3:I1 = TT: GOSUB 4480:C1 = CT
4680 I1 = NT: GOSUB 4480:C2 = CT
4690 IF   (C1 = 0) OR (C2 = 0) THEN ES = 63: GOSUB 1190
4700 TY = 3: GOSUB 4550: RETURN
4710 I1 = TT:I2 = 1: GOSUB 4480:C1 = CT
4720 I1 = NT: GOSUB 4480:C2 = CT
4730 I1 = TT:I2 = 2: GOSUB 4480:C3 = CT
4740 I1 = NT: GOSUB 4480:C4 = CT
4750 T1 = 0:T2 = 0: IF   C1 = 0 OR C2 = 0 THEN T1 = - 1
4760 IF   C3 = 0 OR C4 = 0 THEN T2 = - 1
4770 IF   (T1 = - 1) AND (T2 = - 1) THEN ES = 65: GOSUB 1190
4780 TY = 3:C3 = CT
4790 I1 = NT: GOSUB 4480:C4 = CT
4800 T1 = 0:T2 = 0: IF   C1 = 0 OR C2 = 0 THEN T1 = - 1
4810 IF   C3 = 0 OR C4 = 0 THEN T2 = - 1
4820 IF   (T1 = - 1) AND (T2 = - 1) THEN ES = 65: GOSUB 1190
4830 TY = 3: GOSUB 4550: RETURN
4839 '   ********** OPEN STACK FRAME **********
4840 FR(LV) = NL:LV = LV + 1:NL = 4: RETURN
4849 '   ********** CLOSE STACK FRAME **********
4850 LV = LV - 1:NL = FR(LV): RETURN
4859 '   ********** ALLOCATE ADDRESS FOR VARIABLE **********
4860 RE = 1
```

```
4870 IF   NA(VA,1) < > 8 THEN 4890
4880 RE = (NA(VA,3) - NA(VA,2) + 1)
4890 NA(VA,8) = LV
4900 NA(VA,7) = NL
4910 NL = NL + RE
4920 RETURN
4929 '   ************ STACK A REFERENCE TO A VARIABLE ************
4930 S1 = S1 + 1:ST(S1) = VA: RETURN
4939 '   ************ STACK AN INDEXED REFERENCE TO A VARIABLE **********
     **
4940 ST(S1) = - ST(S1): RETURN
4949 '   ************ POP A REFERENCE TO A VARIABLE ************
4950 DR = ST(S1)
4960 IF   DR < 0 THEN OC = 12: ELSE OC = 02
4970 DR = ABS (DR)
4980 P1 = NA(DR,7):P2 = NA(DR,8): GOSUB 5440
4990 S1 = S1 - 1
5000 RETURN
5009 '   ************ STACK A CONSTANT ************
5010 OC = 0:P1 = 0:P2 = SC: GOSUB 5440
5020 RETURN
5029 '   ************ NEGATE AN INTEGER ************
5030 OC = 1:P1 = 0:P2 = 1: GOSUB 5440
5040 RETURN
5049 '   ************ BINARY INTEGER OPERATION ************
5050 OC = 1:P1 = 0:P2 = - ((BO = 7) * 4 + (BO = 8) * 5 + (BO = 9) * 2 + (
     BO = 10) * 3)
5060 GOSUB 5440: RETURN
5070 '   ************ NEGATE A BOOLEAN ************
5080 OC = 1:P1 = 0:P2 = 16: GOSUB 5440
5090 RETURN
5099 '   ************ BINARY BOOLEAN OPERATION ************
5100 OC = 1:P1 = 0:P2 = - ((BO = 5) * 15 + (BO = 6) * 14): GOSUB 5440
5110 RETURN
5119 '   ************ COMPARISON ************
5120 OC = 1:P1 = 0:P2 = - ((BO = 16) * 8 + (BO = 15) * 9 + (BO = 11) * 10
     + (BO = 13) * 11 + (BO = 14) * 12 + (BO = 12) * 13): GOSUB 5440: RETURN

