CUT HERE FOLD HERE

UCSD PASCAL Quick Reference Card

by

David Fox and Mitch Waite

Declaration/Structure
- CONST
- TYPE
- VAR
- BEGIN
- PROCEDURE
- FUNCTION
- USES
- PROGRAM
- SET
- END

Input/Output
- PAGE
- READ
- READLN
- WRITE
- WRITELN

Constant Identifiers
- FALSE
- TRUE
- MAXINT

Numeric Functions
- ABS
- ATAN
- ARCTAN
- COS
- EXP
- LN
- LOG
- ROUND
- SIN
- SQR
- SQRT
- TRUNC

Types
- BOOLEAN
- CHAR
- INTEGER
- LONG INTEGER
- REAL
- STRING

Ordinal Functions
- ORD
- PRED
- SUCC

Flow of Control
- CASE
- FOR-DO/DOWNTO
- IF-THEN
- IF-THEN-ELSE
- REPEAT-UNTIL
- WHILE
- GOTO
- EXIT

Misc Functions
- CHR
- ODD

Operators
- DIV
- AND
- NOT
- MOD
- OR
- IN

String Functions and Procedures
- CONCAT
- COPY
- DELETE
- INSERT
- LENGTH
- POS
- STR

SPECIAL CHARACTERS

(*	used to start a comment
*)	used to end a comment
{	used to start a comment
}	used to end a comment
[ ]	used in array declarations, to surround subscripts, sets
. .	used to indicate range in subrange types, arrays and sets

ALGEBRAIC OPERATORS

Symbol	*Description*	*Operand Type**	*Result Type**
+	Addition	I or R	I or R
	Set union	Any Set type	Same as operand
−	Subtraction	I or R	I or R
	Set difference	Any SET type	Same as operand
*	Multiplication	I or R	I or R
	Set intersection	Any SET type	Same as operand
/	REAL division	I or R	R
DIV	INTEGER division	I	I
MOD	Modulus (A MOD B yields the remainder when dividing A by B)	I	I
:=	Assigns value to		

* I = INTEGER, R = REAL

RELATIONAL OPERATORS

=	Equal
<>	Not equal
<	Less than
>	Greater than
<=	Less than or equal
>=	Greater than or equal
NOT	Logical "Not"
AND	Logical "And"
OR	Logical "Or"
IN	SET membership

PROGRAM STRUCTURE

```
PROGRAM ProgName;                 Declares name of program
  Declarations

PROCEDURE Proc1Name;              Declares name of a procedure

  Declarations
BEGIN
  :
END;

FUNCTION Func1Name;
  Declarations
BEGIN
  :                               Declare name of a function
END;

BEGIN (* Main Program *)          Main program section begins
  :
END. (* ProgName *)               Main program section ends
                                  (note period after final END)
```

Program or Block Declarations

```
CONST   Const1Name  = constant;
        Const2Name  = constant;
              :
        ConstNName = constant;

TYPE    Type1Name  = type;
        Type2Name  = type;
              :
        TypeNName = type;

VAR     Var1Name, Var2Name: type;
        Var3Name          : type;
              :
        VarNName          : type;
```

Procedure Parameter List

```
PROCEDURE   ProcName(Val1Param, Val2Param : type;
                     VAR  Var1Param       : type;
                     Val3Param            : type);
```

Function Parameter List

```
FUNCTION  FuncName(Val1Param,              : type;
                   Val2Param,Val3Param : type;
                   VAR  Var1Param      : type) : type;
```

NAMING CONVENTIONS

1. Names start with a letter.
2. Characters that follow must be either letters or numbers.
3. Only first eight characters are guaranteed to be recognized by the computer.
4. Names may *contain* Pascal "reserved words" but can't *be* reserved words.
5. Variations in different versions of Pascal (UPPER and lower case, other characters might be legal).

STANDARD (BUILT-IN) IDENTIFIERS

Constants

FALSE and TRUE	Boolean values
MAXINT	Maximum integer value

Types

The types with an asterisk (*) are available in UCSD Pascal:

BOOLEAN	CHAR	INTEGER
LONG INTEGER*	REAL	STRING*

FUNCTIONS

Numeric Functions

Name	*Parameter Type**	*Result Type**	*Description*
ABS(x)	I or R	Same as param	Returns absolute value of *x*

ATAN(x) or ARCTAN(x)	I or R	R	Returns the inverse tangent of *x* in radians
COS(Angle)	I or R	R	Returns the cosine of *Angle*
EXP(x)	I or R	R	Returns e to the xth power (e^x)
LN(x)	I or R	R	Returns the natural logarithm of *x* (*x* must be greater than 0)
LOG(x)	I or R	R	Returns the Logarithm to the base 10 of *x*
ROUND(x)	R	I	Round off *x* to the nearest integer
SIN(Angle)	I or R	R	Returns the sine of *Angle*
SQR(x)	I or R	Same as param	Returns *x* squared (x^2)
SQRT(x)	I or R	R	Returns the square root of *x* (*x* must be positive)
TRUNC(x)	R or L	I	Converts *x* to integer without rounding

