

PASCAL USER'S GROUP

USER'S  
GROUP

# PASCAL NEWSLETTER

NUMBER 8

COMMUNICATIONS ABOUT THE PROGRAMMING LANGUAGE PASCAL BY PASCALERS

MAY, 1977

## TABLE OF CONTENTS

#	0	POLICY	#
*			*
#	1	EDITOR'S CONTRIBUTION	#
*			*
#	2	HERE AND THERE WITH PASCAL	#
*	2	News	*
#	3	Conferences	#
*	6	Books and Articles	*
#	7	Applications	#
*			*
#	8	ARTICLES	#
*			*
#	8	"Development of a Pascal Compiler for the C.I.I. IRIS 50. A Partial History." - Olivier Lecarme	#
*			*
#	11	"A Further Defence of Formatted Input" - B. A. E. Meekings	#
*			*
#	12	"Proposals for Pascal" - George H. Richmond	#
*			*
#	15	"A Proposal for Increased Security in the Use of Variant Records" - William Barabash, Charles R. Hill, and Richard B. Kieburtz	#
*			*
#	16	"Update on UCSD Pascal Activities" - Kenneth L. Bowles	#
*			*
#	18	"Some Comments on Pascal I/O" - Chris Bishop	#
*			*
#	19	OPEN FORUM FOR MEMBERS	#
*	22	Special Topic: Standards	*
#	40	IMPLEMENTATION NOTES	#
*	40	Checklist	*
#	40	General Information	#
*	40	Software Writing Tools	*
#	40	Portable Pascals	#
*	42	Feature Implementation Notes	*
#	44	Machine Dependent Implementations	#
*	64	Index	*
#	65	ALL PURPOSE COUPON	#

**RENEW!!**

## PASCAL 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, general purpose language with a small and systematic 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/227 Exp Engr/  
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/The University/  
SOUTHAMPTON/S09 5NH/United Kingdom/ (telephone 0703-559122 x2387)

## PASCAL NEWSLETTER POLICIES

The Pascal Newsletter is the official but informal publication of the User's Group. It is produced quarterly (usually September, November, February, and May). A complete membership list is printed in the November issue. Single back issues are available for \$1 each. Out of print: #s 1,2,3,4 \*  
POLICY

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 software writing tools for a Pascal environment.

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

ALL THE NEWS THAT FITS, WE PRINT. PLEASE SEND WRITTEN MATERIAL TO THE NEWSLETTER SINGLE SPACED AND IN CAMERA-READY FORM. USE NARROW MARGINS; (LINE WIDTH 18.5 CENTIMETERS). REMEMBER, ALL LETTERS TO US WILL BE PRINTED UNLESS THEY CONTAIN A REQUEST TO THE CONTRARY.

- Andy Mickel, editor, John P. Strait, associate editor, April 26, 1977.



UNIVERSITY OF MINNESOTA  
TWIN CITIES

University Computer Center  
227 Experimental Engineering Building  
Minneapolis, Minnesota 55455

(612) 376-7290

This issue with so many important topics is late. I think that George Richmond deserves another round of thanks for the early work he did on Pascal Newsletter. With this, the fourth issue I've done, I have to say that it is a lot of work. Without Sara Graffunder and Jim Miner, who edited the Here and There and Implementation Notes sections respectively, this issue would not have appeared.

#### \* RENEW

We've lowered the cost of PUG membership by keeping the price the same (\$4<sup>1977</sup> < \$4<sup>1976</sup>)!

This is the last (and first) renewal notice you'll get. Please renew, especially if you think we are doing some good in the world. If you are not reading your own copy of the Newsletter, why not help us out: join for yourself (we need more members to keep the price the same). Just think of it as giving up eating out one night in the next year. And we won't refuse additional (no strings attached) contributions!

#### STANDARDS

See the Open Forum section for a series of letters.

#### MICROPROCESSOR Pascal

See the Here and There News section under Charles Bacon, P.M. Lashley, Steve Legenhausen, Andy Mickel, David A. Mundie, and see Implementation Notes under both "Comment: Microprocessors" and under individual specific manufacturers's names. And Ken Bowles's article.

#### Pascal Newsletter #9.

Deadline for written contributions is July 15. Changes in POLICY: #4 is now out of print. All written material must now be single spaced and typed with narrow margins. We are running out of room!

#### THIS ISSUE (#8)

Unfortunately we have had to cut material from this issue ("all the news that fits..."). George Richmond sent a 5 page bibliography which we couldn't find room for. It had only 15 new entries over his last one in #4, and is incomplete these days if you keep up with Pascal Newsletter. We were also unable to print a Roster increment as we did in #7. I regret this because it is the roster which enables Pascalers to get together especially if they are in the same area. This time the number of new members totals 345! It would have taken 6 full pages to print in a new compressed format! We just couldn't afford it. We also had to reformat every contribution to save space, and omit extraneous material.

But, we have no shortage of material (unlike the disease which afflicted the FORTRAN Bulletin, the LISP Bulletin, the SNOBOL Bulletin, etc.).

We have had many suggestions regarding the newsletter. We want to keep it informal and interesting and prevent its degeneration into a slick, useless, "professional" journal.

#### PUG and Pascal Newsletter Mechanics

PUG now has 943 members in 29 countries and 47 states! We need more members to stay financially solvent (we are currently in the black, barely) and we need them as well as renewals early in the academic year (preferably before August 15). I now strongly disagree with my earlier idea (and Mike Hagerty's letter in this issue) of becoming affiliated with ACM (like STAPL under SIGPLAN). Did you know that according to Garth Foster (January 6, 1977) STAPL (SIGPLAN Technical Committee on APL) only had 973 members after more than 5 years in existence? If we affiliated with ACM, the price would probably double, but we'd be compensated with fancy letterhead on the stationary.

## EDITOR'S CONTRIBUTION

PUG has a broad base with many non-academic members. We have kept the price low, publicized PUG in unconventional ways (unlike ACM) and in the process have become known in industry where the real changes can be made. We just completed our fourth mass mailing (350) on March 28 to the holdouts from George Richmond's old mailing list from newsletters 1-4.

I would like to encourage all PUG members to use their imaginations in making Pascal and PUG more visible. Write letters to the editor of popular trade journals such as COMPUTERWORLD, DATAMATION, COMPUTING EUROPE, etc. Distributors of compilers should send an All Purpose Coupon to each recipient of their implementations. Write to SIGCSE (Pascal's strong point is Computer Science Education). I can't do all of these things.

I've noticed some big discrepancies in PUG membership at several universities which have a fair amount of Pascal users. It seems that some local people have not done all they can to tell their users about PUG. Why is it for example that at the University of Minnesota there are 48 PUG members, at Lehigh University 13, at Indiana University, the University of Texas, and the Technical University of Berlin 7, and at the University of Illinois, Georgia Tech, the University of Southwestern Louisiana, Cornell, and the Imperial College London, etc. there are 6 PUG members while at the University of Colorado there is only 3, the University of Washington only 2, and at the University of Manitoba, SUNY Buffalo, and the University of Massachusetts only 1?

#### BACK ISSUES

I'm sorry that we are slow, but we are not in the publishing business. As I stated in #7 we have had terrific growing pains resulting from not realizing back in September how popular PUG was going to be. We are temporarily out of print with #5 and this holds up mailing 5,6, and 7 to new members as we cannot afford postage for separate mailings. As it is, it is very expensive to mail back issues. At PUG "central" here in Minnesota, we have no secretaries. John and I (with help from people like Sara and Jim) have opened all our own mail, answered with personal notes all inquiries, handwritten most addresses on envelopes, handled all the typing, mailing of back issues, filing, accounting, the mailing label data base; and sent invoices and bills to persons who haven't paid. That's right, we never planned on some people not paying. Those who still owe PUG money are: Bengt Norstrom, Lars Magnusson, Bernhard Nebel, Roland F. Bloemer, Stanley B. Higgins, Karl J. Astrom, Wayne Fung, John S. Sobolewski, T. Hardy, Ada Szer, and John Nolan. This is as of today, and I wouldn't be surprised to see their money soon, and I don't in any way want to imply that each does not eventually intend to pay!

#### SUMMARY

I want to thank all of those who have helped this year, especially Judy Mullins, David Barron, Carroll Morgan and Tony Gerber (who have enabled Australasian re-mailing with zero compensation) and Tervio Hikita for remailing #7 to Japanese members. Finally many thanks are due to the University Computer Center here at the University of Minnesota, particularly Peter Patton, our director, and Lawrence Liddiard our associate director for systems for enabling PUG to thrive.

*Andy*

- April 26, 1977

# HERE AND THERE WITH PASCAL

## NEWS (ALPHABETICAL BY LAST NAME)

Charles Bacon, 10717 Burbank Dr., Putomac, MD 20854 (PUG member): "I am interested in a Pascal running on a RSX-11M system as well as on the KI-10...also on any 8080 system." (\* 1/10/77 \*)

Mark Becker, 300 Collingwood Ave., Fairfield, CT 06432 (PUG member): "I'd like to locate a version of PASCAL for the PDP 11 that does not use or require the Floating Point Processor." (\* 1/31/77 \*)

(\* From the newsletter of the University Computer Center at the University of Southern California, 1020 W. Jefferson Blvd., Los Angeles, CA 90007: UCC has added several JCL procedures (for its IBM 370 system) so that users can invoke the University of Manitoba version of Pascal. The procedures perform one-step monitor; compile; compile, load and go; compile, linkedit; compile, linkedit, and go; load and go; linkedit and go; and compile and punch an object deck. 1/1/77 \*)

Gary Boss, 517 N. 7th St., Bismarck, ND 58501 (PUG member): "I am interested in knowing about chess programs written in Pascal."

Kevin W. Carlson, 1531 Simpson St. Madison, WI 53713 (PUG member): (\* Wants to know if there is a group of Pascal users in or near Madison. 2/9/77 \*)

C. R. Corner, 514 S. 9th St., Moorhead, MN 56560 (PUG member): "I'm trying to implement Pascal on the PDP-8 and on the PDP-11. Any suggestions?" (\* 3/1/77 \*)

Frederick C. Cowan, The Aerospace Corporation, Mail station A2-2043, P. O. Box 92957, Los Angeles, CA 90009 (PUG member): "I am interested in the mods to make [release 2 of PASCAL 6000-3.4] run on the 7600 under Scope 2.2." (\* 5/18/77 \*)

Mattia Hmeljak, Ist. di Elettronica ed Elettronica, Università di Trieste, Trieste, Italy (PUG member): "In Trieste University a CDC computer exists and a Pascal compiler is implemented there.

We have also an HP-2100 mini-computer and we would like to run some programs there for teaching and for research. For these reasons we intend to implement the Pascal compiler on this machine.

