

PASCAL USERS GROUP

# Pascal News

Communications about the Programming Language Pascal by Pascalers

Number 24

- Pascal Standards: Progress Report
- Status Report on Version 3.0
- WRITENUM — A Routine to Output Real Numbers
- TREEPRINT — A Package to Print Trees on Character Printers
- Three Proposals for Extending Pascal
- Announcements

Number

24

JANUARY 83

EX LIBRIS: David T. Craig  
736 Edgewater  
[# ] Wichita, Kansas 67230 (USA)

## POLICY: PASCAL NEWS

(Jan. 83)

- *Pascal News* is the official but *informal* publication of the User's Group.

**Purpose:** The Pascal User's Group (PUG) promotes the use of the programming language Pascal as well as the ideas behind Pascal through the vehicle of *Pascal News*. PUG is intentionally designed to be non political, and as such, it is not an "entity" which takes stands on issues or support causes or other efforts however well-intentioned. Informality is our guiding principle; there are no officers or meetings of PUG.

The increasing availability of Pascal makes it a viable alternative for software production and justifies its further use. We all strive to make using Pascal a respectable activity.

**Membership:** Anyone can join PUG, particularly the Pascal user, teacher, maintainer, implementor, distributor, or just plain fan. Memberships from libraries are also encouraged. See the ALL-PURPOSE COUPON for details.

- *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 15.5 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.

# Pascal News

Communications about the Programming Language Pascal by Pascalers

---

**JANUARY 1983**

**Number 24**

---

## **2 COMPILERS NOTES**

### **APPLICATIONS**

- 3** A Pascal Bibliography *By Tony Hayes*

### **PASCAL STANDARDS**

- 20** Pascal Standards: Progress Report *By Jim Miner*  
**20** Status Report on Version 3.0 of the Pascal Test Suite *By B.A. Wickmann*

### **ANNOUNCEMENTS**

- 23** Distribution of the Edison System  
**23** Pascal Chosen as Sil  
**23** Pascal: A Problem Solving Approach  
**24** Modula-2

### **ARTICLES**

- 25** WRITENUM — A Routine to Output Real Numbers  
*By Doug Grover and Ned Freed*  
**27** TREEPRINT — A Package to Print Trees on any Character Printer  
*By Ned Freed and Kevin Carosso*  
**32** Three Proposals for Extending Pascal *By R.D. Tennent*  
**32** The Where-Clause: A Proposed Extension to Pascal *By R.D. Tennent*  
**34** Proposals for Improved Exception Handling in Pascal *By R.D. Tennent*  
**37** The Definition Block: A Proposed Extension to Pascal *By R.D. Tennent*

## **40 OPEN FORUM**

## **42 IMPLEMENTATION NOTES COUPON**

## **45 SUBSCRIPTION COUPON**

## **47 LICENSE APPLICATION**

Hello

This is Pascal News and my name is Charlie Gaffney. Much has happened since I received my March #22-23 Issue. I am the publisher of USUS News. USUS is the UCSD p-System User Society. The p-system was developed to bring Pascal to micro computers. Our USUS News was modeled on Pascal News. We have a lot of information in USUS but it was a chore to read because of bad original and photo copy material used for printing.

I sought a typesetter and found we could typeset and print for only 10% increase in cost. This is a small premium cost to have a readable newsletter. We typeset in August and received many compliments so far.

I thought of our model Pascal News and called Rick Shaw to explain our (USUS) improvement and ask if he needed help.

But Rick had his own story to tell. The work at Pascal Users Group was not performed by a group but by one man, Rick Shaw. He was hard pressed to keep up with the business of PUG.

An offer had been made by the "Journal of Pascal & Ada" to take all pending articles and publish them.

I made a counter offer to maintain PUG as it is under new management. Rick thought that was a nice idea, but the problems would persist and PUG would fail either now or later. After three phone calls Rick decided to let me try.

