

PASCAL USERS GROUP

*Rm 217 Brouse Copy*

# Pascal News

NUMBER 20

COMMUNICATIONS ABOUT THE PROGRAMMING LANGUAGE PASCAL BY PASCALERS  
DECEMBER, 1980

7185.1

Recommended?

Return to:

PASCAL USERS GROUP  
P.O. Box 888524  
Atlanta, GA 30338

Return postage guaranteed  
Address Correction requested

Bulk Rate  
U.S. Postage  
PAID  
Atlanta, Ga.  
Permit No. 2854

ATIN: ROOM 217 BROUSE COPY [81]  
UNIV. OF MINNESOTA  
UCC : 227EX  
MINNEAPOLIS, MN 55455

POLICY: PASCAL NEWS

(15-Sep-80)

- \* Pascal News is the official but informal publication of the User's Group.
- \* Pascal News contains all we (the editors) know about Pascal; we use it as the vehicle to answer all inquiries because our physical energy and resources for answering individual requests are finite. As PUG grows, we unfortunately succumb to the reality of:
  1. Having to insist that people who need to know "about Pascal" join PUG and read Pascal News - that is why we spend time to produce it!
  2. Refusing to return phone calls or answer letters full of questions - we will pass the questions on to the readership of Pascal News. Please understand what the collective effect of individual inquiries has at the "concentrators" (our phones and mailboxes). We are trying honestly to say: "We cannot promise more than we can do."
- \* Pascal News is produced 3 or 4 times during a year; usually in March, June, September, and December.
- \* ALL THE NEWS THAT'S FIT, WE PRINT. Please send material (brevity is a virtue) for Pascal News single-spaced and camera-ready (use dark ribbon and 18.5 cm lines!)
- \* Remember: ALL LETTERS TO US WILL BE PRINTED UNLESS THEY CONTAIN A REQUEST TO THE CONTRARY.
- \* Pascal News is divided into flexible sections:
  - POLICY - explains the way we do things (ALL-PURPOSE COUPON, etc.)
  - EDITOR'S CONTRIBUTION - passes along the opinion and point of view of the editor together with changes in the mechanics of PUG operation, etc.
  - HERE AND THERE WITH PASCAL - presents news from people, conference announcements and reports, new books and articles (including reviews), notices of Pascal in the news, history, membership rosters, etc.
  - APPLICATIONS - presents and documents source programs written in Pascal for various algorithms, and software tools for a Pascal environment; news of significant applications programs. Also critiques regarding program/algorithm certification, performance, standards conformance, style, output convenience, and general design.
  - ARTICLES - contains formal, submitted contributions (such as Pascal philosophy, use of Pascal as a teaching tool, use of Pascal at different computer installations, how to promote Pascal, etc.).
  - OPEN FORUM FOR MEMBERS - contains short, informal correspondence among members which is of interest to the readership of Pascal News.

IMPLEMENTATION NOTES - reports news of Pascal implementations: contacts for maintainers, implementors, distributors, and documentors of various implementations as well as where to send bug reports. Qualitative and quantitative descriptions and comparisons of various implementations are publicized. Sections contain information about Portable Pascals, Pascal Variants, Feature-Implementation Notes, and Machine-Dependent Implementations.

----- ALL-PURPOSE COUPON ----- (15-Sep-80)

Pascal User's Group, c/o Rick Shaw  
 P.O. Box 888524  
 Atlanta, Georgia 30338 USA

**\*\*NOTE\*\***

- Membership fee and All Purpose Coupon is sent to your Regional Representative.
- SEE THE POLICY SECTION ON THE REVERSE SIDE FOR PRICES AND ALTERNATE ADDRESS if you are located in the European or Australasian Regions.
- Membership and Renewal are the same price.
- Note the discounts below, for multi-year subscription and renewal.
- The U. S. Postal Service does not forward Pascal News.

