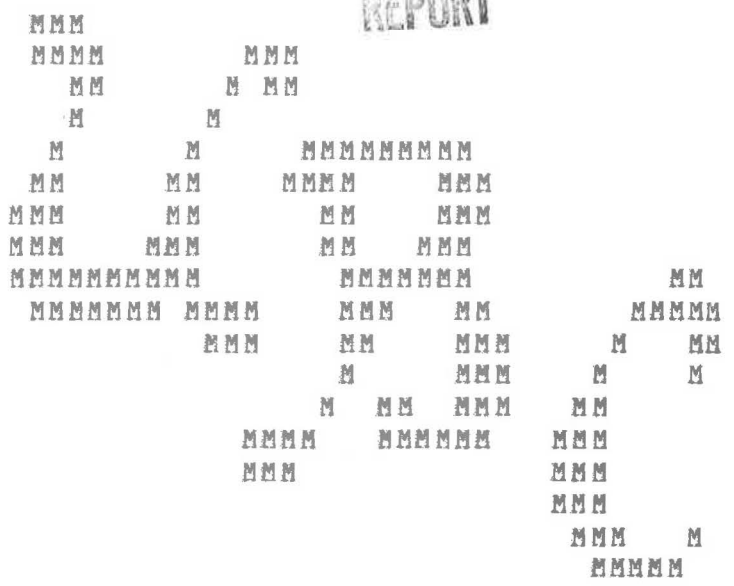

PASCAL/UBC Summary

by

Bary W. Pollack

Technical Manual TM-1

Sep 23, 1976

Department of Ccmputer Science
University of British Columbia
Vancouver, B. C.

PASCAL/UBC SUMMARY

Notation

Capitalized words should appear as given.
word is .... means "the definition of word is ...."

{ .... }	a syntactic object
[....]	optional item
[....]*	optional item, may appear zero or more times
[....]+	optional item, may appear one or more times
...	repeat previous construct zero or more times
,...	as above, separate occurrences with commas
;...	as above, separate occurrences with semicolons
|	separates alternatives
b	a required blank (one or more)

Special Symbols

```
AND ARRAY BEGIN CASE CONST DIV DO DOWNTO ELSE END FILE  FOR
FUNCTION  GOTO  IF  IN  LABEL  MOD  NIL NOT OF OR PROCEDURE
RECORD REPEAT SET THEN TO TYPE UNTIL VALUE VAR WHILE WITH
+   -   *   /   &   |   ¬   .   ,   :=   ;   :   @   ..
=   <>   ¬=   <   <=   >=   >   '   "
(   )   (*   *)   (.   .)
```

Comments

" comment " or (* comment *)

Identifiers (called 'id' below)

letter [letter | digit | _]*

Constants

```
digits [ . digits ] [ E [ + | - ] digits ]
#hexdigits
TRUE FALSE MAXINT ECL NIL
'any EBCDIC characters'
(A quote within a string must appear as ''.)
```

Characters

letter	ABCDEFGHIJKLMNOPQRSTUVWYXZ
digit	0123456789
hexdigit	0123456789ABCDEF

Sets

```
SET ( [ expression,... ] )
(. [ expression,... ] .)
```

Operator	Operation	Operand Type(s)	Result Type
:=	assignment	all except files	---
Arithmetic			
+	unary +	integer, real	integer, real
-	unary -	integer, real	integer, real
+	addition	integer, real	integer, real
-	subtraction	integer, real	integer, real
*	multiplication	integer, real	integer, real
DIV	integer division	integer	integer
/	real division	integer, real	real
MOD	modulus	integer	integer
Relational			
=	equality	SSSP	Boolean
<> ¬=	inequality	SSSP	Boolean
<	less than	scalar, string	Boolean
>	greater than	scalar, string	Boolean
<=	less or equal, set inclusion	scalar, string, set	Boolean
>=	greater or equal, set inclusion	scalar, string, set	Boolean
IN	set membership	1st operand scalar 2nd operand set	Boolean
Logical			
AND &	conjunction	Boolean	Boolean
OR |	disjunction	Boolean	Boolean
NOT ¬	negation	Bcolean	Boolean
Set			
+	union	any set type T	T
-	set difference	any set type T	T
*	intersection	any set type T	T

