PASCAL USER'S GROUP

PASCAL NEWSLETTER

NUMBER 6

COMMUNICATIONS ABOUT THE PROGRAMMING LANGUAGE PASCAL BY PASCALERS

NOVEMBER, 1976

TABLE OF CONTENTS

0 POLICY

1 EDITOR's CONTRIBUTION

5 HERE AND THERE WITH PASCAL

News

Conferences

Books

Errata

Back Issues

Membership Roster

33 ARTICLES

"Indexed Files"

- S. Knudsen

"The Need for Hierarchy and Structure in Language Management"

- G. Michael Schneider

"On the Suitability of a Pascal Compiler in an Undergraduate Teaching Environment"

- A. M. Addyman

"Pascal Potpourri"

- Richard J. Cichelli

"The Case for Extending Pascal's I/O"

- Michael Patrick Hagerty

"General Thoughts on Pascal Arising out of Correspondence Between Southampton and Tasmania"

- Arthur Sale

OPEN FORUM FOR MEMBERS

IMPLEMENTATION NOTES

Checklist

Portable Pascals

Compilers and Software Tools

ALL PURPOSE COUPON
POLICY -- PASCAL USER'S GROUP AND PASCAL NEWSLETTER

USER'S GROUP POLICIES

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

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

Membership - is open to anyone: particularly the Pascal user, teacher, maintainer, implementor, distributor, or just plain fan. Institutional memberships, especially libraries, are encouraged. Membership is per academic year ending June 30. Anyone joining for a particular year will receive all 4 quarterly issues of Pascal Newsletter for that year. (In other words, back issues are sent automatically.) First time members receive a receipt for membership; renewers do not to save PUG postage.

Cost of membership per academic year is $4 and may be sent to:
Pascal User's Group/ Andy Mickel/University Computer Center/ University of Minnesota/Minneapolis, MN 55455 USA/ phone: (612) 376-7290

In the United Kingdom, send £2.50 to:
Pascal Users' Group/ Judy Mullins/Mathematics Department/University/SOUTHAMPTON/S09 5NH/United Kingdom/ (telephone 0703-559122 x2387)

NEWSLETTER POLICIES

The Pascal Newsletter 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
#4 available from George Richmond/Computing Center/U of Colorado/Boulder/80309

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, and important events (meetings, publications, 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 Pascal's environment.

Miscellaneous features include bibliographies, questionnaires, and membership lists. Editor's notes are in Pascal style comments (**).

WRITTEN INFORMATION FOR THE Newsletter IS EASIER TO PRINT IF YOU TYPE ALL MATERIAL ½ OR DOUBLE SPACED SO THAT IT IS IN "CAMERA-READY" AND "PHOTO-REDUCIBLE" FORM FOR THE PRINTER. REMEMBER, ALL LETTERS TO US 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, John P. Strait, associate editor. Nov. 10, 1976.
EDITOR'S CONTRIBUTION

PART I - Standards

Wow! It took only one issue of PUG's Pascal Newsletter to bring on an avalanche of "Where do we go from here?"! It was first put clearly in print with a short note in PUGN #3 by George Poonen who noted that various implementations had diverged and that a standard was necessary. Now we have: Tony Addyman, Frank Brewster, Charles Hedrick, and Willett Kenyon (see News in HERE AND THERE); Mike Schneider, Rich Cichelli, and Arthur Sale (see ARTICLES); and Steve Young, Tony Addyman (again), Duke Haiduk, Judy Mullins, Arthur Sale (again), and Tim Bonham (see OPEN FORUM) all discussing the topic of standards. The concern, I believe, is out of our desire to see, I thought once, that some aspects were left undefined. And that other features were omitted with the Revised Report and the Axiomatic Definition. These two concise and elegant (although not perfect — but yet what do you want?) documents were produced by Niklaus Wirth and his associates and coworkers. And I believe that Pascal has merit because it was produced by a single man of the calibre of Niklaus Wirth, who (as evident from his work) profoundly understands programming language design, from linguistics to implementation. This one person could decide what to meld when meeting all of the design goals set out from the start.

I wanted to do what I could to call for adherence to what Niklaus Wirth says it is. And that Andy Mickel can't arbitrarily restrain attempts to change it because 1) Andy fears destruction of the language by attempts to "save" it, or 2) Andy doesn't want them to destroy the essential simplicity of Pascal which is probably its most likely reason for success. They also pointed out that we don't have an officially accepted standard; a "political standard" if you will. Really, when that concept dawned on me it made sense. A major computer manufacturer, when choosing a common language for all its software development, democratically decided to pick the one that most of its programmers wanted to use. With the choice of language X 30%, Pascal version A 25%, Pascal version B 13%, and Pascal version C 27%, language X won by a plurality (and by default) and too bad — as we all can see. If we want Pascal to ultimately and completely succeed, we can't have this!

Now how do we resolve the conflict(s)? Many persons suggest a "PUG Standards Committee", and frankly, although I think committees are inherently evil, I don't see any other choice. The alternative at this point is to lower our expectations, quit striving for excellence, quit "dreaming the impossible dream" of seeing Pascal take over the majority of industrial and academic computing (wiping out Cobol and Fortran within our lifetimes). Then we could say regretfully — "wow, Pascal's nice, but..." as so many of our half-hearted supporters and critics do now.

I feel that: 1) we should continue to debate this topic; 2) a PUG Standards Committee when set up should be small (less than 8 members); 3) its charter be initially agreed on so as to limit its power; 4) within the committee's initial charge


Also among the many ill-conceived suggestions for "improvements" to the language by users, there were some very few that seemed reasonable to dyes - in-the-wool Pascaler. There was no mechanism for sounding these out for worthiness and acceptance, save writing to Niklaus and Urs in Zurich. This has been very frustrating because we didn't know where we were heading. We were told on the one hand, "no more changes." We relaxed and said "fine." Then a revision came along and we felt cheated. We weren't kept informed of what other users had suggested, either.

Rich and Mike have pointed out that Pascal can't continue to be what Niklaus Wirth says it is. And that Andy Mickel can't arbitrarily restrain attempts to change it because 1) Andy fears destruction of the language by attempts to "save" it, or 2) Andy doesn't want them to destroy the essential simplicity of Pascal which is probably its most likely reason for success. They also pointed out that we don't have an officially accepted standard; a "political standard" if you will. Really, when that concept dawned on me it made sense. A major computer manufacturer, when choosing a common language for all its software development, democratically decided to pick the one that most of its programmers wanted to use. With the choice of language X 30%, Pascal version A 25%, Pascal version B 13%, and Pascal version C 27%, language X won by a plurality (and by default) and too bad - as we all can see. If we want Pascal to ultimately and completely succeed, we can't have this!

Now how do we resolve the conflict(s)? Many persons suggest a "PUG Standards Committee", and frankly, although I think committees are inherently evil, I don't see any other choice. The alternative at this point is to lower our expectations, quit striving for excellence, quit "dreaming the impossible dream" of seeing Pascal take over the majority of industrial and academic computing (wiping out Cobol and Fortran within our lifetimes). Then we could say regretfully - "wow, Pascal's nice, but..." as so many of our half-hearted supporters and critics do now.

I feel that: 1) we should continue to debate this topic; 2) a PUG Standards Committee when set up should be small (less than 8 members); 3) its charter be initially agreed on so as to limit its power; 4) within the committee's initial charge

* This brings to mind two acronyms: John Easton's SHAFT or Society to Help Abolish Fortran Teaching, and Mitch Wand's ACS or the American Cobol Society - analogous in meaning to the American Cancer Society.
EDITOR'S CONTRIBUTION

The action should be to get the Revised Report (User Manual and Report, Second Edition, third printing) accepted as an official standard as is (even if only provisionally); 5) later the committee could recommend subsequent actions.

Look up the articles in this issue of PUG Newsletter by Mike and Rich with their excellent analyses of the current situation. Rich bluntly hints that many features are best left to separate software writing tools. In all honesty, I don't see how Arthur Sale can say in his October 22 letter to Judy Mullins, "Of course I agree that standard Pascal must be adhered to" and also say that it is best in specific cases to add features that all Burroughs Algol programmers are used to. Pascal was meant to be a departure from the past. See also the article "Experience from the Standardization of the SIMULA Programming Language", by Jacob Palme, SOFTWARE, Practice and Experience, Vol. 6, No. 3 July-Sept, 1976, pp 405-409. (It seems that each issue of SOFTWARE, Practice and Experience always has some good articles for the practical programmer!)

We are indeed in a unique position in computer science history as people (rather than large organizations) responsibly influencing an influential language.

PART II - Pascal User's Group and Pascal Newsletter

1) PUG has 516 members in 22 countries and 43 states. (We had 317 at last writing.) I'm sorry this newsletter is so late. But this year the November issue will have in it feedback to the September issue.

2) Ms. Judy Mullins and Prof. D. W. Barron of the University of Southampton have done us all a favor by creating a European distribution center for PUG newsletters and a clearing house for PUG memberships in the United Kingdom! Judy was concerned that members in the U.K. would not get fast mail service, while at the same time having to pay a relatively high exchange rate for $4. We in fact had decided to send the first 2 newsletters (#5 & #6) air mail because we could afford it and Pascal needed the shot in the arm. What has transpired between Southampton and Minnesota is no less than 6 letters east to west and 5 letters and a phone call west to east on the subject of cheaper ways to send the newsletters (air freight, etc.) These 11 letters are not reproduced here; they mostly contained calculations and mechanics of mailing.

3) While we are on the subject of finances, I'm happy to report that we're doing just about right. We've been able to afford to send out 250 issues of #4, and do a large mailout requesting implementation information. We still plan to print and mail #7 and #8, so don't worry. The next sheet contains a breakdown:

<table>
<thead>
<tr>
<th>Description</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>516 members @ $4</td>
<td>$2064.00</td>
</tr>
<tr>
<td>8 members not paid yet</td>
<td>32.00</td>
</tr>
<tr>
<td>6 members for 2 years</td>
<td>24.00 extra</td>
</tr>
<tr>
<td>1 member for 5 years</td>
<td>16.00 extra</td>
</tr>
<tr>
<td>ABM + JPS contribution</td>
<td>29.00</td>
</tr>
<tr>
<td><strong>Total Assets</strong></td>
<td><strong>$2101.00</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>postage, mass mailings</td>
<td>$ 52.00</td>
</tr>
<tr>
<td>refunds for overpayment</td>
<td>4.00</td>
</tr>
<tr>
<td>printing and mailing #5</td>
<td>487.10 (700 printed, 368 mailed)</td>
</tr>
<tr>
<td>buying 230 copies of #4</td>
<td>100.00 no bill for mailing yet</td>
</tr>
<tr>
<td>postage for #5 backissues</td>
<td>27.40 so far</td>
</tr>
<tr>
<td>printing newsletter titles</td>
<td>5.60</td>
</tr>
<tr>
<td><strong>Total Expenses</strong></td>
<td><strong>$676.10</strong></td>
</tr>
</tbody>
</table>

Theoretical balance = 2101.00 - 676.10 = 1424.90

Cash on hand                                    | $  77.76 |

PUG UCC Account                                  | $1353.30 |

Actual balance =                                  | $1431.06 |

4) Backissues. See the section in HERE AND THERE. Our offer to send #4 to persons in North America who didn't already get one directly from George Richmond expired on October 2nd. We simply ran out. But we did buy time. And now the problem of trying to include information in #5 that was in #4 is not as acute because #6, #7, and #8 will gradually make up for that. We will be updating nearly all the news which appeared in #4. So for those of you who joined after October 2nd and still want the newsletter #4, order one from George Richmond.

5) I apologize for announcing our policy of: "all the news that fits, we print" in the same issue that we put the policy into practice. We modelled the policy after SIGPLAN Notices. Feedback to Newsletter #5 has been mostly favorable; the unfavorable comments have been largely unwritten. Some heretofore unwritten comments went like this:

"Your organization could be improved."

"It was fun reading the News section in HERE AND THERE."

"It's good to see the correspondence you had with Zurich."

"It's taken a long time to get my newsletter in the mail."

"The articles you printed weren't so hot."
6) Last issue we tried to plan events so that you would receive the newsletter at the beginning of September. But we didn't come close. Our cutoff date for material was supposed to be July 15, but it lagged to July 31. We began putting the newsletter together July 24. We went to press August 13 (and here's the bad news) 25 days later we got our 700 copies on September 7. We had it all in the mail September 9. In the U.S. we know (so far) that some arrived as late as October 21! This issue will probably arrive by Christmas (no kidding) but we began November 4 to put it together and we are going to press November 15 - much better than last time, except we have a late start. Our cutoff for material for this issue was originally October 1 but lagged to November 5. Issue #7 will probably be smaller as it will go to press probably before we get reaction to this issue. By being smaller, it also won't cost as much to print.

7) Offers to help. In #5, N. Solntseff and W. Richard Stevens offered to help with the User's Group. Now that some things have been established, several tasks are becoming clear. These are:

- managing distribution of software writing tools for Pascal written in standard Pascal
- managing distribution or cataloging of library and applications programs for Pascal written in standard Pascal
- maintaining a bibliography on all publications about Pascal (including articles and books)

Any takers?

8) Two encouraging trends. First, with microprocessor interest spreading (real computer power to the people!) it is important to have a Pascal subset compete with BASIC in 16K. Mark Rustad understands this very well - see his Motorola 6800 description in IMPLEMENTATION NOTES. Mark would like to hear from those persons interested. Second, John and I have been getting lots of inquiries about Pascal and implementations in the form of phone calls and letters - with most of them from persons in industry. Predominate are small software writing firms and minicomputer companies. So next time someone says Pascal is okay, but it's not "real world" tell them that it's happening right now.

9) Thank you to all the people who have sent in information to print - that makes the newsletter. Thanx to John, Tim Bonham, Jim Miner, and Herb Rubenstein for halping put together this issue.

-Andy

November 14, 1976
ANNOUNCEMENT OF A PASCAL USERS' GROUP DISTRIBUTION CENTRE IN THE UNITED KINGDOM

AIMS
1. To expedite distribution of the P.U.G. Newsletter to the U.K. and the rest of Europe, the Near and Middle East and Northern Africa.
2. To collect memberships in P.U.G. from U.K. members avoiding high bank charges on transfers of £ to $.

DISTRIBUTION
1. Central P.U.G. at Minnesota will send the original of the newsletter to Southampton for reprinting.
2. Newsletters will be mailed (second-class postage) from Southampton to members in Europe, the Near and Middle East and Northern Africa.

PASCAL USERS' GROUP MEMBERSHIPS
1. The address for U.K. Region memberships is
   Pascal Users' Group
c/o Judy Mullins
Mathematics Department
The University
SOUTHAMPTON. S09 5NH
(telephone 0703-559122 x2387)
2. Members can pay £2.50 by cheque or postal order to PASCAL USERS' GROUP (UK) at the above address, and will receive a receipt and member certificate directly.
3. Membership forms will be forwarded at short intervals to Minnesota (at least in time to catch the next newsletter); a copy is kept at Southampton.

AVOIDING CONFUSION
1. There is only one membership list and labelling program - Minnesota's.
2. Therefore anyone can join directly by writing to the U.S.A.
3. Using the U.K. Distribution Centre only saves money.
4. No matter how he/she joined, a member with an address in the U.K. will receive newsletters via Southampton.
5. All correspondence other than subscriptions (such as change of address, articles for the newsletter, or questions about compilers) must go direct to Minnesota. If it inadvertently arrives at Southampton it will be sent on by airmail.

**NEWS (ALPHABETICAL BY LAST NAME)**

A. M. Addyman, Department of Computer Science, The University, Manchester M13 9PL
United Kingdom (PUG member): "I would like to join the Pascal Users' Group. Also, I am engaged in an effort to have Pascal standardised by a major standard's organisation, e.g., ANSI or ISO. How may I use your newsletter to contact people who would be interested in this, or alternatively to discover that there is considerable opposition?" (*8/10/76*)

Urs Ammann, Institut fur Informatik, ETH - Zentrum, CH-8092 Zurich, Switzerland (PUG member): "...By the way: What is your philosophy with the letters you received as to their publication in the Newsletter? I was somewhat astonished to see private correspondence in it. While I agree that this kind of information distribution makes editorship most easy, it is my strong opinion that any letter which is not explicitly marked as 'letter to the editor' should not be published in full length, since this clearly exceeds or even contradicts [sic] the purpose of private correspondence.

"Please don't misinterpret this statement! I have nothing against transparency, on the contrary! Any information of general interest you find in your correspondence should be passed on. But you will agree that with some effort from the editor, information can be passed on without letting everybody read private correspondence...." (*9/29/76*)

Diosdado P. Banatao, 3000 Bilbo Drive, San Jose, CA 95121 (PUG member): "I would like to be a member of the Pascal Users Group... My interests are in microprocessors and microcomputers and involved in both hardware and software design...." (*10/19/76*)

Philip N. Bergstresser, 128 Jackson Ave., Madison, AL 35758 (PUG member): "We at TRW Systems are using Pascal on the CDC 7600, CDC 6400 and 11-ASC and claim the Guinness record for program size." (*9/27/76*)

Frank M. Brewster, 4701 Kenmore Ave #1009, Alexandria, VA 22304 (PUG member): "...It's been pointed out that many BASICs are 'non-standard'. I have yet to hear anyone ask, 'Why?'. The answer seems obvious: the language initially didn't have 'legal' provision for many of the users' real problems. The current ANSI BASIC proposal still demonstrates this failing. E.g., the CHR and SEQ(or OR) functions are optional; how can anyone do general work without these functions? So BASICs will continue to be 'non-standard', as people fill in the gaps. If a car were sold without say, steering wheel, no one should complain if a buyer adds a tiller. The point is that if the automotive designer finds steering wheels uninteresting and refuses to specify them as standard equipment, the user has two options (assuming he buys the car in spite of its failings): design his own steering apparatus, or cooperate with others in filling the gap in the 'standard'. If the designer won't see the issue, users will. The letters in the newsletter mention, e.g., array passing and formatted input problems. Apparently Wirth's not concerned. If you and others do nothing, then everybody either abandons Pascal or invents their own wheel (tiller?). But why don't those of you with early and practical experience with the language:-list your complaints & problems, ranked, one list per man. (Maybe in a newsletter section, 'What's wrong with Pascal?'?) -compare notes for similarities -see if you can agree on solutions to any of these -implement experimental changes; test till working -promulgate as PUG-US 'extensions'

"The last item is the tackiest one. "A camel is a horse designed by a committee." Standards - the real ones, in actual use - are designed by those who are actually working in the field, in the course of their work. So if you and other of the few presently experienced Pascal users won't add to or alter Wirth's pronouncements, don't be surprised at the later irreverence of others.

"All of you (me too someday) may owe a lot to Wirth. His opinions deserve respect and attention. But if he's to be treated as God, and his language as the ten commandments, how can Pascal be improved? The time to 'standardize' is not now, but after user problems have been faced frankly, and solutions found...." (*10/29/76*)


"All of the above systems have their drawbacks. My interest is in a better portable environment for use on µCPU applications. I am very happy with the CDC 6600 version 3.4 at Purdue University; however, achieving the same degree of performance on a mini-computer has been and will continue to be a challenge Mr. Stephen C. Schwarm, a coworker, is in the process of starting a DECUS SIG PASCAL for PDP users of Pascal." (*9/13/76*)

---

**HERE AND THERE WITH PASCAL**

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

(NEWS FROM MEMBERS, CONFERENCES, NEW BOOKS, APPLICATIONS PROGRAMS, ETC.)

K. Frankowski, Computer Science Dept., 114 Lind Hall, University of Minnesota, Minneapolis, MN 55455 (PUG member): "If one wants to have formatted reads, simply read several integers (if they are run together) as one integer and use div and/or mod to extract the values desired." (*10/15/76*)

Dennis Graham, Amdahl Corp., 1250 E. Arques Ave., Sunnyvale, CA 94086 (PUG member): "I am interested in running Pascal on an Amdahl-470 V/6 system and am contacting the University of Manitoba about their compiler." (*10/26/76*)

David J. Griffiths, Academic Computer Centre, Tyler Hall, University of Rhode Island, West Warwick, RI 02881 (PUG member): "I am investigating the possibility of implementing Pascal on our IBM360-370. Concurrent Pascal would be the ideal, since we wish to implement more advanced operating systems, however, we are prepared to settle for less." (*10/3/76*)

Donald E. Grimes, 90 Sylvia Street, Arlington, MA 02174 (PUG member): "Congratulations on a timely Newsletter #5, and thanks for your efforts in establishing PUG." (*10/8/76*)

Charles Hedrick, 183 Commerce West, University of Illinois, Urbana, IL 61801 (PUG member): "When considering operational definitions of portability maybe it is useful to distinguish among versions that are:

- machine independent
- semi-machine dependent and concepts universal
- machine dependent and site independent

"The last choice may not be so bad for Pascal to shoot for." (*10/15/76*)

Carl Henry, Computer Center, Carleton College, Northfield, MN 55057 (PUG member): "...we are using...the University of Illinois version (Mickunas, et al) and runs under DOS V4 on an 11/20, (very little use has been made of it so far.).

"A brief description of our facilities: 6 PDP-8s - homebrewed version of TSS/8 and OS/8; PDP 11/20 - DOS, RT-11, RSTS V4; PDP-11/40 - RSTS V6, UNIX V6." (*10/15/76*)

Mark Hersoy, 323 Village Drive Apt. 534, East Lansing, MI 48823 (PUG member): "Currently modifying P2 version of Janus compiler for readability, fixing bugs, expanding subset processed, and improving portability.

"All work being done on Michigan State University's CDC 6500." (*10/4/76*)

Brian W. Johnson, 1525 Westlake, Plano, TX 75075 (PUG member): "I am particularly interested in \(\mu\) processor versions. We have it on the PDP-10 and PDP-11 at UT Dallas." (*11/4/76*)

Willett Kempton, 2512 San Gabriel St., Austin, TX 78705 (PUG member): "Thanks for the newsletters. ...I was delighted to see that dynamic array parameters will be implemented in CDC Pascal; this is clearly an extremely important feature, and I would urge that it become a feature of the standard language, rather than an extension available in some implementations (and not others).

"Keep after those implementers to accept standard Pascal programs!!" (*10/27/76*)

C. A. Lang, Cambridge University Press, Pitt Building, Trumpington St., Cambridge CB2 1RP, United Kingdom (PUG member): "We are interested in publishing books concerned with Pascal." (*10/26/76*)

Michael Lutz, School of Computer Science and Technology, Rochester Institute of Technology, Rochester, NY 14623 (PUG member): "...I would also appreciate any information you might have on Pascal implementation for the Xerox (Honeywell) Sigma 5 - 9 and PDP 11 computers. We have both a Sigma 9 and a PDP 11/T34 (with 48K words of memory) here at R.I.T., and we are interested in obtaining Pascal for use in our courses...." (*10/27/76*)

John Montague, Los Alamos Scientific Laboratory, Group C 11 - Mail Stop 296, Los Alamos, NM 87545 (PUG member): "...we plan to bring up Pascal on the CRAY-1, probably using the P-code compiler to bootstrap." (*10/18/76*)

Judy Mullins, Computer Studies Group, Department of Mathematics, The University, Southampton S09 5NH, United Kingdom (PUG member): "...Pascal is alive and happy in Southampton. One hundred nineteen-year-olds are pushing in programs by the hundreds ... and doing amazingly well. I do believe it is the language that is so friendly that increases their interest and output...

"...I was wondering if it would be appropriate to have a section of PUGN for exchange of course ideas, examples etc. This would have to be firmly controlled space-wise, but could prove very informative especially for universities who have Pascal but don't teach it yet. Later on a survey on the use of Pascal in teaching would be of great interest. Addyman's survey showed Pascal is growing and therefore its growth should be monitored every year.

"Another thought was for book reviews. Pascal primers are beginning to proliferate and we have strong views on the ones we've seen. Once again, to have the right effect this section would need to be controlled, and I'm not sure that we want to start issuing PUG marks of approval or anything like that. However, reviews in normal journals are only opinions and it does seem fitting for opinions of Pascalers on Pascal books to be in the Pascal Newsletter." (*11/3/76*)
Fred Powell, Computer Center, Mary Baldwin College, Staunton, VA 24401 (PUG member): "We have Pascal P2 and are interested in implementing Pascal on an IBM 1130 and possibly a System 3. Other possibilities include investigating data bases and disk access techniques with Pascal." (*9/24/76*)

Douglas H. Quebbeman, 2235 Lombardy Drive, Jeffersonville, IN 47130, (PUG member): "...Having seen the article in the June '76 Random Bits (Indiana University's Computing Center Newsletter) on the Pascal User's Group, I decided to join. I am a student and part-time operator-programming consultant and have only recently begun using Pascal, but I am quite enthused about its flexibility (especially considering my wrestling bouts with Fortran) and hope to become more proficient in it. So, thanks (for forming the User Group) and I hope to hear from you soon." (*9/24/76*)

Peter A. Rigbee, Code 5494, Naval Research Laboratory, Washington, DC 20375 (PUG member): "...My connection with Pascal is that my group is trying to get Per Brinch Hansen's SOLO operating system to run on a PDP 11/40, and once this is done, will be using Pascal as a primary systems programming language...." (*8/25/76*)

Sérgio de Mello Schneider, Departamento de Computação, Unv. Federal de S. Carlos, C.P. 384, 13560 S. Carlos SP, Brazil (PUG member): "We have a HP 2100A at our installation (32K words core, 2 disks, 1 tape, DDS) and we are looking for a Pascal compiler. There is no way we can produce one in the next 3 years. Could you help us?" (*10/21/76*)

Stephen C. Schwarm, E. I. du Pont de Nemours Co., 101 Beech St., Wilmington, DE 19898 (PUG member): "I am chairman of DECUS SIG Pascal and I will be glad to help with distribution any systems on DEC PDP-11's." (*9/27/76*)

Dave Tarabar, Data General Corp., Field Engineering, 235 Old Connecticut Path, Framingham, MA 01701 (PUG member): "I was very pleased to receive and read the first PUG Pascal Newsletter. It was full of interesting information. The newsletter will be very useful in publishing the correspondence with Zurich and other implementors and your summary of all known implementations was great. Keep up the good work." (*10/18/76*)

William P. Taylor, L-315, University of California, PO Box 808, Livermore, CA 94550 (PUG member): "I am interested in obtaining information about implementations of Pascal on 16-bit mini-computers. I am especially interested in implementations for the PDP-11 as we will be getting one soon. Also, some of my fellow employees here at Lawrence Livermore Laboratory wish to implement a structured programming language like Pascal for system development on a new mini-computer." (*10/3/76*)

---

**APPLICATIONS PROGRAMS**

**STANFORD UNIVERSITY**

**STANFORD LINEAR ACCELERATOR CENTER**

*Request for Programs*

**Pascal Users:**

I am presently doing research on Pascal to determine how various parts of the language are used and what patterns of execution occur in actual programs. This will be similar to a study done by Knuth on Fortran (1).

In this regard I am interested in obtaining a sample of programs from a wide range of users in hopes that the results of this study might be representative of the actual use of Pascal.

If you have or know of programs which can be lent to this effort, I would very much like to hear from you. I can be contacted by mail at:

Wall Drop 88  
Stanford Linear Accelerator Center  
P. O. Box 4349  
Stanford, CA 94305

or by phone at

(415) 854-3900 X2202.