The News will be typeset and I hope you approve of our new appearance. The articles

you submit may be in any format because they will now be typeset. It is possible to enlarge the program listings if they are submitted in a narrow format of 15.5 cm wide.

### **Business**

I have decided to pay a small business to update:

1. the member list
2. new and renew members
3. banking records

Membership costs have gone up but if you pay for two years the third year is free.

Back issues have tied up a great deal of money. We have articles and programs just waiting for you. Buy a set. Buy a complete set. Buy a set for your friends.

### **A little about me**

I am an electrician, and I work for Chevrolet in Parma, Ohio. I have no college education and no formal computer training. My experience with computers involved the purchase of a Western Digital microengine, 16 bit computer. The computer uses p-code as defined by UCSD p-System and directly implements the code without an interpreter. Pascal News and USUS News, and 25 text books, have been my teachers. I thank them and each of you.

Charlie

# A Pascal Bibliography

By Tony Heyes  
Blind Mobility Research Unit,  
Department of Psychology,  
University of Nottingham  
England

## Introduction

The Pascal Bibliography is a package of programs written in standard Pascal and should therefore be easily transported. It enables users to store references and to retrieve them either by AUTHOR name or by KEYWORD; or logical combinations of AUTHORS and KEYWORDS. The bibliography is designed for human use; it uses very explicit prompts.

## Design Philosophy

The bibliography consists of a collection of ITEMS. Each ITEM takes the form of:-

One line devoted to AUTHOR or ADDRESSEE names.

Two lines devoted to TITLE or ADDRESS.

Two lines devoted to LOCATION.

DATE ITEM NUMBER.

Two lines devoted to KEYWORDS.

For example:-

HEYES A.D.,FERRIS A.J.,ORLOWSKI R.J.  
COMPARISON BETWEEN TWO METHODS  
OF RESPONSE FOR  
AUDITORY LOCALISATION IN THE AZI-  
MUTH PLANE.  
J. ACOST. SOC. AMER., 58; 1336-1339

1975 260  
DEAFNESS,LOCALISATION,AUDITORY  
DISPLAYS  
STEREOPHONIC SOUNDS,KINAESTHESIS

If ITEMS are addresses the convention is to store the address on the two lines of title.

For example:-

BLOGGS J.B.  
Mr.J.B.Bloggs\13 Fishpond Rd.\ Beeston,  
Nottingham.\ NG7 2RD\ U.K.  
Tel 0602-251234

1980 27  
ADDRESS,CIRCULATION LIST,XMAS  
CARD

Note the use of the backslash [\] to indicate the start of a new line. Note also that additional information

such as the telephone number can be stored on the location lines. Note, finally, the date has little meaning in this context.

Items may be located by running the program "bibout".

Items may be APPENDED or CHANGED by running the program "bibin".

Both programs are well supplied with prompts and are very simple to use.

Since additions and changes require that the current DICTIONARY be recompiled and this takes time, the actual changes take place during the night. The instructions to implement the changes reside in a PENDING TRAY until the night time run. The user will remain unaware of this slight restriction unless he tries to locate an ITEM during the day on which the ITEM was loaded.

## Method of Use

The following assumed the use of the UNIX operating system. Login with your user name, give your password, respond to the first system prompt "%" with "cd bib", ie. change directory to "bib". In answer to the next system prompt, "%", you may select any one of the programs from within the package.

These are:-

- a) "bibbin" to enter new items or to change an ITEM.
- b) "bibout" to search the bibliography for an ITEM.
- c) "outdict" to produce a hard copy of the current DICTIONARY.
- d) "cat scratch lpr" to output a hard copy of the SCRATCH FILE.

NEW USERS SHOULD ASK IF THEY MAY HAVE ACCESS TO AN ESTABLISHED BIBLIOGRAPHY AND THEN TRY USING "bibout" TO LOCATE ITEMS OF INTEREST.

To logout respond to the system prompt "%" by typing "control Z".

