PASCAL USER'S GROUP

Pascal News

(FORMERLY PASCAL NEWSLETTER)

NUMBER 12

COMMUNICATIONS ABOUT THE PROGRAMMING LANGUAGE PASCAL BY PASCALERS

JUNE, 1978

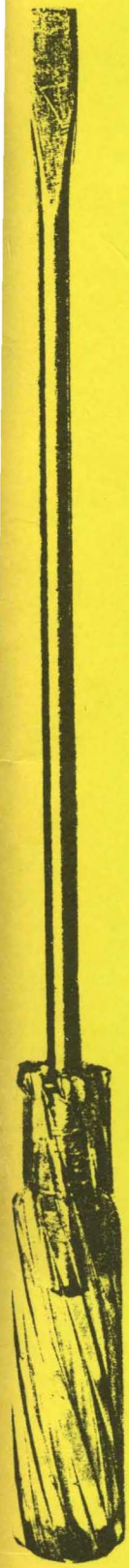


TABLE OF CONTENTS

POLICY: PASCAL NEWS (78/04/15)

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

  Pascal News contains all we (the editors) know about Pascal; we use it as the vehicle to answer all inquiries because our physical energy and resources for answering individual requests are finite. As PUG grows, we unfortunately succumb to the reality of (1) having to insist that people who need to know "about Pascal" join PUG and read Pascal News - that is why we spend time to produce it! and (2) refusing to return phone calls or answer letters full of questions - we will pass the questions on to the readership of Pascal News. Please understand what the collective effect of individual inquiries has at the "concentrators" (our phones and mailboxes). We are trying honestly to say: "we cannot promise more than we can do."

* An attempt is made to produce Pascal News 3 or 4 times during an academic year from July 1 to June 30; usually September, November, February, and May.

* ALL THE NEWS THAT FITS, WE PRINT. Please send material (brevity is a virtue) for Pascal News single-spaced and camera-ready (use dark ribbon and 18.5 cm lines!).

* Remember: ALL LETTERS TO US WILL BE PRINTED UNLESS THEY CONTAIN A REQUEST TO THE CONTRARY.

* Pascal News is divided into flexible sections:

  POLICY - tries to explain the way we do things (ALL PURPOSE COUPON, etc.).

  EDITOR'S CONTRIBUTION - passes along the opinion and point of view of the editor together with changes in the mechanics of PUG operation, etc.

  HERE AND THERE WITH PASCAL - presents news from people, conference announcements and reports, new books and articles (including reviews), notices of Pascal in the news, history, membership rosters, etc.

  APPLICATIONS - presents and documents source programs written in Pascal for various algorithms, and software tools for a Pascal environment; news of significant applications programs. Also critiques regarding program/algorithm certification, performance, standards conformance, style, output convenience, and general design.

  ARTICLES - contains formal, submitted contributions (such as Pascal philosophy, use of Pascal as a teaching tool, use of Pascal at different computer installations, how to promote Pascal, etc.)

  OPEN FORUM FOR MEMBERS - contains short, informal correspondence among members which is of interest to the readership of Pascal News.

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

* Volunteer editors are (addresses in the respective sections of Pascal News):

  Andy Mickel - editor
  Jim Miner and Tim Bonham - Implementation Notes editors
  Sara Graffunder - Here and There editor
  Rich Stevens - Books and Articles editor
  Rich Cichelli - Applications editor
  Tony Addyman - Standards editor
  Scott Bertilson, John Easton, and Steve Riesman - Tasks editors

Policy

PASCAL USER'S GROUP

USER'S

GROUP

ALL PURPOSE COUPON

****

(78/04/15) •

Pascal User's Group, c/o Andy Mickel
University Computer Center: 227 EX
208 SE Union Street
University of Minnesota
Minneapolis, MN 55455 USA

← *Clip, photocopy, or reproduce, etc. and mail to this address.*

/ / Please enter me as a new member of the *PASCAL USER'S GROUP* for ___ Academic year(s) ending June 30, _____ (not past 1982). I shall receive all the issues of *Pascal News* for each year. Enclosed please find _____. (* Please see the POLICY section on the reverse side for prices and if you are joining from overseas, check for a PUG "regional representative." *)

/ / Please renew my membership in *PASCAL USER'S GROUP* for _____ Academic year(s) ending June 30, _____ (not past 1982). Enclosed please find _____.

/ / Please send a copy of *Pascal News* Number(s) _____. (* See the *Pascal News* POLICY section on the reverse side for prices and issues available. *)

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

/ / You messed up my address / phone. See below.

/ / Enclosed please find a contribution (such as what we are doing with Pascal at our computer installation), idea, article, or opinion which I wish to submit for publication in the next issue of *Pascal News*. (* Please send bug reports to the maintainer of the appropriate implementation listed in the *Pascal News* IMPLEMENTATION NOTES section. *)

/ / None of the above. _____

_____

_____

_____

Other comments:

From: name _____

mailing address _____

_____

_____

_____

phone _____

computer system(s) _____

date _____

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

POLICY

JOINING PASCAL USER'S GROUP?

- membership is open to anyone: particularly the Pascal user, teacher, maintainer, implementor, distributor, or just plain fan.
- please enclose the proper prepayment (checks payable to "Pascal User's Group"); we will not bill you.
- please do not send us purchase orders; we cannot endure the paper work! (If you are trying to get your organization to pay for your membership, think of the cost of paperwork involved for such a small sum as a PUG membership!)
- when you join PUG anytime within an academic year: July 1 to June 30, you will receive all issues of Pascal News for that year unless you request otherwise.
- please remember that PUG is run by volunteers who don't consider themselves in the "publishing business." We produce Pascal News as a means toward the end of promoting Pascal and communicating news of events surrounding Pascal to persons interested in Pascal. We are simply interested in the news ourselves and prefer to share it through Pascal News, rather than having to answer individually every letter and phone call. We desire to minimize paperwork, because we have other work to do.
- American Region (North and South America): Join through PUG(USA). Send $6.00 per year to the address on the reverse side. International telephone: 1-612-376-7290.
- European Region (Europe, North Africa, Western and Central Asia): Join through PUG(UK). Send £4.00 per year to: Pascal Users' Group/ c/o Computer Studies Group/ Mathematics Department/ The University/ Southampton SO9 5NH/ United Kingdom. International telephone: 44-703-559122 x700.
- Australasian Region (Australia, East Asia - incl. Japan): Join through PUG(AUS). Send $A8.00 per year to: Pascal Users Group/ c/o Arthur Sale/ Dept. of Information Science/ University of Tasmania/ Box 252C GPO/ Hobart, Tasmania 7001/ Australia. International Telephone: 61-02-23 0561.

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

RENEWING? (Costs the same as joining.)

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

ORDERING BACKISSUES OR EXTRA ISSUES?

- our unusual policy of automatically sending all issues of Pascal News to anyone who joins within an academic year (July 1 to June 30) means that we eliminate many requests for backissues ahead of time, and we don't have to reprint important information in every issue--especially about Pascal implementations!
- Issues 1, 2, 3, and 4 (January, 1974 - August, 1976) are out of print.
- Issues 5, 6, 7, and 8 (September, 1976 - May, 1977) are out of print. (A few copies of issue 8 remain at PUG(UK) available for £2 each.)
- Issues 9, 10, 11, and 12 (September, 1977 - June, 1978) are available from PUG(USA) all for $10 and from PUG(AUS) all for $A10.
- extra single copies of new issues (current academic year) are: $3 each - PUG(USA); £2 each - PUG(UK); and $A3 each - PUG(AUS).

SENDING MATERIAL FOR PUBLICATION?

- check the addresses for specific editors in Pascal News. Your experiences with Pascal (teaching and otherwise), ideas, letters, opinions, notices, news, articles, conference announcements, reports, implementation information, applications, etc. are welcome. "All The News That Fits, We Print." Please send material single-spaced and in camera-ready (use a dark ribbon and lines 18.5 cm wide) form.
- remember: All letters to us will be printed unless they contain a request to the contrary.

MISCELLANEOUS INQUIRIES?

- Please remember that we will use Pascal News as the medium to answer all inquiries, and we regret to be unable to answer individual requests.

UNIVERSITY OF MINNESOTA
TWIN CITIES

University Computer Center
227 Experimental Engineering Building
Minneapolis, Minnesota 55455

(612) 376-7290

The DEADLINE for PUGN 13/14 is August 15. Tony Addyman is now PUG's new Standards Editor. Don't forget to renew if you need to--check your mailing label.

Personal Observations

1) Pascal-P has enabled a great many people to learn about compilers who otherwise would never have had the chance. Do you realize the implications? These same people (myself included) will never be able to look at other compilers for other languages (especially the ones peddled by manufacturers) the same way from now on. Our critical eyes probably won't be able to endure them either.
2) Please see the Books and Articles section for the article entitled: "Ambiguities and Insecurities in Pascal," which is the first, good, critical article about Pascal to appear (yes, we know about Habermann's article). The most memorable passage is in the conclusion:

> "...Because of the very success of Pascal, which greatly exceeded the expectations of its author, the standards by which we judge such languages have also risen. It is grossly unfair to judge an engineering project by standards which have been proved attainable only by the success of the project itself, but in the interests of progress, such criticism must be made."

3) Many people are now decrying the lack in Pascal of "business-oriented" language features such as indexed-sequential access methods for file processing, packed decimal data types, and other inefficient ways of doing computing. I would suggest a Business Procedure Library similar to the IMSL and NAG mathematical and statistics libraries for numerical (old term = 'scientific') people. We should use the simple, but versatile tools (language features) we already have to build what we need for other things.
4) We need more news (notices, articles, opinions, etc.) for Pascal News about teaching experiences with Pascal.

How is Pascal User's Group? (*new members especially please read this*)
PUG has now grown too large to handle it in the personal manner we have in the past. Membership stands at 2147+. We used to be extremely efficient, because I, for one, could keep it all in my head and remember who was a member from where, which joined when and how. It was like stamp collecting. We have resorted to dropping all kinds of services we never promised to do but nevertheless did. Now when a new member joins, all he or she receives is backissues, and no personal reply, receipt, or answers to questions. PUG is another example illustrating limits to growth.

The event that seems to have changed the situation permanently was the first full-page article about Pascal in the April 27 issue of Computerworld--the largest and most widely-read computer journal in the United States. The following Monday we received 83 pieces of mail in one day (old record for a single day was 39 pieces, while typical mail in the past averaged 20-30 pieces/week.)! Do you realize how much time it takes to open 83 pieces of mail? Remember, we don't have secretaries.

PUG(USA) has managed to break even in the past--including this year--but we must raise the rates to $6 per year. Postage and printing costs keep rising. David Barron at PUG(UK) announces new rates of 4 per year and Arthur Sale at PUG(AUS) announces a $A2 decrease (now $A8). Please see their notices following. At least now our rates are more normalized. We have kept the rate low to attract members and to spread Pascal as fast and as far as possible. We as a group are an exceptionally broad base of people, and I think that is a real accomplishment. And remember, we accept no advertizing.

