PASCAL USER'S GROUP

USER'S

GROUP

PASCAL NEWSLETTER

NUMBER 5

COMMUNICATIONS ABOUT THE PROGRAMMING LANGUAGE PASCAL BY PASCALERS

SEPTEMBER, 1976

TABLE OF CONTENTS

POLICY -- PASCAL USER'S GROUP AND PASCAL NEWSLETTER

USER'S GROUP POLICIES

Membership - is open to anyone: particularly Pascal users, teachers, maintainers, implementors, distributors, or just plain fans.
Institutional memberships, especially libraries are encouraged.
The cost of membership is $4 per academic year ending June 30. Anyone joining anytime for a particular year will receive all 4 quarterly issues of *Pascal Newsletter* for that year. (In other words back issues are sent automatically). See ALL PURPOSE COUPON on back cover.
First time members receive a receipt for membership; renewers do not to save money for PUG on postage.

Purposes - are to promote the ideas behind Pascal as well as the use of the programming language Pascal. Pascal is a practical language with a a small, systematic, and general purpose structure which is being used for:

* teaching programming concepts
* developing reliable "production" software
* implementing software efficiently on today's machines
* writing portable software

Let's get more users involved - urge your Pascal friends to join PUG whether face to face or maybe through an announcement in your installation's local newsletter.

NEWSLETTER POLICIES

The *Pascal Newsletter* is the official but informal publication of the User's Group. It is produced quarterly (usually September, November, February, and May). A complete membership list is printed in the November issue. Single back issues are available for $1 each. Out of print: #s 1,2,3.

The contribution by PUG members of ideas, queries, articles, letters, and opinions for the *Newsletter* is important. Articles and notices concern: Pascal philosophy, the use of Pascal as a teaching tool, uses of Pascal at different computer installations, portable (applications) program exchange, how to promote Pascal usage at your computer installation, and important events (meetings, publication of new books, etc.).

Implementation information for the programming language Pascal on different computer systems is provided in the *Newsletter* out of the necessity to spread the use of Pascal. This includes contacts for maintainers, documentors, and distributors of a given implementation as well as where to send bug reports. Both qualitative and quantitative descriptions for a given implementation are publicized. Proposed extensions to Standard Pascal for users of a given implementation are aired. Announcements are made of the availability of new program writing tools for a Pascal environment.

Miscellaneous features include bibliographies, questionaires, and membership lists.

ALL WRITTEN INFORMATION FOR THE *Newsletter* IS EASIER TO PRINT IF YOU WILL TYPE ALL MATERIAL 1½ OR DOUBLE SPACED SO THAT IT IS IN "CAMERA-READY", "PHOTO-REDUCIBLE" FORM FOR THE PRINTER. REMEMBER, ALL LETTERS TO ME WILL BE PRINTED IN THE *Newsletter* UNLESS THEY CONTAIN A REQUEST TO THE CONTRARY. AN OVERRIDING GUIDE SEEN IN AN OLD MAD MAGAZINE APPLIES: "*all the news that fits, we print!*" - Andy Mickel, editor, August 5, 1976.
John P. Strait, assoc. editor

UNIVERSITY OF MINNESOTA
TWIN CITIES

University Computer Center
227 Experimental Engineering Building
Minneapolis, Minnesota 55455
(612) 376-7290

PART I - In General

Hi! And here is the first issue of the Pascal User's Group *Pascal Newsletter*. Very important: read POLICY on inside front cover. My editorial comments appear throughout the newsletter as Pascal style comments enclosed in "(*" and "*)". At this writing, PUG has a phenomenal 317 members. We have grown steadily since we began soliciting members in April. Some members have had so much faith in our continued existence that they have signed up for several years!

There were two members who suggested that the word "user's" in PUG's name be changed to "users'" instead (see HERE AND THERE). To me it doesn't really matter. I could argue that PUG belongs to each individual member - it is more like a federation.

For the record here are some of the events which led to PUG. We have much to owe George Richmond, of the University of Colorado (and Lyle B. Smith before him) for tending the fire nearly alone in North America for several years. George began *Pascal Newsletter* with issues in January and May of 1974 and February, 1975. In the third (February) issue he sent a sort of "SOS" to all persons listening: "...there is a need for a strong Pascal Users Group...the present mechanism for distribution and support will become more inadequate." After talking with other Pascalers - namely Alfred Towell at Indiana University, Dave Tarabar at the University of Massachusetts, and George - John Strait and I sent a letter to the editor in July, 1975 stating our desire to participate in a User's Group.

A year ago at ACM '75 in Minneapolis, a Pascal User's Group meeting was held spontaneously on October 22 at the urging of Richard Cichelli of Lehigh University and R. Warren Johnson of St. Cloud State University. Thirty-five persons attended and we decided to work toward a more permanent organization using *Pascal Newsletter* as a communications medium. I organized a mailing list and little happened. When I talked to George in December about the newsletter editorship, he said he wanted to do one more issue himself. So we planned our first issue for April '76.

But as things turned out, George was delayed by terrific work loads at his computer center which incidentally hurt his other Pascal duties. After a few months we decided to organize more thoroughly and push the date for our first newsletter to September. In April and May we sent out 400 general solicitations to join PUG to computer centers and computer science departments at universities in the United States and Canada. We also placed an announcement in SIGPLAN Notices for May. In the mean time we were hoping that George's Newsletter #4 would appear to announce the transition to the persons on his already established mailing list of more than 500.

As it stands, George's last newsletter will be appearing just before this one, and to avoid the complication of excessive requests for back issues and duplicating material, we will purchase copies of his newsletter to send as a free "bonus extra" to persons in the United States and Canada who are not on George's list. Persons on Georges list who are not PUG members should read the transition information in Newsletter #4 and join PUG hopefully. I estimate we will gain at least another hundred members this way.

Those of you who received a receipt for membership may be interested that the pug dog is a sort of joke - I for one would not feel secure having such a weakling for my guardian (a sort of a pig of a pug the way it was drawn). Actually Pascal User's Group is a shoestring operation run right now in the spare time of a couple of systems programmers. The important implication here is that John and I cannot be very responsive to individual requests - all we can promise to do is the newsletter and rest assured we'll print everything that comes to our attention.

With this issue of the newsletter we hope things begin to improve, because I realize all is not good with Pascal right now. I can tell from PUG's mail. Confusion reigns. But I'm an optimist - we'll pull through.

Next issue I will supply an accounting of our costs so far. (In the post-Watergate spirit of full disclosure.)

Now I must give credit where credit is due for PUG and *Pascal Newsletter* #5:

John Eisenberg for his idea of collecting phone numbers,
Richard Cichelli for suggesting guidelines for the cost of membership,
Wilhelm Bürger for suggesting user's group memberships rather than
newsletter subscriptions,
Al Towell for miscellaneous encouragement and suggestions,
SICDOC Systems Documentation Newsletter for the idea for an "ALL PURPOSE
COUPON",
Christi Mickel for doing the mass mailing of 400 and for processing
memberships in PUG,
Computers and People "Computer Directory and Buyer's Guide" for an organizing
and reference tool,
SIGPLAN Notices for publicity in their May issue,
Les Kerr for the suggestion to print a roster of members in an early issue
of the newsletter (see next issue),
John Strait for creating the mailing list data base and for innumerable
suggestions,
Michael Schneider for helping design our cover letter and SIGPLAN announcement,
James Dorr (editor of Indiana University's Random Bits) for producing our
front cover title,
Niklaus Wirth and Urs Ammann for encouragement,

EDITOR'S CONTRIBUTION

George Richmond and Jan Hurst for providing last minute transition suggestions in August,
The computer center newsletters of Lawrence Berkeley Labs (Ed Fourt), the Middle-Illinois Computer Coop (Don Klett), Purdue University, and the University of Minnesota for articles publicizing PUG,
Twin Cities ACM's Bits and Bytes (Judith Kruntorad) for announcing PUG,
and finally the University Computer Center, University of Minnesota for providing a warm home for PUG.

PART II - Pascal at the University of Minnesota

Our computer installation consists of a CDC Cyber 74 running large scale batch and 30 interactive terminals, and a CDC 6400 running 150-200 interactive terminals only. Pascal has been available here since summer of 1972 when a colleague of mine (and now PUG member) Steve Legenhausen suggested we obtain the Pascal compiler. Usage here has been boosted mainly by applying William Waite's principles for giving a language processor (organism) support at a computer installation (ecosystem). Now after the Computer Science Department's very successful initial year of replacing FORTRAN with Pascal in its curriculum, usage is respectable. In the last fiscal year (July-June) at Minnesota the major FORTRAN compiler (MNF) was run 810,000 times on both machines; BASIC 477,000; Pascal (#3!) 103,000; FTN (CDC FORTRAN) 69,000; COBOL 49,000; Assembler 44,000, SNOBOL 40,000; and etc.

John Strait, Lawrence Liddiard and I have cooperated with Urs Ammann of ETH Zürich over the past year in producing the second release of Pascal 6000-3.4. We helped mainly in the effort to reduce core requirements for the compiler. Our group continues to maintain the compiler for the KRONOS/NOS operating system for CDC 6000/Cyber 70,170 series machines and in fact several other sites run our version. The main changes have been for interactive access because 85% of Pascal's use here is interactive. We are now cooperating with George Richmond to make our mods for KRONOS/NOS available with the distributed version.

PART III - My Concerns

As I mentioned earlier, all is not well with Pascal. Mainly, considering the design goals of the language (reiterated in the POLICY section inside the front cover) we are suffering.

First, people continue to ignore the combination of these design goals when making suggested "improvements" to the language. These are the subject of several letters and comments which appear in this issue of the newsletter.

Secondly, several bad implementations are being circulated and in turn are giving Pascal an unnecessary, bad reputation as a language. See IMPLEMENTATION NOTES. Also many implementors have taken the liberty to implement something significantly less than Standard Pascal and call it Pascal. What about portable software then?

Finally, confusion proliferates at to what the next developments will be and where implementations will come from. This situation should improve with the regular appearance of this newsletter.

We all need to pull together to help remove a major obstacle to our being able to respectably use Pascal: its low percentage of usage in the world's computing. We can do it; it will happen. A major indicator is the "Pascal explosion" in the current computer science literature.

I consider the IMPLEMENTATION NOTES section to be very important. Here we will hope to find increasingly complete and usable information for spreading the "virus" of Standard Pascal.

- Andy

August 10, 1976

HERE AND THERE WITH PASCAL (NEWS FROM MEMBERS, CONFERENCES, NEW BOOKS, APPLICATIONS PROGRAMS, ETC.)

CONFERENCES

Pascal User's Group session at ACM '76....Wally Wedel, PUG member from the University of Texas at Austin will chair a PUG meeting at this year's ACM conference in Houston, Texas. The conference extends from Wednesday, October 20 to Friday, October 22 in the Hyatt Regency Hotel. Wally has made arrangements with the conference organizers and SIGPLAN; the exact time of the meeting will be printed in the schedule handed out at the conference on Wednesday. Proposed topics of discussion are: Interactive I/O conventions/Extended character set treatment/ Implementations and user experience with implementations/Documentation standards for variations.

Pascal: Implementation and Application....D. W. Barron has announced a two day Symposium organised by the Computer Studies Group, University of Southampton, United Kingdom during 24-25 March 1977. Sessions include: The language and its implementation/Pascal in systems programming/Pascal in research and education/ Pascal, the future. Well known authorities have been invited to speak. To receive further details when available - write to: Conference Secretary, Department of Mathematics, The University, Southampton, SO9 5NH, United Kingdom.

NEW BOOKS

Algorithms + Data Structures = Programs by Niklaus Wirth, Prentice Hall, 1976, 366 pages, hardcover, $15.

A Primer on Structured Programming Using PASCAL by Richard Conway, David Gries, and E. C. Zimmerman, Winthrop Pub., 1976, 420 pages, paperbound. (*Note: for price write to PUG member Michael Meehan, Winthrop Pub., 17 Dunster St., Cambridge, MA 02138.*)

Introduction to Problem Solving and Programming with Pascal by G. Michael Schneider, David Perlman, and Steven W. Weingart, Wiley, to be published in 1977. (*Note: for more info write to PUG member Michael Schneider, C. Sci. Dept., 114 Lind Hall, Univ. of Minnesota, Minneapolis, MN 55455.*)

Study Guide, Introduction to Computer Science by Kenneth L. Bowles, to be published. (*Note: for more info write to K. Bowles, Univ. of California, San Diego, La Jolla, CA 92093.*)

Standard Pascal by J. W. Atwood, to be published. (*Note: for more info write to J. W. Atwood, Dept. of Comp. Sci., Sir George Williams Campus, Concordia Univ., Montreal, Quebec, Canada H3G 1M8.*)

NEWS (alphabetical by last name)

O. Beaufays, Mathematiques Appliques, Universite Libre de Bruxelles, Bruxelles 1050 Belgium (PUG member): "...we are using this language for teaching..."

Scott Bertilson, RR 2, Spicer, MN 56288 (PUG member): "James Martinson and I are interested in microcomputer versions of Pascal, Pascal-S, or Concurrent Pascal"

Albrecht Biedl, Institut fuer Softwaretechnik, Technische Universitat Berlin, 1000 Berlin 10, Germany VSH 419 (PUG member): "I enclose copies of the PASCAL Info we have published in 76 for a growing Pascal community at the Technical University of Berlin" (Numbers 0 (1976-01-05), 1 (1976-02-26), 2 (1976-03-03), and 3 (1976-06-15))

Richard J. Cichelli, 901 Whittier Drive, Allentown, PA 18103 (PUG member):
"What we need next is a program which reads up Pascal relocatible binaries and proposes overlay structures. It should report field lengths vs. various overlay alternatives. It really is about time that many of the clerical and record keeping tasks of programming be automated. Pascal users should have the best tools for program development. Do you have any suggestions in this area? (source and object file maintenance systems?)
"Pascal users accumulate 25% of all charges on Lehigh's system.
"The last correspondence of the Zurich-Minnesota letters that I have was dated November 25. Incidentally I think your letters should be more supportive to them. I feel that reasoned discussion with the community at large is the way to resolve some of these technical issues. Let Wirth and Hoare set the principles and ideals. Help Urs with his implementations and help create a forum for information interchange. We need to promote growth and change in an environment of mutual cooperation.
"I believe any changes in implementation should be discussed with the users. Only organizers like you can facilitate the necessary communication. Note, I am not opposed to changes per se. If Pascal 6000 is to grow it must be a living, changing language. Rational change is possible only with the cooperation of the user community.
"I like the name PUG. See how Wirth likes it.
"I would like to see a user profile on PUG members: usage statistics, application environments, etc."

Kurt Cockrum, 3398 Utah, Riverside, CA 92507 (PUG member): "I am particularly interested in microprocessor (8080) implementations of Pascal."

R. G. Dickerson, School of Information Sciences, The Hatfield Polytechnic, Hatfield AL10 9AB, United Kingdom (PUG member): "...we have a DEC 10 and use Nagel's (Hamburg) Pascal compilers. We are going to use Pascal as a first language for our B.Sc. in computer science (we have about 200 undergraduates on the degree)."

Doug Dyment, 6442 Imperial Ave. W. Vancouver, B.C. V7W 2J6 Canada (PUG member): "My current interest in Pascal is an evaluation of its use as a system programming language. Good luck with the new group."

Gerhard Friesland, Institut Fuer Informatik, Universitat Hamburg, 2 Hamburg 13, Germany (PUG member): "My interest is based on participation in the transport of a compiler onto the PDP-10 and current work on an interactive programming system, implemented via a compiler-compiler in Pascal."

Dale Grit, Dept of Computer Science, Colorado State University, Fort Collins, CO 80523 (PUG member): "We're using Pascal to a limited extent (e.g. the compiler course). We are still stuck with teaching FORTRAN in our intro course, but our approach to FORTRAN is to teach them to develop problem solutions in a "thinking" language (which just happens to have Pascal control constructs) and them how to mechanically go from there to FORTRAN.
"Another approach we hope to try is to teach Pascal for 8-9 weeks of the semester, then teach FORTRAN as a restrictive subset.
"We have a student doing a summer project to put up Hansen's sequestial Pascal and his concurrent Pascal. We are getting a Cyber 18 this fall and plan to make it a Pascal machine."

Sam Gulden, Dept of Mathematics, Lehigh University, Bethlehem, PA 18015 (PUG member): "Enclosed you will find applications from some of the members of our local Pascal Users Group. I am looking forward to the newsletter having been an enthusiastic Pascal user for about two years. We have used Pascal here to tackle some interesting mathematical problems. Perhaps we will report on them in the future."

Michael Hagerty, 18 Hamilton Road, Arlington MA 02174 (PUG member): "Our work is with large data bases (200 cards for each of 300000 people). Processing is constrained by I/O and takes 20000 seconds on a CDC6400. We are therefore implementing GETBUF and PUTBUF routines to read more data at a time from tapes.
"I am working on a paper which will define an additional structure for Pascal, the environment. This concept will allow the easy integration of a form of overlays, a more dynamic (moving FL) system, as well as the inclusion of a "systems text" for compilation. Once completed, I will send you a copy. The implementation will have to wait until I can find time to sketch out the loader needed to handle multiple environments.
"Included in our implementation of Pascal is a copy of Michael Condict's reformatter....I feel that one function of PUG would be to see that software written in Pascal is made available to the larger community..."

Charles Hedrick, 183 Commerce West, University of Illinois, Urbana, IL 61801 (PUG member): "If we are going to have a language which is implemented and maintained entirely by users, as seems likely (no computer manufacturers have made offers to do it), it is clear that there should be at least a lot of communication between people doing work on the same machine. Preferable would be to have one person for each machine maintain a common version which gets everybody's bug fixes. Notice I say bug fixes and not enhancements or extensions. To keep track of all of these would be more work, and would probably detract from the stability of the system. Alas, I have no candidate to propose at the U. of I. for this. I am not a full-time programmer (I teach and do research as an asst. prof. and don't have time for such things). The person who works on our local version has been unable to get funding for Pascal work, and may well be switched to a system other than the DEC 10 anyway. (We are in the process of getting a new computer system.)"

William C. Hopkins, 207 Ridgewood Drive, Amherst NY 14226 (PUG member): "Note: as there are several users, shouldn't the name be "Pascal Users' Group" ?"

Ed Katz, Computer Science Dept., Box 4-4330, USL Station, University of SW Louisiana, Lafayette, LA 70504 (PUG member): "We had such success teaching Pascal-S last year on our Honeywell Multics system, that we plan to use a full implementation this year."

Thomas A. Keenan, Software Systems Science, Division of Mathematical and Computer Sciences, National Science Foundation, Washington, DC 20550 (PUG member): "Good luck with your venture."

Leslie R. Kerr, David L. Johnson and Associates, 10545 Woodhaven Lane, Bellvue, WA 98004 (PUG member): "I would like to thank you for taking the initiative in founding a Pascal user's group, which I feel is long overdue. I hope I will be able to contribute in some way to its success.
"I would like to see the roster of PUG members published in an early issue of the Newsletter."

Jan Kok, Mathematisch Centrum, Tweede Boerhaavestraat 49, Amsterdam, The Netherlands "The Mathematical Centre Amsterdam (Mathematisch Centrum) is engaged in constructing a numerical mathematics procedure library in Pascal, to be available on the CDC Cyber 73-28 computer of the Academic Computer Centre at Amsterdam."

O. Lecarme, I.M.A.N., Universite de Nice, Parc Valrose, 06034 Nice Cedex, France (PUG member): "I am planning to create a smaller but similar group for the French speaking community, and I will be very happy to maintain good communication with you."

Chris Martin, Computing Services, The Hicks Building, University of Sheffield, Sheffield S10 2TN United Kingdom (PUG member): "We have the Belfast compiler on an ICL 1900 and though at the moment it isn't very widely used, I expect the rush will start when I get the Montreal Compiler Writing System installed."

Joseph Mezzaroba, Dept of Mathematics, Lehigh University, Bethlehem, PA 18015 (PUG member): "I have been using Pascal (as implemented for the CDC-6400), here at Lehigh University for the past two years. I will be teaching Computer Science at Villanova University starting in September and I would like to get either a Pascal or ALGOL-W Compiler on Villanova's IBM 370."

Carlton Mills, Mills International, 203 North Gregory, Urbana, IL 61801 (PUG member): "Has anybody defined any structured escape language constructs? Has anybody defined any macro facilities? We are about to."

Judy Mullins, Department of Mathematics, The University, Southampton United Kingdom S09 5NH (PUG member): "I enclose ... $8 ... for two year's subscription to the Pascal Users' Group and Newsletter. This advance payment was prompted by the large banker's commission on small sums such as $4....
"Arising from this, I was wondering whether it would be useful or possible to a arrange some kind of branch of P.U.G. in the U.K. for collecting subscriptions. ... a convenient and cheaper form of membership may encourage more members in the U.K. There are certainly many institutions using Pascal now, and interest is spreading....
"As co-organizer with Prof. Barron of the Pascal Symposium for next March, I could get the thing started. Others in our group are writing a Pascal compiler for ICL's new 2970 computer, so we shall always have a vested interest in Pascal."

Maurice O'Flaherty, 444 Merville Garden Village, Newtown Abbey, Co. Antrim, N. Ireland (PUG member): "I am at present finishing my thesis for an M. Sc. in Computer Science and Applications, and having used Pascal I would like to continue my interest in it."

George Richmond, Computing Center, 3645 Marine St. University of Colorado, Boulder, CO 80309 (PUG member): "The Computer Science Dept. here will be converting to Pascal this fall, starting in the introductory courses."

Steve Reisman, Clinical Systems Division, School of Denistry, University of Minnesota, Minneapolis, MN 55455 (PUG member): "We are using Pascal for scheduling and grading for the School of Denistry."

Staffan Romberger, Scoputer Science, Royal Institute of Technology, S-10044 Stockholm, Sweden (PUG member): "Here at the Computer Science Department of Royal Institute of Technology there is a growing interest in Pascal. We have access to the Pascal and Pasrel compilers for DEC-10 from Hamburg and there are also compilers for PDP-11 and movements towards writing compilers for other computers."

David Slocombe, The Globe and Mail, 444 Front St. West, Toronto, Ontario M5V 2S9 Canada: "Although we don't now have a Pascal compiler (we intend to check out the Stony Brook implementation as soon as we have time), we have followed the development of the language almost from the beginning and two of us here have used Pascal as a design language for some years. It sure would be nice not to have to hand-compile!"

N. Solntseff, Dept. of Applied Mathematics, McMaster University, Hamilton, Ontario Canada L8S 4K1 (PUG member): "I am interested in participating in the users' group and am willing to contribute my time in any capacity.
"Incidentally, I would be happier if the title of the group were the more grammatical "Pascal Users' Group". "

W. Richard Stevens, Kitt Peak National Observatory, P.O. Box 26732, Tucson, AZ 85726: "Here at Kitt Peak I have just installed the 6000-3.4 compiler on our CDC 6400 and am currently trying to generate interest in the language. In addition, I would like to volunteer any services of myself to help the User's Group.
"Will the User's Group have any affiliation with the current distribution center at the University of Colorado?"

William Waite, Software Engineering Group, Dept. of Electrical Engineering, University of Colorado, Boulder, CO 80302: "Thank you for your invitation to join the Pascal User's Group. Lack of funds makes it impossible for me to accept personally;...You ask why PLAP was not written in Pascal. The answer is obvious - lack of portability. We have been attempting to cure this problem, as well as doing some research in intermediate language design. Unfortunately we have succeeded in the latter while the former still eludes us. The whole story is a sad one, resting upon the inadequacy of our tools. I believe that we will lick the problem eventually, but until I see the evidence I shall write portable programs in another language."

"I was interested that the Pascal usage at Minnesota has exceeded that of RUN and FTN. Have you been using it in an introductory course? We plan to do so next fall, and I would appreciate any comments you have.

"It seems to me that one of the most difficult problems faced by Pascal is one that could be considered irrelevant: the ordering of multidimensional arrays. Since the language definition <u>implies</u> row-major order, it seems that Pascal-FORTRAN communication might be very difficult. How have you handled the problem when perverting FORTRAN library routines?"

Wally <u>Wedel</u>, Computation Center, University of Texas at Austin, Austin, TX 78712 (PUG member): "We are running Nagel's DEC-10 Pascal, Brinch Hansen PDP-11 Pascal, Wirth's CDC6000 Pascal. Wilhelm Burger has done extensive work on both the DEC-10 and CDC6000 implementations."

Lecture Notes in Computer Science, No. 18

(*When purchasing, specify the "Springer Study Edition" - it's 35% cheaper!*)

<u>PASCAL - USER MANUAL AND REPORT</u> by K. Jensen and N. Wirth, Springer-Verlag, 1974, 1975, 167 pages, paperbound.

<u>Corrections to 2nd Edition</u>

p.51, 1.16:	"setop(output)" → "setop(output);"
p.56, 1.-6:	"fi" → "f(i)" "g(i+1)" → "g(j+1)" "gi" → g(j)
p.63, Fig.10a:	Number sequence should be reversed.
p.69, 1.23:	"stricly" → "strictly"
p.77, 1.18:	move line 3 places to the left
p.98, 1.10:	append " ';"
p.102, 1.7:	last word should be "or"
1.20:	"bufffer" → "buffer"
p.103, 1.-6:	"scaler" → "scalar"
1.-7:	"char, and alfa" → "and char are listed"
p.124, 1.-14 and 1.-15:	"14" → "15"
p.127, 1.27:	"18.A" → "4.A"
p.133, 1.3:	"two" → "to"
p.135, 1.5:	"althought" → "although"
1.30:	"subtstitute" → "substitute"
p.140, 1.11:	"structure type" → "structured type"
p.158,	delete lines -12... -8.

N. Wirth
10.May 1975

<u>Additional Suggested Corrections</u>

September 26, 1975 University Computer Center
University of Minnesota

<u>Page/Line</u>	
13/-3	"if" → "If"
69/-8	"the readability" → "readability"
72/-3	"element in the array" → "component in the structure"
81/2	"extent" → "extend"
117/	in the syntax chart for expression, change: ≠ ≤ ≥ to <> <= >=
119/13	move the message right by 1 position
126/-19 to -14	move the "i" index entries to the next page
153/16	"(and at least once)" → "(at least once)"
162/3	"<u>end of line</u>" → "<u>end</u> <u>of</u> <u>line</u>"

Designing Data Structures by Step-wise Refinement

A Tutorial (*Note: by Richard J. Cichelli*)

Keywords: Data structures, step-wise refinement, top-down design, systematic programming, PASCAL.

Abstract

Dijkstra [1] and Wirth [2] have defined the principles of systematic programming. They illustrated these principles by designing programs whose control structures reflected hierarchical abstractions of their logic flow. In this paper, systematic programming principles are applied to the design of a program's data structures.

Overview

This paper begins with a reexamination of the Queens problem: a traditional program design problem. An alternate data structure is devised for the program and relevant design issues and terminology are discussed.

A step-wise, top-down design of the data structures for a Soma cube [3] solver is then presented. The data definitional capabilities of PASCAL [4] aid in the design process.

The Queens Problem Revisited

Both Dijkstra and Wirth use a traditional backtracking problem to illustrate systematic programming. The problem is to write a program which places eight hostile Queens on a chess board such that no Queen threatens another. The programs are extended to find all 92 solutions.

Following Dijkstra's program, the main routine might be:

```
begin initialize; generate end.
```

Initialize clears the board and generate recursively calls itself to place a Queen on each column.

```
procedure generate;
   var h: 0..7;
begin
   for each_row h do
   begin
      if square_is_free then
      begin place_queen_on_square;
         if board_full then print_solution else generate;
         remove_queen_from_square (* backtrack *)
      end
   end
end;
```

Dijkstra tests whether a square is free by noting:

1) for each column N ($0 \leqslant N \leqslant 7$), generate only places one Queen,
2) for each row H, the boolean array column[H] is used to mark a row as taken, and
3) for the 30 diagonals, the boolean arrays up[N-H] and down[N+H] complete the masking.