As a first step... we intend to write a P-code interpreter using the Pascal-written interpreter and translating it into H-P Algol. Therefore we would be glad to know if someone else is working to implement Pascal on the same mini-computer... We thank you also for any information you will consider useful to give us for our work." (\* 2/5/77 \*)

Stanley B. Higgins, Dept. of Medicine, Vanderbilt University, Nashville, TN 37232 (PUG member): "... our group operates a PDP-11/40, PDP-11/34 and a PDP-11/55... software... by DEC... RT-11 and RSX-11M operating systems... We would be most interested in knowing of [Pascal compilers]." (\* 2/23/77 \*)

Robert L. King, 1452 Sandra Dr., Indicutt, NY 13760 (PUG member): "If possible, please forward information on free or very inexpensive Pascal compilers for an IBM 370/178 under VSI with 3330's and 9-track tapes." (\* 2/1/77 \*)

Joseph Lachman, Computer Center, University of Illinois at Chicago Circle, Box 4348, Chicago, IL 60680 (PUG member): "... At present the UICC computer center has no Pascal compiler. Any advice you could offer us relative to the availability, quality, and costs of PASCAL compilers that will run on IBM/370 or DEC PDP-11 computers would be greatly appreciated." (\* 4/5/77 \*)

J. Larmouth, Director, Computing Laboratory, University of Salford, Salford M5 4WT, England (PUG member): "Having moved to Salford from Cambridge, I have ceased work on Pascal. Unfortunately there was nobody available at Cambridge to continue the work. so

that our efforts towards a 370 implementation should be considered abandoned.

"We did produce and distribute an interpreter system but Cambridge... does not have the man-power to continue even this service.

"Sorry this is all so negative. My interest in Pascal remains, although you might be interested to know that I am perhaps more interested in TUCLID, as would, I think, be most members of PUG if they knew more about it." (\* 1/5/77. For information about tuclid, consult B. W. Lampson, et. al., "Report on the Programming Language tuclid," SIGPLAN Notices, 12:2 (February 1977); and G. J. Popek, et. al., "Notes on the Design of TUCLID," SIGPLAN Notices, 12:3 (March 1977), 11-19. \*)

P. M. Lashley, Director of Computing CSCS, POB 764, 114 S. Bullard St., Silver City, NM 88061: (\* From a letter to the editor of Byte, 2:2 (February 1977), 77-78 \*) "I write primarily in response to Mr. Skye's letter in your August issue. I can only conclude that he had been with IBM too long, otherwise he would not attempt to debase the 8080 with FORTRAN or PL/1. FORTRAN is a virtual pterodactyl, flying solely by inertia, whereas PL/1 is much better, but too rambling in construction. If he indeed takes up the admirable task of writing a high level compiler for the 8080, he would be better advised to base his compiler on a fully structured language such as PASCAL." (\* The letter goes on for several paragraphs. \*)

Steve Legenhausen, 12 Barnard Street, Highland Park, NJ 08904 (PUG member): "I think it is absolutely important that persons promoting Pascal realize the danger of BASIC's becoming the permanent and only language on microprocessors. One only has to pick up any issue of the computer hobbyist magazines such as Dr. Dobbs Journal, Byte, Kilobaud, Creative Computing, etc., to find that each is filled with BASIC. Some effort should be put forth to promote Pascal in this medium." (\* 12/31/76 \*)

Chris P. Lindsey, Computing Coordinator, Harvey Mudd College, Claremont, CA 91711 (PUG member): "Do you know of a well-documented, error free version of PASCAL which runs on a DECsystem-10 with a KA processor?" (\* 1/77 \*)

R. A. Lovestadt, 20427 SE 192, Renton, WA 98055 (PUG member): "Any PASCAL work on an HP5000?" (\* 2/10/77 \*)

William Lyczko, Software Development, NCR Corporation/Terminal Systems, 950 Dunby Road, Ithaca, NY 14850 (PUG member): "I am interested in any information you may have on implementation of PASCAL for microprocessors." (\* 1/7/77 \*)

Philip J. Malcolm, former address Zeus-Hermes Consultants Ltd., Shropshire House, 2-10 Copper Street, London; new address c/o Bank of Adelaide, 11 Leadenhall St., London EC3V 1LP, England (PUG member): "Zeus-Hermes is... investigating the possibility of adopting a Pascal -- or Modula -- type language for in-house development of mini- and micro-computer software across a broad range of target machines.

"Ideal would be a compiler:

written in its own source language; and executable on a microcomputer (with say 32-64K bytes of RAM, diskettes); and easily transportable to different target machines; and relying on a very small run-time monitor/support package.

"We would be delighted to hear from those possessing or working towards such a system." (\* 1/3/77 \*)

Andy Mickel, Univ. Computer Center, 227 Exp. Inqr., U. of Minnesota, Minneapolis, MN 55455 (PUG member) reports receiving an educational questionnaire from Intel about computer courses and micro-processors. The question, "What programming languages are used?" contained the check-off answers Fortran, Algol, PL/1, PL/M, Basic, and Pascal. Not included were Cobol, Lisp, Snobol, etc. (\* Andy's response to the catch-all question at the end was, "When are you going to support Pascal or a Pascal-subset and give up on Basic?" \*)

David A. Mundie, French Department, 302 Cabell Hall, University of Virginia, Charlottesville, VA 22903 (PUG member): "Is Zilog really making a microprocessor that executes PASCAL constructs as its machine-level language (Byte, v.2, no. 4, April 1977, p. 140)?" (\* 4/3/77 Will a PUG member please write Zilog to ask, then send the answer to the newsletter? \*)

Mark O'Bryan, Computer Center, Western Michigan University, Kalamazoo, Michigan 49001: ". . . I'm in charge of PASCAL implementation and maintenance at WMU. We have an old version of NAGSL's compiler for the PDP-10 and will be releasing it for use here in early March. I'll keep you informed on user reaction when it happens.

Gene H. Olson, 421 County Road J, Apt. 512, Hopkins, MN 55343 (PUG member): "The best argument against formatted reads has yet to appear in the PUG newsletter. In processing large amounts of formatted data (the supposed rationale of formatted reads) keypunch or similar errors cause both formatting and content errors which render formatted reads useless. In other words, in a production environment, the program must check data character-for-character as it is coming in." (\* 2/25/77 \*)

Jerry L. Ray, 21320 Oldgate Rd., Ithaca, NE 68022 (PUG member): "I am attempting to sell the idea of using PASCAL instead of FORTRAN as a first language in a Computers & Business course. Any information to support my argument (institutions using PASCAL, etc., as well as the structure aspects) would be greatly appreciated." (\* 4/8/77 \*)

R. Waldo Ruth, Computer Science Dept., Taylor University, Upland, IN 46989 (PUG member): ". . . I would also like to know about the availability of a PASCAL package to run on DEC 11 systems under RSTS or RT-11." (\* 2-24-77 \*)

Carl W. Schwarcz, Digital Equipment Corp., MR 1-2/t27, 200 Forest Street, Marlboro, MA 01752 (PUG member): ". . . While employed by Control Data I was responsible for the design and implementation of two compilers for a Pascal-based programming language ('the Software Writers' Language') for the Cyber 170 and Cyber 270." (\* 1/25/77 \*)

Arthur I. Schwarz, Hughes Aircraft Co., Bldg. 150/MS A222, Culver City, CA 90230 (PUG member): "Our installation is currently interested in gaining some expertise in using PASCAL. We would like to obtain a compiler for use on our Sigma 9 computer, or lacking this, a compiler with accessible code generators for either the CDC or IBM computer lines." (\* 2/8/77 \*)

Wayne Seipel, Box 8259 U.T. Station, Austin, TX 78712 (PUG member): "The University of Texas Computer Science department needs a PASCAL compiler for a [Data General] Nuva 30. The department has just purchased 2 processors, each with 32K words of memory and a 10Mega-byte disk. These will be used by both graduate and undergraduate students in a hands-on environment. Current plans call for the development of an operating system, and a PASCAL compiler would make life orders of magnitude easier. Any information on a compiler (compiled, standard, PASCAL1, or PASCAL2) will be greatly appreciated.

Contact either James Peterson, Computer Science Dept., University of Texas, Austin, TX 78712, or myself." (\* 3/14/77 \*)

Kevin Weller, 147 Cornell Qtrs., Ithaca, NY 14850 (PUG member): "How anyone implemented PASCAL on a PDP 11/45? (Is a Jovial interpreter available?)" (\* 1/21/77 \*)

Nicholas Wybolt, 576 Lau Street, Hillside, NJ 07205 (PUG member): "Here at NJIT, Pascal is beginning to be used in a junior-level course in algorithms and data-structures; there is also individual interest in Pascal among faculty members and the student body.

"The student branch of the ACM is attempting to act as a medium of information in this matter. We are interested in your group and any related publications and activities. . . ." (\* 2/4/77 \*)

(\* From a press release by the U. S. Department of Defense distributed by the British Computer Society, March 22, 1977, on "The U. S. Department of Defense High Order Language Effort," to reach a consensus on a common high order language for embedded systems, p. 8 \*):

"Without exception, the following languages were found by the evaluators to be inappropriate to serve as base languages for a development of the common language: FORTRAN, COBOL, YACPOL, CMS-2, JOVIAL 373, JOVIAL 339, SIMILA 87, AEDOL 40, and CORAL.

"Proposals should be solicited from appropriate language designers for modification efforts using any of the languages, PASCAL, PL/I, or BASIC 68 as base languages from which to start. These efforts should be directed toward the production of a language that satisfies the DoD set of language requirements for embedded computer applications."

# HERE AND THERE WITH PASCAL

## CONFERENCES

International Federation of Information Processing Societies (IFIP), August 8-12, 1977 in Toronto. (\* Would a PUG member who is there organize and publicize a Pascal User's Group gathering. We would, but we won't be there. Also, send in a resume of the meeting for Newsletter No. 9. Thanks. \*)

ACM '77, Seattle, Washington, October 17-19, 1977. (\* The same here for Newsletter No. 10. \*)

REPORT on the Third Annual Computer Studies Symposium at Southampton (March 24-25)

### "PASCAL - THE LANGUAGE AND ITS IMPLEMENTATION"

A little over halfway in this whirlwind, 48 hour happening, the medieval banquet began. David Barron (the baron) and Judy Mullins (the baroness) enjoyed the honor of reigning over and hosting the attendees; it was a delightful time indeed.

And so was the whole symposium! I must commend Judy for organizing the symposium down to the last detail and thank David for making it a reality. It was a success by several different measures. Around 134 persons attended. The proceedings officially listed (including speakers and last minute replacements): Austria 3; Belgium 4; Canada 1; Denmark 7; France 4; Germany 16; Great Britain 72; Ireland 8; Netherlands 2; Sweden 9; Switzerland 5; and the USA 3; The proceedings contain the texts of all 11 presentations and will be published later this year (see Books section). All except Per Brinch Hansen's which will appear in an IEEE publication.