John Fanning


[*Note: John also enclosed a note which said: "With regards to the enclosed request, I expect that the mentioned study will complete sometime in the first quarter of 1977. I would be most happy at that time to provide a summary of the results for the Newsletter if you are interested - ... Does there exist some formal mechanism for Pascal program interchange (between users), and, if so, who is running it and how can I contact them?"*]
Third Annual Computer Studies Symposium

University of Southampton

AIMS

Few languages since FORTRAN have had the same run-away success as Niklaus Wirth's PASCAL, which shows signs of becoming a de facto standard for Computer Science teaching and research, as well as pointing the way to a new generation of sparse, simple languages.

The purpose of this symposium is to explore "what's going on" in PASCAL at the present time. Leading authorities will describe new implementations and applications in systems programming, research and education. The symposium will end with an open discussion about the future of PASCAL.

In the tradition of the Southampton symposium, speakers will be allowed ample time for their presentations, together with provision for a discussion at the end of each lecture. Attendance will be kept to 100 and it may be necessary to limit applications from each institution. Applicants are expected to have a working knowledge of PASCAL.

Full preprints of the Proceedings will be available on registration; the Proceedings will subsequently be published in book form.
BOOKS AND ARTICLES

(*There has been no news of new books on other computer science literature. Apologies for the void in this section in this issue.*)

A. M. Addyman and N. R. Addyman, "Which Language?", Computer Bulletin, June, 1976, pp 31-33. [An article which surveys the language used at various institutions teaching computer science]

(*Last issue there were a couple mistakes in the list of books; these are corrected below.*)


Introduction to Problem Solving and Programming with Pascal, by G. Michael Schneider, Steven W. Weingart, and David M. Perlman, Wiley, to be published in late 1977. (*A complete soft cover manuscript will be available March 1, 1977 and may be ordered from Michael Schneider, Computer Science Department, 114 Lind Hall, University of Minnesota, Minneapolis, MN 55455. Such copies may be duplicated (once received) for local use.*)

PASCAL User Manual and Report, by Kathleen Jensen and Niklaus Wirth, Springer-Verlag, 1974, 1975, 167 pages, paperbound, $5.90, Second (study) edition. (*This book is selling well; it's in its third printing which now incorporates the errata that appears on the next page as reproduced from Newsletter #4. In Newsletter #5 we printed out of date errata because no one kindly informed us of anything more up to date. So like the implementation notes, we are only as good as what people send us to print. Note that this errata includes the change to the language PASCAL - namely the generalization of the read and write procedures to perform I/O on files of any type, not just text files.*)
Errata to
PASCAL
User Manual
and Report

KEY:
p = page number
l = line number
(Blank lines are ignored and negative line numbers are counted from the bottom)
c = code
(that is:
r = replace
i = insert)

PAGE 10

NOVEMBER, 1976

13 -3 r "if" by "If"
45 -18 r "unsigned constants" by "<constant>
51 16 r "(output)" by "(output):
56 -6 r "fi" by "f{i+1}"; "g{i+1}" by "g{j+1}"; "gi" by "g{j}
63 2 r "k" by "n"; "2" by "n-1"; "j" by "n-2";
69 3 r (n-1) by 2, "n" by 1
86 81 The procedure read can also be used to read from a file f which is not a textfile.
read{f,x)
In this case stands for
begin x := fi; get(f) end

87 81 The procedure write can also be used to write onto a file f which is not a textfile.
write(f,x)
In this case stands for
begin fi := x; put(f) end

117 r (diagram expression) "<" by "<"; "2" by ";"; ";" by "<"
128 -10 r "neither be formal nor non local" by
"not be declared on intermediate level"
170 177: assignment to function identifier not allowed here
179: multidefined record variant
180: X-opt of actual proc/func does not match formal declaration
181: control variable must not be formal
181: constant part of address out of range
205: zero string not allowed
206: integer part of real constant exceeds range
260: too many exit labels
124 -15 r "14" by ";"
124 -14 r "14" by ";"
127 27 r "18,A" by ";.A"
133 3 r "two" by "to"
135 5 r "although" by "although"
135 30 r "substitute" by "substitute"
140 11 r "structure type" by "structured type"
161 -17 r whole line by
"addition to the procedures set and put. The textfiles these"
161 -16 r whole line by
"standard procedures apply to must not necessarily represent"
162 3 r "and of line" by "and of line"
162 -15 r The procedure read can also be used to read from a file f which is not a textfile. read{f,x) is in this case equivalent to
x := fi; get(f).
162 -6 r The procedure write can also be used to write onto a file f which is not a textfile. write(f,x) is in this case equivalent to
fi := x; put(f).
PAST ISSUES OF PASCAL NEWSLETTER

Reproduced below is a complete description of Newsletters 1, 2, 3, and 4. Numbers 1, 2, and 3 are out of print, but they did appear in issues of SIGPLAN Notices, the ACM Special Interest Group on Programming Languages monthly journal. Number 4 is available for $1.00 from George H. Richmond, Computing Center, 3645 Marine Street, University of Colorado, Boulder, CO 80309.

#1 January, 1974 (also SIGPLAN Notices Vol. 9 No. 3 1974 March) 8 pages.
Table of Contents
1 From the Editor
2 Current CDC Pascal Compiler
3 Cost of the CDC Compiler
4 Forthcoming Versions of the CDC Compiler
5 Other Pascal Compilers
6 Modifications to CDC Pascal
7 Other Documentation

#2 May, 1974 (also SIGPLAN Notices Vol. 9 No. 11 1974 November) 18 pages.
Table of Contents
1 From the Editor
2 History of Pascal
3 Pascal for non-CDC machines
4 Pascal 6000-3.4 - N. Wirth
5 Pascal and Portability - N. Wirth

#3 February, 1975 (also SIGPLAN Notices Vol. 11 No. 2 1976 February) 19 pages.
Table of Contents
1 From the Editor
2 Pascal User Manual and Report
3 Pascal Questionnaire Results
4 History of Pascal, Revised - G. Richmond
5 Bibliography
6 Portable Pascal
7 A Generalization of the Read and Write Procedures - N. Wirth
8 Corrections to Pascal 6000 - 3.4
9 Pascal 6000 - 3.4 Interactive Operation
10 Letters to the Editor

#4 July, 1976 (copies may be obtained for $1.00 from George Richmond, the editor, as explained on the previous page) 103 pages.
Table of Contents
0 From the Editor
1 Correspondence
(altogether 36 letters and notices including much implementation information)
81 A New Release of the Pascal-P System - Ch. Jacobi
88 Pascal User's Group
90 Pascal Implementors List
100 Bibliography (Literature about the Programming Language Pascal)
For our mutual benefit in communication, here is the 516 member PUG roster spanning 22 countries and 43 states. It is sorted (intelligently, we think) by zip (mail) codes (U.S. first) and then alphabetically by country. You can see at a glance who is at a well known organization at a well known place or who is in your area (or on your street!). Now, if you need an index by last name, there is one at the end, cross-referencing with zip (mail) code.

For our mutual benefit in communication, here is the 516 member PUG roster spanning 22 countries and 43 states. It is sorted (intelligently, we think) by zip (mail) codes (U.S. first) and then alphabetically by country. You can see at a glance who is at a well known organization at a well known place or who is in your area (or on your street!). Now, if you need an index by last name, there is one at the end, cross-referencing with zip (mail) code.

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>Phone</th>
<th>City</th>
<th>State</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>David Tarabar</td>
<td>Field Engineering, Data General Corporation, 235 Old Connecticut Path, Framingham, MA 01701</td>
<td>(617) 620-1200</td>
<td>Lowell</td>
<td>MA</td>
<td>01852</td>
</tr>
<tr>
<td>Lloyd Dickman</td>
<td>25 Hawthorne Village, Concord, MA 01742</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aeron Sanyer</td>
<td>Dept 330, The Foxboro Company, Foxboro, MA 02035</td>
<td>(617) 543-8750 x2029</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Warren R. Brown</td>
<td>Dept 330, The Foxboro Company, Foxboro, MA 02038</td>
<td>(617) 543-8750 x2025</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Albert S. Brown</td>
<td>PK3-1/M2, Digital Equipment Corporation, 146 Main Street, Maynard, MA 01754</td>
<td>(617) 697-5111 x2391</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ronald F. Brender</td>
<td>ML3-5/820, Digital Equipment Corporation, 146 Main Street, Maynard, MA 01754</td>
<td>(617) 697-5111 x2520</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Victor S. Miller</td>
<td>Dept of Mathematics, Bldg 2, U of Massachusetts, Harbor Campus, Boston, MA 02125</td>
<td>(617) 287-1900 X3170/X3161</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Michael Neuman</td>
<td>Winthrop Publishers, 17 Dunster Street, Cambridge, MA 02138</td>
<td>(617) 686-1750</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N. Amos Gileadi</td>
<td>Applied Systems Group, ML 21-4 E-20, Digital Equipment Corporation, 146 Main Street, Maynard, MA 01754</td>
<td>(617) 697-5111 X4402/X3888/X6472</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R. Sterling Eanes</td>
<td>400 Totten Pond Road, Waltham, MA 02154</td>
<td>(617) 890-6900</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R. Krasin</td>
<td>FIRST DATA CORP, 400 Totten Pond Road, Waltham, MA 02154</td>
<td>(617) 890-6701</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hank Edwards</td>
<td>20 Brackett Road, Framingham, MA 01701</td>
<td>(617) 620-1066 (home)</td>
<td>Framingham</td>
<td>MA</td>
<td>01701</td>
</tr>
<tr>
<td>Michael Neuman</td>
<td>575 Technology Square, Honeywell Information Systems, Cambridge, MA 02139</td>
<td>(617) 491-6300</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gabriel Chang</td>
<td>575 Technology Square, Honeywell Information Systems, Cambridge, MA 02139</td>
<td>(617) 491-6300</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>William F. Shaw</td>
<td>ML3-5/840, Digital Equipment Corporation, 146 Main Street, Maynard, MA 01754</td>
<td>(617) 697-5111</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F. J. Corbato</td>
<td>NE43-514, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139</td>
<td>(617) 253-6001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Donald E. Grimes</td>
<td>90 Sylvia Street, Arlington, MA 02174</td>
<td>(617) 646-4129</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MIKE LEMN
782 WEBSTER HALL
4415 FIFTH AVENUE
PITTSBURGH PA 15213
(412) 624-6454

S. L. GULDEN
DEPT. OF MATH
LEHIGH UNIVERSITY
BETHLEHEM PA 18015
(215) 691-7000 X341

RICHARD J. CICHELLI
901 WHITTIER DRIVE
ALLENTOWN PA 18103
(215) 797-9690

C. E. BRIDGE
ENGINEERING DEVELOPMENT LAB
E. L. DU PONT DE NEMOURS AND CO.
101 BEECH STREET
WILMINGTON DE 19898
(302) 774-1731

GARY LINDSTROM
COMPUTER SCIENCE DEPT.
U OF PITTSBURGH
PITTSBURGH PA 15260
(412) 624-6455

V. LALITA RAO
506 W. THIRD STREET APT. 4
BETHLEHEM PA 18015
(215) 865-6448

CHESTER J. SALWACH
2124 DIAMOND STREET
SELLERSVILLE PA 18960
(215) 723-8301

STEPHEN C. SCHWARTZ
E. L. DU PONT DE NEMOURS CO.
101 BEECH ST.
WILMINGTON DE 19898
(302) 774-1669

MARY LOU SOFFA
COMPUTER SCI. DEPT.
335 ALUMNI HALL
UNIVERSITY OF PITTSBURGH
PITTSBURGH PA 15260
(412) 624-6454

RAMON TAN
P.O. BOX 2
BETHLEHEM PA 18016
(215) 866-7195

ROBERT KEZELL
UNIVERSITY COMPUTER ACTIVITY
TEMPLE UNIVERSITY
PHILADELPHIA PA 19122
(215) 787-8527

MIKE FRAME
FIRST DATA CORP.
2011 EYE ST., NW
WASHINGTON DC 20006
(202) 872-0580

HOWARD E. TOMPKINS
COMPUTER SCIENCE DEPT.
INDIANA UNIVERSITY OF PA
INDIANA PA 15701
(415) 357-2524

STEPHEN TITCOMB
1111 NORTH BLVD.
BETHLEHEM PA 18017

FRANK RYSIOKI
COMPUTER ACTIVITY
TEMPLE UNIVERSITY
BROAD AND MONTGOMERY
PHILADELPHIA PA 19122

TERRY P. MEMLIN
SCIENTIFIC RESEARCH UNIT - DPSA
NATIONAL INSTITUTE OF DENTAL HEALTH
BETHESDA MD 20014

ATTENTION: RUTH DROZIN
FREAS-ROKE COMPUTER CENTER
BUCKNELL UNIVERSITY
LEWISBURG PA 17837
(717) 524-1436

RANCE J. DELONG
MORAVIAN COLLEGE
BETHLEHEM PA 18018

JOHN T. LYNCH
BURROUGHS CORP.
P.O. BOX 517
PAOLI PA 19301

WAYNE RASBAND
BLDG 35 ROOM 2A-03
NATIONAL INSTITUTES OF HEALTH
BETHESDA MD 20014
(301) 496-4957

DANIEL C. HYDE
COMPUTER SCIENCE PROGRAM
BUCKNELL UNIVERSITY
LEWISBURG PA 17837
(717) 524-1392

MARILYN HOFFMAN
531 W. UNION BLVD.
BETHLEHEM PA 18018
(215) 865-6937

E. L. ROWE
BURROUGHS CORP.
P.O. BOX 517
PAOLI PA 19301
(215) 648-2218

DAVID A. GOMBERG
DEPT. OF MATH. STAT. AND COMP. SCI.
AMERICAN UNIVERSITY
MASSACHUSETTS & NEBRASKA AYES.
WASHINGTON DC 20016
(202) 686-2393

JOHN W. ADAMS
DEPT. OF I.E.
19 PACKARD LAB
LEHIGH UNIV.
BETHLEHEM PA 18015

JOHN A. WEAVER
2112 PENNSYLVANIA AVE. F-6
BETHLEHEM PA 18018
(215) 867-1085

JOHN D. EISENBERG
COMPUTING CENTRE
SMITH HALL
U. OF DELAWARE
NEWARK DE 19711
(302) 738-8441 X57 (OFFICE)
(302) 453-9059 (HOME)

ARTHUR A. BROWN
1101 NEW HAMPSHIRE AVE., NW APT.1002
WASHINGTON DC 20037
(202) 785-0716

JOHN F. MADDEN
COMPUTER CENTER
1111 NORTH BLVD.
BETHLEHEM PA 18017

JOHN D. EISENBERG
COMPUTING CENTRE
SMITH HALL
U. OF DELAWARE
NEWARK DE 19711
(302) 738-8441

PETER A. RIGSBEE
CODC 5494
NAVAL RESEARCH LABORATORY
WASHINGTON DC 20375
(202) 767-3181

DAVE ENGLANDER
302 SUMMIT STREET
BETHLEHEM PA 18015
(215) 865-9027

JOSEPH A. MEZZAROBA
E. GREENVILLE PA 18041
(215) 691-7000 (OFFICE)
(215) 679-9900 (HOME)

WILLIAM Q. GRAHAM
COMPUTING CENTER
U. OF DELAWARE
13 SMITH HALL
NEWARK DE 19711
(302) 738-8441

PETER A. RIGSBEE
CODC 5494
NAVAL RESEARCH LABORATORY
WASHINGTON DC 20375
(202) 767-3181
THOMAS A. KEENAN
DIVISION OF MATHEMATICAL AND COMPUTER
NATIONAL SCIENCE FOUNDATION
WASHINGTON DC 20550
(202) 632-7346

CAROL A. ODIN
SOFTWARE TECHNIQUE INC.
100 POMMANDER WALK
ALEXANDRIA VA 22314
(703) 549-0646

GREGORY J. WINTERHALTER
DIGITAL COMMUNICATIONS
135 TECHNOLOGY PARK
NORCROSS GA 30092
(404) 445-1400

ATTN: DIRECTOR
NORTHEAST REGIONAL DATA CENTER
203 SSB
U OF FLORIDA
GAINESVILLE FL 32611
(904) 392-2061

SHMUEL PELEG
COMPUTER SCIENCE CENTER
U OF MARYLAND
COLLEGE PARK MD 20742

ATTN: J. F. MCMINTYRE - LIBRARIAN
COMPUTING CENTER
GILMER HALL
U OF VIRGINIA
CHARLOTTESVILLE VA 22903
(804) 924-3731

ATTENTION: JERRY M. SEGERS
OFFICE OF COMPUTING SERVICES
GEORGIA INSTITUTE OF TECHNOLOGY
ATLANTA GA 30332
(404) 894-4676

ATTN: LIBRARIAN
CIRCA
411 WEIL
U OF FLORIDA
GAINESVILLE FL 32611
(904) 392-0907

BEN SHNEIDERMAN
DEPT. OF INFO. SYS. MGMT.
U OF MARYLAND
COLLEGE PARK MD 20742

STEPHEN J. HARTLEY
113 FERNCILFF DR.
WILLIAMSBURG VA 23185
(804) 229-0337 (HOME)
(804) 827-2897 (WORK)

GERALD N. CEDERQUIST
ELECTRONICS RESEARCH BLOG
FES STL/ASD
GEORGIA INSTITUTE OF TECHNOLOGY
ATLANTA GA 30332
(404) 894-3417

FRED L. SCOTT
BROWARD COMMUNITY COLLEGE
3501 DAVIE ROAD
FT. LAUDERDALE FL 33314
(305) 581-8700

JACOB C. Y. WU
SYSTEM SCIENCES DIVISION
COMPUTER SCIENCES CORPORATION
8720 COLESVILLE ROAD
SILVER SPRING MD 20910
(301) 580-1545 X276

ANN D. DAVIES
UNIVERSITY COMPUTER CENTER
VIRGINIA COMMONWEALTH UNIVERSITY
1015 FLOYD AVE.
RICHMOND VA 23284
(804) 770-6339

PHILIP H. ENSLOW JR.
SCHOOL OF INFO. AND COMP. SCI.
GEORGIA TECH
ATLANTA GA 30332
(404) 894-3192

DONALD B. CROUCH
DEPT. OF COMPUTER SCIENCE
U. OF ALABAMA
P.O. BOX 6916
UNIVERSITY AL 35486
(205) 348-6363

RAINER F. MCCOOL
MCCOOL COMPUTER SERVICES
9557 LONG LOOK LAKE
COLUMBIA MD 21045

DAVID A. HUGH
529 HELM DRIVE
NEWPORT NEWS VA 23602
(804) 874-3387

JAMES N. FARMER
OFFICE OF COMPUTING SERVICES
GEORGIA TECH
225 NORTH AVE., NW
ATLANTA GA 30332
(404) 894-4660

PHILIP N. BERGSTESSERT
128 JACKSON AVE.
MADISON AL 35758
(205) 837-2400

JOE C. ROBERTS
613 MARKET STREET
POCOMOKE CITY MD 21851
(804) 824-3411 X641

J. C. KNIGHT
LUNDEY RESEARCH CENTER
M/S 125A
NASA
HAMPTON VA 23665

JOHN J. GODA JR.
SCHOOL OF INFORMATION AND COMPUTER SCI.
GEORGIA TECH
ATLANTA GA 30332
(404) 894-3131

ATTENTION: DAVID MADISON
ADVANCED SOFTWARE TECHNOLOGY DEPT.,
TEXAS INSTRUMENTS INC.
304 WYNN DRIVE
HUNTSVILLE AL 35806
(205) 837-7510

ANDREW S. POUCHIK
11623 CHARTER OAKS #202
RESTON VA 22090

FRED W. POWELL
COMPUTER CENTER
MARY BALDWIN COLLEGE
STANTON VA 22401
(703) 885-0811

JOHN P. WEST
OFFICE OF COMPUTING SERVICES
GEORGIA TECH
225 NORTH AVE., NW
ATLANTA GA 30332
(404) 894-4676

SAMUEL T. BAKER
1310 STONEWALL BLVD.
MURFREESBORO TN 37130
(615) 896-3362 (HOME)
(615) 741-3531 (OFFICE)