5129 '   ************ ASSIGN ************
5130 DR = ST(S1)
5140 IF   DR < 0 THEN OC = 13: ELSE OC = 3
5150 DR = ABS (DR)
5160 P1 = NA(DR,7):P2 = NA(DR,8): GOSUB 5440
5170 S1 = S1 - 1: RETURN
5179 '   ************ READ OPERATION ************
5180 OC = 8:P1 = 0:P2 = - (IM = 1) * 2: GOSUB 5440: RETURN
5189 '   ************ WRITE OPERATION ************
5190 OC = 8:P1 = 0:P2 = - ((IM = 1) * 3 + (IM = 4) * 1): GOSUB 5440
5200 RETURN
5209 '   ************ OPEN A LABEL FRAME ************
5210 FL = FL + 1: RETURN
5219 '   ************ CLOSE A LABEL FRAME ************
5220 FOR  I = 1 TO 12:LS(1,I,FL) = 0:LS(2,I,FL) = 0: NEXT I:FL = FL - 1: RETURN


5229 '   ************ NEW LABEL ************
5230 LS(NW,1,FL) = 0
5240 LS(NW,2,FL) = PC
5250 RETURN
```

```
5259 '   ************ FUTURE LABEL ************
5260 LS(NW,1,FL) = 1
5270 LS(NW,2,FL) = 0: RETURN
5279 '   ************ EXPECTED LABEL ************
5280 LS(NW,1,FL) = 0
5290 EL = 2
5300 OC = 19:P1 = LS(NW,EL,FL):P2 = PC: GOSUB 5440
5320 EL = EL + 1
5330 IF   LS(NW,EL,FL) = 0 THEN LS(NW,2,FL) = PC: RETURN
5340 GOTO 5300
5349 '   ************ JUMP ************
5350 OC = 6
5360 IF   LS(NW,1,FL) = 0 THEN P1 = 0:P2 = LS(NW,2,FL): GOSUB 5440: RETURN

5370 FOR  I = 2 TO 12
5380 IF   LS(NW,I,FL) = 0 THEN JA = I: GOTO 5400
5390 NEXT I
5400 LS(NW,JA,FL) = PC:P1 = 0:P2 = 0: GOSUB 5440: RETURN
5409 '   ************ JUMP IF FALSE ************
5410 OC = 7: GOTO 5360
5419 '   ************ PROCEDURE ADDRESS ************
5420 NA(NW,4) = PC: RETURN
5430 OC = 4:P1 = NA(NW,8):P2 = NA(NW,4)
5439 '   ************ OUTPUT CODE ************
5440 PRINT £2,OC,P1,P2;:'   OUTPUT VALUES IN OC,P1,P2 TO OUTPUT FILE
5450 IF   OC < > 19 THEN PC = PC + 1
5460 RETURN
5469 '   ************ PARAMETER LIST ************
5470 NW = EN:SE = 1: GOSUB 1170:ZP = 0
5480 IF   SY < > 18 THEN 5560
5490 OC = 5:P1 = 0:P2 = 1: GOSUB 5440
5500 SE = 18: GOSUB 1170
5510 ZP = ZP + 1: GOSUB 2530: GOSUB 4570
5520 IF   SY < > 21 THEN 5550
5530 SE = 21: GOSUB 1170
5540 GOTO 5510
5550 SE = 17: GOSUB 1170: IF   ZP < > NA(NW,2) THEN ES = 102: GOSUB 1200
5560 GOSUB 5430
5570 OC = 5:P1 = 0:P2 = - NA(NW,2): GOSUB 5440
5580 TY = NW: GOSUB 4550
5590 RETURN
5600 IF   FV = 0 THEN RETURN
5610 SE = 24: GOSUB 1170
5620 GOSUB 1210:NA(NW,1) = BT: RETURN
```

TEST PASCAL PROGRAM

```
PROGRAM RECUR ;
VAR TEST : INTEGER ;
FUNCTION DIGITS (N:INTEGER):INTEGER ;
   VAR M:INTEGER ;
   BEGIN IF N<10
         THEN DIGITS := N
         ELSE BEGIN M:=N DIV 10 ;
                    DIGITS:=N-M*10+DIGITS(M)
              END
   END ;
BEGIN
      TEST :=315 ;
      WRITE (DIGITS(TEST)) ;
      REPEAT
      READ (TEST) ;
      WRITE (DIGITS(TEST))
      UNTIL TEST =100
END.
```

The test program will first display the number 9.

You will then be allowed to enter a number.

The test program will then display the sUm of the digits entered.

The test program shows how recursion works.