		USA	Europe	Aust.
[ ] Enter me as a new member for:	[ ] 1 year	\$10.	£6.	A\$ 8.
[ ] Renew my subscription for:	[ ] 2 years	\$18.	£10.	A\$ 15.
	[ ] 3 years	\$25.	£14.	A\$ 20.
[ ] Send Back Issue(s)	! _____ !			
[ ] My new address/phone is listed below				
[ ] Enclosed please find a contribution, idea, article or opinion which is submitted for publication in the <u>Pascal News</u> .				
[ ] Comments:	_____			
	_____			

ENCLOSED PLEASE FIND: \$ _____
CHECK no. _____

NAME \_\_\_\_\_

ADDRESS \_\_\_\_\_

PHONE \_\_\_\_\_

COMPUTER \_\_\_\_\_

DATE \_\_\_\_\_

#### JOINING PASCAL USER'S GROUP?

- Membership is open to anyone: Particularly the Pascal user, teacher, maintainer, implementor, distributor, or just plain fan.
- Please enclose the proper prepayment (check payable to "Pascal User's Group"); we will not bill you.
- Please do not send us purchase orders; we cannot endure the paper work!
- When you join PUG any time within a year: January 1 to December 31, you will receive all issues of Pascal News for that year.
- We produce Pascal News as a means toward the end of promoting Pascal and communicating news of events surrounding Pascal to persons interested in Pascal. We are simply interested in the news ourselves and prefer to share it through Pascal News. We desire to minimize paperwork, because we have other work to do.

- American Region (North and South America): Send \$10.00 per year to the address on the reverse side. International telephone: 1-404-252-2600.
- European Region (Europe, North Africa, Western and Central Asia): Join through PUG (UK). Send £5.00 per year to: Pascal Users Group, c/o Computer Studies Group, Mathematics Department, The University, Southampton SO9 5NH, United Kingdom; or pay by direct transfer into our Post Giro account (28 513 4000); International telephone: 44-703-559122 x700.
- Australasian Region (Australia, East Asia - incl. Japan): PUG(AUS). Send \$A10.00 per year to: Pascal Users Group, c/o Arthur Sale, Department of Information Science, University of Tasmania, Box 252C GPO, Hobart, Tasmania 7001, Australia. International telephone: 61-02-23 0561 x435

PUG(USA) produces Pascal News and keeps all mailing addresses on a common list. Regional representatives collect memberships from their regions as a service, and they reprint and distribute Pascal News using a proof copy and mailing labels sent from PUG(USA). Persons in the Australasian and European Regions must join through their regional representatives. People in other places can join through PUG(USA).

#### RENEWING?

- Please renew early (before November and please write us a line or two to tell us what you are doing with Pascal, and tell us what you think of PUG and Pascal News. Renewing for more than one year saves us time.

#### ORDERING BACK ISSUES OR EXTRA ISSUES?

- Our unusual policy of automatically sending all issues of Pascal News to anyone who joins within a year means that we eliminate many requests for backissues ahead of time, and we don't have to reprint important information in every issue--especially about Pascal implementations!
- Issues 1 .. 8 (January, 1974 - May 1977) are out of print. (A few copies of issue 8 remain at PUG(UK) available for £2 each.)
- Issues 9 .. 12 (September, 1977 - June, 1978) are available from PUG(USA) all for \$15.00 and from PUG(AUS) all for \$A15.00
- Issues 13 .. 16 are available from PUG(UK) all for £10; from PUG(AUS) all for \$A15.00; and from PUG(USA) all for \$15.00.
- Extra single copies of new issues (current academic year) are: \$5.00 each - PUG(USA); £3 each - PUG(UK); and \$A5.00 each - PUG(AUS).

#### SENDING MATERIAL FOR PUBLICATION?

- Your experiences with Pascal (teaching and otherwise), ideas, letters, opinions, notices, news, articles, conference announcements, reports, implementation information, applications, etc. are welcome. Please send material single-spaced and in camera-ready (use a dark ribbon and lines 18.5 cm. wide) form.
- All letters will be printed unless they contain a request to the contrary.

PASCAL NEWS #20

DECEMBER, 1980

## Editor's Contribution

### RENEWING

This is the last issue of the year. (Bet you thought it would never get here!!) So if you have not renewed yet, RENEW NOW !!! It is easy to tell if you need to renew, because all you have to do is look at your mailing label. (Except in the Australasian Region.) If the number in square brackets says 80 (ie. "[80] ") then this is your last issue. This number is the year your subscription expires.