FRANK BREWSTER
4701 KEHINH AVE #1009
ALEXANDRIA VA 22304
(703) 370-6645

STEVEN M. BELLLOVIN
DEPT. OF COMP. SCI.
U OF NORTH CAROLINA
CHAPEL HILL NC 27514
(919) 933-5698

T. P. BAKER
DEPT. OF MATH
225 LOVE BUILDING
FLORIDA STATE U
TALLAHASSEE FL 32304
(904) 444-2580

ATTENTION: GORDON R. SHERMAN
COMPUTER CENTER
200 STOKELY MGMT. CENTER
U. OF TENNESSEE
KNOXVILLE TN 37916
<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>Phone Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charles Hedrick</td>
<td>183 Commerce West, U of Illinois, Urbana, IL 61801</td>
<td>(217) 333-4515, (217) 356-8425</td>
</tr>
<tr>
<td>M. D. Mickunas</td>
<td>297 DCL, U of Illinois, Urbana, IL 61801</td>
<td>(217) 333-6351</td>
</tr>
<tr>
<td>Carlton Mills</td>
<td>Mills International, 203 North Gregory, Urbana, IL 61801</td>
<td>(217) 328-2436 (Home), (217) 333-6971</td>
</tr>
<tr>
<td>Attm: Receiving Clerk</td>
<td>CERL - SOC, U.S. Army, P.O. Box 4005, Champaign, IL 61820</td>
<td>(217) 352-6511</td>
</tr>
<tr>
<td>Fred P. Baker</td>
<td>302 E. Gregory, Champaign, IL 61820</td>
<td>(217) 344-7511</td>
</tr>
<tr>
<td>Avrum Itzkowitz</td>
<td>505 E. Clark Apt. 22, Champaign, IL 61820</td>
<td>(217) 359-9644 (Home), (217) 352-6511 (Work)</td>
</tr>
<tr>
<td>Daniel M. O'Brien</td>
<td>601 E. Clark #10, Champaign, IL 61820</td>
<td>(217) 356-2718</td>
</tr>
<tr>
<td>Jack Thompson</td>
<td>Mid. Illinois Computer Co-op, Cottonwood Road, Edwardsville, IL 62025</td>
<td>(618) 288-7268</td>
</tr>
<tr>
<td>Dennis S. Andrews</td>
<td>Information Processing, Southern Illinois University, Carbondale, IL 62901</td>
<td>(618) 942-6063</td>
</tr>
<tr>
<td>Larry D. Landis</td>
<td>United Computing Systems, 2523 Washington, Kansas City, MO 64108</td>
<td>(816) 221-9700</td>
</tr>
<tr>
<td>Jeffery M. Razafsky</td>
<td>United Computing Systems Inc., 500 W. 26th Street, Kansas City, MO 64108</td>
<td>(816) 341-4491</td>
</tr>
<tr>
<td>Howard D. Pyron</td>
<td>Math &amp; C.Sci., U of Missouri, Rolla, Rolla MO 65401</td>
<td>(314) 341-4491</td>
</tr>
<tr>
<td>Steven S. McKnight</td>
<td>Department of Computer Science, U of Kansas, Lawrence, KS 66045</td>
<td>(913) 222-2222</td>
</tr>
<tr>
<td>Lynne J. Baldwin</td>
<td>Dept. of Math/Comp. Sci., U of Nebraska, Box 688, Omaha, NE 68101</td>
<td>(402) 554-2836</td>
</tr>
<tr>
<td>Terry E. Weymouth</td>
<td>C/O J.W. Weymouth, 6110 Meadowbrook Lane, Lincoln, NE 68510</td>
<td>(402) 472-3480</td>
</tr>
<tr>
<td>Sharad C. Seth</td>
<td>Dept. of Comp. Sci., U of Nebraska, Lincoln, NE 68588</td>
<td>(402) 472-3480</td>
</tr>
<tr>
<td>Terry M. Walker</td>
<td>Computer Science Dept., U of Southern Illinois, P.O. Box 4-4330, Lafayet, LA 70504</td>
<td>(318) 234-7349</td>
</tr>
<tr>
<td>John Nunally</td>
<td>Harding College, Searcy, AR 72143</td>
<td>(501) 268-6161 X257</td>
</tr>
<tr>
<td>Richard V. Andree</td>
<td>Math Dept., U of Oklahoma, Norman, OK 73019</td>
<td>(405) 325-3410</td>
</tr>
<tr>
<td>Ralph Hostenstine</td>
<td>2515 Crestmont #208, Norman, OK 73069</td>
<td></td>
</tr>
<tr>
<td>Stephen A. Pitts</td>
<td>305 East Jarman Drive, Midwest City, OK 73110</td>
<td>(405) 732-4060</td>
</tr>
<tr>
<td>Gilbert J. Hansen</td>
<td>3104 Bonniebrook Drive, Plano, TX 75075</td>
<td>(214) 243-7837</td>
</tr>
<tr>
<td>Brian W. Johnson</td>
<td>1525 Westlake, Plano, TX 75075</td>
<td>(214) 690-2885</td>
</tr>
<tr>
<td>Dennis J. Frailey</td>
<td>Comp. Sci. Dept., Southern Methodist Univ., Dallas, TX 75222</td>
<td></td>
</tr>
<tr>
<td>W. J. Meyers</td>
<td>4-2145, The Timbers, 13447 N. Central Expwy, Dallas, TX 75243</td>
<td>(214) 231-4869</td>
</tr>
<tr>
<td>Name</td>
<td>Address</td>
<td>Phone</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>MARK OVERGAARD</td>
<td>APIS DEPT. C-014, U OF CALIFORNIA - SAN DIEGO LA JOLLA CA 92093</td>
<td>(714) 452-2728</td>
</tr>
<tr>
<td>ATTENTION: EDWARD E. BALDOVICH</td>
<td>SCIENCE AND TECHNOLOGY DIVISION GENERAL RESEARCH CORP. 5383 HOLLISTER AVE., SANTA BARBARA CA 93105 (805) 964-7724</td>
<td></td>
</tr>
<tr>
<td>DENNIS GRAHAM</td>
<td>ANDAIL CORP. 1250 E. ARQUES AVE., SUNNYVALE CA 94086 (408) 735-4602</td>
<td></td>
</tr>
<tr>
<td>JOHN C. BEATTY</td>
<td>LAWRENCE LIVERMORE LAB BOX 808 LIVERMORE CA 94550 (415) 447-1100 X3114</td>
<td></td>
</tr>
<tr>
<td>MICHAEL S. BALL</td>
<td>CODE 2, NAVAL UNDERSEA CENTER SAN DIEGO CA 92132</td>
<td></td>
</tr>
<tr>
<td>ATTENTION: COMPUTER SCIENCES INSTITUTE U OF CALIFORNIA RIVERSIDE CA 92507</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROBERT ALAN DOLAN</td>
<td>SPEECH COMMUNICATIONS RESEARCH LAB 800A MIRAMONTE DRIVE SANTA BARBARA CA 93109 (805) 965-3011</td>
<td></td>
</tr>
<tr>
<td>M. H. MACDOUGALL</td>
<td>ANDAIL CORP. 1250 EAST ARQUES AVE. SUNNYVALE CA 94086 (408) 735-4856</td>
<td></td>
</tr>
<tr>
<td>WILLIAM P. TAYLOR</td>
<td>UNIVERSITY OF CALIFORNIA P.O. BOX 808 LIVERMORE CA 94550 (415) 455-6729</td>
<td></td>
</tr>
<tr>
<td>NEIL W. WEBRE</td>
<td>DEPT. OF COMP. SCI. AND STAT. CALIF. POLY. STATE UNIV. SAN LUIS OBIS CA 93401 (805) 546-2996</td>
<td></td>
</tr>
<tr>
<td>BRYAN L. HIGGINS</td>
<td>SCIENCE APPLICATIONS INC. 8201 CAPWELL DRIVE OAKLAND CA 94621 (415) 562-9163</td>
<td></td>
</tr>
<tr>
<td>JOHN M. GRAM</td>
<td>COMPUTING FACILITY U OF CALIFORNIA IRVINE CA 92717 (714) 833-6844</td>
<td></td>
</tr>
<tr>
<td>JAMES L. BEUG</td>
<td>DEPT. OF COMP. SCI. CALIFORNIA POLYTECHNIC STATE U SAN LUIS OBIS CA 93403 (805) 546-1255</td>
<td></td>
</tr>
<tr>
<td>NIKLAUS WIRTH</td>
<td>XEROX RESEARCH CENTER 3353 COYOTE HILL ROAD PALO ALTO CA 94304</td>
<td></td>
</tr>
<tr>
<td>JEFFREY BARTH</td>
<td>COMP. SCI. DIVISION 573 EVANS HALL U OF CALIFORNIA BERKELEY CA 94720 (415) 642-4948</td>
<td></td>
</tr>
<tr>
<td>JOHN F. HUERAS</td>
<td>DEPT. OF INFORMATION AND COMPUTER SCI U OF CALIFORNIA IRVINE CA 92717</td>
<td></td>
</tr>
<tr>
<td>JON F. HUERAS</td>
<td>DEPT. OF INFORMATION AND COMPUTER SCI U OF CALIFORNIA IRVINE CA 92717</td>
<td></td>
</tr>
<tr>
<td>GARY BABDOCK</td>
<td>110-E RICHMOND ROAD CHINA LAKE CA 93555 (714) 446-6733</td>
<td></td>
</tr>
<tr>
<td>PAUL HEDKEL</td>
<td>INTERACTIVE SYSTEMS CONSULTANTS P.O. BOX 2345 PALO ALTO CA 94305 (415) 965-0237</td>
<td></td>
</tr>
<tr>
<td>BLAND EWING</td>
<td>DEPT. OF ENTOMOLOGY 157 GIANNINI HALL U OF CALIFORNIA BERKELEY CA 94720 (415) 642-6660</td>
<td></td>
</tr>
<tr>
<td>WILLIAM L. COOPER</td>
<td>ORG 4400 INTERSTATE ELECTRONICS 707 E. VERMONT ANAHEIM CA 92805 (714) 772-2011</td>
<td></td>
</tr>
<tr>
<td>APRIL MILLER CONVERSE</td>
<td>SEISMIC ENGINEERING BRANCH U.S.G.S. 345 MIDDLEFIELD ROAD MENLO PARK CA 94025</td>
<td></td>
</tr>
<tr>
<td>DAVID C. LUOKHAM</td>
<td>COMP. SCI. DEPT. A.J. LABORATORY STANFORD UNIVERSITY STANFORD CA 94305 (415) 497-4971</td>
<td></td>
</tr>
<tr>
<td>SUSAN L. GRAHAM</td>
<td>COMP. SCI. DIVISION-EECS 511 EVANS HALL U OF CALIFORNIA BERKELEY CA 94720</td>
<td></td>
</tr>
<tr>
<td>G. CARRICK</td>
<td>MS970 FOUR-PHASE SYSTEMS INC. 19333 VALCO PARKWAY CUPERTINO CA 90104 (408) 255-0900 X281</td>
<td></td>
</tr>
<tr>
<td>WILLIAM L. COOPER</td>
<td>ORG 4400 INTERSTATE ELECTRONICS 707 E. VERMONT ANAHEIM CA 92805 (714) 772-2011</td>
<td></td>
</tr>
<tr>
<td>DAVID C. LUOKHAM</td>
<td>COMP. SCI. DEPT. A.J. LABORATORY STANFORD UNIVERSITY STANFORD CA 94305 (415) 497-4971</td>
<td></td>
</tr>
<tr>
<td>SUSAN L. GRAHAM</td>
<td>COMP. SCI. DIVISION-EECS 511 EVANS HALL U OF CALIFORNIA BERKELEY CA 94720</td>
<td></td>
</tr>
<tr>
<td>G. CARRICK</td>
<td>MS970 FOUR-PHASE SYSTEMS INC. 19333 VALCO PARKWAY CUPERTINO CA 90104 (408) 255-0900 X281</td>
<td></td>
</tr>
<tr>
<td>DAVID W. SIEGTZ</td>
<td>5421 WILLOWICK CIR. ANAHEIM CA 92807 (714) 772-2811</td>
<td></td>
</tr>
<tr>
<td>GLENN T. EDENS</td>
<td>XACONICS DIV. XEROX 350 POTRERO AVENUE SUNNYVALE CA 94086 (408) 738-4800 (XACONICS) (415) 494-4464 (XEROX/PARC)</td>
<td></td>
</tr>
<tr>
<td>BRIAN SIGMUND</td>
<td>P.O. BOX 1751 FREMONT CA 94538</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Address</td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------</td>
<td></td>
</tr>
<tr>
<td>Edward R. Friedman</td>
<td>10012</td>
<td></td>
</tr>
<tr>
<td>Gerhard Friesland</td>
<td>2 Germany</td>
<td></td>
</tr>
<tr>
<td>John Fung</td>
<td>55414</td>
<td></td>
</tr>
<tr>
<td>Edward F. Gehring</td>
<td>47907</td>
<td></td>
</tr>
<tr>
<td>W. Morven Gentleman</td>
<td>NZL 301 Canada</td>
<td></td>
</tr>
<tr>
<td>A. J. Gerber</td>
<td>2006 Australia</td>
<td></td>
</tr>
<tr>
<td>David W. Giedt</td>
<td>92607</td>
<td></td>
</tr>
<tr>
<td>N. Amo Gileadi</td>
<td>01754</td>
<td></td>
</tr>
<tr>
<td>Ed Glaser</td>
<td>54020</td>
<td></td>
</tr>
<tr>
<td>John J. Goda Jr.</td>
<td>50352</td>
<td></td>
</tr>
<tr>
<td>Helmut Golde</td>
<td>98195</td>
<td></td>
</tr>
<tr>
<td>David A. Gomberg</td>
<td>20016</td>
<td></td>
</tr>
<tr>
<td>J. Goodson</td>
<td>SO9 SKH United Kingdom</td>
<td></td>
</tr>
<tr>
<td>Gerhard Goos</td>
<td>D-7500 Germany</td>
<td></td>
</tr>
<tr>
<td>Dennis Graham</td>
<td>94086</td>
<td></td>
</tr>
<tr>
<td>Susan L. Graham</td>
<td>94720</td>
<td></td>
</tr>
<tr>
<td>William G. Graham</td>
<td>19711</td>
<td></td>
</tr>
<tr>
<td>John M. Gram</td>
<td>22717</td>
<td></td>
</tr>
<tr>
<td>Kristina Greacen</td>
<td>55455</td>
<td></td>
</tr>
<tr>
<td>Mike Green</td>
<td>78284</td>
<td></td>
</tr>
<tr>
<td>R. Greiner</td>
<td>95014</td>
<td></td>
</tr>
<tr>
<td>David J. Griffiths</td>
<td>02881</td>
<td></td>
</tr>
<tr>
<td>Donald E. Grimes</td>
<td>02174</td>
<td></td>
</tr>
<tr>
<td>Martin L. Griss</td>
<td>84112</td>
<td></td>
</tr>
<tr>
<td>Dale H. Griss</td>
<td>80523</td>
<td></td>
</tr>
<tr>
<td>William Grosky</td>
<td>48202</td>
<td></td>
</tr>
<tr>
<td>Jonathon R. Gross</td>
<td>55240</td>
<td></td>
</tr>
<tr>
<td>Roger Gulkibian</td>
<td>61801</td>
<td></td>
</tr>
<tr>
<td>S. L. Gulden</td>
<td>18015</td>
<td></td>
</tr>
<tr>
<td>Michael Hagerthy</td>
<td>02174</td>
<td></td>
</tr>
<tr>
<td>Harry P. Haiduk</td>
<td>79015</td>
<td></td>
</tr>
<tr>
<td>Joel M. Halpren</td>
<td>55455</td>
<td></td>
</tr>
<tr>
<td>Ronald J. Ham</td>
<td>01754</td>
<td></td>
</tr>
<tr>
<td>Donn Hanani</td>
<td>55409</td>
<td></td>
</tr>
<tr>
<td>Michael Z. Hanani</td>
<td>ISRAEL</td>
<td></td>
</tr>
<tr>
<td>Gilbert J. Hansen</td>
<td>75075</td>
<td></td>
</tr>
<tr>
<td>Brian Hanson</td>
<td>55455</td>
<td></td>
</tr>
<tr>
<td>Stephen J. Hartley</td>
<td>23185</td>
<td></td>
</tr>
<tr>
<td>Keith Hauer-Lowe</td>
<td>55417</td>
<td></td>
</tr>
<tr>
<td>Paul Heckel</td>
<td>94305</td>
<td></td>
</tr>
<tr>
<td>H. G. Hendens</td>
<td>48684</td>
<td></td>
</tr>
<tr>
<td>Charles Hendrick</td>
<td>61801</td>
<td></td>
</tr>
<tr>
<td>Dennis Heimbigner</td>
<td>90278</td>
<td></td>
</tr>
<tr>
<td>Juha Heinonen</td>
<td>SF-33103 Finland</td>
<td></td>
</tr>
<tr>
<td>T. S. Heines</td>
<td>44115</td>
<td></td>
</tr>
<tr>
<td>David Helfinmotion</td>
<td>55303</td>
<td></td>
</tr>
<tr>
<td>Carl Henry</td>
<td>55057</td>
<td></td>
</tr>
<tr>
<td>William Henry</td>
<td>10003</td>
<td></td>
</tr>
<tr>
<td>Mark Herzey</td>
<td>49823</td>
<td></td>
</tr>
<tr>
<td>Bryan L. Higgins</td>
<td>94923</td>
<td></td>
</tr>
<tr>
<td>Terjo Nikita</td>
<td>11215 Japan</td>
<td></td>
</tr>
<tr>
<td>W. A. Hinton</td>
<td>51201</td>
<td></td>
</tr>
<tr>
<td>Thea D. Hodge</td>
<td>55455</td>
<td></td>
</tr>
<tr>
<td>Timothy W. Lienke</td>
<td>55057</td>
<td></td>
</tr>
<tr>
<td>Marilyn Hoffman</td>
<td>18018</td>
<td></td>
</tr>
<tr>
<td>H.-J. Hoffmann</td>
<td>D-6100 Germany</td>
<td></td>
</tr>
<tr>
<td>Timothy J. Hoffmann</td>
<td>55455</td>
<td></td>
</tr>
<tr>
<td>David W. Hogan</td>
<td>78791</td>
<td></td>
</tr>
<tr>
<td>William C. Hopkins</td>
<td>14226</td>
<td></td>
</tr>
<tr>
<td>Frank H. Horn</td>
<td>53706</td>
<td></td>
</tr>
<tr>
<td>Frederick A. Hoss</td>
<td>70122</td>
<td></td>
</tr>
<tr>
<td>David A. Hough</td>
<td>23602</td>
<td></td>
</tr>
<tr>
<td>Rosemary Bowbrig</td>
<td>06413</td>
<td></td>
</tr>
<tr>
<td>Ralph Honestine</td>
<td>73069</td>
<td></td>
</tr>
<tr>
<td>Richard Huser</td>
<td>77843</td>
<td></td>
</tr>
<tr>
<td>Jon F. Heera</td>
<td>92717</td>
<td></td>
</tr>
<tr>
<td>Steven L. Huysen</td>
<td>48824</td>
<td></td>
</tr>
<tr>
<td>Daniel C. Heyse</td>
<td>17837</td>
<td></td>
</tr>
<tr>
<td>M. Elizabeth Ibarra</td>
<td>11973</td>
<td></td>
</tr>
<tr>
<td>Avrum Itzkowitz</td>
<td>61820</td>
<td></td>
</tr>
<tr>
<td>Christian Jacobi</td>
<td>CH-8022 Switzerland</td>
<td></td>
</tr>
<tr>
<td>George D. Jelatis</td>
<td>55455</td>
<td></td>
</tr>
<tr>
<td>Mitchell R. Joelson</td>
<td>55455</td>
<td></td>
</tr>
<tr>
<td>Christopher K. Johansen</td>
<td>02166</td>
<td></td>
</tr>
<tr>
<td>Brian W. Johnson</td>
<td>75075</td>
<td></td>
</tr>
<tr>
<td>Robert T. Johnson</td>
<td>87545</td>
<td></td>
</tr>
<tr>
<td>R. I. Johnson</td>
<td>56202</td>
<td></td>
</tr>
<tr>
<td>R. Warren Johnson</td>
<td>56301</td>
<td></td>
</tr>
<tr>
<td>Warren Johnson</td>
<td>70504</td>
<td></td>
</tr>
<tr>
<td>Ed Katz</td>
<td>70504</td>
<td></td>
</tr>
<tr>
<td>Mark J. Kaufman</td>
<td>92026</td>
<td></td>
</tr>
<tr>
<td>Thomas A. Keenan</td>
<td>20550</td>
<td></td>
</tr>
<tr>
<td>Ginger Kelly</td>
<td>77001</td>
<td></td>
</tr>
<tr>
<td>Willem Kempston</td>
<td>76705</td>
<td></td>
</tr>
<tr>
<td>James A. Kendall</td>
<td>77030</td>
<td></td>
</tr>
<tr>
<td>Leslie R. Kerr</td>
<td>98004</td>
<td></td>
</tr>
<tr>
<td>Robert Kezell</td>
<td>19122</td>
<td></td>
</tr>
<tr>
<td>D. B. Killen</td>
<td>50066 Australia</td>
<td></td>
</tr>
<tr>
<td>D. B. Kilien</td>
<td>70118</td>
<td></td>
</tr>
<tr>
<td>Mike Kimber</td>
<td>MSV 239 Canada</td>
<td></td>
</tr>
<tr>
<td>M. A. Keiluk</td>
<td>84112</td>
<td></td>
</tr>
<tr>
<td>Christina Knight</td>
<td>23665</td>
<td></td>
</tr>
<tr>
<td>Sven Erik Knuudsen</td>
<td>CH-8022 Switzerland</td>
<td></td>
</tr>
<tr>
<td>Carsten Hanne</td>
<td>2000 Germany</td>
<td></td>
</tr>
<tr>
<td>Alan A. Kortedoja</td>
<td>48103</td>
<td></td>
</tr>
<tr>
<td>R. Krasin</td>
<td>02154</td>
<td></td>
</tr>
<tr>
<td>Anthony P. Kyne</td>
<td>3052 Australia</td>
<td></td>
</tr>
<tr>
<td>Jarl. Lasberg</td>
<td>1 Norway</td>
<td></td>
</tr>
<tr>
<td>R. B. Lake</td>
<td>44106</td>
<td></td>
</tr>
<tr>
<td>Dan Lalierte</td>
<td>55455</td>
<td></td>
</tr>
<tr>
<td>Larry D. Landis</td>
<td>64108</td>
<td></td>
</tr>
<tr>
<td>Steve Landry</td>
<td>70504</td>
<td></td>
</tr>
<tr>
<td>David Landskov</td>
<td>70504</td>
<td></td>
</tr>
<tr>
<td>C. A. Lang</td>
<td>CAB 1R United Kingdom</td>
<td></td>
</tr>
<tr>
<td>Charles L. Lawson</td>
<td>91103</td>
<td></td>
</tr>
<tr>
<td>O. Lecarme</td>
<td>06034 France</td>
<td></td>
</tr>
<tr>
<td>Henry F. Ledgard</td>
<td>01002</td>
<td></td>
</tr>
<tr>
<td>Steve Lendenhausen</td>
<td>08904</td>
<td></td>
</tr>
<tr>
<td>Mike Lemon</td>
<td>15213</td>
<td></td>
</tr>
<tr>
<td>L. Richard Lewis</td>
<td>48127</td>
<td></td>
</tr>
<tr>
<td>A. C. W. Leun</td>
<td>THE NETHERLANDS</td>
<td></td>
</tr>
<tr>
<td>P. Liag</td>
<td>95014</td>
<td></td>
</tr>
<tr>
<td>Lawrence A. Liddard</td>
<td>55455</td>
<td></td>
</tr>
<tr>
<td>Dennis R. Hoel</td>
<td>55455</td>
<td></td>
</tr>
<tr>
<td>Shiha Lin</td>
<td>55455</td>
<td></td>
</tr>
<tr>
<td>John E. Lind</td>
<td>55455</td>
<td></td>
</tr>
<tr>
<td>Gary Lindstrom</td>
<td>15260</td>
<td></td>
</tr>
<tr>
<td>Sten Ljungkyss</td>
<td>SWEDE</td>
<td></td>
</tr>
<tr>
<td>Luigi Logrippo</td>
<td>KIN 65S Canada</td>
<td></td>
</tr>
<tr>
<td>Ralph L. London</td>
<td>90291</td>
<td></td>
</tr>
</tbody>
</table>
Indexed Files.

by S. Kaufmann, Institut fur Informatik
E. T. H., Zurich
translated by J. H. Loe, SRPCE
University of Minnesota

In addition to the possibility of dividing sequential files into segments (creating a "segmented file"), it is also possible to construct, read, and modify indexed files. This feature also covers the need for rapid location and modification of segments.

An indexed file may be thought of as a sequential file divided into segments (that is, as a segmented file). Each segment describes a possibly empty series of component-type elements and is a "logical record" in CODASYL terminology. In contrast to segmented files in which a segment can be located through the use of a segment number relative to the previous segment, a segment of an indexed file can be found through the use of a specific segment reference address (a so-called random index), which is returned from the system during the write operation.

Declaration:

<file type> := indexed file of <type>

Example:

type file = indexed file of t
The component type <type> cannot be char: Indexed textfiles are not implemented.

The standard functions EOF and FDSK are defined for sequential files and are likewise valid with indexed files. The standard procedures PUT, GET, and RESET (hence READ and UNITS) are defined as for sequential files. The procedures REWRITE, PUTF, and GET, however, are defined for indexed files as follows:

RESET(f) positions f at the beginning. This allows the first of the segments described by the file to be read.

REWRITE(f) initializes the writing of a new segment at the end of the file f. The new segment is therefore not written at the beginning.

PUTF(f,k) must be called when a new segment is to be closed. The segment index (of type [1..2^31]) corresponding to the segment location is returned in k.

REWRITE(f,k) initializes for rewriting the segment with index k. k must be an index that was returned from PUTF.

PUTF(f) must be called to close a rewrite operation.

If a segment that is longer than the original segment is rewritten, segments following it may be overwritten.

GET(f) is called to initialize the reading of the next segment. An indexed file can therefore be read as a sequential file.

GET(f,k) initializes the reading of a segment with the index k. k must be an index that was returned from a call to PUTF.

Some program examples will clarify the way in which indexed files are used. The basic declarations are:

var
index: array [1..n] of t;
i, k: integer; p: boolean;

- Write an indexed file f with n segments; the reference address of each segment is maintained in an array of indices.

  rewrite(f);
  for i := 1 to n do
    begin
      while p do
        begin (*) fill f indulge put(f) end;
        putseg(f, index[i])
      end
    end;

- Append a new segment. Its index is returned in k.

  rewrite(f);
  while p do
    begin (*) fill f indulge put(f) end;
    putseg(f,k)
  end;

- Sequentially read an indexed file.

  reset(f);
  while not eof(f) do
    begin (*) inspect f indulge set(f) end;
    getset(f)
  end;

- Read a segment with index k.

  getset(f,k);
  while not eof(f) do
    begin (*) inspect f indulge set(f) end;
  end;

- Revert a segment with index k.

  rewrite(f,k);
  while p do
    begin (*) fill indulge put(f) end;
    putseg(f)

*Received 7/22/76*
The Need for Hierarchy and Structure in Language Management

by

G. Michael Schneider
Department of Computer Science
University of Minnesota

I find it quite ironic that so much concern is being paid problems of structure and organization of statements within the PASCAL language but so little to the structure and organization of the management of the language itself. By this I mean that there is currently lacking a formal administrative hierarchy for the handling of questions relating to language standards, specifications, and extensions.

When PASCAL usage was small and consisted of only a few installations, language management could easily be handled by doing whatever you wanted to or by verbal agreements among all parties concerned. Disagreements could be settled by simple exchange of letters, telephone calls or over coffee.

The usage of PASCAL has, I believe, now left this embryonic stage of development. Merely witness the nearly 500 members of the PASCAL Users Group or the dozens of Universities now using it as their primary language.

Yet while the growth of the language has been phenomenal the administration of the language has not. It has remained a loose knit, informal mechanism composed of the creators, users, and maintainers of the language. This is a chaotic way to administer any large system and, worst of all, leaves the language open to chaotic, unstructured growth. It is also frustrating. To whom do we submit suggestions on changes, deletions, improvements, or extensions to the language? To whom do we submit our "beautifully lucid" arguments on what needs to be done? Currently there is no one. This groundswell of frustration was clearly demonstrated by the dozens of letters received by the Newsletter shortly after it began, which described suggested improvements or changes. A few of the suggestions I felt were good, most quite bad. That, however, is not the important point. What is important is that these letter writers had been searching for a vehicle to formally submit proposals and immediately leaped at the Users Group and its publication as just that vehicle. But, the Users Group has absolutely no official status as the arbiter of language standards. The needed administration is still lacking.

What I propose is that we (the User's Group members) begin to discuss what is needed for the proper administration of PASCAL. I initially suggest that we adopt the following proposal:

The PASCAL User's Group nominate and vote on a PASCAL Standards Committee composed of about 10-15 members. This committee must initially perform three functions:

1) Attempt to seek formal recognition for itself with such groups as SIGPLAN, ACM, and ANSI.
2) Certify an official PASCAL standard. While this will probably be the specifications found in the PASCAL report, it should clear up certain "grey areas" (e.g., dispose).
3) Draw up a "constitution" which spells out the role of the committee, its term in office, the philosophy to be used in evaluating proposed standards, and a formal procedure for submitting proposals to the committee.

The committee should now accept and consider suggestions from throughout the user community. It should solicit opinions and arguments on the proposal, evaluate all suggestions in light of the stated philosophy of the PASCAL language and decide to reject it, accept it as a new standard, accept it as a standard extension, or postpone any decision. Major decisions could be put to a vote of the full membership if necessary.

The above proposal omits a great amount of detail that can be worked out by the committee and the membership. It would be presumptuous of me to impose any further my own feelings on how such a standards committee should operate.

What I care about are not really the details anyway. I care about bringing order and structure to the area of language management -- the same goals that PASCAL brought to language design.

(*Received 10/1/76*)
On the suitability of a Pascal Compiler in an undergraduate teaching environment

Before Pascal was adopted by my parent department for teaching purposes, it was necessary to demonstrate that a suitable compiler was available. We have access to a CYBER 72 running a timesharing service under NOS and consequently acquired from Zurich the 6000-3.4 compiler. The performance of the compiler during installation gave rise to a great deal of optimism and a few reservations. The optimism stemmed from the quality of the compiler; the reservations from a few obvious problems caused by the change from SCOPE 3.4 to NOS. These problems were almost entirely caused by the change in the method of use of the compiler not by defects on the code.

The local modifications were all introduced with one purpose in mind - to facilitate the use of Pascal for undergraduate teaching. The modifications can be roughly divided into two categories.

1. Modifications to ease the use of Pascal
   a) the compiler ignores leading line numbers
   b) compilation diagnostics are sensible with the L-option
   c) post-mortem dump output is re-formatted for 70 character wide devices.
   d) dayfile messages were re-ordered so that the fault reason appears on the terminal.
   e) terminal control introduced - a user interrupt will produce a post-mortem dump.
   f) the post-mortem dump gives traceback information in terms of line numbers not core addresses

2. Modifications to improve throughput
   a) A G+ option to automatically enter a correctly compiled program
   b) A N option, which allows the use of blank common for stack + heap.

This reduces the possibility of rollouts which may be caused if a memory request for an increase in field length were made.

Note
To minimise store requirements we wish to run Pascal in REDUCE mode, and under NOS the KRONSUS 'trick' to avoid field length reduction after a relocatable load does not work.

   c) Output buffers which are on-line to a terminal are not flushed by the Pascal run-time system at the end of a run. This is left to the timesharing system. This change was made as the result of a poor benchmark performance.

   To demonstrate that the performance of the compiler was satisfactory a simple benchmark was designed to compare Pascal with Algol 60 and Fortran. It was believed that this benchmark would saturate our system.

The benchmark consisted of running 75 jobs as rapidly as possible from 15 terminals (5 from each terminal) with no other users on the machine. The experiment was repeated 4 times for each language; each time with a different job. The amount of real time elapsed was measured in each case. These figures include the time for terminal I/O which was expected to be small by comparison with the total time.

For the experiment we used:
   a) a Zurich Pascal compiler with local mods (but not 2c above)
   b) an FTNTS compiler
   c) an ALGOL 4 compiler via a procedure file which included utilities for handling the line number problem.
   d) 15 'volunteers', many of whom had never used the system before.

Jobs 1 and 2 involved similar programs. In Job 1 the program was altered to introduce a compilation fault. Job 2 compiles OK and is executed.

Jobs 3 and 4 are related in a similar way. The programs used were genuine student exercises. The 'same' program was used for each language.

Results

<table>
<thead>
<tr>
<th>Job Number</th>
<th>Algol Time in seconds</th>
<th>Fortran Time in seconds</th>
<th>Pascal Time in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>701</td>
<td>245</td>
<td>426</td>
</tr>
<tr>
<td>2</td>
<td>1956</td>
<td>450</td>
<td>190</td>
</tr>
<tr>
<td>3</td>
<td>840</td>
<td>254</td>
<td>*</td>
</tr>
<tr>
<td>4</td>
<td>2967</td>
<td>440</td>
<td>220</td>
</tr>
</tbody>
</table>

*The Pascal experiment was terminated as the performance was unsatisfactory.

Modification 2c was included and experiment repeated. The results were 153, 155, 141 and 201 seconds respectively.

These figures are interesting for two reasons
1. the improvement from 426 to 153 for Job 1
2. the fact that Job 3 took less time than Job 1

Fact 1 can be explained by the introduction of the extra modification which reduced the core <-> disc traffic by

75 x 4 x 250000 words per benchmark

= 61.4 million characters
for the benchmark involving compilation errors.
3.

Fact 2 can be attributed to the human learning process. As the experiment progressed the volunteers were able to type the commands faster because they were more familiar with the system.

In fact the performance of the Pascal compiler is such that the figures presented for the second experiment can only be regarded as a lower bound on the throughput because the terminal I/O now accounts for a significant proportion of the time measured, e.g.

In a faulty compilation benchmark:-

| no. of characters typed by human at a terminal | 65 |
| system at a terminal | 540 |

assuming typing speeds of 3 and 10 chars/sec. this accounts for 76 seconds at every terminal. It seems likely that the system is not being saturated by this benchmark when using Pascal.

Conclusions

1. The performance of Pascal is satisfactory.
2. These figures represent a lower bound on its performance. More accurate figures would have required the use of a greater number of terminals (to saturate the system) and repetition of the experiments.

In the context of the experiments this would have been a waste of time.