David Barron, U of Southampton, opened the symposium with a talk entitled "Perspectives on Pascal" which looked at the past, present and future and concluded with a call to join a "Society to Combat Well-meant Attempts to Change Pascal (SCWACP)."

Urs Ammann, ETH, Zurich, was introduced as the great-grandfather of all Pascal compiler writers and summarized his work over the last 6 years in "The Zurich Implementation."

Jim Welsh, Queen's U, Belfast, likewise introduced as the grandfather of Pascal compiler writers detailed development and performance of "Two ICL 1900 Pascal Compilers."

David Watt, U of Glasgow, presented an extensive description of "A Pascal Diagnostics System" for the ICL 1900 implementation.

Mike Rees, U of Southampton, presented a description of the Pascal compiler effort on the ICL 2970 underway for the past 9 months in "Pascal on an Advanced Architecture."

Judy Mullins, U of Southampton, did not dream up hypothetical architecture, but rather critically combined existing architectural features in designing "A Pascal Machine?"

The next day began with Per Brinch Hansen, U of S. California describing his "Experience With Modular Concurrent Programming" and his opinions of the future.

Pierre Desjardins, U of Montreal, substituted for Olivier Lecarme, U of Nice, and sketched an overview of "Pascal and Portability" issues.

Brian Wichmann, National Physical Laboratory, Middlesex, coalesced various aspects on "The Efficiency of Pascal" in comparison to other languages and in different environments.

Graeme Webster, Teeside Polytechnic, advised others who introduce Pascal into the curriculum with a talk on "Pascal in Education."

There were two discussion sessions. Brian Wichmann led the first on "Pascal on Minis and Micros" and I introduced the second on "The Future of Pascal" concerning standards and extensions issues.

In between time, the opportunity to talk and argue with other Pascalers from so many places was a real treat for all, I'm sure. I managed to meet 48 people, and in the process confessed to Urs that it was hard to get used to intense, sudden exposure to so many cultural backgrounds.

Perhaps the long range accomplishment of the symposium was to pass on a consensus to the rest of us in PUG regarding standards. See OPEN FORUM.

- Andy Mickel, April 17, 1977



**Third Annual Computer Studies Symposium  
"PASCAL - the LANGUAGE and its IMPLEMENTATION"  
University of Southampton, March 1977**

SYMPOSIUM ATTENDEES, (127 pictured here; not all names and faces known together!): A full list of names appears in the symposium proceedings.



**Third Annual Computer Studies Symposium  
"PASCAL - the LANGUAGE and its IMPLEMENTATION"  
University of Southampton, March 1977**

**SYMPOSIUM SPEAKERS, (pictured from left to right):** David Barron, Per Brinch Hansen, Andy Mickel, Pierre Desjardins, Graeme Webster, David Watt, Mike Rees, Urs Ammann, Brian Wichmann, Jim Welsh, and Judy Mullins.

## BOOKS AND ARTICLES

(\* D. W. Barron, working with Rich Stevens, has offered to take over this section. What follows is a notice of the policy for the section, beginning with No. 9 \*)

### POLICY

In this section we shall try to keep PUG members up-to-date with the PASCAL literature under the general headings languages, Textbooks, Implementation, Applications. At the least we shall give a brief citation of title, author and publisher. If possible we shall include a brief abstract and, if the importance warrants it, a critical review. In addition from time to time we shall give (hopefully) complete annotated bibliographies of selected areas: with the feedback from PUG members we should be able to build up a really comprehensive guide to the PASCAL literature.

Books and papers in the established journals are fairly easy to keep track of, but internal reports present much more difficulty. If you (or your institution) produce a report that you are willing to circulate, please send me a copy of the title page, or better still a copy of the report. The address is:

David Barron	or	W. Richard Stevens
Pascal User's Group (U.K.)		Kitt Peak National Observatory
Department of Mathematics		P. O. Box 26732
The University		Tucson, AZ 85726
SOUTHAMPTON, SO9 5NH		U.S.A.
U.K.		

As with the rest of PUG, the success of this enterprise will rest largely on the enthusiasm and help of the membership.

10 February 1977

David Barron

(\* The policy begins with the next issue. What follows is our new information about books and articles, and a review. \*)

### BOOKS

A Concurrent Pascal Compiler for Minicomputers, by Al Hartman, to be published by Springer-Verlag as Volume 50 in their Lecture Notes in Computer Science. Probably available by the end of April 1977. (\* Al writes that the book will be of especial interest to "... any of your membership using the Concurrent Pascal or Sequential Pascal compilers developed at Caltech for the PDP-11/45 minicomputer." \*)

Introduction to Computer Science, by Ken Bowles (U. of California, San Diego), to be published by Springer-Verlag in October 1977. (\* The book is computer graphics oriented and uses Pascal as the teaching vehicle. Note the change of title from our citation in No. 5. \*)

Introduction to Programming and Problem Solving with PASCAL, by G.M. Schneider, D. Perlmutter, and S. Weingart, to be published in hardback by Wiley and Sons in January 1978. A camera-ready manuscript of the book can be obtained by writing

Gene Davenport, Editor  
John Wiley and Sons Publishers  
605 Third Avenue  
New York, NY 10016

The manuscript may, with written permission, be duplicated for class use until the publication of the book.

Pascal--the Language and its Implementation, proceedings of the Symposium in Southampton, March 24-25. (\* At press-time, there is as yet no definite publisher and publication date. Perhaps details will be settled in time for publication in No. 9 \*)

Structured Programming and Problem Solving with PASCAL, by Richard Kieburtz, Department of Computer Science, SUNY at Stony Brook, Stony Brook, NY 11794, to be published by Prentice-Hall sometime in 1977. (\* This is rumored; we aren't sure of the title, etc. We hope we'll have the facts in time for No. 9. \*)

## ARTICLES

"Efficient Implementation and Optimization of Run-time Checking in Pascal," by Charles N. Fischer and Richard J. LeBlanc, SIGPLAN Notices, 12:3 (March 1977); 19-24. (\* from the abstract \*): "Complete run-time checking of programs is an essential tool for the development of reliable software. A number of features of the programming language PASCAL (arrays, subranges, pointers, record variants (discriminated type unions), formal procedures, etc.) can require some checking at run-time as well as during compilation. The problem of efficiently implementing such checking is considered. Language modifications to simplify such checking are suggested. The possibility of optimizing such checking is discussed."

"Proceedings of the All-Union Symposium on Implementation Techniques for New Programming Languages, Novosibirsk 1975."

(\* This publication came to us from David Barron, who received it from PUG member S. Pokrovsky, Computing Centre, USSR Academy of Sciences, Novosibirsk 630090, USSR. Most of the articles are in Russian, but the number of bibliographical references to publications about Pascal lead us to believe that the articles might be of interest to PUG members. Would someone who reads Russian (easily) volunteer to read and abstract the relevant articles for No. 9? We'll send a copy of the journal to you if you write to us in Minneapolis. The abstracts could go to David Barron for the next newsletter. \*)

"Programming Languages: What to Demand and How to Assess Them," by Niklaus Wirth, Berichte des Instituts für Informatik, t. T. H. Zurich, No. 17 (March 1976), 1-24.

(\* from the abstract \*): "The software inflation has led to a software crisis which has stimulated a search for better methods and tools. This includes the design of adequate system development languages."

This paper contains some hints on how such languages should be designed and proposes some criteria for judging them. It also contains suggestions for evaluating their implementations, and emphasizes that a clear distinction must be made between a language and its implementation. The paper ends with concrete figures about a Pascal implementation that may be used as yardstick for objective evaluations."

An extract from "Professor Cleverbyte's Visit to Heaven," by Niklaus Wirth, Berichte des Instituts für Informatik, t. T. H. Zurich, 17 (March 1976), 25-31. (\* To appear in Software Practice and Experience \*)

(\* from the abstract \*): "The following fable is a grotesque extrapolation of past and current trends in the design of computer hardware and software. It is intended to raise the uncomfortable question whether these trends signify real progress or not and suggests that there may exist sensible limits of growth for software too."

"The Software Development System," by C. G. Davis and C. R. Vick, IBM Transactions on Software Engineering, 3:1 (January, 1977), 69-84. (\* A summary by PUG member Nick Sointeff, who sent in the citation \*): Implementation of PDL-2, an extension of Pascal, to, among other things, include concurrent processing. Are also writing an OS in PDL-2.

"Some High-level Language Constructs for Data of Type Relation: An Investigation based on Extensions to Pascal," by Joachim W. Schmidt, Bericht Nr. 31, Institut für Informatik, Hamburg, January 1977.

(\* from the abstract \*): "For the extension of high-level languages by data types of mode relation, three language constructs are proposed and discussed:  
- a repetition statement controlled by relations  
- predicates as a generalisation of boolean expressions  
- a constructor for relations using predicates."

The language constructs are developed step by step starting with a set of elementary operations on relations. They are designed to fit into PASCAL without introducing too many additional concepts." (\* These extensions, which process relational data bases, are being experimentally implemented in Nagel's DTC-10 compiler at Hamburg \*)

## BOOK REVIEW

INTRODUCTION TO PASCAL, C.A.G. Webster, Heyden and Son, 1976.  
No. of pages: 129. Price: \$5.50, \$11.

For several years now there has been an increasing need for an introductory text on programming which uses Pascal as the vehicle. Unfortunately, Webster has not given us that book. The following may indicate why.

In the preface the author claims coverage of an "essentially full version of the language, discussing where appropriate ... the original report and its revision (sic)". In fact the book describes the original (1971) language and supplements this with incomplete and inaccurate summaries of the 1972 revision. No warning is given that the language has developed vigorously since 1972. This makes the book almost useless in conjunction with compilers for the latest version of the language, Standard Pascal.

One might expect that a book on Pascal would pay some heed to modern ideas of programming methodology. Instead we find algorithms introduced by machine-language programs and illustrated by "spaghetti" flowcharts. The concept of stepwise refinement ("top-down programming") is not mentioned until three quarters of the way through the book and then only in the context of procedure declarations. No substantial guidance is given in vital areas such as program design, testing, debugging, correctness and maintenance. It might almost be a book on BASIC!

These global defects are compounded by a list of errors of fact, omission and commission which leaves a blemish in almost every page. The following are just a few of the more serious.

(a) Variable parameters of procedures are (wrongly) said to be passed by reference, in a section which manages to make the (very simple) parameter-passing rules of Pascal seem almost incomprehensible in their complexity.

(b) Several examples of bad practice in the use of real arithmetic are hardly compensated by a superficial warning about the comparison of real values.

(c) The operator NOT changes the state of the following operand, according to Chapter 4.

(d) Chapter 6, under the heading "Initialising variables", describes the definition of symbolic constants. The initialization of variables is also described, but without warning that the feature is not part of the defined language.