We have always tried to keep this operation simple: no special services, no special rates for special mailing, etc. I know I just wouldn't have time otherwise. I set up PUG so that it can be dismantled within one week and all money refunded! Charging a little more money this year will allow us to hire a part-time secretary to handle the growing clerical workload. The most time-consuming process is to process memberships and update the mailing list. We usually batch 3 or 4 weeks of mail before we process it!

Editor's Contribution

- Andy

78/05/02.

Dear Andy,

Here is our cost estimates for Australasian distribution for 1978/9. As you will see, I am recommending a lowering of the fee to $A8.00. Last year's fee was based on estimates from our printery which in the even proved slightly high, and of course the amalgamation of issues 9 and 10 saved us postage. Consequently we have a small reserve, and I have been able to budget for exactly balancing costs with subs in 1978/79, carrying any inflation in postage and the costs of carrying stocks of back copies out of the reserve.

There may be some request for refunds from people who paid for two years. I'd rather not be involved in sending out cheques, and I suggest we treat this the same as with people who pay for two years in a price rise situation: we don't ask for more so we shouldn't give refunds.

Printing cost per issue		$1.00
Postage : Australia	$0.70	
New Zealand	$1.20	
Singapore	$2.00	
averaged over subscribers		$1.00 approx.
Cost per issue		$2.00

Recommended subscription for 1978/79 = $8.00 (Australian)

Yours sincerely,

Arthur Sale

A.H.J. Sale,
Department of Information Science.

PASCAL USERS GROUP - European Region Subscriptions 1978/79

We regret that steep increases in the cost of printing compel us to increase the subscription to £4 PER ANNUM.

We regret the increase, but even at this figure we shall only just break even. Without volunteer labour, charges would be much higher.

Please remember that cheques must be in sterling, drawn on a British (or Irish) bank Processing sterling cheques drawn on foreign banks, or non-sterling cheques is prohibitively expensive.

If you have a Post-Giro account, you can pay by direct transfer into our account number 28 513 4000.

RENEWALS take time, which is precious. Why not subscribe for two or more years?

(********)

REMEMBER TO RENEW YOUR PUG MEMBERSHIP. NOTE THAT THE RATES HAVE CHANGED!

(********)

Here and There With Pascal

NEWS

Pascal Jobs

(* PUG Member Jack Laffe has been keeping track of some of the jobs for Pascalers which have been advertised in recent months. We decided to publish it as one more indication of the currency of Pascal. This is not a "Help Wanted" section; in fact, these jobs may have been filled. We may continue publishing this section, when space permits, if someone like Jack will compile the list. *)

(* The first job was advertised in CACM in January. The others all appeared in Computer World on the date indicated with the job description. *)

Softech: compiler design.
Dunhill Personnel Inc.: 78/01/02
Modular Computer Systems: 78/01/23
National Cash Register: compiler design for Pascal-like language: 78/01/30
Timeshare: applications, systems: 78/02/06
Amdahl: systems programmer with Pascal experience: 78/02/27
California State Universities and Colleges: instructional consultants in Pascal: 78/03/06
GTE Sylvania: software engineers: 78/03/20
Houghton-Mifflin: Pascal programmers: 78/3

Tidbits

Richard E. Adams, 239 Chatham Road, Columbus, OH 43214: "Did you hear that Burroughs was implementing Pascal on a microprocessor? (They were advertising for people in Computerworld." (* 78/02/09 *)

Wayne Andrews, Electronics Department, Weber State College, 3750 Harrison Blvd., Ogden, UT 84408: "We just put Pascal on our Dec-10 system and are trying to get going on the project." (* 78/03/06 *)

C. Bailey, Bailey and Associates, 1144 S. Atlanta, Tulsa, OK 74104: "I have an Altair with 32K memory, 2 Altair Floppy Discs and a Decwriter. So I am interested in Pascal as implemented on the Altair or Altair-like CPU. I am employed as a programmer/analyst for the Altair and HP and SG minis." (* 77/12/30 *)

Francis H. Beardon, Manager of Projects, Data Systems, Cincinnati Electronics, 2630 Glendale-Milford Road, Cincinnati, OH 45241: "We at Cincinnati Electronics Corporation are interested in Pascal as a possible standard programming language for our developed software systems because of its projected portability." (* 78/02/13 *)

David J. Bell, 609 Craig Ave., Campbell, CA 95008: "My personal system is a Processor Tech SOL-10, with externally expanded memory and I/O. I am interested in developing a Pascal translator for this computer, and for the HP2112 I use at work." (* 78/03/10 *)

Brad Blasing, 1308 Centennial Hall, Univ. Of MN, Minneapolis, MN 55455: "We have implemented the Netherlands Pascal compiler on our 11/40 running UNIX. Runs fast for an interpreter. It's a good hybrid of the P2 and P4 compiler. Could use a bit more user-type documentation." (* 78/04/02 *)

William R. Blatchley, Measurement Systems Div., Siemans Corp., 3 Computer Drive, Cherry Hill, NJ 08002: "We are engaged in test equipment design and development for memory devices and have a possibly immediate need for a Pascal implementation on a PDP-11 for testing magnetic bubble memories." (* 78/04/12 *)

Damon Blom, 72 Sandburg Drive, Sacramento, CA 95819: "I am presently using Pascal on an IBM 370/168 computer using a Pascal compiler written in XPL. I will be getting shortly the Pascal compiler written in Pascal developed at SLAC, Stanford University." (* 78/03/05 *)

Richard J. Cichelli, 901 Whittier Dr., Allentown, PA 18103: "Joseph Mezzaroba at Villanova has supervised two projects to implement Pascal-S on the 370. One using the AAEC Pascal compiler (Pascal-S in Pascal--takes 100 seconds to compile) and the IBM PL/1 compiler (Pascal-S in PL/1--takes 36 minutes to compile). Pascal-S in Pascal was 30 per cent smaller and ran five times faster." (* 78/04/14 *)

Roger Creamer, CTB/McGraw-Hill, Del Monte Research Park, Monterey, CA 93940: "Also, any specific information which you could provide on Pascal implementations for the IBM 370 and DEC PDP-11 would be much appreciated." (* 78/04/14 *)

Anthony Conti, Box 1201, Concord, NH 03301: "I am a user of a Data General Eclipse S200 minicomputer and am interested in running and maintaining Pascal on it." (* 78/01/12 *)

Jean-Louis Decoster, Lyss-Str. 21, CH-2560 Nidau, Switzerland: "Could you inform me too if a Pascal compiler is alre_ady implemented for "Could you inform me too if a Pascal compiler is already implemented for the Motorola 6800?" (* 78/03/15 *)

Alan Delwiche, Computer Programming Instructor, Leland High School, 6677 Camden Ave., San Jose, CA 95120: "Would you please send me any information regarding versions of Pascal for an 8080 or Z80 microprocessor. We have a 32K Cromemco with dual minifloppy drives." (* 78/02/08 *)

Shaun Devlin, 6854 Cedarbrook, Birmingham, MI 48010: "I would also appreciate it if you could direct me to anyone who has or is planning to implement Pascal on a Texas Instrument 990/9900 system." (* 78/01/05 *)

Bob Dietrich, M.S. 60-456, Tektronix, Inc., P.O. Box 500, Beaverton, OR 97077: "Am bringing up solo-concurrent Pascal under RSTS/E time sharing system (PDP-11). Also involved with Swedish and BSM Pascals for PDP-11." (* 78/03/08 *)

Robert Emerson, Honeywell Information Systems, 9555 S.E. 36th St., Mercer Island, WA 98040: "Another interest of mine is implementing a Pascal compiler on the Honeywell Level 6 mini computer. Any tips for compiler implementation would also be appreciated." (* 78/01/17 *)

Mel R. Fisher, Business Dept., Calvary Community Church, 1175 Hillsdale Ave., San Jose, CA 95118: "I am in the process of writing specialized programs for our church records, bookkeeping, and data of this nature. The Pascal language sounds very interesting, and I would appreciate any further information that you could supply me with. We currently have an IMSAI 8080 48K memory, with floppy disk video display and printer." (* 78/02/15 *)

George H. Golden, Sr., Computer Center, SUNY-Fredonia, Fredonia, NY 14063: "We are trying to get Pascal running on the Burroughs B-4700. It runs. But takes too much core." (* 78/04/10 *)

Robert M. Green, Robelle Consulting, Ltd., No. 130, 10th Ave., Delta, BC V4M 3T9: "Could you let me know if there are any implementations of Pascal for the Hewlett-Packard 3000 computer? If not, I am interested in implementing it. Is there any way I can get a copy of the Portable Pascal compiler, version P4?" (* 78/02/02 *)

R. Gunzenhauser and R. Kleine-Homann, Institut fur Informatik, Universitat Stuttgart, 7 Stuttgart 1, Azenbergstr. 12, Germany: "We use Pascal as the first programming language for our freshman students and for high-school teachers.
"We offer Pascal at our German Computer TR 440; besides we have a DEC PDP 11/40 computer (OS DEC RSX 11-M, 92kBytes) and wish to implement Pascal or a Pascal subset like Pascal-S.
"We would be very obliged if you could send us information about Pascal implementations on RSX 11-M you know." (* 78/03/15 *)

Robert O. Harris, University College London, Computer Center, 19 Gordon Street, London WC1A 0AH, United Kingdom: "I read the bit on PUG finances and noticed that PUG (UK) were the big loss makers, so I reckon its time to stop reading the library copy and pay

for my own." (* 78/02/27 *)

Carroll Hennick, Autologic, Inc., 1050 Rancho Conejo Blvd., Newbury Park, CA 91320: "Your letter in SIGPC Notes was welcome."