A. M. Addyman

(*Received 10/4/76*)

PASCAL Potpourri

by Richard J. Cichelli

Topics for the PASCAL user:

- Direct access files
- "Standard" PASCAL
- Software tools

Direct Access Files for PASCAL

The following is presented as an approach to direct access files in PASCAL. We begin with a discussion of current PASCAL file facilities.

Sequential Files in PASCAL

The PASCAL Revised Report defines only sequential files for PASCAL. Thus, a file is a sequence of zero or more items of the same type. A window, or buffer variable, into the sequence is defined. It is referenced by a buffer pointer. Only one element of a file may be accessed at any time. A predicate EOF (end of file) is defined such that when it is true, the operation WRITE (file item) or PUT (buffer variable) is valid. If EOF is false, READ (file item) or GET (buffer variable) is possible. As a side effect of the READ and WRITE operations, the buffer pointer is moved through the sequence. EOF becomes true during a sequence of READs when no items remain beyond the buffer pointer. EOF remains true after a WRITE. The operations RESET and REWRITEx move the buffer pointer to the beginning of the sequence.

PASCAL sequential files look like tapes.
The Notions of Direct Access Files

Most mass storage based operating systems present files to the user as named data sets (i.e. groups of related items associated under a cataloged name in a directory). The user can request access to a data set by supplying the system with its name. PASCAL sequential files are easily provided in most operating systems. However, the vast majority of third generation operating systems give the user an alternative to the tape-like file organization capabilities of PASCAL files. This alternative allows data items to be accessed directly. That is, if the "file" consists of 1000 items, the user can access the 439th without passing the 433th or rewinding from the 440th.

For direct access files there is no notion of a buffer pointer, and thus there is no EOF, RESET, or REWRITE. Any item may be read or modified "in place". READs and WRITEs can occur in any order.

The nearest notion to this idea that is defined in the PASCAL Report is that of arrays. I propose to extend the PASCAL array concept to provide direct access facilities.*

To accommodate this extension to the language, I propose that the type declaration for arrays be extended from

```
Array
```

to

```
LONG
```

A "long" array will be one which might reside on direct access secondary mass storage.

Consequences of the Notation

Treat the direct access files as arrays requires only relative record I/O capabilities from the operating system. It seems to me that this provides the potential for all direct access facilities at the most fundamental level. It suggests that such advanced notions as "indexed sequential access" will have to be implemented by the programmer or as utilities in terms of the above primitives.*

Implementation Details

Direct access files are used in two basically different ways - as bulk temporary work space and for fast, non-sequential access to permanent data using keys based upon content or relationships.

To serve the first need, the long array can simply be a local array variable. In a virtual memory environment the word "long" might be ignored by the compiler. As far as the programmer is concerned, this type of long array is equivalent to a (possibly) slow access array.

For the second case, long arrays are global to the program. They will be named as formal file parameters in the program heading just as global files are now. Their declarations will be in the variable declarations of the program, or level 0, block.

If a long array file doesn't exist when a program declaring it is executed, one should be created (and should remain upon program termination). If one does exist but is incompatible with

* Andy informs me that someone has already implemented indexed files in PASCAL. From my conversations with him, I do not believe I am duplicating this effort.

** This is quite simple. A relative record file (long array) is used for the records and another is used for the related index which maps from primary key designator to the record number (i.e. long array index). Both arrays might be mapped into the same system file by using record variant parts for array elements.
the program definition, a fatal error should result. Other fatal error conditions will arise if the actual file is sequential or if it is the wrong size (i.e. any type mismatch).

Several programmer notations could be used to guide the compiler in mapping the data items into efficient store. For example, the declaration "PACKED LONG ARRAY" might cause the compiler to try to block the records efficiently. By extending the dollar sign comment notation, the programmer might suggest blocking factors and the number of core resident direct access record areas.

User Interface

Long arrays will be used exactly like in core arrays. Of course, a long array of files or a long array of long arrays can not be permitted. Other than this obvious restriction, a long array will be like any other array. Access notation will be identical.

There is one glaring disadvantage with this scheme, however. When a programmer writes expressions using long array elements, he may invoke significant overhead ... and it will be almost completely invisible to him in the code text. The phrase "long array" just doesn't suggest long moments of computer toil to the programmer. The best sort for a long array may in no way resemble the best for an in-core array.

I suspect, however, the days of slow access rotating magnetic storage are limited. Solid state bulk memory seems destined to overtake disks. Our notation may be more appropriate for the future than the present.

Conclusion

I suggest that the concept of "long arrays" is sufficient for direct access file facilities and is consistent with the design goals of PASCAL in its simplicity and clarity.

Standards and the Language PASCAL

The standards game is one played by programming language users. Those with problems in search of solutions look for new language features. Those with programs searching for customers with problems look to enforce standards. We PASCALers should have a total view of the standards problem. We should realize that no existing programming language standard is a success at its stated goals and that no language has succeeded without a standard. With respect to standards, PASCAL has significant advantages over the popular poorer languages. First, it is defined only in the Revised Report and not in some vendor's implementation. Second, as a language it seems particularly easy to formalize in a humanly understandable fashion. In short, it has a small and regular syntax.

Do We Need a Formally Recognized Standard?

Let us consider the population concerned with PASCAL and PASCAL standardization: language designers, language implementors, program writers, and employers of program writers.

Language Designers

In our case, this is Dr. Niklaus Wirth. He says PASCAL is what he says it is. But, in fact, PASCAL is too important and too widely used to have its scope defined and limited by one man.
We all have a legitimate say in this and we can and should exercise our responsibility. It is important, however, to recognize that the current success of PASCAL is based on its elegant design. We must seek to preserve its simplicity and clarity above all else.

Language Implementors

Dr. Urs Ammann and his group implemented PASCAL in an efficient and robust fashion on the CDC 6600 computers. Because many users confuse a programming language with its particular implementations, Ammann's fine implementations have been the wellspring of PASCAL user growth. Because many implementors have followed Ammann's lead, it is likely that the PASCAL compiler is the most efficient language processor at any shop which has one.

Implementors desire standards to guide their compiler writing. Frequently however, in order to interface or compete with existing languages, they stretch or reinterpret the standards to meet real or imagined user implementation needs. Implementors and compiler maintainers should take great care not to let ad hoc patches to an implementation become de facto changes to the language standard.

Fortunately, no hardware vendor has tried to make PASCAL its own. But we all know that PASCAL will soon be a vendor product. This should not be viewed as auguring potential corruption, but instead as a sign of maturation. We should recognize it as such and provide vendors with an excellent standard to work from. I personally anxiously await the day when Seymour Cray, Gene Amdahl, and Ken Olsen market PASCAL machines.

Users: Managers and Programmers

For obvious reasons, organizations and their representatives (i.e. managers) want standardization in a programming language. Every organization has learned Whitney's lesson about interchangeability. In programming this means adherence to standards.

Programmers have problems to solve. There are things which could be added to PASCAL that might make one programmer's job easier. The problem is to address the entire user community. Frankly, some languages are better than PASCAL for some applications: use COBOL's Report Writer for reports, use SNOBOL for string manipulation, etc. PASCAL can't be all things to all people and still be simple, concise and easily implemented. Remember the PL/I syndrome - multi-million dollar compilers won't solve anyone's problems. There is a revolution coming in computer software as more programmers learn how to do more things simply.

Getting a Recognized Standard

A standards committee should be set up. (I would particularly like to see Dr. Waite as a member.) This committee would represent users and designers first, implementors and vendors second. Its purpose would be to get a document approved by both PUG members and the ANSI-X3J3 committees. International standardization is also desirable. Additionally the committee would be charged with

* It is the naive manager who thinks hardware vendors desire standards. As the current efforts with big languages indicate, all they want to do is exclude the competition.
certifying that a particular implementation conforms to the standard.

Only with formal recognition will PASCAL be adopted by large conservative organizations and selfish vendors.

There is danger is having a committee for this purpose. When COBOL was being designed two committees were formed. Since the problem of business data processing was regarded as so big, one committee was asked to deliver a quick interim report to use to "make do." The second committee was to solve the DP language problem. The first committee report is in - its product was COBOL. We are still waiting for the long range committee's report.

One final word on previous failures. The new FORTRAN is an obvious disaster; the PL/I standard is an abomination. We can do better! We need not be either upward compatible with previous or a vendor's puppet. We can do it right if we get together and try.

Software Tools for PASCAL

PASCAL implementations for new environments are occurring with ever increasing frequency. As PASCAL is used for more and more production programming, it is important that a universal set of ancillary software tools be agreed upon. Some of these tools can be defined in an environment independent way so that when written in standard PASCAL they can become part of a universal PASCAL software development facility. I here propose an initial list. With PUG membership help the list will develop into a working specification and a powerful set of programming tools.

PASCAL Compilers

Currently there exist PASCAL compilers which produce absolute code, relocatable code, macro code (PASCAL-J) and interpreted code (PASCAL-P). Portable versions exist (PASCAL-P and PASCAL-J). Compiler trunks exist. A standard PASCAL subset (PASCAL-S) exists.

For compiler writers there should be a standard PASCAL language test set. This universal set of PASCAL programs would exercise new PASCAL compilers and help implementors gain confidence in the correctness of their compilers.

An interactive interpreter should be developed. This system would provide interactive symbolic run time debugging facilities: breakpoints, interactive dumps, etc. It should be easy to do better than PL/I's Checkout compiler.

The Lecarme and Bochmann compiler writing systems are also important tools for any shop engaged in language development.

Source Program Tools

Wirth has written a cross reference program. Perhaps, if the variable names were improved, a standard version of this program could be among the software tools. A formatter or "pretty printer" is essential for producing documentation quality listings. Mike Condict's might be a good starting place.

A code instrumenter is a very important debugging and refining tool. Instrumenters insert statement counters or timers so that reports of relative usage of code can be made. An instrumenter is invaluable in optimizing programs.

A high level macro preprocessor would also be a valuable facility.
Source Libraries

The CDC source library utility program UPDATE is currently used for distribution of the SCOPE versions of PASCAL. It seems to me that a mini-version of UPDATE (with only sequential program libraries) could be implemented in PASCAL. This would help standardize the distribution of PASCAL tools. (Incidentally, CDC's UPDATE is the best source library system I have ever seen. I think its quality should be emulated.)

For truly large systems (50,000+ lines) a source code data base is desirable. Such a system keeps track of which programs access what data and provides for standard file and record descriptions among programs, etc. I understand such a system for PASCAL exists but is a deep dark military secret.

Documentation Preparation

W. Burger implemented part of Waite's PLAP in PASCAL. We need a universal PLAP-like tool to maintain manuals and other documentation in machine readable form. Justification and hyphenation and facilities for producing high quality printing in upper and lower case should exist. PASCAL documentation should be distributed in machine readable form for ease of publication and distribution.

Object Program Facilities

Work is now in progress on programs which load PASCAL absolute binaries. Facilities for overlay processing should be provided. Automated aids which help create effective overlay structures should be provided. A binary decoder is also a useful tool.

Other Programs

An efficient table processor with facilities like COBOL Report Writer would be desirable. Current work on PASCAL data base management systems, mathematical function libraries, and computer aided instruction systems augur the day of increased use of PASCAL in business, engineering, and education. In the area of function libraries (for mathematics or business), facilities should be provided for not only linking in binary modules but also for including source modules.*

Conclusions

Obviously, where environmental conditions permit we should have a universal PASCAL program implementing each software aid. Where the environmental factors prevent this, we should seek to provide a standard user interface to the desired functions.

Conclusion

The ideas presented in this paper are perhaps still ill-formed. They are meant as a starting point for serious discussion. I hope there will be reaction and feedback from PUG members.

*In my opinion, merging programs at the source level is to be preferred to binary level linking. PASCAL compilers are typically faster than linking-loaders.

(*Received 10/12/76*)
THE CASE FOR EXTENDING PASCAL'S I/O

Michael Patrick Hagerty
Abt Associates Inc.

With the introduction and subsequent increase in popularity of PASCAL, a number of papers concerning the language, its features and deficiencies, have appeared in various journals and newsletters. Champions of the language have extolled the virtues of its structure and unambiguous grammar using both example and theory as justification of its usefulness. PASCAL critics, on the other hand, have questioned the claim of the proponents that PASCAL will replace FORTRAN, pointing to the inadequacies of the language in several areas. Wirth (1974) defends the absence of certain "favorite features" as necessary to avoid inefficient programming solutions or reliance upon features which are contrary to the aim of clarity and reliability. When the features being debated refer to the flexible input of large amounts of data, the critics hold the stronger hand, and with much justification.

As a user of PASCAL in an environment where large files of data are the rule rather than the exception, I find the argument that PASCAL's native input facility is sufficient to be without merit. Much of the data analyzed at AAI is produced by the Bureau of the Census or other government agencies and is available only in fixed-format records in multi-file volumes. The absence of a formatted input capability is not merely inconvenient in this instance; it is self-defeating. Several alternatives have been adopted as stopgap measures, including the use of FORTRAN subroutines to handle all read operations. However, it is obvious that if PASCAL is to become one of the more common languages, it must possess an I/O capability which is useful to those who process large amounts of data as well as the compiler writers.

To gain insight into what is required, it is first necessary to examine the deficiencies in PASCAL from the data analyst's point of view. The following list represents a minimal set of those deficiencies:

- PASCAL I/O is asymmetric in that no READ operation exists which is the inverse of the formatted WRITE.
- PASCAL I/O is further asymmetric in that certain types (ALFA and BOOLEAN) may be written, but cannot be input using the native READ procedure.
- Although the most powerful facet of PASCAL is its structuring facility, there exists no simple, direct method of transmitting RECORDS to and from formatted textfiles.
- PASCAL requires the inefficient use of data storage media by not allowing the user to maintain his data in multi-file volumes. Only data between the portion immediately addressed upon RESET and the first EOF can be examined.

As the major portion of data collected in both the business and research communities is stored as formatted files of records more or less card-image size, the absence of a feature which will allow the direct read of specific columns rather than freefield is an extreme shortcoming. The choice of formatted files was not made only for convenience, although the availability of this feature in other languages did encourage its use. It does require space, disk or tape, to store large amounts of data, and the requirement that each variable be separated from its neighbors by a blank(s), gobbles up more space, and therefore costs more.

If it were only the very large data bases which were formatted, an argument could be advanced for special, custom-tailored I/O for these applications. This position loses ground when considered in light of most applications packages which allow both form of input. AAI is a very heavy user of the SPSS and other statistical packages. Within the past year, the freefield input facility of SPSS has been exercised not twice; once to test that it worked correctly; and once again on a problem with only ten cases. With any survey of over 10-20 observations, it is also much more economical (and accurate) to have the data collected in fixed format without blank delimiters.

In the present situation, each user community is left on its own to develop and implement as part of their library, a formatting reader which meets their own needs. The upshot of this, as clearly described by Eisenberg (1976), is that PASCAL will become another BASIC in the area of I/O. As most users are aware, BASIC programs from one system have a very low probability of running under another system as each manufacture, or vendor, chooses to implement I/O in a slightly different manner. The computing world can well do without this form of chaos.

Turning to the second area of concern, we find that there exist certain types of variables in PASCAL which possess a unique property: although we may print them out to see what value they are assigned, it is not possible to read them in directly. ALFA and BOOLEAN are examples of these types, however other implementations may be busy installing additional varieties. In languages which preceeded PASCAL, an unclad rule existed for I/O, "If you can write it out, you can read it back in."

The third mentioned deficiency is directly related to the first two. It is incongruous that a language as well structured as PASCAL would fall into the trap of requiring the user to transmit the elements of his well-structured records element by element. This deficiency is magnified by the lack of a defined formatting facility for input, the existence of the "special" types, and the absence of a formatting tool which would tie the size of the individual elements to the order within the record itself. FORTRAN has FORMATs; COBOL and RPG have PICTURES; and PASCAL has nothing comparable. It has been rumored that one implementation of PASCAL uses FORTRAN FORMATs, although this hardly seems
to be the optimum solution to the problem. What is needed is a fresh
look into what it means to tie a format specification, even if it im-
plies freefield, to a given element of a structure. Unlike FORTRAN,
it should be capable of diagnosing at compile time attempts to read
or write integers with decimal points and other such errors.

The fourth deficiency is representative of the attempt to stay clear of
defining what constitutes the implementation of files within the context
of a system. By requiring that all data sets processed by PASCAL con-
sist of one file, the interface between various systems is kept simple.
Unfortunately, this requires the user to either keep one file on each
tape or disk library, or copy off a single file from the multi-file
volume to satisfy PASCAL. Keeping partially filled tapes and/or copying
desired files does require added expense.

The remainder of this paper will be devoted to several recommendations
which, if adopted, will remedy most of the problems in the area of I/O.
The form of the recommendations will be to first present what the
construct will look like, followed by an explanation of how it is to be
implemented. Each of the constructs will be based within the scope
of the following declarations:

```
CONST NCPW = (* Number of Characters Per Word - ALFA TYPE *)
VAR A: ALFA;
B: BOOLEAN;
I: INTEGER;
T: TEXT;
F: FILE OF arbitrary type;
N: INTEGER; (* number of characters read or written *)
D: INTEGER; (* number of places to the right of decimal *)
```

READ (T, I:N) will convert N characters beginning with the current
position of the file T to INTEGER and store the result in I.
Leading blanks are to be treated as zeros, but trailing or imbedded
blanks represent an error and should be diagnosed as such. The
number may be signed or unsigned.

READ (T, R:N:D) will convert N characters beginning with the current
position of the file T to REAL retaining D characters as the
fraction. Blanks before the whole number and following the fraction
are to be considered zeros. Imbedded non-digits are errors. The
only exception is that a decimal may be punched in the data which
would override the D specification.

READ (T, B) will read a BOOLEAN variable B freefield from the file T.
TRUE and FALSE will be the allowed character patterns.

READ (T, A:N) will store the following N characters from the file T
which, if adopted, will remedy most of the problems in the area of
I/O.

REAL (T, B:N) will be expected to find the characters TRUE or FALSE
within the next N character on the file T. Where N is less than
5, only the first N characters will be matched.

READ (T, A) will store a left-justified ALFA of at most NCPW characters
within the next character on the file T and continue with characters until
the number of characters is equal to NCPW, a blank character is en-
countered, or EOLN(T)=TRUE. While spanning leading blanks, EOLN
will be ignored. EOLN will be cleared on return from the proce-
dure if it terminated the transfer of characters.

READ (T, A:N) will store the following N characters from the file T
left-justified in the variable A with blank padding out to NCPW.
No more than NCPW characters may be transferred. EOLN will, as
an installation parameter, cause a normal termination with added
blanks out to NCPW.

OPENR (F) will cause the system to RESET a file without rewinding it.
This allows the user to position the file before executing the
PASCAL program.

OPENR (F) is the non-rewinding version of REWRITE.

PUTEOF (F) instructs PASCAL to write the output buffer, followed by
an EOF, and invoke OPENR beyond the EOF. PUTCLOSE is the tool for
creating multi-file volumes, a PUTSEG for EOFs.

GETEOF (F) will cause an End-of-File to be read (skipped), with the con-
current resetting of EOF(F) to FALSE. OPENR(F) will then be invoked
to open the file past the EOF. If two contiguous EOFs are detected,
this will imply the end of the volume, and EOF(F) will be reset to
TRUE upon return from GETEOF(F). If EOF(F) is not initially TRUE,
data will be skipped until the EOF is encountered, and the normal
processing of GETEOF will continue.

GETEOF (F, N) instructs the system to skip N EOF marks on file F. When
the N parameter is specified, GETEOF is analogous to GETSEG(F,N),
with file instead of segment marks. GETEOF(F) is equivalent to
GETEOF(F,N).

The above eleven proposed extensions to the language are directed to
overcoming three of the four mentioned deficiencies. The code necessary
to implement these features is simple and readily installed in any com-
plete implementation of PASCAL. (As an example, although trivial, the
code for reading ALFAs is included as an Appendix). The observant
reader will notice that no attempt has been made to provide a workable
solution for the third problem: formatted and freefield I/O for structures.
It is conceivable that a mechanism can be devised which is compatible with the syntax, grammar and structure of PASCAL, and will allow the separate assignment to different elements in a structure of specific formats. After giving the matter much thought, I am at a loss to produce a new and uniquely appropriate scheme. It appears at first blush that some hybrid of PICTURES and FORMATS might work.

For want of an adequate solution, we at AAI have adopted a strategy which we know to be flawed. It is unacceptable as a solution to the problem of records and textfiles because it simply ignores the existence of textfiles altogether. By specifying a single word format descriptor for each element in a record, it is a simple matter to have a procedure decode a whole record quite efficiently. By building arrays of

\[
\text{TYPE FORMAT = PACKED RECORD}
\]

\[
\begin{align*}
\text{FTYPE:} & \text{[ALFA, BOOLEAN, INTEGER, REAL]:} \\
\text{DSIZE:} & 0..777B; \text{(* DECIMAL PLACES *)} \\
\text{NSIZE:} & 0..777B; \text{(* NUMBER OF CHARACTERS *)} \\
\text{STBIT:} & 1..777777B; \text{(* STARTING BIT IN WORD *)} \\
\text{NSWORD:} & 1..777777B; \text{(* NUMBER OF WORD IN RECORD *)}
\end{align*}
\]

of the same size as the records to be read, filling in the five elements with appropriate descriptors, and passing that array, along with a segment of text already read, the text can be converted.

The procedure we use has four parameters: a vector of characters; the FORMAT array; the resultant decoded RECORD of data; and an integer which specifies the number of elements to decode. For the sake of efficiency, the routine was coded in the assembly language of our machine. The only trick is to manage to get into PASCAL a whole segment of text to decode. This is accomplished by judguing the I/O buffer allocated by PASCAL and allocating a array on top of that buffer. Data is then read into that buffer, and decoded directly out.

CDC, as well as most other manufacturers, provides a powerful read facility which will initiate a read of a specified number of words (or bytes), and read until either the list is satisfied, or the record on the input device is depleted. It is this feature which I would propose to be an implementation dependent feature.

READBUF (F, X, N) will initiate the reading of N items of X (which is an array with at least N elements) from file F. This operation merely initiates the read, but does not guarantee its completion.

WRITEBUF (F, X, N) is the inverse of READBUF and initiates the write of N elements of the array X. Once again, completion is not guaranteed.

COMPLETE (F) forces the system to complete any pending I/O operation on file F, entering a recall state if necessary.

BUFSIZE (F) is a function which will return an integer representing the number of elements actually transferred on the last operation on the file F. It is clear that BUFSIZE should not be used until COMPLETE has forced the end of the operation.

These additional, implementation dependent, features allow the machine to optimize its I/O operations and give the user the opportunity to overlap some independent processing while the machine attends to the task of moving data.

The sophisticated user should recognize that while the suggestions made in this paper apply most directly to the sequential access of large amounts of data (the issue of greatest importance to AAI), no attempt has been made to come to grips with the host of other access methods. Files of every variety, indexed, keyed, and hashed, exist; the need now is to propose the manner in which PASCAL will address them. Given a language with data structures as powerful as PASCAL, no effort should be spared to provide an equally powerful set of I/O operations. The challenge is to devise an implementation independent scheme for doing it.

REFERENCES:


PROCEDURE RDA (VAR F: TEXT; VAR A: ALFA); CONST NCPW = 1U (* NUMBER OF CHARACTERS PER WORD *) VAR I: INTEGER; CHBUF: ARRAY [1..NCPW] OF CHAR; BEGIN IF EOF(F) THEN BEGIN MESSAGE("* TRIED TO READ PAST EOLN/EOf()") HALT END; WHILE (F++ = 2) AND NOT EOF(F) DO GET(F); (* GET LEADING BLANKS *) IF NOT EOF(F) THEN (* COLLECT CHARACTERS FOR ALFA *) BEGIN I := 0; REPEAT I := I + 1; CHBUF[I] := F; IF I = NC THEN EDIT; CHBUF[I] := F; UNiL (F++ = 1) OR (I = NCPW) OR EOF(F); IF EOF(F) THEN BEGIN MESSAGE("* CLEAR EOLN FLAG IF SET") BEGIN I := I + 1; CHBUF[I] := F END; END ELSE (* RDA *) BEGIN (* RDX *) (* READ ENCG COLUMN ALFA IN FIXED FORMAT, HAGERTY *) PROCEDURE RDX (VAR F: TEXT; VAR A: ALFA; VC: INTEGER); CONST NCPW = 1U (* NUMBER OF CHARACTERS IN A WORD *) VAR I: INTEGER; CHBUF: ARRAY [1..NCPW] OF CHAR; BEGIN IF EOF(F) THEN BEGIN MESSAGE("* TRIED TO READ PAST EOLN/EOf()") HALT END; IF NCPW < NC THEN BEGIN MESSAGE("* ALFA FIELD WIDTH ERROR") HALT END; IF NOT EOF(F) THEN (* COLLECT ENCG CHARACTERS *) BEGIN I := 0; REPEAT I := I + 1; CHBUF[I] := F; IF EOF(F) THEN EDIT; UNiL (I = NC) OR EOF(F); WHILE I = NCPW DO (* FILL REMAINDER OF WORD WITH BLANKS *) BEGIN END; END ELSE (* RDA *) END; END; END; END; END; /* END OF */

**GENERAL THOUGHTS ON PASCAL**

**REPRINTED OUT OF**

**CORRESPONDENCE BETWEEN SOUTHAMPTON AND TARANIA**

Arthur Sale
University of Tasmania

**MIXED LANGUAGES**

Here is the focus of the survival of PASCAL. If it is possible for programmers to access and use the vast library of FORTRAN mathematical routines that have been developed, then there is hope that the scientific community might be encouraged to transfer their skills and effort into PASCAL from FORTRAN. A tremendous benefit, and greatly to be desired. If this is not possible (to mix languages) then even this slim hope must fade away. The inertia and ecological success of FORTRAN is too great for any naive competitor to survive and thrive.

I note many PASCAL compilers produce assembly code for their machine. Shades of IBM5401! Still, the approach does allow mixed languages at little cost, but some attention must then be paid to (i) efficiency, and (ii) ease of use of the joint system.

On the Burroughs B6700, the problem is much more significant as no assembly language exists. (For those who marvel at this, know that no computing centre director would want one for the security risk it would be (B6700 integrity relies on software), and know that the structure of the executable code file would require an elaborate assembler to specify all that is necessary..) Burroughs Algol is the lowest level of the B6700. Consequently achievement of mixed languages requires either compilation into Algol (disregarded!) or the construction of structured code-files that are quite incredibly complex by the standards of monolithic machines. The binder in fact has to be able to re-arrange code (especially in the outermost block) for own and pre-defined objects; associate names; check parameter compatibility; and all this for ALGOL/FORTRAN/PL/1/COBOL—at least.

Nevertheless, even in this case, the achievement of mixed-language programming must be attempted. Even more must this be the case in simpler situations; the only exceptions being mono-language systems such as Brinch-Hansens. And these are not addressed to the same purpose as viable general-utility compilers.
PORTABILITY

It is important to realize that standardization is not a good in itself; the present benefits to computer science of standardization are:

1. Transferability of skills between compilers,
2. Portability of programs written in standard languages, and
3. Exchange and development of compatible compilers is made more easy.

It must be realized that the semantics of the language is quite as important as the syntax in realizing objective (2), and in this regard there are any number of difficulties in standard Pascal.