These three boolean arrays are involved in testing if a square is free, placing a Queen, and removing a Queen.

To improve the speed of the program we can augment Dijkstra's data structures and, by having the program know more, trade space for time. Observe that the column, up, and down arrays are simply marking instances of the same type of thing. Each masks off a direction on the board. In total there are 46 such directions on the chess board - 16 for the rows and columns and 30 for the up and down diagonals.

Since the mask for each square [N,H] is a unique set, this loop invariant calculation can be done once in the initialization code. (See revised program in Figure 1.)

The Soma Cube

The Soma puzzle consists of seven pieces; six of the pieces are made by joining four cubes together, and the remaining piece is made up of only three cubes. The problem is to fit the pieces together to form a 3x3x3 solution cube. There are 240 unique solutions to the puzzle. (The seven pieces are shown in Figure 2.)

It is evident that the simple backtracking algorithm which worked for the Queens problem will also work for the Soma cube. Pieces in the Soma cube are placed like Queens on the chess board. A piece is included in the solution cube only if it "fits".

To find a solution, we start with an empty solution cube and place pieces one by one until all pieces are placed or the current piece does not fit. If the piece does not fit, the previously placed piece is removed and replaced elsewhere, and the search continues until a solution is found or the piece locations are exhausted.

The difficulty here is that the search space is so large that efficient testing of whether a piece fits is essential for an effective program. The possible solution locations of any piece need only be calculated once if we have a data structure like the boardmask data structure in the Queens program. We can create such a data structure by top-down design methods.

The Soma Cube Data Structure

The Soma data structure is operated upon by two routines; the first initializes it, and the second generates solutions with it. Although it is composed of many parts, this data structure is properly viewed as a single named entity. It holds all the data relevant to the seven Soma pieces. In PASCAL we declare

```
        var pieces: array [piece] of piecedescription;
```

This declares pieces to hold the same type of information for each piece. Since we wish to treat each piece in the same way, this is the appropriate overall structure.

Next we need to declare the types piece and piecedescription. There are seven pieces and so the declaration

```
        piece = 1..7;
```

is appropriate.

For each piece the piecedescription must completely describe the information local to a piece. It will basically consist of a list of locations. The length of this list varies from piece to piece, and in addition, during backtracking we will need to know where we are in the list. Since these items are not of the same type, the record structure is needed:

```
        piecedescription =
           record
              listlength, whereat: listsize;
              positionlist: array [listsize] of positions
           end;
```

We can postpone the calculation of the maximum listsize by declaring the type

```
        listsize = 0..maxlist;
```

Maxlist will be a declared constant.

Positionlist is declared an array of positions because we expect each possible description of the location of a piece to be of the same type. Since each piece will fill a set of locations in the solution cube, the declaration

    positions = set of locations;

seems natural. There are 27 locations in the 3x3x3 solution cube. We thus declare

    locations = 1..27;

To calculate maxlist we observe (usually with a little difficulty) that piece 1 can be placed in 144 unique orientations within the solution cube. It obviously can occupy more places than any other piece.

The final list of declarations that we have built up appears below:

```
const
   maxlist = 144;
type
   piece = 1..7;
   locations = 1..27;
   positions = set of locations;
   listsize = 0..maxlist;  (* one to spare *)
   piecedescription =
      record
         listlength, whereat: listsize;
         positionlist: array [listsize]  of positions
      end;
var
   pieces: array [piece]  of piecedescription;
   somacube: positions;
```

How Will the Soma Program Work?

The first phase of the program will generate the piece descriptions. It will have to take each piece and rotate and translate it through all 27 locations. Duplicates should not

be entered into the positionlist. As the positions are added to the positionlist, listlength is incremented.

The backtracking phase begins after all positionlists are complete. For each piece, whereat (which is initialized to zero) is incremented from zero to listsize. It is the index to the positionlist of the position under examination. Pieces which fit are joined into the somacube solution. The "fit" test is simply the set operation

    ((positionlist [whereat] meet somacube) eq [ ])

where [ ] is the empty set. To add in a piece which fits we write

    somacube := positionlist [whereat] join somacube;

Piece removal during backtracking only requires set differences. (These operations are very fast in most PASCAL implementations because hardware boolean logic is used for set operations.)

A complete version of a PASCAL Soma cube solver can be found in [5] .

Summary

We have shown that top-down design can be applied to complex problems in data structure design. It is hoped that the examples chosen illustrate the desirability of PASCAL-like type definitional capabilities for the top-down design of data structures. Languages without such type definitional capabilities are as deficient for data structure design as those which lack block control structures are for structured programming.

(*Received 2/21/76*)

References

1. Dahl, O.J., Dijkstra, E.W. & Hoare, C.A.R. Structured Programming, Academic Press, London, 1972.

2. Wirth, Niklaus. Systematic Programming: An Introduction. Prentice-Hall, Inc., Englewood Cliffs, N.J., 1973.
   Wirth, Niklaus. "Program development by stepwise refinement", Communications of the ACM 14, 4, 1971.

3. Introducing Soma, Parker Brothers, 1969.
   Scientific American, October, 1958.

4. Jensen, K. & Wirth, N. PASCAL User Manual and Report. Springer-Verlag, New York, 1974.

5. Cichelli, R.J. & DeLong, R. "Solutions to the Soma Cube Problem", SIGPLAN Notices, October, 1974.

6. Cichelli, R.J., Gulden, S.L., & Condict, M.N. "Another Solution to the Soma Cube Puzzle", SIGPLAN Notices, December, 1975.

```
type
   zeroto7 = 0 .. 7;
   directions = 0 .. 45;
   directionmask = set of directions;
   rowmask = array [zeroto7] of directionmask;
   boardmask = array [zeroto7] of rowmask;

var
   j, k : integer;
   queenindx : integer;
   queens : array [zeroto7] of zeroto7;
   mask : boardmask;
   board : directionmask;

procedure generate;
var
   colhgt : zeroto7;
begin
    for colhgt := 0 to 7 do
    begin {  test square colhgt free  }
        if ((board meet mask[colhgt][queenindx]) is []) then
        begin  {   set queen on square   }
            queens[queenindx] := colhgt;
            board := board join mask[colhgt][queenindx];
            queenindx := queenindx + 1;
            {   test if board is full   }
            if (queenindx = 8) then
            begin  {   print board   }
                for k := 0 to 7 do write(' ',queens[k] : 2);
                write(eol)
            end else generate;
            {   remove queen from board   }
            queenindx := queenindx - 1;
            board := board - mask[colhgt][queenindx]
        end
    end
end;

begin  {   initialize the empty board   }
   queenindx := 0;
   board := [];
   for j := 0 to 7 do
   for k := 0 to 7 do mask[j][k] := [j,(15+(k-j)),(23+(k+j))];

   generate
end.
```

Figure 1

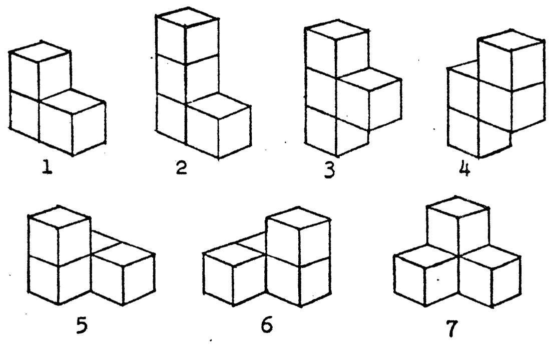


Figure 2

To the Editor:

If the "science" of Computer Science is the experimental investigation of algorithms, then effective programming is essential for the computer scientist's computing "laboratory" testing. Good programming is an art, an engineering design art. Without it, the computer scientist's investigative procedures are suspect: with it, good design helps clarify previously obscure algorithms.

In our software engineering course at Lehigh University we have emphasized that the principles of top-down design [1] and structured programming [2] apply not only to a program's function code but also to its data structures. The data definitional capabilities of PASCAL [3] greatly facilitate the top-down formulation of data structure hierarchies. Good programs establish corresponding levels of abstraction between their control structures and their data structures.

The following student Soma Cube program effectively uses data and control hierarchies. In his enthusiasm for this software engineering problem, Michael Condict made the transition from a design project to the experimental and scientific investigation of space filling algorithms. It is, to the best of our knowledge, the ultimate Soma Cube program (at least until next semester).

Richard J. Cichelli
Samuel L. Gulden
Mathematics Department
Lehigh University

Another Solution to the Soma Cube Puzzle

Michael N. Condict
Lehigh University

This Soma Cube [4] solution generator evolved through a series of refinements from a program similar to that of DeLong's [5]. The run time of the first version was approximately 90 seconds on Lehigh University's CDC 6400. By examining the bound unfilled spaces (connected holes in the assembling cube) and pruning when no piece would fit, the run time was reduced to about 30 seconds.

The experiment was extended to include McKeeman's [6] revised Soma algorithm which is based on Combinatory Theory considerations. Performance of the McKeeman algorithm is dependent on the order in which pieces are selected for insertion. The rough execution times for the various algorithms are given below:

Brute force	90 sec.
Hole analysis pruning	30 sec.
McKeeman order (V,L,T,Z,S,R,Y)	60 sec.
McKeeman order (L.Y,S,R,T,Z,V)	15 sec.
Combined McKeeman & hole analysis	13 sec.

References
1. Wirth, N. "Program development by stepwise refinement". Communications of the ACM 14, 4, 1971.
2. Dahl, O.J., Dijkstra, E. W. and Hoare, C.A.R. Structured Programming. Academic Press, London, 1972.
3. Wirth, N. and K. Jensen. PASCAL User Manual and Report. Springer-Verlag, New York, 1974.
4. Introducing Soma. Parker Brothers, 1969.
5. Delong, R. "Solutions to the Soma Cube Problem". SIGPLAN Notices, Oct.,1974.
6. McKeeman, W.M. "Solving Space-Filling puzzles". CEP REPORT. Volume 6, No. 1, May, 1974.

(*Received 3/11/76*)

(*Note: This contribution was printed incompletely in SIGPLAN Notices Oct., 1975 and incorrectly in SIGPLAN Notices Dec., 1975.*)

```
CONST
   LASTCUBE= 26;
   K3321=249;

TYPE
    CONFIGSETNOS= 0..28;
       CUBETYPES= (EDGE, CORNER, FACE, CENTER);
    COMBINATIONS= 0..11;
 PIECECONFIG..AT= 0000..K3321;
   AMTSFORCONFIG= 0..50;
           HOLES= 0..24;
           SIDES= 1..6;
            AXES= (Z,Y,X);
           CUBES= 0..LASTCUBE;
         CUBESP1= -1..27;
          PIECES= 1..7;
        PIECESP1= 1..8;
       POSITIONS= 0..145;
      PIECECUBES= 1..4;
  PIECEPOSITIONS= SET OF 0..58;
          SHIFTS= -26..LASTCUBE;
       THEPIECES= ARRAY [PIECES] OF
                  RECORD
                     PACKD: ARRAY [POSITIONS] OF PIECEPOSITIONS;
                     UNPACKED: ARRAY [PIECECUBES] OF CUBES;
                     PRINTLIST: PIECEPOSITIONS;
                     CONFIGCOMBINATION:
                        ARRAY [COMBINATIONS] OF CONFIGSETNOS;
                     NAME: CHAR;
                     FIRSTCONFIG,
                     CURRENTCONFIG,
                     LASTCONFIG: CONFIGSETNOS;
                  END /*RECORD*/;
        THECUBES= ARRAY [CUBESP1] OF
                  RECORD
                     ALLOWEDSHIFT: SET OF 0..15;
                     ROTATEABOUT: ARRAY [AXES] OF CUBES;
                  END /*RECORD*/;
VAR
   FIRSTPOSITION,
    LASTPOSITION: ARRAY [CONFIGSETNOS] OF POSITIONS;
        NUMBEROF: ARRAY [CUBETYPES,CONFIGSETNOS] OF 0..4;
      COMBNUMBER: COMBINATIONS;
         COUNTOF: ARRAY [CUBETYPES] OF 0..14;
        CUBETYPE: CUBETYPES;
      MAXALLOWED: ARRAY [CUBETYPES] OF 0..12;
  SOLUTIONNUMBER: INTEGER;
       TESTSHAPE,
   SOLUTIONSHAPE: PIECEPOSITIONS;
            CUBE: CUBESP1;
           PIECE: PIECESP1;
            SOMA: THEPIECES;
             ONE: THECUBES;
        HOLESIZE: HOLES;
        HOLECUBE: ARRAY[HOLES] OF CUBESP1;
          TYPEOF: ARRAY [CUBES] OF CUBETYPES;
    ROTATEVALUES: ARRAY [AXES,CUBES] OF CUBES;
     PIECEVALUES: ARRAY [PIECES,PIECECUBES] OF CUBES;
      NAMEVALUES: ARRAY [PIECES] OF CHAR;
      ADJACENTTO: ARRAY [CUBESP1] OF
                  RECORD
                     NUMBERADJSIDES: SIDES;
                     NEIGHBORING: ARRAY [SIDES] OF CUBESP1;
                     CUBESURROUNDED: PIECEPOSITIONS;
                  END;
VALUE
   TYPEOF=
      (CORNER,EDGE,CORNER, EDGE,FACE,EDGE, CORNER,EDGE,CORNER,
       EDGE,FACE,EDGE, FACE,CENTER,FACE, EDGE,FACE,EDGE,
       CORNER,EDGE,CORNER, EDGE,FACE,EDGE, CORNER,EDGE,CORNER);
   ROTATEVALUES=
    ( 6, 3, 0, 7, 4, 1, 8, 5, 2,
     15,12, 9,16,13,10,17,14,11,
     24,21,18,25,22,19,26,23,20,
      2,11,20, 5,14,23, 8,17,26,
      1,10,19, 4,13,22, 7,16,25,
      0, 9,18, 3,12,21, 6,15,24,
     18,19,20, 9,10,11, 0, 1, 2,
     21,22,23,12,13,14, 3, 4, 5,
     24,25,26,15,16,17, 6, 7, 8);
   PIECEVALUES=
    ( 0, 1, 4, 7,
      9,18,19,21,
     10,11,20,23,
     10,11,13,20,
      1, 3, 4, 7,
      0, 1, 4, 5,
      1, 3, 4, 4);
   NAMEVALUES= (≡L≡, ≡Y≡, ≡S≡, ≡R≡, ≡T≡, ≡Z≡, ≡V≡);
   ADJACENTTO=
      /*CUBE  ≡ ADJ        NEIGHBORS          FILLER       */
      /*----  -----  ----------------   --------------*/
      • -1 ♦   (0,                          0,0,0,0,0,0,0,
      •  0 ♦    3,              1, 3, 9,   0,0,0,0,
      •  1 ♦    4,           0, 2, 4,10,   0,0,0,
      •  2 ♦    3,           1,     5,11,  0,0,0,0,
      •  3 ♦    4,        0,     4, 6,12,  0,0,0,
      •  4 ♦    5,        1, 3, 5, 7,13,   0,0,
      •  5 ♦    4,        2, 4,     8,14,  0,0,0,
      •  6 ♦    3,        3,     7,    15, 0,0,0,0,
      •  7 ♦    4,        4, 6, 8,     16, 0,0,0,
      •  8 ♦    3,        5, 7,        17, 0,0,0,0,
      •  9 ♦    4,     0,      10,12,18,   0,0,0,
      • 10 ♦    5,     1,     9,11,13,19,  0,0,
      • 11 ♦    4,     2,   10,    14,20,  0,0,0,
      • 12 ♦    5,     3, 9,    13,15,21,  0,0,
      • 13 ♦    6,     4,10,12,14,16,22,   0,
      • 14 ♦    5,     5,11,13,    17,23,  0,0,
      • 15 ♦    4,     6,12,    16,    24, 0,0,0,
      • 16 ♦    5,     7,13,15,17,    25,  0,0,
      • 17 ♦    4,     8,14,16,       26,  0,0,0,
      • 18 ♦    3,     9,       19,21,     0,0,0,0,
      • 19 ♦    4,    10,    18,20,22,     0,0,0,
      • 20 ♦    3,    11,    19,    23,     0,0,0,0,
      • 21 ♦    4,    12,18,    22,24,      0,0,0,
      • 22 ♦    5,    13,19,21,23,25,       0,0,
      • 23 ♦    4,    14,20,22,    26,      0,0,0,
      • 24 ♦    3,    15,21,    25,         0,0,0,0,
      • 25 ♦    4,    16,22,24,26,          0,0,0,
      • 26 ♦    3,    17,23,25,             0,0,0,0,
      • 27 ♦    1,    27,                   0,0,0,0,0,0);
```

```
PROCEDURE PRINTSET(WORD: PIECEPOSITIONS);
   VAR I: 0..58;
   BEGIN
      WRITE(≡[≡);
      IF WORD ISNT [] THEN
      BEGIN
         I:=0;   WHILE NOT (I IN WORD) DO I:=I+1;
         WRITE(I:2);
         FOR I:=I+1 TO 58 DO IF I IN WORD THEN WRITE(≡,≡,I:2);
      END /*IF[]*/;
      WRITE(≡]≡);
   END /*PRINTSET*/;


PROCEDURE INITIALIZEVARIABLES;
   VAR AXIS: AXES;
   BEGIN
      FOR PIECE:=1 TO 7 DO WITH SOMA[PIECE] DO
      BEGIN
         NAME:=NAMEVALUES[PIECE];
         FOR CUBE:=1 TO 4 DO
            UNPACKED[CUBE]:=PIECEVALUES[PIECE,CUBE];
      END /*FOR PIECE*/;
      FOR AXIS:=Z TO X DO
         FOR CUBE:=0 TO LASTCUBE DO WITH ONE[CUBE] DO
            ROTATEABOUT[AXIS]:=ROTATEVALUES[AXIS,CUBE];
      PIECE:=1;  SOLUTIONSHAPE:=[]; SOLUTIONNUMBER:=0;
      MAXALLOWED[EDGE]:=12;  MAXALLOWED[FACE]:=6;
      MAXALLOWED[CORNER]:=8; MAXALLOWED[CENTER]:=1;
   END /*INITIALIZEVARIABLES*/;


PROCEDURE FINDALLPIECEPOSITIONS;
   VAR              PIECE: PIECES;
      TESTPOS,POSITION: POSITIONS;
           TRANSLATION: CUBESP1;
              ROTATION: 0..24;
         TEMPORARYPIECE: PIECEPOSITIONS;
    CONFIGURATIONSETNO: CONFIGSETNOS;
            CONFIGTYPE: PIECECONFIGURATIONS;
        CONFIGPOSITION: AMTSFORCONFIGURATION;
        SORTEDPOSITION: ARRAY [PIECECONFIGURATIONS,AMTSFORCONFIGUR]
                           OF PIECEPOSITIONS;
            TESTCONFIG: PIECECONFIGURATIONS;
    NUMBERPOSITIONSFOR: ARRAY [PIECECONFIGURATIONS] OF POSITIONS;


   PROCEDURE ROTATE(W:AXES);
      VAR CUBE: 1..4;
      BEGIN
         FOR CUBE:=1 TO 4 DO WITH SOMA[PIECE] DO
            WITH ONE[UNPACKED[CUBE]] DO
               UNPACKED[CUBE]:=ROTATEABOUT[W];
      END /*ROTATE*/;


   FUNCTION PIECEFITS(TRANSLATION: CUBESP1): BOOLEAN;
      VAR
                         CUBE: PIECECUBES;
         XSHIFT,YSHIFT,ZSHIFT: -2..2;
                        SHIFT: SHIFTS;
                     TESTCUBE: CUBES;
                     LASTCUBE: CUBESP1;
      BEGIN
         IF TRANSLATION=27 THEN PIECEFITS:=TRUE ELSE
         WITH SOMA[PIECE] DO
         BEGIN
            TEMPORARYPIECE:=[]; PIECEFITS:=TRUE;
            FOR CUBETYPE:=EDGE TO CENTER DO COUNTOF[CUBETYPE]:=0;
            SHIFT:=UNPACKED[1]-TRANSLATION;
            XSHIFT:=TRANSLATION MOD 3    -
                    UNPACKED[1] MOD 3;
            YSHIFT:=(TRANSLATION DIV 3) MOD 3    -
                    (UNPACKED[1] DIV 3) MOD 3;
            ZSHIFT:=TRANSLATION DIV 9    -
                    UNPACKED[1] DIV 9;
            LASTCUBE:=27;
            FOR CUBE:=1 TO 4 DO
            WITH ONE[UNPACKED[CUBE]] DO
               IF NOT([XSHIFT+2,YSHIFT+7,ZSHIFT+12] LE
                  ALLOWEDSHIFT) THEN PIECEFITS:=FALSE
                  ELSE
                     BEGIN
                        TESTCUBE:=UNPACKED[CUBE] - SHIFT;
                        IF TESTCUBE ISNT LASTCUBE THEN
                        BEGIN
                           COUNTOF[TYPEOF[TESTCUBE]]:=
                              COUNTOF[TYPEOF[TESTCUBE]] + 1;
                           TEMPORARYPIECE:=TEMPORARYPIECE
                                           JOIN [TESTCUBE];
                        END /*IF TESTCUBE*/;
                        LASTCUBE:=TESTCUBE;
                     END /*IF*/;
         END /*ELSE*/;
      END /*PIECEFITS*/;


   PROCEDURE FINDALLOWEDSHIFTS;
      VAR          CUBE: CUBES;
         AXISSHIFT,LIMITS: -2..2;
         SETINDEX: 2..12;
              AXIS: AXES;
      BEGIN
         FOR CUBE:=0 TO 26 DO
         WITH ONE[CUBE] DO
         BEGIN
            ALLOWEDSHIFT:=[];
            FOR AXIS:=Z TO X DO
            BEGIN
               CASE AXIS OF
                  X: BEGIN LIMITS:=CUBE MOD 3; SETINDEX:=2 END;
                  Y: BEGIN LIMITS:=CUBE DIV 3 MOD 3; SETINDEX:=7 END;
                  Z: BEGIN LIMITS:=CUBE DIV 9; SETINDEX:=12 END;
               END /*CASE*/;
               FOR AXISSHIFT:= -LIMITS TO (2-LIMITS) DO
               ALLOWEDSHIFT:=
                    ALLOWEDSHIFT JOIN [AXISSHIFT+SETINDEX];
            END /*FOR AXIS*/;
         END /*FOR CUBE*/;
      END /*FIND ALLOWEDSHIFTS*/;
```

```
BEGIN /*FINDALLPIECEPOSITIONS*/
   FINDALLOWEDSHIFTS;
   CONFIGURATIONSETNO:=0;
   WRITE(EOL,≡1THE SET C(P) IS:≡,EOL);
   WRITE(≡                         C  F  V  ≡      (ALL POSS. CONFIGUR.)≡);
   WRITE(EOL);   WEOR(OUTPUT);
   FOR PIECE:=1 TO 7 DO
   WITH SOMA[PIECE] DO
   BEGIN
      FOR CONFIGTYPE:=0000 TO K3321 DO
         NUMBERPOSITIONSFOR[CONFIGTYPE]:=0;
      POSITION:=0;
      FOR ROTATION:=1 TO 24 DO
      BEGIN
         TRANSLATION:=0;
         WHILE NOT PIECEFITS(TRANSLATION) DO
            TRANSLATION:=TRANSLATION+1;
         IF TRANSLATION LT 27 THEN
         BEGIN
            PACKD[POSITION+1]:=TEMPORARYPIECE;
            TESTPOS:=1;
            WHILE TEMPORARYPIECE ISNT PACKD[TESTPOS] DO
               TESTPOS:=TESTPOS+1;
            IF TESTPOS = POSITION+1 THEN
               FOR TRANSLATION:=TRANSLATION TO 26 DO
                  IF PIECEFITS(TRANSLATION) THEN
                  BEGIN
                     CONFIGTYPE:=0;
                     POSITION:=POSITION+1;
                     PACKD[POSITION]:=TEMPORARYPIECE;
                     FOR CUBETYPE:=EDGE TO CENTER DO
                        CONFIGTYPE:=CONFIGTYPE*4 +
                              COUNTOF[CUBETYPE];
                     NUMBERPOSITIONSFOR[CONFIGTYPE]:=
                        NUMBERPOSITIONSFOR[CONFIGTYPE] + 1;
                     SORTEDPOSITION[CONFIGTYPE,NUMBERPOSITIONSFOR
                              [CONFIGTYPE]] := TEMPORARYPIECE;
                  END /*IF PIECEFITS, FOR TRANS., IF TESTPOS*/
            END /*IF TRANSLATION*/;
         ROTATE(Z);
         IF ROTATION MOD 4 IS 0 THEN ROTATE(Y);
         IF ROTATION IS 16 THEN ROTATE(X);
         IF ROTATION IS 20 THEN ROTATE(Y);
      END /*FOR ROTATION*/;
      FIRSTCONFIG:=CONFIGURATIONSETNO+1;
      POSITION:=0;
      FOR CONFIGTYPE:=0000 TO K3321 DO
         IF NUMBERPOSITIONSFOR[CONFIGTYPE] GT 0 THEN
         BEGIN
            TESTCONFIG:=CONFIGTYPE;
            CONFIGURATIONSETNO:=CONFIGURATIONSETNO + 1;
            WRITE(≡    EL. NO. ≡,CONFIGURATSETNO:2,≡: [≡);
            FOR CUBETYPE:=CENTER DOWNTO EDGE DO
            BEGIN
               NUMBEROF[CUBETYPE,CONFIGURATIONSETNO]
                              :=TESTCONFIG MOD 4;
               WRITE((TESTCONFIG MOD 4):2,≡,≡);
               TESTCONFIG:=TESTCONFIG DIV 4;
            END;
            WRITE(≡]≡,EOL); WEOR(OUTPUT);
            CONFIGPOSITION:=1;
            FIRSTPOSITION[CONFIGURATIONSETNO]:=POSITION + 1;
            FOR POSITION:=POSITION+1 TO (POSITION +
                          NUMBERPOSITIONSFOR[CONFIGTYPE]) DO
            BEGIN
               PACKD[POSITION] :=
                  SORTEDPOSITION[CONFIGTYPE,CONFIGPOSITION];
               CONFIGPOSITION:=CONFIGPOSITION+1;
            END;
            LASTPOSITION[CONFIGURATIONSETNO]:=POSITION;
         END /*IF NUMBERPOSITONSFOR*/;
      LASTCONFIG:=CONFIGURATIONSETNO;
      WRITE(≡                                           ≡);
      WRITE(NAME,≡ PIECE HAS≡,LASTPOSITION[CONFIGURATIONSETNO]:4,
         ≡ POSITIONS.≡,EOL);
      WEOR(OUTPUT);
   END /*FOR PIECE*/;
   FOR CONFIGTYPE:=SOMA[1].FIRSTCONFIG TO SOMA[1].LASTCONFIG DO
      LASTPOSITION[CONFIGTYPE]:=FIRSTPOSITION[CONFIGTYPE];
END /*FINDALLPIECEPOSITIONS*/;


PROCEDURE FINDHOLEAT(CUBE: CUBESP1);
   VAR  SIDE: SIDES;
   BEGIN
      TESTSHAPE:=TESTSHAPE JOIN [CUBE];
      HOLESIZE:=HOLESIZE+1;
      HOLECUBE[HOLESIZE]:=CUBE;
      WITH ADJACENTTO[CUBE] DO
      IF NOT (CUBESURROUNDED LE TESTSHAPE) THEN
      FOR SIDE:=1 TO NUMBERADJSIDES DO
         IF NOT (NEIGHBORING[SIDE] IN TESTSHAPE)
         THEN FINDHOLEAT(NEIGHBORING[SIDE]);
   END /*FINDHOLEAT*/;


PROCEDURE FINDSURROUNDEDCUBEPOSITIONS;
   VAR SIDE: SIDES;
   BEGIN
      FOR CUBE:=0 TO 27 DO
      WITH ADJACENTTO[CUBE] DO
      BEGIN
         CUBESURROUNDED:=[];
         FOR SIDE:=1 TO NUMBERADJSIDES DO
            CUBESURROUNDED:=CUBESURROUNDED JOIN [NEIGHBORING[SIDE]];
      END /*FOR CUBE*/;
   END /*FINDSURROUNDEDCUBEPOSITIONS*/;


PROCEDURE STOREDPELEMENT;
   VAR  PIECE: PIECES;
   BEGIN
      COMBNUMBER:=COMBNUMBER+1;
      WRITE(≡    ELEM.NO.≡,COMBNUMBER:3,≡: [≡);
      FOR PIECE:=1 TO 7 DO WITH SOMA[PIECE] DO
      BEGIN
         CONFIGCOMBINATION[COMBNUMBER]:=CURRENTCONFIG;
         WRITE(CURRENTCONFIG:2,≡,≡);
      END;
      WRITE(≡]≡,EOL);
   END /*STORE DPELEMENT*/;
```

```
PROCEDURE FINDALLCONFIGURATIONCOMBINATIONS;
   VAR
      CONFIGURATIONSETNO: CONFIGSETNOS;
        COMBINATIONVALID: BOOLEAN;
   BEGIN
      WITH SOMA[PIECE] DO
         FOR CONFIGURATIONSETNO:=FIRSTCONFIG TO LASTCONFIG DO
         BEGIN
            COMBINATIONVALID:=TRUE;
            FOR CUBETYPE:=EDGE TO CENTER DO
            BEGIN
               COUNTOF[CUBETYPE]:=COUNTOF[CUBETYPE]+
                  NUMBEROF[CUBETYPE,CONFIGURATIONSETNO];
               IF COUNTOF[CUBETYPE] GT MAXALLOWED[CUBETYPE] THEN
                  COMBINATIONVALID:=FALSE;
            END;
            IF COMBINATIONVALID THEN
            BEGIN
               PIECE:=PIECE+1;
               CURRENTCONFIG:=CONFIGURATIONSETNO;
               IF PIECE = 8 THEN STOREDPELEMENT
               ELSE FINDALLCONFIGURATIONCOMBINATIONS;
               PIECE:=PIECE-1;
            END;
            FOR CUBETYPE:=EDGE TO CENTER DO
               COUNTOF[CUBETYPE]:=COUNTOF[CUBETYPE]
                     - NUMBEROF[CUBETYPE,CONFIGURATIONSETNO];
         END /*FOR*/;
   END /*FINDALLCONFIGURATIONCOMBINATIONS*/;


PROCEDURE PRINTSOLUTION;
   VAR PIECE: PIECES;
   BEGIN
      SOLUTIONNUMBER:=SOLUTIONNUMBER+1;
      WRITE(≡ SOL.≡,SOLUTIONNUMBER:4);
      FOR PIECE:=1 TO 7 DO
      WITH SOMA[PIECE] DO
      BEGIN
         WRITE(≡  ≡,NAME,≡=≡);
         PRINTSET(PRINTLIST);
      END /*FOR PIECE*/;
      WRITE(EOL);      WEOR(OUTPUT);
   END /*PRINTSOLUTION*/;


PROCEDURE GENERATE;
   VAR
      HOLEISNOTFILLABLE: BOOLEAN;
      TESTPIECE: PIECEPOSITIONS;
      POSITION: POSITIONS;
   BEGIN
      WITH SOMA[PIECE] DO
      FOR POSITION := FIRSTPOSITION[CONFIGCOMBINATION[COMBNUMBER]]
                  TO LASTPOSITION[CONFIGCOMBINATION[COMBNUMBER]] DO
      BEGIN
         TESTPIECE:=PACKD[POSITION];
         IF SOLUTIONSHAPE MEET TESTPIECE IS [] THEN
         BEGIN
            TESTSHAPE:=SOLUTIONSHAPE JOIN TESTPIECE;
            CUBE:=-1;
            REPEAT
               REPEAT CUBE:=CUBE+1
               UNTIL NOT (CUBE IN TESTSHAPE);
               HOLESIZE:=0;
               FINDHOLEAT(CUBE);
               IF HOLESIZE IS 3 THEN
                  HOLEISNOTFILLABLE :=
                     (HOLECUBE[2] - HOLECUBE[1]) =
                     (HOLECUBE[3] - HOLECUBE[2])
               ELSE HOLEISNOTFILLABLE:= HOLESIZE MOD 4 IN [1,2];
            UNTIL HOLEISNOTFILLABLE;
            IF CUBE IS 27 THEN
               BEGIN
                  PRINTLIST:=TESTPIECE;
                  SOLUTIONSHAPE:=SOLUTIONSHAPE JOIN TESTPIECE;
                  PIECE:=PIECE+1;
                  IF PIECE IS 8 THEN PRINTSOLUTION ELSE GENERATE;
                  PIECE:=PIECE-1;
                  SOLUTIONSHAPE:=SOLUTIONSHAPE - TESTPIECE;
               END /*IF CUBE IS 27*/
         END /*IF*/
      END /*FOR*/
   END /*GENERATE*/;


BEGIN
   INITIALIZEVARIABLES;
   FINDALLPIECEPOSITIONS;
   COMBNUMBER:=0;
   FOR CUBETYPE:=EDGE TO CENTER DO COUNTOF[CUBETYPE]:=0;
   WRITE(EOL,≡1THE SET D(P) IS:≡,EOL);
   WEOR(OUTPUT);
   FINDALLCONFIGURATIONCOMBINATIONS;
   FINDSURROUNDEDCUBEPOSITIONS;
   WRITE(≡1≡,EOL);
   FOR COMBNUMBER:=COMBNUMBER DOWNTO 1 DO GENERATE;
END /*MAIN*/.
```