### THIS ISSUE

This issue contains the full text of the "Second Draft" of the proposed ISO Pascal Standard. I hope it is the last one we publish; because it is the last one! Andy Mickel (remember Andy?! ) was present at the X3J9 meeting in Huntsville, and has also been doing plenty of long distance politicking for this standard. He asked if he could write a guest editorial and the text follows.

Rick



UNIVERSITY OF MINNESOTA  
TWIN CITIES

University Computer Center  
227 Experimental Engineering Building  
208 Union Street S.E.  
Minneapolis, Minnesota 55455

1981-01-08

This special issue of Pascal News presents the second draft proposal of the ISO Pascal Standard now out for public comment and voting by the appropriate national bodies. More formally this document is known as (revised) DP7185.1.

[Alice Droogan, ISO TC97/SC5 Secretariat said to send all comment to:  
Joint Pascal Committee, c/o Larry B. Weber, IBM, General Products Division,  
555 Bailey Avenue, San Jose, CA 95150 USA. See also bottom of page 69,  
Pascal News #18]

As was reported by Jim Miner on page 74 of Pascal News #19, the first draft received 11 yes and 4 no votes. Most of the people I know associated with ISO Pascal Standards activities (including myself) expect unanimous approval on this draft. There are several things I can say about this:

1. The ISO Pascal standard is badly needed now and is overdue, but it will have set speed records in approval.
2. Even though the draft standard is imperfect (and always will be) the realization among those experts from the ISO Working Group is that extra time spent on the draft in an effort to perfect it has reached the point of diminishing returns.
3. This draft can be expected to be very close to the final standard.
4. Pascal users will at last benefit from a single standard when it will be adopted by the national standards groups in ISO member countries (such as in the USA by ANSI/IEEE/NBS and the Federal Govt.).

What I wrote two years ago in an editorial in Pascal News #14 which introduced the third BSI working draft of a Pascal Standard still applies:

Pascal Standards should be given special consideration (in other words, there are not necessarily applicable precedents found in the standards processes of other languages). The Pascal Standards process has been a model phenomenon in Computer Science history.

First and foremost Pascal was designed by a single person (Niklaus Wirth) and is not a committee-designed language. Pascal Standards Committees have so far rightly refrained from adding committee-designed features. Secondly, Pascal is the first major programming language standardized outside the United States. As I've said before, it has European origins but to be more accurate, Pascal is truly international. I think that's wonderful and neat!

Pascal is in very wide use (even though there are dozens of programmers ignorant of its impact and uses). Its design goals mentioned in my Pascal News #14 editorial have been met and exceeded (even though there are plenty of computing people who deny this).

Finally, let me reiterate the implications of an imperfect Pascal standard. In

the time given, with the people involved, and with the resources we've had, it's a remarkable achievement. (Thank you, Tony Addyman!) And it is still imperfect. But now the existence of a finished standard is more important than spending any more time.

In spite of the attitude of many of us technical people, you can't always fix certain things--technical problems don't always have clean solutions. It's not clear in some cases that solutions can be attained. In other words, if you put enough constraints on a problem, it could be the case that the set of solutions is empty.

Therefore, regarding the conformant-array feature I am happy; after having listened to the large volume of discussion, I know that it is equivalent in quality to any alternative. To repeat a familiar refrain, if there had been a natural solution, Niklaus would have incorporated it in the first place.

He's said so himself.



american national standards institute, inc · 1430 broadway, new york, n.y. 10018 · (212) 354-3300

Cable: Standards, New York

International Telex: 42 42 96 ANSI UI

January 21, 1981

Dear Mr. Shaw:

Enclosed please find second draft proposal ISO/DP 7185 - Specification for the Computer Programming Language - Pascal. This document is being circulated to 97/5 committee members for voting on by March 31, 1981.

Comments on the document are welcome and will be considered but must be in written form and must be received by March 31, 1981. Please address all comments to ANSI's X3J9 Chairman:

Dr. Carol Sledge  
On-Line Systems, Inc.  
115 Evergreen Heights Drive  
Pittsburgh, PA 15229

Comments should be clearly marked with the name, address and telephone number of the commentor, the section and subsection to which the comment applies, and a rationale or explanation for any proposed text changes. Specific proposed text changes are the most desirable form for comments, but general changes or criticisms, or questions, are also welcome.