First of these is perhaps character set. Except for those computers that persist with 6-bit characters, the EBCDIC and ASCII character sets must be regarded as the de facto standards of the industry. All Pascal compilers should have the capability of working in either (preferably both) of these two character sets. The implications are that no-one is justified in inventing a new character set, nor in allowing any other character set to be used unless it is a firm commitment by the computer/operating system; and that even in DDC and other 6-bit machines an effort should be made to provide ASCII and EBCDIC characters. I can think of no more common portability trap than that of ignoring character collating order. Pascal-P was caught.

Another, not so obvious, is the practice of allowing any-length identifiers, but permitting only the first few characters to be significant ignoring the tail. If a program is compiled on a computer with true any-length identifiers (e.g. 86700), then it is quite on the cards that it will be compiled incorrectly by some other computer without warning, as if the names are TEMPATTOPKIN, TEMPATTOPKFLUE. No, identifiers must be true any-length, or fixed up to a length n with at the very least a mandatory warning or better an error thereafter.

There are many more portability traps which act so as to limit the real portability of programs written on one computer to quite a long way below that which is achievable at the present state of knowledge. They require more attention in the case of Pascal. I find it infuriating to receive a program that the author claims is "portable" only to find that he or she is obviously not aware of the most obvious requirements for portability.

INCLUSION OF SOURCE TEXT

The 86700 compiler has a feature (as with all Burroughs compilers) for including source text from a named file in the compilation. The included text may be, but is not usually, listed. It may include further included files, up to nesting depth defined by the number of file buffers reserved (in the Pascal compiler: a depth of 6). In fact this is used as a structuring aid in the 86700 Pascal compiler source, which consist of some 20-30 files (some with alternatives) linked into a tree-like structure by inclusion references. The facility is also useful for including library routines (in Pascal source of course) as in special i/o routines, mathematical routines, graph-plot routines, etc. It may postpone the need for the use of relocatable binary or linked versions of Pascal programs...

The 86700 construct is a compiler option: it appears on a single line which begins with a $, and has the following syntax:

\[
\text{\$ INCLUDE filename \text{ sequencenumberrange}}
\]

Examples:

\[
\$ INCLUDE PACKAGE1
\$ INCLUDE ARTHUR/STANDARD/TYPES 30000-60000
\]

Quite clearly, such a construct in Pascal could also be embedded in the compiler option Wirth has implemented, if only it were not so restrictive in syntax. I would commend such a facility to all Pascal compiler-writers, preferably adhering to the above syntax. The default should be to omit listing the included text.
Files are PASCAL's biggest problem. In a modern context, PASCAL's files are an anachronism. PASCAL should have access to random-access files as well as sequential; should be able to communicate with terminal devices as well as card readers and line printers; should be able to at least specify file attributes; and should have a record-oriented I/O subsystem.

On top of this files do not fall into the same class as other VAR objects, since for the most part their life (extent) is not limited to the extent of the program or a procedure thereof. They may be already existing (in which case the declaration is more of the nature of a specification), or may live on past the program's death (and often have to conform to external requirements such as line-length).

Therefore all normal VAR operations on files should be carefully avoided. The assignment and comparisons on files should be regarded as absolutely meaningless, as should possibilities such as arrays of files, or records containing files, and other similar nonsense.

The B6700 implementation will include attribute lists in the FILE type declaration (whether in the TYPE or VAR part) according to the following syntax:

\[ \text{FILE} \rightarrow (\rightarrow \text{machinespecificattributelist})\] OF \(\rightarrow\)

Examples:

VAR

\[\begin{align*}
\text{INPUT} & \quad \text{(KIND=READER)} \quad \text{OF} \quad \text{PACKED ARRAY OF CHAR;} \\
\text{CODE} & \quad \text{(KIND=PACK,TITLE=EXECUTABLE/CODE/TEST,UNIT=WORD,}\ \\
& \quad \text{DIRECTORY=30,BLOCKSIZE=300,HWUSE=OUT)} \\
& \quad \text{OF} \quad \text{SECTORRECORDTYPE;}
\end{align*}\]

The attribute list probably has to be machine-specific at the present state of operating systems. The declared size of the record of the file is checked for compatibility with any specified attribute.

All B6700 files are automatically allowed to be accessed sequentially or randomly, so this question does not arise specifically. Environment variables and attribute changes can be done via calls on the operating system.

STANDARDS

Adherence to the PASCAL standard, interpreting this to mean the language defined in the PASCAL Report, and the axiomatic definitions thereof, must be a very high priority of any PASCAL compiler writer/maintainer. There are nevertheless several sticky problems which face any person in these categories. Let me expose them.

1. There is the question of whether to implement a strict PASCAL or to extend it with various features. Of course there are some areas where PASCAL must be extended (see later), but every extension provides a user temptation and reduces portability of the resulting programs. This works towards implementing PASCAL as she is defined, and that alone.

2. There are the problems associated with undefined parts of PASCAL; for example the elaboration of a CASE where the case expression evaluates to a value not matched by any label. A compiler writer has to do something, and these flaws or loopholes in the definition are left to individual discretion.

3. There are places in PASCAL where the language is seriously deficient; primarily in treatment of files and I/O. Individuality here is necessary but can be seen to be clearly tending towards the Algol and BASIC messes.

4. There are places in the PASCAL definition where the antecedents of the present state show through, in an unwarranted manner. Examples of these are (1) the CDC influence in the curious PROGRAM construct (often unnecessary, and derived from CDC FORTRAN), the use of .or: from Algol in array declarations (when TO is more explicit and less obscure), and the insistence on FORTRAN's archaic control character at the start of a printed line!

Subjectively, up to as far as I am capable of it, it seems to me that PASCAL has in fact been frozen too soon, before the defects have had a proper chance to be eradicated. It is therefore of prime importance that some procedure be adopted whereby evolutionary change in the language can be controlled, otherwise proliferation of dialects is inevitable. Some of the afterthoughts should be recognized as such, and removed from the province of the standard defining document.

(*Received 10/31/76*)
Mr. Andy Mickel
PASCAL Users Group
UGC: 227 Exp. Engr.
University of Minnesota
Minneapolis, MN 55455

Dear Andy:

What happened to that new PASCAL release announced last March?

One of my students has just completed a CAI system in PASCAL. It features a lesson creation language which has excellent expressive ability (it is possible to create structured CAI lessons with it). The compiler for this language is quite efficient. Its output code is interpreted by a small (12.5K words) PASCAL program. The interpreter features good run-time debugging aids which in many ways resemble PASCAL's PMD.

In addition to the lesson compiler and interpreter, there is an object lesson decoder and student monitoring facility. The student monitoring routines record and report individual student scores. Lessons can also be set up which administer tests. The monitoring facility reports a trace of each student's lesson sessions question by question. The system keeps track (on a permanent file) of each student's status and can be used to sequence students thru a series of lessons and tests.

In total, the system is the most versatile and efficient CAI system I have ever seen. To a great extent the viability of the system can be attributed to the fact that it was built with PASCAL. Should we write up the system for the Newsletter?

Sincerely,

Richard J. Cichelli

(*Note: The student mentioned above is PUG member David Englander.*)
Dear Dr. Mickel,

I would like to join the PASCAL users group, and receive your newsletter. I am a doctoral candidate in anthropology, and my uses of PASCAL are for organization and analysis of data on language acquisition and on cognitive variation. Since programming for anthropological applications usually involves handling odball data (when the study is not a simple statistical one), the ability to structure that data has a rather liberating effect on one's programming!

However, I have found the lack of formatted read capability rather annoying at times, and I wonder if other users feel the same way. It would certainly not be difficult to add this capability to the compiler in such a way that it would provide upward compatibility with the Jensen and Wirth (1974) standard, for example:

\[\text{READLN}(f, x1:w1, x2:w2, x3:w3);\]

where \(x_n\) is a variable of type integer, real, or packed array of char. Variables formatted to positions past the end of line should be assigned zero (or blank in the case of strings) or there would be transportability problems between systems which truncate trailing blanks and those which do not. Real numbers such as \(0.5\) should not cause RDN to halt the program (in formatted or freefield reads)! Honestly -- you'd think ETH never writes programs which have to read the output of Fortran programs (i.e. statistical packages).

I'm looking forward to seeing my first PASCAL users group newsletter -- tell me if there are any dues or anything.

Sincerely,

Willett Kempton

OPEN FORUM FOR MEMBERS

(Short, informal correspondence)
Andy: Some time ago you sent to all PUG members a set of corrections for the PASCAL User Manual and Report, Second Edition. In the process of moving from Drake to West Texas State I have buried my copy in stacks of yet unpacked papers. Would you please send me another copy. I would appreciate such. We are in the process of getting PASCAL up and running on our DEC-10. I am working without letup on the process of converting to PASCAL some rather open-minded colleagues. I will use PASCAL as the vehicle language in the data structures course this spring. (Is there any other way to do it?)

Do you know if anyone has yet to build a really interactive version of PASCAL?

When does the first edition of the PASCAL Newsletter come out? I am looking forward to receiving it.

Sincerely,

H. P., "Duke" Haiduk
Assistant Professor
Computer Information Systems
Standard Pascal book by Bill Atwood: I am somewhat afraid by this title, and I will write separately to Bill. What is exactly Standard Pascal, I don't know, and I think it should be very dangerous that anybody (but Wirth) could say that his own idea of the language is the standard, without any discussion with the community of Pascallers. Many people have different ideas on the subject, and I say more on the subject below.

Notes about Pascal in France and other French-speaking parts of Europe (Pierre Deshardins could say much more than me about the French-speaking part of America): Pascal is used in some important or smaller universities as a vehicle for teaching programming and writing software, especially in Paris (Institut de Programmation), Toulouse (Université Paul Sabatier), Bruxelles (Université Libre), Lausanne (Ecole Polytechnique Fédérale), Montpellier, Nice, Neuchâtel, etc. Some important Universities (Grenoble, Rennes) do not use it because they have made high investments in either Algol W or Algol 68, which are much better tools than Fortran and therefore stronger obstacles to the shift to Pascal. Implementations of Pascal have been made in Grenoble on the IBM 360, Paris on the CII Iris 80 and 10070, Neuchâtel on the IBM 1130, and one is being made in Nice on the CII Iris 80. More details follow the "implementation" part of the present letter. A two-days meeting about Pascal took place in Nice one-year ago. Sixty persons from France, Belgium and Switzerland attended. You will receive by separate mail the text of the majority of the papers which were presented. Additionally, there were panels about the use of Pascal in teaching, its implementation and its changes or extensions.

I plan to organize a similar meeting in May or June 77.

About the paper by Richard Cichelli: I think it would be a very useful policy to request of people sending Pascal programs to write them in the publication language and not in any particular hardware or implementation language. By "publication language", I mean the form used in both Wirth's book and in the Pascal reports from Zurich: from use of lower-case letters, underlined keywords, use of every single and aesthetic available characters, such as { and } for comments, : and ; and so on. By "hardware or implementation language", I mean the form used in Jensen & Wirth's book (because of limitations on the character set) and on every implementation: generally only upper-case letters, (• and ••••) for comments, ••, ••, ••••, •••, •••• and so on. I think the publication language is the only truly readable and aesthetic form, and that every implementation should be free to give its own interpretation of the characters which are not available on its particular hardware, provided it conforms to the general rules stated by Wirth himself. In fact, any implementation language which can be translated into another one by means of a one-page Pascal program should be acceptable, and this includes rational variants for keywords. In Cichelli's paper, examples in the text generally conform to the publication language, comments excepted, but figure 1 does not underlines keywords and uses a character set not particularly readable, and the complete program seems to me a good example of what should be never done: only upper-case letters, no underlined keywords, • for strings, •• and •••• for comments as well as •••• and ••••, and so on. I know well that it is very difficult to have any secretary to type correct program texts, and that is probably the reason why Jensen & Wirth's book was typed by a computer, but it can be done and I think it is worth the trouble. Of course, these remarks are not criticisms about Cichelli's paper, which I find very interesting and useful.

About the paper by Timothy M. Bonham: I agree about point 1. About point 2, I must recognize that to criticize Habermann about the use of "•••" instead of •••• was a petty point and probably a criticism of the typesetter, but I think also that proof-reading exists for removing such errors. The symbol •••• itself would cause problems in some scanners, since it would be the only three-characters delimiter in the language.

I have a more important criticism about the CDC 6000 compiler, which considers •••• and •••• as equivalent, for some historical and obscure reason. That is probably the main obstacle to a very simple and natural extension of the case statement, viz. the use of a subrange as a case label, by similarity with the notation for sets.

About point 3, I think that this long discussion should not be necessary if the distinction between publication and implementation language was clearly done: if you have the left arrow and like it, use it; but it is much more uncommon than you think, and there are much more Algol users than APL users on the Eastern side of the Atlantic.

About point 4, I agree, more especially as the French version of Pascal keywords uses bas and haut (up and down) as translations for to and downto. About point 5, I disagree, because three different syntaxes...
for comments seem really too much. As it is, the current syntax has more advantages than disadvantages. Once more, however, I think that every particular lexical convention which may be translated into the standard one (if "standard" is really defined) by a one-page Pascal program should be acceptable (but not for publication!).

Pascal bibliography: my translation in French of Wirth's first book (Systematic programming) is at last in hands of the publisher (Masson). I expect the book to be released during 1977.

Implementation notes: Although CII 10070 is a nickname for Xerox Sigma 7, the CII Iris 80 is another machine, more precisely an extension of the first one. Moreover, the CII operating system is different from Xerox and transporting a Pascal compiler from a Xerox Sigma 7 to a CII Iris 80 probably would not be a trivial job.

A Pascal compiler for both CII machines has been written by Messrs. Thibault and Mancel of IRIA (Research institute in Informatics and Automatics, a French government agency), by bootstrapping the first CDC 6600 Pascal compiler. It has now been upgraded to accept Standard Pascal and to allow separate compilation, and it is officially distributed by IRIA, a case which seems unique. Its overall performance seems to be quite good, and it is used in French universities which have one of these machines.

The CII Iris 50 is a completely different machine, much smaller, and we have some trouble in Nice when trying to implement Pascal. Pascal-P presently works interpretatively, but it is unusable for programs larger than one page, and consequently it cannot be used as a tool for bootstrapping a true compiler. I plan to write a brief paper for describing the bootstrap method which will be used, and which seems to be a unique one. Maybe it could be done in time to be included in newsletter number 6.

A Pascal compiler for the IBM 360, which was probably the first one, has been done in one of the universities of Grenoble. Unfortunately, the people who made it had no time nor support for distributing it, although it seems to have impressive performances in execution time (but less good in storage needed for compilation). People to contact are Messrs. Henneron and Tassart (Informatique & Mathématiques Appliquées, B.P. 53, 38041 Grenoble-Cedex, France).

Implementations for Pascal-P, Pascal-S and finally full Pascal have been done for the IBM 1130 and are in use at the University of Neuchâtel (Centre de Calcul, Chantemerle 20, Ch-2000 Neuchâtel, Switzerland).

A complete and standard compiler for the Xerox Sigma 6, 7 and 9 has been done by Pierre Desjardins, who can give you all desirable information. Anyway, it seems to be a very good implementation, especially in the domain of compatibility and conformity with the standard.

I hope that some of these informations will be of interest to you, and that my poor English will not be a hindrance. If you managed to read this long letter in its entirety, thank you for your long-suffering. I look forward to any news.

Sincerely yours,

O. LECAME
September 17, 1976

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

Dear Andy,

In response to John Eisenberg's article 'In Defense of Formatted Input' I would like to make the following remarks:

1. Formatted I/O statements are usually wrapped up in a package of confusing notations which detract from the readability of a program.

2. It is not clear that a system routine which does ord(ch) - ord('0') would be any better than the user's own routine and in addition, it is unlikely to respond in a flexible manner to exceptional numbers (e.g. beyond machine precision).

3. Formatted I/O still does not solve the general problem of number to string and string to number conversions.

The University of Illinois PASCAL compiler has a rather elegant solution to this topic. The implementation of PASCAL allows the user to 'read' or 'write' numbers or strings to and from arrays as well as files.

Sincerely yours,

Robert E. Novak
Mr. Andy Mickel  
PASCAL Users Group  
UCC: 227 Exp. Engr.  
University of Minnesota  
Minneapolis, MN 55455

29th September 1976

Dear Andy:

The first PUG Newsletter was really well done - keep up the good work.

I was sorry to read that Dr. Waite initially declined to join because of the dues cost. Since I suggested the membership rate (on the principle that "there ain't no free lunch") and I regard Dr. Waite as having potentially great positive influence on PASCAL development, I hereby contribute his dues. I hope Dr. Waite will use the Newsletter as the two-way communications channel it was meant to be.

I enclose an article from the British Computer Society Bulletin. The article, entitled "Which Language?", shows that within the next five years PASCAL could become the language of choice for university computer science programs.

I also enclose an article which addresses a mishmash of topics. In it are presented some frankly half-baked ideas. I hope the membership can cook them down (by stepwise refinement).

Sincerely yours,

Richard J. Cichelli
Dear Andy:

I have some further comments on PASCAL, based on my experience as the local implementor of the Hamburg DEC-10 version. First, on the compiler itself. My comments, as published in the Newsletter (No.5) were possibly a bit too harsh. They did not make it clear that my objections were to the interface between the compiler and the system, not to the reliability of the compiler or the quality of the code (aside from one bug in parameter passing - my blanket attack on procedure linkage seems to be more applicable to the older PASCAL compiler, rather than PASREL, the one we are now using). However the main thing I have to say is that I am unable to report on the usage of PASCAL. As far as I know there isn't any, except a few computer operators who use it to do homework that was supposed to be done on the Computer Science Department's PDP-II PASCAL system. I thought you might find my analysis of this situation useful.

Our lab is entirely a research organization. Much of our computer programming is "peculiar" in one way or another, and PASCAL turns out not to be very useful for it. First of all, we are often doing unusual input/output operations: controlling a robot, or doing random access work. PASCAL can hardly be blamed for not having the facilities to handle real-time device control. We have even had to modify the operating system slightly for that. But it does not support random access file manipulation, or for that matter any non-buffered kind of I/O.

Second, we do a considerable amount of work in artificial intelligence. PASCAL certainly has the ability to build complex data structures, but these abilities are rather low-level compared to LISP, or even SAIL. To do what we do with LISP in PASCAL, one would apparently have to write a memory-management system, either garbage-collecting or reference counting, and then build up a collection of basic procedures to manipulate the structures. I.e., one would nearly have to rewrite LISP in PASCAL. More about this below, under runtime memory management.

Finally, we do a good bit of what might be called "system hacking." This includes writing system programs such as a mail system, various programs for displaying information about system status and parameters, for analyzing dumps of monitor crashes, editors, etc. All of these programs require us to make many obscure monitor calls and to transform data between the form used in monitor calls and usable external forms. PASCAL provides no facilities for doing monitor calls (since these are by definition machine-dependent), nor does it supply the large library of conversion procedures that SAIL, for example, does (SIXBIT to ASCII, etc.).

More generally, the lack of string processing facilities makes many tasks rather inconvenient. This really falls into the same category as my complaint that PASCAL is not LISP, since we are again talking about a lack of run-time memory, management. String processing appears to require this as much as list processing. By run-time memory management I mean the ability to do NEW repeatedly, without eventually running out of space. This seems to require garbage collection, reference counts, or some such scheme, as well as an ability to get more memory from the operating system when it is needed. I realize that this is a controversial issue. If PASCAL is intended solely as an implementation language, it should not supply a built-in memory-management scheme since that is one thing that an implementor will want to design for himself. But I believe it is unreasonable to expect applications programmers to include a garbage collector and/or memory allocator in every program. I find it hard to believe that any serious use can be made of NEW without some memory
management. Indeed the DEC-10 PASCAL compiler uses extensions to PASCAL to keep its symbol table within bounds. Possibly built-in garbage collection should be available only when specifically requested. Then implementors who wish to design their own memory management would be free to do so.

Also, no attempt seems to have been made to facilitate decoding anything other than numerical input items. I sat down to write a program that would read a program name from the terminal and then run it. I knew that I would have to do the running by a call to an assembly-language subroutine. But when writing the filename scanner began to look like writing the syntax analyzer for a small compiler, I gave up and used SAIL. In fact, this was my last attempt to use PASCAL. (Filenames on the DEC-10 are not trivial: A full-blown one might be DSKB:PGM.EXE[5,731,SAIL,SRC]. Almost any part can be left out, and gets a default value. Also, the bracketed item can come first). I think the real issue is what kinds of problems PASCAL is intended to attack.

It is unreasonable to expect any general-purpose language to have the peculiar capabilities of LISP. That one could write a LISP interpreter in PASCAL is possibly all that should be asked. The inability to handle messy I/O and "system hacks" seems more serious, though. If we give up on that it seems to limit PASCAL to compiler writing and a replacement for FORTRAN. (Its shortcomings as a replacement for COBOL have been noted elsewhere.) The best language for such things on the DEC-10 is SAIL, a Stanford University extended ALGOL. It makes no pretense at machine-independence. There is a syntax for doing monitor calls directly. Any I/O mode available in the monitor may be explicitly specified (and is handled automatically, so that buffered and unbuffered I/O can be done with the same higher-level language constructs). A huge library of special purpose procedures is provided, including conversion procedures that allow one to make sense of data in funny monitor formats. If all else fails, one may insert sections of assembly language in the middle of a SAIL program. Accumulators are freed for your use, and constructs are defined to let you refer to the address of array elements, record elements, etc. in higher-level terms. SAIL also has several data types that depend upon runtime memory-management: e.g., strings, lists, and records. (I believe these three classes are separately garbage collected.)

SAIL is certainly not the ideal programming language, especially when judged by the "structured programming" and machine-independence ideals that guided the design of PASCAL. But I find it hard to imagine myself adopting a language for day-to-day use that did not have most of its features: the ability to use all of the facilities available in the operating system, run-time memory-management, and a good library of special-purpose procedures. Unless PASCAL can find a way to incorporate some of this in a reasonably structured way, I think it will not get beyond introductory computer science courses. Unfortunately, PL/I comes very close to meeting my goals (especially when used with OS/360). I fear I may find myself teaching students PL/I when I had hoped to use PASCAL.

Sincerely yours,

Charles L. Hedrick
Assistant Professor of Business Administration

CLH/jh
Mr. Andrew B. Mickel
Editor, Pascal Newsletter
Computer Center
University of Minnesota
Minneapolis, Minn. 55455

Dear Andy,

Upon reading the last issue of the Pascal Newsletter I was surprised to find that you do not distinguish between private letters and letters to the Editor. I strongly disagree with your elimination of this distinction, and fear that some of the writers of the published letters may resent your zeal for transparency.

I suggest that letters for publication must explicitly be addressed to the Editor of the Pascal Newsletter (as shown above), and that no other letters will be published.

Yours sincerely,

Prof. N. Wirth

Mr. Andy Michel
227 Experimental Eng. Bldg.
University of Minnesota
Minneapolis, MN 55455

Andy:

I have just recently finished reading from cover to cover the Pascal Newsletter Number 5. You guys deserve a big commendation from all advocates of progress in programming languages especially advocates of PASCAL and its future development.

With such a common forum as the Newsletter, perhaps we can interact in such a way as to encourage if not force, a "standard" series of improvements to PASCAL.

In one of the letters (I have searched several times for the specific one) there was mention of an implementation of Brinch Hansen's concurrent PASCAL. Could you please give me information concerning the availability of concurrent PASCAL for the DEC System - 10. That appears to me to be a desirable way to go with the operating systems course.

Keep up the good work. I will support you in spirit and continue to send in my monetary support for the Newsletter.

Sincerely,

R. P. "Duke" Haiduk
Pascal User's Group,
C/- Andy Mickel,
University Computer Center, 227 Exp. Engr.
University of Minnesota, MINNEAPOLIS, MN. 55455

Dear Mr. Mickel,

PASCAL NEWSLETTER

Judy Mullins, of the University of Southampton ICL 2900 project, has I believe been sending you copies of the correspondence we are generating between Tasmania and Southampton on the common problems of implementing PASCAL on highly structured computers (B6700 and ICL2900) instead of the more usual monolithic machines. I enclose therefore my reply to the last letter she sent you, in case you want to include it in the Newsletter. It contains, I think, discussions of several important issues.

Some time later I must write a view for the Newsletter specifically on PASCAL development, for it is clearly going off the rails (clear to me anyway). A great pity, and something should be done to remedy the residual problems in definition and to encourage greater interchange of programs and program portability. I shudder at all those PDP11 and IBM 370 implementations!

I enclose also a release on the status of the B6700 PASCAL compiler, which I ask you to print. It should clarify the situation for any interested B6700 users on a machine which is noted for the rarity of any new compilers.

Yours sincerely,

A.H.J. Sale,

Professor of Information Science.

Encls.

Dear Professor Sale,

Thank you for your long and most informative letter. It has raised two points which are of interest to both our Universities as well as, I believe, the Pascal community in general. These are

What brand of Pascal should be implemented?

and How is it implemented on high-level machines?

We have had long discussions on these questions, both previously, and in the light of your letter. Here follow our views.

What? We believe, for better or for worse, in Standard Pascal, as laid down in Jensen and Wirth. The 1900 compiler which we are bootstrapping is a full implementation of the Standard with one exception: the Pascal I/O routines are kept (e.g. write (eol), not (ch=eol) do read (ch) etc). Considering the mess the Zurich compilers got into when Ammann tried to rationalize the readln procedure for integers, we consider this a wise choice. However, Jim Welsh is proposing to support readln, writeln in tandem with the other set, to ease portability problems. To be realistic one must accept that there is no such thing as a fully portable program, but if the changes are few and well understood then an intelligent programmer will have a small, albeit irritating, task to bring an alien program up on another machine. The syntactic changes in your implementation are good ones except for the % comment which reminds me too much of assembler, and the ELSE case label (more of that anon). But if you allow all these goodies and students get to know of them, it merely widens the communication gap both between the students and the excellent text books that are beginning to appear on Pascal, and the students' programs and other machines. Burroughs installations have for many years fed the Algol 60 communication gap, and gaily ignored its consequences. Certainly, B 5700 Xalgol programs are more...
readable and better in many respects than their equivalents on conventional machines. Let us hope their ideas have been copied! Experience has shown that people will stick to a standard unless it is quite intolerable. Fortran is just inside the tolerable, BASIC (as defined at Dartmouth in 1971) is not. Extensions beyond with the result that the Standardization Committee cannot hope to produce a useful document now. Pascal must have a more promising future.

What should be done then? Firstly, we believe that Pascal is well within the tolerable. It is a sparse language which gives it its main advantage - tight and fast implementation. It is well-defined, well-documented and supported by academic articles in readily available journals. It also represents a great leap forward in the use of data structures and readable self-documenting programs. Therefore we shall implement Pascal as in the Report, and that alone is pretty good. Secondly, there are holes that cannot be ignored and two of the root problems are file I/O and diagnostic aids. We are taking the line that both of these should be engrossed as little as possible on the Pascal source. What additional system will be needed for files on the 2970 will only be considered in about January and I shall keep you informed. If you could send me full details of your I/O interface, it will be around at the vital moment and we can see if the 2970 version can be made to conform at all.