## The Programs

- a) "bibin"  
The opening prompt allows the selection of one of the following options:-  
APPEND

The prompts should be sufficiently explicit, but note:-

- (1) Authors and keywords should be separated by commas. Since they are used in the dictionary they should not spill over the end of a line. They can be any length but only the first 20 characters are significant.
- (2) The terminal will probably be set to produce lower case letters. The program will automatically convert them to upper case. If you wish to override this, begin each line of text with a backslash [\].
- (3) The date must be a single integer e.g. 1980.
- (4) If addresses are to be stored use the two title lines, close pack but indicate new lines with a backslash [\].
- (5) A personal local storage reference may be kept on the second location line. It should be enclosed in square brackets; e.g. [BM760] means that a copy of this ITEM is in the BM library, entry number 760.

#### CHANGE

Answer the prompts but please take note of the following:-

- 1) You must know in advance the ITEM number of the ITEMS you require to change.
- 2) You have to retrieve the ITEMS from the bibliography so CHANGE is relatively slow; be patient. It saves time, if you are changing more than one ITEM to make the changes in numerical order of ITEM number.
- 3) You retrieve the ITEM to be changed from the bibliography, the changed ITEM goes into the PENDING TRAY. If you change the same ITEM more than once in a single day only the last version will survive.

#### SPECIAL FACILITY

This option moves the contents of the SCRATCH file into the PENDING tray. It can be used for moving ITEMS from one bibliography to another. Since SCRATCH is a text file, ITEMS may be changed using an editor and then loaded back into the PENDING tray. (Clever stuff!!).

#### b) "bibout"

The computer will count the ITEMS in the bibliography and then offer the option of producing a HARD COPY of the dictionary or doing a SEARCH for ITEMS.

#### SEARCH

You may either search by NUMBER or, more usually by using the DICTIONARY.

You may opt to send the results either to the TERMINAL or to the SCRATCH FILE for subsequent printing.

#### SEARCH by NUMBER

The search is terminated by asking to search for item number zero [0].

A block of ITEMS may be searched for by asking to search for item number minus one [-1]. You will then be asked for the lowest and the highest item numbers of the block.

#### SEARCH by DICTIONARY

You will be asked for a word i.e. an AUTHOR

name or a KEYWORD. The computer will look this up in the DICTIONARY and list the ITEM numbers of all ITEMS containing this word in their AUTHOR or KEYWORD string. If you are doing a single word search answer the next prompt will a full stop [.] , and then the instruction to LOOK UP. If, however, it is a multiple word search give the next word. Once again the corresponding ITEM number list will be printed out. The answer to the prompt "AND, OR or NOT" enables you to combine the current ITEM number list with the previous ITEM number list. For instance:-

AND Only numbers present in both lists are retained.

OR All numbers from both lists are retained.

NOT Numbers present in the current list are deleted from the previous list.

A new current list is printed out showing the results of the selection. The search sequence may be continued for any number of logical combinations of words. At any time a search for the ITEMS in the current list may be initiated by giving a full stop [.] . After which you may either LOOK UP the selected ITEMS or, if you have made a mistake in your list combinations simply RESTART. There is one special word, namely \*\*\*, this word will match all the dictionary.

#### c) "outdict"

No prompts and no option, simply type "outdict" in answer to the system prompt "%" to obtain a hard copy of the current DICTIONARY.

Note, you must have first prepared a copy of the DICTIONARY by running the appropriate HARD COPY option of "bibout".

#### d) "opr scratch"

This program is run to obtain the printed output from "bibout", provided the option had been chosen to send the output to the SCRATCH FILE.

No prompts and no options, simply type "opr scratch" in answer to the system prompt "%" to obtain a hard copy of the contents of the SCRATCH FILE.

N.B. If you would like to list the SCRATCH FILE to the terminal to check the contents then run "cat scratch".