IN DEFENSE OF FORMATTED INPUT

John Eisenberg

University of Delaware

One of the few things which, in my opinion, mars the "symmetry" of Pascal is its lack of formatted input. I would like to present some examples of "typical" basic computer science problems which would be inconvenient or next to impossible without the use of formatted input.

<u>Example 1</u>: A survey has been given to 60 members of a Psych 100 class. The survey consists of rating 75 candidates for political office on a scale of 1 to 5. These data have been punched onto 60 cards, each containing a contiguous stream of 75 digits. Your task is to produce a table giving the mean rating for each candidate, along with the standard deviation, skewness, and kurtosis of the ratings.

This problem can, of course, be solved without formatted input by defining a string, and using ord(ch)-ord('0') to extract digits. However, this is hardly the point of the problem, and, in fact, detracts from the true object of the exercise. The psych student is really not interested in the use of "ord" or strings at this point; he is interested in statistics.

<u>Example 2</u>: Your Ph.D. Advisor has been teaching an undergrad course, and has compiled a card deck with the following information on it: Columns 1-9 contain a student's social security number; columns 11 to 70 contain 20 3-digit numbers, each of which represents a score on a pop quiz (ranging from 0 to 100). Your task is to print out the mean and standard deviation of each quiz, a correlation matrix of all quizzes, and a

frequency table of the following form:

```
RANGE     N
-----     -

100-90    25
89-80     35        (etc.)
```

Again, it is possible to use "ord" to extract the three-digit numbers, but it is even less pleasant than before, and detracts even further from the point of the exercise.

At this point, someone will object that there is no need to punch the data as a contiguous stream of numbers, and I agree. However, I will note that most psych surveys I have seen try to put as much data on a single card as possible. (Note that we could not put all the data on one card if we put blanks between numbers.) Besides saving keypunching time and storage space (both of which are the equivalent of money!), the people doing the surveys are interested in the resultant statistics, and couldn't care less about nicely-spaced input. In fact, I have even been told to keypunch real numbers as a contiguous stream in order to avoid "wasting time typing spaces!"

Now, for a two-part example where formatted input is a near-necessity. This is a business problem, but I feel justified in giving it as an example -- Wirth, in the Pascal report, says he put records and files into Pascal in order to "...make it possible to solve commercial type problems with Pascal, or at least employ it successfully to demonstrate such problems in a programming course."

<u>Example 3</u>: The owner of a local jeans store has hired you, ace programmer, to do his inventory control and market research.

<u>Part I</u>:

Each day, you are given a deck of punched cards giving the day's sales total; one card per transaction. Each card looks like this:

```
     Columns 1-7     Serial number of jeans
     Columns 9-10    Waist size (range 26..54)
     Columns 12-13   Length (range 26..38)
     Columns 15-19   Quantity sold
```

You may assume that transaction records have been sorted by serial number; thus all purchases of #5050217 will come before all purchases of #7070217. Your task is to print out a chart for each serial number giving a sales record for that day.

So far, so good. No formatted input has been needed to accomplish the first half of the example. Now, the clincher:

Part II:

As we all know, the serial number on your pair of jeans is not arbitrary; the 5050217 signifies regular blue denim; the 7070217 signifies bell-bottoms. In fact, the serial number is formatted as follows:

```
First 3 digits style
               505=regular, 707=bells, 303=cutoffs
Next 2 digits  color
               02=blue,03=green,04=brown
Last 2 digits  material
               17=denim,25=hopsack,36=doubleknit
```

Your task is to take 5 days of purchases, and produce a chart for each color, style, and material that shows the sales distribution by size. What conclusions can you draw from your results?

This example is hardly far-fetched; most identifying numbers on clothing contain this sort of packed information. The point is, however, that there are a lot of applications which use this sort

of data representation. Other examples include:
1) Indiana license plates, whose first letter and digit serve as an index to an alphabetical list of county names.
2) Illinois driver's license numbers, which contain year of birth, county of residence, and oodles of other information to help prevent forgery of the license, all packed into 11 digits.

In all these cases, repunching the data to separate the fields is clearly out of the question. These examples point out the need, or at least the great desirability, of having formatted input.

Although I will agree that it is very easy for an installation to write its own formatted input routines as standard functions, this brings up the problem of transportability. It seems clear that the feature is useful enough to be "built-in" by many Pascal users. However, these site-specific intrinsics will not transport easily from installation to installation. This is, in fact, one of the failings of BASIC; people provided their own functions and "extensions" to do things that they needed (and BASIC didn't have). Now, one would be hard-pressed to find two places where all the "extensions" agree.

In conclusion, it appears that formatted input can be useful in many cases, and almost necessary in others. It seems appropriate, then, that such a feature should be added at the base language level rather than having every Pascal installation re-invent its own formatted input wheel.

(*Received 4/26/76*)

Overlays: A Proposal

James F. Miner
Social Science Research Facilities Center
25 Blegan Hall
University of Minnesota
Minneapolis, Minnesota 55455

As Pascal gains wider availability and use, it shows itself to be a versatile and practical tool for serious production work. However, given the recency of implementations, it is understandable that a number of implementation features commonly employed in production work are not yet available. For some programming projects, the ability to reduce the amount of storage required for a program's object code is of fundamental importance. Even if a program is proven to be correct, it may not fit on a given machine or it may suffer from poor performance due to operating system restrictions. Both of these conditions occur in practice, and a number of techniques for avoiding them are widely used. The use of overlays is one such technique. An implementation extension for Pascal is hereby proposed to provide this capability.

The goals of this effort are numerous, including effectiveness of the overlay mechanism in providing space reductions, simplicity of notation and understanding for the user, security from unintended ill-defined results, and efficiency of both compilation and execution mechanisms. In addition the scheme should be implementable in a wide variety of environments.

Usage Considerations.

Pascal, with it s block structure, provides an excellent opportunity for the implementation of a secure, simple, and relatively transparent overlaying scheme. Procedures (and functions) often delimit major sections of the program which will be executed either infrequently, or which have (nearly) disjoint calling sequences, or both. Thus the overlay scheme, which requires division of a program into overlays, should take advantage of the existing division into procedures. This should be done in such a way that the effects of the overlaying are directly visible to the programmer and can be easily controlled. In my opinion, this can best be accomplished by the indication of overlay structuring at the source level. It is at this level that the programmer interacts with and understands the program.

Let us assume that the programmer may specify any procedure (or function) to be an overlay with a prefix symbol such as OVERLAY before the procedure heading. The prefix has the advantage of being short and easy to use, but as visible as the procedure heading itself. The extent of the overlay is defined by the procedure body, so confusion is minimized. And finally, the compiler is informed immediately when it is about to process an overlay -- the significance of which depending upon the implementation.

What meaning should the programmer attach to the OVERLAY prefix? The best answer should be "no effect", in terms of the correctness of his program or the restrictions placed upon him by the implementation. The programmer must be advised that the invocation of an overlay will be more expensive than the invokation of a normal procedure, but the decision of how to avoid excessive overhead should be left to the user.

Some implementation restrictions may be reasonable, such as against passing overlays as parameters (i.e., formal overlays), or passing procedures or functions as parameters to overlays. A complete implementation would allow both of these cases, but they might be difficult to implement, and both can be readily detected (and thus restricted) at compile time. Disallowing recursive overlay calls is an entirely different situation because compile time checks are ineffective, and run time tests are to be avoided if possible. In addition, the normal scope rules for calling should not be constrained for overlays; functions as well as procedures should be overlay-able; and by all means, parameters and non-local references should be allowed. These latter three capabilities are desirable to maintain the basic similarity between normal and overlayed procedures.

Finally, it is always possible for an implementation to simply ignore the OVERLAY prefix (just as PACKED might be ignored) either entirely or only after a given static overlay nesting depth limit has been reached.

Implementation Considerations.

It is instructive to examine the common characteristics of overlay mechanisms used with Fortran. Of those with which I am familiar, three notable aspects are:

(1) Overlays are classified into levels, where only one overlay at a given level can be in core at one time. That is, overlays of the same level are loaded over each other. (Depending on the scheme, overlays at different levels may or may not overlap in core.) The lowest level overlay constitutes a main overlay which is always in core, which first receives control when the program is executed, and which initially causes other overlays to be loaded. The run time system (RTS) usually is contained in the main overlay.

(2) In calls between overlays a load operation may be necessary so such calls either are trapped, or are made explicitly by the user, to a routine in the RTS which determines whether the called overlay is in core, and if not, causes it to be loaded and executed.

(3) By placing strategic constraints on calling sequences these schemes ensure that a routine to be returned to is in core. Thus the return can be performed exactly as though the program were not overlayed at all. The constraints necessary for this include prohibition of calls to overlays of the same or lower level which would cause overlay loading (thus possibly destroying the calling routine). Because the main overlay is always in core, calls to its routines are allowed (but may cause undefined results if overlay loading occurs, and a return is subsequently attempted).

The constraints placed upon calling sequences in the Fortran schemes may be 'reasonable' given the kind of language involved, but certainly represent weaknesses of security since compile time checks cannot prevent undefined results and run time checks are often not made.

The proposed Pascal overlay facility has the following features:

(1) The level of each overlay is determined at compile time from the static nesting of the source. These levels are analogous but not equivalent to block nesting levels. The compiler assigns a unique identification to each overlay, in the form of a m-tuple or vector of ordinals, where m is the maximum number of overlay levels in addition to the main level allowed by the implementation, and where the ordinal zero is reserved as a null marker. The nesting of the overlays is represented as follows:

(a) The main overlay is represented by an all-zero (null) vector.

(b) Overlays at the first level (after the global overlay) are identified by unique non-zero ordinals in the first element of their vectors. Remaining elements are zero.

(c) In general, overlays at the n-th level are represented by vectors with elements one through n non-zero, and n+1 through m zero. In addition, to represent the static nesting, all level n overlays which have the same encompassing overlays are assigned vectors differing only in the n-th elements.

(2) The configuration of overlays which are core resident at any point in time is represented by the Overlay Display Vector (ODV), the value of which is an m-tuple as defined in (1). The ODV is maintained by the run time system, and is modified only when overlays are loaded. The following rules are observed by the overlay mechanism:

(a) Before any block in an overlay at level n can be invoked, the ODV must equal the overlay's vector in at least the first n

elements. If this requires the loading of overlays, then the value of the ODV prior to the call must be saved on the stack.

(b) If before returning from a call the non-zero elements of the stacked ODV value are equal to the corresponding elements of the ODV, then no loading is required to complete the return.

(c) If loading is required to complete the return, then the i-th through the n-th level overlays indicated by the stacked ODV value must be loaded, where i is the lowest position of the stacked ODV value not equal to the corresponding position of the ODV, and where n is the highest non-zero position in the stacked ODV value.

(3) Assuming suitable restrictions on procedure and function parameters (below), the rules in (2) guarantee the following conditions:

(a) Encompassing blocks are guaranteed to be in core. Thus calls to and returns from encompassing blocks, and goto's which exit to encompassing blocks need not be trapped.

(b) Non-encompassing (including local) blocks must be in core unless prefixed. Thus calls to and returns from these blocks must be trapped only if the called block bears the OVERLAY prefix.

'Suitable restrictions' are that a procedure or function cannot be passed to blocks 'outside of' (non-local to) the overlay in which the parameter is declared. Thus, a block bearing the OVERLAY prefix may occur as an actual parameter only in calls to other blocks which are local to it.

(4) Easing the restrictions on procedure and function

parameters in (3) requires that the block-parameter descriptor (which contains the parameter's entry point and static link) be augmented by the tuple of the overlay which contains the actual parameter. All calls to and returns from formal parameters must be trapped.

While the above points define the basic mechanism, a number of problems may occur in specific implementations. These will usually result from attempts to use existing loader or operating system software or conventions. A simple example is where the loader may not provide information to the run time system to establish the area in core for heap and stack. The run time system should be provided with the space required for the largest possible configuration of overlays.

A more difficult problem may appear in the mapping of the proposed hierarchical overlay structure to the structure supported by the loader or operating system. The most serious mis-match will probably be in the restrictions enforced at load time on the number of entry points per overlay, or on the direction of calls between overlay levels. In addition, there may not be a 'nice' mapping between the proposed identification scheme and the existing conventions.

It is beyond the scope of this paper to delve into particular implementations, except to note that this scheme has been partially implemented for the CDC 6000/Cyber 70 series. It is hoped that this proposal will stimulate discussion on the topic of overlaying, and that interested readers will forward comments or criticism to the author or editor.

(*Received 7/9/76*)

"'MINOR' PROBLEMS IN PASCAL"

Timothy M. Bonham

In this article I intend to discuss some small but significant 'problems' or possible areas for improvement in PASCAL. Before I begin, I would like to include three sections: a disclaimer, an ommission, and a defense.

DISCLAIMER: First of all, I like PASCAL. If I didn't, I wouldn't spend time writing about it. I think that it's easily one of the best languages around. I greatly admire the logical clarity and 'austerity' of the language, the way program text tends to clearly show the progress of the actual execution, and the fact that the language tends to encourage good programming practices(compared to some languages(for example, FORTRAN or PL/1) which tend to do just the opposite.) But I don't think that the language is perfect; therefore it's worthwhile to try to find areas that might be improved.

OMMISSION: I will not discuss in this article two areas that are frequently said to be major deficiencies of PASCAL; the general read-write procedures (which are difficult to use, especially in an interactive environment) and the definition and manipulation of common business-type files(which is less clear in PASCAL than in some other languages). I will also not discuss some problems that I do not think are specifically the fault of the language itself; for example, the unpleasantness and unhandyness to the user of most present PASCAL implimentations.

DEFENSE: to the charge that I'm being picky about small and unimportant elements of the language--I don't think that they are so unimportant. These elements are some of the most frequently used in the language and often make up a large part of the program. Also, several references have indicated that it is 'minor' inconsistancies like these that cause a large part of the difficulty in learning a language and problems with such elements account for a very sizable proportion of the compilation errors in programs. In any case, I don't think simply classifing awkward elements as small and unimportant is any reason not to try to improve them.

The main portion of this article will note specific PASCAL elements that I feel are problems, the reasons for this, and some possible improvements.

1. identifiers-- It is commonly acknowledged that identifiers should be chosen so that they have mnemonic significance to the programmer. Well chosen identifiers provide a form of self-documentation within the program itself. The most common obstacle preventing mnemonicly significant identifiers is the limitation upon identifier length. PASCAL as defined has no specified maximum limit on identifier length. However, one of the goals of PASCAL(and presumably of programmers who use it) is to have a standard language "in order to facilitate the interchange of programs". For this, programs must distinguish identifiers within the first eight characters as required by Standard PASCAL. This is clearly a harsh restriction. FORTRAN is well known for encouraging incomprehensible variable names, and Standard FORTRAN permits six characters, while many present compilers permit seven,eight or even ten characters. In this area, PASCAL is only minimally better than Standard FORTRAN and equal or even worse than many present day FORTRAN compilers.

A further problem with identifiers in PASCAL is the lack of a break character. This commonly results in identifiers which consist of several words run together, with resulting loss of clarity. The recommendation of a specific break character presents difficulties. The use of the hyphen(as in COBOL) maintains similarity to normal language usage, but requires additional restrictions on the spacing of expressions to distinguish the use of the same symbol for subtraction(though these restrictions are not unfamilar, since they are the same as those normally used in writing and typesetting). The use of a seperate character for breaks(like the underscore in PL/1) imposes no such additional restrictions , but requires the addition of another element to the character set. The use of the space character as a break character(as in ALGOL) is unpleasant in several respects: it introduces unfortunate complexities into the compiler; it provides serious potential for misuse(it is easy to make the same identifier look very different); and(though this may be just an oversight), the space character is not included in the basic symbols of Standard PASCAL.Whatever character is chosen, the availability of a seperator character clearly enhances the readibility of programs.

2. the subrange symbol(..)--the use of this symbol as an abbreviated form of listing a subrange of a type is very unfortunate in that it is similar to, but not quite the same as the ellipsis (...) commonly used in the English languang and in mathematical notation. Items such as this should be either exactly the same, or sufficently different to prevent confusion; the worst case is to be 'almost' the same. In at least one published case, an author(or possibly, his typesetter) unconsciously used three periods instead of two throughout his entire program, and was subsequently criticized for it. I consider the main error not to be that of this user, but of the language, which positively encourages this type of user error. I cannot think that the saving of a single character in writing the program is worth this confusion. (A similar example in PL/1: many numerical formats are the same as those in FORTRAN-except for the use of a comma rather than a period. This has caused vast--and completely unnecessary-- anguish to FORTRAN programmers attempting to work in PL/1).

3. the assignment symbol(:=)--the programmers operation of assignment is very different from the mathematicians assertion of equality and the language should clearly distinguish the two. Anyone who has ever tried to explain the statement X = X + 1 to a student learning BASIC knows this: "you see, = means equals only in IF statements, otherwise, = doesn't mean equals, it means...". The use of a symbol other than the equals sign for assignment(unlike BASIC and PL/1) is good, but the symbol chosen(:=) perpetuates the confusion by containing within it an echo of the equality symbol. A better symbol might be the left arrow (←) as used in APL; which does not have such confusion. Also, the left arrow more clearly reflects the action of the program during execution; it is, as has been said, "almost onomatopetic". In addition, this symbol is easily recognized by APL users (the := may be readily recognizable by ALGOL users, but there are more users of APL than ALGOL, and their lead is growing). I recognize that this symbol is not available on all I-O devices, but it is on the majority of them, and on others it could be replaced by a two character synonym, similar to the way comment symbols are handled. (perhaps the less than and hyphen symbols (<-) would be good).

In addition, the left arrow symbol saves one character, a minor advantage; more importantly, it provides higher relialibility. In the := symbol, there is a chance of skipping one of the characters in the symbol and thus converting it into another valid symbol. (Luckily, in PASCAL, unlike some languages, such an accidentally created construct would always be invalid at its present place in the program, and thus should generate an error message).

4. ascending directional symbol of the for clause (to)-- This symbol itself, when encountered in a program, gives no specific indication of direction to the reader. The user is able to discern that the value changes occur in an ascending sequence only through comparison to the alternative symbol, downto which clearly indicates a descending sequence(or through a study of the language manual). A better symbol, which would make the direction of change readily apparent to the reader; and also be more congruent with the alternative symbol downto, would be the symbol upto. The additional clarity to the readerwould, I feel, be easily worth the minimal cost of two extra characters in the symbol.

5. A final controversial issue is that of the comment structure.
There are three main types of comments that are

commonly included within a program: prologue comments--this is general information about the program (author, date-written, size, etc.); block summary--these comments emphasize the function and purpose or indicate the logical unity of a block of code; local information--this spotlights critical steps or meanings for a small section of code. These three types of comments are clearly different in both their nature and their location within the program. The comment structure should clearly distinguish them. Unfortunately, the comment scheme of PASCAL does not do so; all must be written using the same (* and *) brackets. This scheme, though flexible(in location) and versatile (regarding length of comments) lacks reliability; forgetting (or mispunching) the closing bracket turns the remainder of the program into comments.

Prologue comments are normally located near the beginning of the program; they consist of a solid block of commentary, usually of a fairly standardized nature. The best structure for these is a dedicated block with a predefined, standard format, similar to that of the Identification Division of COBOL. The fill-in-the-blank nature of such a structure is especially important in view of its tendancy to encourage the automatic inclusion of such a block. As an example of this, both COBOL and FORTRAN provide a mechanism for including such information within the program; COBOL in a predefined format, and FORTRAN through the use of the general comment scheme; yet many more COBOL programs include this information than FORTRAN programs.

Block summary comments are usually located at at the beginning and/or end of various blocks of the program. They normally consist of several full lines of comments. The present PASCAL comment structure seems to work well here, though the scheme presented below for local information might prove better.

Local information comments are usually made up of a line or two before a statement or the last half of the lines that the statement is written on. A possible improvement over the present PASCAL comment structure might be the use of a symbol like { or (* to indicate that the remainder of the present line is to be considered commentary. This scheme uses the end of the line as an implicit closing bracket to the comment. This improves the reliability of the comment structure (by eliminating the problem of the missing ending bracket) and accommodates both of the styles of loacl information comments mentioned. The only cost to the language user is the neccessity of repeating the beginning symbol if the comment is three or more lines in length. Local information comments normally are not, and should not be this long.

Overall, it seems to me that PASCAL has eliminated most of the major faults of many other languages; but, unfortunately, has also eliminated some of the (often minor) elements of these languages that worked well. This is easy to do, a languages vices are much more noticed than its virtues. I feel that the elements that I have discussed in this article can be changed without altering the main philisophy of PASCAL and, probably, without much difficulty in modifing the compilers.

I hope that this article will stimulate discussion on these issues; and would gladly welcome comments on it; either in private communications to me (Tim Bonham, D605/1630 S. 6th. St., Minneapolis, MN 55454) or in items published in the PUG newsletter.

(*Received 7/15/76*)

Dynamic Array Parameters

by Ch. Jacobi, E.T.H., Zurich

(*What follows are pages 6-10 of a proposed description of dynamic array parameters which is probably of interest to many PUG members. Pages 1-5 were not sent; they were in German.*)

Corrections to PASCAL - User Manual and Report

p.72, 1.12 and 1.13
'11.3' --> '11.4'

p.73, 1.4
It is also possible to write a procedure without fixing the bounds of the index type of array parameter. In the formal parameter section the name is not followed by its type, but by an array declaration which has a scalar index type. The bounds of the index types then are taken for each parameter from the corresponding actual parameter. These array parameters are called dynamic array.
Program 11.3 shows the case of a dynamic array parameter. The standard functions low and high deliver the index bounds.

p.83 bottom
4. A dynamic array must not appear as an actual parameter if the corresponding formal parameter is not a dynamic array. (Depending on which version is implemented)
Neither must they have dynamic array parameter.