Sincerely,

*Alice Droogan*  
Alice Droogan  
Secretariat ISO/TC 97/SC 5

AD/MAC  
Encl.

*- J. L. J.*

## DP7185 SPECIFICATION FOR THE COMPUTER PROGRAMMING LANGUAGE Pascal

CONTENTS	Page
Foreword	1
0. Introduction	2
1. Scope of this standard	2
2. References	2
3. Definitions	3
4. Definitional Conventions	3
5. Compliance	4
5.1 Processors	4
5.2 Programs	5
6. Requirements	6
6.1 Lexical Tokens	6
6.2 Blocks, scope, activations	9
6.3 Constant-definitions	11
6.4 Type-definitions	12
6.5 Declarations and denotations of variables	24
6.6 Procedure and function declarations	28
6.7 Expressions	45
6.8 Statements	51
6.9 Input and output	59
6.10 Programs	65
6.11 Hardware representation	67
APPENDICES	
A. Collected syntax	69
B. Index	77
C. Required Identifiers	83
TABLES	
1. Metalanguage symbols	3
2. Dyadic arithmetic operations	47
3. Monadic arithmetic operations	47
4. Set operations	48
5. Relational operations	49
6. Alternative symbols	68

## Foreword

The language Pascal was designed by Professor Niklaus Wirth to satisfy two principal aims:

- to make available a language suitable for teaching programming as a systematic discipline based on certain fundamental concepts clearly and naturally reflected by the language.
- to define a language whose implementations could be both reliable and efficient on then available computers.

## Second Draft Proposal

However, it has become apparent that Pascal has attributes which go far beyond these original goals. It is now being increasingly used commercially in the writing of both system and application software. This standard is primarily a consequence of the growing commercial interest in Pascal and the need to promote the portability of Pascal programs between data processing systems.

In drafting this standard the continued stability of Pascal has been a prime objective. However, apart from changes to clarify the specification, two major changes have been introduced:

- the syntax used to specify procedural and functional parameters has been changed to require the use of a procedure or function heading, as appropriate (see 6.6.3.1). This change was introduced to overcome a language insecurity;
- a fifth kind of parameter, the conformant array parameter, has been introduced (see 6.6.3.7). With this kind of parameter, the required bounds of the index-type of an actual parameter are not fixed, but are restricted to a specified range of values.

## 0. INTRODUCTION

The appendices are included for the convenience of the reader of this standard. They do not form a part of the requirements of this standard.

## 1. SCOPE OF THIS STANDARD

1.1 This standard specifies the semantics and syntax of the computer programming language Pascal by specifying requirements for a processor and for a conforming program. Two levels of compliance are defined for both processors and programs.

1.2 This standard does not specify

- the size or complexity of a program and its data that will exceed the capacity of any specific data processing system or the capacity of a particular processor;
- the minimal requirements of a data processing system that is capable of supporting an implementation of a processor for Pascal;
- the method of activating the program-block or the set of commands used to control the environment in which a Pascal program is transformed and executed;
- the mechanism by which programs written in Pascal are transformed for use by a data processing system;
- the method for reporting errors or warnings;
- the typographical representation of a program published for human reading.

## 2. REFERENCES

None.

## Second Draft Proposal

## 3. DEFINITIONS

- 3.1 error. A violation by a program of the requirements of this standard whose detection by a processor is optional.
- 3.2 implementation-defined. Possibly differing between processors, but defined for any particular processor.
- 3.3 implementation-dependent. Possibly differing between processors and not necessarily defined for any particular processor.
- 3.4 processor. A compiler, interpreter, or other mechanism which accepts the program as input and either executes it, prepares it for execution, or both.

## 4. DEFINITIONAL CONVENTIONS

The metalanguage used in this standard to specify the syntax of the constructs is based on Backus-Naur Form. The notation has been modified from the original to permit greater convenience of description and to allow for iterative productions to replace recursive ones. Table 1 lists the meanings of the various meta-symbols. Further specification of the constructs is given by prose and, in some cases, by equivalent program fragments. Any identifier that is defined in clause 6 as the identifier of a predeclared or predefined entity shall denote that entity by its occurrence in such a program fragment. In all other respects, any such program fragment is bound by any pertinent requirement of this standard.