* I = INTEGER, R = REAL, L = LONG INTEGER

Ordinal Functions

Name	*Parameter Type**	*Result Type**	*Description*
ORD(x)	O	I	Returns the position which *x* holds in its data type
PRED(x)	O	Same as param	Returns the predecessor of *x*†
SUCC(x)	O	Same as param	Returns the successor of *x*†

I = INTEGER, O = Ordinal
† if none exists, there will be an error

Other Functions

Name	*Parameter Type**	*Result Type*	*Description*
CHR(x)	I	CHR	Returns a character which has the ASCII value *x*
ODD(x)	I	BOOLEAN	Returns *TRUE* if *x* is odd, otherwise returns *FALSE*

* I = INTEGER

String Functions and Procedures

In the following String intrinsics, the parameters *StartPos, Pos* and *Size* are *INTEGERs.* All other parameters are *STRINGs.*

Name	*Result Type**	*Description*
CONCAT(Strl, Str2, ..., StrN)	S,F	Returns a new string which is the concatenation of *Str1* through *StrN*
COPY(SourceStr, StartPos, Size)	S,F	Copies from *SourceStr* beginning at *StartPos* taking *Size* characters
DELETE(SourceStr,StartPos,Size)	P	Removes *Size* characters from *SourceStr* beginning at *StartPos*
INSERT(Source, Dest, Pos)	P	Inserts *Source* into *Dest* at *Pos*
LENGTH(Str)	I,F	Returns the length of *Str*
POS(Pattern, SourceStr)	I,F	Returns the position of the first occurrence of *Pattern* in *SourceStr*
STR(x, DestStr)	P	Converts *x* (either an I or a *LONG INTEGER*) to a *STRING.* Result is assigned to *DestStr*

* I = INTEGER, S = STRING, F = Function, P = Procedure

INPUT/OUTPUT INTRINSIC PROCEDURES

PAGE(OUTPUT);	Causes the screen to clear.
READ(Char1);	If *Char1* is a *CHAR* type variable, *READ* will accept a single character without having to press RETURN.
READLN(Var1);	Accepts data from keyboard and places in Var1 (requires RETURN keypress)*.
WRITE(Var1);	Prints parameter on screen and leaves cursor at end of line (no carriage return/linefeed issued)*. (See *WRITELN* for more examples.)
WRITELN(Var1);	Prints data on screen (with carriage return/linefeed)*.
WRITELN(Var1, Var2, . . ., *VarN*);	Printing multiple variables
WRITELN('Here"s a string:', String1);	Printing literals
WRITELN(IntNum : 4, RealNum : 7 : 2);	Using formatted printing

* Var1 – *VarN* can be of type *CHAR, INTEGER, LONG INTEGER, REAL, STRING*

FLOW OF CONTROL COMMANDS

In the following examples, any statement may be substituted by a Compound Statement.

Command	*Description*

CASE	Use when you want to select one of many statements to execute. The *statement* following the *constant* which matches the value of the *case-index* is executed. *Constant-list* is a list of constants separated by commas.
	CASE case-index OF
	  constant-list : statement;
	  constant-list : statement;
	  :
	  constant-list : statement;
	END;

EXIT	Use to prematurely leave a procedure or function. *EXIT(ProcName);*
FOR	Use when you want to repeat a statement(s) a specific number of times. *FOR control-value : = initial-value TO* *final-value DO statement;* *FOR control-value : = initial-value DOWNTO* *final-value DO statement;*
IF-THEN	Use when you want to execute a statement(s) only if a specific condition is true. *IF condition THEN statement;*

IF-THEN-ELSE	Use when you want to execute one of two statements.
	IF condition THEN statement1
	  ELSE statement2;

REPEAT-UNTIL	Use when you want to repeat a statement(s) until a specific condition is true. Statement will execute at least once.
	REPEAT
	  statement1;
	  statement2;
	  :
	  statementN;
	UNTIL condition;

WHILE	Use when you want to repeat a statement(s) only while a specific condition is true. Statement(s) may not execute at all if condition starts out false. *WHILE condition DO* *statement;*

RESERVED WORDS

The words with an asterisk (*) following them are not covered in this book:

AND	ELSE	MOD	RECORD*	VAR
ARRAY	END	NIL*	REPEAT	WHILE
BEGIN	FILE*	NOT	SET	WITH*
CASE	FOR	OF	THEN	
CONST	FUNCTION	OR	TO	
DIV	GOTO*	PACKED*	TYPE	
DO	IF	PROCEDURE	UNTIL	
DOWNTO	LABEL*	PROGRAM	USES	