As regards diagnostics, we are lucky here. Glasgow have implemented a diagnostic package for symbol table dumps, profiles and tracing on the 1900 compiler and will be putting it onto the 2900 as soon as our compiler is ready. (They are getting a 2980 eventually.) The system is simple and induces minimal overheads at execution time. A few parasitic procedures write information to disc and in the event of a failure, the post mortem program (written in Pascal) comes in to dissect the corpse and produce its report. All very clinical! I enclose a copy of Glasgow's documentation. In conjunction with David Watt and Bill Findlay we are going to change the user interface from the horrible prognostic to more recognisable directives (proposals attached). Global options will come before the first Pascal statement (be it CONST, PROGRAM OR PROCEDURE) e.g.

```
  OPTIC LIST = FALSE;
  RETRACE = 500;
```

with suitable defaults, local options come in as comments e.g.

```
(*$ LISTOFF, TRACEOFF *)
```

One further hole is that of separate and mixed language compilation. There is a strong feeling to keep to the Burroughs idea of all input in Source. Could you send us details of the 68700 INCLUDE options as we shall no doubt have to write our own software to do this? By the way, what do you think the meaning and implications of assigning one file to another are? If one could do this in a sensible way, then each INCLUDEd file will be activated in the compiler by a simple input assuming! The decision for mixed language programming is a sticky one, but necessary because of the tremendous library resources available in Fortran. We hope to be able to provide this and the INCLUDE option. For once, the 2980 design is helpful: the compiler output format proposed for March 1977 and thereafter, Object Module Format, can be input to the collector or the leader.

I hope you don't mind if I send a copy of this letter to Andy Nickel for the P.U.G. newsletter as I had been meaning to write to him on the same topic. On the phone about distributing newsletters, he reiterated his concern about the health of Pascal and the extent to which it is diversifying. No one is quite sure what to do, except worry, but I agree that a committee is not the answer.

I haven't broached the second question (i.e. How) but that will have to wait for another letter. Before I close, two quick comments.

**ELSE case label:** I had the opportunity to discuss this with Wirth and his objection is a very valid one i.e. the programmer will put the ELSE there to catch values which he expects, but wants to treat in such-and-such a way. What will happen is that it will also catch inevitable values that "can't possibly occur" and the program will produce wrong results, when it should have halted in error.

**SYNTACTIC ERROR** Your syntactic options (TO instead of OR, OF instead of:) become almost justifiable if you provide a macro processor, written in Standard Pascal which will convert any 68700 program to the Standard. This will put some rein on the extent of the changes, and can be given to any serious programmer who leaves the B6700.
The University of Tasmania
Postal Address: Box 252C, G.P.O., Hobart, Tasmania, Australia 7001
Telephone: 23 0581. Cables 'Tasuni' Telex: 58150 UNTAS
INFORMATION SCIENCE DEPARTMENT.

22nd October, 1976.

Miss J. Mullins,
Department of Mathematics,
The University of Southampton,
SOUTHAMPTON, S09
UNITED KINGDOM.

Dear Miss Mullins,

Thank you for your letter. Let me try to answer your specific points; some of which I agree with, and some of which horrify me. I have drafted part of this letter as a person-person communication, and part as open comments on aspects of the PASCAL development cycle.

Standards.

Of course I agree that standard PASCAL must be adhered to, but not to the absolute exclusion of all deviation. A few features of PASCAL ought not to have been standardized (incontrovertibly in my opinion), and if absolute adherence is demanded I fear for the future fate of the language. Consider that programmers do not in general change languages unless they perceive advantages in several areas. A delightful language embedded in a lousy implementation, or with poor facilities, may not displace a poor language which has evolved into a good support situation. PASCAL vs FORTRAN for example....

I cannot agree with your comments on the likeliness of adherence to standards. Standards are maintained as limits by programmers for reasons which are quite independent of the tolerability or otherwise of the language. Indeed, I think that PASCAL is quite as much under pressure from burgeoning differences as was BASIC, and for very similar reasons. It should not have escaped your notice that adherence to the limits of the FORTRAN standard is not the common practice of FORTRAN programmers, not yet of the supplier's compiler-writers. Where it is adhered to, it arises out of the need to write portable software. This is the prime determinant of adherence to standards by programmers; a certification process is of some effect with compiler-writers.

Frankly, I think that insufficient thought has been given to ensuring a healthy future development of PASCAL. Just as insufficient thought was given to the problems of its portability. The very sparseness of the compilers are an encouragement to diversity, with all the effects one can witness in the PDP-11 and IBM 370 implementations, not the reverse. I have given a lot of thought to the ecology of languages, and perhaps I can best give notice of a dispute on suitable protection of PASCAL's niche.

Character codes

Ouch! If you are honest about standards you should realize that inventing any new character set (even if it be ASCII offset by 64) is just plain stupid. No-one outside Europe would contemplate it, even if ASCII and EBCDIC are not all one could wish. Neither is your alternative, of course. If I were involved in your project I'd scribble with a huge red pen through that proposal, and tell you to go and live with the standards. Fortunately, the axiomatic definition of PASCAL char admits that the alphabets might not form a compact set (as they do not in EBCDIC), nor an order between space/digits/lowercase-uppercase. Why is it so important to you, anyway?

I hope this will make you not quite so pleased about PICS. The set constructs you quote should be possible given any large enough set implementation, but if one has to be aware of some structure of the language's character set, I see no reason why you need to have a new one of your own. Try

```
IF CH IN ['A' TO 'Z', '0' TO '9', 'a' TO 'z'] THEN ....
```

Frankly, I cannot either see why you think all ASCII's low 64 control codes and lower-case are so unimportant. Perhaps you are not thinking of an Interactive environment; just of a number-crunch set of users with batch?

If you can convince me that you can supply a program (written in Pascal of course) that will accept Pascal program source text written to use PICS, and will produce equivalent Pascal program source text that uses EBCDIC, I might almost begin to think the innovation just acceptable!

B6700 S comment

Yet, it is reminiscent of assembler isn it? But not all assembly language features (nor yet FORTRAN) are bad. Although PASCAL's comment facility is streets ahead of Algol's, BASIC's or COBOL's, it still has a few defects, such as a propensity for swallowing text without trace. This is a minor addition (experience shows that this sort of comment is preferred by programmers than having to close one off) to keep B6700 compatibility, and to entice B6700 programmers to transfer. A few lollies help now and then will do wonders.

I have in mind too prime requirements for extensions: an extension should fall into one of two classes: it can be removed by a simple context-free program to produce a standard-compatible version, or the facility provided is quite unavailable in the standard. This falls into the first class.

ELSE in CASE

I have heard the argument you attribute to Wirth before, but I cannot give it much force. In fact, PASCAL leaves undefined the action of a CASE where the expression evaluates to a value not matched by a case label. Consider the situation (I nearly wrote 'case') where

```
(i) the value is in-range of the type, but not represented, and
(ii) the value is out-of-range of the type.
```
It is arguable that in the first case the effect should be 'do-nothing' on the analogy of IF-THEN; or it should be a run-time error on special arguments. I personally incline to the second (as Wirth), but the first is far from indefensible. Only in the second case should a run-time error always be caused; and this may be for reasons quite independent of the structure of the CASE, but simply because of the type involved. There are a few nasty spots with semi-infinite types (integer)...

I remain unrepentant: ELSE in CASE is a feature which mirrors the way programmers think and offers ways of expressing things that are unbearably cumbersome without it (try a CASE on char in a lexical analyser), though I will concede that some poor programmers can misuse it. But this is true of other features of PASCAL, for example in the REPEAT where the relation is expressed as an "escape" condition instead of a "continue-looping" condition, encouraging widening the likelihood that infinite loops are created.

As a further argument I will ask you to put yourself in the position of a compiler-writer writing a compiler which you know will be used on the other side of the world from you. You use hundreds of CASEs. All of them should be proof against flaws and bad input. Your end-users will curse you (and not a little) every time the compiler crashes because of an out-of-range CASE. Are you then willing to forego ELSE in favour of labious and error-prone lists of alternatives not expected? Robustness is as much a virtue as correct behaviour with expected input....

Mixed-language

You put your finger on a severe and very important spot. See my longer comments. The B6700 problems revolve around the complex structure of the code-file, and the complex actions taken by the BINDER to reorganize the outer-block stack for OWN objects, and the segment dictionary for VALUE objects. Tie this up with stack cleanup activities, files, and binding external objects (including files) by name into externally compiled procedures, and checking parameter compatibility at bind-time, and you may realize that the complexity of a codefile with the B6700/87700 minor differences.

Examples which might be self-explanatory (unlike T+, etc.):

```
SET LIST,TABLE,CODE,LINEINFO,EXTRACT
RESET LIST
RESET LISTINCL
ERRORLIMIT=100, HEAP=2000
```

The B6700 native style is

```
$SET LIST, TABLE, CODE, LINEINFO
```

while a PASCAL adoption might be

```
(SET LIST, TABLE, CODE, LINEINFO)
```

Diagnostics

No real comment; all seem good ideas, though my Burroughs experience leads me to suspect that I would spend more time fighting the operating system than anything else. I have in mind also allowing a facility so that interactive users can browse through the saved state making enquiries of variables, etc. Dumps of any sort are sledgehammers, where screwdrivers will often do.

Compiler options

Unless you are absolutely stuck on every detail of the CDC PASCAL implementation, you will not implement compiler options in the same way. The mechanism for specification is too terse, unimaginative, and not self-explanatory. Judging from previous remarks emanating from Southampton, there is probably no fear on that score.

May I suggest that the B6700 style is quite good, whether it begins with a $ on a newline, or is enclosed within a PASCAL comment. Look it up in the manuals (Algol say), but briefly one can SET, RESET or POP an option name, each being regarded as on a bit-stack (48-bits deep). SET and RESET push the stack down and insert the new value. Some options can have numeric values, in which case SET/RESET/POP are not relevant, and a few are special (INCLUDE). User-defined options are also allowed, allowing the user to set up source text which is parameterizable at compile-time (as assembly-code programs could often do). We use user-options to control the insertion of checking-code in the semantic code-generation routines (compiler debugging) and probably for B6700/87700 minor differences.

```
Examples which might be self-explanatory (unlike T+, etc.):
SET LIST,TABLE,CODE,LINEINFO,EXTRACT
RESET LIST
RESET LISTINCL
ERRORLIMIT=100, HEAP=2000
```

The B6700 native style is

```
$SET LIST, TABLE, CODE, LINEINFO
```

while a PASCAL adoption might be

```
(SET LIST, TABLE, CODE, LINEINFO)
```

Sets

Why do you not make a first implementation of sets which stores them as a packed array of bits, and therefore pointed to by a descriptor? Since there are no set-constants as such (the set-constructor is funny), you can get away with it, and operations on sets can be done either by in-line code, or by intrinsic procedure calls. Bit-testing (in s) can be implemented by using the low 5 bits (=32 bits of your word) as a bit-within-word index, and the remnant of the set-element as a word-within-array index. This is how we shall implement any-size sets, though we move from one-word sets to this. You might choose to implement any-size and introduce an optimization for small sets later.... It might get you out of that silly character problem.

Bounds checking

The Welsh's technique of compile-time bounds checking was going to be built into our compiler, but has not been on the first attempt. Primarily this is because we have been focussing on our main problem: the B6700 and its system, and not so much on nice features of the compiler. I think I have a long list of run-time and compile-time improvements which we shall consider when the compiler reaches its second stable point (with a comprehensive file and I/O facility).

Incidentally, I am not at all sure that read(ch) dominates our compiler's speed; intuitively it seems unlikely given the very efficient lexical analyser. My estimate at the moment is that the speed is mainly limited by the symbol table, and by the code generators.
It does not surprise me to hear that Warwick University do not have compiler expertise. It is very rare in 86700 installations; no one writes compilers for 86700s, or so it seems to us. All sorts of people fudge the existing ones, but that is a quite different activity from creating a complete system. Tell me of your impressions of 86700 expertise available close to you. This is one of the reasons we have been accumulating information in this area, so that we can become experts and hopefully contribute to the future of structured computers.

Arrays, and off-stack storage

I suppose you'd noted that our compiler does not store any multi-word object on-stack (apart from the double-word items such as double-precision and possibly complex variables). Instead there is a descriptor (one-word) for each multi-word object, which describes a linear piece of core containing the object (array of integer, array of record, array of array, record, array of record, etc.) If large enough (say more than 1024 words) the linear store will be segmented into the 86700 256-word pages.

It is not feasible to store arrays within the stack for two good, though not absolute, reasons. Firstly the stack address space is limited; for usual nesting of procedures to 2k to 4k (11 or 12 bits of displacement address), and it is locked into core (not overlayable). One doesn't want to use it up too fast, though it is conceivable that records which do not contain arrays as elements could be in the stack. Secondly, it is presumed that descriptors in the stack refer to off-stack allocated storage, by the MEG. To be sure the Algol compiler has a curious piece of code that can allocate an in-stack array (it turns interrupts off and does some weird things) but I have not yet been able to evoke this action, nor is it likely to be very nice. You see, our solutions are different from your initial ones.

Pointers

Since pointers point only to things in the heap, and since the heap descriptor location is known to the compiler, we store pointers as one-word integers (with zero used as the nil value at present). Since a single vector on the 86700 is limited by the software to about 150k words, we could pack it into less (say 20 bits) if we need to. I'll probably change the nil value to some out-of-bounds index so that the hardware checks will trap it.

Speed and Space

It may interest you to see those sample jobs I sent you and note that PASCAL compiles the twiddly job at about 10% - 20% slower than ALGOL or FORTRAN, but executes in about 3/4 the code space (4k instead of 8k average) and about 70% of the data space (5k). Space is a global property, so the small size is very welcome. Speed is a local property of programs, and perhaps some tuning will quite reverse the situation, though perhaps not by much.

I await your next letter with interest; I too have forwarded a copy to the Newsletter.

Yours sincerely,

A.H.J. Sale
Professor of Information Science

Trans Union Systems Corporation
111 West Jackson Boulevard
Chicago, Illinois 60604
312/431 3300

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

Dear Mr. Mickel:

Looking over the September issue of the PASCAL newsletter, I quickly concluded that the Tokyo compiler is the only one likely to serve our needs. Since we contemplate using PASCAL in commercial applications we need a reasonably reliable and efficient implementation, and the only other one that meets those criteria is the Manitoba compiler; but it supports I/O only on a card reader and a printer, which (for us) is absurd. But the most recent published information on the Tokyo compiler is dated from the middle of May. Has there been any progress since then? We're strongly interested in getting a copy of it as soon as it is available to run under IBM's OS/370 control program.

Very truly yours,

Jonathan Sachs

October 29, 1976

Page 62
TIMOTHY N. BORLAND
6005/1635 S. 6th St.
Minneapolis, MN 55404
11-04-76

CURRENTS ON SEVERAL ITEMS:

Dynamic Array Parameters: Jacobi's proposal is very welcome; it will fill an area in Pascal that many users have perceived as wanting in comparison to other languages. Further, the structure seems to fit well into the present Pascal syntax and does not appear too difficult to implement. However, one area of the proposal seems debatable to me—the standard functions 'low' and 'high'. It seems to me that one of the reasons for the success of Pascal is the close resemblance of many of its features to those of other languages such as ALGOL and Fortran. This makes the task of programmers who already know those languages and are attempting to learn Pascal much easier. I strongly agree with C. A. H. Roore (in his article "Hints for Programming Language Design") that a main task for the language designer is consistency and consolidation—both within the language, and, if possible, with other languages. For this reason I would suggest that these standard functions be given the names 'lowbound' and 'highbound' instead of 'low' and 'high'. This would be more consistent with the names used for the similar functions in ALGOL and PL/I. I do not think that the minor disadvantage of slightly longer names is significant; especially in consideration of the way in which the names 'lowbound' and 'highbound' more clearly express the meaning of these standard functions.

Standardization...It is becoming clear that Pascal is in need of an "official" standard, formally published by some group such as ANSI. The language is presently plagued by a host of inconsistent and contradictory additions, extensions, and modifications. If this trend is allowed to continue, Pascal will soon become no more portable than ANSI. There were several comments on this in newsletter #5. I would like to add my voice to those who seem to be calling for a Pascal Standards Committee to define a formal "standard". I'm somewhat nervous about a committee—especially the tendency they seem to have to compromise rather than choose the best; but I don't know of any other acceptable way to go. Hopefully the committee structure will be similar to that of SIMULA, where the committee members are themselves implementors (thus ensuring that the implementations are standard and the standards are implementable); and that there will be a lot of communication between the committee and the users. I would be very willing to assist such a committee in any way that I could, and I hope that something is organized soon, before Pascal becomes more a collection of similar dialects than a single standard, portable language.

Implementations...Does anyone have any information on a Pascal compiler for the IBM System 37 (if there are none available, this would seem to be a good project for some computer science student, since this machine is widely distributed.) Also, does anyone know of a compiler for the Control Data 3200?
IMPLEMENTATION NOTES

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

1) A checklist to consider when sending us information about distributed versions of Standard Pascal.

2) Pascal-P, a "portable" compiler of Pascal for a hypothetical "stack machine". It comes on tape as a kit and may be used to bootstrap compilers onto real computer systems.

3) Other portable compilers: Pascal Trunk compiler, Pascal-J, Pascal-S, and Concurrent Pascal.

4) Implementation independent software writing tools.

5) Compilers and software writing tools for specific computers sorted by computer system.

Our policy is to print only new information. If you do not find what you are looking for in Newsletter #6, check #5.

As Newsletter #6 goes to press, we still do not have enough implementation and distribution information. However, thanks to Timothy Bonham, we sent requests for information to over 80 known implementors late in October. The responses we have received since then have been very gratifying. We thank all of you who have taken the time to respond, the replies have been a big boost for this section of the newsletter.

Again we must stress that we need more information. The PUG Newsletter is the focal point for communications dealing with Pascal; implementors and distributors must keep us informed. We encourage users to share their experiences by sending qualitative and quantitative descriptions of particular implementations. Please realize that individual requests for information are a great drain on our resources.

Those sending information are encouraged to consider the checklist, and if possible, to supply a short order form (both "camera ready"). To further the spread of Pascal, and avoid duplication of effort, this section should be kept complete and up to date. —John.

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

CHECKLIST - CHECKLIST - CHECKLIST - CHECKLIST - CHECKLIST - CHECKLIST

1. Names, addresses, and phone numbers of implementors and distributors.


3. Operating system(s), minimal hardware configuration, etc.

4. Method of distribution (cost, magnetic tape formats, etc.).


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, compilation execution speed compared to other language processors (e.g. FORTRAN))

9. Reliability of compiler or interpreter (poor, moderate, good, excellent).

10. Method of development (from Pascal-P, hand coded from scratch, bootstrapped, cross-compiled, etc.; effort to implement in man months, experience of implementors).
It seems that Pascal-P has been stabilized/frozen. In a letter of 14 Sep 76 to George Richmond, Niklaus Wirth stated: "As for Pascal-P, where we have done a major revision this past spring, I cling to the hope that we can leave it at that, merely continuing the handling of new orders."

To order Pascal-P, use the updated form on the following pages (* we apologize for the disjointness of the two parts of the form *). See Newsletter #5 for more complete information on Pascal-P, in particular for explanations of the installation parameters and magnetic tape format.

If you are in Europe, Asia, or Africa, order from:
Ch. Jacobi
Institut fur Informatik
E.T.H.-Zentrum
CH-8092 Zurich
Switzerland
(phone: 01/3262 11)

If you are in North or South America, order from:
George H. Richmond
Computing Center: 3645 Marine St.
University of Colorado
Boulder, CO 80309
U.S.A.
(phone: (303) 492-8131)

If you are in Australia, New Zealand, or Oceania (*Antarctica too?):
Carroll Morgan
Basser Department of Computer Science
University of Sydney
Sydney, N.S.W. 2006
Australia

Order form for the revised Pascal P system.

Please provide us with your revised Pascal P system according to the specifications on the this form.

Address for delivery of the system

The characteristics of our installation are

Machine type

Operating system

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

intsize
realsize
charsize
boolsize
ptsize
setsize
stackelsize
strlength
intbits
sethigh
ordmaxchar
intal
realal
charal
boolal
ptral
setal
stackal

(* price information for options B and C is unavailable. *)

(for options B and C is unavailable. *)

PASCAL-P

IMPLEMENTATION NOTES

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

Our interpretive PASCAL-P system has been running since November 1975. This summer the portable compiler was rewritten in Burroughs B5700 compatible ALGOL. This new version of the compiler generates P-code which is subsequently "executed" by our P-code assembler/interpreter. Compile times have improved by a factor of 16 as compared to the interpretive system. I expect a further improvement of two to four will be achieved by rewriting the procedures INSYMOL and NEXTCH.

Using the (slightly modified) PASCAL source code supplied with the PASCAL-P implementation kit, this new compiler has successfully compiled both the PASCAL-P compiler and the P-code assembler/interpreter. The source code for the PASCAL-P compiler contains several records (lines in PASCAL terminology) longer than 80 characters. These had to be rewritten/shortened to make them acceptable to our new compiler. (We expect input to come from cards or a teletype.) Also, one or two long string constants had to be broken in two to satisfy the STRGLTH restriction of our system.

The source code for the P-code assembler/interpreter was a bit more troublesome since it is written in standard PASCAL rather than PASCAL-P. The problem areas were: too many long constants in procedure INIT; the standard types TEXT and ALFA; the standard procedures RESET, REWRITE, and PACK; an argument of type BOOLEAN in a WRITE invocation; an octal (that's right, folks!) format specification in a WRITE invocation; an actual parameter that was a string constant passed to a procedure expecting a PACKED array of type CHAR; the attempt to reference the procedure PUSH in procedure EX3 (PUSH is local to procedure EX0); string constants too long for our system; standard I/O procedures without file parameters; semicolons preceding the final END in case statements (surprise!); and a function (BASE) of type subrange.
Having completed the rewrite of the compiler, the next step was obvious — modify it!

The error codes emitted by our new compiler are different from those emitted by the Zurich compiler. The compiler tests for over 250 different syntax errors and each of these errors is now associated with a unique error code. This allows the corresponding error messages to be more explicit and, hence, useful to the novice users of our system.

We have added another (extremely useful) type of comment to our PASCAL system. A percent sign (%) is used to signal the compiler that the rest of the current source image is to be ignored (our system respects card boundaries in this case). This allows the programmer to place short documenting comments after the percent sign. This type of comment is very useful. It is much less error prone than the multi-character comment delimiters in PASCAL and PL/I. It speeds up compilation by reducing the number of characters the compiler must "look at." It encourages proper documentation by placing the comment alongside the code. (Assembly language programmers have been doing this for years with satisfying results.)

A future report will outline some of the ways in which our PASCAL system is being used in support of our Computer Science program here at the University of Wisconsin - Eau Claire.

Dr. Bruce A. Pumplin
Since October 1975 reports and system tapes have been distributed to 252 institutions:

86 companies
119 universities
31 research laboratories
16 others

AEG - Telefunken
Amald Corporation
Analog Technology Corporation
Basic Timesharing Inc.
Bell Telephone Laboratories
Boeing Aerospace Corporation
Bolt, Beranek & Newman
Burroughs Corporation
Comptek Research Inc.
Computer Automation
Computer Consoles Inc.
Computer Sciences Corporation
Digital Equipment Corporation
E. I. du pont de Nemours
Electromagnetic Systems Laboratories
First Data Corporation
Fisher Controls Company
John Fluke Manufacturing Company
Fordham Corporation
General Automation Inc.
General Electric Company
General Radio Company
General Research Corporation
GRI Computer Corporation
GTE Laboratories
Hewlett Packard Corporation
Hitachi Research Laboratory
Honeywell Inc.
Inco Inc.
Interton Corporation
Intel Corporation
Interdata Inc.
Intermetrics Inc.
International Business Machines
Arthur D. Little Inc.
Logiscon Inc.
Manufacturing Data Systems
Media Reactions Inc.
Metrology Engineering Center
Mills International
Mobydata
National Cash Register
Nuclear Data Inc.
Republic Electronics
Rockwell International
Sanders Associates
Softco Inc.
Sperry Univac
Squibb & Sons
System Development Corporation
Technology Marketing Inc.
Tektronix Inc.
Texas Instruments Inc.
TMW Systems Group
Varettek Computer Systems Inc.
Varian Associates
Wang Laboratories
Westinghouse Electric Corporation
Xerox Research Center

Bell Northern Research (Canada)
C.C.E.T.T. (France)
Databank Systems (New Zealand)
Dynamec Corporation (Canada)
EDG-Bentet (Norway)
Edmor Information Consultants (Canada)
Electronics Corporation of India
L.M. Ericsson (Sweden)
Finnin Tractor & Equipment (Canada)
Ikslas S.A. (Spain)
IMAG (France)
Logica Ltd. (England)
C. Olivetti (Italy)
Philips CFI (France)
Philips (Germany)
Regnearclonal (Denmark)
Reuters Ltd. (England)
Christian Roysing (Denmark)
Siemens Ag (Germany)
Softlab (Germany)
Software Sciences Ltd. (England)
Systems Designers Ltd. (England)
Telecinco (Spain)

Brandeis University
California Polytechnic State University
California State University, Northridge
Carnegie-Mellon University
Case Western Reserve University
Clarkson College
Cornell University
Duke University Medical Center
Georgia Institute of Technology
Harvard University
Johns Hopkins University
Kansas State University
Massachusetts Institute of Technology
Michigan State University
Naval Postgraduate School, Monterey
New York University
Princeton University
Purdue University
Oral Roberts University
Rangamani State University
Stanford University
Syracuse University
The College of Wooster, Ohio
University of Arizona
University of California, Berkeley
University of California, Irvine
University of California, San Diego
University of Colorado
University of Delaware
University of Florida
University of Iowa
University of Maryland
University of Michigan
University of Minnesota
University of North Carolina, Chapel Hill
University of Texas, Austin
University of Utah
University of Washington
University of Wisconsin
University of Wyoming
Washington State University
Washington University, Missouri
West Coast University
Western Washington State College
Virginia University

Australian National University
Bezalel Academy (Israel)
Carleton University (Canada)
Chalmers Technological University (Sweden)
Concordia University (Canada)
Durham University (England)
Ecole Polytechnique Federale (Switzerland)
Eidg. Technische Hochschule (Switzerland)
Faktshochschule Reutlingen (Germany)
Simon Fraser University (Canada)
Imperial College (England)
Instituto Politecnico Nacional (Mexico)
Katholieke Universiteit Leuven (Belgium)
Keio University (Japan)
Johannes Kepler Universität (Austria)
Kyoto University (Japan)
McGill University (Canada)
Monash University (Australia)
Nou-Technikum Buchs (Switzerland)
Politecnico di Milano (Italy)
Queen's University (Canada)
Queen's University (Northern Ireland)
Royal Institute of Technology (Sweden)
Seikei University (Japan)
Technical University of Delft (The Netherlands)
Technical University of Denmark
Technical University of Eindhoven (The Netherlands)
Technische Hogeschool Twente (The Netherlands)
Technischen Hochschule Munchen (Germany)
Technische Universitat Berlin (Germany)
Technische Universitat Wien (Austria)
Trinity College (Ireland)
Universidad Politecnica (Spain)
Universidad Simon Bolivar (Venezuela)
Universite Catholique de Louvain (Belgium)
Universite de Montreal (Canada)
Universite Liraris (Belgium)
University of Aarhus (Denmark)
University of Adelaide (South Australia)
University of Alberta (Canada)
University of Amsterdam (The Netherlands)
University of Bonn (Germany)
University of Bradford (England)
University of British Columbia (Canada)
University of Brussels (Belgium)
University of Cape Town (South Africa)
University of Copenhagen
University of Dortmund (Germany)
University of Edinburgh (United Kingdom)
University of Glasgow
University of Hamburg (Germany)
University of Karlsruhe (Germany)
University of Kaiserslautern (Germany)
University of Kiel (Germany)
University of Manitoba (Canada)
University of Newcastle upon Tyne (England)
University of New South Wales (Australia)
University of Oslo (Norway)
University of Saskatchewan (Canada)
University of St. Andrews (Scotland)
University of Stuttgart (Germany)
University of Sydney (Australia)
University of Tampere (Finland)
University of Tasmania (Australia)
University of Tokyo (Japan)
University of Toronto (Canada)
University of Tromsø (Norway)
University of Trondheim (Norway)
University of Warwick (England)
University of Waterloo (Canada)
U.S.S.R. Academy of Sciences
Vrije Universiteit (The Netherlands)

Michael Reese Hospital, Illinois
Southwest Regional Laboratory
Stanford Artificial Intelligence Laboratory
Stanford Linear Accelerator Center
Stanford Research Institute
University Hospitals, Cleveland, Ohio
U.S. Army, Fort Meade, Maryland
U.S. Army Research Laboratory, Illinois
USC Information Sciences Institute

Australian Atomic Energy Commission
Central Institute for Industrial Research (Norway)
CERN (Switzerland)
Commonwealth Scientific and Industrial Research Organ (Australia)
Koninklijke/Shell Laboratorium (The Netherlands)
National Physical Laboratory (England)
Norwegian Defense Research Establishment
Oesterreichische Studiengesellschaft fuer Atomenergie (Austria)
Max Planck Institut fur Biochemie (Germany)
Royal Radar Establishment (England)

Electromagnetic Systems Laboratories, Sunnyvale, California
Jet Propulsion Laboratory
Los Alamos Scientific Laboratory
M.I.T. Lincoln Laboratory
Naval Electronics Laboratory, California
Naval Research Laboratory, Maryland
Naval Undersea Center, California
Naval Underwater Systems Center, Connecticut
New Mexico Institute of Mining & Technology
Oregon Museum of Science and Industry
Pacific Marine Environmental Laboratory, Washington
Research Triangle Institute, North Carolina
AN AUTOMATIC FORMATTING PROGRAM FOR PASCAL

Jon Hueras and Henry Ledgard
Dept. of Computer and Information Science
Univ. of Mass. / Amherst, Ma. 01002

Imposing formatting restrictions necessarily imposes a burden on a programmer, particularly on a student programmer, since he must keypunch or type in the entire program himself. It is therefore useful to have a facility for taking arbitrarily formatted source code and automatically prettyprinting it. However, the design of any such prettyprinter must deal with several serious issues.