Judy Herron, Computer Sciences Dept., Mt. San Antonio College, 1100 North Grand Avenue, Walnut, CA 91789: "In my recent reading, references to Pascal seem to pop up everywhere--although I have yet to see one line of source code.
"I'm interested in learning what I can about the language, and its implementations. What manufacturers offer Pascal? Is there a compiler available for our Altair 8800, Xerox 530, or IBM 1130? It sounds as though Pascal is used mainly for the teaching of structured programming techniques. Are business and industry adopting it also?" (* 77/12/28 *)

Bruce Hillegass, Digital Equipment Corp., 146 Main St., Maynard, MA: "I obtained your name off a Pascal document located on one of our DEC Sys-10's.
"Pascal is virtually unsupported on all of our in-house systems, and there are numerous versions of the compiler around. I have been interested in Pascal for quite a while, and I'm in the process of learning the language. I am exploring the possibility of writing a compiler using Pascal as the language and I'm looking into Pascal as a language used in micro-programming.
"I would appreciate any information you may have on Pascal activities in university environments especially on the DEC Sys-10." (* 78/01/27 *)

Robert M. Hofkin, APIS Dept. C-014, Univ. Of CA-San Diego, La Jolla, CA 92093: "Language extensions seem necessary, but the syntax. Let's not have another PL/I! Also--why wasn't Cichelli's review of Ken Bowles' book critical? It sounded more like a product announcement from IBM." (* 78/03/17 *)

David Holland, P.O. Box 38243, Houston, TX 77088: "In case you don't know already, T.I. Are getting ready for a Pascal compiler on a ROM for their 16-bit TMS9900 MP." (* 78/01/31 *)

William F. Holmes, Washington University, School of Medicine, 660 South Euclid Ave., St. Louis, MO 63110: "We are not using Pascal at present, but are seriously considering it for the PDP-11 (including the LSI-11) and the 8080 or 6800. We also have Computer Automation's LSI-2's, but unfortunately do not use their operating system." (* 78/01/30 *)

William C. Hopkins, 1101 Bondsville Rd., Downingtown, PA 19335: ". . . still working on a Univac 90/70 implementation." (* 78/02/26 *)

Gary M. Huckabay, Department of Mathematics, Cameron University, Lawton, OK 73505: "I would appreciate information concerning the following: i) language definition, ii) implementation at any computer site, iii) any suggestions on implementation, iv) any information concerning implementation on the Hewlett-Packard 3000, Series II." (* 78/01/26 *)

Phil Hughes, P.O. Box 2847, Olympia, WA 98507: "I have been studying and debating whether to implement Pascal on a micro for over 6 months. The article 'Pascal vs. Basic' made me aware of two things: 1. There is a Pascal Newsletter. 2. I have been wasting my time thinking about what would make Basic better.
"Please send me information on obtaining the Pascal Newsletter and any information you may have about implementations of Pascal on micros (particularly M6800's)." (* 78/01/19 *)

Joseph M. Jolda, Bartlett High School, Negus St., Webster, MA 01570: "I've been trying to build something around the IBM Assembler but I'm running into all sorts of problems It seems as though Pascal has the possible answer for me." (* 78/01/09 *)

Ralph Johnson, 1592 N. Broad, Galesburg, IL 61401: "I am rewriting Concurrent Pascal for the PDP-11/40 which should take about two weeks. If no one else has done this, I will send you the few changes that need to be made to the PDP 11/45 version." (* 78/01/04 *)

Here and There With Pascal

Adnnan Khan, 222/7, Block-E, (Opp. Walton Training Centre), Walton Road, Lahore, Cantt., Pakistan: "I would like to get some knowledge about the new developments made after my contribution of Source Library Mechanism for Pascal 1900, under George III, which has also reduced the compilation time by one third. My project also involved translation of some NAG routines into Pascal." (* 78/01/17 *)

James R. Kochanocicz, Dedicated Systems Inc., 180 N. Michigan Ave., Chicago, IL 60601: "We are presently using Pascal on a Sperry Univac V-76 series computer." (* 78/03/19 *)

Charles Kuhlman, New York City Criminal Justice Agency, 305 Broadway, New York, NY 10007: "We are preparing to gear up a DEC PDP 11/70 RSX-11P system and are contemplating use of Pascal for some applications. . . . Do you know specifically of any RSX 11/70 versions of Pascal?" (* 78/03/06 *)

Roland L. Lee, 645 35th Ave., San Francisco, CA 94121: "I am thinking of writing a compiler for the Z-80 and would like some information on existing resident Pascal compilers that you know of for the Z-80." (* 78/04/01 *)

Alan M. Lesgold, LRDC Computer Facility, University of Pittsburgh, 3939 O'Hara St., Pittsburgh, PA 15260: "I would be interested in knowing of sources, if they exist, for a 6800 cross-compiler that would run on a PDP-10 or PDP-15 and also for a PDP-15 compiler. I am very interested in implementing Pascal as our primary source language." (* 78/01/12 *)

Bruce MacAnespie, 600 N. Hickory Ave., Apt. 18, Bel Air, MD 21014: "If you can supply me with any contacts or information regarding Pascal compilers or interpreters implemented on Burroughs B6700 or B7700 Computer systems, please send it by return mail. Having been a Burroughs Algol fan for some years, I am extremely interested in a language that promises to be the next generation of decent software implementation languages." (* 78/03/08 *)

Mario Magidin, Direccion Genereal de Sistemas y Procesos Electronicos, Subdireccion de Sistemas "B," Corregidora No.8, Centro, Palacio Nacional, Mexico 1, D. F.: "We are the computing facility of the Mexican Ministry of Budget and Planning. With the aid of a CDC Cyber-173 we are supposed to satisfy all the computing requirements of the Ministry, thus, large, so-called commercial type systems are constantly under development and/or running at our place.
"Up to now, all these systems have been programmed in COBOL, and although we are painfully aware of the shortcomings of this approach, (particularly with CDC's COBOL) our solutions were directed mainly towards the use of a preprocessor of the type of Weinberg's Metacobol.
"The idea of replacing COBOL with PASCAL has arisen. I would deeply appreciate your comments on this idea." (* 78/03/31 *)

Bill Marshall, Jr., Sanders Associates, Inc., 24 Simon St., Nashua, NH: "I've been praising and promoting Pascal for five years now . . . it's about time I put my money where my mouth is!"

Irv McKnight, 505 Cypress Point, No. 52, Mountain View, CA 94040: "I have an S-100 8080 system with a NorthStar Disc. Several of us are looking into making the U.C. San Diego Pascal system live in the NorthStar." (* 78/03/27 *)

Ronald D. McRaney, P.O. Box 10097, Station 1, Houma, Louisiana 70360: "I am in the process of putting together a Pascal dedicated PDP 11/03 for my personal use. (* 78/01/04 *)

J. Scott Merritt, 655 S. Fairoaks Avenue, Apt. L-216, Sunnyvale, CA 94086: "Tried to find CACM article mentioned on Page 87 of PUG 11. It wasn't in Dec. '77 or anywhere else I looked. Where can I find it?" (* 78/03/11 We don't really know either; will you write to Amsterdam to ask? *)

Rolf Molich, Software Development Manager, Dansk Data Elektronik Aps., Generajtorvej 6A, DK-2730 Herlev: "Further, I would appreciate it very much if you could tell me the name and address of any person or institution that you may have heard of who is currently developing a Pascal compiler (not an interpreter) for the{ Intel 8080 microcomputer." (* 78/01/24 *)

Allan Moluf, 2317 Knob Hill, Apt. 9, Okemos, MI 48864: "I would like to suggest a new approach for Pascal compilers on small machines. Syntax table-directed parsing techniques are now getting acceptable error recovery and should result in much smaller compilers. If PUG members know of anyone working in this area, please suggest Pascal as a useful language to implement. Most of the code generation and library routines are available in a portable compiler, which should result in an easy project." (* 78/03/21 *)

Freeman L. Moore, Department of Computer Science, Pearce 203-B, Central Michigan University, Mount Pleasant, MI 48859: "For your records, CMU has a Univac 1106 computer with our version of Pascal from U.S. Naval Undersea Center, by M.S. Ball, version 1.1C4." (* 78/03/04 *)

Olav Naess, Welhavensgt. 65, Bergen Norway: "I am interested in a Pascal compiler for the Z-80 system I am building." (* 78/01/17 *)

Heidi L. Neubauer, Coordinated Sciences Lab, Univ. Of Illinois, Urbana, IL 61801: "I am using Pascal to write machine problems assigned in an operating systems course I am taking at the Univ. Of Illinois as a graduate student in Computer Science. Our class has used both standard Pascal and a souped-up version with concurrent processes and semaphores (still under development but workable)." (* 78/03/07 *)

William I. Nowicki, C.S.R.L. Tech B626, 2145 Sheridan Road, Northwestern University, Evanston, IL 60201: "My special interest is the implementations of Pascal for mini-computers, especially PDP-8's and PDP 11's." (* 78/01/08 *)

David J. Pesec, 20130 Miller Avenue, Euclid, Ohio 44119: "I also am wondering if there is any copy of Pascal that will run on a Honeywell Series 60 processor." (* 78/01/30 *)

David Powers, 259A Trafalgar St., Petersham NSW 2049, Australia: "I have a TEC-9900 system (based on the TMS9900) on which I hope to eventually be able to use Pascal. I would therefore ask if you are able to assist in this--do you know of a Pascal compiler for the 9900, or of any way I could get (with a view to modifying for use with my system) the Pascal source for a compiler with a code generator for the PDP-11. . . or one of the other micros.
"I have been working on an implementation of Pascal-S for the 6502 (using 4-byte words) in the form of a cross-compiler (based on the compiler part of the Wirth Pascal-S interpreter as implemented in Pascal) to an 'ICODE' which runs on an interpreter (only partially debugged, as yet, being a translation of Wirth's 'interpret' procedure) running in 4K (5K-6K with floating point) using the Jolt 'DEMON' monitor. Are you aware of any similar implementations having been undertaken? Has anyone done, to your knowledge, the apparently feasible, but rather time-consuming conversion of this compiler into Pascal-S?"

Steven R. Rakitin, Combustion Engineering, Inc., Mail Stop 9488-4BB, 1000 Prospect Hill Road, Windsor, CT 06095: ". . . I am interested in the potential use of Pascal as a Process Design Language." (* 78/01/24 *)

Mike Rebmann, Memorex Corp., Communications Div., San Tomas at Central Expressway, Santa Clara, CA 95052: "We are potentially interested in adopting Pascal as a replacement for assembly language for programming our 1380 front end communications processor. Does the User's Group have any information on adopting Pascal for this purpose? I would be especially interested in the following kinds of stuff: 1. Compiler development (cost, time, feasibility of using 'weird' hardware features), 2. Cutting over a software development group to use the language (planning training, phasing). 3. Compatibility with existing software--it would be very hard to justify rewriting our existing product line software. 4. Support software development--library system, loaders, etc." (* 78/03/03 *)

D. Roberts, Computing Laboratory, University College of North Wales, Dean Street, Bangor, Gwynedd LL57 1UT, Wales, UK: "We have recently put H.H. Nagel's implementation of Pascal on our DECsystem 10." (* 78/03/17 *)

James D. Rogan, Comshare, Inc., P.O. Box 1588, Ann Arbor, MI 48106: "I have . . . included some documentation on the Pascal compiler implemented on our company's computers. The use of the language is primarily for application production systems software. To date, COMSHARE has written marketable products in Pascal and we can currently cross-compile source for the Sigma 9 and an INTEL 8080 machine." (* 78/02/16 *)

Jon D. Roland, Computer Retailers Assn., Micro Mart, 1015 Navarro, San Antonio, TX 78205: "We plan to support Pascal and extensions thereof extensively during the years ahead. We expect Pascal and APL to emerge as the leading higher-level languages, although Cobol will probably remain popular among many of our business customers." (* 78/03/28 *)