(e) There are many lexical, syntactic and logical errors in the programming examples, some of them seemingly calculated to cause the maximum confusion for the beginner. For example, despite a warning early-on about the precedence of relational operators in Pascal, almost all the more complex Boolean expressions in the book are wrongly bracketed (or not bracketed at all). The following, given as a way of skipping characters up to a certain point in the end of file, is quite typical:

```
REPEAT UNTIL (input) AND NOT eof (input)
```

In short, avoid this book.  
W. Findlay  
University of Glasgow

## APPLICATIONS

(\* Reports of applications come to the Newsletter from PUG members, primarily. If you know of applications which use PASCAL, please send us the details. \*)

### Progress Report on PLT - March, 1977

PLT (Programming Language for Teaching) is a machine independent CAI/CMI (Computer Aided Instruction/Computer Managed Instruction) system implemented entirely in PASCAL-6000. PLT features a concise structured lesson creation language implemented with a fast single pass compiler, an efficient interactive interpreter, and full lesson and student monitoring facilities.

The PLT system will automatically step individual students through a series of lessons and tests. Reports by student and/or lesson-tests can be generated using the system's reporting facilities.

PLT is in full production use at Lehigh University and is being used to implement a series of lessons on PASCAL programming.

PLT will be released as an unsupported product after completion of its system internals manual (about April 30, 1977). For further information please write to

Richard J. Cichelli      Christmas-Seacon Hall 14  
Computer Science Group    Lehigh University  
Department of Mathematics    Bethlehem, PA 18015

### RUNOFF text formatter

A version of RUNOFF (the well-known text formatter available on the DEC-10 and other machines) is available in Pascal on the CDC Cyber 175. Educational institutions may get it on a free exchange basis provided you send a tape, expect no immediate response on bug-fixes, and do not distribute it to others. Documentation is also available. Write to Bob Foster, Computing Services Office, University of Illinois, Urbana, IL 61801.

### SLIP--A System for cross compilation.

Programs are written in Pascal. The target code is macro-generated with Stage 2 (using an intermediate code). Complete. Available for distribution. Michel Galinier (\* PUG member \*), Université P. Subatier-Informatique, 118 Route de Narbonne, 31077 Toulouse Cedex, France. (\* 1/5/77 \*)

(\* from a news brief in Electronics, March 17, 1977, p. 140 \*): "Electro-Scientific Industries, Inc., Portland, Ore., is beginning to offer its own compiler for use by others on the [DEC PDP-11]. At the end of last year, Computer Automation Inc., Irvine, Calif., announced a combined compiler-interpreter for its own mini-computers. Both companies point out that Pascal . . . is simpler to use than either Fortran or Basic. ISI is aiming at applications in automated test and data-acquisition systems and in its own laser trimmers. Computer Automation likes it for developing compilers and translators."

(\* T.T. Bell, D.C. Bixler, and M.E. Dyer, "An Extendable Approach to Computer-Aided Software Requirements Engineering," in Structured Design, Infotech State of the Arts Conference, 10/18-20/76, pp. 3-27, also in IEEE Transactions on Software Engineering, 3:1 (January, 1977), 49-60) report that TRW used Pascal as the implementation language in its computer-aided system for maintaining and analyzing system software requirements. (\*): "The simulator generator transforms the [Abstract System Semantic Model (\* a database \*)] representation of the requirements into simulator code in the programming language PASCAL. The flow structure of each [Requirement Network (R.NET) (\* the class of processing flow specifications \*)] is used to develop a PASCAL procedure whose control flow implements that of the R.NET structure. Each processing step (ALPHA) of the R.NET becomes a call to a procedure consisting of the model or algorithm for the ALPHA. The models or algorithms are written in PASCAL." (\* from p. 18 \*)

### PASCAL PRINTER PLOTTER

A listing (6 pages) of the Pascal code for the printer platter described in PUGN No. 7 is available free. Write to Herb Rubenstein, University Computer Center, 227 Experimental Engineering, University of Minnesota, Minneapolis, MN 55455.

# ARTICLES

## DEVELOPMENT OF A PASCAL COMPILER FOR THE C.I.I. IRIS 50. A PARTIAL HISTORY.

Olivier LECARME  
Université de Nice

### 1. ENVIRONMENT

The history which is the subject of the present paper takes place in the University of Nice, a medium-scale University with about fifteen thousand students. The department of computer science is very small, but delivers two different B.Sc. degrees in computer science (informatics), and has full graduate programs. About two hundred students attend these undergraduate and graduate courses.

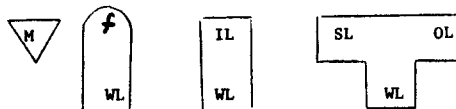
The University computing center serves the whole academic community with a C.I.I. Iris 50 computer, a medium-scale machine comparable in power and capabilities with an I.B.M. 360 model 40 or 44. The multi-programming system Siris 3 allows the execution of several jobs in fixed size partitions, the biggest possible size being 220 K bytes (stand-alone mode), and the most current sizes being 64K or 96K bytes. The department of computer science accesses this computer by remote batch processing, with a mean return time of about one hour.

Programming languages available are an assembly language (without macro definition capabilities), Fortran and Cobol. None of these languages may be considered an adequate support for teaching computer science, and especially for teaching programming to future specialists. Successive attempts to implement Pascal have consequently been done, with variable success. The main difficulties encountered are the lack of dedicated manpower, the weakness of software tools available, and the small amount of storage normally available on the computer.

### 2. FORMALISM FOR DESCRIBING TRANSLATORS

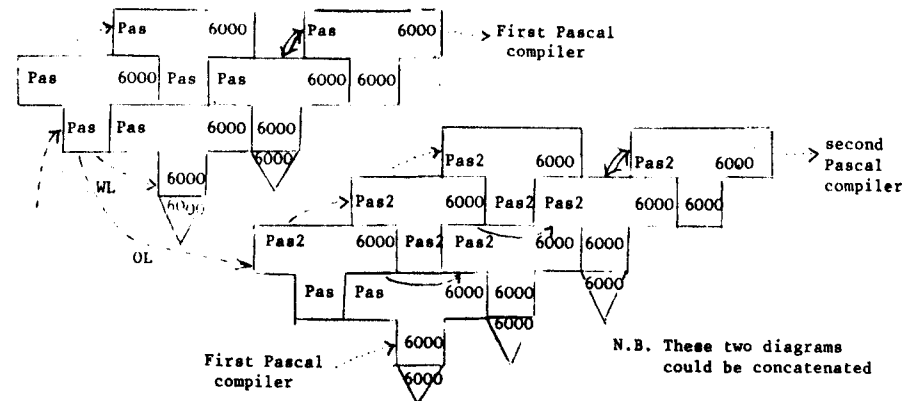
The so-called T-diagrams of Earley and Sturgis are very useful to describe the generation of translators by complicated bootstraps.

The four following different symbols describe the different programs :



The first one is a hardware processor, i.e. a computer or machine M. The second one is a software processor, i.e. a program realizing some unspecified function *f*, and written in the programming language WL. The last two are software processors realizing a specified function : the first one is an interpreter for language IL, and the second one is a translator from source language SL to object language OL.

By concatenation of different symbols, and provided that the same language always appears on both sides of any concerned frontier, we can describe the generation of a compiler by another one. We use four different arrows describing the translation processes : the solid arrow indicates that the target program is the translation of a source program by a programmed translator ; the dashed arrow indicates that the target program is produced by hand, either from scratch (no source program), or by modifying one of the languages concerned (this language is specified along the arrow) ; the dotted arrow indicates that the target program is a copy of another one, without modification ; the double-ended arrow indicates that two programs must be identical for validating the bootstrap. Using this formalism, we can describe the history of the two principal Pascal compilers for the CDC 6600 (the following diagrams abusively simplify the history) :



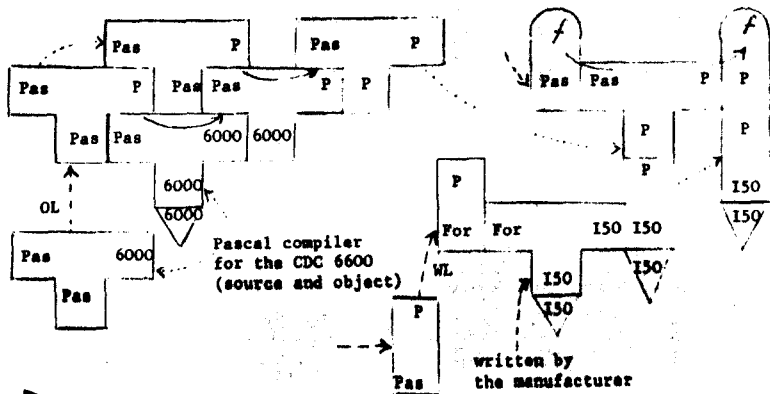
### 3. FIRST ATTEMPT WITH PASCAL-P

A translation into Fortran of the interpreter for P-code (first version of Pascal-P) had been made in Paris for a CDC 3600, and carefully written for being really portable. In fact, it was quickly implemented on the Iris 50, but it gave so disastrous performances in time and space that the compilation of a tiny program (by interpreting the compiler) would have necessitated exclusive use of the computer for about one hour. Explanations of this phenomenon are simple and interesting.

The packing and unpacking of P-code instructions fields were done with multiplications and divisions by powers of 2. The Fortran compiler did not recognize the special form of these operations, and generated ordinary code, which was especially complicated for integer arithmetic. This partly explains the slowness of the interpretation, other factors being the use of Fortran input-output routines, and the heavy overlay loading necessary because of memory problems.

These memory problems are partly due to the weaknesses of the first P-code, corrected in the last version : either the length of each data object must be explicitly indicated in its representation, or each object must be allowed the same storage size. This second solution had been chosen in Paris, because of better time performances, but it necessitated two memory words (32 bits) for each type Boolean, character, integer and real, because of the 58 bits necessary for sets when interpreting the compiler. The interpreter needs a "memory" of 24 K "words", i.e. in our case 192K bytes, plus the interpreter itself, the interpreted program and the Fortran execution support (principally input-output).

The following diagrams describe the two phases of the generation and use of a Pascal implementation with Pascal-P.