SSSP is scalar, set, string, or pointer

Procedure and Function Declarations

```
PROCEDURE id [ ( { [VAR] id,... : typeid |
                   PROCEDURE id,...  |
                   FUNCTION id,... : typeid };... ) ]

FUNCTION id  [ ( { [VAR] id,... : typeid |
                   PROCEDURE id,...  |
                   FUNCTION id,... : typeid };... ) ]
                   : typeid
```

Body

```
[ LABEL  label,... ; ]
[ CONST  [ id = constant ;]+ ]
[ TYPE   [ id = type ; ]+ ]
[ VAR    [ id,... : type ; ]+ ]
[ VALUE  [ id = constant ; |
           id = ( { [ digits * ] constant },... ) ; ]+ ]
[ { Procedure | Function }... ]
BEGIN statement;... END
```

Procedures, Functions, and Main Programs

A Procedure is Procedure Declaration ; body ;
A Function is Function Declaration ; body ;
A Main Program is body .

Procedures and Functions may have their bodies replaced by FORWARD or FORTRAN.

Statements

variable := expression Must have compatible types.
The precedence of operators from highest to lowest is:

4	NOT	¬							
3	*	/	DIV	MOD	AND	&			
2	+	-	OR	|					
1	<	<=	=	>=	>	<>	¬=	IN	

Procedure call
BEGIN statement;... END
WHILE Boolean expression DO statement
REPEAT statement;... UNTIL Boolean expression
FOR id := expression1 [TO | DOWNTO] expression2
DO statement
The control variable must be of non-real scalar type.
IF Boolean expression THEN statement [[;] ELSE statement]
CASE expression OF [constant,... : statement] ;... END
<> : denotes the "default case".
GOTO label
WITH variable,... DO statement

A statement may be empty.
Statements may be labeled: label : statement
Only one label may appear per statement.
label is an integer such that 0<label<=9999.
Boolean expression is an expression yielding the value TRUE or FALSE.

Types

```
simpletype
ARRAY (. simpletype,... .) OF type
ARRAY ( simpletype,... ) OF type
RECORD fieldlist END
SET OF simpletype
FILE OF type
@typeid

fieldlist is  [ id,... : type ] ;...
              [ CASE [ id : ] typeid OF
                 { constant,... : [ ( fieldlist ) ] } ;... ]
simpletype is  INTEGER | CHAR | BOOLEAN | ( id,... ) |
               constant .. constant |
               any typeid designating one of these.
               REALs are "simple" in some contexts.
FILEs may not be components of RECORDs, ARRAYs, or FILEs.

All objects must be declared before they are used except:
    Pointers  where the pointed to object is defined later;
    @typeid  where, "typeid" is defined later;
    Procedures may make forward references so long as the
         referenced procedure has been declared "FORWARD"
         or "FORTRAN" before the reference is made.
```

MTS Commands for PASCAL/UBC

```
$RUN PASC:GO SCARDS=source SPUNCH=object SPRINT=listing
     PAR=options fileassignments b ; comments

$RUN PASC:COMP          -ditto-
$RUN object+PASC:MON    -ditto-
```

options (separated by commas)

```
BATCH        - disable all interactive features
DUMP         - produce dump of PSW and registers on run error
EX=#         - maximum number execution stack memory pages
GO           - no compile; execute file  -P.OBJ
GO=fdname    - no compile; execute file  fdname
LI=fdname    - use file  fdname  as a run time library
NERRS=#      - maximum number of allowed run time errors
NEW=#        - maximum number 'NEW' stack memory pages
NOGO         - compile only; no execution
NOPMD        - omit post mortem dump on run error
PAGES=#      - maximum number of printed pages
TIME=#       - maximum amount of cpu time
TR=fdname    - use the translator in file  fdname

# = a positive integer        fdname = a file or device name

fileassignment is  [ b  PASCALname=MTSname ]*
```