Table 1. Metalanguage symbols

Meta-symbol	Meaning
=	shall be defined to be
>	shall have as an alternative definition
:	alternatively
.	end of definition
{x}	0 or 1 instance of x
{x}	0 or more instances of x
{x y}	groupings: either of x or y
"xyz"	the terminal symbol xyz
meta-identifier	a non-terminal symbol

## Second Draft Proposal

A meta-identifier shall be a sequence of letters and hyphens beginning with a letter.

A sequence of terminal and non-terminal symbols in a production implies the concatenation of the text that they ultimately represent. Within 6.1 this concatenation is direct; no characters may intervene. In all other parts of this standard the concatenation is in accordance with the rules set out in 6.1.

The characters required to form Pascal programs are those implicitly required to form the tokens and separators defined in 6.1.

Use of the words of, in, contains and closest-contains when expressing a relationship between terminal or non-terminal symbols shall have the following meanings.

the x of a y: refers to the x occurring directly in a production defining y.

the x in a y: is synonymous with "the x of a y".

a y contains an x: refers to any x directly or indirectly derived from y.

the y closest-contains an x: that y which contains an x but does not contain another y containing that x.

These syntactic conventions are used in clause 6 to specify certain syntactic requirements and also the contexts within which certain semantic specifications apply.

## 5. COMPLIANCE

NOTE. There are two levels of compliance - level 0 and level 1. Level 0 does not include conformant array parameters. Level 1 does include conformant array parameters.

## 5.1 Processors

- A processor complying with the requirements of this standard shall:
- if it complies at level 0, accept all the features of the language specified in clause 6, except for 6.6.3.6(e), 6.6.3.7 and 6.6.3.8, with the meanings defined in clause 6;
  - if it complies at level 1, accept all the features of the language specified in clause 6 with the meanings defined in clause 6;
  - not require the inclusion of substitute or additional language elements in a program in order to accomplish a feature of the language that is specified in clause 6;
  - be accompanied by a document that provides a definition of all implementation-defined features;
  - detect any violation by a program of the requirements of this standard that is not designated an error;
  - treat each violation that is designated an error in at least one of the following ways:

## Second Draft Proposal

- 1) there shall be a statement in an accompanying document that the error is not reported;
  - 2) the processor shall have reported a prior warning that an occurrence of that error was possible;
  - 3) the processor shall report the error during preparation of the program for execution;
  - 4) the processor shall report the error during execution of the program, and terminate execution of the program.
- (g) be accompanied by a document that separately describes any features accepted by the processor that are not specified in clause 6. Such extensions shall be described as being 'extensions to Pascal specified by ISO7185: 198-'.  
 (h) be able to process in a manner similar to that specified for errors any use of any such extension;  
 (i) be able to process in a manner similar to that specified for errors any use of an implementation-dependent feature.

## 5.2 Programs

A program complying with the requirements of this standard shall:

- (a) if it complies at level 0, use only those features of the language specified in clause 6, except for 6.6.3.6(e), 6.6.3.7 and 6.6.3.8;
- (b) if it complies at level 1, use only those features of the language specified in clause 6;
- (c) not rely on any particular interpretation of implementation-dependent features.

NOTE. The results produced by the processing of a complying program by different complying processors are not required to be the same.

## 5. REQUIREMENTS

## 6.1 Lexical tokens

NOTE. The syntax given in this sub-clause (6.1) describes the formation of lexical tokens from characters and the separation of these tokens, and therefore does not adhere to the same rules as the syntax in the rest of this standard.