Typically, automatic prettyprinters take a heavy hand in formatting a program, right down to every last semicolon. Such a scheme either formats everything in a rigid fashion, which is bound to be displeasing, or else it provides the programmer with a voluminous set of "options". Furthermore, such a scheme must do a full syntax analysis on the program, which means that it falls prey to the bane of all compilers: error recovery. Thus, before a program may be prettyprinted, it must be completely written and debugged. If the programmer wishes to prettyprint a program still under development, he is out of luck, or else he must do it by hand, in which case he has no need for an automatic prettyprinter when he is done.

We believe that it is not necessary to impose more than a minimum set of restrictions, and that any prettyprinter should yield to the programmer's discretion beyond this minimum. No matter how many options a prettyprinter has, it cannot possibly have one to please everyone in every possible case. We further believe that a prettyprinter should not commit itself to a full syntax analysis. It should only do prettyprinting on a local basis, dealing with individual constructs rather than entire programs.

In order to demonstrate these assertions, we have designed and implemented, in PASCAL, just such a prettyprinter. It is intended mostly as an editing aid, and thus does not include most of the "kitchen sink" facilities used by other prettyprinters. It simply rearranges the spacing and indentation of certain constructs in order to make the logical structure of a program more visually apparent. Furthermore, the prettyprinter forces only a minimum amount of spacing and indentation where needed. Any extra spaces or blank lines found in the program beyond the minimum required are left there. This leaves the programmer a good deal of flexibility to use as he sees fit.

The prettyprinter that we have implemented is written entirely in standard PASCAL (according to the Revised Report in Jensen and Wirth), and should compile and run using any PASCAL compiler that supports this standard. We have compiled it using the PASCAL 6000-3.4 compiler from Zurich, and run it in 11K (octal) of core on our CDC CYBER 74. The program, as written, is highly modularized and table-driven, and is therefore extremely easy to modify and upgrade.

A copy of the program, with documentation, is available from H. F. Ledgard: $7 for a standard 7-track tape or cards, or $2 with a user-supplied tape.
This letter is in response to your inquiry re our B1726 Pascal implementation. Unfortunately this project was abandoned over a year ago because of lack of time, and the (then) continuing lack of suitable documentation from Burroughs.

However, I can give you details of other B1726 Pascal implementations. These are:

(1) Pascal implemented by Elliott Organick's group at the University of Utah. This compiler (and interpreter) is based substantially on Brinch Hansen's Solo Sequential Pascal compiler (i.e., it compiles in 7½ passes) and we have a copy of it. We haven't had much experience with it yet, but from what we have seen so far we're not very impressed. The compiler appears to be somewhat glitchy and unfortunately does not adhere to the conventions observed by Burroughs-supplied compilers.

(2) At the European B1700 University Users Meeting on July 15-16, 1976, the following projects were mentioned:

(a) University of Zurich (which is not ETH, incidentally) are working on

   (i) implementation of a P-code interpreter for Pascal.

(b) P. Albrich of the University of Karlsruhe (Germany) is implementing Concurrent Pascal. Nothing was mentioned about (Sequential) Pascal.

(c) M. Ellison of the University of Newcastle-upon-Tyne is implementing "another Pascal Machine architecture". He reports that an interpreter for this system's machine code has been de-bugged and that it is to be benchmarked with "Zurich's Pascal-P version 1.0".

There is to be another European user's meeting at Karlsruhe in February 1977. Our own contact for all the European information is through the University of Newcastle-upon-Tyne.

I hope the above information will enable you to obtain the material required for the Newsletter.

Yours sincerely,

Antony Gerber
The University of Tasmania is developing a compiler for PASCAL to produce executable programs on the Burroughs B6700/B7700 computer systems. The compiler is currently operational but with only a simple I/O system (default file declarations and PASCAL I/O statements). Current work concentrates on implementing general file and file attribute declarations, and on extending I/O statements to a usefully sized set.

Our current policy is not to release the compiler until it reaches this next stable state as otherwise copies of the simplified version will proliferate. We would welcome any expressions of interest from B6700 users in the compiler, and will add addresses to our list of interested people, as this will determine the level of support that will be necessary to maintain the final system. The anticipated release date is December 1976.

Compiler information
The Burroughs PASCAL compiler is a true compiler for the B6700: it generates a code-file which can be executed by the system. The code-file contains no BINDINFO and cannot be bound to any other language at present (as is the case with DASIC). The execution speed and compilation speed of PASCAL are comparable to the Algol compiler, though much less code and data space is needed for compilation.

The compiler is based on PASCAL-P, but is written in Burroughs Algol. It maintains standard treatment of PASCAL features with the addition of B6700-compatible compiler-options and other features. The aim is to produce a compiler which will comply with the B6700 conventions as embodied in the Algol/FORTRAN/COBOL/etc compilers for that system, and for which transferable skills from other Burroughs subsystems can be retained.

Arthur Sale
Professor of Information Science
University of Tasmania
Box 252C G.P.O.
Hobart, Tasmania 7001

Dear Timothy Bonham,
Burroughs B6700 PASCAL Implementation

1. Names, etc. of implementors
   Professor A.H.J. Sale (Phone Tasmania (STD 002) 23-0561 Ext. 435)
   Dept. of Information Science
   University of Tasmania
   Box 252C G.P.O.
   Hobart, Tasmania 7001.

2. Machine tested on
   Burroughs Model 111 B6700 processor, with vector mode hardware, 196k words
   of main store, disk, 4 pack drives, etc. Machine operates in university
   environment with heavy interactive use.

3. Operating System, etc.
   Burroughs MCP Version 11.8 operating system with (few) minor local
   modifications for accounting, etc. Minimal system to operate: not known,
   but unlikely to be any B6700 that small (store demands are low, little
   else is critical).

4. Method of distribution
   Not officially released (December?), nor cost determined. Format will be
   via magnetic tape (both 7-track and 9-track drives available).

5. Documentation available
   Under development. Will be in form of user manual as well as supplement
   to Pascal User Manual and Report.

6. Maintenance policy
   To be maintained for teaching use within University as well as larger
   aims. Reported bugs should be fixed as soon as possible, with notification
   to users. Duration of support not yet determined; several developments
   are also pending.

7. Standard-compatibility
   Does implement PASCAL in Report with following exceptions where noted with reasons:
program heading: reserved word "program" is synonymous with procedure; no parameters (files) are permitted after the programheading.
Reason: CDC anachronism of no utility in our installation, and likely to be confusing.

set constructor of form A..B not implemented.
Reason: future plan.
FORTRAN control character on print line not implemented. Reason: a ridiculous feature to standardize.

Full Pascal I/O not implemented. Reason: future plans, present scheme is PASCAL-1-like.

Extensions
Various reserved words, character set transliterations.
Burroughs comment facility
ELSE in CASE
File attributes in declarations
Format declarations
Extensive Burroughs-compatible compiler options (Pascal option mode not implemented).

8. Compiler or interpreter, etc.

Compiler:
generates B6700 code-files which are directly executed by the B6700 with MCP. There is as yet no BINDINFO in the codefile so that it is not possible to link Pascal to modules compiled by other systems.
written in B6700 Algol entirely.

Characteristics:
compiles about 20% slower than FORTRAN or ALGOL, but in about 2/3 of their space (for test programs about 4-5k words on average instead of 5-10k). Elapsed compilation times similar, though PASCAL slower. Speed should be improved by eventual tuning.
executes at same speed as FORTRAN and ALGOL (code is very similar and optimal) and takes generally longer elapsed residence time primarily due to MCP intervention to create new segments for record structures (not present in FORTRAN/ALGOL). Elapsed residence times about 20% greater than equivalent ALGOL.
one-pass system: code generated is very close to optimal for B6700 unless checking requirements of Pascal intervene to inhibit this.
options include listing of object code in symbolic and/or absolute form, editing of input, etc.

9. Reliability
Excellent. Only one system crash during testing attributed to Pascal (in run #2), and a total of two serious bugs uncovered during extensive testing. On a machine with minimal protection against aberrant compilers as is the B6700, a high level of confidence is essential. The compiler code-generator section incorporates many reasonableness checks on the code to trap some flaws before they get executed and to aid in tracing errors.

10. Method of development
Hand-coded using Pascal-P as a guide/model. All other paths offered much greater difficulty on B6700 due to special nature of machine/system.

I trust the above comments meet your need. I enclose a copy of a brief technical report on some thoughts on Pascal implementation around August of this year. I am, and intend to remain, a member of P.U.G.

Yours sincerely,
A.H.J. Sale... Prof. of Information Science.

Dear Mr. Mickel,

Until now we have an interpreting PASCAL-System running on the B6700. We got the "PASCAL-Implementation-Kit" with the P2-Compiler (as described in Nori, e.a., "The PASCAL P-Compiler Implementation Notes", ETH Zürich) and translated the assembler/interpreter for the hypothetical stack computer from PASCAL to Burroughs Extended Algol. So we can compile PASCAL-programs by interpreting the PASCAL-Compiler and execute the generated Code for the stack computer by interpreting it. This technique is very time- and space-consuming: The PASCAL-Compiler needs about 30 min B6700 processor time to compile itself.

We didn't complete the whole bootstrap yet, because of several problems concerning the generation of B6700 machine code in general.

Sincerely

A.H.J. Sale
Dear Mister Bonham,

According to your letter dated October 25, 1976, directed to Prof. Br. G. Goos, I will give you some information about the state of our PASCAL-activities.

We have an interpretive PASCAL-system running on the B 6700. It is based on the PASCAL-P2 compiler and the assembler/interpreter from Zürich. We translated the latter one to a Burroughs Extended Algol-Program. The interpretive version is naturally very time and space consuming. (The compiler compiles itself in about 30 minutes CPU-time.) The system is available on magnetic tape for a nominal charge of § 20.00 when tape is supplied, § 30.00 otherwise. We have a short note for users of the system (in German only).

Another project on the B 6700 is based on an early version of the PASCAL-JANUS-Compiler from Boulder, Colorado. PASCAL is translated via JANUS, STAGE II and an assembler to B 6700 machinecode. We didn't test the system enough to say how reliable it is.

Both projects were not further developed nor maintained because of a lack of manpower at our institute and diminished computing capacity on the B 6700.

Sincerely,

(U. Kastens)

---

Olivier Lacarme of the Université de Nice, Laboratoire D'Informatique, Parc Valrose, 06034 Nice Cedex, France, has helped to clear up our confusion about the CII machines. In a letter of 16 Sep 76, he wrote:

"Although CII 10070 is a nickname for Xerox Sigma 7, the CII Iris 80 is another machine, more precisely an extension of the first one. Moreover, the CII operating system is different from Xerox and transporting a Pascal compiler from a Xerox Sigma 7 to a CII Iris 80 probably would not be a trivial job. A Pascal compiler for both CII machines has been written by Messrs. Thibault and Mancel of IRIA (Research institute in Informatics and Automatics, a French government agency), by bootstrapping the first CDC 6000 compiler. It has now been upgraded to accept Standard Pascal and to allow separate compilations, and it is officially distributed by IRIA. A case which seems unique. Its overall performance seems to be quite good, and it is used in French universities which have one of these machines.

"The CII Iris 50 is a completely different machine, much smaller, and we have some trouble in Nice when trying to implement Pascal. Pascal-P works interpretively, but it is unusable for programs larger than one page, and consequently it cannot be used as a tool for bootstrapping a true compiler. I plan to write a brief paper for describing the bootstrap method which will be used, and which seems to be a one. Maybe it could be done in time to be included in newsletter number 6." (* perhaps Newsletter #7 *)

CONTROL DATA CYBER 18 (an implementation exists)

2550 (Control Data supports a cross-compiler on the 1000/Cyber 70, 170 series)

3300 (Implementations exist)

3600 (an implementation exists)

6000/Cyber 70, 170 series (see also Newsletter #5)

There is very little fresh news on this implementation. It is rumored that Zurich has written a first modset (UPDATE1) to Release 2 of Pascal 10000. We at Minnesota have not received it yet. There is a new price list for distribution tapes, but no new order form. See Newsletter #5 for the old one.

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

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

If you are in North or South America, order from:

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

CII 10070, Iris 50, Iris 80

Olivier Lacarme of the Université de Nice, Laboratoire D'Informatique, Parc Valrose, 06034 Nice Cedex, France, has helped to clear up our confusion about the CII machines. In a letter of 16 Sep 76, he wrote:

"Although CII 10070 is a nickname for Xerox Sigma 7, the CII Iris 80 is another machine, more precisely an extension of the first one. Moreover, the CII operating system is different from Xerox and transporting a Pascal compiler from a Xerox Sigma 7 to a CII Iris 80 probably would not be a trivial job. A Pascal compiler for both CII machines has been written by Messrs. Thibault and Mancel of IRIA (Research institute in Informatics and Automatics, a French government agency), by bootstrapping the first CDC 6000 compiler. It has now been upgraded to accept Standard Pascal and to allow separate compilations, and it is officially distributed by IRIA. A case which seems unique. Its overall performance seems to be quite good, and it is used in French universities which have one of these machines.

"The CII Iris 50 is a completely different machine, much smaller, and we have some trouble in Nice when trying to implement Pascal. Pascal-P works interpretively, but it is unusable for programs larger than one page, and consequently it cannot be used as a tool for bootstrapping a true compiler. I plan to write a brief paper for describing the bootstrap method which will be used, and which seems to be a one. Maybe it could be done in time to be included in newsletter number 6." (* perhaps Newsletter #7 *)

CONTROL DATA CYBER 18 (an implementation exists)

2550 (Control Data supports a cross-compiler on the 1000/Cyber 70, 170 series)

3300 (Implementations exist)

3600 (an implementation exists)

6000/Cyber 70, 170 series (see also Newsletter #5)

There is very little fresh news on this implementation. It is rumored that Zurich has written a first modset (UPDATE1) to Release 2 of Pascal 10000. We at Minnesota have not received it yet. There is a new price list for distribution tapes, but no new order form. See Newsletter #5 for the old one.

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

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

If you are in North or South America, order from:

George H. Richmond
Computing Center: 3645 Marine St.
University of Colorado
Boulder, CO 80309
(phone: (303) 492-8131)
If you are in Australia, New Zealand, or Oceania, order from:

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

$30 for Release 2 tape and document.

CONTROL DATA 7600/CYBER 76

Pascal on the CYC 7600

This Pascal compiler is essentially the Zurich 6000 -3.4 compiler.
The run-time system is based on that of Hans Jornaastad (see Newsletter #4).
The compiler is release 2. It was developed by cross-compiling from our
CYBER 72.
The compiler is currently running under SCOPE 2.1.3 and will re-compile
itself on a 'half-size' (32K SCM) machine.

Compilation Speed

57000 characters/second approx. Compiler re-compiles in less
than 10 seconds.

Execution Speed

Pascal execution speed has been measured by using the obvious encoding
in Pascal of Wichmann's Synthetic Benchmark (see Computer Journal Vol.19 No.1).
The units are in thousands of Whetstones.

<table>
<thead>
<tr>
<th>Compiler and</th>
<th>No routine</th>
<th>Array Board</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimisation level</td>
<td>Checking</td>
<td>Checking</td>
</tr>
<tr>
<td>ALGOL 60 (UNS)</td>
<td>1996</td>
<td>1130</td>
</tr>
<tr>
<td>PASCAL</td>
<td>6650</td>
<td>6240*</td>
</tr>
<tr>
<td>FTN (OPT=1)</td>
<td>9345</td>
<td>3174**</td>
</tr>
</tbody>
</table>

* Using T + option i.e. all run time checks included
** forces OPT = 0

Maintenance

The situation is unclear at present. UMRCC will presumably, out
of self-interest assist with bugs in the 7600 dependent code. The vast
majority of the compiler and library is standard Zurich code.

Contacts

Mr. H. D. Ellison or Mr. A. P. Hayes at
UMRCC, Oxford Road, Manchester M13 9PL, England

Dear Andy:

I want to tell you about an implementation of Pascal for the Cray
Research CRAY-1 computer done here at Los Alamos. I will follow the
checklist outline on page 44 of PASCAL NEWSLETTER #5.

1. Implementor and maintainer:
   Robert T. Johnson
   C-11, MS/296
   Los Alamos Scientific Laboratory
   P. O. Box 1663
   Los Alamos, New Mexico 87545
   phone: (505) 667-5014


3. Operating System: It was called the Benchmark Operating System,
   and is now called IOS after several extensions and enhancements.

4. The compiler has not been distributed elsewhere, and no arrange­
   ments have been made for the distribution.

5. Documentation consists of a two-page description of how to access and
   use the compiler.

6. Maintenance on this compiler is suspended; the compiler is at
   the end of its usefulness on the CRAY-1, because support for it
   cannot keep up with system development and changes. The
   compiler has been superseded for our development work by a
   cross-compiler for the Model language designed and implemented
Mr. Andy Mickel

DIGITAL D.~TA

Mr. Andy Mickel

The quality and format of Newsletter #5 were impressive. Keep up the good work.

Sincerely,

Bob Johnson

xc: ISO-5 (2)

DATA GENERAL Nova 800, Nova 1200, Supernova, Eclipse
(no reported implementations - we need information)

DIGITAL EQUIPMENT (DEC) pnp-8
(an implementation is underway)

DIGITAL EQUIPMENT (DEC) pnp-10, DEC SYSTEM-10

(see also Newsletter #5)

UNIVERSITAT HAMBURG
INSTITUT FUR INFORMATIK

Mr. Andy Mickel

University Computer Center
University of Minnesota
427 Experimental Engineering Building
Minneapolis, Minnesota 55455 USA

Dear Mr. Mickel,

the PASCAL Newsletter number 5 has been studied by me with great interest.

I congratulate for the lively exchange of opinions that you enable by your effort. I would like to supplement the two contributions which refer to the DEC SYSTEM-10 PASCAL-compiler developed at Hamburg.

Mr. J. McCool who seems to be utterly disappointed by our compiler sent me two letters. The first letter from him dated March 16, 76, arrived here April 9, 76 presented a program listing with the remark that this program started execution and then never could be nudged out of the terminal input state - no matter what was typed in. I resubmitted the program, run it and detected that the initial READLN(TTY) had been forgotten. Therefore, the heavily recursive program was always one character ahead of what it meant to be. In order to avoid any delay due to the Easter holidays 76 I wrote him a letter indicating the trouble and mailed it personally on Good Friday. A few days later (April 26, 76) I received a letter (dated March 26, 76) from Mr. McCool complaining of not having received an answer to his first letter and indicating two additional compiler errors - the date error (2-Jan-73) and the fact, that 182,1 on input turned out to be 173,9999 on output due to the conversion routines. Both letters were sent by ordinary mail. Apparently Mr. McCool did not realize that Hamburg is on this side of the Atlantic and a letter by surface mail takes longer to get here than the time after which he shot off his second letter complaining about not being "serviced" properly.

I mailed a fix for the date error (essentially one instruction needs to be added in the runtime support) within less than a week - and that has been all I heard from Mr. McCool.

I mention this to indicate with what expectations one might be confronted if one provides programs for which not even a mailing charge has been asked by me - on the understanding that no regular maintenance can be given.

The contribution by Mr. Heirich shows how the complete availability of source code for compiler and runtime support can be used to up-
grade the compiler by the recipients. At hambury we only have a 16-bit processor - so we could not simply the special conversion instructions available with the 32-bit processor. I have been interested to learn that the use of these instructions can speed up some program. This factor of 42. This is only one of many examples where the compiler is being written to adapt to running, not by resident.

The very specific list of uncountable features in our FORTRAN compiler given by the author will not be created for the first time. A new compiler version which currently is introduced in our installation:

1. Inline core for loop, loop, and I/O-buffers is allocated. Moreover, a compiler option allows to specify the amount of core with which a program should be executed after compilation. The new compiler has been fully integrated into the DECsystem-10 Generic Command Language (GCL). The COMPIL, LOAD and LAYOUT commands can be given together with a large number of compiler options in the standard GCL format.

2. Lower case characters are no longer simply dropped. If they are input from a PASCAFILE or CHAP, lower case characters are converted to upper case characters; only the control characters are dropped except for END which is treated as END OF FILE. The problem with simply accepting lower case characters consists in the difficulty to exclude runtime errors due to the single letter Dopple on symbolic names. The new compiler - in particular conversion of CHRIS as a superset of CHAP has been introduced. This covers the complete set of 7 lower case characters but will result in runtime error messages if it is used in run-time operations.

3. The file PASCAFILE does no longer appear.

4. The generation of a listing can be controlled by the usual LIST/MONITOR switch in the sample statement.

Note: The compiler has been originally developed for students. It is my experience that more trouble results from working with outdated listings than from creating a listing. Further, users do not even have to be printer - at least it is available if the compilation error results from typing errors or incompatible "corrections".

5. The standard compiler options COMPIL, LIST, etc. commands with option switches is now available.

7. These errors are known to me and I thought they had been fixed in the version distributed July 75. I am sorry for the trouble. Mr. Lecied refers to as a poor design of parameter passing is due to the fact that we pass parameters in up to 4 registers instead of 6. The situation arises when the compiler cannot determine the program which to use due to typing errors or incompatibility of the assembly language.

5. The desire for error recovery and output fatal error is quite understandable. Moreover, it requires a complete redesign of the (assembly)

input conversion routines - something we simply had not yet time to accomplish.

3. Has been treated above.

The main improvements of the new compiler version not yet mentioned in the reply to Mr. Lecied's points are:

10. The PASCAL and LOCAL declaration has been implemented.
11. SETS out of procedure/functions are implemented.
12. The implementation of formal procedure/function is adapted to the new compiler version. The use introduced at Technical Highschool Montre, Kunschaar, Netherlands, by a student of Dr. C. Bron in an earlier compiler version that he obtained from Hambury has been implemented for the new compiler version.
13. PACK and UNPACK have been adapted to the standard definition given in the PASCAL user manual and report. The new compiler version allows to give these procedures indicating length of the array to be packed/unpacked and the starting address in the packed array.
14. The standard functions DATE and TIME are implemented as in PASCAL. Two additional standard functions CLOCK and TIMEOUT have been introduced with integer function value to determine CPU- and real-time in millisecond.
15. New standard functions NEWC and LAST have been introduced to determine the first and last byte of a scalar or subrange.
16. NEWREG and UNFIRENEW allow to determine the respective index values for array definitions to ease an easier change of array size.
17. H1 and H22 are available standard functions.
18. User defined scalar variables can be input and output using the assembly representation of the value contexts. The conversion from to the internal representation is taken care of by the runtime support. Values for SETS variables will likewise be converted from run-time input set to character representation as it appears in a source program SETS constant.
19. The introduction of a CALL enables the termination of the current PASCAL program and the immediate start of another program. Communication between programs is provided by standard DEC TOPCOP files.
20. Procedures are available to determine the value of option switches adapted to the function name from within the PASCAL program.
21. The PASCAL source level debugger system has been enlarged by an optional stack and heap dump in source level code.
22. A POST-MONITOR facility at source language level has been appended to the PASCAL system.
23. Runtime checks have been extended to cover a larger number of possible trouble points. If a program has been compiled with the following options, any runtime error detected, e.g. checking variables by the TOPS-10 monitor - will transfer control to the PASCAL system.
24. PASCAL procedures/functions and MACRO-10 routines can be compiled separately and included together with FORTRAN routines into a relocatable object code library.
25. A machine readable manual for this DECsystem-10 PASCAL implementation in English is available.
This system has been implemented by Mr. J. Fishell and is currently being tested at our institute. Before distribution, I would like to have an extended period of production use at our installation to avoid disappointment after distribution.

A recent article about the distribution of our PASCAL compiler version of July 1975 might be of interest.

<table>
<thead>
<tr>
<th>Universities</th>
<th>Industry</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA and Canada</td>
<td>22</td>
<td>16</td>
</tr>
<tr>
<td>Europe</td>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>South America</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>and Australia</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>37</td>
<td>15</td>
</tr>
</tbody>
</table>

These are the installations that I know, since the compiler could be freely distributed it might well be possible that it is used in places not known to us. From the total known number of 56, I have to subtract the installation of ICS since I assume that they discontinued use of our compiler after their bad experiences. Another five requests for our compiler have arrived in the meantime.

May I add a last remark concerning reports about PASCAL compiler experiences: since all work done so far seems to proceed on a nonprofit basis by voluntary contributions, the feedback of trouble spots to people generating a compiler version is not optimal. One should, therefore, encourage the distribution of specific complaints as, e.g., those given by Dr. Naur for our compiler version. My unspecific complaint should preferably be returned to the writer with the slightly formulated expectation that he supports his claims by at least giving details of where he encountered trouble, indicating the compiler version to which his remarks apply and from whom he obtained it. Such an editorial policy might aid to the value of the PASCAL User Group Newsletter since - among other reasons - it might make the user community aware of trouble not yet identified at their installation.