#### Acknowledgements

I gratefully acknowledge the encouragement and support I have received from Roger Henry and Chris Blunsdon.

The bibliography was originally intended for use by the members of the BLIND MOBILITY RESEARCH UNIT it is however available to any members of the Pascal Users Group. Would anyone wishing to take up this offer please contact Tony Heyes to arrange medium of transportation.

#### NOTES FOR IMPLEMENTORS

The following notes outline the steps the imple-

menter should take in order to establish a new bibliography. After this groundwork, the user can use the shell commands *bibin*, *bibout*, and *outdict* to build and manipulate the bibliography.

1. The bibliography system requires 6 workfiles named b1 to b6. The recommended practice is for the user to devote a directory to the bibliography, say 'user/bib'. The workfiles can be created easily using the cat command. E.g

```
cat > b1      Z
```

File b3 requires a link named scratch. This can be created by the command —

```
ln b3 scratch
```

2. b6 is used as a temporary scratch file during the overnight run. It grows to be as large as b1. If there is insufficient room on the user's disc b6 may be coerced on to another disc.
3. The bib directory must contain the following shell commands:-

```
bibin      Bibin.out b1 b2 b3 b4 b5
bibout     Bibout.out b1 b2 b3 b4 b5
bibupdate  Bibupdate.out b1 b2 b3 b4 b5 b6
outdict    (1pr b4;rm b4;>b4)&
```