(Depending on which version is implemented)
5. Dynamic arrays will be packed only in one dimension. A dynamic array declaration

```
        packed array[scalar1,... ,scalarN] of type-id
```

accepts as actual parameter, beside an equally declared dynamic array, also an array declared as

```
    array[scalar1,... ] of packed array[scalarN] of ...
```

p.97, 1.31
5. The standard procedures pack and unpack must not have dynamic arrays as actual parameter.

p.103, 19
'and parameters' --> ', parameters and index bounds of dynamic arrays.'

p.107 bottom

low(x)	x is a dynamic array variable, the result is the lower bound of the index of the array.
high(x)	x is a dynamic array variable, the result is the upper bound of the index of the array.

p.108, 1.5
and dynamic array types.

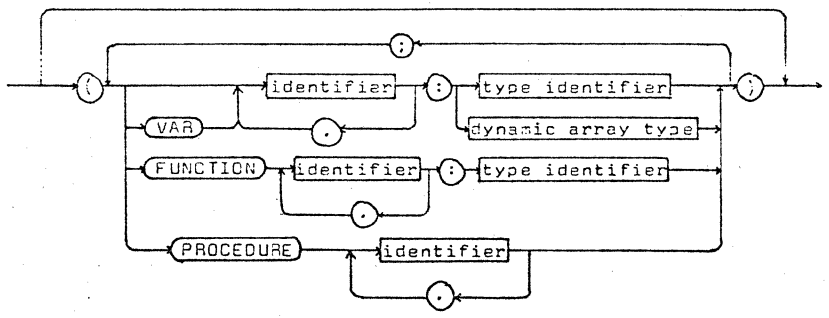
p.109, 1.13
low, high

p.112 replace 1.20 - 1.22
(Depending on which version is implemented, the part with packed should be omitted.)

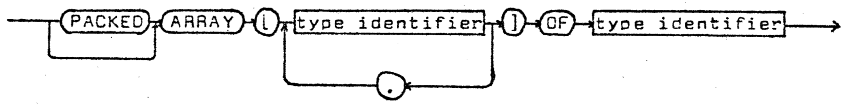
```
<formal parameter section> ::= <extended parameter group> |
        var <extended parameter group> |
        function <parameter group> |
        procedure <identifier> {,<identifier>}

<extended parameter group> ::= <identifier> {, <identifier>} :
        <type identifier> |
        <identifier> {, <identifier>} : <dynamic array type>

<dynamic array type> ::= array [ <scalar type identifier>
        {, <scalar type identifier>} ] of <type identifier> |
        packed array [ <scalar type identifier>
        {, <scalar type identifier>} ] of <type identifier>

<scalar type identifier> ::= <identifier>
```

p.117 replace the diagram parameter list

parameter list



dynamic array type



p.121 bottom

400: Dynamic arrays must have a scalar index type
401: Packing and unpacking of dynamic arrays are not
     implemented
402: Dynamic array expected
403: Type identifier or dynamic array type expected

(Depending on which version is implemented)
404: Formal procedure must not have dynamic array parameter

p.126 alphabetical
dynamic array   11.A

p.151 after last line of 8.1.4.
In this case the arrays must not be dynamic.

p.152, l.24
The variable and the expression must not be dynamic arrays.

p.158 replace l.35 - l.39
(Depending on which version is implemented,
the part with packed should be omitted.)

```
<formal parameter section> ::= <extended parameter group> |
    var <extended parameter group> |
    function <parameter group> |
    procedure <identifier> {,<identifier>}

<extended parameter group> ::= <identifier> {, <identifier>} :
    <type identifier> |
    <identifier> {, <identifier>} : <dynamic array type>
<dynamic array type> ::= array [ <scalar type identifier>
    {, <scalar type identifier>} ] of <type identifier> |
    packed array [ <scalar type identifier>
    {, <scalar type identifier>} ] of <type identifier>
<scalar type identifier> ::= <identifier>
```

p.158, l.-1
The index type of a dynamic array type is substituted for every parameter by the subrange used as index in the corresponding actual parameter.

p.159 bottom
(Depending on which version is implemented)
Packed dynamic arrays are only packed over their last dimension. An array of the form

            packed array[scalar1,scalar2] of typeA

must in the calling procedure be declared as

        array[scalar1] of packed array[scalar2] of typeA.

p.164, l.18

low(x)	x is of dynamic array type, and the result is the lower bound of the index type of the corresponding actual parameter for x.
high(x)	x is of dynamic array type, and the result is the upper bound of the index type of the corresponding actual parameter for x.

p.166 bottom
8. Dynamic array types should only be used elementwise
   (Exceptions are procedure and function parameter)

p.169 alphabetical
dynamic array   10.

(*Received 7/22/76*)

502 E. Healey #207
Champaign, IL 61820
5 November 1975

Hello Andrew,

Enclosed find the only information I have about the PDP 11/20 version of PASCAL. From a report of a friend using the compiler, there may still be bugs in it. I will try to get you a name of someone to contact about distribution, progress, etc. Unfortuaately I do not work with that group or their machine and have no real channel to them. I may do some PASCAL work on the machine, though, just to see how it is.

I do have some rather promising information in another area. Chances are you probably know about it already, but in case you don't: Wirth has come out with a subset of PASCAL (called PASCAL-S) and has written:

N. Wirth <u>PASCAL-S: A Subset and its Implementation</u>
Eidgenossische Technische Hochschule Zurich
Institut fur Informatik -- Report # 12

I believe the date on it is June 1975--but don't quote me! I got a copy of the manual from Dr. M.D. Mickunas, who received it from Dr. Jurg Nievergelt. Dr. Nievergelt was on sabbatical in Switzerland and sent it back from there. (Both teach here at the University of Illinois). I'm enclosing 2 pages detailing the major differences between the subset and full PASCAL.

If you want a copy of the whole schmeer, I'll be glad to get one and mail it to you. (Again, if you don't already have one).

One noteworthy feature of the manual is its chapter 7, which contains a version of the compiler and interpreter, written in full PASCAL. The compiler produces code for a hypothetical stack machine, which the interpreter (naturally) interprets.

The following is pure speculation (especially pending some funding) but: Rick Simkin and I are thinking of translating it for the 360. We were originally thinking of rewriting it into PL/I but neither of us is terribly keen on it. (Even though it would be the easiest, I guess I for one just want to avoid PL/I wherever possible) What we may well do is have me write the compiler in SPITBOL (for speed of character manipulation) and have Rick write the interpreter in FORTRAN (for raw speed). I will definitely get back to you on this if we can con anyone into giving us funds to do it.

That is basically all the info I have for right now.

As for your request for a new name for the group, how about

<u>A</u>ssociation of <u>P</u>ascal <u>P</u>rogrammers, <u>L</u>td. (APPL)

Sorry, that's the best I could come up with at 11:30 PM. At any rate, I think we should try to avoid "<u>P</u>ASCAL <u>I</u>nterest <u>G</u>roup" unless we really want to shock and/or offend. The more I think about it, the better it sounds...must be the late hour.

Sorry also about the typing errors; I have never really fully believed in proofreading.

John Eisenberg
Eisenberg

PS--You may want to include phone numbers on future mailing lists; some developments or exchanges are not handled best by the US Postal Service. Also, most Universities have WATS lines for "University business."

PPS--My office phone: (217) 333-1719; the office is shared by a lot of people so don't be surprised if you hear a female voice answering...

p.72 insert new page

```
{ program 11.3
 extend program 11.2 }

program minmax4(input,output);

const n = 20; m= 15;
type listn = array[1..n] of integer;
     listm = array[1..m] of integer;
var a: listn; b: listm;
    i,min,max: integer;

procedure minmax(var g: array[integer] of integer;
                 var j,k: integer);
   var i,l,h,u,v: integer;
begin l := low(g); h := high(g);
   j := g[l]; k := j; i := l+1;
   while i<h do
   begin u := g[i]; v := g[i+1];
      if u>v then
      begin  if u>k then k := u;
             if v<j then j := v
      end else
      begin  if v>k then k := v;
             if u<j then j := u
      end;
      i := i+2
   end;
   if i=h then
      if g[h]>k then k := g[h]
      else if g[h]<j then j := g[h];
end; {minmax}

begin
   for i := 1 to n do
      begin read(a[i]);  write(a[i]:3)  end;
   writeln;
   minmax(a,min,max);
   writeln(min,max,max-min); writeln;
   for i := 1 to m do
      begin read(b[i]); write(b[i]:3)  end;
   writeln;
   minmax(b,min,max);
   writeln(min,max,max-min)
end.


 -1 -3  4  7  8 54 23 -5  3  9  9  9 -6 45 79  3  1  1  5
        -6          79          85

 45 43  3  6  1 34 -8  1  4 34  3  8 -1  3 -2
        -8          45          53
```

p.72 insert new page

```
{ program 11.3
 extend program 11.2 }

program minmax4(input,output);

const n = 20; m= 15;
type listn = array[1..n] of integer;
     listm = array[1..m] of integer;
var a: listn; b: listm;
    i,min,max: integer;

procedure minmax(var g: array[integer] of integer;
                      var j,k: integer);
   var i,l,h,u,v: integer;
begin l := low(g); h := high(g);
   j := g[l]; k := j; i := l+1;
   while i<h do
   begin u := g[i]; v := g[i+1];
      if u>v then
      begin  if u>k then k := u;
             if v<j then j := v
      end else
      begin  if v>k then k := v;
             if u<j then j := u
      end;
      i := i+2
   end;
   if i=h then
      if g[h]>k then k := g[h]
      else if g[h]<j then j := g[h];
end; {minmax}

begin
   for i := 1 to n do
      begin read(a[i]);  write(a[i]:3)  end;
   writeln;
   minmax(a,min,max);
   writeln(min,max,max-min); writeln;
   for i := 1 to m do
      begin read(b[i]); write(b[i]:3)  end;
   writeln;
   minmax(b,min,max);
   writeln(min,max,max-min)
end.


 -1 -3  4  7  8 54 23 -5  3  9  9  9 -6 45 79  3  1  1  5
        -6             79          85

 45 43  3  6  1 34 -8  1  4 34  3  8 -1  3 -2
        -8             45          53
```

502 E. Healey #207
Champaign, IL 61820
5 November 1975

Hello Andrew,

Enclosed find the only information I have about the PDP 11/20 version of PASCAL. From a report of a friend using the compiler, there may still be bugs in it. I will try to get you a name of someone to contact about distribution, progress, etc. Unfortuaately I do not work with that group or their machine and have no real channel to them. I may do some PASCAL work on the machine, though, just to see how it is.

I do have some rather promising information in another area. Chances are you probably know about it already, but in case you don't: Wirth has come out with a subset of PASCAL (called PASCAL-S) and has written:

N. Wirth <u>PASCAL-S: A Subset and its Implementation</u>
Eidgenossische Technische Hochschule Zurich
Institut fur Informatik -- Report # 12

I believe the date on it is June 1975--but don't quote me! I got a copy of the manual from Dr. M.D. Mickunas, who received it from Dr. Jurg Nievergelt. Dr. Nievergelt was on sabbatical in Switzerland and sent it back from there. (Both teach here at the University of Illinois). I'm enclosing 2 pages detailing the major differences between the subset and full PASCAL.

If you want a copy of the whole schmeer, I'll be glad to get one and mail it to you. (Again, if you don't already have one).

One noteworthy feature of the manual is its chapter 7, which contains a version of the compiler and interpreter, written in full PASCAL. The compiler produces code for a hypothetical stack machine, which the interpreter (naturally) interprets.

The following is pure speculation (especially pending some funding) but: Rick Simkin and I are thinking of translating it for the 360. We were originally thinking of rewriting it into PL/I but neither of us is terribly keen on it. (Even though it would be the easiest, I guess I for one just want to avoid PL/I wherever possible) What we may well do is have me write the compiler in SPITBOL (for speed of character manipulation) and have Rick write the interpreter in FORTRAN (for raw speed). I will definitely get back to you on this if we can con anyone into giving us funds to do it.

That is basically all the info I have for right now.

As for your request for a new name for the group, how about

<u>A</u>ssociation of <u>P</u>ascal <u>P</u>rogrammers, <u>L</u>td. (APPL)

Sorry, that's the best I could come up with at 11:30 PM. At any rate, I think we should try to avoid "<u>PASCAL</u> <u>I</u>nterest <u>G</u>roup" unless we really want to shock and/or offend. The more I think about it, the better it sounds...must be the late hour.

Sorry also about the typing errors; I have never really fully believed in proofreading.

John Eisenberg
Eisenberg

PS--You may want to include phone numbers on future mailing lists; some developments or exchanges are not handled best by the US Postal Service. Also, most Universities have WATS lines for "University business."

PPS--My office phone: (217) 333-1719; the office is shared by a lot of people so don't be surprised if you hear a female voice answering...

EIDGENÖSSISCHE
TECHNISCHE HOCHSCHULE ZÜRICH

Institut für Informatik

Clausiusstrasse 55
CH-8006 Zürich
Ø 01 / 32 62 11

Mr. Andrew B. Mickel
University of Minnesota
Computer Center
227 Experimental Eng. Bldg.
MINNEAPOLIS, Minnesota 55455

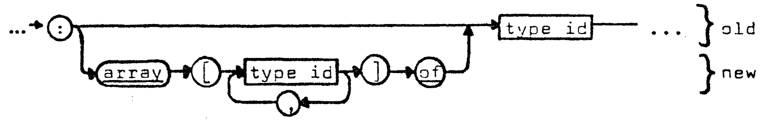
U.S.A.

November 22, 1975

Dear Andy,

I have some news for you:

1) One of our students is actually implementing dynamic array parameters as part of his work for his master degree. The PASCAL syntax will presumably be (change in PARAMETER LIST only):



ex. procedure p(a : array [integer] of real);

The additional standard functions LOW and HIGH deliver actual array bounds (in this example LOW(a,1)≡ LOW(a) is ok. While LOW(a,2) results in a compile time error message because "a" was declared to be a 1-dimensional array only. The second parameter must always be a constant.

In case that the implementation will prove to be well done (and compile FL is not charged too much) we will certainly consider its incorporation in Release 2. As this work will not be completed before early February there is no reason why you should send your tape by the end of November! Christmas or even early January would be fine.

2) Another student tries to implement a (type checking) value part as a semester job. If he succeeds I am willing to make it part of RELEASE 2 too. However, I strongly doubt whether he will find a feasible solution and work it out in time. In any case he will not complete his work before February.

By the way: I have no good arguments for the value part and against initialisation in the var part. Binding initialisations to types seems to be inappropriate because they are rather attributes to variables than to types. This is why I would prefer

```
    var  i =o,j = 1: integer ; c1 = (o,1), c2 = (1,o): complex
to
    var  i,j :integer = 0; c1,c2 :complex = (o,1).
```

- 2 -

However, with the former notation one pass compilation can become cumbersome because the type of the variable is not known at the moment the (possibly structured) value is read. Also, I guess that at the time declarations are written initial values may still be unknown. So a strong separation between declaration and initialisation is perhaps legitimate.

3) The idea of implementing a constructor function has been postponed. We have (for the time being) nobody who could do it. Our new assistant will only join us in January and not in December as I wrote you.

4) Hope-fully DISPOSE will not die! I will talk to Wirth about it. In my opinion NEW without an "inverse" is useless.

We were pleased to hear that the first PASCAL User's group meeting was (spontaneously!) held at ACM '75.(Two weeks ago a so called PASCAL-Day has been organised by the Swiss chapter of the ACM. About 50 people attended which, following the chairman, was quite a success).

Please inform us when you have news concerning duties of the Newsletter editorship.

Sincerely,

[illegible]

Urs Ammann

UA:ua

(SHORT, INFORMAL CORRESPONDENCE)

OPEN FORUM FOR MEMBERS

UNIVERSITY OF MINNESOTA
TWIN CITIES

University Computer Center
227 Experimental Engineering Building
Minneapolis, Minnesota 55455

November 25, 1975

Professor Niklaus Wirth
E.T.H./Clausiusstrasse 55
CH-8006 Zurich
SWITZERLAND

Dear Niklaus:

I am writing to you to ask some questions concerning the language Pascal. As you may know we at Minnesota have been corresponding with Urs frequently on the subject of implementation details for the CDC 6000 Pascal compiler. I am pleased to say that this interchange has been quite rewarding for us and hopefully beneficial to the compiler effort at Zurich. Because we included questions in our letters to Urs which concern the language Pascal, he suggested that we write to you, which we are doing now.

I am sure your work now continues to move ahead in the areas of programming language design. It seems to me that when enough improvements or results of research work are made, then they may become manifest in a new language because of the undesirable consequences of changing a language which is heavily being used as a practical tool. You have stated that Pascal "will not be subjected to changes" with the appearance of the User Manual in a letter to me December 19, 1974. I am disillusioned to face changes in the language Pascal and extensions to the CDC implementation.

The reason for the confusion and disillusionment on my part is that I must often explain to users why Pascal must not be changed ("preaching the gospel") and then be undercut by certain changes. There are some extensions to the CDC version which everyone says are worthwhile and which I tell people will be added "eventually" (e.g. a value initialization facility). While we seem to be making progress on the latter, let me enumerate instances of the former:

1. Definite change to the language -- the removal of DISPOSE as a standard procedure in the second edition of the User Manual. Why?

2. We sent you a copy of our letter to Urs about the error trap label and the changes to the READ procedure for integers and reals.

3. And what about other changes (such as extending READ and WRITE to act on files of any type)?

Obviously you have received much pressure from users to make these changes. Do all CDC users have to suffer these changes? Could it be suggested that these are optional extensions for the people who want/need them. George Richmond distributes all of your updates to us in North America as "mandatory". Could a statement be issued to the effect that specific previous changes are optional?

A Pascal User's Group is being formed in North America. The first meeting of a group was informally held at ACM '75 here in October. One very heartening aspect of the meeting was that many people who spoke expressed the same concern: "Will there be a proliferation of non-compatible extensions in different implementations of Pascal?" At stake are portable Pascal programs which are written beyond standard Pascal. The statement was also made that "Pascal is our only viable hope to Deep-Six (bury) Fortran", and we cannot afford to blow our (perhaps last) chance to do that. For if we fail, we become another reason why Fortran should not be challenged.

An important group of people in this regard are the maintainers and implementors of Pascal compilers. They must be both responsible and energetic-hence the need for a User's Group. The Pascal Newsletter has been the only previous vehicle available; and even though George has done a good job, it must be published more regularly, and include articles discussing issues such as extensions in various implementations.

Sincerely,

Andy

ABM/md

EIDGENÖSSISCHE
TECHNISCHE HOCHSCHULE ZÜRICH

Institut für Informatik

Clausiusstrasse 55
CH-8006 Zürich
Ø 01 / 32 62 11

Mr. A. Mickel
University of Minnesota
Computer Center
227 Experimental Engineering Bldg.

Minneapolis, Minnesota 55455

December 10, 1975

Dear Andy,

I thank you very much for your letter and appreciate your concern about the language Pascal. In fact I share it and I sympathise with your attitude against changes. But you really put me on the defensive, and I feel obliged to clarify a few points of possible misunderstanding.

First of all I wish to <u>distinguish clearly between language and implementation</u>. This is a very important distinction; the language is defined by the Report alone, and intentionally leaves many details unspecified that an implementation inherently must define one way or another. Secondly we must <u>distinguish between change and extension</u>. I have repeatedly stated that I wish to refrain from changes of existing features of the language. But obviously I cannot prohibit other people to implement extensions, nor even - alas - to introduce local changes. After all, the compiler is distributed in source form which facilitates the incorporation of changes. We ourselves have currently several students who implement extensions, but I don't know whether they will ever become part of our distributed CDC compiler. Even if they did, this would in no way imply that they become part of Standard Pascal. Actually, I am sure they would not, because we cannot afford changing (extending) all the published manuals and documentations. Hence, some current work on a data initialization facility will at most become a part of the CDC 6000-3.4 system, a facility, however, that does in no way modify any existing feature. My attitude and intent are to be very tight with allowing extensions to go into our distributed and maintained compiler.

You mention three issues in particular:

1. The removal of DISPOSE. The trouble is that it had never really existed. Moreover, it has become increasingly clear that a truly satisfactory realization would be quite difficult and of an unwelcome complexity. It is quite possible to implement a non-checking version which, however, bears the danger of leaving "dangling references" hanging around. From the point of security,

nothing less than an automatic mark-scan garbage collector would suffice. Now, anyone is of course free to implement a DISPOSE of his own liking as an additional predefined (not "standard"!) procedure, but I would consider it a mistake to define DISPOSE as a standard procedure when we had no generally satisfactory solution of implementation.

2. The error trap label recently introduced is clearly an extension to our implementation (6000-3.4), and cannot be considered as a language change.

The change in the READ procedure for integers and real numbers was, in hindsight, a rash decision for which I must apologise. As you know, this change has been revoked because of unfavourable reactions. Hence, only leading blanks are skipped as before.

However, in order to avoid the awful situation that the apparently correct program

```
while not eof(input) do
     begin read(x); write(x)
     end
```

produces unexpected results (such as writing the last number twice, if it is followed by blanks), the procedure "read" is changed such that an error will be indicated, if the end of the file is reached while skipping (leading) blanks. Please note two points:

a) Above program is indeed correct, if x is of type char. The earlier change had been motivated by the desire that it should also be correct for x: integer/real.

b) The above indicated change in the "read" procedure is <u>not</u> a change of the <u>language</u> PASCAL, for the language definition does not say what happens, if read(x) hits the end of the file. Nevertheless, some programs that "ran" may die in the future. They should never have run in the first place!

3. The "change" of letting READ and WRITE be applicable to files of any type is not a change, but an extension, even an obvious and straight-forward extension. It is fully documented in both Manual and Report. What other language changes are you referring to?

I think it is an advantage, if Pascal centers follow our distributed updates, but we certainly cannot force anyone to do so. The next update will be, as I am aware, considerable. But again, it incorporates no language changes, merely improvements of the compiler which ultimately are in the interest of all users of Pascal. The new version will eliminate several restrictions which are sometimes quite bothersome. Again, they are of course optional to those who don't wish to profit from our efforts.

I am very pleased to hear about the formation of a Pascal user group, and of course I share the hope that Pascal may contribute to overcome the Fortran age. But of course there are limits to the support of portability. Every installation will always provide some features that are convenient on its machine, but difficult and often counterproductive if imitated on other computers. I see no use in aiming for transferability of programs that use such local features, i.e. which don't adhere to Standard Pascal.

Here at Zürich we would also welcome a Newsletter appearing more regularly. But at the same time I must confess our inability to run it, which doesn't mean that we shall not be happy to contribute and to support it. Please keep me informed about future developments.

Sincerely yours,

Niklaus Wirth

Niklaus Wirth

cc: G. Richmond

NW:ht

Oct. 1975

Literature about the Programming Language PASCAL

N. Wirth, The programming language Pascal and its design criteria.
Paper presented at Conf. on "Software Engineering Techniques", (NATO Science Committee) Rome, Oct. 1969, and published in "High Level Languages", Infotech State of the Art Report 7, 1972.

- The programming language Pascal, Acta Informatica 1, 35-63 (1971).

- The design of a Pascal compiler,
Software - Practice and Experience 1, 309-333 (1971).

J. Welsh and C. Quinn, A Pascal compiler for ICL 1900 series computers,
Software - Practice and Experience 2, 73-77 (1972).

R. Schild, Implementation of the programming language PASCAL,
Lecture Notes in Economics and Mathematical Systems, 75 (1972).

G. Friesland, et al., A Pascal Compiler bootstrapped on a DEC-System 10,
Lecture Notes in Computer Science 7, 101-113 (Springer-Verlag 1974).

C.A.R. Hoare and N. Wirth, An axiomatic definition of the programming language Pascal, Acta Informatica 2, 335-355 (1973).

N. Wirth, Systematic Programming,
Prentice-Hall, Inc., Englewood Cliffs, N.J. 1973.

K. Jensen and N. Wirth, Pascal - User Manual and Report,
Lecture Notes in Computer Science, Vol. 18 (1974), and Springer Study Edition (1975), both Springer-Verlag.

N. Wirth, Systematisches Programmieren (Taschenbuch)
Teubner-Verlag, Stuttgart, 1973.

P. Brinch Hansen, Operating System Principles,
Prentice-Hall, Inc., Englewood Cliffs, N.J. 1973.

A.N. Habermann, Critical comments on the programming language Pascal,
Acta Informatica 3, 47-57 (1973).

U. Ammann, The method of structured programming applied to the development of a compiler. International Computing Symposium 1973, A. Günther et al., Eds., North-Holland (1974).

O. Lecarme and P. Desjardins, More comments on the programming language Pascal, Acta Informatica 4, 231-243 (1975).

N. Wirth, An assessment of the programming language Pascal,
IEEE Trans. on Software Engineering 1, 2, 192-198 (1975), and SIGPLAN Notices 10, 6, 23-30 (1975).

U. Ammann, Die Entwicklung eines Pascal-Compilers nach der Methode des strukturierten Programmierens. ETH-Diss. 5456 (1975).

P. Brinch Hansen, The purpose of Concurrent Pascal,
SIGPLAN Notices 10, 6, 305-309 (1975).

N. Wirth, Algorithmen und Datenstrukturen,
Teubner-Verlag, Stuttgart (1975).

- Algorithms + Datastructures = Programs,
Prentice-Hall, Inc., Englewood Cliffs, N.J. 1975.

UNIVERSITY OF MINNESOTA
TWIN CITIES

University Computer Center
227 Experimental Engineering Building
Minneapolis, Minnesota 55455

December 29, 1975

Prof. Niklaus Wirth
Institut für Informatik
E.T.H./Clausiusstrasse 55
CH-8006 Zurich
Switzerland

Dear Niklaus,

Thank you very much for your nice letter which arrived Monday, December 15. Also thanks for enclosing the updated bibliography, which is valuable to have. I am grateful that you took the time to explain a lot of things. Shortly after I mailed my last letter (November 25) to you, we received a letter from Urs containing the retraction of most of UPDAT10. We were very happy to see that.

I didn't mean to put you on the defensive when I wrote my letter. I really feel bad about that. My intention was to get some substantive information which you have provided to my satisfaction. You are right: I as a local maintainer (and I am sure others like me) was confusing both the language with the implementation and also changes with extensions. After I got your letter I have to admit that I spent quite a bit of time going over all of our correspondence (including that with Urs) over the past year and noticing how many times I failed to make these distinctions. And when I wrote my letter of November 25, I remember trying to decide (and wondering) under what categories to identify certain changes. Now when I reread that letter I also see that I failed to communicate several things properly. In several cases I misrepresented what I meant to say.

Looking at your letter now: "The language is defined by the Report alone which intentionally leaves many details unspecified that an implementation inherently must define in one way or another." Fine; I understood from having read the original Report (with the long introduction - before it was published in Acta Informatica) and the "Axiomatic Definition" that leaving certain details unspecified is an advantage. Being somewhat partial to semantics over syntax, I would hope that the "Axiomatic Definition" helps to define the language, too.

Now to consider "change." I agree with the notion that Pascal itself represents a major departure from the past (and past mistakes). And I personally do not mind change in and of itself if it is properly motivated.

Of course the environment for changing the language Pascal has itself changed over the last 5 years so that now it is difficult to effect change (more users, changes to existing documentation (as you stated), and portability considerations, to name several). What pops into my mind is that we have several Pascal Reports. The original (1970), the Revised (1972), and then two more (July and December 1973) that were small improvements on the way to the book: Pascal, User Manual and Report (revised, revised Report?) and now the second edition of P, UM & R which contains the change about DISPOSE. I guess if we only go by published (other than technical reports) versions there is only one Revised Report - the one in P, UM & R. I just want to show you that these minor details helped confuse me. So my last letter had only one valid gripe about changing the language Pascal - the elimination of DISPOSE in the second edition and your explanation is entirely satisfactory. But you didn't announce it officially, and I wanted to raise the question because I wasn't sure.