Sincerely yours,

H. Lagel

---

DIGITAL EQUIPMENT (DEC) PDP-11

Stephen C. Schwarm of E.I. du Pont de Nemours Co., 101 Beech Street, Wilmington, DE 19898, has written us: "I am Chairman of DECUS SIG PASCAL and I will be glad to help with distributing any systems on DEC PDP-11s." (*maybe Steve can help organize this section of the implementation notes.*)

PDP Systems
Box 566, Station A
Toronto, Ont.
Canada M5A 1M5
25-Oct-76

Former Address:
Box 702, Suite 6
Saskatoon, S. Mex.
Canada S7N 0C0

To: Persons interested in my PDP-11
Pascal implementation,
cc: Pascal Newsletter

Gentlemen:

This letter is to advise you of the status of my Pascal implementation for the PDP-11. I apologize for some delay in writing this letter; I've been busy moving.

In a word, my compiler is defunct.

My implementation was never more than a spare-time project, and progress was slow as I have been very busy. My move to Toronto essentially eliminates both my spare time and my cheap machine access. As my compiler never cared anywhere near operational status and it is most unlikely that I will be able to do more work on it any time soon, I have abandoned the project.

Henry Spencer
Member of Technical Staff
Dear Tim;

Thanks for your letter. We had noticed the small mention of ESI Pascal in the last PUG newsletter and were a little concerned because it made reference to the U. of Ill Pascal and implied that ours was essentially the same. ESI Pascal was based on the U. of Ill bootstrap compiler (not the student compiler they are now offering), but has been so completely rewritten and reshaped that we have no hesitation in claiming it as our own creation. Briefly, our compiler runs under RT-11 on any PDP-11 processor in 16K or more and compiles full Pascal with a few differences. The enclosed documents will explain more.

Taking your points in order:

1) John Ankcorn did most of the work on the compiler. He and I are the Pascallians here and can be reached at this phone.

2) All 11's. The compiler source has assembly conditionals to shape it for the desired machine.

3) RT-11, 16K words. We have an RSX-11M version in the works.

4) see enclosures

5) Our supplement is enclosed. We are working on more.

6) Probably will be one year of unlimited fixes and updates, followed by an annual subscription service.

7) Basically yes, but see the second page of the supplement.

8) Compiler (Pascal to Macro assembler text). Fits in 12K with the extra space taken by the symbol table. See the enclosures. John is preparing a paper describing many of the details.

9) Excellent. It has been used on our laser trimming systems for more than a year, and we have assiduously searched for and eliminated bugs.

10) The compiler is written in Macro-11. We started with the U. of Ill. bootstrap, changed the syntax scanner, totally changed the code generation, wrote our own expression analyzer, and wrote our own support package for arithmetic, math and file handling.

Effort has been on the order of 2 man-years, though some of this time was spent on the applications software for our systems. It was the first compiler for each of us, which is why we are grateful to the Illini for their initial help.

I am not sure any of this is directly suitable for publication, and I think you should resist the newsletter becoming an advertising forum. Nevertheless, we modestly think that ESI Pascal is probably the smallest, fastest, most complete and most reliable compiler for the PDP-11, and we are not about to hide our candle under a basket.

Timothy Bomham
University of Minnesota

Nov. 2, 1976

David Rowland
manager, programming
Mr. Timothy Bonham
Pascal Implementations
University Computer Center
227 Experimental Engineering Building
University of Minnesota
Minneapolis, Minnesota 55455

Dear Mr. Bonham:

In response to your letter of October 22, I have taken over responsibility for Pascal implementation here at Santa Clara from Dan Lewis. There has been essentially no progress on implementation since the last contact with George Richmond in May of 1975.

Current plans are to initially implement Pascal via Pascal-P on the University's HP3000/Series II, which is running under MPE with 256 K words of memory. A very rough completion date is January, 1978 (we hope to beat this, but given the realities of implementor time availability, January is as good a guess as any).

Following completion of this task, we intend to implement a (still undefined) subset of Pascal for the Department's HP2100, running under DOS III with 32 K words of memory. The implementation will be in Pascal and cross-compiled from the 3000.

I'll keep in touch as the implementation progresses.

Incidentally, I've enclosed my PUG membership-application.

Sincerely,

Ronald L. Danielson
Assistant Professor

RLD:dlm
Encl.

University of Waterloo

Mr. Andy Mickel, Editor
Pascal Newsletter
University Computer Center
227 Experimental Engineering
University of Minnesota
Minneapolis, MN 55455

Dear Sir:

Having just read the PASCAL Newsletter Number 4, with its list of PASCAL implementations, I thought I should draw to your attention the PASCAL implementation for the Honeywell 6000 and Level 66 Series machines which we completed for Honeywell earlier this year.

This compiler, an independent implementation of the full language which is not related to any previous PASCAL compiler, has been a commercial product of Honeywell Information Systems since May 1976. To my knowledge, it is the first PASCAL implementation to be officially distributed through and maintained by a major manufacturer.

Yours truly,

W. Morven Gentleman
Director

WMG:cm

(*note: manuals are available from Honeywell Sales*)
IBM SYSTEM 360/370 (this section also includes the Hitachi 8000 series and the Amdahl 470. see also Newsletter #5)

PASCAL 8000 Implementation Note

November 5, 1976

1. Implementor: Teruo Hikita and Kiyoshi Ishihata
Department of Information Science
University of Tokyo
Tokyo 113 Japan
phone 03-812-2111 ext 2947


3. Operating system: OS7 (not yet)

"Bootstrapping PASCAL Using a Trunk"
(These technical reports are available from the Department)

5. Documentation: (not yet decided)

6. Maintenance policy: (not yet decided)

7. Differences from Standard Pascal:
Standard procedures pack and unpack are not implemented.
Files must be declared at main program level.
A few novel language features are included.

8. Characteristics of the compiler:
Written in Pascal (about 5200 source lines).
Compiler object size is about 100 kbytes.
Compiling speed is about 350 line/sec.
Execution speed is comparable to FORTRAN-compiled objects.

9. Reliability of the compiler: Good

10. Method of development: Modified Naegeli's trunk compiler and bootstrapped it by Pascal-P
(about three man-months).

I believe this implementation differs from other 370-versions in two important ways, and therefore it might be of interest to the PASCAL User's Group.

1 The Environment

The compiler is implemented on a 370 model 155 with 256K bytes of main memory, under the CMS/VS operating system. CMS/VS is by far the most popular 370-operating system, but it is mainly used in well installations where business-oriented processing, in domination.

2 The Purpose of the Implementation

Our site is to use Pascal in a production environment where the bulk of work is in the file-processing field, i.e. the administration (economic and others) of a large hospital.

The only programming languages available are FORTRAN and COBOL, and since we have a good FORTRAN-millieu, FORTRAN is dominating, even in "pure" file-processing applications. Naturally we hope to replace them (at least partially) with Pascal. However, in spits of all the virtues of Pascal, it lacks the necessary facilities for our kind of applications. A number of features will have to be added to the language, and I will list a few which we consider to be most important.
To do the file-processing you must have file-handling routines, and since Pascal only supports sequential file-structures, extensions will have to be developed. Generally speaking, one of it say that we have two requirements:

- A way to explicitly control all forms of secondary storing, like an interface to all available access-routines for data transfer.
- A way to define the file and its structure.

PAS/US has no "file-name" where Job-Control-Language may be used to describe the file (record-length, 'blocking'-factor etc.). This forces us either to make our own file-name or with a special control language or to make modifications to Pascal's syntax.

We have chosen the first alternative, not because it is the "best" solution, but rather because we want to keep our version of the language compatible with the standard.

External Procedures

It is not only the support for separate compilation of Pascal procedures we consider to be important. Far more important is the support for the inter-language communication, to be able to call routines written in other languages, whether they are user-written application routines, library programs, operating system, database or data-communication software.

External Records

When external procedures are used, the data-transfer between them will create problems since global variables may not be accessed, and since the data-transfer through parameters has certain limitations. We have therefore introduced a new data type-External records.

In examples:

```pascal
    type
      arrrec = record
        .
      end;
    var
      n: expr;

    procedure example(var array: arrrec);
    begin
      .
    end;
```

Annexing the new (reserved) word external to an ordinary record type definition will cause all variables of that type to be allocated as separate modules. The name of each module is the variable-name. This allocation is static, the variable of the example is not allocated on the run-time stack, but the scope of the variables the procedure where it is declared. It is therefore best understood by comparing it with other FORTRAN/HAPPE COLON! (Our main reason for implementing it), or the JAVA INTERNAL attribute in P/S.

The features mentioned here are what we consider to be most important for us to implement, we would very much recommend on this, especially for users who (also to) use Pascal for file-processing applications, to learn now they solve similar or other problems.

Yours sincerely

Ivan Velev
Mr. Andy Mickel  
University Computer Center: 227 Exp Engr  
Minneapolis, Minnesota 55455  

Dear Andy,

We have been working on a PASCAL compiler for the 360/370 series for the past year. The compiler design was done by Dr. Jan V. Garwick, with implementation by Dr. Garwick, Paul Merillat and myself in PL360 and Chris Strachey's GPM. We are about 95% done, and it is a full compiler for PASCAL with the following exceptions:

1) GOTO's and labels are unsupported (and are flagged with a warning if used).  
2) UNPACKED arrays are not supported.  
3) Sets of characters are not allowed.  
4) Tag field specifications in NEW and DISPOSE are ignored, the record is allocated with the maximum space needed.  
5) Procedure or program segments each must not exceed 4K bytes.  
6) A predefined procedure, CLOSE, has been added to facilitate file operations.

Extensive compile time and runtime error checking is done. The runtime checking is optional, and the compiler will generate runtime checks by use of a toggle which may be set or reset at any time during compilation. There are extensive compile time facilities, including a reformatter and cross referencer.

We have been testing the compiler for about a month now, and the results are good. Runtime facilities are still undergoing debugging, but should be completely done by September 31. We are going to use the compiler in our introductory C.S. course, and hope to unearth pathologies that would not come out in use by an experienced PASCAler.

Distribution of an OS version is expected to start by January 1st, and Dr. Tom Marler of our C.S. Department will handle inquiries.

If anybody has any questions about the compiler (other than distributional), I would be glad to answer them.

We think the PASCAL newsletter is great, keep up the good work.

Sincerely,

Robert Knight, Director  
Office of User Services

RK:pq
STONY BROOK's PASCAL/360 - A STATUS REPORT - NOVEMBER 1976

The Stony Brook Pascal Compiler for IBM 360 and 370 computers is alive and well. As of November, 1976 more than 65 copies have been issued, and several installations are using the compiler under a live load. The compiler project continues at Stony Brook, and a second release is planned for January, 1977.

Release 1 was issued in June, 1976. It provides an almost complete implementation of Standard Pascal except for a variation in the means of specifying print field widths in the Write procedure. Not implemented in Release 1 are nonstandard .files and the standard procedure Dispose. The compiler has been successfully installed under the OS/360, NFT, V21 and V22 operating systems, and under VM/CMS with modifications to the OS interface. A DOS interface is nearing completion. At present, the main storage requirement is 160K bytes including space for file buffers.

The compiler is coded in XPL, with an assembler-coded monitor that provides the interface with OS. We do not have good statistics on compilation speed, but Release 1 has 1.93 CPU-second overhead to compile a trivial program on a 360/65. This is believed to be mainly due to the complexity of opening and closing files under OS/360.

The execution time of several compiled Pascal programs has been compared with that of equivalent translation into ALGOL, whose compiler is known to produce good, though not optimized code. The Pascal programs execute faster, in nearly all cases.

Although it would be foolhardy to allege that a compiler that has been field-tested for less than six-months is bug-free, we believe that the majority of errors have probably been corrected. The three updates have repaired all errors reported as of November 1, 1976, as well as improving the resiliency of the compiler in the presence of Pascal source program errors, and reducing the storage requirements from 180 to 160K bytes. Updates are issued in the form of source-language (XPL or BAL) patches to be input to a card-oriented editor. Both the editor and an XPL compiler are furnished on the distribution tape.

Present work is directed toward completing the implementation of nonstandard files, management of heap storage, and external compilation, all of which will be included in Release 2. This release, subject to later updates to correct errors and improve performance, will be the production version of the compiler.

Future work will be directed toward producing an edition specialized for student use. This will offer the same capabilities in a compile-and-go version, except for a limitation on the size of programs that can be compiled. The design target on the main storage requirement is 120K bytes. Compilation speed will be improved, primarily through the use of corefiles and interpass data communication buffers to reduce I/O.

The compiler already includes excellent syntax error recovery, intelligible error messages and runtime diagnostics that enhance its usefulness in education.

For those interested in acquiring the Pascal/360 compiler, the cost is $175.00, which includes distribution, complete system documentation (when available), and maintenance at least through August, 1977.

A 50-page User's Guide is available at a cost of $1.00 per copy in quantities of a dozen or more. The User's Guide is intended as a supplement to Jensen and Wirth, and tells everything that a user needs to know about the compiler.

At no cost whatsoever, one can obtain a packet giving additional information on the Pascal/360 compiler by sending a request to:

Pascal Compiler Project
Department of Computer Science
SUNY at Stony Brook
Stony Brook, New York 11794
Dear Andy,

I was quite surprised to see my last letter to you printed in the Newsletter. Nevertheless, since it did appear, I feel compelled to follow up on my comments about the Stony Brook PASCAL compiler for the IBM 5/370.

At the time I wrote the letter, the compiler was, indeed, buggy. However, response from them has been excellent. I have since received and installed two updates; the cover letter with the second stated that it fixed all reported bugs. I have since run at least one medium-to-large (700 statements) program using it, with no trouble. And the post-mortem histogram — showing how many times each statement was executed — is a most useful feature.

Complaints about the compiler? Sure, there’s always something that could be improved. The compiler is a bit too big (180K), and a bit too slow for small programs (high fixed overhead per compilation), and, perhaps most serious, they omitted the standard formatted-write notation. And it would be nice if the compiler wrote out standard OS-format object modules.

I should note that I ordered the Stony Brook compiler in preference to the Manitoba version, since it seems more suited to use with production-quality programs. Particularly serious restrictions (from my point of view) in the Manitoba compiler are its lack of I/O, its lack of a full version of NEW, and its restriction on the size of procedures (4K).

--Steve Bellovin

cc: William Barabash, SUNY at Stony Brook
Monsieur,

En réponse à votre lettre du 25 Octobre 1976 voici le point des travaux faits sur le compilateur Pascal.

- Il est opérationnel sur
  360/67 avec OS/MVT
  510/148 avec VS/MVT
- Demande REGION 220 K pour s'autocompiler.
- Distribution sur bandes magnétiques 9 pistes/800 bpi.
- Il existe un supplément au manuel du langage Pascal, décrivant l'implémentation sur IBM.
- Langage Pascal accepté est conforme au standard 74 à quelques exceptions près.
  Il manque READ/WRITE mais l'installation est prévue pour la fin 1976.
- Améliorations successives ont obtenues par compilation.
- La vitesse d'exécution moyenne est:
  
  | compilateur standard | 6000 lignes sources | 105 secondes CPU
  | compilateur "dope" | 6000 lignes sources | 84 secondes CPU

Ajouts non standards :
- Cf. manuel spécification 360
- procédures assembleur.

Le compilateur Pascal a aussi été installé en CP/CMS.

Je vous prie d'agréer, Monsieur, l'expression de mes sentiments distingués.

Olivier Lecarme of the Université de Nice, Laboratoire D'informatique, Parc Valrose, 06034 Nice Cedex, wrote us in a letter of 16 Sep 76:

"A Pascal compiler for the IBM 360, which was probably the first one, has been done in one of the Universities of Grenoble. Unfortunately, the people who made it had no time nor support for distributing it, although it seems to have impressive performances in execution time (but less good in storage needed for compilation). People to contact are Messrs. Henneron and Tassart (Informatique & Mathématiques Appliquées, B.P. 53, 38041 Grenoble-Cedex, France."

IBM 1130

Olivier Lecarme of the Université de Nice, Laboratoire D'informatique, Parc Valrose, 06034 Nice Cedex, in his letter of 16 Sep 76:

"Implementations for Pascal-P, Pascal-S and finally full Pascal have been done for the IBM 1130 and are in use at the University of Neuchatel (Centre de Calcul, Chantemerle 20, CH-2000 Neuchatel, Switzerland)."
ICL 1900  (an implementation exists)
2970   (an implementation is underway)

INTEL 8080  (we need more implementation information)

INTERDATA 7/16
Rod Steel of Tektronix, Inc., MS 60-456, P.O. box 500, Beaverton, OR 97707 reported: "If we can find the resources, we may bring up a P-compiler on the Interdata 7/16 at TEK."

Michael S. Ball of the Naval Undersea Center, San Diego, CA 92132, wrote in his report on the Univac 1110 implementation that the Center has cross compilers, running on a Univac 1110 and generating machine code for the Interdata 7/16, for both Concurrent Pascal and Sequential Pascal. See his report for more information.

INTERDATA 70  (no known implementations)

MICRODATA 800  (no known implementations)

MITSUBISHI MELCOM 7700  (an implementation exists)

---

MOTOROLA 6800

Pascal Implementations
University Computer Center
227 Experimental Engineering
University of Minnesota
Minneapolis, MN 55455

Gentlemen:

This letter is in response to your letter dated October 25 in which you requested PASCAL implementation information. Following are my responses to each of your ten points.

1. Implementor, maintainer:
   Before Nov. 26:
   Mark D. Rustad
   Moorhead State University
   Computer Center
   329 4th Ave. S.
   Moorhead, MN 56560

   After Nov. 26:
   Mark D. Rustad
   585 Harriet Ave.
   Apt. #13
   St. Paul, MN 55112

   As yet there is no distributor.

2. The implementation is specifically designed for the Motorola 6800-based MITS Altair 6800, but can easily be transported to any 8-bit machine (the Zilog Z-80 would be highly recommended).

3. Since implementation is not complete, precise information is not yet available, however, the compiler will definitely run on an 8-bit microprocessor with 32K bytes, a TTY and no disk capability. It is likely that the memory requirement will be somewhat under 32K.

4. No distribution since implementation is not complete.

5. No documentation available yet.

6. This compiler is, more or less, my hobby so a specific maintenance policy cannot be stated.
7. This implementation is of a subset of PASCAL which I call PASCAL-M (PASCAL for microprocessors). Due to the very limited resources of microprocessors, PASCAL-M does not include the following PASCAL features:
   a. no files - all I/O via READ, WRITE
   b. no REAL type
   c. no declared scalar types
   d. no variant records
   e. no LABEL section
   f. no COTO statement
   g. no WITH statement
   h. no FOR statement (use WHILE instead)
   i. no CASE statement (may be put back in)
   j. no run-time checks yet
   k. standard procedures are: READ, WRITE, NEW, RELEASE, READLN, WRITELN, ORD, CHR, EOLN, MARK

   It is possible that the final implementation will have the CASE statement reinstated and that I may produce additional implementations for those having more resources to include REAL and FILE types.

8. The compiler produces an interpreter code which is output onto an external medium such as paper tape which is then loaded with the interpreter for execution. The compiler is written in the subset of PASCAL which it compiles and is about 2200 lines of code. The compiler should compile usable programs in under 32K bytes. The compilation and execution speeds can not yet be tested.

9. The reliability of the compiler seems to be excellent.

10. PASCAL-H was developed from PASCAL-P2 and is being cross-compiled by Mike Ball's UNIVAC 1100 PASCAL. I would estimate that about two man-months have gone into this implementation and I expect that about one more man-month to complete it. I have found the PASCAL compiler much easier to work on and understand than I expected and I believe that this is attributable to the language it is written in (PASCAL).

   I will be preparing both documentation and reports on this implementation of PASCAL for publication once implementation is completed. For your information, all that remains is to debug the 'C-COR' (what I call my interpretive code, like P-COR) interpreter.

Sincerely,

Mark Rustad

--

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

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

PRIME P-400

Phillip H. Enslow of the School of Information and Computer Science, Georgia Tech, Atlanta, GA 30332, has informed us that Georgia Tech is bootstrapping a compiler for the Prime P-400 using Pascal-P4. The P-400 is a large "mini" with a 32 bit word, and 512 million words of hardware supported virtual memory for each of 64 possible users.

SEL 8600 (an implementation exists)

SIEMENS 4004/157

H.-J. Hoffmann of the Fachbereich Informatik, Techn. Hochschule, Steubenplatz 12, D-6100 Darmstadt, Germany, wrote us: "We have implemented PASCAL P2 in three different versions (fully interpretive, SC-code automatically translated to assembly language, code emitters for assembly language) for SIEMENS 4004/157 computer. Usage in some systems programming work."

TELEFUNKEN TR-440 (an implementation exists)

TEXAS INSTRUMENTS TI-ASC TI-980A (an implementation exists)
Mr. Andy Mickel  
University Computer Center  
227 Exp Engr  
University of Minnesota  
Minneapolis, MN 55455  

Dear Mr. Mickel:  

Thank you for the Pascal newsletter. I just got number 5 and enjoyed it greatly.  

As you know, we have a quite complete implementation of Pascal for the Univac 1100 series. I have enclosed some performance data on the compiler and generated code which you may find of interest. We kept the implementation as close as possible to standard Pascal, with extensions only to allow interface to the Univac Exec, and for compatibility with other systems whose code we wanted to use. The restrictions are essentially the same as those in the CDC compiler. We have been using the Pascal compiler for about nine months, and its reliability has been quite good. It should soon approach excellent.

We are using the compiler for general purpose programming and "systems" programming for the Univac and other machines. Its usage is steadily increasing, and is currently about 60 to 70 compilations a day. This compares to Fortran which is about 600, but of course, each Fortran subroutine is counted separately. User response has been quite favorable, and the interface to the user is at least as good as the rest of the language processors available for the 110 series. A large, but unknown, percentage of the use is interactive (demand node in Univac terms).

One major use of the system has been the development of compilers for Concurrent Pascal and Sequential Pascal for an Interdata 7/16. These compilers are based on those supplied by Per Brinch Hansen, and generate machine code for the 7/16. They are currently operating as cross compilers, running on the Univac 1110 and generating code for the Interdata. We are currently in the process of moving them to the Interdata for self-compilation. The project has been a very interesting exercise in machine independence, and the code which must be changed when moving the compilers from the Univac 1110, a 36 bit 1's complement machine, is surprisingly small. We have not measured it accurately, but it is on the order of one to two percent.

These compilers are highly optimizing compilers, and the direct machine code they generate is up to twenty percent smaller than the interpretive version generated by the original compilers. Since there was no attempt to make the interpretive code compact, this is not surprising. The next project along these lines is to modify the compiler to generate code for the Interdata 8/32.

One problem which we have is keeping up to date on various extensions and changes to the CDC compilers. As you mentioned in the newsletter, this compiler has served as an unofficial standard for compatibility, and we would like to know about things like the "VALUE" section before we see them in some code from another installation. Perhaps this data could be published in the newsletter as it becomes available? Since we are promoting the language as leading to portability, we should practice what we preach.

Finally, where can I obtain copies of the new documents from Wirth's group. I am particularly interested in the paper on Pascal-S.

Sincerely,

Michael S. Bell

PERFORMANCE OF THE PASCAL 1100 SYSTEM  
14 October 1976

The following performance was measured on a Univac 1110. All times given are totals, including both CAU time and CCER time.

1. Compiler Performance.

The compiler performance was measured as it compiled itself. The compiler is 7,434 lines of code, including comments and blank lines. It compiles into 34,875 words of code and literals. The library adds 5,912 words (including some data area), for a total of 40,787 words. The Univac compiler interface routines account for 4,605 words of the library. The data space allocated for the compiler is 16,104 words, and while compiling itself the compiler uses 8,068 words in the heap and 7,444 words in the stack.

The compilation rate is 105 lines per second with an output listing, and 110 lines per second without a listing.


The compiled code was compared with that generated by the NAUCLG and ASCII FORTRAN processors. For both Pascal and NAUCLG, tests were done both with and without run-time checks. The FORTRAN compiler never generates run-time checks, but does allow for three different levels of optimization. The normal mode provides no optimizations, and optional modes provide local and global optimizations. The local optimization mode was chosen as the standard of comparison, since the short test programs which were used provide an unusually simple case for the global optimizer, and allow it to perform much better than would be expected for the average program.

The programs used as a basis for comparison were taken from Wirth’s paper on the design of a Pascal Compiler. They are all programs which are easily written in all three languages, and so do not use the expressive power of Pascal. In addition, the time taken to call a simple procedure with four value parameters were measured for each processor. The results are summarized in the following tables.
Table 1. Procedure Call Times.

<table>
<thead>
<tr>
<th></th>
<th>PASCAL</th>
<th>NUALG</th>
<th>FORTRAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>26.4</td>
<td>108.6</td>
<td>21.9</td>
</tr>
<tr>
<td>Rel</td>
<td>1.21</td>
<td>4.96</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Table 2.

The program listed on the left side of Table 2 are:

- **PART** compute the additive partitions of a number (30 in this case) and print the results. This uses recursion for Pascal and NUALG, and a hand simulated stack for FORTRAN.
- **PARTNP** the same as above, but with no printing
- **SORT** sort an array of 1,000 numbers by a bubble sort
- **MATMUL** matrix multiply of two 100 by 100 matrices
- **COUNT** count the characters in a file and print the number of times each occurs. The file was 124,000 characters long.

The program listed on the left side of Table 2 are:

- **PART** compute the additive partitions of a number (30 in this case) and print the results. This uses recursion for Pascal and NUALG, and a hand simulated stack for FORTRAN.
- **PARTNP** the same as above, but with no printing
- **SORT** sort an array of 1,000 numbers by a bubble sort
- **MATMUL** matrix multiply of two 100 by 100 matrices
- **COUNT** count the characters in a file and print the number of times each occurs. The file was 124,000 characters long.

---

**VARIAN 620** (no known implementations)

V73

**California State University, Chico**
Chico, California 95929

Department of Computer Science
(916) 895-6442

November 2, 1976

Mr. Timothy Bonham
Pascal Implementations
University Computer Center
227 Experimental Engineering Building
University of Minnesota
Minneapolis, MN 55455

Dear Mr. Bonham:

Thank you for your interest in our activities at California State University, Chico.

Due to some staff changes, our Pascal project has not been completed. The implementation is planned on a Varian V73. Pending the completion of hardware changes, this project will remain stagnant for at least another year.

Sincerely,

Orlando S. Madrigal, Ph.D.
Chairman & Professor
Department of Computer Science

OSM:It

**XEROX Sigma 6, Sigma 7, Sigma 9** (see also CII 10070)

Olivier Lecarme, Universite de Nice, Laboratoire D'Informatique, Parc Valrose, 06034 Nice Cedex, France, in his letter of 16 Sep 76:

"A complete and standard compiler for the Xerox Sigma 6.7 and 9 has been done by Pierre Desjardins, who can give you all desirable information. Anyway, it seems to be a very good implementation, especially in the domain of compatibility and conformity to the standard."

("We do not have Pierre Desjardins's correct address, can someone help? ")
Clip, photocopy, or reproduce, etc. and mail to: Pascal User's Group
c/o Andy Mikel
University Computer Center
227 Exp Engr
University of Minnesota
Minneapolis, MN 55455
(phone: (612) 376-7290)

// Please renew my membership in the PASCAL USER'S GROUP for the next academic year ending June 30. I shall receive all 4 issues of Pascal Newsletter for the year. Enclosed please find $4.00. (*When joining from overseas, check the Newsletter POLICY section for a PUG "regional representative".*)

// Please send a copy of Pascal Newsletter Number _____. Enclosed please find $1.00 for each. (*See the Newsletter POLICY section for issues out of print.*)

// 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 USA

return postage guaranteed