4. Finally, an entry must be made in the UNIX table 'crontab' so that bibupdate will be executed during the night.

```
program Bibin(input,output,bank,dict,scratch,dlist,PendingTray);
(* To ADD, CHANGE or REMOVE items,
instructions left in a PendingTray file 'pending',
actual changes made by running "Bibupdate.p" *)
(* written by Tony Heyes, Blind Mobility Research Unit,
Department of Psychology, The University,
Nottingham, U.K. *)

label 10;

const   LineLn = 70;
        RowLn  = 20;
        HiTag  = 10000;
        NonDate = -1066;

type    string = packed array [1..LineLn] of char;
        item = record
            authors,title1,title2,
            place1,place2 : string;
            date          : integer;
            key1,key2     : string
        end;
        word = packed array [1..20] of char;
        row = array [1..RowLn] of integer;
        dic = record
            name      : word;
            numbers  : row;
            cont      : boolean
        end;
        TagItem = record
            tag : integer;
            entry : item
        end;

var     empty,entry : item;
        bank : file of item;
        PendingTray,TempPendingTray : file of TagItem;
        dlist,scratch : text;
        dict : file of uic;
        TagEntry : TagItem;
        ch,AppendOption,ChangeOption,MainOption,HelpOption,
        SpecialOption : char;|chge : boolean;
        a,n,nn,count : integer;

procedure InlChar (var ch : char);
(* to read the first character of a word typed into the terminal *)
begin
    ch := input^;
    while not (ch in ['A'..'Z','a'..'z']) do
```

```
begin (* skips along until first character found *)
    get(input);
    if eoln(input)
    then
        begin
            writeln;
            write('ERROR: character required .... ')
            end;
        ch := input^
    end;
    while not eoln(input) do (* skips over rest of line *)
        get(input)
    end; (* of InlChar *)

procedure InlInt (var int : integer);
(* to read an integer and not cause a fatal error if a
character is given *)
var ch : char;
    a,OrdZero : integer;
    NegFound : boolean;
begin
    repeat (* skips along until integer is found *)
        get(input);
        if eoln(input)
        then
            begin
                writeln;
                write('ERROR: digit required .... ')
                end;
            ch := input^
        until ch in ['-','+','0'..'9'];
        if ch='- '
        then
            begin
                NegFound := true;
                get(input);
                ch := input^
            end
        else
            begin
                NegFound := false;
                if ch='+'
                then
                    begin
                        get(input);
                        ch := input^
                    end
                end;
            a := 0;
            OrdZero := ord('0');
            repeat
                a := 10*a+ord(ch)-OrdZero;
                get(input);
                ch := input^
            until not (ch in ['0'..'9']);
            while not eoln(input) do (* skips over rest of line *)
                get(input);
            if NegFound
            then
                int := -a
            else
                int := a
            end; (* of InlInt *)

        procedure VDUinString(var str : string);
        (* to input from terminal *)

        var i,n : integer;
            ch : char;
            AllCaps : boolean;
        begin
            n := 0;
            AllCaps := true;
            repeat
                n := n+1;
                read(ch);
                if (n=1) and (ch=' ')
                then
                    n := 0;
                if (n=1) and (ch='\')
                then
                    begin (* defeat automatic shift with '\ ' *)
                        AllCaps := false;
                        n := 0
                    end;
                if n<0
                then
                    begin
                        if AllCaps
                        then
                            if ch in ['a'..'z']
```

```

        then
            ch := chr(ord(ch)-32);
        str[n] := ch
    end
until eoln(input);
for i:=n+1 to LineLn do
    str[i] := ' '
end; (* of VDUinString *)

procedure ScratchInStr(var str : string);
(* input from file scratch *)
var n,i : integer;
    ch : char;
begin
    if not eof(scratch)
    then
        begin
            n := 0;
            repeat
                read(scratch,ch);
            until (ch=';') or (eof(scratch));
            while (not eoln(scratch)) do
                begin
                    read(scratch,ch);
                    n := n+1;
                    str[n] := ch;
                end;
                if n+1<=LineLn
                then
                    for i:=n+1 to LineLn do
                        str[i] := ' ';
                    end
                end;
            end;
        end; (* of ScratchInStr *)

function ScratHoldsItems : boolean;
(* to inspect the SCRATCH FILE and check that ITEMS are complete *)
var count,LineNo : integer;
    FaultFound,HeadingError,NegFound : boolean;
procedure CheckLine;
var CharCount : integer;
    LineTooLong,BadLine : boolean;
begin
    LineNo := LineNo + 1;
    CharCount := 1;
    BadLine := false;
    LineTooLong := false;
    get(scratch);
    while (not eoln(scratch)) and (CharCount < LineLn + 9) do
        begin
            get(scratch);
            CharCount := CharCount + 1;
            if (CharCount = 9) and (scratch^ <> ':') then
                BadLine := true;
            end;
            if CharCount < 9 then BadLine := true;
            while not eoln(scratch) do
                begin
                    get(scratch);
                    if scratch^ <> ' ' then LineTooLong := true
                end;
            end;
            if BadLine then
                begin
                    FaultFound := true;
                    writeln('Line',LineNo : 4,' bad line ':' missing.')

```

```

begin
    if not NegFound then (* no ITEMS present *)
        begin
            HeadingError := true;
            writeln('SCRATCH does not contain ITEMS.')