4. SECOND ATTEMPT WITH PASCAL-P

A completely new translation of the interpreter into Fortran was done, using the second version of Pascal-P. Some assembly language routines were used for packing and unpacking fields, manipulating length indicators for data, etc. All possible gains in time and size were carefully searched and programmed. The final version, used during the last academic year, permitted the compilation and execution of half-page programs at a tolerable cost, but no more. A student trying to have a four page program compiled would have exhausted all his available funds for one month ! This taught to students an extraordinary carefulness when

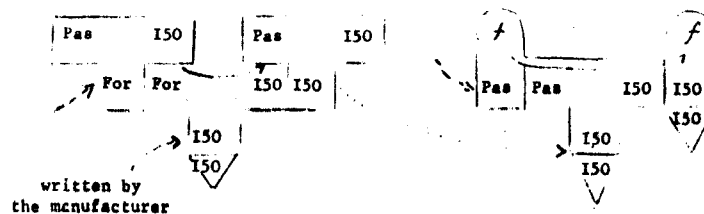
trying their programs, and in many cases they managed to get a complete working program in only one run. This result is not so bad, but other frustrations were not tolerable, and this tool could not have been used for more than one year.

Moreover, the poor performances of this compiler made completely impossible to use it as a tool for developing an actual compiler, generating machine code : the compilation of the Pascal-P compiler would have necessitated more than 8 continuous hours of exclusive usage of the computer. This was, practically, absolutely impossible, especially within our local context.

The same diagrams in the preceding section describe this compiler.

5. A STUDENT JOB

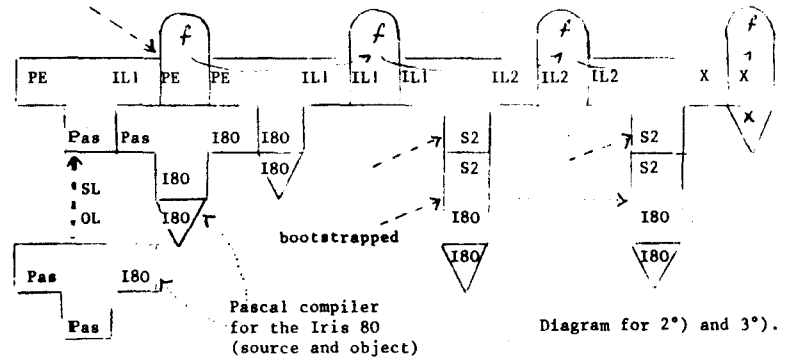
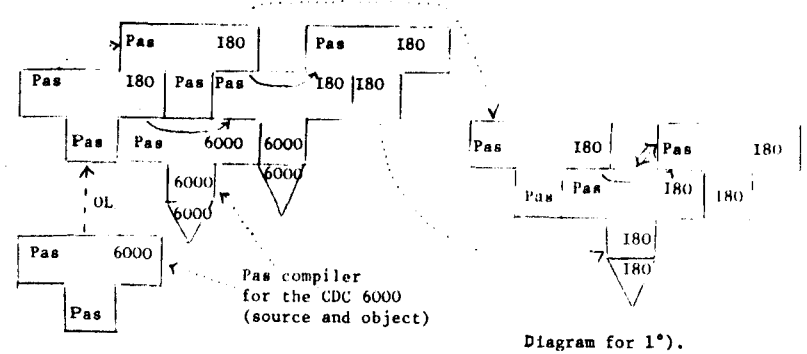
During the same academic year, and using only his spare time, an undergraduate student wrote, partly in Fortran and partly in assembly language, an in-core load-and-go Pascal compiler. It happened, by incredibly good fortune, that this compiler was sufficiently well written to be usable by students. It is used during the present academic year, and gives a compilation cost of about 1.4% of that of the Pascal-P second compiler. It implements standard Pascal with only a few very minor restrictions, but prescribes very small limits on the size of programs, the number of identifiers, the complexity of all declarations, and so on. All in all, it is only a teaching tool for small programs, and it is completely impossible to make it usable for implementing a full compiler. However, it was a mean to wait for something better and more general, without too much frustration.



6. DEVELOPMENT OF THE TOOLS FOR THE FUTURE BOOTSTRAP

The compiler of section 3, 4 and 5 are only toy compilers ; they cannot compile themselves, and consequently they cannot be used for developing themselves. Since we have no other computer available in the vicinity, and no adequate software tool on our computer, many possible solutions had to be eliminated, especially those which use a powerful macro-generator. No funds were available for repetitive travels between Nice and other computing center, and still less for computer time in other centers.

The chosen solution is consequently very original, somewhat complicated to describe, but needing only a minimal amount of programming. Moreover, this programming is not done by us, but in another (wealthier) University, Université Paul Sabatier in Toulouse, France. Several processors are available in that later place : 1°) A Pascal compiler running on a CII Iris 80 computer, bootstrapped from a CDC 6000 computer by Didier THIBAUT, and described in Pascal Newsletter #7 ; it is a compiler from Pascal to Iris 80 machine language, written in Pascal and bootstrapped into Iris 80 machine language, according to the customary diagram :

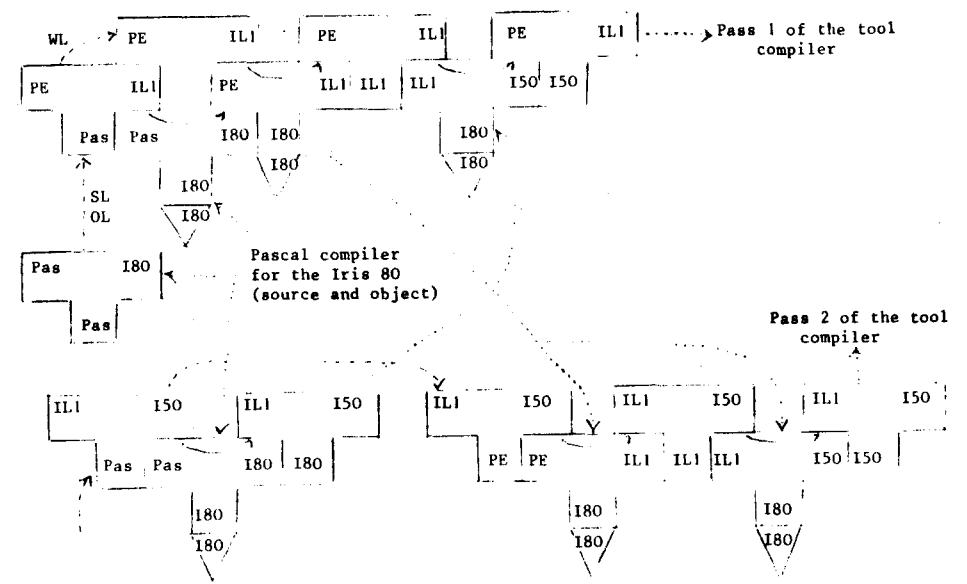


2°) A compiler from a subset of Pascal (called Pascal-E) to an intermediate language IL1, written in Pascal. 3°) A translator from IL1 to a second intermediate language IL2, written in Stage 2; Stage 2 is itself an interpreter, implemented via a full bootstrap. IL1, IL2 and the corresponding compiler and translator were developed in Toulouse as a tool for cross-compiling, on the Iris 80, Pascal programs for several mini-computers.

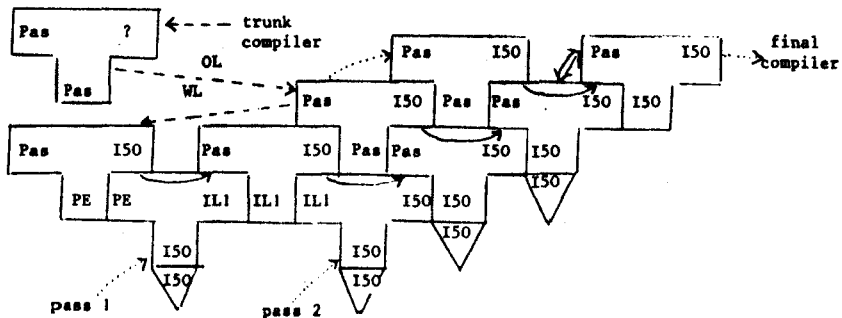
IL1 is a quadruplet language, as suggested by Gries and used between two passes in many compilers ; IL2 is a machine oriented language, especially designed for being easily translated into machine code for mini-computers, by a second translator written in Stage 2. The diagram above is for 2°) and 3°).

7. DEVELOPMENT OF THE TOOL AND FINAL COMPILERS

We wanted to avoid the bootstrap of a complete compiler, made by changing the object language on an existing one, because of the difficulties of debugging and tuning such a large program when using two computers distant by 600 kilometers. By writing by hand, in Pascal-E, a translator from IL1 to Iris 50 machine code, we can obtain the tool compiler we need, but in a two-pass form. This translator is very easy to write since IL1 is a quadruplet language. Some modifications are also necessary for the compiler from Pascal-E to IL1, to write it in the subset of Pascal it accepts, but these modifications are in fact trivial, since the only things to do are to replace, for example, writeln (code) by writelnc , or write (output,c) by write (c). By doing six translations on the Iris 80, according to the following diagram, we obtain a two-pass Pascal-E compiler usable on the Iris 50 :



The future and final compiler for the Iris 50 will be obtained by filling holes in the "trunk compiler" of H.H. Nägeli, as it was done with success in Tokyo for the Hitac 8000 ; it will be implemented with the tool compiler :



The two-pass tool compiler is intended to be usable in next March. It implements the minimal subset of Pascal needed for writing the compiler. Omitted features are reals, file declarations, most standard procedures or functions. Standard files necessary for implementing the compiler are built-in, an accessed via several pre-defined procedures. No other part of the language is omitted and packed structures are implemented. The translator from IL1 to Iris 50 produces code for the linkage editor, one module per procedure. This will permit segmentation and overlaying of the compiler, but without any means for checking the validity of access to global variables.

The final compiler will implement exactly standard Pascal, and produce modules for the linkage editor, as the preceding one. The whole process should be terminated at the end of 1977, and several computing centers are already interested in the final product. (\*Received 77/03/09.\*)

#### A FURTHER DEFENCE OF FORMATTED INPUT

B. A. E. MEEKINGS

Computer Studies Department  
University of Lancaster

In PUGN # 7, Barron and Mullins attempt to demolish the case for formatted input. Without wishing to blow up the controversy beyond reasonable proportion. I would like to add a voice in favour of formatting.

I feel that Barron and Mullins have rather missed the point, inasmuch as input data is unfortunately not always under our control; in addition, of course, it is unreasonable to exclude a feature from the language simply because it can be done in another (more long-winded) way. I have

implemented formatted input in my Pascal P4 compiler for the following, I think very good, reasons:

- i. it is a simple modification to make, with relatively little overhead.
- ii. it is entirely within the "spirit" of Pascal, making input consistent with output, which it currently is not
- iii. it allows for reading in character strings to complement the similar output feature
- iv. I am working in general area of simulation languages, and hence attempting to woo users away from the traditionally Fortran based languages - I see formatted input as one less obstacle for them to overcome in the transition.

I am not in any sense an advocate of Fortran, but I do feel that the association between formatted input and Fortran is no valid reason for its exclusion from other languages.

In short, the addition of formatted input, to supplement the existing unformatted input facilities, can only enhance an already versatile language.