6.1.1 General. The lexical tokens used to construct Pascal programs shall be classified into special-symbols, identifiers, directives, unsigned-numbers, labels and character-strings. The representation of any letter (upper-case or lower-case, differences of font, etc) occurring anywhere outside of a character-string (see 6.1.7) shall be insignificant in that occurrence to the meaning of the program.

```
letter = "a"|"b"|"c"|"d"|"e"|"f"|"g"|"h"|"i"|"j"|"k"|"l"|"m"|"n"|"o"|"p"|"q"|"r"|"s"|"t"|"u"|"v"|"w"|"x"|"y"|"z" .
```

```
digit = "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9" .
```

6.1.2 Special-symbols. The special-symbols are tokens having special meanings and shall be used to delimit the syntactic units of the language.

```
special-symbol = "+"|"-"|"*"|"/"|"="|"<"|">"|"["|"]"|
                "."|"|"|"'"|"^|"{"|"}"|
                "<>"|"<="|">="|"="|"..|" word-symbol .
```

```
word-symbol = "and"|"array"|"begin"|"case"|"const"|"div"|"
              "do"|"downto"|"else"|"end"|"file"|"for"|"
              "function"|"goto"|"if"|"in"|"label"|"mod"|"
              "nil"|"not"|"of"|"or"|"packed"|"procedure"|"
              "program"|"record"|"repeat"|"set"|"then"|"
              "to"|"type"|"until"|"var"|"while"|"with" .
```

6.1.3 Identifiers. Identifiers may be of any length. All characters of an identifier shall be significant. No identifier shall have the same spelling as any word-symbol.

```
identifier = letter (letter | digit) .
```

## Examples:

```
X      time      readinteger W04  AlterHeatSettings
InquireWorkstationTransformation
InquireWorkstationIdentification
```

6.1.4 Directives. A directive shall occur only in a procedure-declaration or function-declaration. The directive forward shall be the only required directive (see 6.6.1 and 6.6.2). Other implementation-dependent directives may be provided. No directive shall have the same spelling as any word-symbol.

```
directive = letter (letter | digit) .
```

NOTE. On many processors the directive external is used to specify that the procedure-block or function-block corresponding to that procedure-heading or function-heading is external to the program-block. Usually it is in a library in a form to be input

## Second Draft Proposal

to, or that has been produced by, the processor.

6.1.5 Numbers. An unsigned-integer shall denote in decimal notation a value of integer-type (see 6.4.2.2). An unsigned-real shall denote in decimal notation a value of real-type (see 6.4.2.2). The letter "e" preceding a scale factor shall mean 'times ten to the power of'. The value denoted by an unsigned-integer shall be in the closed interval 0 to maxint (see 6.4.2.2 and 6.7.2.2).

```
digit-sequence = digit {digit} .
unsigned-integer = digit-sequence .
unsigned-real =
  unsigned-integer "." digit-sequence ["e" scale-factor] |
  unsigned-integer "e" scale-factor .
unsigned-number = unsigned-integer ; unsigned-real .
scale-factor = signed-integer .
sign = "+" ; "-" .
signed-integer = [sign] unsigned-integer .
signed-real = [sign] unsigned-real .
signed-number = signed-integer ; signed-real .
```

## Examples:

```
1e10      1      +100      -0.1      5e-3      87.35E+8
```

6.1.6 Labels. Labels shall be digit-sequences and shall be distinguished by their apparent integral values, that shall be in the closed interval 0 to 9999.

```
label = digit-sequence .
```

6.1.7 Character-strings. A character-string containing a single string-element shall denote a value of char-type (see 6.4.2.2). A character-string containing more than one string-element shall denote a value of a string-type (see 6.4.3.2) with the same number of components as the character-string contains string-elements. If the string of characters is to contain an apostrophe, this apostrophe shall be denoted by an apostrophe-image. Each string-character shall denote an implementation-defined value of char-type.

```
character-string = "\"" string-element
  {string-element} "\"" .
string-element = apostrophe-image ; string-character .
apostrophe-image = "'" .
string-character =
  one-of-a-set-of-implementation-defined-characters .
```

## Examples:

```
'A'      ';'      ''''
'Pascal'  'THIS IS A STRING'
```

## Second Draft Proposal

6.1.8 Token separators. The construct

```
"(" any-sequence-of-characters-and-separations-of-lines-not-
  containing-right-brace ")"
```

shall be a comment if the "(" does not occur within a character-string or within a comment. The substitution of a space for a comment shall not alter the meaning of a program.

Comments, spaces (except in character-strings), and the separation of consecutive lines shall be considered to be token separators. Zero or more token separators may occur between any two consecutive tokens, or before the first token of a program text. There shall be at least one separator between any pair of consecutive tokens made up of identifiers, word-symbols, labels or unsigned-numbers. No separators shall occur within tokens.