Richard Roth, 5 North Salem Road, Ridgefield, CT 06877: "I implemented P-2 stack machine on Micro-Data 810 (but never finished compiler) and would like to get Pascal running on 8080/Z80 system under my disk OS (an advanced TOPS-10-like operating system)." (* 78/02/01 *)

Beardsley Ruml, 2nd, 3045 Ordway Street, N.W., Washington, DC 20008: "I would like to participate in [an implementation on a Z-80/8080] if possible but, if not, certainly want to be one of the first users." (* 78/01/25 *)

Robert L. Schoenfeld, Rockefeller Univ., 1230 York Ave., New York, NY 10021: "Interested in Concurrent Pascal and Modula for laboratory applications." (* 78/03/23 *)

Mike Settle, ICP, 2925 Merrell Road, Dallas, TX 75229: "I am not presently a user, BUT I WANT TO BE. I am particularly interested in 8080 and Z-80 implementations. I sure would like to see Pascal replace BASIC in the personal and home computing environment." (* 78/02/24 *)

Al Shpuntoff, Morningside College, Sioux City, IA 51106: "I would be delighted to be able to teach some of our courses using the facilities of Pascal, but alas, we are still using an antique IBM 1130 computing system. Still, the widespread availablility of Pascal Compilers for mini-computer systems raised hopes. A direct question to one of the participants in these conversations brought forth the suggestiond that you would know of the existence of a Pascal Compiler for the 1130 if anyone would." (* 78/04/07 *)

Michael L. Sieman, 6103 Harwood Ave., Oakland, CA 94618: "I would also be interested in knowing if the Pascal User's Group has available any other publications, particularly ones concerning the implementation of Pascal on small machines (I'm thinking especially of the DEC LSI-11 under the RT-11 system), or article indexes to past issues of the Pascal News (and are back issues available?)." (* 78/03/23 *)

George A. R. Silver, Earlham College, Richmond, IN 47374: "I am particularly interested in any recent issues which have reviews of implementations of Pascal on PDP 11/70's."

Roger Sippl, 1806 Toyon Lane, Newport Beach, CA 92660: "I learned Pascal while a student at UC Berkeley on the many versions of the compiler on a PDP 11/70, while it was being written and debugged. Not the recommended way to learn a language, but it had its merit. "I am now working as a consultant in California with a special interest in medical computer applications."

James A. Stark, M.D., 485 34th St., Oakland, CA 94609: "My computer resources are: IBM 370/148 at Univ. Of Calif. At San Francisco (Medical School) that has a batch Pascal compiler. UNIX at U.C. Berkeley has just completed the installation of a new interactive version by Joy, Graham, and Haley (complete with manual). I have a home brew 8080 with floppy on which I hope to install UCSD's version and a 6502 presently sitting that will be used to interface my I/O Selectric if and when I get a missing board from Numan Computer Exchange." (* 78/03/28 *)

Quentin F. Stout, Dept. Of Mathematical Sciences, SUNY-Binghamton, Binghamton, NY 13901: "Finally, I would greatly appreciate it if you could tell me where I could obtain a Pascal compiler for an IBM 370/158 under VS1. We are an academic institution which cannot afford a large fee, so we would probably have to obtain it from another university." (* 78/03/17 *)

Jeff Stroomer, 224 Heritage Lane, Exton, PA 19341: "Do you (or any of your readers) know

of a way to get Pascal's IF-THEN-ELSE's into LL(1)? I already know how to monkey with with LL(1) tables to make the parser work the right way, but that's not what I'm interested in; I want a grammar that's truly LL(1)." (* 78/01/13 *)

Roy Touzeau, Computer Science Dept., Univ. Of Montana, Missoula, MT 59812: "We have a version of Pascal for the DEC-10 working on the DEC-20." (* 78/03/07 *)

Mike Travis, Interdata, Inc., 3080 Olcott St., Suite 125A, Santa Clara, CA 95051: "I have just received the KSU version of Pascal which runs on an INTERDATA 8/32. We are now in the process of bringing it up in a multi-terminal environment in our local data center." (* 78/02/13 *)

Tim Walsh, 174 E. Maujer Street, Valley Stream, NY 11580: "I hope to implement a sub-set of Pascal on my 'KIU-1 expanded' sometime this year." (* 78/01/09 *)

Bill Winspur, Mgr., Computer Serv., Computer Dept. For Health Sciences, Univ. Of Manitoba, 753 McDermot Ave., Winnipeg, Manitoba R3E OW3, Canada: "We are installing a CYBER 171 in March and plan to use Pascal on it. We are also getting into uProcessor applications and are particularly interested in a rumour of Pascal for the 8080." (* 78/02/03 *)

C. Dudley Warner, 16345 Los Gutos Blvd., No. 41, Los Gutos, CA 95030: "I have Z80 based uC w/64K mem etc.--running Pascal under CP/M and USCD 'Pascal.'" (* 78/03/08 *)

Anna Watson, 3705 Delwood Drive, Panama City, FL 32407: "My objective is to determine rather quickly whether we should specify a Pascal compiler in a new computer specification for use by our present Algol users. Hopefully, study of a Pascal Primer plus the Pascal News can indicate if Pascal can serve our needs." (* 78/03/20 *)

Chip Weems, Dept. Of Computer Science, Oregon State Univ., Corvallis, OR 97331: "I enjoyed talking to Tim B[onham] at the W. Coast Comp. Faire. Tell him that I'm rewriting my Pascal summary card, and will send him a copy when it's finished." (* 78/03/28 *)

John Withrow, DEC, MR1-1/A86, 200 Forest St., Marlboro, MA 01752: "I'm using the Pascal compiler on the DECSYSTEMS (10 and 20) here at Maynard and Marlboro, MA; as well as implementing a Pascal (subset) compiler." (* 78/01/25 *)

Sandra Wright, Defence and Civil Inst. Of Environmental Medicine, P.O. Box 2000, Downsview, Ont. M3M 3B9, Canada: "We plan on implementing Pascal under UNIX and RT-11 early in 1978." (* 77/11/30 *)

FRENCH/ENGLISH -- ENGLISH/FRENCH Pascal IDENTIFIERS

(* We received the following list of correspondences between French and English Pascal identifiers from Patrick Ward at the University of Montreal. He credits Olivier Lecarme and Pierre Desjardins with the original translation. Since we expect this to be used simply as a reference by those reading programs in the other language, we are omitting CDC-specific identifiers and those local to Montreal. We also have a list made by A. Tisserant at Nancy. His list is slightly different. We'd appreciate some clarification from the Sous-Groupe Pascal about what is standard for the French identifiers. *)

French	English	English	French
abs	abs	abs	abs
allera	goto	and	et
alors	then	arctan	arctan
arctan	arctan	array	tableau
arrondi	round	begin	debut
avec	with	boolean	booleen
bas	downto	case	cas
booleen	boolean	char	car
car	char	chr	carac
carac	chr	const	const
carre	sqr	cos	cos
cas	case	dispose	rendre
const	const	div	div
cos	cos	do	faire
dans	in	downto	bas
de	of	else	sinon
debut	begin	end	fin
detasser	unpack	eof	fdf
div	div	eoln	fdln
ecrire	write	exp	exp
ecrireln	writeln	false	faux
ensemble	set	file	fichier
entier	integer	for	pour
entmax	maxint	forward	plusloin
entree	input	function	fonction
et	and	get	prendre
etiqu	label	goto	allera
exp	exp	if	si
faire	do	in	dans
faux	false	input	entree
fdf	eof	integer	entier
fdln	eoln	label	etiqu
fichier	file	ln	ln
fin	end	maxint	entmax
fonction	function	mod	mod
haut	to	new	nouveau
impair	odd	nil	nil
jusque	until	not	non
lire	read	odd	impair
lireln	readln	of	de
ln	ln	or	ou
mettre	put	ord	ord
mod	mod	output	sortie
nil	nil	pack	tasser
non	not	packed	paquet
nouveau	new	page	page
ord	ord	pred	pred
ou	or	procedure	procedure
page	page	program	program
paquet	packed	put	mettre
plusloin	forward	read	lire
pour	for	readln	lireln
pred	pred	real	reel
prendre	get	record	struct
procedure	procedure	repeat	repeter
programme	program	reset	relire
rac2	sqrt	rewrite	recrire
recrire	rewrite	round	arrondi
reel	real	set	ensemble
relire	reset	sin	sin
rendre	dispose	succ	succ
repeter	repeat	sqr	carre
si	if	sqrt	rac2
sin	sin	text	texte
sinon	else	then	alors
sortie	output	to	haut
struct	record	true	vrai
succ	succ	trunc	tronc
tableau	array	type	type
tantque	while	unpack	detasser
tasser	pack	until	jusque
texte	text	var	var
tronc	trunc	while	tantque
type	type	with	avec
var	var	write	ecrire
vrai	true	writeln	ecrireln

PASCAL IN THE NEWS

Australian (* A national daily newspaper *), February, 1978: Article in the "Computers" section about the Australian Atomic Energy Commission's compiler for the IBM 360 and 370 systems.

Byte, April, 1978: A letter from Stephen Smith describing the status of his work on a Pascal compiler, based on a subset of Pascal, for microcomputers. He is now testing the parsing procedures on a DECsystem 10.

Computer Weekly, February 23, 1978: NCR, Dundee, Scotland, is beginning to design and implement a language based on Pascal.

Computerworld, March 20, 1978: Hewlett-Packard's new language for operating system implementation, Syspal, combines many features of Pascal, Modula, Euclid, and Concurrent Pascal.

Computerworld, April 24, 1978: Richard Cichelli describes the "revolutionary" growth in use of Pascal, this despite the resistance of mainframe and system vendors. A short history of Pascal and the extent of implementations is presented.

Computing, January 5, 1978: A letter to the editor from R. J. Allwood in response to David Barron's earlier article in Computing. Allwood announces his reasons for rejecting a changeover from FORTRAN to Pascal and states what a tempting new language would look like.

Computing Europe, March 16, 1978: David Barron notes the choice of Pascal as a base for the U.S. Department of Defense language IRONMAN.

DARCOM (U.S. Army Materiel Development and Readiness Command) sent letters to PUG members on February 1, 1978, asking for their responses to a series of questions about use and implementation of Pascal. Purportedly, DARCOM is selecting a standard system software programming language. (* DARCOM got your name by copying it from the published roster. PUG has a general policy of not releasing the roster in machine-retrievable form.*)

Data-Link (* published by ACM-Los Angeles *), February 1978: G. S. Khalsa, managing partner of the Pasadena Byte Shop, is reported to view Pascal as becoming the standard language for micro business systems.

Datamation, February, 1978: A short announcement about PUG and Pascal News with information about how to join appeared in the "Source Data" section.