(\*Received 77/03/21.\*)

(\* Meekings is not a PUG member yet.\*)

#### THE REPRESENTATION OF PASCAL FOR COMPUTER INPUT

The original lexical definition of Pascal was closely tied to the CDC character set. The current implementation allows for complete representation of all Pascal elements in the ASCII character set except the up arrow which is used for pointer and file references. In this case, circumflex is the ASCII character that is used.

This lexical representation should have a 48 character alternative for computer systems with restricted character sets. Some obvious equivalences are period-period (..) for colon (:), (this is almost always true for the CDC implementation), period-comma (.,) for semicolon (;), and period-equal (.=) for replacement (:=). Additionally, two letter alternatives for relational operators should be allowed. Brackets for subscripts could be (. and .) or (/ and /).

Whether or not to always accept the 48 character representations is an open question.

#### COMPILE OPTIONS

The Report [1] does not mention compiler options enclosed in comment symbols, but perhaps this means of defining compiler options should be formalized. Several compile time options like listing control, code generation, and source line width should be universally defined.

#### INTERNAL CHARACTER SET

Pascal could be made the first language to standardize the ordering of characters for the basic data type CHAR. This standard could be ASCII. Thus the basic data type CHAR will have 128 elements. At the moment, the CDC implementation is stuck with the anachronism of a character set based on a 6-bit element. It would be reasonable for text files to be mapped between the internal ASCII set and the external operating system character set. The normal character set for a particular machine could be accessed without translation by using a packed file of the appropriate integer subrange type.

An alternative solution to the problem of antiquated character sets would be to provide several CHAR types. ASCII could be a keyword which defines the 7-bit ASCII character set. EBCDIC would define that 8-bit set. The local machine implementation of characters would be CHAR. There should be character set conversions across assignment statements.

#### REMOVAL OF CURRENT RESTRICTIONS AND ASYMMETRIES

The restrictions and asymmetries outlined below are made with reference to the CDC implementation of Pascal.

First, and foremost, the designation PACKED should tell the compiler to optimize storage usage instead of speed of access in data structures. It should not have any other effect upon constructs in the language. Unfortunately for the CDC implementation, this is not true. One cannot compare or output unpacked arrays of characters, but this can be done for packed arrays of characters. This particular asymmetry is reminiscent of Fortran in its arbitrariness.

There is an implementation problem in passing elements of packed structures as VAR parameters which will probably have to remain.

A bothersome restriction is set size. Sets should be allowed to have any size, not just some convenient but fixed machine size. It is difficult to justify the exclusion of the last 4 elements of the CDC character set just because there were only 60 bits in the CDC word.

If a subrange declaration of INTEGER exceeds the normal precision of the usual representation, an automatic extension to multiple precision arithmetic should occur. There should be some way to declare the minimal precision required for the REAL type so that multiple precision arithmetic could be used if necessary. In this manner, a precision sensitive algorithm could be run on different precision machines with good results.

FUNCTIONS should be able to return any type. Identifiers should be unique to their entire length.

#### THE PROGRAM DECLARATION

Aside from specifying the name of the main program, the program declaration contains a list of file names. The current usage of the declaration in the CDC implementation is to allow immediate opening of all files upon entry into the main program and to establish the ordering of files for the positional substitution of system file names that is possible in CDC operating systems. The first action is unnecessary. The second action should be clarified in the Report [1] or regarded as a CDC implementation feature. Neither INPUT or OUTPUT should be mandatory on the program declaration. All files must be opened explicitly before test or data transfer. Otherwise, an error is diagnosed. Close operations should also be available.

#### VARIANT RECORDS

The current definition of variant records is quite useful and has been cleverly utilized to subvert type checking within the CDC implementation of the language. This is unaesthetic even though it is necessary. There ought to be a better way.

Unfortunately, the current definition of the language does not allow the tag field of variant records to be automatically set when a variant record is allocated or a variant field is stored and to be tested for correct type when a field is fetched. This checking should be done to protect the run-time system from the lazy or careless user. Perhaps another formal compiler option should be defined to disable this type of protective code.

#### THE CASE STATEMENT

A decision in a case statement may be implemented by a jump table or series of tests. The compiler should choose which technique to use based on the type of expression involved. Perhaps a type identifier in addition to the normal expression would help narrow the range of values and allow the faster jump table to be selected. In any case, an ELSE exit is highly desirable. It is a waste of time to force the programmer to protect each case statement with an if statement. Also, it would then be possible to make case statement tests on strings, large integers, or real numbers. Another extension would be to allow the subrange notation for case labels so that ranges of values could be directed to a statement.

#### BOOLEAN EXPRESSIONS

Boolean expressions should be computed only as far as necessary to establish the value of each subexpression. AND is FALSE when the left operand is FALSE. Then the right operand could be ignored. Similarly, OR is TRUE when the left operand is TRUE. The ultimate value of the entire expression would still be correct by doing this partial evaluation, and the expression of loop termination conditions when indices go out of bounds will be much simpler.

#### CONSTANTS, DECLARATIONS, AND CONSTRUCTORS

The Pascal language needs a means of constructing structured constants. In fact, Wirth [2] has defined constructors for this

purpose. It should be implemented.

In constant declarations, it should be possible to perform compile time computations using constants and previously defined constant identifiers.

#### VALUE INITIALIZATION AND OWN VARIABLES

The current Pascal has a large core requirement because it does not have value initialization and it is not overlaid. Value initialization of structured variables can be done using the constructors mentioned above.

Value initialization should be possible in procedures other than the main program. These variables would be initialized on each procedure entry. On most machines, this will require run time code for initialization instead of loader initialization.

Own variables (in the sense of Algol 60) should be allowed, and would be initialized just once at load time.

#### PROCEDURE AND FUNCTION TYPES FOR COMPILE TIME CHECKING

One omission in the definition of Pascal in the usual strict compile time type checking is the unchecked correspondence between declaration and usage of procedures and functions passed as parameters to other procedures and functions. This omission opens the run time system to mysterious collapse when procedures are incorrectly called. This compile time check can be done in one pass compilation if procedure and function type identifiers can be defined. The type would have the attributes of denoting a procedure or function, the number of parameters, and the type and VAR property of each parameter or result. This would actually simplify the syntax of a parameter list by eliminating the need for the keywords FUNCTION and PROCEDURE. If the parameter position is typed by a procedure type identifier, then the actual name of a procedure must be passed at call.

#### DYNAMIC ARRAY PARAMETERS

Although some limited means of passing variable sized arrays is desperately needed in Pascal, Jacobi's proposal [3] is too limited in scope. A dynamic array parameter should be indicated in a parameter list by the inclusion of the keyword DYNAMIC before the type identifier. Any actual parameter which conforms to the type, except for array bounds, would be accepted. This allows for arbitrary packed structures. The prohibition against other than element wise access should only apply down to the last array substructure.

#### NEW BASIC TYPES AND OPERATORS

A major advantage of Pascal over other programming languages is its expressive power in data structures. Because more information about the data being operated on is available to the compiler, better code can be generated to handle the manipulations. For this reason, the basic types of Pascal should be expanded. The COMPLEX data type is one that should be added.

For similar reasons, the exponentiation operator should be added to the language.

Another extension I propose requires more justification because of its impact on the implementation. STRING should be added as a basic type along with operators, standard procedures, and functions for concatenation, extraction, pattern matching, and type conversion. The closest approach available now is a record composed of an integer character count and an array of characters. This is an inadequate alternative as the compiler cannot easily recognize this as a string and the programmer is burdened with providing a plethora of auxiliary routines. The resulting code is less efficient than what is possible if the type STRING was defined.

Of course, well defined automatic coercions (in the sense of Algol 68) must be available between strings and arrays of characters. Additional standard procedures and functions equivalent to the CDC Fortran ENCODE and DECODE routines should be available. When possible, the compiler should revert to the older pattern of fixed sized array of characters instead of treating all character string constants as STRINGS.

#### TRANSFER FUNCTIONS

Transfer functions between scalar types and their character string names should be available. There should be a type check defeating function which regards its source and destination as bit fields of some appropriate width. This function would eliminate the need for the variant record subversion. Inverse functions for ORD across all basic types might be considered to be the type check defeating mechanism.

#### EXTENSION OF RELATIONAL OPERATORS TO STRUCTURED TYPES

Relational operators already extend to structured types in the one case of packed array of character. They should extend in this manner to all structured types. To do so there must be an ordering of elements within a structured type from first to last and the comparison must take place in this order. This straightening should apply to several other areas of the language as in input/output and constant formation.

#### FILES AND TEXT FILES

The Report [1] allows attaching the keyword PACKED to file types but the CDC implementation does nothing with it. Actually, there is a confusion in this area of file types. There are really three types of files. There are unpacked files, packed files, and text files. The type FILE OF CHAR, PACKED FILE OF CHAR, and TEXT are not equivalent.

In particular, an unpacked file of some type aligns items of the type on any particular machine word (or byte) boundary that is convenient and provides quick access. A packed file of type is implemented with every reasonable effort to not waste one bit of disk or memory space. Specifically, on the CDC machine, FILE OF BOOLEAN would be stored with one boolean value per 60 bit word and PACKED FILE OF BOOLEAN would be stored with 60 boolean values in one word. Also, with packed file of subrange of integer type, it should be possible to access any packing of data on disk independent of word boundaries. The only operations available on packed or unpacked file types are GET and PUT (or the shorthand READ and WRITE with no type conversion) along with assorted status testing and positioning operations.

The files of type TEXT are fundamentally different from the other two file types. First, the procedures READ and WRITE are available with their full formatting and type conversion possibilities. But a text file is not a FILE OF CHAR. It is a specially handled character file with lines of text. It has line boundaries which a FILE OF CHAR does not have. In fact, each line of text should be treated like a value of type STRING.

Also, text files come in two varieties, paged and unpagged. This attribute is established by declaration at compile or open time. An unpagged TEXT file would be associated with devices such as card reader, card punch, magnetic tape, or teletype input. A paged TEXT file would be associated with a line printer or teletype output. TEXT files must operate with the correct order of input and output on interactive devices. It may be necessary to declare files as being interactive in order to keep the run time system straightforward.

The user should not be responsible for placing carriage control in column 1 of every line of paged output. The paged output routine should normally provide a blank for the line printer but omit it for teletypes. A call to the PAGE procedure should set up carriage control (like page eject or form feed) as needed.

Text files are subject to translation between the operating system character set and the internal character representation. The rules for skipping from one line to the next have not yet been well formed and will have to account for the straightening process of structured types. The problem of reading blanks before end of file should be resolved once and for all. It should be possible to read one line of text into one STRING type variable and perform type conversion later.

For paged text files, it should be possible to automatically invoke user supplied procedures at top and bottom of forms. Other user supplied procedures could be invoked on various fault conditions for all file types.