Compiler Options

Compiler options appear in comments as follows:

(*$list of options separated by commas*)

e.g., "$A+,B-,C-,D-,E-,K+,L+,N-,P-,Q-,S-,T+,U+,X+,7+,_+"

Option	Default	Meaning
A	+ on	Perform subrange checking on Assignments
B	- off	Allow Byte allocation. Default is half-word
C	- off	Print object Code as each statement is processed
D	- off	Output snapshot & post mortem dump Debug tables
E	- off	Eject; force current line to start a new page
K	+ on	Forces an error if Case index out of range
L	+ on	List source program
N	- off	Allow Non-alignment of data
P	- off	Print object code after each procedure
Q	- off	SeQence nos.: on = cols 1-72; off = cols 1-100
S	- off	Force Standard Revised PASCAL
T	+ on	Force Testing: = A, K, X
U	±	Auto Underlining: on in batch; off interactively
X	+ on	Check indeX range in subscripts
7	+ on	Compile code for IBM 370
_	+ on	Permit underbar as an alphabetic character

Standard Functions and Procedures

```
ABS(IR) : IR          - func: absolute value
ARCTAN(R) : R         - func: arctangent
CHR(I) : C            - func: convert integer to character
COS(R) : R            - func: cosine
EOF[ (F) ] : B        - func: end-of-file
EOLN[ (F) ] : B       - func: end-of-line
EXP(R) : R            - func: exponential
GET(F)                - proc: input next object
HALT                  - proc: terminate execution
INSERT(I,J,K) : I     - func: shift I left J bits, 'OR' with K
LINENO(F) : I         - func: last read line number of file F
LN(R) : R             - func: natural logarithm
NEW(P[,T,...])        - proc: instantiate a new P
ODD(I) : B            - func: true if I odd
ORD(C) : I            - func: convert character to integer
PACK(A,S,D)           - proc: FOR J:=U TO V DO D(J):=A(J-U+I)
PAGE[ (F) ]           - proc: start new page on F
PRED(S) : S           - func: predecessor
PUT(F)                - proc: output next object
READ[ (F[,O,...]) ]   - proc: read values
READLN[ (F[,O,...]) ] - proc: read values; to EOL
RESET(F[,G])          - proc: rewind F; attach to fdname G
REWRITE(F[,G])        - proc: close F; attach to fdname G
ROUND(R) : I          - func: round to nearest integer
```

```
SIN(R) : R              - func: sine
SQR(IR) : IR            - func: square
SQRT(R) : R             - func: square root
SUBSTR(A,S,N) : B       - func: substring of A start=I, length=N
SUCC(S) : S             - func: successor
TRUNC(R) : I            - func: truncate to integer
UNPACK(D,A,S)           - proc: FOR J:=U TO V DO A(J-U+I):=D(J)
WRITE[ (F[ ,O,... ]) ]  - proc: write values
WRITELN[ (F[ ,O,... ]) ]- proc: write values; terminate line

I,J,K = integer                 R = real
IR = integer,real               B = Boolean
S = scalar                      C = character
F = file                        P = pointer
N = integer constant            T = optional tag
A = ARRAY(.U..V.) OF CHAR   (a string)
D = ARRAY(.W..X.) OF CHAR   (a string)
G = ARRAY(1..N) OF CHAR   (a [logical] file name b)
O = any variable except file (input)
    any expression except file (output)
```

Predefined Objects

```
CONST    MAXINT  = 2147483647;

TYPE     BOOLEAN = (FALSE,TRUE);
         TEXT    = FILE OF CHAR;
         ALFA    = ARRAY(1..10) OF CHAR;

VAR      INPUT,OUTPUT : TEXT;
         RCODE        : -32767..+32767;
        "Return Code -- automatically set on return from a
         FORTRAN routine.  May be set by the user as well."
```