I don't want to put Urs in the middle of this discussion but just to be sure that I tell you as much as I can on the issue of changing the language I want to add that it was Urs who said specifically: "By the way: would you please send letters concerning the language PASCAL to Wirth and not to me" in a letter to us October 29 in response to our criticism of UPDAT10 in a letter to him (and copy to you) October 9. This implied that we had dealt with the language itself and not just extensions to the implementation which I now know (thanks to your letter) to be the case.

Concerning the change in the READ procedure for integers and reals and also looking at our October 9 letter, I quoted your letter of December 9, 1974 about the User Manual describing Standard Pascal and Urs' November 28, 1974 letter saying that the schema in the User Manual for reading from a textfile will be valid no longer if our suggested change for reading were adopted. Two points here: 1) the schema concerned must not be part of the User Manual describing Standard Pascal as this feature involves an implementation-defined definition, and 2) your indication that there will be a change to READ after all turns out to be exactly the same one we suggested and sent in a letter November 7, 1974 which produced the replies from you and Urs (see enclosure). We rejected our own change in the light of those replies.

Continuing with your letter, I understand now that error trap labels are an extension to the CDC implementation. But in spite of what you said about extensions, I have an intuition that they are slightly contrary to the philosophy of Pascal. I apologize for putting down in my letter the reference to the extension of READ and WRITE to handle files of any type, and "other changes" referred fact to other extensions. I was in the mood to question everything at that point. I did not at all dislike the extension to READ and WRITE. I disagree that it is fully documented in the User Manual and Report. Can you give me the specific reference?

I too think it is an advantage if Pascal centers follow your distributed updates. I hope that our deeds as well as talk support this. We have maintained all along that working with you was far more productive than going off on our own (as several installations have done). We in fact (as stated in the letter of October 9) incorporated every update fully except UPDAT10.

Now I see that you aren't necessarily working on a new language at all, but possibly testing and implementing new language features (extensions) in the CDC 6000 implementation. O.K. Before your letter, it had been my mistaken belief that it was important to adhere as closely as possible to Standard Pascal because of the argument that too many additional features leads to writing programs with a greater probability of being non-standard. I am therefore glad that it is your attitude and intent to be very tight with extensions in your distributed version. Now I guess that the issue is whether Standard Pascal is a general enough language to be really practical in terms of portability (whereas ANSI Fortran IV doesn't even come close because it is not general purpose). Features were therefore left out of Standard Pascal which are impractical to implement on all machines. Is this a correct interpretation? What I meant in my last letter about proliferation of non-standard extensions was simply that in the implementations on major machines, if a feature is to be added which has been added in another implementation, and it can be implemented in exactly the same way, then it should be done the same way so as not to be arbitrarily different. For example the CDC 6000 implementation can serve as a model (since it is first on a number of features). A feature such as a data initialization facility if implemented in the CDC 6000 version could be used with identical syntax by another implementation such as the DECsystem 10 version if it is convenient on that machine. By major machines I mean Burroughs 6700, etc., DECsystem10, IBM 360/370, Univac 1100 series, CDC 6000/Cyber 70/170 series, and Honeywell 6000 series.

There are many extensions to the CDC 6000 version and it is important that they be documented fully and in one place. (In this regard the User Manual is out of date.) So at our installation, our local documentation satisfies this need, but I want to help other installations as well. Maybe the Newsletter will help this.

Another subject that I did not convey properly in my last letter was about the User Group and Newsletter. I did not mean to complain, but rather make the case for the need. And I forgot to say that we in Minnesota were going to volunteer! I am glad you like the idea of a Pascal User's Group. In fact it is now certain that we in Minnesota will be producing the Newsletter and doing so more regularly. George and I have talked several times on the phone recently about this; he will do his last Newsletter (#4) very soon. Our goal for the Newsletter will be not only to maintain the quality George has set, but also to include more articles such as language philosophy and news of other implementations. Tentatively we want to put out a 20 page Newsletter four times a year for a $4.00 subscription. We will also photo-reduce contributions directly and not retype them. The User's Group may evolve into something official like STAPL for APL (under SIGPLAN auspices). George has a large mailing list in North America. I'm not so sure how to handle a large number of overseas members because of mailing costs.

As to your contributions and support, they are very welcome. May I suggest that an official statement about your policy on the CDC 6000

implementation maintenance would make a nice article. And would it be possible for you and your people to directly mail us your articles in the future about updates, documentation, etc., instead of us having to wait for the relay from George in Colorado? This will certainly help if we are going to put out the Newsletter.

A week ago I obtained a copy of your new book, A + DS = P, which I can't wait to finish reading! A few questions (maybe I pay too much attention to details) - is it the case that certain notational differences arise between Standard Pascal and Pascal, the publication language? Specifically, the symbols ≠, ≥ , ≤ , ∧ , ∨ , ¬ reappear. Also, the syntax diagram for field list in a record declaration has some errors in it. (See enclosed)

It was most revealing to have read and studied your letter and I want to say "thanks" again. The other glimpse of where we are headed that proved valuable was the article which appeared this summer in SIGPLAN Notices from the proceedings of the Reliable Software Conference. If you have any further reactions to what I have said, I would very much like the privilege of seeing them. Thanks in advance!

Happy New Year to you and your associates at the Institut für Informatik from us at the University Computer Center.

Sincerely,

Andy

ABM/ks

Enclosures

cc: U. Ammann, G. Richmond

LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720 □ TEL. (415) 843-2740

Math&Computing Group
LBL
12 Jan 75

Andy Mickel
U Minn Computer Center
227 Experimental Engineering Bldy
Minneapolis 55455

Dear Andy --

Thanks for your recent letter and enclosures. We (LBL) would certainly like to be on the mailing list of the new Pascal User's Group. Addressee should be me, Ed Fourt, at the above address. If you ~~x~~ ever want to try and reach me via phone (it ain't easy), number is (415) 843-2740 ex 5074.

I will put a notice in our newsletter that anyone who wishes to be ~~m~~ involved with the user's group ~~xxxi~~ should write you. Eric Buchbinder, to whom you co-addressed your letter, is an enthusiastic user but not responsible for the maintainance of the compiler -- I don't know his mailing address, presumably he'll send it to you.

We have Pascal up on our 7600, which is probably unusual. ~~[illegible]~~ Main reason we have it is that we don't use CDC's Scope 2, which lacks CIO, so our mods are probably of no help to others.

Most happy to hear you're re-implementing VALUE. Any time estimates??? -- I mean, when you'll have it??? Pascal is damnably lame without it, sez me. Once it's (back) in, maybe I'll start pushing Pascal harder to the general user community -- usage is probably lower here than is really warrented. But the big drawback to Pascal is, it keeps on playing those dirty tricks on you like abolishing VALUE...

Thanks for writing and hope to hear from you again

Ed Fourt

Ed Fourt

UNIVERSITY OF WASHINGTON
SEATTLE, WASHINGTON 98195

Department of Computer Science, FR-35

January 14, 1976

Mr. Andrew B. Mickel
University Computer Center
227 Experimental Engineering Building
University of Minnesota
Minneapolis, Minnesota 55455

Dear Andy:

Thank you kindly for your letter of January 5, 1976. I hope that you will place me on the mailing list of the newsletter and other mailings. Please send the Pascal newsletter also to:

Computer Information Center
Academic Computer Center
Mail Stop FC-10
University of Washington
Seattle, Washington 98195

and to Mr. Fred Dunham
Academic Computer Center
(address as above).

For reference, my telephone number is (206)-543-9264.

We are using the Pascal2 compiler from Zurich (via Colorado) on our CDC 6400-CYBER 73 system. In addition to the updates from Colorado (we are currently on Update 8), we have made a number of local modifications. The primary one made substantial changes in the scanner, allowing multicharacter representations for a number of symbols (e.g.'.,' for;, PTR for "uparrow",'..=' for :=, etc.). These were done to allow easy use of IBM 026 keypunches. Also, we have changed the compiler to the SCOPE 3.4.4 end-of-line convention, added line numbers to the listing, and a few minor things. Other changes are planned for the future, but will depend on student help. A copy of our Pascal announcement is enclosed.

We would be very interested in learning more about your changes; reduction of compiler size is quite important to us. I am thinking also of breaking the compiler into overlays so that we can run in 40k (octal); have you thought about that? If you send me a tape of your mods, preferably in standard UPDATE format, we will reciprocate (please use 7 track tape, if possible). Could we also have your available literature?

Do you have any experience using the FORTRAN-PASCAL linkage? I happen to believe that the future of PASCAL depends on two things: (a) having a (minimal) standard language that all implementations adhere to -- perhaps as defined in the Revised report, and (b) in providing for all implementations an environment in which it is very easy to mix (sub)-programs written in FORTRAN and PASCAL. Any thoughts would be welcome.

I hope to hear from you soon; thanks again for writing. When is newsletter No. 4 going to appear?

Sincerely yours,

[illegible]

Hellmut Golde
Professor

HG:yi
Enclosure

THE UNIVERSITY OF TEXAS AT AUSTIN
COLLEGE OF NATURAL SCIENCES
AUSTIN, TEXAS 78712

Department of Computer Sciences
Painter Hall 3.28

March 8, 1976

Andy Mickel
University Computer Center
227 Experimental Engineering Building
Minneapolis, Minnesota 55455

Dear Andy,

I am finally sending you the notes I promised you some time ago. In the meantime you should have received the tape containing the BOBSW system.

I modified our PASCAL system somewhat so that the user has knowledge about how much space is used for stack and heap. A control card might look as follows:

```
PASCAL2.OPT = PT, C = 7, W = 4, B = 4.
```

This gives the compiler 7K to work with, the generated program will have at least 4K, and the buffer size for files of type TEXT is 1K. The work space is measured when a procedure is entered, also when an item is placed on the heap.* The code generation for measurement can be turned off. The user is now independent of setting the fieldlength, which may vary with compiler options or routines from libraries. By setting the right workspace an optimal fieldlength is achieved.

I also fixed a bug in the code generation (see example). As this was the first time I really looked at the code generation I have to say I was puzzled! I just hope the new version will do a better job!

About a month ago a PASCAL translator (written in UCI-LISP) was finished. It does all the type checking and it produces a prefix form which is used by a verification system to generate verification conditions. The grammar I used is included.

How is your PASCAL Users' group coming along? Got any new developments from Zurich? Please, keep me informed. Looking forward to your next newsletter, I remain

Wilhelm Burger

Wilhelm Burger

WB/mm

* The actual used workspace is put into the dayfile.
Enclosed: DEC-10 document, big program, grammar, letter to Urs (10/24/75) letter from Urs (10/24/75

RICHARD J. CICHELLI
901 WHITTIER DRIVE
ALLENTOWN, PENNSYLVANIA 18103

March 9, 1976

Mr. Andy Mickel
University of Minnesota
Computing Center
227 Experimental Engineering Bldg.
Minneapolis, Minnesota 55455

Dear Andy:

Glad to hear you liked my paper.

Is the newsletter going only to SIGPLAN members? What I'm thinking is that if not, then you might want an actual Soma cube program to include. I enclose a copy of the Condict program. It's a Xerox copy of the original photo copy and I hope it is a good enough reproduction. The original was sent to (and butchered by) SIGPLAN. It might help clarify things. If you decide to use it, I recommend you use the bottom half of the first page as an introduction.

We are forming a Lehigh chapter of the Pascal Users Group. Joseph Mezzaroba (see attachment for description of some of Joe's work) of the mathematics department is the principal organizer. We expect about 25 active members to start with. You may wish to sign up the entire group to receive the newsletter. Joe can be contacted in care of the Mathematics Department, Lehigh University, Bethlehem, Pa.

Rance DeLong of Moravian College and I gave a talk to our local DEC RSX Users Group on SIL's (Systems Implementation Languages). We are users of PDP 11's. Discussed were: PASCAL (as per Per Brinch Hansen of California Institute of Technology), C (Bell Labs' UNIX) and BLISS (William Wulf of Carnegie Mellon with HYDRA). I understand that a committee at DEC is evaluating SIL's. (Committee members: Rodger Hamm and Leon Spitz at DEC in Maynard, Mass.) They are leaning towards BLISS. I believe this is very unfortunate because BLISS is the most machine dependent of the languages under consideration. I was hoping they would follow the example of CDC and use PASCAL (e.g. the Series 17). I believe that machine dependent SIL's (called MOL's - machine oriented languages) are a step backwards in systems programming. DEC would be better off following the lead of HP, Burroughs, and CDC. Maybe PUG could have some influence on this decision process.

Anyway, the talk was very well received. Most RSX users were surprised to find out that for any given hardware, UNIX supported 10 times the number of interactive users that DEC software did.

Rance is hopeful of getting Brinch Hansen's PASCAL running under UNIX. Any PUG members who have already done this should contact us.

Sincerely,

Rich

Richard J. Cichelli

R. J. Cichelli, J. Goodling, S. L. Gulden, J. Mezzaroba
Lehigh University

A CALCULATION OF THE HAMMING WEIGHTS FOR THE BINARY CODE (73, 28)

The binary code (73,28) is obtained as the row space of the incidence matrix of the projective plane of order 8 where the entries of matrix are taken to be in GF(2). The number of non-zero vectors in this vector space is $2^{28}-1 = 268,435,455$ A program to calculate the Hamming weight, i.e. the number of non-zero components, of each vector was written in the programming language PASCAL-6000 and run on the CDC-6400. The algorithm used was a result of a series of refinements. The first algorithm we designed was estimated to require 20,000 CPU hours; the final algorithm enabled us to perform the calculations in 2 CPU hours. By implementing a multi-precision arithmetic program in PASCAL the weights of the orthogonal code (73,45) were calculated in about 30 seconds using the MacWilliams equations. We comment finally that the structure of PASCAL made the task of program design easier than had been anticipated.

Presented at Conference on Computations in Algebra and Number Theory
August 24-29, 1975
University of New Brunswick

UNIVERSITY OF MINNESOTA
TWIN CITIES

University Computer Center
227 Experimental Engineering Building
Minneapolis, Minnesota 55455

March 15, 1976

George H. Richmond
Computing Center: 3645 Marine Street
University of Colorado
Boulder, CO 80302

Dear George:

Even though it would have been better coming from you, we feel that time is running out this school year and therefore must announce the transition of the PASCAL Newsletter to the Pascal User's Group ourselves. I hope this is okay with you. Just the same, the precise thing is happening that I didn't want: namely some people are already confused about the presense of the User's Group without knowing its relationship to the existing Newsletter.

We have received much mail from prospective members and to this add the people who attended ACM '75. To broaden the base of support to users, teachers, etc., we are doing a mass mailing to the largest university Computer Science Departments and Computer Centers. Enclosed is our cover letter and coupon.

We have sent an announcement to SIGPLAN Notices. Plans now are to issue the newsletter in September, November, February and May, and have membership end on June 30 of the year. So membership is really geared to an academic year and we plan to send all 4 issues of a given year to a person who joins any time during the year. We are now soliciting 76-77 membership.

Now you can see that our announcement in Newsletter #4 needs revision.

Zurich wrote to us that Release 2 would be ready by now but we have no word as to specifics, do you?

Now that #5 is back to September, 1976, I hope that takes the pressure off of you. To have the best impact for the User's Group however, it would really help if #4 appeared no later than May. I know you will do what you can.

Also, please send as soon as possible:

- a copy of your mailing list of any PASCAL users past or present,
- the xerox copies of correspondence you have had, code corrections and suggestions sent you, bug reports, and examples of local documentation,
- any and all information on other implementations.

George H. Richmond
March 15, 1976
Page 2

The last item is most important. Confusion proliferates especially as to the best IBM 360 version and where to get a Univac 1100 version.

If you don't have time to write a reply please call me Friday afternoon 612/376-7290. Thanks a lot George! Hope to see your membership soon.

Sincerely,

Andy

Andy

ABM:bln

Enclosure

UNIVERSITY OF SOUTHERN CALIFORNIA

UNIVERSITY COMPUTING CENTER • 1020 W. JEFFERSON BLVD. LOS ANGELES, CALIFORNIA 90007 (213) 746-2957

April 5, 1976

Andy Mickel
University Computer Center: 227 Exp Engr
University of Minnesota
Minneapolis, MN 55455

Dear Andy:

In your announcement of the formation of a PASCAL User's Group, I came across a memo from you to Tim that states that there are now six IBM implementations of PASCAL. Since Tim no longer works here, I would like some information about these implementations. Would you send me information concerning where these implementations are, if they are available for purchase, whom to contact concerning each one, how they are differentiated, etc. Academic Services currently has a version from the University of Manitoba, but because of its core requirements, it is awkward to use at our installation. We are looking for a one-step monitor that can run in 120K. Do you know of any such implementation?

I don't know how well you knew Tim, but he is currently working for the World Health Organization. If you care to contact him, his address is:

Tim Gill
poste restante
1211 Geneva 27
Switzerland

We are joining the PASCAL User's Group under separate cover. Thank you for any information you can supply me.

Sincerely,

Susan L. Stallard

Susan L. Stallard
Academic Services

4/13

SS:ks

Georgia Institute of Technology SCHOOL OF INFORMATION AND COMPUTER SCIENCE / (404) 894-3152 / ATLANTA, GEORGIA 30332

May 4, 1976

~~Mr~~. Andy Mickel
Pascal User's Group
UCC: 227 Exp. Engr.
University of Minnesota
Minneapolis, Minnesota 55455

Dear Mr. Mickel:

I goofed in my note on the PUG application blank. I still need Newsletter #2 (and #4 when you get it).

Here at Tech, as at many places, there is a distinct separation between the computer center and the computer science department. The center has had the PASCAL compiler on the CYBER for about 15 months now, but I do not think that there was much use of it made. It appears that I am the first member of the department to promote its use. I handle the operating system series and in the graduate course we went through Brinch-Hansen's text so that we could then move on to his SOLO system during the operating system lab course. As you probably know, the SOLO package includes two PASCAL compilers --- one for sequential PASCAL (pretty much the standard) and one for Per's concurrent PASCAL. The whole thing is written for the PDP-11/45. Right now I have three teams trying to bring up his sytem. One of them is doing it on the CYBER. The approach there is to rewrite the kernel and interpreter in CYBER PASCAL, make the necessary interfaces to have the I/O look like the PDP-11, and run the whole thing under NOS. Another group is working on the Burroughs B-5700. They are rewritting the kernal and interpreter in ESPOL (ALGOL), and when they finish the Burroughs will run under SOLO alone, the MCP will not be used at all. The third group is working with our PDP-11 which does not have floating point and has some other I/O differences. All indications thus far are that all three groups have tasks that are comparable.

I also have a group of special project students doing some system programming for the Motorola M6800. I have been pushing them into the use of PASCAL for the initial design of their programs.

A couple of questions: How many updates have been issued for PASCAL 6000-3.4 thus far? Who is distributing them now? Is that going to be Colorado or you in the future? I have heard some talk about a new version of PASCAL for the CYBER (for obvious reasons). Any other news on that?

Sincerely,

Phil

Philip H. Enslow, Jr.
Associate Professor

PHE/ngh

P.S. Did you see the initial announcements of the new CDC line --- CYBER 18? The only high level language I saw mentioned was PASCAL.

HARRY M. MURPHY, JR.
3912 HILTON AVENUE, N.E.
ALBUQUERQUE, NEW MEXICO 87110
TEL: (505) 881-0519

23 May 1976

Pascal User's Group
c/o Andy Mickel
University Computing Center
227 Experimental Engineering Building
University of Minnesota
Minneapolis, Minnesota 55455

Dear Sir:

Please enroll me as a member of the Pascal User's Group as announced in the May 1976 SIGPLAN Notices. My personal check for $4.00 for the current year is enclosed.

My home address and telephone number are given above; my "official" address and telephone number are:

> Air Force Weapons Laboratory
> SAT (Mr. Harry M. Murphy)
> Kirtland AFB, New Mexico 87117
> Tel: (505) 264-9317

We recently acquired the Pascal 6000-3.4 compiler from the University of Colorado and now have it running on one of our CDC-6600 computers. I am just starting to investigate to what extent Pascal may be a serious competitor with FORTRAN for the writing of scientific computer programs. Until now, FORTRAN has been used almost exclusively for such work at AFWL.

I selected as my first non-trivial Pascal program a general-purpose weighted least-squares polynomial fitting program which reads x,y,z triplets from cards (one triplet per card) and which fits second through fifteenth order polynomials to the data weighted according to the values of z_i. Of course, I immediately ran into the difficulty of writing general-purpose procedures - such as one to solve a system of linear equations - whose arguments include arrays whose dimensions are not known until the procedure is called. This is a very serious omission from Pascal, since it implies a serious lack of generality in procedures processing arrays.

I am looking forward to receiving the next issue of the Pascal Newsletter when it appears next September.

Sincerely,

[illegible]

---

structured systems

Los Altos Office
(RRC/Structured Systems)
(415) 948-0877

200 Third Street Los Altos, California 94022

26 May 1976

PASCAL USER'S GROUP
c/o Andy Mickel
University Computer Center: 227 Exp Engr
University of Minnesota
Minneapolis, Minnesota 55455

To Whom It May Concern:

We are currently working on one of the largest software projects undertaken in PASCAL. There are currently seventeen PASCAL programmers coding different parts of the system, and it is expected that twenty-five will be required for the latter phases of the project. We are using a version of the University of Illinois compiler, which runs on and generates code for the DEC PDP-11/45 machines which are being used for both the development and target systems. The compiler has been extensively modified to allow the integration of separately compiled code modules and data bases, to extend pure PASCAL in certain dimensions, and to make use of certain hardware features of the PDP-11/45 and PDP-11/70.

Other members may be interested to learn that we have an immediate need for at least five programmer/analysts with PASCAL experience. Experience with compilers for PASCAL-like languages and with the PDP-11, along with some industrial as well as academic experience, would be relevant. We will be able to pay extraordinary fees (between $2000 and $3000 per month) for a well-qualified wizard. The different positions on this project are expected to last between three and ten months, with the possibility of further structured programming and systems work on other projects, according to the interest of the programmer.

We would be very grateful if the possibilities for participation in this project could be brought to the attention of interested and qualified members of the computer science community, either through the Pascal Newsletter or in any other forum which seems appropriate. Interest applicants should contact David Shaw at (415) 966-2082 or (415) 948-0877 (messages) immediately.

Many thanks for your assistance. We look forward to receiving the Pascal Newsletter, and would be grateful to receive all back issues and any membership and/or installation lists which may be available. Please bill us for any associated cost.

Sincerely,

David E. Shaw

David Elliot Shaw
Division Manager

DES:mbh

AMERICAN REVOLUTION BICENTENNIAL 1776-1976

A Division of International Monitor, Inc.

JET PROPULSION LABORATORY *California Institute of Technology • 4800 Oak Grove Drive, Pasadena, California 91103*

27 May 1976
Refer to: 914.12/2592

Pascal User's Group
c/o Andy Mickel
Univ. Computer Cntr., 227 Exp. Engr.
University of Minn.
Minneapolis, MN 55455

Dear Mr. Mickel:

You will find enclosed my check for $4.00 for membership in the Pascal User's Group.

My own main interests are in numerical analysis and mathematical software. I am a member of the IFIP Working Group 2.5 on Numerical Software and I was editor of the SIGNUM Newsletter from 1972 through 1975. I try to keep informed on programming language developments that may have significance for the development of portable library-type mathematical software.

So far, the Pascal language appears to be unsuitable for this type of application, due to the lack of something equivalent to the adjustable dimension feature of Fortran. For example it does not seem to be possible to write a library procedure in Pascal to operate on an n by n matrix A provided by the calling program.

I would be particularly interested in obtaining information on a Pascal compiler for the Univac 1108. So far we do not have a Pascal compiler on our JPL Univac 1108 systems.

I wish you and your colleagues good luck in launching the Pascal Newsletter.

Sincerely yours,

Dr. Charles L. Lawson, Supervisor
Applied Mathematics Group
MS ~~FH-D/308~~ 125/128

CLL;lh

Telephone 354-4321 *Telex 675421* *Twx 910-588-3294*

UNIVERSITY OF COLORADO
BOULDER, COLORADO 80302

COMPUTING CENTER

18 June 1976

Mr. Andrew B. Mickel
University of Minnesota
University Computer Center
227 Experimental Engineering Bldg.
Minneapolis, Minnesota 55455

Dear Andy:

I have finally managed to assemble the material I promised you several weeks ago. The delay has been mine. Enclosed you will find a multitude of items including the rough draft of the fourth PASCAL Newsletter. Let me outline them briefly.

1. Various documents written by Wirth's group: "On Code Generation in a PASCAL Compiler", "PASCAL-S: A Subset and its Implementation", "The PASCAL(P) Compiler: Implementation Notes", and "An Axiomatic Definition of the Programming Language PASCAL".
2. Rough draft of Newsletter 4. Please send me your comments as soon as you can.
3. Various items associated with PASCAL3. distribution - note on tape contents
4. Various items associated with PASCAL-P. distribution
5. Copies of Newsletters 1, 2, and 3.
6. A computer printout of the full PASCAL mailing list.

When I get a chance, I will make copies of all of our previous correspondence from Switzerland.

Hope to hear from you soon.

Sincerely,

George H. Richmond

George H. Richmond

GHR:ejh

Enclosures

UNIVERSITY OF NORTH CAROLINA AT CHAPEL HILL

Department of Computer Science

18 June 1976

Dear Andy,

Thanx for the information on the new PASCAL-P compilers. My own work is with compiler-writing systems, and I planned on using the P-code compiler as a base for (extensive) modifications. Doing so saves the hassle of me writing scanners, parsers, and other irrelevant (but necessary!) subroutines.

My second goal, of course, was to install a usable PASCAL compiler here at UNC. In fact, just last week I installed the Stony Brook compiler on our 360/165 -- should anyone inquire, it seems to be buggy. I've already sent off several bug reports; I have a few more sitting on my desk. Because of that, I doubt that I'll order the P4 compiler; I've spent too much of my department's money on PASCAL compilers, and I don't think they'd be too happy about yet another. I have completed (I think) a set ofmodifications to the P2 version that should let me do what I want, though it is apparent that I will not be producing a production-quality compiler. At any rate, I'll wait at least until George Richmond has it available before making any decisions.

Incidentally, I suspect that the PUG will (eventually) be deluged with requests/comments/bugs in the SUNY compiler. They are no longer running student jobs on a 370, and I suspect thatthey'll discontinue maintenance in about a year.

I'm glad to hear that PUG is catching on. If I find time and a topic, maybe I'll send you an article. In the meantime, my main efforts (other than finishing my dissertation) are devoted to convincing people here to use PASCAL.

[illegible]

--Steve

(*Steve Bellovin*)

[illegible] Hall, Chapel Hill, North Carolina 27514 Telephone 919-933-

TEXAS DEPARTMENT OF MENTAL HEALTH AND MENTAL RETARDATION

TEXAS RESEARCH INSTITUTE OF MENTAL SCIENCES
1300 Moursund, Texas Medical Center, Houston, Texas 77030 713 797-1976

JOSEPH C. SCHOOLAR, Ph.D., M.D.
DIRECTOR

Kenneth D. Gaver, M.D.
Commissioner

July 1, 1976

Pascal User's Group
c/o Andy Mickel
University Computer Center: 227 Exp. Engr.
University of Minnesota
Minneapolis, Minnesota 55455

Dear Andy:

I'm happy to know that you're forming a Pascal User's Group, and am enclosing my membership dues. I don't currently have access to a Pascal Compiler, but hope that you can give me information on how to get one. I'm writing to George Richmond for a Pascal-P2 order form, but would be interested in leads on existing implementations for the following machines:

CPU	Operating System	Main Memory (words)	Disk Capacity (Bytes)
PDP 11/55	RT-11	32K Core	5 Meg. Disc.
HP 2100	RTE	32K Core	4.8M Disc.
CDC 7316	Nos. 1.0	131 K	Plenty

I'm most interested in getting Pascal for the PDP 11/55. This system will be used for signal analysis of biological signals such as EEG and evoked potentials. I hope to use Pascal for applications and system level programming instead of Fortran.