#### FORMATTED INPUT AND OUTPUT

It is not necessary to resurrect the Fortran format to handle text file formatting in Pascal. The WRITE procedure field width specifications are fine. They should be extended to the READ procedure. It should be possible to read and write delimited strings of characters. There should be an option for separator characters other than blank between input or output items.

#### FILE HANDLING

Text files should be processed strictly sequentially. Random positioning should be allowed on non-text files. Since most operating systems provide for file structures that are more complex than currently defined in Pascal, there should be some generally agreed upon extensions to file operations that are not mandatory. The CDC implementation does have the extensions of SEGMENTED files. The CDC version needs additional extension for multiple file files. For example, add GETEOF, PUTEOF, and WHILE NOT EOI (for End-Of-Information).

The current CDC implementation does a rather poor job of file positioning at open and close time. Explicit file open and close operations are necessary. A rewind or no rewind option is vital for both. Other file attributes like system file name, buffer size, procedures for handling data exceptions should have reasonable defaults but be open to user specification.

#### OVERLAYS

The Pascal language needs overlays. The first use would be to reduce the size of the compiler by doing value initialization functions in one overlay and the main compilation process in another. A halfway overlay attempt already exists in the CDC implementation to issue compiler error messages.

Designation of overlays can be achieved by compile time options in comments or by adding the keyword OVERLAY to the syntax. The choice of which to use is open and should be decided. Overlays are organized by procedure or groups of procedures. Explicit overlay calls should not be necessary as in CDC Fortran. The compiler can recognize a call to a different overlay and generate the appropriate code.

A good proposal for an overlay mechanism has already been made [3]. However, it already exceeds the capabilities of the CDC operating system. To accommodate that system, no more than two levels of overlays could be allowed and the implementation would be even easier if overlaid procedures could be called only from the outer block.

As stated in the overlay proposal, overlays can be viewed as is the designation PACKED. A particular Pascal implementation will try to follow the overlay directives and the program will always run correctly. However, the object code may not be as deeply overlaid as specified.

#### PREAMBLES AND POSTAMBLE

A compiler does not stand by itself within a computer system. A well developed language system must have a wide range of subprograms available for use. One reason that Fortran will be hard to displace is the large number of subprograms already developed for it.

The implementation of separately compiled procedures in CDC Pascal was a gigantic step forward in increasing the usability of the language. But now the user is burdened with declaring all external procedures he intends to use. The declaration is necessary but it should come from the language system rather than the user.

The compiler cannot be reassembled every time a new subprogram is added to the library, and it should not carry declarations for every possible external subprogram when only a small number of them for a specific application will be accessed.

The solution is to allow the selection of several preambles which initialize the compiler to a particular application environment. The compiler would look to the preamble for each declaration section (PROGRAM, LABEL, CONST, TYPE, VAR, and subprograms) first and then compile the corresponding user declaration section. Preambles should be input as ordinary text or specially processed system text records.

A provision for a postamble would be useful to allow driver main programs in a student environment or for a non-code producing dummy main program when compiling library subprograms.

The preambles and postamble allow a user job to be compiled in any desired environment. By allowing full procedure parameter description in the preambles, including procedures passed as parameters, complete compile time checking of all external subprogram linkages can be obtained.

Also, some mechanism of protecting access to the elements of a structured type introduced in the preamble is desirable. This would be useful in making certain data structures appear as basic types to the user.

- [1] Jensen, Wirth, "Pascal User Manual And Report", 2nd Edition, Springer-Verlag, 1975.
- [2] Wirth, "Algorithms + Data Structures = Programs", Printice-Hall, 1975.
- [3] Pascal User's Group, "Pascal Newsletter", No. 5, September 1976.

(\*Received 77/03/24.\*)

## A PROPOSAL FOR INCREASED SECURITY IN THE USE OF VARIANT RECORDS

WILLIAM BARABASH  
CHARLES R. HILL  
RICHARD B. KIEBURTZ  
SUNY AT STONY BROOK  
STONY BROOK, NEW YORK 11794

The use of variant records in most Pascal implementations is dangerous because most compilers do not emit a check for conformity with the value of the tagfield when a variant field is referenced. Indeed, the latest version of the Revised Pascal Report defines a language in which the tagfield may even be absent, making conformity checks impossible! Even so, when the tagfield is present and the compiler does emit conformity checks automatically, the programmer still has the ability to dynamically assign values to the tagfield.

We propose that the variant field of a record be protected from such abuse, either accidentally or intentionally. This means that the compiler should be required to emit conformity checks when a variant field is accessed; that the tagfield must always be present in every variant record; and that the programmer not be allowed to alter the tagfield in a variant record by means of a simple assignment statement.

Currently, a variant record can be created dynamically when the standard procedure `New` is applied to a pointer variable that is bound to a variant record type. This standard procedure has the ability to initialize tag fields to constants specified in the call. We propose that thereafter the type of the variant record is frozen by the values of the tagfields. The fields within the record can all be referred to; however, if a field in the variant part of the record is referred to, the tagfield will automatically be tested for conformity.

This is not sufficient, because variant records in a Pascal program can reside in the stack, being created on block entry. Such records can only be initialized to "undefined". Also, during the lifetime of a dynamically created variant record, it may be created and used, then put on a free list, then used subsequently. The subsequent user might want a dif-

ferent set of values assigned to the tagfields of the record. To get around these difficulties, we propose a new standard procedure which will

- 1) set all of the fields in the record to "undefined", then
- 2) initialize the tagfields in the record to the constant values specified in the call.

A call to this procedure would be exactly like a call to standard procedure `New`, except that the first parameter would designate an already-existing record variable instead of a pointer variable. Such a procedure might be called "Renew". Note that the use of `Renew` has one chief drawback, namely that when a variant record is created, space must be allocated for the largest possible variant field. On the other hand, if a variant record is created by means other than the standard procedure `New`, the maximum space must be allocated anyway. Furthermore, garbage collection would be simplified: there would be no need to provide more than one parameter to standard procedure `Dispose`.

Lastly, it might be argued that enforced run-time conformity checks when a variant field is frequently referred to can severely degrade the performance of a Pascal program. We propose a slightly modified with statement which can open the scope of a variant record with a tagfield value assertion. The assertion is checked at run-time once every time the with statement is entered. Within the body of the with statement, any reference to a variant field of the record can be checked for conformity with the asserted values of the tagfields at compile time. Such a statement would have the syntax

```
with recordvariable (const 1,..., const N) do S
```

meaning that we assert that the variable "recordvariable" has tagfields whose values are "const 1", ..., and "const N", as if the call

```
Renew (recordvariable, const 1,..., const N)
```

was made to initialize the record.

(\*Received 77/03/27.\*)

## Update on UCSD PASCAL Activities

Kenneth L. Bowles  
Institute for Information Systems  
University of California San Diego  
La Jolla, California 92093  
(714)452-4526

17 April, 1977

### LSI-11 Software

UCSD has recently started using a single user software system for microcomputers, with all major programs written in PASCAL. The compiler is based on the P-2 portable compiler distributed by the ETH group at Zurich, but it generates compressed pseudo-code for a much revised P-machine interpreter. As currently implemented on the LSI-11, compile speed is about 700 lines per minute (1000 on the PDP11/10). The system includes an interactive monitor, editor, utility file handler, and debugging package in addition to the compiler and interpreter. With 56K bytes of main memory, and dual floppy disk drives, it has proven more convenient and faster to do all software development on the microcomputer than to cross compile from a big machine. Whereas we have been using versions of this system that depend on I/O support from the RT11 operating system distributed by Digital Equipment Corp., our new system is independent of any external software support. The resident monitor, interpreter, and run-time support package occupy an aggregate of about 10K bytes of memory.

Operation of large programs is facilitated through the concept of "Segment Procedures", which are rolled into memory only while actually invoked. The compiler (20K bytes), editor, and file handler are all separate segment procedures. One segment procedure can call others, and segment procedures may be declared nested within other segment procedures, to allow flexibility in memory management. The user's data space expands (or contracts if necessary) to take advantage of as much memory as possible after the appropriate code segments have been loaded.

Our plan is to have the new system completed to the point where it may be released to others by mid summer, 1977, with documentation package included. During the summer, we also plan to complete a graphics support package (including an editor for graphics oriented CAI materials), an assembler for PDP11 native code, and a compiler option allowing selected PASCAL procedures to generate native code rather than P-machine pseudo code. The system is designed to make relatively painless the problem of adding native code routines programmed in assembly language, allowing a user to augment the set of built-in functions and procedures where efficiency is important. This note has been composed and printed using a proprietary extended version of the text editor intended for use with a CRT display, which should be ready for release by late summer. The system should be usable on any PDP11 system capable of bootstrap loading from RX11-compatible floppy disk drives, or from the drives supplied with the Terak Corporation LSI-11 based machines (see next section). Further details may be obtained, on request to the address given in the heading, in separate notes titled "Status of UCSD PASCAL Project", and "Preliminary Description of UCSD PASCAL Software System".

### LSI-11 Hardware

In addition to the well advertised PDP11/03 systems available from Digital Equipment, several smaller companies are offering stand-alone computers based on the LSI-11 that would be directly suitable for our

software. We have been particularly interested in using a stand-alone machine with low cost graphic display for interactive educational applications. In connection with the EDUCOM Discount Program (see EDUCOM Bulletin, Spring, 1977), it now appears virtually certain that the Terak Corporation 8510A will be available to member institutions for about \$5300 per machine (LSI-11, 56K bytes RAM, single floppy disk, CRT for superimposed but independent text and graphics, keyboard, RS232 asynchronous interface for network or printer connection). An example of the graphic display of this machine is attached to this note.

### Other Microcomputers

Anyone who attended the West Coast Computer Faire in San Francisco should have come away impressed that small stand-alone microcomputers are big business and here to stay. It is possible to re-implement our PASCAL based software system on systems based on any of the most popular microprocessors within about 3 months of work by one programmer. At UCSD we have started to re-implement for the Zilog Z80 OEM series of modules, which could serve as the basis for PASCAL interpretive operation roughly as fast as the LSI-11. At the Faire, we talked with principal officers of most of the well known microcomputer manufacturers who sell to the hobbyist market, and encountered almost uniform enthusiasm for the idea of making PASCAL available on an industry-wide basis. On the basis of those conversations, there is a reasonable chance that our PASCAL system will be available later this year for use with the 8080A, 6502, and M6800 microprocessors in addition to the LSI-11 and Z80.

### Proposal for Manufacturer Independent PASCAL System

There is widespread frustration, among those who make and sell microcomputer systems, that only BASIC is generally available, and that no two BASIC implementations are alike. Many of those we talked with at the Faire asked whether PASCAL would be standardized, to avoid the problems they encounter with non-standard BASIC (in addition to providing a more powerful programming vehicle). Even a casual reading of the PASCAL User Group newsletter is enough to convince one that: a) people are finding it necessary to enhance PASCAL for their own particular applications; b) the heterogeneity of the enhancements already reported is so great that no committee exercise is likely to produce a standard.