Datamation, February, 1978: A proposed multiprocessor system for the U.S. Navy, constructed by Lawrence Livermore Laboratories in California, contains a Pascal compiler, developed under subcontract by the Computer Sciences Department of Stanford Univ.

Instruments and Control Systems, December 1977: A report of a Pascal compiler under development by Texas Instruments to meet Dept. Of Defense specifications. The article suggests that TI's Pascal could become a de facto standard for minis and micros. Unlike Intel's PL/M, Pascal is not a proprietary language.

Journal of the Hewlett-Packard General Systems Users Group, January/February 1978: A short article introducing Pascal and some of its features and containing information about how to join PUG.

Mini-Computer News, April 27, 1978: A new Pascal software package for the DS990 packaged disk systems is announced by Texas Instruments. TI suggests that its Pascal, closely compatible with standard Pascal, has many applications in areas traditionally dominated by FORTRAN and COBOL.

PATCH (Univ. Of Notre Dame Computing Center's newsletter), March 1978: UND has recently installed Pascal.

UMMUG (Publication of the Univ. Of Minnesota Microcomputer Users Group), The University of Minnesota's recent acquisition of Terak computers and with them UCSD's Pascal compiler/interpreter is discussed.

Vogelback Computing Center Newsletter (Northwestern Univ.), April, 1978: In announcing a short course on Pascal, an article mentions the widespread acceptance of Pascal.

CONFERENCES

Australian Universities Computer Science Seminar, held February 23-24, 1978, University of New South Wales:

(* We received a letter from Tony Gerber saying that "everyone (Carroll (* Morgan *), Ken Robinson, Arthur Sale, Jeff Tobias, & Gordon Cox from AAEC, myself) was there." In addition, Tony sent us copies of two papers read at the conference:

G. W. Cox and J. M. Tobias, "An Implementation of Pascal for International Business Machines or The Impossible Takes a Little Longer."

(* From the abstract *) The programming language Pascal has successfully implemented for IBM360 and IBM370 computers under the OS/360 family of operating systems. The compiler is written in Pascal and fully supports Standard Pascal with some significant extensions. Interesting aspects of the relationship between the language and the IBM360 architecture are discussed. Surprisingly enough, the IBM360/370 general purpose architecture readily lends itself to an efficient implementation of a high-level language such as Pascal, although some features are impossible to realise.

Experiences in attempting to encourage a body of scientists to use Pascal in preference to FORTRAN are drawn on, with the conclusion that until a revised standard for Pascal is achieved, Pascal will never become a universally used programming tool.

Arthur Sale, "Mismatches and Conflicts Arising out of the Burroughs B6700/B7700 MCP and a Pascal Implementation."

(* From the abstract *) This paper draws on experiences of implementing a Pascal compiler on a Burroughs B6700 computer. Since these machines are designed for high-level language programming solely, and the operating system (MCP) is highly structured, the conflicts between the assumptions commonly made by Pascal adherents, or built into the language, and the facilities offered by the operating system posed some interesting conflicts which are examined herein.

Universite de Nice, Informatique, Mathematiques et Automatique, Manifestations Informatiques de Juin 1978, conference to include a meeting of the Pascal sub-group on the 13th and 14th of June.

(* Sorry we didn't know about this conference in time for the last issue. We'll hope to have titles of the papers presented by next time. *)

Second West Coast Computer Faire, March 3-5, 1978, San Jose, California. (* Pascal News editor Tim Bonham attended. He collected a dozen PUG memberships and reported that he "could have sold 100 Pascal User Manuals and Reports on the floor for twice their price." He also saw several demonstrations of Pascal on micros. Several papers of special interest to Pascalers are part of the proceedings *):

Sassan Hazeghi and Lichen Wang, "A Short Note on High Level Languages and Microprocessors."

(* From the abstract *) In this note, some of the practical aspects of bridging the gap between high level programming language and computer hardware are discussed. Several possible strategies are considered and the method of half-compiling-half-interpreting

is studied. In dealing with address space limitation (or tight memory situation) and slow speed of micro processors running an interpreter, a measurement and analysis technique is suggested. This analysis not only gives a good estimate of the timing and storage requirement before the actual implementation, it also helps to optimize the speed and storage usage of the implementation. The note concludes with some results concerning the implementation of the programming language Pascal on a family of micro-processors.

H. Marc Lewis, "An Experimental Pascal-like Language for Microprocessors.

(* From the abstract *) This paper describes an experimental Pascal-like high level language oriented to microprocessor implementation and use. The design criteria include modest memory requirements, self-compilation, simplicity, reasonable access to hardware features, and ease of extensibility. Program structure, data declarations, and control structures are described and examples given. Novel features of the language are discussed. An appendix gives a formal description of the language via syntax graphs.

Chip Weems, "An Introduction to Programming in Pascal."

(* From the abstract *) This paper will concentrate heavily on the use of the Pascal language at the beginner's level. A minimal knowledge of some other programming language such as FORTRAN, BASIC, or ALGOL is assumed.
The areas which will be covered are simple and structured statements in Pascal, simple and structured data types, plus procedures and functions. Emphasis will be placed on using Pascal statements, although some discussion of the power of user defined data types will also be included.
A list of machine models for which implementations of Pascal are known to exist is provided as an appendix.

BOOKS AND ARTICLES

PLEASE SUBMIT ALL NOTICES OF Pascal

BOOKS, ARTICLES, ABSTRACTS, etc.

to Rich for this section. Thanks, Andy.

Editor: Rich Stevens

Kitt Peak Nat'l Observatory
P. O. BOX 26732
Tucson, AZ 85726 USA

(phone: 1-602-327-5511)

APPLICATIONS

Patricia R. Mohilner, "Using Pascal in a FORTRAN Environment," Software Practice and Experience, 7:3 (June-July 1977), 357-362.
(* Summary of a review by R.A. Jones in Computing Reviews, January 1978. *) Mohilner demonstrates some problems encountered in attempting to write graphical applications programs in Pascal when the existing library of plotting routines was written in FORTRAN. She shows the solutions to those problems, but the example suggests that the problems she describes will likely be encountered by all installations.

V.A. Nepomniaschy, and L.V. Chernobrod, "Automatic Program Verification," Problems of Programming, 1976, pp. 63-80.
(* From the English summary in the table of contents *) "Describing the preliminary version of the system for proving assertions about programs (SPRUT). The deduction system herein is Hoare's system for proving correctness of programs. The input is a Pascal program with assertions. The verification condition generator outputs the list of lemmas to be proved by other blocks of the system. In algebraic and logical reduction of expressions simplification strategies are used, including axioms and lists of subgoals."

Gary J. Nutt, "A Comparison of Pascal and FORTRAN as Introductory Programming Languages," SIGPLAN Notices, 13:2, February 1978, pp. 57-62.
(* From the abstract *) "The Department of Computer Science at the University of Colorado has recently made the transition from FORTRAN to Pascal [in introductory courses], and this paper offers and informal discussion of the experiences of one instructor during that change."

IMPLEMENTATIONS

Urs Ammann, "On Code Generation in a Pascal Compiler," Software Practice and Experience, 7:3 (June-July 1977), 391-423.
(* From the abstract, as reported in Computing Reviews, January 1978. *) "This report deals with code generation in a Pascal compiler. It gives insight into the run-time organization of data and the use of the hardware registers of athe underlying machine (a CDC 6400). It is shown how the compiler maintains a description of the register contents and uses this description to generate efficient code. Several examples of compiled code are discussed."

Forest Baskett, "The Best Simple Code Generation Technique for WHILE, FOR, and DO Loops," SIGPLAN Notices, 13:4 (April 1978), pp. 31-32.
(* From the abstract *) "This code generation technique for WHILE, FOR, and DO loops is simple to implement and usually results in the best loop code in the absence of flow analysis. Also the technique makes it possible to move code from inner loops without doing flow analysis and without ever moving code from a less frequently executed block to a more frequently executed block."

Kenneth L. Bowles, "The USCD Pascal Project," Educom, 13:1 (Spring, 1978), pp. 2-7.
(* From the summary *) "Small stand-alone microcomputers can serve as the basis for running a sophisticated general-purpose interactive software system capable of supporting CAI, word processing, data processing, and other interactive tasks in addition to development of the software itself. The project described in this article has implemented such a software system using the Pascal programming language. The system is designed to be nearly machine-independent, and currently runs on a number of microprocessors, including the popular LSI-11, 8080, and Z80."

G. W. Cox and J. M. Tobias, "An Implementation of Pascal for International Business Machines or the Impossible Takes a Little Longer." (* See CONFERENCES section *)

Sassan Hazeghi and Lichen Wang, "A Short Note on High Level Languages and Microprocessors." (* See CONFERENCES section *)

H. Marc Lewis, "An Experimental Pascal-like Language for Microprocessors." (* See CONFERENCES section *)

Arthur Sale, "Mismatches and Conflicts Arising out of the Burroughs B6700/B7700 MCP and a Pascal Implementation." (* See CONFERENCES section *)

J. Welsh, "Economic Range Checks in Pascal," Software--Practice and Experience, Vol. 8 (1978), 85-97.
(* From the abstract *) "A Pascal implementation is described which exploits the information provided by subrange type declarations to minimize the run-time checking involved in detecting range violations. An evaluation of its performance is given, and some possible modifications are discussed. (* It pays to use sub-ranges. *)

LANGUAGES

Borge Christensen, "COMAL: Structured Basic," People's Computers, 6:4 (Jan.-Feb. 1978), pp. 36-41.
(* From the table of contents *) ". . . adding Pascal's algorithmic structures to BASIC."

M. Iglewski, J. Madey, and S. Matwin, "A Contribution to the Improvement of Pascal," SIGPLAN Notices, 13:1 (January, 1978), pp. 48-58.
(* From the introduction *) "The purpose of this paper is twofold. First of all we would like to present some of our proposals, concerning the desirable corrections in the Revised Report on Pascal and possible slight extensions of the language. Secondly we want to argue with some of the critical remarks on Pascal as formulated several months ago by Conradi."

Charles Lakos and Arthur Sale, "Is Disciplined Programming Transferable, and is it Insightful?" (* Received in January; may be published by now; more news from Arthur Sale or from PN 13 *)
(* From the abstract *) ". . . The paper applies the thought processes advocated by [E.W.] Dijkstra to [two] problems and indicates the insights that the authors gained from this. In both cases algorithms new to the authors were derived, and the properties of these are also examined. The paper . . . demonstrates that the techniques advocated by Dijkstra are indeed transferable to other programmers, and that this transfer yields better insight into the activity we call programming."

David Mundie, "Pascal vs. BASIC," People's Computers, 6:4 (Jan.-Feb. 1978), pp. 41-47.
(* From the table of contents *) "A polemical comparison of the two as general-purpose microprocessor languages."