```

```

        writeln(placel);
        writeln(place2);
        writeln(date:8,'          Item number :',n :5);
        writeln(key1);
        writeln(key2)
    end
end; (* of OutRecord *)

procedure GetReference(n : integer);
(* to count through bank to find an ITEM *)
begin
    if n<count
    then
        begin
            reset(bank);
            count := 1
        end;
    while (count < n) and (not eof(bank)) do
        begin
            count := count+1;
            get(bank)
        end;
    if eof(bank)
    then
        begin
            writeln;
            writeln(' You have only got',count -1,' Items.');
            writeln;
            goto l0
        end
    else
        OutRecord(bank^,n)
    end; (* of GetReference *)

procedure change(var entry : item; m : integer);
(* to change the mth. ITEM *)
var line : integer;
    DHOOption,LineOption : char;
    str : string;
begin
    writeln;
    writeln;
    repeat
        write('Do you wish to DELETE or MODIFY .... ');
        InlChar(DHOOption)
        until DHOOption in ['D','d','H','h','M','m'];
        if DHOOption in ['D','d']
        then
            begin
                empty;
                entry := empty
            end
        else
            begin
                writeln;
                writeln('You may REPLACE a line,');
                writeln('move to the NEXT line,');
                writeln('or SKIP to the end of the item. ');
                writeln;
                line := 0;
                repeat
                    line := line+1;
                until
                    write('Do you wish to APPEND, to CHANGE, ');
                    writeln('to use the SPECIAL facility, ');
                    write('or to FINISH .... ');
                    InlChar(MainOption)
                    until MainOption in ['A','a','C','c','S','s','F','f'];
                (* MainOption= S is a special facility,
                used for loading from 'scratch' to 'PendingTray' *)

                case MainOption of
                    'A','a': (* TO APPEND *)
                        begin
                            writeln;
                            repeat
                                write('Do you need help
                                [YES or NO] .... ');
                                InlChar(HelpOption)
                                until HelpOption in ['Y','y','N','n'];
                                if HelpOption in ['Y','y']
                                then
                                    begin
                                        writeln;
                                        writeln('NOTES.');
                                        write('(a) Authors and keywords separated');
                                        writeln(' by a comma ",,");
                                        write('(b) To remove the automatic conversion to ');
                                        writeln('upper case letters');
                                        write(' begin a line of text with');
                                        writeln(' a backslash "\.');

```

```

        case line of
            1: authors := str;
            2: title1 := str;
            3: title2 := str;
            4: placel := str;
            5: place2 := str;
            7: key1 := str;
            8: key2 := str
        end; (* of case *)
    end
end
else
    begin
        writeln('Date ',entry.date :4);
        writeln;
        repeat
            write('REPLACE, NEXT line or SKIP to end .... ');
            InlChar(LineOption)
            until LineOption in ['R','r','N','n','S','s'];
            if LineOption in ['R','r']
            then
                begin
                    writeln('Type replacement date ');
                    write(': ');
                    InlInt(entry.date)
                end
            end;
            until ((line=8) or (LineOption in ['S','s']));
        end;
        writeln;
        writeln('Modified item reads : ');
        writeln;
        OutRecord(entry,m);
        writeln;
    end; (* of change *)

begin (* MAIN PROGRAM *)
    count := HiTag;
    n := 1;
    reset(PendingTray);
    rewrite(TempPendingTray);
    while not eof(PendingTray) do
        begin (* copy down existing contents of file
        'PendingTray' *)
            TempPendingTray^ := PendingTray^;
            put(TempPendingTray);
            get(PendingTray)
        end;
        rewrite(PendingTray);
        reset(TempPendingTray);
        while not eof(TempPendingTray) do
            begin (* copy back 'PendingTray' and count contents *)
                PendingTray^ := TempPendingTray^;
                put(PendingTray);
                get(TempPendingTray);
                n := n+1
            end;
            rewrite(TempPendingTray);
        repeat
            writeln;
            repeat
                write('Do you wish to APPEND, to CHANGE, ');
                writeln('to use the SPECIAL facility, ');
                write('or to FINISH .... ');
                InlChar(MainOption)
                until MainOption in ['A','a','C','c','S','s','F','f'];
            (* MainOption= S is a special facility,
            used for loading from 'scratch' to 'PendingTray' *)

            case MainOption of
                'A','a': (* TO APPEND *)
                    begin
                        writeln;
                        repeat
                            write('Do you need help
                            [YES or NO] .... ');
                            InlChar(HelpOption)
                            until HelpOption in ['Y','y','N','n'];
                            if HelpOption in ['Y','y']
                            then
                                begin
                                    writeln;
                                    writeln('NOTES.');
                                    write('(a) Authors and keywords separated');
                                    writeln(' by a comma ",,");
                                    write('(b) To remove the automatic conversion to ');
                                    writeln('upper case letters');
                                    write(' begin a line of text with');
                                    writeln(' a backslash "\.');

```