The rapid proliferation of Pascal without support from a major manufacturer is the most encouraging note I've seen in several years. I hope that the User's Group can keep the mementum up while providing an arena for discussion of problems, improvements, standardization, portability, etc. I hope to hear from you soon and am looking forward to the Newsletter in September. Thank you for your help.

Sincerely,

James A. Kendall
Psychophysiology Section

JAK:jj

An Equal Opportunity/Affirmative Action Employer

The University of Calgary
2920 24 AVE. N.W.
CALGARY, CANADA
T2N 1N4

DEPARTMENT OF COMPUTER SCIENCE
TELEPHONE: (403) 284-6316

July 22, 1976.

Andy Mickel, P.U.G.,
UCC: 227 Exp. Engr.,
University of Minnesota,
MINNEAPOLIS, MN 55455,
U.S.A.

Dear Andy,

At the University of Calgary, Pascal is mainly a teaching language. I've been working with it heavily in my research on several fronts, mainly as the vehicle for a program to solve Go problems.

Some students here have taken the Pascal 1 compiler as a base and have developed a very interesting language of their own in the attempt to firm up a number of Pascal's rather obvious clay feet. Anyone interested in this kind of thing should direct an enquiry to Mr. James Gosling; I won't steal their thunder here.

Like most of us, I suppose, I am not a "professional Pascal programmer" (what a droll concept!) but just a person who has found Pascal an excellent vehicle for my thoughts; of all the programming languages I have available, it is the "least bad". But it isn't the best.

When talking about this lovely language with skeptical physicists and the like, the rock I always founder on is that Pascal is incompatible with Fortran. One can't write Pascal subroutines to be called by Fortran, one can't use named common, and one must use a run-time-stack. Hence, potential Pascal users have to convert everything or forget it.

They forget it, and this seems sad to me. If a Pascal implementation could go into direct competition with Fortran, I believe the world would benefit. And I see nothing in the Pascal language to forbid the development of such an implementation, given a convention for named common ("own", if you prefer), a "nonrecursive" attribute for procedures and functions and something to allow the passing of arrays when the subprogram must be given the dope vector at run time.

Surely all this is obvious, and I'm just writing it to ask: Who out there has written a Fortran-subverting Pascal compiler?

Very truly yours,

[illegible]

Stephen Soule.

ABT ASSOCIATES INC.
55 WHEELER STREET, CAMBRIDGE, MASSACHUSETTS 02138
TELEPHONE • AREA 617-492-7100

23 July 1976

Andy Mickel
PASCAL Users' Group
UCC: 227 Exp. Engr.
University of Minnesota
Minneapolis, MN 55455

Dear Andy:

Thank you very much for the mods you sent me in the mail, and the nice phone call. However, I feel it my duty to take you to task over a small issue, the form of your modsets. Although MODIFY is in many ways superior to UPDATE as a library maintenance tool, PASCAL is distributed in UPDATE format, and in fact mods in the past have always been sent out in UPDATE format. If we are to be serious in our attempts to bring about the revolution in computing we support, we must take extra efforts in the direction of standardization.

The reason for the "heat" is that, although several, if not all, of the modifications you sent will be useful to us, the numbering scheme (MODIFY's) is completely incompatible with the form in which we maintain our compiler (UPDATE), and cannot be installed without a great expenditure of time and effort.

I would therefore like to propose a "standard for interchange" of modifications and extensions to PASCAL-6000:

- All modsets would be in UPDATE compatible form.
- All modsets would be based upon Release 2 and would state all dependencies upon other modsets.
- All modsets would contain, as a minimum, the name and institution of the author, the date of the mod, and a brief explanation of the extension/correction.

I would also like to suggest the formation of a "standards committee" which would review proposed extensions and modifications for compatibility with the spirit and letter of PASCAL. Although this may seem unnecessarily formal, without some standard (and regulated growth), PASCAL will become another BASIC, a hodge-podge with so much variation that it is almost guaranteed that one man's

BASIC program will not run (or even compile) on another man's machine. Should this happen, the possibility that PASCAL will replace or compete with FORTRAN will remain only a dream.

I would like to recommend that Nick Wirth be involved in this effort, and every attempt be made to solicit ACM involvement and sponsorship of the standard. With the rapid growth of PASCAL, the time to strike is now. Waiting for an indigenous group of PASCAL users or manufacturers to sponsor such an attempt might prove to be disappointing.

To further the development of the interchange standard, I have enclosed a small tape on which I wish you would write the modsets you presently have (the ones you sent me) in UPDATE compatible form. I would be very grateful if you would then return the tape to me for inclusion in our compiler.

Sincerely yours,

[illegible]

Michael Patrick Hagerty
Director of Systems Research
and Design

enc.

P.S. Add to the three points earlier that mods to the Zurich compiler should be made through one centralized distribution point, so that the "willy-nilly" exchange will not become a stumbling block to development of interesting ideas and features.

I would be most appreciative if you would write your "PASCAL at the U of M" on the tape as a second file. Thanks...

E X T R A P A G E :

This note, written a day after the previous two pages is to serve as an apology for the harshness of the earlier text. I do hope that those comments will not be interpreted as a personal assault, but as constructive and supportive of your present efforts. Sort of like a call to "gird up your loins."

In the meantime I have had another opportunity to go over the listings you sent, and would like to suggest that the formatted read routines for real and integer be included in a manner similar to the formatted write routines:

```
READ (f, I:w, R:w:d)
```

This should prove useful in the business community, and allow them to move away from PICTUREs in COBOL.

I also had the chance to look through Urs' masterpiece, and noticed that the VAR declarations are stored, as you said, in push-down order (LIFO). I thought you mentioned something about a mod to reverse this order, and after looking though the modsets you sent, I did not find one. If you have a modset which allocates the items left to right, I would appreciate receiving a copy of that set as well.

Assuming that you will find time to produce the document, "U of M PASCAL User's Guide," I would be overjoyed to receive a copy when it is completed.

Thanks again for the material and the call. Looking forward to a long and close relationship, I remain

Sincerely yours,

[illegible]

P.S. Good luck on getting the mods that were to become Release 3 out of Nick and Urs. Those mods, with the ~~constant~~ variable and structure initialization from your shop, will prove to be very powerful indeed.

UNIVERSITY OF COLORADO
BOULDER, COLORADO 80302

COMPUTING CENTER

23 July 1976

Dr. Niklaus Wirth
Eidgenossische Technische Hochschule
Institut fur Informatik
Clausiusstrasse 55
CH-8006 Zurich, Switzerland

Dear Dr. Wirth:

I appreciate your concern expressed in your letter of 17 June 1976, so I will clarify my position (and that of the Computing Center) with regard to PASCAL.

Recently, I have been working long hours on other projects. That situation has eased somewhat. Also, interest in PASCAL has grown over the years so that some of the responsibilities that I initially carried have been shifted and others will follow. The result has been slow response to letters and orders. That situation will improve as things get better organized here.

The first change was to involve our group's able secretary, Jan Hurst, in several aspects of PASCAL. For the last year, she has been processing orders for PASCAL2 and now PASCAL3.

The second step was the PASCAL Newsletter. One of the problems was the irregular intervals at which it was published. Andy Mickel has now assumed that duty and promises to put the newsletter on a regular basis. As material for a newsletter has been accumulating for over a year, I agreed to publish one more newsletter. Urs Ammann recently received a rough draft and I hope to have the fourth newsletter in the mail to you about 1 August and back from the printers in bulk by mid-August.

The third change was to select a distributor for Australia (and surrounding regions) that would serve the same purpose as I do in the U.S.A. Overseas mailings have always been a problem in shipment time, reliability of delivery, postage costs, and completion of payments. Carroll Morgan has assumed this position now.

That leaves Jan and me with the filling of orders and mailing of corrections for PASCAL (CDC and portable versions) in the United States. This task is manageable and we will continue to do it. Once the fourth newsletter is out, the remaining problem will be cleaned out.

Dr. Niklaus Wirth -2- 23 July 1976

In the long run, it may be possible for the PASCAL User's Group to become a self-supporting organization like VIM (for CDC users) and SHARE (for IBM users). At that point, we should consider turning over distribution of PASCAL (for all machines) to them.

I hope this letter will clarify my position and put your mind at ease somewhat. I am most enthusiastic about PASCAL and hope to see it grow with each passing year.

Sincerely,

George H. Richmond

George H. Richmond

GHR:ejh
cc: A. Mickel

The IMPLEMENTATION NOTES section of the newsletter is organized as follows:

1) A checklist for more information about distributed implementations of of Standard Pascal.
2) Information concerning Pascal-P, a "portable" compiler of Pascal for a hypothetical "stack machine". It comes on tape as a kit and is used to produce compilers for real computer systems.
3) Other portable compilers: Pascal Trunk compiler, PASCAL-J, and Pascal-S.
4) Information about compilers for real computers sorted by computer system.

(*Note: We simply don't have enough implementation/distribution information. People especially need to know those things that will make them intelligent "consumers" of Standard Pascal systems (see POLICY section inside front cover). One need not adhere to a rigid format when sending this information for inclusion in the newsletter. However it would probably be a great thing to follow the checklist below, and if you desire, supply a short order form (both "camera-ready"). Users of particular implementations are encouraged to share their experiences by sending qualitative and quantitative descriptions. Attach either of these (distribution info or experiences) to an ALL PURPOSE COUPON. Please realize that individual requests to me for implementation information outside the context of the newsletter will be a great hassle.

I must apologize for the incomplete nature of the information following. It will take at least until Newsletter #6 to get fully organized. By far the most requests that have come to me have been for DECsystem 10, PDP-11, and IBM 360/370 Pascal compilers. This issue of the newsletter will concentrate on those and a few others. -Andy*)

CHECKLIST

1. Names and addresses and phone numbers of implementors, maintainers, distributors
2. machine(s) (manufacturer, model/series)
3. operating system(s), minimal hardware configuration, etc.
4. method of distribution (cost, magnetic tape formats, etc.)
5. documentation available (machine retrievable?, in form of supplement to the the book Pascal User Manual and Report?)
6. maintenance policy (for how long? future development plans? accept bug reports?)
7. fully implements Standard Pascal? (why not? what's different?)
8. compiler or interpreter? (written in what language? length in source lines, compiler or interpreter size in words or bytes (specify base) of ___bits, compilation speed in characters/second, compilation speed compared to other language processors (e.g. FORTRAN), execution speed compared to other language processors (e.g. FORTRAN))
9. reliability of compiler or interpreter (poor, moderate, good, excellent)
10. method of compiler or interpreter development (from Pascal-P, hand coded from scratch, bootstrapped, cross-compiled, etc; effort to implement ___ man months, experience of implementors)

EIDGENÖSSISCHE
TECHNISCHE HOCHSCHULE ZÜRICH

Institut für Informatik

Clausiusstrasse 55
CH-8006 Zürich
✆ 01 / 32 62 11

PASCAL-P

[illegible] 197[illegible]

- 1 -

A new release of the PASCAL-P system.

Terminology:

Pascal P1:	either of the early Pascal P systems (released in March and July 1973 respectively).
Pascal P2:	the Pascal P system released in May 74.
Pascal P3:	the new Pascal P system with the same hypotnetical machine as the one underlying the Pascal P2 system.
Pascal P4:	the new Pascal P system with a slightly modified hypothetical machine (allowing a more efficent implementation).

Pascal P3

The compiler is improved in many details. It does, however, still generate code for the old P2 assembler interpreter. The characteristics of the P3 system are:

- ´Full´ compatibility with the P2 system.
- The two records of the assembly code produced by the compiler are terminated by the symbol ´Q´ instead of two ´end of line´.
- All known bugs are corrected.
- Character set independence.
- Runtime tests are included (indexing, assignement to subrange variables, and case selection are checked for legality.)
- The standard functions ´succ´ and ´pred´ are implemented.
- The usual default conventions ´readln = readln(input)´ etc. hold.

Pascal P4

The compiler generates code for a modified assembler-interpreter. The characteristics of the P4 system are:

- It contains all the improvements of the Pascal P3 system.
- An enlarged set of instructions is used. All instructions now handle exactly one type of expression (or have a type indicator). This allows to eliminate book-keeping of type information at runtime and of tag fields in the stack. No implicit type conversion takes place any more. Instead, explicit type conversion instructions are generated by the compiler.
- The compiler respects possible alignment conditions for the allocation of data.
- A runtime test on pointer values is provided.
- A test on runtime stack overflow is generated by the compiler at procedure entry.

Explanations of the installation parameters

intsize,realsize,charsize,boolsize,setsize,ptrsize:
Number of addressable storage units to be reserved for variables of type integer, real, character, boolean, set, pointer. As to ´setsize´, remember that a set must be able to hold at least 48 elements if you intend to use the system to bootstrap the compiler.

intal,realal,charal,boolal,setal,ptral:
Variables of the corresponding types will be given an address which is a multiple of these alignment constants.

stackelsize: Minimum size for a value on the expression stack. The expression stack is that portion of the stack which is used for the evaluation of expressions. ´Stackelsize´ has to be equal to or a multiple of ´stackal´.

stackal: Alignment constant for a valu on the expression stack. ´Stackal´ must be a multiple of all other alignment constants and must be less or equal to ´stackelsize´.

strglgth: Maximum length of a string. (in fact all strings will be of length ´strglgth´). A string must be able to hold the character representation of a number (real or integer) with its sign. The minimum length for a bootstrap is 12.

intbits: Number of bits used for representing an integer without the sign. So the largest integer is

$$2^{\text{intbits}} - 1$$

sethigh,setlow: Maximum and minimum ordinal values for the element of a set.

ordmaxchar,ordminchar: Maximum and minimum ordinal values of the character set.

Depending on the alignment conditions there may be two possibilities for the assignment of store on top of the expression stack.
- Each stack element requires the same amount of store: In this case ´stackelsize´ has to be greater than or equal to the maximum of the other size constants.(Remember: ´stackelsize´ is a multiple of ´stackal´)
- No waste of store: A new element on the expression stack has to be placed at the next position allowed by the alignment constant ´stackal´. In this case ´stackelsize´ has to be less than or eqaul to the maximum of the other size constants.

Format of the Tape

No. of tracks	:	7
Density	:	800 bpi
parity	:	odd
Physical record length	:	5120 frames the last physical record of a file may be shorter than 5120 frames.

Code: second octal digit / first octal digit	0	1	2	3	4	5	6	7
0		A	B	C	D	E	F	G
1	H	I	J	K	L	M	N	O
2	P	Q	R	S	T	U	V	W
3	X	Y	Z	0	1	2	3	4
4	5	6	7	8	9	+	-	*
5	/	(	)	$	=	␣	,	.
6	´	[	]	:				
7	↑		<	>				;

The end of line is represented as a series of two to eleven 00 frames.

The last eight frames of a file have no meaning (the last 8 frames of the trailing short record of a file).

Interrecord gap : 3/4"
End of file gap : 6"
End of information = 2 end of file gaps

The handling charges include the costs for generating the binary versions of the compiler, a minitape, and postage.

with kind regards

[illegible]

Ch. Jacobi

IMPLEMENTATION NOTES

(SOURCE INFORMATION, PROPOSALS FOR EXTENSIONS TO STANDARD PASCAL, BUG REPORTS, PROGRAM WRITING TOOLS, ETC.)

Order form for the revised Pascal P system.

Please provide us with your revised Pascal P system according to the specifications on next page.

Address for delivery of the system

The characteristics of our installation are

Machine type	
Operating system	

Installation parameters (to be filled for case 'A' and 'B' below)

intsize		intal	
realsize		realal	
charsize		charal	
boolsize		boolal	
ptrsize		ptral	
setsize		setal	
stackelsize		stackal	
strglgth			
intbits			
sethigh		setlow	
ordmaxchar		ordminchar	

We order

A ☐ (For new users of Pascal P or users of the Pascal P1 system)
- Pascal P4 compiler(in Pascal).
- Pascal P4 compiler (in P4 code).
- An assembler interpreter of P4 code (in Pascal, for documentation purposes, all alignement and size constants are set to 1).
- Pascal P compiler implementation notes with update list.

Charge SFr 160.-

B ☐ (For users of the Pascal P2 system)
- All of the above package, plus:
- Pascal P3 compiler (in Pascal).
- Pascal P3 compiler (in P3 code, = P2 code).
- Pascal P4 compiler (in P3 code). (1/2 bootstrap, for compilation on the P2 machine of test programs for the P4 interpreter.)
- A file containing the changes made with line numbers.

Charge SFr 290.-

C ☐ (For users who have access to a CDC 6000 Computer and want to experiment with the compiler)
- Pascal P4 compiler with some changes, so that it is accepted by the Pascal 6000 compiler (in Pascal). (All installation parameters set to a standard value.)
- An assembler interpreter (in Pascal, as in package 'A').
- Pascal P compiler implementation notes with update list.

Charge SFr 80.-

(*Note: prices are most certainly different in the Americas and Australia - more next newsletter.*)

Signature :

If you are in Europe, Asia, or Africa, order Pascal-P from:

Ch. Jacobi
Institut für Informatik
E.T.H.-Zentrum
CH-8092 Zürich
Switzerland (phone: 01/ 32 62 11)

In North or South America:

George H. Richmond
Computing Center: 3645 Marine St.
University of Colorado
Boulder, CO 80309 USA (phone: (303) 492-8131)

(note change in address and phone)

In Australia:

Carroll Morgan
Basser Department of Computer Science
University of Sydney
Sydney, N.S.W. 2006 Australia

PASCAL-P Questionnaire

Please fill out and return at your earliest convenience.

Your address: .... Our address: Institut für Informatik
.... PASCAL-P Questionnaire
.... ETH-Zentrum
.... CH-8092 Zürich

0. Do you want to be kept on our PASCAL-P mailing list? ☐ yes ☐ no
1. Will you order or have you ordered the June 1976 version of our PASCAL-P system? ☐ yes ☐ no
2. Did you receive an earlier version of our PASCAL-P system? ☐ yes ☐ no(exit)
3. Which version? ☐ March 1973 ☐ July 1973 ☐ May 1974
4. This PASCAL-P system
   ☐ was never installed. Reason: ....(exit)
   ☐ was installed and was operating, but isn't used any more. Reason: ....(exit)
   ☐ is operating.
5. ☐ It is interpretive. The interpreter is written in ....(language).
   ☐ It was bootstrapped. Method: ....
6. The total effort to bring this PASCAL system to the present state was ....man-months.
7. Comparison of compilation requirements:

programming language	space: number of kilo-words needed by compiler	speed: number of character processed per central processor second
PASCAL		
FORTRAN		
....		

The maximum number of central memory words available for a single job is ....K words (1 kilo-word = 1 K words = 1000_{10} words).
A central memory word has ......bits.

8. Comparison of execution requirements:

programming language	kilo-words needed by runtime support	compactness of code (>1⇔compacter than FTN)	speed (>1⇔quicker than FORTRAN)
PASCAL			
FORTRAN		1	1
....			

9. The actual length of the PASCAL compiler is ....source lines.
   The number of replaced or inserted lines is ....

10. The reliability of the compiler is

    ☐ poor ☐ moderate ☐ good ☐ excellent

11. The use of PASCAL is ☐ batch ☐ interactive.

12. The PASCAL system is used for

purpose	usage			
	none	little	moderate	extensive
experimenting				
student courses				
production				
....				

13. Tendency of usage: ☐ increasing ☐ stable ☐ decreasing

14. Standard PASCAL constructs which are not available:

    ☐ program heading ☐ several standard procedures/functions
    ☐ formal procedures/functions ☐ sets of the form [<exp>..<exp>]
    ☐ files ☐ ....

15. Local extensions of PASCAL are ....
    ....

16. The PASCAL system is running
    on a ....(machine type)
    under ....(operating system)

17. The main problem with the PASCAL system is ....
    ....

18. Do you distribute your PASCAL system? ☐ yes ☐ no(exit)

19. The PASCAL system was sent to ....places.

exit: Comments:...
    ....
    ....

Thank you for your collaboration!

(*Note: This is old, Pascal-P2 ordering information from George Richmond for persons in the Americas. It is included here to provide further details which are certainly similar for Pascal-P3 and Pascal-P4.*)

The PASCAL-P Distribution Tape

The PASCAL-P distribution tape may contain either an unconfigured or configured version of the PASCAL-P compiler. The tape may be written as a CDC binary format tape (800 bpi, 7 track, NRZI, odd parity, 5120 character records), as a blocked BCD format tape (800 bpi, 7 track, NRZI, even parity, 1600 character records), or as a nine track tape (1600 bpi, 9 track, PE, 1600 character records). Your tape is:

___ Unconfigured
___ Configured
___ CDC format
___ BCD format
___ Nine-track format

Unconfigured Compiler

The unconfigured tape contains three files of information followed by the same three files again as a duplicate in case of a tape error. The contents of the three files are:

File 1	Interpreter
File 2	Unconfigured Compiler
File 3	Editor

The interpreter is a non-operational Pascal program which documents how to interpret P-code. The interpreter reads P-code from a file and stores it in memory. Then it interprets the P-code for execution.

The unconfigured compiler is written in Pascal and generates P-code for output instead of machine code. This compiler is unconfigured because of the double dollar signs ($$) placed in the text for replacement by the editor. To produce a compiler system instead of an interpreter system, the code generators must be rewritten for the target machine.

The editor reads the unconfigured compiler and configuration parameters and produces two versions of the compiler. The first version is acceptable to the standard Pascal compiler for CDC machines and the second one is acceptable to the PASCAL-P compiler. The first one is compiled and run on a CDC machine. The second one is accepted as input to the first PASCAL-P compiler and P-code is generated for the target machine.

Configured Compiler

The configured tape contains three files of information followed by the same three again as a duplicate in case of a tape error. The contents of the three files are:

File 1	Interpreter
File 2	Configured Compiler
File 3	P-code Compiler

The interpreter is the same one that is on the unconfigured tape.

The configured compiler is the second version output by the editor as described above. It has the configuration parameters supplied with the order for PASCAL-P inserted into the unconfigured compiler.

The P-code compiler is written in P-code. It is the output of the PASCAL-P compiler run mentioned above. It is equivalent to the result of running the configured compiler in file 2 against itself. Once a P-code interpreter is constructed, it should be possible to compile the compiler and produce the same P-code as in file 3.

Character Sets

The attached character set table indicates the correspondence between Pascal graphics and code combinations on tape. At the left, ETH PASCAL-P is the graphics used in this system. Column A is used for CDC format tapes. Column B is used for blocked BCD format tapes. And column C is used for nine-track format tapes.

Note

For BCD and nine-track format tapes, the record size is 80 characters blocked 20 for a block size of 1600 characters. It is unfortunate that some of the lines of the unconfigured and configured compiler source are longer that 80 characters. These long lines will appear as two consecutive 80 character records. Depending upon the text manipulation facilities available at your installation, you may rebuild the long lines or split the source statement at a more appropriate point.

Control Data Corporation Graphics							Display Code (Octal)	External BCD (Octal)	Nine-Track Code (Hex)	IBM Graphics (PN Train)	Recommended Character Conversions		Recommended Display Code Conversion
CDC 63	CDC 64	CDC ASCII	TTY	CDC 713	ETH PASCAL	ETH PASCAL-P	A	B	C		Code (Hex)	Graphics (PN Train)	
①	:	:	②	②	①	①	00	②	②	②	②	②	②
A	A	A	A	A	A	A	01	61	C1	A	C1	A	
B	B	B	B	B	B	B	02	62	C2	B	C2	B	
C	C	C	C	C	C	C	03	63	C3	C	C3	C	
D	D	D	D	D	D	D	04	64	C4	D	C4	D	
E	E	E	E	E	E	E	05	65	C5	E	C5	E	
F	F	F	F	F	F	F	06	66	C6	F	C6	F	
G	G	G	G	G	G	G	07	67	C7	G	C7	G	
H	H	H	H	H	H	H	10	70	C8	H	C8	H	
I	I	I	I	I	I	I	11	71	C9	I	C9	I	
J	J	J	J	J	J	J	12	41	D1	J	D1	J	
K	K	K	K	K	K	K	13	42	D2	K	D2	K	
L	L	L	L	L	L	L	14	43	D3	L	D3	L	
M	M	M	M	M	M	M	15	44	D4	M	D4	M	
N	N	N	N	N	N	N	16	45	D5	N	D5	N	
Ø	Ø	Ø	Ø	Ø	Ø	Ø	17	46	D6	Ø	D6	Ø	
P	P	P	P	P	P	P	20	47	D7	P	D7	P	
Q	Q	Q	Q	Q	Q	Q	21	50	D8	Q	D8	Q	
R	R	R	R	R	R	R	22	51	D9	R	D9	R	
S	S	S	S	S	S	S	23	22	E2	S	E2	S	
T	T	T	T	T	T	T	24	23	E3	T	E3	T	
U	U	U	U	U	U	U	25	24	E4	U	E4	U	
V	V	V	V	V	V	V	26	25	E5	V	E5	V	
W	W	W	W	W	W	W	27	26	E6	W	E6	W	
X	X	X	X	X	X	X	30	27	E7	X	E7	X	
Y	Y	Y	Y	Y	Y	Y	31	30	E8	Y	E8	Y	
Z	Z	Z	Z	Z	Z	Z	32	31	E9	Z	E9	Z	
0	0	0	0	0	0	0	33	12	F0	0	F0	0	
1	1	1	1	1	1	1	34	01	F1	1	F1	1	
2	2	2	2	2	2	2	35	02	F2	2	F2	2	
3	3	3	3	3	3	3	36	03	F3	3	F3	3	
4	4	4	4	4	4	4	37	04	F4	4	F4	4	
5	5	5	5	5	5	5	40	05	F5	5	F5	5	
6	6	6	6	6	6	6	41	06	F6	6	F6	6	
7	7	7	7	7	7	7	42	07	F7	7	F7	7	
8	8	8	8	8	8	8	43	10	F8	8	F8	8	
9	9	9	9	9	9	9	44	11	F9	9	F9	9	
+	+	+	+	+	+	+	45	60	50	&	4E	+	76
-	-	-	-	-	-	-	46	40	60	-	60	-	
*	*	*	*	*	*	*	47	54	5C	*	5C	*	
/	/	/	/	/	/	/	50	21	61	/	61	/	
(	(	(	(	(	(	(	51	34	6C	%	4D	(	75
)	)	)	)	)	)	)	52	74	4C	<	5D	)	70
$	$	$	$	$	$	$	53	53	5B	$	5B	$	
=	=	=	=	=	=	=	54	13	7B	#	7E	=	63
							55	20	40		40		
,	,	,	,	,	,	,	56	33	6B	,	6B	,	
.	.	.	.	.	.	.	57	73	4B	.	4B	.	
≡	≡	#	"	"	'	'	60	36	6E	>	7F	"	61
[	[	[	[	[	[	[	61	17	7F	"	6C	%	51
]	]	]	]	]	]	]	62	32	E0	⑥	4A	¢	⑨
:	%	%	:	:	:	:	63	16	7E	=	7A	:	⑨
≠	≠	"	'	'	≠	②	64	14	7C	⑧	7D	'	74
→	→	_	&	&	{	②	65	35	6D	_	6D	_	
∨	∨	!	③	③	∨	②	66	52	D0	⑦	4F	|	77
∧	∧	&	④	④	∧	②	67	37	6F	?	50	&	45
↑	↑	'	↑	∧	↑	↑	70	55	5D	)	7C	@	64
↓	↓	?	#	#	}	②	71	56	5E	;	7B	#	54
<	<	<	<	<	<	<	72	72	C0	⑧	4C	<	52
>	>	>	>	>	>	>	73	57	5F	¬	6E	>	60
≤	≤	@	⑤	⑤	≤	②	74	15	7D	'	5A	!	⑨
≥	≥	\	?	?	≥	②	75	75	4D	(	6F	?	67
¬	¬	^	⑤	⑤	¬	②	76	76	4E	+	5F	¬	73
;	;	;	;	;	;	;	77	77	4F	|	5E	;	71

① End of line
② Not used
③ Carriage return
④ Line feed
⑤ Escape to extended Display Code

⑥ 0-2-8
⑦ 11-0 or 11-2-8
⑧ 12-0 or 12-2-8
⑨ Conversion lost

PASCAL Trunk Compiler

In 1975, H. H. Nägeli developed a "trunk" compiler to help transport Pascal compilers to other machines. The trunk is a source program of a compiler written in Pascal, in which machine dependent parts are marked and clearly separated from machine independent parts, and detailed comments are provided for an implementor how to describe algorithms for these machine dependent parts. For example, Teruo Hikita of the University of Tokyo used Pascal-P to interpret the trunk compiler modified for the IBM 360 compatible Hitachi Hitac 8000 series, with very good results.