Jim des Rivieres and Ted Venema, "Euclid and Pascal," SIGPLAN Notices, 13:3 (March 1978), pp. 57-69. (* From the abstract *) "The programming language Euclid was intended for writing system programs that could be verifiable by state-of-the-art verification methods. Since verification was not an explicit goal in the design of Pascal, it is not surprising that this gave rise to differences between the two languages. The Euclid designers intended to change Pascal only where it fell short of this goal. This paper examines differences in the two languages in the light of this objective. These differences are roughly grouped under the headings verification, system programming, and user-oriented changes."

Abraham Silberschatz, Richard B. Kieburtz, and Arthur Bernstein, "Extending Concurrent Pascal to allow dynamic resource management," IEEE Transactions on Software Engineering, SE-3:3 (May 1977), 210-217.
(* One sentence from a review *) "The authors of this paper propose an extension to the programming language Concurrent Pascal to allow more flexible dynamic resource management. They introduce a new type called manager. . . ."

Tennent, R. D., "Language design methods based on semantic principles," Acta Informatica, 8:2 (1977), 97-112.
(* From the abstract *) "Two language design methods based on principles derived from the denotational approach to programming language semantics are described and illustrated by an application to the language Pascal. The principles are, firstly, the correspondence between parametric and declarative mechanisms, and secondly, a principle of abstraction for programming languages adapted from set theory. Several useful extensions and generalizations of Pascal emerge by application of these principles, including a solution to the array parameter problem, and a modularization facility."

Arthur Sale, "Stylistics in Languages with Compound Statements" (* Article may have been published in Australia; check with Arthur Sale; more information in PN 13. *)
(* From the abstract *) "This short communication discusses a stylistic problem which arises in languages with use both statement separators such as semicolons, and begin-end bracketting structures, such as Pascal. It suggests that an alternative to the traditional rules which have evolved from Algol 60 is preferable."

Chip Weems, "An Introduction to Programming in Pascal." (* See CONFERENCES section. *)

J. Welsh, W. J. Sneeringer, and C.A. Hoare, "Ambiguities and Insecurities in Pascal," Software--Practice and Experience, Vol. 7 (1977), 685-696.
(* This is the best critical article to have appeared about Pascal. The ambiguities discussed are equivalence of types (name equivalence vs. Structural equivalence), scope rules and one-pass compilation, and set constructors. The authors point out the following "insecurities": features whose implementation either risks undetected violation of rules of the language or run-time checking that is too expensive to be tolerable: variant records, functions and procedures as parameters, range violations, uninitialized variables, and dangling references. *)

TEXTBOOKS

(* See reviews for Bowles, Conway, Grogono, and Schneider texts. *)

S. Alagic and M. A. Arbib, The Design of Well-Structured and Correct Programs, New York: Springer-Verlag, 1978, 292 pages, $12.80. (* See description in No. 11. We hope to have a review in No. 13. *)

Richard Kieburtz (* Updated information *), Structured Programming and Problem Solving with Pascal, Englewood Cliffs, NJ: Prentice-Hall, 1977, 320 pages, $12.80.(* Similar to Conway; see reviews. *)

A Guide to PASCAL Textbooks
Richard J. LeBlanc and John J. Goda
School of Information and Computer Science
Georgia Institute of Technology
Atlanta, GA

G. Michael Schneider, Steven W. Weingart and David M. Perlman, An Introduction to Programming and Problem Solving with PASCAL, John Wiley, 1978. ($12.95)

Among the strongest features of this book are its coverage of the complete programming processes (from problem specification through debugging and maintenance) and its emphasis on good programming style. Most of PASCAL is well presented, with the examples giving a good demonstration of how the features of the language should be used. The weakest part of the book is the presentation of "advanced" features such as variant records and pointers. Its coverage of programming fundamentals makes this an excellent text for an introductory course for students with little or no programming experience.

Peter Grogono, Programming in PASCAL, Addison-Wesley, 1978. ($9.95)

While this is an introductory book, it concentrates more on the syntax of PASCAL and less on programming methodology than does the book by Schneider et. al. It is easy to read and has syntax charts integrated with the text rather than in an appendix. Grogano includes very good coverage of the advanced features, particularly pointers and dynamic data structures. The presentation of user-defined types is not as well-organized as it could be. While this book could be used as the text for an introductory course, its lack of coverage of programming fundamentals and its strength in the area of the advanced features make it best for students who have some programming experience.

Richard Conway, David Gries and E. Carl Zimmerman, A Primer on PASCAL, Winthrop, '76. ($10.95)

This book is based on a PL/1 book by Conway and Gries and it shows. The only structured type discussed is arrays and the discussion of user defined scalar types is very weak. The material on programming methodology is not at all integrated with the presentation of the language, so it is necessary to skip around in the book when it is being used as a text in an introductory course. There are errors in the presentation of PASCAL that are clearly not typographical. In general, this book fails to capture the idea that the features of PASCAL can actually make program development easier than if FORTRAN were being used.

Kenneth L. Bowles, (Microcomputer) Problem Solving Using PASCAL, Springer-Verlag, 1977. ($10.95)

(See PASCAL News #11 for a more thorough review.) This basically appears to be a good book, but the language presented is not standard PASCAL. Bowles microcomputer PASCAL (with extensions for graphics and string handling) is used and the examples are heavily dependent on use of the extensions. This tends to make much of the book confusing to a student who does not have Bowles' system available.

ARTICLES WANTED

(* Tim Bonham has been surveying publications which might want articles about Pascal and has supplied us with the following list. *)

Wayne Green
Kilobaud
Peterborough, NH 03458
(603) 924-3873

Especially interested in articles which answer the questions "Why should I use Pascal?" "What uses does Pascal have in the real world?" and "How would I gain from using Pascal?" Maybe later on could use articles on how to program in Pascal. Will pay.

Stan M. Sokolow
Solus News
1690 Woodside Dr.
Redwood City, CA 94061
(415) 368-3331

Wants articles introducing and "justifying" Pascal. Especially interested in machines which are software compatible with the SOL (8080-based with cassette operating system).

Northstar Newsletter
2547 Ninth St.
Berkeley, CA 94710

Especially interested in software for the Northstar products: micro disk system and Horizon computer system. Introductions and justifications.

Larry Steckler, Editor
Radio-Electronics
200 Park Ave. S.
New York, NY 10003
(212) 777-6400

Introductions to programming in Pascal.

ROSTER INCREMENT (78/04/22)

The names listed below represent people who have renewed, changed address, or joined PUG since the roster increment was printed in PUGN 11. Some interesting statistics:

renew date	number	state	number	country	number
77	238	CA	349	USA	1535
78	1557	MN	156	UK	145
79	249	MA	107	Canada	88
80	54	TX	88	Germany	70
81	10	NY	83	Australia	42
82	7	PA	80	Netherlands	35
				Sweden	31
				Switzerland	27
				Denmark	23

RUSSEL J. ABBOTT 91330
REESA ABRAMS 87106
RICHARD E. ADAMS 43214
TONY ADDYMAN M13 9PL UNITED KINGDOM
PETER ALTMANN D-5100 GERMANY
J. AMBLER UNITED KINGDOM
RICHARD M. AMEYE 01752
CHRISTOPHER AMLEY 55455
DAVID E. ANDERSON 53201
PETER ANDERSON 07102
MATS APELKRANS S-350 93 SWEDEN
ROBERTO ARGUETA EL SALVADOR
MICHAEL C. ARMSTRONG 94804
WILLIAM J. ARTHUR 92624
ANTTI ARVELA SF-33340 FINLAND

ATTN: GORDON R. SHERMAN 37916
ATTENTION: G. TER HOFSTEDE M5V 2S9 CANADA
ATTENTION: JERRY W. SEGERS 30332
ATTENTION: R. D. BERGERON 03824
ATTENTION: SANDRA WRIGHT M3M 3B9 CANADA
ATTN: BETTE BOLLING-LIBRARIAN 45036
ATTN: CECICO CHILE
ATTN: CENTRE DE RECHERCHE F-34075 FRANCE
ATTN: CHICO PROJECT - CDC 95929
ATTN: COMPUTER SCIENCE PRESS INC. 20854
ATTN: CUPERTINO LIBRARY 95014
ATTN: DEPT DE MATHEMATIQUE AND INFORMATIQUE F-35031 FRANCE
ATTN: DEPT OF BIOCHEMISTRY LE1 7RH UNITED KINGDOM
ATTN: DEPT. OF COMPUTER SCIENCE 38677
ATTN: DOCUMENTATION OFFICER NEW ZEALAND

ATTN: FREIE UNIVERSITAT BERLIN D-1000 GERMANY
ATTN: FRIEDA S. COHN 53706
ATTN: HELEN SMITH N2L 3G1 CANADA
ATTN: INFORMATION SERVICE CENTER 55113
ATTN: INSTITUT FUER INFORMATIK D-2000 GERMANY
ATTN: INSTITUT FUR INFORMATIK D-2300 GERMANY
ATTN: LAFAYETTE COLLEGE 18042
ATTN: LIBRARY 91109
ATTN: LIBRARY - COPY 2 94305
ATTN: NORTHWEST MICROCOMPUTER SYSTEMS INC. 97401
ATTN: PROGRAMMING ADVISOR 89507
ATTN: PROGRAMMING ADVISOR 89154
ATTN: PURCHASING OFFICE 2600 AUSTRALIA
ATTN: SAM CALVIN 09175
ATTN: SCHOOL OF INFORMATION SCIENCES 2616 AUSTRALIA
ATTN: SERIALS DEPT. 52242
ATTN: SPECIAL INTERACTIVE COMPUTER LAB 55455
ATTN: STUDENT COMPUTING COOP 92093
ATTN: SUSAN BOWIE - DOCUMENTS ROOM 97403
ATTN: THE LIBRARIAN ME99 1LH UNITED KINGDOM
ATTN: UNBOUNDED COMPUTING 94086
ATTN: UNIVERSITAT DE GRENOBLE F-38041 FRANCE
ATTN: TECHNICAL RESEARCH CENTRE OF FINLAND SF-02150 FINLAND
LEE D. AURICH 90230
ROBERT M. BAER 94941
C. BAILEY 74104
F. T. BAKER 20760
JOHN BANNING 94305
ANN BARDIN 47401
ANN V. BARROW UNITED KINGDOM
FRANCIS H. BEARDEN 45241
E. R. BEAUREGARD 17013
GARY BECKWITH 55119
M. D. BEER L69 3BX UNITED KINGDOM
DAVID A. BEERS 92701
E. MAURICE BEESLEY 89557
DAVID J. BELL 95008
ABDUL RASAQ BELLO 55408
STEVEN M. BELLOVIN 27514
GILBERT BERGLASS 14221
ALLEN BERGLUND 85005
THOMAS BERNER D-2000 GERMANY
JOEL BERSON 01821
SCOTT S. BERTILSON 55455
K. S. BHASKAR 68508
LEONARD BINSTOCK 20901
CHRIS M. BISHOP NEW ZEALAND
JUDY M. BISHOP 2001 SOUTH AFRICA
K. W. BIXBY 92634
ANTHONY BJERSTEDT 74171
R. B. T. BLACK DD2 3XX UNITED KINGDOM
ROBERT E. BLANTON 01778
DAMON BLOM 95819
W. ASHBY BOAZ 22101
PETER BOCK 20052
JOHN R. BOGAN 94010
REFORD BOND 73102
RONALD V. BOSSLET 02154
GREG BOURQUE 80901
WILLIAM M. BRACK HONG KONG
JOHN A. BRIGGS 84047
JERRY R. BROOKSHIRE 77840
DAVID H. BROWN 92037
R. LEONARD BROWN 22903
CHRIS BRYCE L8S 4K1 CANADA
MARTIN BUCHANAN 20007
DONALD D. BURN 61801
DEBORAH BUSCH 13346
R. BUSH 97420
RICKY W. BUTLER 23601
JAMES D. CALLADINE L0C 1K0 CANADA
T. M. CAMBRON 32901
JIM CAMERON 43023
SHERRY L. CAMERON 92714
ROBERT L. CANNON 92208
ROY CARLSON 97077
P. CARR BA2 7AY UNITED KINGDOM
JOHN CARTER SR. 95050
GEORGE P. CHENEY 01854
C. W. CHILDERS 94550
EARL N. CHRISTIANSEN 68134
ROBERT G. CLARK FK9 4LA UNITED KINGDOM
W. E. CLARK 92138
JENNIFER CLARKE 02115
JIM CLARKE 76010
RICHARD CLAYTON M20 UNITED KINGDOM
GEOFFREY A. CLEAVE 3165 AUSTRALIA
GERHARDT C. CLEMENTSON 80204