As an alternative, we believe that a chance exists to establish a defacto standard for PASCAL, at least for small systems, by starting a bandwagon effect in the microcomputer industry. A good definition of the underlying language for such a standard is contained in the Jensen and Wirth "PASCAL User Manual and Report". To implement a complete interactive software system, with adequate efficiency to run on a microcomputer, we have found it necessary to add built-in functions and procedures for handling text and graphics, and an EXIT(<procedurename>) built-in for clean termination of highly recursive programs. We have implemented SETs of up to 255 members in a way that uses memory efficiently, as well as Packed Arrays of BOOLEAN. For READ from a keyboard, the implied GET has to happen before the implied transfer from the window variable associated with the file. For handling floppy disks and other small storage media, we use the DEC standard of 512 byte blocks, and allow logical records conforming to any structured type allowed in PASCAL. In most other respects, we have been able to conform closely to the language defined in the Jensen/Wirth book.

If one common PASCAL based software system were to become available almost simultaneously for most of the mass distribution microcomputers, that system would establish the basis of a defacto standard for small stand-alone computers. Changes to such a system would certainly be needed with experience, but those changes might well be made readily available to most users through "down line loading" of object code through the dialed telephone network. Control of the PASCAL language standard might well be vested, at least temporarily, in a committee appointed by the PASCAL User's Group. Fast turnaround communications among the members of such a committee could be supported by "Electronic Mail" techniques over the dialed telephone network. The verbal responses we received from the manufacturers at the Computer Faire suggest that an unusual opportunity, that may not be repeated, exists in mid 1977 to establish a defacto standard in the manner described here. We invite the PASCAL User's Group to join with us at UCSD in bringing this about this summer. In most respects, the language and system definition design questions can be separated from implementation details. We have sought support to allow some of the advanced computer science students at UCSD to perform the implementation work on as many of the microcomputers as possible. Representatives of other institutions would be welcome to work with us in La Jolla, either on system definition or implementation. However we will not be able, ourselves, to devote a major percentage of our working time on definition of a standard.

Interested readers are invited to request copies of the following separate notes pertaining to the points discussed in this section: "An Appeal for Support of Manufacturer Independent Software", "Direct-Dialed Tele-Mail", "Proposal for EDUCOM Software/Courseware Exchange", "Minimum Cost Tele-Mail", "Student Projects for UCSD PASCAL System", "The Quest for a Cheap General Purpose Stand-Alone Computer".

#### Introductory Textbook

For the last two years we have used PASCAL as the basis of the large attendance introductory course in problem solving and programming at UCSD. The course is based on a textbook by this writer, that so far has been printed in the campus print shop. Student responses have been unusually favorable, and the course reaches more than two-thirds of the undergraduate population even though it is treated as elective for most majors. This response results partly from the non-numerical approach of the book, partly from student interest in our interactive system on the PDP11's, and partly from our use of Keller's Personalized System of Instruction (PSI) as a teaching method. Though suitable for PSI, the book can also be used as the basis for a conventional course. At the invitation of Professor David Gries, acting as computer science area editor for Springer Verlag publishers, the book will be published in paperback form this summer. The production schedule will be tight, and we anticipate that the first copies will be available barely in time for the start of fall quarter classes in late September. Springer is interested in knowing who might be interested in using the book and when. Unfortunately, alterations to make the non-numerical approach more readily accessible on many machines will make it difficult to circulate advance copies of the final text until late June at the earliest. We will be happy to forward inquiries to Springer.

Though very popular with the students, the non-numerical approach of the book has been difficult to sell to most of our publishers. The approach used so far has depended upon programming examples using English text, and requires STRING variables and supporting built-in functions that we have added to PASCAL. In spite of this, the students learn the same programming skills that are taught in courses using traditional algebraic problem examples.

Since the inception of our project, we have wanted to orient the course to teaching with graphics oriented problem examples, using an approach motivated by the "Turtle Graphics" used by Seymour Papert of MIT. The microcomputers now becoming available make it possible to teach with a graphics orientation at virtually no higher price than needed for non-graphics materials. Accordingly, the textbook will be revised to augment, and often replace, the text oriented examples with graphics examples. For potential users lacking a microcomputer with graphics display, several alternate possibilities exist. Our built-in functions and procedures for graphics should be relatively easy to add to existing PASCAL compilers for other machines, and we will supply documentation to assist in that process. A description of the built-in's needed is contained in the note "Status of UCSD PASCAL Project" already cited. The implementation will assume a graphic display based on the "bit-map" principle, for which many devices are available in the microcomputer industry. Alternate display drivers will also be provided for the Tektronix 4006, 4010, ... series of direct-view storage tube terminals. Successful, though crude, plotting of the graphic output will also be possible on ordinary line printers. High quality graphic output is possible on matrix printers such as those made by Printronix (the graphic example attached to this note), Gould, Varian, and Versatec.

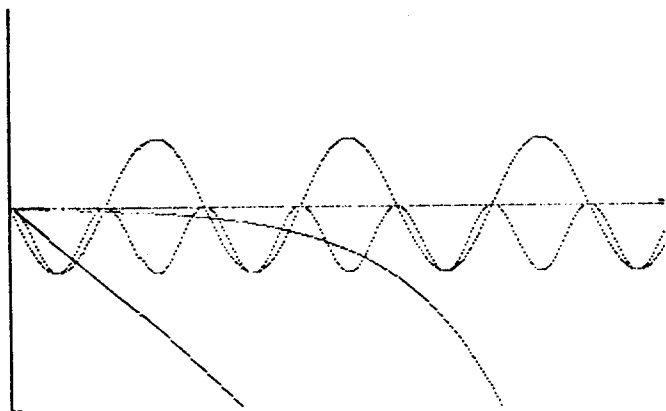
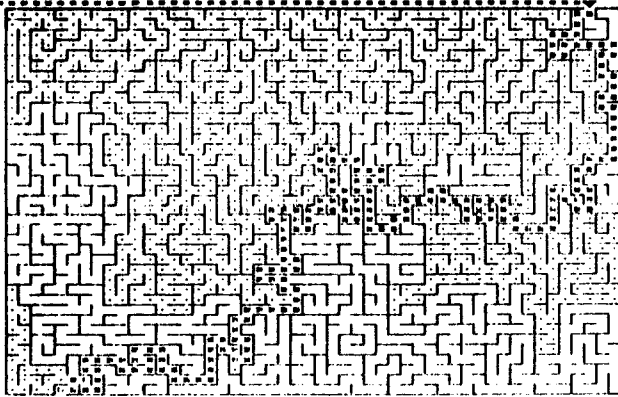
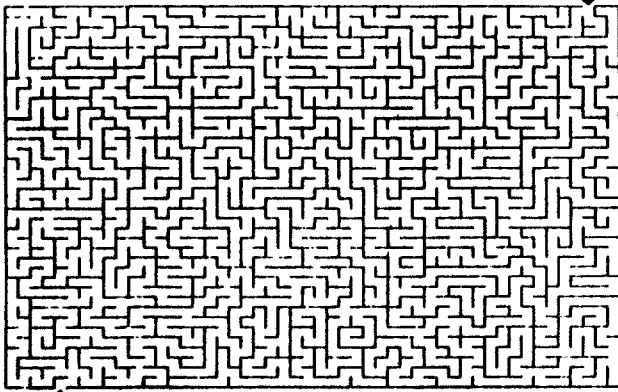
#### B6700 PASCAL Compiler

A PASCAL compiler which generates native code for the B6700 is now in operation at UCSD and available for distribution from the UCSD Computer Center. The compiler is written in PASCAL, and is based on the same variant of the P-2 portable compiler on which we have based the microcomputer implementation. Compile speed is about 5000 lines per minute of logged processor time. This compiler has been used for teaching large classes at UCSD for the last two months. As far as we know, most of the serious bugs in the original P-2 compiler have been corrected in both the B6700 and microcomputer implementations. The B6700 compiler provides access to most of the extensive file handling features of the B6700. At present, no implementation documentation has been completed for the B6700 compiler. The Computer Center will almost certainly generate such a document given an indication of interest in using this compiler by other institutions. Readers interested in obtaining a copy of the B6700 compiler should contact Henry Fischer, UCSD Computer Center, La Jolla, CA 92093 (714)452-4050.

#### Apology to Correspondents

I offer an apology to the many people interested in our PASCAL work who have tried unsuccessfully to reach me by telephone or letter in the last few months. Currently I must depend upon several pooled secretaries who are not easily accessible. Having been occupied with a heavy teaching schedule, and with a committee assignment consuming one or two full working days per week, the correspondence has piled up. The series of titled notes and position papers cited earlier have been generated in self defense as a way to answer the many inquiries. The committee assignment has entered a dormant period. Future written requests for these papers will be answered promptly, but telephone inquiries may remain difficult until the re-write of the book is completed.

(\*Received 77/04/20.\*)



These figures drawn by small PASCAL programs on Terak Corp 850A at UCSD

Contact: K L Bowles  
(714) 452-4526

#### SOME COMMENTS ON PASCAL I/O

While admitting that PASCAL has I/O specifications involving the concept of files and the GET and PUT statements that are consistent with the flavour of the language and with theoretical manipulation of data, I feel that it is lacking in simple, easy to use I/O and in flexible I/O.

In any practical programming application, I/O is used for two main functions:

- (a) Input of data from, and output of results to the real world.
- (b) Permanent storage of data external to the program but internal to the computer, e.g. on tapes or disk.

Concerning the first function, I feel that, notwithstanding the READ function in PASCAL, the use of TEXT files can be rather cumbersome and tedious. This is particularly so when dealing with string input (what delimits the string?) and when being used by a beginning programmer. I would like to see some form of simple I/O akin to the free format I/O of the PL/I GET LIST and PUT LIST concepts.

I have less of a complaint concerning the second function, but would suggest that information to be stored is often not homogenous as is effectively required by PASCAL files. One could argue that different types of data should be stored in different files, but this raises the problem of correlating the data in the files. Alternatively, one could use a file based on a RECORD type with a variant part, but this implies a varying size to the logical units of the file and may be difficult or cumbersome to implement on some computers. Finally it would be nice to be able to easily randomly access files and to update existing files in place.

I have not yet sufficiently formalised any alternative or additional I/O specifications for PASCAL and would be interested in hearing from anyone with ideas along these lines. Note that I consider it essential that any such specifications should as far as possible follow the PASCAL principle of being machine independent.

Chris Bishop  
Computing Centre  
University of Otago  
P.O. Box 56,  
Dunedin  
NEW ZEALAND.

(\*Received 77/04/07.\*)