H. H. Nägeli is at the Institut für Informatik, E.T.H., Zürich.

PASCALJ

The Software Engineering Group at the University of Colorado Department of Electrical Engineering has implemented a Pascal compiler which generates JANUS intermediate code. The "mobile programming system" JANUS is totally portable - even to the point of defining its own character set. It is available on several computers such as the CDC6400 and the Xerox Sigma 3. There have been several releases of PASCALJ: September, 1975, February, 1976, and one for this month, September, 1976. Write B.W. Ravenel or C.B. Mason for distribution information at:

Software Engineering Group, Department of Electical Engineering, University of Colorado, Boulder, CO 80309

PASCAL-S

As documented in the report: "PASCAL-S: A Subset and its Implementation", June, 1975 by Niklaus Wirth of the Institut für Informatik, E.T.H. Zürich, PASCAL-S defines an official subset to be used to aid in teaching programming. The abstract of the report is given below:

"Pascal-S is a subset of the programming language Pascal selected for introductory programming courses. This report describes an implementation that is especially designed to provide comprehensive and transparent error diagnostics and economical service for large numbers of small jobs. The system consists of a compiler and an interpreter and is defined as a single, self-contained Pascal program. This machine-independent formulation in a high-level language facilitates its construction and is a prerequisite for easy portability."

Standard Pascal constructs omitted in Pascal-S are: scalar and subrange types, pointers, set and file types, with and goto statements, the passing of procedures and functions as parameters, and several standard procedures. The only file operations are read on input and write on output. The report contains a complete listing of the compiler and interpreter on 34 pages!

Pascal-S is currently distributed on tape with the second release of the CDC 6000 Pascal compiler and is written to run under that version. However, Ed Katz of the University of Southwestern Louisiana, Lafayette reports that his department implemented Pascal-S from the report in PL/1 for Honeywell Multics in a semester. (see HERE AND THERE)

AMDAHL 470 (see IBM 360/370 series)

BURROUGHS B-1700 (implementations exist)

B-3700, B-4700 (implementations exist)

B-5700 (implementations exist)

B-6700 Several implementations are listed below:

A.H.J. Sale of The University of Tasmania, G.P.O. Box 252C, Hobart, Tasmania Australia 7001 (phone 23 0561) is known to have developed a compiler based on Pascal-P2.

Kenneth L. Bowles of the University of California, San Diego computer center La Jolla, CA 92037 (phone (714) 452-4050) had a Pascal interpreter running but is now more interested in a good PDP-11 implementation.

G. Goos of the Institut fuer Informatik II, 75 Karlsruhe 1, Zirkel 2 Germany, implemented a compiler based on Pascal-P1, and hence may not be standard. However our information is almost 2 years old and this implementation may have been upgraded.

CII IRIS 80, 10070 (see Xerox Sigma 7)

CONTROL DATA CYBER 18 (an implementation exists)

2550 (Control Data supports a cross-compiler on the 6000/Cyber 70,170)

3300 (implementations exist)

3600 (an implementation exists)

6000/CYBER 70,170 SERIES

This implementation has been developed by Urs Ammann of E.T.H. in Zurich for the last 3½ years. Release 1 of the compiler (named Pascal 6000 - 3.4) appeared in May, 1974, and was updated 10 times over the next 1½ years. Release 2 appeared in March, 1976 which incorporated a massive update to Release 1, update10 to improve performance and reduce memory requirements. An error message summary is provided at the bottom of the listing and a working version of the procedure DISPOSE is included.

A new Report describing the implementation is entitled: "On Code Generation in a Pascal Compiler" by Urs Ammann, April, 1976, 40 pages.

Pascal 6000 - 3.4 was produced by rewriting an older compiler and bootstrapping. It is the first Standard Pascal compiler and its documentation is in the last two chapters of the user manual part of the book: Pascal User Manual and Report.

The Release 2 compiler may be run under SCOPE 3.4, KRONOS 2.1, or NOS,NOS/BE operating systems. It may be edited to change it to 63/64 and ASCII subset character sets.

The compiler is a 7000 line Pascal program plus operating system interface routines. Its core requirements are 45000 octal words of 60 bits each for small programs, but this rises to 57000 octal to compile itself. Compilation speed is about 10500 characters/second on a Cyber 74; 54 seconds of processor time are required to recompile the compiler on a Cyber 74. Its efficiency compares

favorably to FORTRAN; compiling speed and compute bound execution speed being about 1.3 times slower, but I/O execution speed being clearly faster. Its reliability has improved to being excellent, and there are only a few minor bugs outstanding.
Pascal 6000 - 3.4 features a number of extensions to Standard Pascal, and only three restrictions. One may not use a file of files; segmented is a reserved word; and standard (built-in) procedures or functions are not accepted as actual parameters to other procedures or functions.
Distribution is currently being handled in Europe, Asia, and Africa by Urs Ammann, Institut für Informatik, E.T.H. - Zentrum, CH-8092 Zürich, Switzerland. In Australia, contact Carroll Morgan, University of Sydney, Basser Department of Computer Science, Sydney N.S.W. Australia 2006. Costs for these distribution points are unknown; tape format is unlabelled, 7-track Scope internal binary.
For North and South America only, another contact point is George Richmond, Computing Center 3645 Marine Street, University of Colorado, Boulder, CO 80309. Information for the distribution from Mr. Richmond is as follows. Included in the documentation package are:

- Literature about the Programming Language Pascal (4 pages)
- The Release2 Distribution Tape (3 pages)
- Note on the Tape Contents (13 pages)
- Pascal: User Manual and Report (170 pages)
- An Axiomatic Definition of the Programming Language Pascal (32 pages)

Cost for Release 2 with documentation is $50 (by check or purchase order) and $10 additional for a 600 foot magnetic tape if one is not supplied. Because of the similarity in documentation between Release 1 and Release 2, a special offer is extended by Mr. Richmond to previous recipients of Release 1 who want to upgrade. For $25 and your supplying a 600 foot tape, and the documents: The Release 2 Distribution Tape and Literature about the Programming Language Pascal. No other material has changed.
Sent with Release 2 is also the Pascal-S subset compiler and the document: PASCAL-S: A Subset and its Implementation (63 pages).
The tape format will be seven track SCOPE 3.4 internal binary and unlabelled (equivalent to KRONOS 2.1 MT,F=SI,LB=KU). At special request, and at no extra cost, KRONOS formats F=I or F=X can be used. In the near future, nine track binary tapes will be available.
The future of the compiler maintenance is uncertain at this writing. Send bug reports to: John P. Strait, University Computer Center, 227 Exp Engr, University of Minnesota, Minneapolis, MN 55455 USA, or call (612) 376-7290.
Several persons have made modifications to Release 2 for operating system interaction. Interactive facilities under KRONOS Telex have been developed by John P. Strait at the University of Minnesota. An agreement is underway with George Richmond so that these mods can be distributed through George Richmond for the Americas. Michael Hagerty has developed mods for SCOPE 3.2 systems and for INTERCOM. He wants these to be distributed centrally and as yet no agreement has been reached. Mr. T.A. Nemeth of the Computing Centre, University of Adelaide has written mods (no language changes) for SCOPE 3.4 and INTERCOM. They are available for $A10+postage from North Terrace, Adelaide, S.A. 5000 AUSTRALIA.
Hans Jøraandstad of CERN announced changes to put Pascal Release 1 up on 7600 and Cyber 76 systems under the SCOPE 2.1.2 operating system (where Record Manager is used because no CIO exists.) There is no distributions information at present.
See the next page for the order form for George Richmond's distribution only.
Future development plans are also uncertain for Pascal 6000 - 3.4. Several complaints keep echoing over and over. For example Albert Steiner of the Vogelback Computer Center, Northwestern University wrote on 4/26/76: 1) Sets ought to be implemented for more than 59 members. 2) Better storage control and management of dynamic allocation is needed - such as obtaining additional dynamic storage if needed, perhaps in conjunction with a manageable OVERLAY or SEGMENT loading scheme. More news next time.

SOFTWARE WRITING TOOLS for the CDC Pascal 6000 - 3.4 implementation have been around. The first such program is a cross-reference utility for producing a numbered source listing with an index to identifiers. The program usually known as XREF was written by Niklaus Wirth, and has been modified several times over the years, most recently for the second release of the compiler. XREF is written in standard Pascal (just over 200 lines), is small and efficient. The 7000 line Pascal compiler with sequence numbers (making each line 90 characters long) took 17.8 seconds to cross reference on a Cyber 74.
Several pretty-print or indenting programs for Pascal programs have been written. At Indiana University, George Cohn wrote RASCAL (Reformat Pascal) which has three options: I for setting indent width, R to select reformatting mode (indenting style) and W to set output width. RASCAL is written as a 1267 card Pascal program (fully RASCALed). (218 cards fully compressed by itself!) (*ICEBOLed?*)
The most circulated pretty-printer is Michael Condict's FORMAT program, written at Lehigh University. Like RASCAL, FORMAT allows options to be imbedded in special comments. Additionally, the user may specify a file for these directives. Options include: A for specifying right justification width of identifiers in declarations, E for selecting block bracket commenting such as end (*procname*). E also can add comments to for, case, while, and if statements. G specifies spacing between symbols, I specifies indenting width, L for amount of indent on statement wrap-around, P number of blank lines between procedures, S number of blanks between two statements on the same line, R input width limits, W output width limits, N write line numbers on FORMATted output, B compress program (*ICEBOL it*), C combine more than one statement per line if possible, D select or deselect outputting of source, and F eliminate formatting and copy verbatim.
Both RASCAL and FORMAT are copyright. FORMAT has been sent to educational institutions, and perhaps in the future, a distribution agreement can be set up.
At the University of Minnesota, a program for pretty printing is being developed called SPRUCE. At the University of California, Berkeley, a pretty printer is called PPRINT.
All the pretty print programs so far suffer from the ability to recover from syntactically incorrect Pascal programs. Comments in the input are generally not handled very well either.
The University of Massachusetts has been working on a "Pascal Assistant" to make interactive programming in Pascal more pleasant. Henry Ledgard, Andrew Singer, and Jon Hueras have developed this system to run under NOS. A User's Guide has been written.
N. Solntseff of McMaster University, Hamilton, Ontario, Canada L8S 4K1, Dept. of Applied Mathematics has sent a report describing EDITABSLIB. It is designed to maintain a library of complete Pascal programs in binary form. Entitled "A Suite of CDC6400 Control-Statement Procedures for the Maintenance of a Binary-Deck Library", its abstract follows:

> "This report describes the implementation and gives examples of the use of EDITABSLIB, a library-maintenance system which allows a user to assemble and maintain libraries of program modules which do not meet the requirements of the standard SCOPE EDITLIB system, namely, Pascal 6000 and COBOL core-image modules, as well as binary (or source) modules of a "minicomputer support system". EDITABSLIB can be used in the batch mode, as well as interactively via INTERCOM. It represents a complete library-maintenance system."

EDITABSLIB is currently running under SCOPE 3.4.3 and a program to manipulate the loader prefixtable for the library is written in Pascal.

Release 2 of Pascal 6000-3.4 Distribution Tape

(Use this only if you are in North America or South America.)

The tape prepared for you by the University of Colorado Computing Center is an unlabeled tape of the following format:

○	MT, F=SI, D=HY	Scope standard binary, seven track
○	MT, F=I, D=HY	Kronos internal binary, seven track
○	MT, F=X, D=HY	Kronos external binary, seven track
○	OTHER: ______________________	

The tape contains seven files of information following by another seven files of the same information. Instructions for installing the Release 2 system can be found in the first file of information.

In order to offer maintenance of the Pascal System the University of Colorado Computing Center needs the name and address of a responsible party. Please fill out the bottom of this form and return it to:

Mr. George H. Richmond
University of Colorado
Computing Center
3645 Marine Street
Boulder, Colorado 80309
USA

- -

Pascal will be maintained by:

Name: ______________________________

Address: ______________________________

____

____

______________________________ Zip Code __________

22 April 1976

CRAY RESEARCH CRAY-1 (No information yet.)

DATA GENERAL NOVA 800, NOVA 1200, SUPERNOVA, ECLIPSE

(No reported implementations - but we need information!)

DIGITAL EQUIPMENT (DEC) PDP-8 (No known implementations)

PDP-11

In wading through the morass of information on PDP-11 versions of Pascal, we found that although many implementations (too many) exist or are in development, very little concrete data is available on operating system and hardware requirements. Distribution and maintenance for most versions remain a mystery. We hope to provide more meaningful summaries in forthcoming issues, and to this end we solicit comments from implementors/distributors/maintainers/users - please refer to the CHECKLIST.

Following is the information we do have. We have an obligation to provide you with complete, up-to-date implementation news, and so will print skimpy descriptions only once.

First and foremost, Per Brinch Hansen has implemented both "sequential Pascal" and his "Concurrent Pascal" for the PDP-11/45. These compilers, written in Pascal, run under the Solo operating system, which itself is written in Concurrent Pascal. A portable version of Concurrent Pascal also exists. Manuals, reports, and distribution tapes <u>have been</u> available from Per Brinch Hansen, Information Science 286-80, California Institute of Technology, Pasadena, CA 91125. However, Richard Cichelli has reported that Brinch Hansen has moved to the U of Southern California and is no longer distributing Pascal. (*How about it, Per?*)

The Pascal Group at the University of Illinois has a completed version of Pascal for the PDP-11/20, running DOS. Although this version seems to be the most widely distributed, it departs widely from Standard Pascal: a) No type <u>set</u> or real, b) No <u>packed</u> <u>arrays</u>, c) <u>Arrays</u> and <u>records</u> must be passed by reference, d) Many non-standard character set equivalences are allowed, e) Many abbreviations for reserved words are allowed, f) Compiler options do <u>not</u> follow the convention of (*$ options*), g) <u>Else</u> allowed on <u>case</u> statements, h) Unconventional extensions have been made to the standard procedures read and write, i) The assignment operator may be used in expressions, introducing semantic ambiguities in evaluation of certain expressions. These differences disallow many programs written in Standard Pascal, having serious consequences for software portability. For more information write: Pascal Group, 267 DCL, University of Illinois, Urbana, IL 61801.

Electro Scientific Industries has completed a compiler for the PDP-11/04 with RT-11. This compiler, written in MACRO-11 assembly language, was based on the University of Illinois implementation. Arthur A. Brown has informed us that this version is available for $1500. Excerpts from a paper by David Rowland describing ESI's use of Pascal appeared in Newsletter #4. Write or telephone: David Rowland. Electro Scientific Industries, 13900 N.W. Science Park Drive, Portland, OR 97229, (503) 646-4141

C. Bron and W. deVries have implemented a cross-compiler from the DEC-10 to any member of the PDP-11 series. There are no operating system requirements for the PDP-11 used. It appears that this version, which was developed from Pascal-P, closely implements Standard Pascal with the exception of files. The compiler generates absolute load-modules in "position independent" code. The development of this compiler is described in "A PASCAL Compiler for PDP-11 Minicomputers", <u>SOFTWARE - Practice and Experience</u>, Vol. 6, pp.109-116 (1976). The system is available on 9-track magnetic tape or on DEC-tape (two reels are necessary) and can be obtained - free of charge - by anyone who sends a tape (please do <u>not</u> send tape as parcel, but as letter) to C. Bron or J. Entrop, Dept. of Electrical Engineering, Twente University of Technology, P.O. Box 217, Enschede, Netherlands. Five files will be written on the tape - the compiler's source code, a documentation file, the text of the runtime support in PAL-11, the code of the runtime support in compiler input format, and an auxiliary program to transform output from the MACRO-11 assembler into compiler input format. The latter two files are only relevant for those who wish to alter the existing runtime-package.

A Pascal compiler based on JANUS has been implemented on the PDP 11/45 running under DOS/Batch operating system. This compiler, written in Pascal, generates code for the standard abstract machine JANUS. William Waite's macro processor STAGE2 is used to translate JANUS into the MACRO-11 assembly language. The compiler compiles itself in 64K words of memory and 604 seconds. The Pascal compiler, Stage2, the Pascal-11 User's Guide, and the whole Pascal-11 system are available from: Lucien Feiereisen, Institut f. Biokybernetic u. Biomed. Technik, Universitaet Karlsruhe D-7500 Karlsuhe 1, Kaiserstrass 12, Germany.

The remainder of the implementation information is skimpy. The following is a list of addresses and short notes:

Timothy W. Hoel, Academic Computer Center, St. Olaf College, Northfield, MN 55057 (507) 663-3096: "I know of (*a Pascal compiler running under the UNIX operating system*) which is a 7-pass translator producing C-code, the main HLL under UNIX." - July, 1976.

Arthur A. Brown, Apt. 1002, 1101 New Hampshire Ave. NW, Washington, DC 20037, (202) 785-0716: "I am currently trying to introduce Pascal in my company's roster of languages, and intend to implement it on a minicomputer of my own during the coming year (*PDP-11/04*)" - 27 May 1976.

Andrew S. Puchrik, 11623 Charter Oaks #202, Reston, VA 22090, (703) 893-4330: "My company (INCO, Inc.) has ordered the Solo Operating System from Cal Tech. "I plan to run the (sequential) compiler under RSX-11D. (We just got version 6B.) The first problem is to emulate a floating point processor or patch-up the compiler. We don't have the f.p.p. on our 11/45." - 24 April 76.

W. H. Huggins, Department of E.E., The Johns Hopkins University, Baltimore, MD 21218: Tim Hoel informed us in August that W.H. Huggins has a version of sequential Pascal which runs under UNIX and produces PDP-11 machine code. Huggins will send it for $150 and a photo copy of Brinch Hansen's Solo contract.

Henry Spencer, Box 302 Sub 6, Saskatoon, Sask. S7N OWO Canada, "implementation in progress with P-code on PDP-11 under UNIX" - October, 1975.

Kenneth L. Bowles, Dept. of Physics and Information Science, Computer Science Division, C-014, University of California, San Diego, La Jolla, CA 92093, (714) 452-4050: Bowles is working on an interpretive sustem based on Pascal-P. He hopes to be able to modify the compiler to allow it to run in 20K or perpaps 16K words of memory for the PDP-11. See his letter to A.H.J. Sale printed in Newsletter #4.

H. H. Nägeli of E.T.H., Zürich reports that the Pascal Trunk compiler is being used to write a cross compiler for the PDP-11 in Zürich.

Gordon Stuart, Camosun College, Technical and Vocational Institute, 1950 Lansdowne Road, Victoria, B.C. V8P 5J2 Canada: Implementing version for the PDP-11/40 status unknown, as of 3 March 76.

Andrew S. Tannenbaum, Wiskundig Seminarium, Der Vrije Universiteir, Amsterdam - 1001 Postbus 7161, De Boelelaan 1081, Netherlands: Implementing version for PDP-11/45 status unknown as of 7 February 76.

DEC SYSTEM-10, PDP-10

For the last several years, H.-H. Nagel of the Institut für Informatik, Universität Hamburg has implemented DEC-10 compilers for Pascal. The one most widely in circulation comes as a dual system: PASCAL is a compiler which produces shared (.SHR) and low (.LOW) segments directly, and the other compiler PASREL, produces relocatable code which can be processed by the loader into shared and low segments. The relocatable compiler, PASREL, has more predefined procedures and functions available than the PASCAL compiler and has a powerful interactive debugging package. These compilers are being improved with respect to efficiency, standardization, and the addition of still missing or desirable language features. H.-H. Nagel wrote on June 14, 1976: "We are just introducing a new version of the DECSystem-10 Pascal compiler. I hope to be able to send some information about this compiler to you in the near future."

Documentation for the system comes on tape. In February, 1976, Wilhelm Bürger of the University of Texas produced a technical report (22B) with Nagel in English entitled: "PASCAL on the DEC10". From this report we find that interactive use of Pascal is described, and that there are several serious omissions from Standard Pascal. They are: a) No label declarations are allowed, b) Not all ASCII characters are allowed - specifically control characters,c) The program declaration is not implemented, and d) Procedures and functions cannot be formal parameters. However subroutine linkage in PASREL is provided for assembler, Algol, Cobol, and Fortran programs!

We have no performance/comparison data available. The compiler runs under TOPS-10.

The method of distribution is in the form of a chain of sites passing a tape from one to the other - a chain. We have no cost information. In Europe write to H.-H. Nagel, Institut für Informatik, SchluterstraBe 70, D-2000 Hamburg 13, Germany. In the US and Canada write to Wally Wedel or Wilhelm Bürger, Computation Center, University of Texas at Austin, Austin, TX 78712. Please specify if you are on a chain. Shipment requires either a DECtape for the PASCAL compiler and 2 DECtapes for the PASREL compiler, PASDDT, crossreferencing program CROSS, and PASCAL-Help file in German. 1 MAGtape can be supplied instead of the DECtapes but DECtapes are preferred. Please indicate if you are willing to be part of the distribution chain - that is whether you are willing to provide a copy of these files to someone else if asked to do so.

Perhaps the proposed new release of the compiler(s) will see an improvement in implementation and distribution information.

Complaints about the DEC-10 implementation are from Jim McCool, IUPUI, Computing Services, 1100 W. Michigan, Indianapolis, IN 46202 (317) 264-3836 who writes: "We are currently searching for a reliable version of PASCAL for the DEC-10 computer. We have the release from the University of Hamburg, but have found it completely unsatisfactory. I am writing you in the hope that, as the head (sic) of the Pascal User's Group, you might know of some other implementation of the language for the DEC-10." (*pretty strong, although vague stuff.*)

Charles Hedrick, 183 Commerce West, University of Illinois, Urbana, IL 61801, (217) 333-4515, and (217) 356-8425 writes: "About the DEC10 PASCAL (PASREL, actually) I understand it was an undergraduate class project. As such it is very good. It is fast and generates fairly good code. However it does not follow the usual conventions for DEC10 compilers, and in general is not well adapted to the DEC10 operating system: 1) Programs cannot simply be loaded and run as with other languages. Core must be explicitly allocated, something that complicates explaining its use to students. I realize that for efficiency reasons, they did not want to have dynamic expansion of core, but the least they could do is start with some default amount greater than 0 that would let simple programs run (e.g. the 4K recommended in the (German language) help file). I have put such a patch in our local version. 2) The compiler cannot be called by the DEC10 compiler-caller COMPIL. This means that the monitor commands COMPILE, EXECUTE, etc. cannot be used. Rather the compiler must be explicitly run, and then the loader must be invoked. Again, an inconvenience with beginners. 3) Lower case characters are ignored everywhere - is source code, comments, strings, filenames, everywhere! 4) A file OUTPUT is created on your disk area even when no output is done to the default channel (or at all!) This file has a "DATE75 error", that is its creation date is 5-Jan-74, whatever the current date. 5) A listing file is always created, and cannot be surpressed. This file is largely useless, as it is usually just a copy of the source. 6) Options to the compiler are included in the source code as comments... DEC10 convention is to use "switches" after the file name. E.g. to get a listing one says .EXECUTE FILE.PAS/LIST. COMPIL passes on the command string to the compiler: FILE.REL,FILE.LST=FILE.PAS/L. 7) Parameter passing in procedures is full of bugs: e.g.

```
                type foo = packed array[1..10] of char;
                procedure foobar(a:foo; b:integer, c: integer, d:foo, e: integer);
                          (*body*);
                   (*etc*)
                foobar('ABCDEFGHIJ',1,2,'IJKLMNOPQR',3);
```

will assign some large negative number to parameter e in foobar. We have fixed this bug, but it is only a minor manifestation of generally poor design of parameter passing. The following case is harder to fix, and we haven't done so. (It would seem to require a complete rewrite of procedure linkage: procedure foo(a: real, b: real); and then foo(1.0,sin(0.0)); (I think it may only fail if there are more than 5 parameters to the procedure.) Calling sin clobbers the display which is being set up for the call of foo, and so the parameters, etc. are garbaged. 8) When a program wants to input a real number, typing 1 or even 1. causes a fatal error. This makes it hard to write programs to be used by non-computer programmers. DEC FORTRAN and all the other languages I know, do the type conversion. At least PASCAL's runtime should let you retype the number rather than blowing up. 9) the KI-10 instruction set is not used. When using a lot of mixed type arithmetic, I have speeded up execution time by a factor of 2 by replacing calls to routines for integer-to-real and vice versa with the equivalent machine operations (FIX and FLOATR).

FOXBORO FOX-1 (an implementation exists)

FUJITSU FACOM 230-38 (an implementation exists)

FACOM 230-55 (an implementation is underway)

HEWLETT PACKARD HP-2100 (no known implementations)

HP-3000 (no known implementations)

HITACHI HITAC 8800/8700 (see IBM 360/370 series)

HONEYWELL SERIES 6 (an implementation is being considered)

H316 (an implementation is underway by Honeywell Corporate Research in Bloomington, Minnesota)

600/6000 SERIES

Robert A. Stryk of Honeywell Corporate Research (612) 887-4356 reports that The University of Waterloo has implemented Pascal on the 6000 series. Honeywell Information Services in Phoenix, Arizona has purchased it and is offering it as a standard software product under revision H of the GCOS operating system. Because it is a supported product, it will cost money - how much is not known. - July 16, 1976.

IBM SYSTEM 360/370

There are three implementations for which there is detailed information. Most impressive is the University of Tokyo's Hitac 8000 implementation which will run on either the Amdahl 470 or the IBM 360/370. The following correspondence describes that version.

Following that is a description of the University of Maniboba IBM 360/370 compiler. It has been in the development stage until recently. It features stability and compatibility with OS.

A widely publicized but less efficient version of Pascal is one from SUNY Stony Brook. It has received criticism for poor reliability.

There have been many requests for IBM 360 implementations. The volume of information reproduced here is in response. From a performance standpoint, the compiler by Teruo Hikita and Kiyoshi Ishihata is excellent. It is written in Pascal, requires only 110K bytes and is only beat in execution speed by the Fortran compiler at full optimization. It is well documented by two reports. The only problems at the present are distribution. Although we have not heard from Mr. Hikita since May 17, it would be nice for a site (such as Southern Cal) to aid in its distribution in North America. It was Susan Stallard's letter (see OPEN FORUM) which stated a desire to see a version with smaller core requirements than the Manitoba Pascal compiler.

February 21, 1976

Mr. A. B. Mickel
University of Minnesota
Computer Center
227 Experimental Eng. Bldg.
Minneapolis, Mi. 55455
U.S.A.

Dear Mr. Mickel;

This is a short notice on our recent implementation of PASCAL based on a "trunk" compiler. Dr. H. H. Naegeli told us to send it to you for the PASCAL Newsletter. (Please see an enclosed copy of his letter.)

An extended version of Standard PASCAL named PASCAL 8000 was designed, and its compiler was implemented on a Japanese computer HITAC 8800/8700 at the Computer Center of the University of Tokyo. This computer is a multi-processor system of 4 CPU with 4 megabytes main memory, and it has a quite similar machine instruction set to that of the IBM 360 and 370 series computers.