BETTY CLIFFORD T2N 1N4 CANADA
D. S. COCHRANE UNITED KINGDOM
JOE COINTMENT 75248
R. COLE SE1 0AA UNITED KINGDOM
HOWARD S. COLEY JR. 95070
TERENCE M. COLLIGAN 02193
ANTHONY CONTI 03301
MICHAEL J. COOK 95014
D. F. COSTELLO 68583
C. J. COTTON 19898
M. MICHEL COURCHESNE H1G 3S5 CANADA
JOHN COUSINS 97005
LAWRENCE F. CRAM 01701
ROGER CREAMER 93940
A. PAUL CROONENBERGHS 92277
SCOTT CRUMPTON 32601
HOWARD CUNNINGHAM 47904
JOHN H. DALY 08108
THOMAS DANBURY 06880
RONALD L. DANIELSON 95051
GENE A. DAVENPORT 10016
SCOTT T. DAVIDSON 92138
ADELINO CARLOS DE SOUSA PORTUGAL
DENNIS S. DELORME 55901
ALAN DELWICHE 95120
SHAUN DEVLIN 48010
JERRY DI MASSA 95051
ROBERIO DIAS BRAZIL
DAVID R. DICK 02104
CLEMENT L. DICKEY 93407
MARY DIEGERT 13902
BOB DIETRICH 97077
M. JEAN-MARIE DIRAND J1K 2R1 CANADA
FELIX DREHER 66762
LARRY DUBY 22209
DOUGLAS DUNLOP 23185
BRUCE J. EDMUNDSON 94040
JOHN EDWARDS 3083 AUSTRALIA
P. S. EDWARDS 3216 AUSTRALIA
H. W. EGDORF 80401
JOSEPH EINWECK 88003
JOHN ELDER BT7 1NN UNITED KINGDOM
LARRY E. ELLISON 08046
ROBERT EMERSON 98040
JAMES C. EMERY 08540
HANNU ERKIO SF-00100 FINLAND
C. B. FALCONER 06504
MICHAEL FAY 60919
LINWOOD FERGUSON 22901
BILL FINCH 95014
MEL R. FISHER 95118
F. G. FLETCHER 02154
DOUG FORSTER 94611
CRAIG FRANKLIN 03060
R. D. FREEMAN UNITED KINGDOM
ERIC D. FREY 03031
DAVID F. FRICK 94025
KARL FRYXELL 91125
MICHEL GALINIER F-31450 FRANCE
DAVID GARDNER 55113
PHILIP GAUDETTE 22301
RICHARD D. GEORGE 60439
CARSON GERMAN 90706
BRANKO J. GEROVAC 02154
CLIFFORD GERSTENHABER 55343
CHARLES J. GIBBONS 68154
RICK GILLIGAN 93407
DON GILMORE 95030
THOMAS GIVENTER 10583
R. STEVEN GLANVILLE 94087
GEORGE H. GOLDEN SR. 14063
DAVID GRABEL 02174
TOM GRABOWSKI 15108
JEFFREY W. GRAHAM 35223
SUSAN L. GRAHAM 94720
DAVID N. GRAY 78769
ARTIE GREEN 19311
ROBERT M. GREEN V4M 3T9 CANADA
SY GREENFIELD 12206
PETER GROGONO CANADA
RICHARD D. HACKATHORN 19174
WILLIAM L. HAFNER 19424
JACQUES HAGUEL J1K 2R1 CANADA
BYRON HALE 94043
RICHARD W. HAMILTON 94702
BRIAN HANSEN 97213
WILL HAPGOOD 02154
ANDY HARRINGTON 93017
MIKE HARRIS 62704

ROBERT O. HARRIS WC1A 0AH UNITED KINGDOM
ROBERT W. HARRIS 21114
DON HARVEY 97070
W. F. HAYGOOD 84121
PETER HAYNES L5N 1K7 CANADA
SASSAN HAZEGHI 94305
RANDALL HEAP 01752
JOHN HEATH 04103
JUDY HERRON 91789
S. J. HIGGINS CA9 3LP UNITED KINGDOM
TERUO HIKITA 158 JAPAN
BUZZ HILL 98632
BRUCE HILLEGASS 01754
ED HIRAHARA 83704
THOM HOARD 55414
C. A. R. HOARE OX2 6PE UNITED KINGDOM
FRED M. HOFKIN 19117
WILLIAM HOLMES 63110
JAMES W. HOLT JR. 40222
JILL A. HOLTZ 63166
WILLIAM C. HOPKINS 19335
MARK HORTON 53706
DAVID A. HOUGH 24502
MICHAEL A. HOUGHTALING 85712
D. J. HOWORTH UNITED KINGDOM
PEI HSIA 35801
PETER YAN-TEK HSU 55455
HARTMUT G. HUBER 22448
GARY HUCKABAY 73501
JACK HUGHES K7L 3N6 CANADA
PHIL HUGHES 98507
EDWARD C. HUMPHREY 78769
M. A. HUSBAND UNITED KINGDOM
DANIEL C. HYDE 17837
ELIZABETH IBARRA 91360
MICHAL IGLEWSKI 00590 POLAND
BERT INGARFIELD 77459
DAVID INTERSIMONE 90250
T. M. N. IRISH NP6 7SB UNITED KINGDOM
PORTIA ISAACSON 75080
HANNU JAAKKOLA SF-33500 FINLAND
TIMOTHY H. JACKINS 94306
STEVE JAY 85721
RON JEFFRIES 93017
JOSEF JINOCH 7 CZECHOSLOVAKIA
DOUGLAS N. JOHNSON 94598
LYTLE JOHNSON 80033
PAUL D. JOHNSON 75006
RALPH JOHNSON 61401
ROBERT T. JOHNSON 87545
S. JOHNSON 85002
JOSEPH H. JOLDA 01570
ALAN JONES IRELAND
DAVID JONES L5N 1K7 CANADA
RUSSELL JONES M5M 3W6 CANADA
MARK JUNGWIRTH 93010
MIKE KAMRAD 55413
KENNETH KAPLAN 50304
HEIKKI KASKELMA SF-00130 FINLAND
RICHARD KAUFMAN 92093
FRED J. KELLER 99210
MIKE KERSEY 77098
ADNNAN KHAN PAKISTAN
JOHN H. KILFOIL 95126
ERICH R. KNOBIL 33313
TOM KNOCHE 92109
STEPHEN KOCH 01730
JAMES R. KOCHANOWICZ 60601
JUDITH KOVETZ ISRAEL
JEFF KRAVITZ 11771
DIETRICH KREKEL D-5000 GERMANY
RICHARD W. KREUTZER 84102
CHARLES KUHLMAN 10007
DARRYL KUHNS 89503
KWAI-SAND LAM D-4440 GERMANY
K. LANG B15 2TT UNITED KINGDOM
GUY LAPALME H3C 3J7 CANADA
G. F. LAPPIN UNITED KINGDOM
JOHN LATRASH 20910
CHARLES L. LAWSON 91103
HENRI A. LE FRIANT 70118
RONALD LEBEL 02139
RICHARD LEBLANC 30332
ROLAND L. LEE 94121
R. GARY LEE 32306
WILLIAM J. LEE 55423
E. J. LEGGE M60 1QD UNITED KINGDOM
CLARENCE LEHMAN 55364
CHARLES T. LEIS 94087

WILL LELAND 53715
JOE LERNER 38163
ALAN M. LESGOLD 15260
KEN LESSEY 97051
JAMES D. LETZ 92805
DAVID J. LEWIS 14850
HORNG JYH LIANG 20854
PING K. LIAO 94545
JACK LIEBSCHUTZ 02139
JOHN E. LIND 55455
FRANK LINDSAY 22209
CHUI FAN LIU 60005
WILLIAM R. LLOYD 94115
CARLO LOCICERO L5L 2E9 CANADA
STEPHEN A. LONGO 19141
ANTHONY P. LUCIDO 77843
CLARENCE W. LYBARGER 55901
STUART LYNNE CANADA
BRUCE MAC ANASPIE 21014
D. W. MACLEAN S7N 0W0 CANADA
ORLANDO S. MADRIGAL 95929
KEVIN T. MAHONEY 01742
DENNIS J. MAINE 92717
STEPHEN MANN 01730
DUANE F. MARBLE 14260
THOMAS A. MARCINIAK 20014
L. R. MARKER 35805
BILL MARSHALL 03060
WILLIAM J. MARSHALL 98033
S. MATTHEWS H4T 1N1 CANADA
STEVE MCFERRIN 94086
FRANK E. MCGARRY 20771
JAMES PATRICK MCGEE 77025
JAMES A. MCGLINCHEY 19403
DAVID J. MCKEE 22209
IRVINE L. MCKNIGHT 94040
WILLIAM L. MCLAIN III 27410
RONALD P. MCRANEY 70360
JOHN MEDCALF 94707
MICHAEL MEEHAN 02138
JIM MERRITT 94086
HANS J. METZDORF CH-6900 SWITZERLAND
KURT METZGER 48105
JOSEPH MEYER 17557
R. N. MILLER 75229
PATRICIA R. MOHILNER 80523
ROLF MOLICH DK-2730 DENMARK
UFFE MOLLER DK-9220 DENMARK
ALLAN MOLUF 48864
CECIL A. MOORE 95051
FREEMAN L. MOORE 48859
RAYMOND MOREL CH-1224 SWITZERLAND
CLEMENT MORITZ 01720
LYALL MORRILL 94114
GREG MORRIS 54701
JWO-WU MOU CHINA
GLEN R. J. MULES 10804
ANN MURPHY 20742
CHARLES MYERS 60611
DONALD V. MYHRA 92705
J. P. M. STOFBERG 19422
GERALD NADLER 01851
OLAV NAESS NORWAY
JOHN NAGLE 95051
GEORGE NAGY 68588
T. A. NARTKER 87801
ED NAYLOR 78761
PETER A. NAYLOR 19422
HEIDI L. NEUBAUER 61801
MARTIN NICHOLS 07301
DANIEL NICHOLSON 55901
DENNIS NICHOLSON 19401
HIROAKI NISHIOKA 544 JAPAN
TONY NOE 91711
JOHN NOLD 15701
MELVIN L. NORELL 90070
BARBARA K. NORTH 66502
LARRY T. NOVAK 75235
WILLIAM I. NOWICKI 60201
M. NUNN SW1P 4RT UNITED KINGDOM
FRANK NUSSBAUM 69626
FRANK W. OECHSLI 94611
DAVID F. OHL 95014