Our language extensions are concerned with constant definitions for structured types, variable initializations, new control structures named "forall" and "loop" statements, and "procedure skeletons" for procedure and function parameters proposed by Lecarme-Desjardins.

The implementation was done by bootstrapping using the PASCAL-P interpretive compiler developed by U. Ammann. Our new compiler is based on H. H. Naegeli's "trunk" compiler. This is a source program of a PASCAL compiler written itself in PASCAL, in which machine dependent parts (code generation, addressing, etc.) are marked and strictly separated from other machine independent parts, and detailed comments are provided which indicate the data or algorithms to be described to the final form by an actual implementor. The version we used was still at developmental stage, though almost completed.

About 1200 lines of the source program have been rewritten to the final form for our HITAC 8800/8700 out of total 5260 lines, and about 400 lines have been newly added for our own language features. This amount of rewriting is considered to be fairly small compared with other works of bootstrapping appeared in the literatures, and one of the values of the trunk compiler of course lies in this point. We felt difficulty during the rewriting process mainly in the expression evaluation scheme. For the other parts, the coding was rather simple and straightforward, though not trivial.

For the details of the language definition of PASCAL 8000 and its implementation based on the trunk, the following two technical reports will be available from our department in March.

T. Hikita, K. Ishihata, PASCAL 8000 Reference Manual.
K. Ishihata, T. Hikita, Bootstrapping PASCAL Using a Trunk.

Sincerely,

[illegible]

Teruo Hikita

[illegible]

Kiyoshi Ishihata

T. Hikita, K. Ishihata
Department of Information Science
Faculty of Science
University of Tokyo
Tokyo 113 Japan

Dear Mr. A. B. Mickel, March 24, 1976

I have sent to you under separate cover the following two technical reports on our recent implementation of PASCAL, which I mentioned in my recent short notice for the PASCAL Newsletter:

"PASCAL 8000 Reference Manual"
"Bootstrapping PASCAL Using a Trunk"

They have been prepared rather hurriedly, and there do remain several typographical errors. But we believe our implementation details would be clear from these reports.

Sincerely Yours,

[illegible]

Teruo Hikita

UNIVERSITY OF MINNESOTA
TWIN CITIES

University Computer Center
227 Experimental Engineering Building
Minneapolis, Minnesota 55455

March 31, 1976

Prof. Teruo Hikita
Information Science Laboratories
Faculty of Science
University of Tokyo
2-11-16 Yayoi, Bunkyo-Ku
Tokyo, 113 JAPAN

Dear Prof. Teruo,

Thanks to both you and your colleague, Mr. Ishihata for coming forward with the Hitac 8800 implementation information for PASCAL. I recently received both your March 24 letter and the copies of the Technical Reports.

I assure you that the new PASCAL Newsletter shall publicize the Hitac 8800 version properly. In this regard, are you willing to distribute the system formally and will you accept bug reports?

I hope that you will be willing to join the PASCAL User's Group. I'm sorry in fact to be so late in answering you, but this was due to delays in setting up P.U.G. and in transferring the editorship duties of the Newsletter.

Thank you very much.

Sincerely,

[illegible]

Andrew B. Mickel

ABM/kp

Enclosures

INFORMATION SCIENCE LABORATORIES

FACULTY OF SCIENCE, UNIVERSITY OF TOKYO
2-11-16 YAYOI, BUNKYO-KU TOKYO, 113 JAPAN

April 16, 1976

Prof. Andy Mickel
University Computer Center: 227 Exp Engr
University of Minnesota
Minneapolis, MN 55455
U.S.A.

Dear Prof. Mickel,

Thank you for your March 31 letter with information on the new Pascal User's Group. I am glad to hear it, and of course I would like to join it. I enclosed a required coupon and a check in this letter.

As for the distribution of our Hitac 8800 version Pascal, we are ready to distribute the system formally and accept bug reports. Actually we have just completed its transportation to another Hitac 8000 series computer. Although the Hitac 8800 is a Japanese computer rather locally used in the world, its machine instruction set is almost compatible to the IBM 360 and 370 series computers, and we may plan the slight modification of the system (mainly its interface with the operating system) to cope with the IBM systems.

I appreciate your kind suggestion and information. I am looking forward to the publication of the new Pascal Newsletter in September.

Thank you very much.

Sincerely,

[illegible]

Teruo Hikita

Enclosures

TH/mk

UNIVERSITY OF MINNESOTA
TWIN CITIES

University Computer Center
227 Experimental Engineering Building
Minneapolis, Minnesota 55455

April 29, 1976

Dear Teruo,

Thanks for joining PUG! We now have over 100 members and I'm looking forward to the first newsletter.

Your good news about willingness to distribute your compiler is very important I think. As you may know there are at least 6 IBM 360/370 Pascal compilers but they all seem very unsatisfactory. I know of 3 which are standard, 2 take 180K Bytes and are multi-pass and the other is not ready yet. (SUNY Stony Brook, U of Manitoba, and New Mexico Tech). Also I've been receiving many requests for IBM 360 versions by new members of the User's Group. It is my opinion that your compiler, being of high quality could be <u>the</u> answer and to ease the distribution matter in this country, it might be wise to find a friendly installation who could in turn pass it on to others.

The most detailed request has come from: Ms. Susan L. Stallard
Academic Services
University Computing Center
University of Southern Californiz
Los Angeles, CA 90007

Maybe you could send her your Tech Reports together with the information about your intention to making an IBM-compatible version.

With the information explosion, I'll have to be putting together a list of questions to ask of every implementer about an implementation and so I plan to send this list to you before September. This way I can print such information in the Newsletter in a standard form.

It has been a pleasure to receive your prompt and timely letters.

Sincerely,

Andy

Enclosures: S. Stallard's letter + my letter to 3 installations

DEPARTMENT OF INFORMATION SCIENCE
FACULTY OF SCIENCE, UNIVERSITY OF TOKYO
2-11-16 YAYOI, BUNKYO-KU TOKYO, 113 JAPAN

May 17, 1976

Andy Mickel
University Computer Center: 227 Exp Engr
University of Minnesota
Minneapolis, MN 55455
U.S.A.

Dear Andy,

Thank you for your April 29 letter, and I am sorry for writing you a bit late.

We have received some other inquiries besides yours on the possibility of distributing our Pascal system, and we are just to begin the modification of the system. The only difficulty is the difference between the operating systems of our Hitac 8800/8700 and IBM 360/370 line computers. It will take at least one month to finally prepare a distribution tape.

By the way, I should note that I am not so optimistic on the transportation of our system. One reason is that our compiler does not generate a load module of IBM systems, but rather it generates absolute (not relocatable) code. Secondly, I am afraid whether we can easily find a friendly institution who handles the distribution.

Anyway, I will be writing to you again at the completion of the work above. We would like to consider at that time on the possibility of distributing our system in the U.S.

Thank you.

Sincerely,

[illegible]

Teruo Hikita
Research Associate

TH/yk

8.1. Performance of the compiler

In the reference [15] is given several experimental data on the performance of the PASCAL compiler for CDC 6000. The execution times on the HITAC 8800/8700 for the same four test programs written in PASCAL and FORTRAN are shown in the following table. The result is that the PASCAL program runs faster than the corresponding FORTRAN program compiled without optimization, which shows that PASCAL can actually be implemented efficiently. But it is slower than the FORTRAN program compiled by a full-optimizing compiler by factor of 2-4 in case of small non-recursive programs. For the recursive programs such as "Partition", PASCAL programs run as fast as those of FORTRAN, probably because of the overhead of moving the data to and from the hand-coded stack in FORTRAN programs, compared with that of the automatic stack allocation of variables in PASCAL.

	PASCAL 8000	PASCAL-P(R+)	PASCAL-P(R-)	FORTRAN(0)	FORTRAN(2)
Matmult (n=100)	5.18 sec.	93.87	79.93	6.91	1.36
Sort (n=2000)	7.98	121.54	119.94	13.86	3.46
Count	70.76	------	------	------	56.71
Partition (n=30)	0.59	3.01	2.81	0.77	0.50

PASCAL 8000	without runtime error checking
PASCAL-P(R+)	with runtime error checking
PASCAL-P(R-)	without runtime error checking
FORTRAN(0)	without optimization (Hitachi OS7 FORTRAN OPT=0)
FORTRAN(2)	with optimization (Hitachi OS7 FORTRAN OPT=2)

8.2. Statistical data of the compiler

The size of the current version of the compiler is about 110 kilobytes, in which machine instructions occupy 89.8% and constants 10.2%.

UNIVERSITY OF MINNESOTA
TWIN CITIES

University Computer Center
227 Experimental Engineering Building
Minneapolis, Minnesota 55455

April 5, 1976

Prof. J. M. Wells
Department of Computer Science
University of Manitoba
Winnipeg, Manitoba
CANADA

Dear Prof. Wells,

We understand you have a PASCAL compiler for the IBM 360/370 series machines. Because this is an important implementation to publicize in the PASCAL User's Group Newsletter, I'm now writing to find out detailed information.

There are at least 6 IBM 360 implementations I know of. Three seem to have been based on the revised version of the language and are compilers: yours, and the ones from PASCAL Compiler Project, Dept. of Computer Science, SUNY at Stony Brook, Stony Brook, NY 11794 and Prof. Thomas S. Nartker, Dept. of Computer Science, New Mexico Tech, Socorro, NM 87801. Were you aware of the existence of the other implementations and if so, how good are they compared with your system?

Are you willing to distribute your system to other sites for a nominal handling charge, provide documentation, and accept bug reports?

I really look forward to hearing from you. Enclosed is some PASCAL User's Group information.

Sincerely,

Andy

Andrew B. Mickel

ABM/kp

Enclosure

THE UNIVERSITY OF MANITOBA

DEPARTMENT OF COMPUTER SCIENCE

WINNIPEG, CANADA R3T 2N2

15 July 1976

Ph. (204) 474-8466

Mr. Andy Mickel
Pascal User's Group
University Computer Center
227 Exp. Engr.
University of Minnesota
Minneapolis, Minnesota
U.S.A. 55455

Dear Mr. Mickel:

I am answering your letter of April 5, 1976 to Professor James Wells. I have been preparing a new release of the Manitoba PASCAL Compiler and therefore waited until this was complete so that I could send you the latest information.

A restricted release was made in December, 1975 to ten test sites. Work on the Compiler has continued and a new release is now generally available under the conditions set out in the enclosed description.

The reports "MANITOBA PASCAL USER GUIDE" and "MANITOBA PASCAL CODE GENERATION" are included on the distribution tape in upper and lower case with printer control characters so that copies can be run off locally.

I wrote to Professor Nartker at New Mexico Tech. concerning their compiler. It seems that they are debugging statements and expressions and hope to have the compiler running by the New Year.

I ordered the SUNY compiler for comparison with ours. In the areas of cost, ease of installation, compile speed, compile-time error messages, formatting capability, distributed documentation, standard scalar types, and compatibility with OS conventions, I think the Manitoba PASCAL Compiler is superior.

cont'd.... 2

Mr. Andy Mickel - 2 - 15 July 1976

To date, thirteen copies of the Manitoba PASCAL Compiler have been distributed to sites in Canada, the United States, and Europe.

There has not been a Pascal Newsletter since the announcement of the initial release was sent to George Richmond, and the Compiler has therefore received very little advertising.

Although the Compiler is released on an "as is" basis, a complimentary copy is being sent to the initial test sites which answered a questionaire, and is being offered to the other sites.

Intensive local testing led to the discovery of several problems in the initial release, but only one "bug" was reported by one of the test sites in six months of use.

I hope the enclosed information reaches you in time for inclusion in the September Newsletter. I would appreciate it if you would acknowledge receipt of this information.

Thank you for your interest.

Sincerely,

W. Bruce Foulkes

W. Bruce Foulkes

WBF:emr

UNIVERSITY OF MINNESOTA
TWIN CITIES

University Computer Center
227 Experimental Engineering Building
Minneapolis, Minnesota 55455
(612) 373-4360

July 26, 1976

Dear Bruce,

You don't know how much I appreciate your waiting to send such complete information on your Pascal compiler. Thanks very much! Rest assured that we will devote space in the Newsletter to it. Thank you very much also for joining the User's Group. We now have 291 members.

Until your letter and enclosures arrived there was much mystery in my mind about IBM 360/370 Pascal implementations when it came to recommending one to somebody. This is because I knew little about yours; all I had received was a complaint from Southern Cal that 160K bytes was too large. The SUNY Stony Brook compiler did not impress me in that 1) it was written in XPL, 2) It was large and slow, and 3) Several people now have reported bugs to me and say that it will become unsupported in a few months. I received a 14 page faded xerox of their documentation.

I received a phone call from New Mexico Tech and found out about their project.

What may interest you, is that a very high quality compiler implementing all the aspects of standard Pascal for an Hitachi 8800 (Amdahl 470/IBM 370) has been written in Japan. It is only beat out in efficiency by the Fortran compiler at full optimization. It is written in Pascal, and takes 110Kbytes. They have produced two nice technical reports which you may want to write for:

Pascal 8000 Reference Manual Version 1.0
Bootstrapping Pascal Using a Trunk

The people are: Teruo Hikita and Kiyoshi Ishihata
Department of Information Science
University of Tokyo
2-11-16 Yayoi, Bunkyo-Ku
Tokyo, 113 JAPAN

Their current work is adapting the compiler to run on standard IBM 370 operating systems.

What alarms me, now that in the last few months I've learned about many other Pascal implementations, is that one of the design goals of Pascal is being subverted: namely that of being a vehicle of portable software. Taking standard Pascal, only Pascal 6000 (for CDC) and this Hitachi 8000 compiler will accept standard programs. The PDP 11, DEC 10, your IBM 370, and etc., are restricted in a small number of fundamental areas. It wouldn't matter if the details were esoteric. But let's look at yours for example:

- some standard identifiers are reserved words
- packed structures are not allowed (the symbol packed could be ignored)
- no program declaration, no GET and PUT, comments are /* */ instead of (* *), no character subranges, no square brackets [and]

What's the deal?

Well, I'm just pointing some things out and don't want to sound too negative. Thanks again for your service in sending these things.

Sincerely, Andy

MANITOBA PASCAL COMPILER

A PASCAL Compiler for IBM 360/370 computers has been developed by the Department of Computer Science at the University of Manitoba. The Compiler was written by W. Bruce Foulkes under the supervision of Professor James M. Wells.

The Compiler is one-pass and uses a top-down parsing strategy. A generated assembler parser is produced by the translator writing system SYNTICS. All semantic routines are written in PL360 and system interfaces are written in Assembler.

The Compiler is not a rewrite, modification, or bootstrap of any previous PASCAL compiler. The Compiler uses some routines provided by the SYNTICS system and borrows some ideas and code from the ALGOL W compiler for code generation, built-in functions, and I/O.

The distributed version of the Compiler requires approximately 180K bytes of memory. This size is variable, but the minimum size for compiling a meaningful program is approximately 160K bytes.

The design strategy has resulted in a very fast compile speed, averaging more than 200 lines of source per second on an IBM 370/158.

Considerable effort has been spent on localized optimizations in areas such as array subscripting, record field accessing, and boolean expression evaluation, with the aim of producing a compiler suitable for the compilation of application programs.

Extensive compile-time checking is performed and approximately 130 different error and warning messages are provided.

The production of run-time checking code for array subscripts, subrange assignments, values returned by PRED and SUCC, etc., can be turned on or off on a line-by-line basis. There are approximately 40 run-time error messages. Each error diagnostic consists of an error message, the location in the current segment, the invalid value if appropriate, and a traceback of all segments invoked.

The Compiler produces OS-compatible object modules and uses standard IBM linkage and parameter lists in calls of external routines. This allows many existing library routines, such as CALCOMP plot routines, etc., to be called from a PASCAL program.

The Compiler supports a subset of the language described by Kathleen Jensen and Nicklaus Wirth in "PASCAL User Manual and Report". The main differences are listed below.

- Only the standard input and output files SYSIN and SYSPRINT are supported. All I/O is done through the use of READ, READLN, WRITE, WRITELN, EOLN, and EOF. The I/O is not exactly standard; in particular, formatting is also allowed on input.

- The program header is not required. SYSIN and SYSPRINT must always be provided.

- PACKED arrays and records are not supported.

- Only the simple forms of procedures NEW and DISPOSE are allowed. Tagfield values must not be specified. No garbage collection is performed.

- Global labels are not implemented.

- Subranges of characters are not allowed.

There are two main limitations imposed by the Compiler. The maximum nest allowed for procedure and function declarations is 5, and all program segments are restricted to 4K bytes of code.

Seven standard scalar types are provided: SHORT INTEGER, INTEGER, REAL, LONG REAL, BOOLEAN, CHAR, and STRING.

Built-in functions include: ABS, SQRT, EXP, LN, LOG, SIN, COS, ARCTAN, SQR, SUCC, PRED, ODD, ROUND, TRUNC, ORD, CHR, CARD, CPUTIME, IAND, IOR, XOR, SLA, SRA, SLL, and SRL.

The source for the Compiler initialization routine is provided. This routine sets the size limits for all compile-time tables, and also sets defaults for compiler flags (such as whether run-time checking code is to be produced). This should allow the Compiler to be tailored to suit the needs of any installation.

An initial release of the Manitoba PASCAL Version 1 Compiler was made in December 1975 to approximately ten sites. Work on the Compiler has continued and a new release is now generally available. The Compiler has undergone considerable usage and has proven to be quite reliable.

A distribution fee of $50.00 (payable to the "Department of Computer Science, University of Manitoba") is required to cover our distribution costs, including a 600-foot 9-track distribution tape. An order must be accompanied by a signed "SOFTWARE RELEASE AGREEMENT".

The present distribution tape contains the six files described below.

1. A description of the tape contents, including sample JCL for installing the Compiler.
2. A load module library containing the Compiler, a one-step monitor, and run-time library routines.
3. The 40-page manual "MANITOBA PASCAL USER GUIDE" in upper and lower case characters, which describes features and restrictions of this implementation.
4. Sample PASCAL programs.
5. IBM 360 Assembler source for the Compiler initialization routine.
6. The 80-page manual "MANITOBA PASCAL CODE GENERATION" in upper and lower case characters, which describes the run-time organization and demonstrates the code generated for most constructs in the language.

UNIVERSITY OF MANITOBA
DEPARTMENT OF COMPUTER SCIENCE
WINNIPEG, MANITOBA, CANADA
R3T 2N2

SOFTWARE RELEASE AGREEMENT

Software (name) Manitoba PASCAL Compiler, Version 1

The undersigned, representing the educational institution or company identified below, accepts the software named above and agrees to the following conditions regarding its use and/or distribution. The University of Manitoba Department of Computer Science in turn, grants to the below-named a non-exclusive and non-transferable license to use the above named software subject to the following conditions.

1. Software is distributed to educational institutions and companies only; not to individuals. The educational institute or company named below agrees to maintain control of the released software, and not to redistribute it to any other individual, institution, or company without the express written permission of the University of Manitoba Department of Computer Science.

2. All credits in listings and/or documentation whether names of individuals or organizations, will be retained in place by the receiving organization unless written release from this responsibility is obtained in writing from the University of Manitoba Department of Computer Science. The Licensee shall take all reasonable precautions to maintain the confidentiality of the coding; at least equivalent to those employed by the Licensee with protection of its own confidential information.

3. All software is released on an "as is" basis, and no warranty as to performance or effect on hardware or other software is expressed or implied. The University of Manitoba Department of Computer Science accepts no liability of any kind in releasing the above-named software.

4. In releasing the above-named software, the University of Manitoba Department of Computer Science accepts no responsibility for installation, maintenance or functioning of that software except that refunds will be made if requested within 90 days, and accompanied by return of all software materials and a statement that no copies have been made or retained. Responses to reasonable requests concerning the above-named software will be made by mail or telephone.

2....

SOFTWARE RELEASE AGREEMENT

5. The License shall be non-exclusive and the University of Manitoba Department of Computer Science shall have the right to grant any further and additional licenses or to make such other use of the coding as it shall desire.

6. The above named software is released on a no-fee basis/as per attached schedule.

   This distribution does not entitle the Licensee to future releases of any of the above-named software.

7. The above named software is released in binary form. The distribution medium is magnetic tape, the format of which is described in the documentation.

(please print)

Name: ______________________

Title: ______________________

Organization: ______________________

Address: ______________________

____

____

Authorized Signature: ______________________

Date: ______________________

____
For the University of Manitoba Department of Computer Science

PASCAL ORDER FORM

Please send a distribution tape containing the Manitoba PASCAL Compiler, Version 3.
I have read and agree to the conditions set out in the SOFTWARE RELEASE AGREEMENT.

SIGNATURE ______________________ NAME ______________________________

POSITION ______________________________________ DATE ______________

EDUCATIONAL INSTITUTION ____

(NOTE: A signed copy of the SOFTWARE RELEASE AGREEMENT must accompany this request.)

Computer System on which the PASCAL Compiler is to run:

Manufacturer ____________ Model ________________ Total Memory ________

Memory available for PASCAL Compiler ______________________________

Operating System with release number (if any) ______________________

Tape Densities available ____

For what purpose do you intend using the compiler?

What other PASCAL Compilers or Interpreters

(a) do you have on your system?

(b) have you used?

Do not send a tape; tapes will be supplied out of the distribution fee ($50, payable to: Department of Computer Science, University of Manitoba).

Name and Address for distribution of the PASCAL Compiler:

____

____

____

____

____

Mail to: PASCAL Distribution Manager,
Department of Computer Science,
University of Manitoba,
WINNIPEG, Manitoba,
Canada R3T 2N2.

The SUNY Stony Brook Pascal compiler is to its credit, very standard. However it is implemented in XPL and is very large (180K bytes). Our request for more information was answered with a 14 page (unfortunately badly faded photocopy) description but with no letter explaining details. An order form was sent and it is reproduced below. Page 7 of the documentation states that it will run under OS/MVT, OS/MFT, and VS/2. It comes with a resident monitor written in BAL.
Two independent parties have reported the unreliability of the compiler. See Steve Bellovin's letter in the OPEN FORUM (18 June 1976). Another site phoned in their complaints.

The Stony Brook PASCAL/360 compiler is being made freely available, and there are no restrictions upon its use. Redistribution is also expressly permitted but no fee is to be charged for redistribution of all or any part of the software or documentation furnished by us.

We must impose a charge of U.S. $175.00 to defray our costs of distribution tape; documentation, maintenance through March, 1977, and distribution of maintenance updates. This charge is not a use fee, nor is any charge being made for development of the compiler.

The distribution tape includes 19 files containing source code and object code for the compiler, an execution monitor that provides an interface to OS/360, some utility programs that will be needed for maintenance, and sufficient documentation to install the system.

The initial distribution will be followed by one copy of the written documentation of the compiler, and by periodic maintenance updates in the form of card decks or minireels of 9-track tape. Maintenance is not subject to any warranty, either explicit or implied, beyond the assurance that the user will receive any updates that we generate through March, 1977.

Professor R.B. Kieburtz

- -

ORDER FORM

TO: PASCAL Compiler Project
Department of Computer Science
SUNY at Stony Brook
Stony Brook, N.Y. 11794

Please send a copy of the PASCAL/360 distribution tape, and put my name on the distribution list to receive further documentation and updates. I agree that you will receive payment of $175.00 from me or my institution.

RECORDING BIT DENSITY:

800 bpi / 1600 bpi (Circle one)

Billing Address, if separate:

____

____

____

NAME: ______________________

INSTITUTION: ______________________

AND ADDRESS: ______________________

____

IBM 1130 (no known implementations)

ICL 1900 (an implementation exists)
2970 (an implementation is planned)

INTEL 8080 (we need more implementation information)

INTERDATA 7/16 (an implementation is underway)
70 (no known implementations)

MICRODATA 800 (no known implementations)

MITSUBISHI MELCOM 7700 (an implementation exists)

MOTOROLA 6800 (we need more implementation information)

NCR CENTURY 100, 200, 300 (no known implementations)

PHILIPS P-1400 (a non-standard implementation exists)

SEL 8600 (an implementation exists)

SIEMENS 150 (an implementation exists)

TELEFUNKEN TR-440 (an implementation exists)

TEXAS INSTRUMENTS TI-ASC (no known implementations)
TI-980A (implementations exist)

UNIVAC 1100 SERIES

Three implementations have been under development. One, by J. Steensgaard-Madsen Datalogisk Institut, Sigurdsgade 41, DK-2200 Copenhagen, Denmark is described in Newsletter #4. There is no distribution information available. Another project is still underway (not complete) by Charles Fischer and Richard LeBlanc at MACC, University of Wisconsin, Madison, WI 53706, (608) 262-7870. Its first release is not expected for a few months yet.

Especially for persons in North America, most encouraging is the news from Mike S. Ball at the Naval Undersea Center in San Diego. On July 16, 1976, he announced that a fully standard Pascal compiler generating relocatable code which can be linked to subprograms written in assembler or FORTRAN. The compiler runs under the EXEC-8 operating system and can be used in Demand mode. Another feature is that "Brinch Hansen style Sequential Pascal" programs are accepted under one compiler option. No performance or resource requirement data is available.

To obtain a copy of the system, write to Michael S. Ball, Code 2522, Naval Undersea Center, San Diego, CA 92132, (714) 225-2365, requesting a copy as a member of USE (*The Univac Users organization*). Include a tape, and if there are any limitations on the format please note them.

This compiler was developed from Pascal-P2, and is documented in a 29 page machine retrievable document entitled: "Pascal 1100". The document is in the form of a supplement to the book Pascal User Manual and Report. Actually the details explained in "Pascal 1100" indicate that this implementation is a powerful tool for writing all kinds of software, especially that for writing new EXEC-8 commands.

Many of the extensions available in the CDC 6000 implementation are present so that portability between these two versions is better than just Standard Pascal.

The compiler accepts the full ASCII character set (strings are packed 4 per 36 bit word - 9 bits per character). Sets have 4 word representations allowing the convenience of defining a set of char.

Minor restrictions from Standard Pascal are as follows: the symbols: entry, processor, and univ are reserved; Sets have at most 144 elements; standard procedures and functions cannot be used a actual parameters to other procedures add functions; it is not possible to construct a file of files.

We are looking forward to more information on Univac 1100 implementations, especially performance data.

XEROX SIGMA 6 (no implementation information)
SIGMA 7 (implementations exist)
SIGMA 9 (no known implementations)

PASCAL USER'S GROUP

USER'S

GROUP

ALL PURPOSE COUPON

****

Clip, photocopy, or reproduce, etc. and mail to: Pascal User's Group
c/o Andy Mickel
University Computer Center
227 Exp Engr
University of Minnesota
Minneapolis, MN 55455

(phone: (612) 376-7290)

/ / Please renew my membership in / enter me as a member of the *PASCAL USER'S GROUP* for the next / current Academic Year ending June 30. I understand that I shall receive all 4 issues of *Pascal Newsletter* for the year. Enclosed please find $4.00.

/ / Please send a copy of *Pascal Newsletter* Number ______. Enclosed please find $1.00 for each.

/ / My new address is printed below. Please use it from now on. I'll enclose an old mailing label if I can find one.

/ / You messed up my address. See below.

/ / Enclosed are some bugs I would like to report to the maintainer of the ____________________________ version of Pascal. Please forward it to the appropriate person so that something can be done about it.

/ / Enclosed please find a contribution (such as what we are doing with Pascal at our computer installation), idea, article, or opinion which I wish to submit for publication in the next issue of *Pascal Newsletter*.

/ / None of the above. ____

____

____

____

Other comments:

From: name ____________________________

address ____________________________

____

____

____

phone ____________________________

date ____________________________

(*Your phone number helps facilitate communication with other PUG members.*)

return to:
University Computer Center
University of Minnesota
227 Experimental Engineering Building
Minneapolis, Minnesota 55455

return postage guaranteed