ERIC OLSEN 92713
GENE H. OLSON 55419
JAMES B. ONEY 94022
P. E. OSMON NW3 7ST UNITED KINGDOM
SUSAN S. OWICKI 94305

JAMES N. O'BRIEN 97229
STEVE O'KEEFE 20229
JACK PAGE SINGAPORE
W. O. PAINE 91103
R. L. PALASEK 44839
RICHARD PALCHIK 95014
PAUL J. PANTANO 95051
T. L.(FRANK) PAPPAS 19083
M. L. PATRICK B20 1LL UNITED KINGDOM
DAVID N. PECK 01756
JIM PECK 16142
ROBERT C. PERLE 08753
BERNARD PERRETTE F-92803 FRANCE
RONALD H. PERROTT 94086
DAVID PESEC 44119
JAMES L. PETERSON 78712
SUSAN A. PETERSON 55432
CHRIS K. PHILLIPS 94305
DAVID M. PHILLIPS 78736
DAVID PICKENS 80302
THOMAS PIRNOT 19530
FRED POSPESCHIL 68005
DAVID POWERS 2049 AUSTRALIA
KARL PRAGERSTORFER A-4060 AUSTRIA
GENE H. PRIESTMAN 95129
FRANK PROSSER 47401
ANDREW S. PUCHRIK 47119
J. R. PUGH S1 1WB UNITED KINGDOM
SHING-KIN PUN 95926
ABBAS RAFII 73019
ROBERT J. RAKER 94104
STEVEN R. RAKITIN 06095
TIM RAND 06268
ROBERT RANSDELL 43202
DONALD D. REDDING 48105
WERNER REMMELE D-8000 GERMANY
JOHN H. REMMERS 48197
TIMOTHY P. ROBERTS 10509
KEN ROBINSON 2033 AUSTRALIA
D. J. ROBSON UNITED KINGDOM
MICHAEL RODBY 50701
JAMES D. ROGAN 48106
J. S. ROHL 6009 AUSTRALIA
JON D. ROLAND 78205
GENE ROLLINS 11794
J. ROSCOE UNITED KINGDOM
PETER N. ROTH 22030
RICHARD ROTH 06877
R. WALDO ROTH 46989
H. J. ROWE LE1 7RH UNITED KINGDOM
HERB RUBENSTEIN 80401
BEARDSLEY RUML 2ND 20008
HOWARD RUTIEZER 60025
JOHN L. RUTIS 97106
DAVID W. SALLUME 93454
ANN D. SANDERSON 23284
TOM SANDERSON 91311
HAROLD S. SCHECHTER 11418
ALAN M. SCHLENGER 95064
CH. SCHLIER D-7800 GERMANY
RICHARD SCHLOTFELDT 55435
ANA-MARIA SCHMIT CH-1007 SWITZERLAND
ED SCHOELL 95051
ROBERT L. SCHOENFELD 10021
PETER U. SCHULTHESS CH-8006 SWITZERLAND
JIM SCHULTZ 95014
WILLIAM M. SEIFERT 87545
MICHAEL SETTLE 75229
LEONARD SHAPIRO 58102
T. K. SHARPLESS 10021
TIMOTHY SHAW 20016
ALAN M. SHERKOW 53212
BRUCE SHERRY 95050
R. G. SHERRY 98107
THOMAS E. SHIELDS 77005
KEITH ALLAN SHILLINGTON 92093
CHARLES B. SHIPMAN JR. 22027
KIM L. SHIVELEY 75231
ARNOLD SHORE 22032
MICHAEL L. SIEMON 94618
LESLIE L. SIFTER 02181
JOHN SIGLE 78284
GEORGE A. R. SILVER 47374
GENE SIMMONS 22030
TIMOTHY M. SIMMONS 72143
SIMON 2308 AUSTRALIA
ANDREW SINGER 02556
ROGER SIPPL 92660
DAVID SKINNER 97330

LEO J. SLECHTA 55165
JOSEPH W. SMITH 92127
TOM SNYDER 92634
DAVID SOLOMONT 02155
RICHARD P. SPRAGUE 92714
ROB SPRAY 75207
R. E. STARCK 15021
JAMES A. STARK 94609
R. A. STASIOR 13088
MICHAEL K. STAUFFER 95051
GREG STEPHEN 62026
WIM STEVENS B-2510 BELGIUM
JIM STEWART 08854
ZHAHAI STEWART 80306
RICHARD A. STONE 55435
QUENTIN F. STOUT 13901
M. J. STRATFORD-COLLINS 06720
GORDON STUART V8P 5J2 CANADA
INDRO S. SUWANDI INDONESIA
MARK A. SWANSON 02174
ROBERT M. SWEENEY 38103
E. SWEET 48109
KEN SYLVESTRE Y1A 3P5 CANADA
DAVID TARABAR 01581
H. TAYLOR K1N 6N5 CANADA
R. F. TAYLOR 90026
DON TERWILLIGER 97005
DANIEL THALMANN H3C 3J7 CANADA
K. TIZZARD EX4 4PU UNITED KINGDOM
HOWARD E. TOMPKINS 15701
ROY TOUZEAU 59812
MIKE TRAVIS 95051
TOM A. TROTTIER M4R 1V2 CANADA
JAMES TRUEBLOOD 19713
JJIM TSEVDOS 15213
HOWARD L. TURETZKY 80220
GERALD F. UHLIG 55112
S. M. VAIDYA 411007 INDIA
TOM VAN DER HOEVEN THE NETHERLANDS
STANLEY C. VESTAL 55413
JOHN VINSEL 03061
LES VOGEL 93940
PATRIK WAHREN S-102 62 SWEDEN
SCOTT WAKEFIELD 94305
JOHN WALKER 94941
JUSTIN C. WALKER 20234
TIM WALSH 11580
P. R. WALWYN KT23 3EZ UNITED KINGDOM
C. DUDLEY WARNER 95030
SANDY WARSHAW 12301
MARK S. WATERBURY 22044
ANNA WATSON 32407
CHIP WEEMS 97331
STEPHEN J. WEINBERGER 64138
STEVEN W. WEINGART 01581
LAUREN WEINSTEIN 90025
MICHAEL D. WEINSTOCK 32548
DAVID H. WELCH 92501
JAMES H. WELLS 92634
D. R. WESTLUND K9K 1J1 CANADA
MARVIN WHITE 97077
RONALD C. WHITES 92701
MAREK WIECHULA B2G 1C0 CANADA
JACK M. WIERDA 60532
J. F. WILKES THE NETHERLANDS
ROY A. WILSKER 02114
MICHAEL WIMBLE 52404
BILL WINSPUR R3E 0W3 CANADA
JOHN WITHROW 01752
ERIC WOGSBERG 94618
BRUCE WOLFE 94965
SHARLEEN WONG 94134
STEPHEN C. WOOD 87108
ARDEN WOOTTON 64151
FULTON WRIGHT JR. 86301
BRUCE YALE 92506
M. J. L. YATES GL52 5AJ UNITED KINGDOM
JAMES CRAIG ZIEGLER 38138
DAVID J. ZOOK 60626

01570 JOSEPH H. JOLDA/ BARTLETT HIGH SCHOOL/ NEGUS STREET/ WEBSTER M A 01570
01581 DAVID TARABAR/ SOFTWARE DEVELOPMENT/ M.S. 71141/ DATA GENERAL CORPORATION/ ROUTE 9/ WESTBORO MA 01581/ (617) 366-8911 X3082
01581 STEVEN W. WEINGART/ MS 71141/ DATA GENERAL CORP./ 15 TURNPIKE RD./ WESTBORO MA 01581/ (617) 366-8911
01701 LAWRENCE F. CRAM/ 1300 WORCESTER RD. APT 101/ FRAMINGTON MA 01 701/ (617) 875-5663
01720 CLEMENT MORITZ/ 30 WETHERBEE ST./ ACTON MA 01720/ (617) 263-27 11
01730 STEPHEN KOCH/ 3M/LINOLEX SYSTEMS INC./ 10 CROSBY DRIVE/ BEDFOR D MA 01730/ (617) 275-1420 X175
01730 STEPHEN MANN/ 3M LINOLEX SYSTEMS INC./ 10 CROSBY DRIVE/ BEDFOR D MA 01730/ (617) 275-1420 X164
01742 KEVIN T. MAHONEY/ MAIL-STOP 2/ GENRAD INC./ 300 BAKER AVENUE/ CONCORD MA 01742/ (617) 369-4400 X317
01752 RICHARD M. AMEYE/ 18 ROYAL CREST DR. - APT. 4/ MARLBORO MA 017 52/ (617) 481-5823
01752 RANDALL HEAP/ MR2-3/M84/ DIGITAL EQUIPMENT CORP./ 1 IRON WAY/ MARLBORO MA 01752/ (617) 481-9511 X6848
01752 JOHN WITHROW/ MR1-1/A86/ DIGITAL EQUIPMENT CORP./ 200 FOREST S T./ MARLBORO MA 01752
01754 BRUCE HILLEGASS/ ML 22-1/B40/ DIGITAL EQUIPMENT CORP./ 146 MAI N STREET/ MAYNARD MA 01754
01756 DAVID N. PECK/ BELLINGHAM STREET/ MENDON MA 01756/ (617) 478-2 320
01778 ROBERT E. BLANTON/ ADVANCED DEVELOPMENT LABORATORY/ BOX J9/ RA YTHEON COMPANY/ BOSTON POST ROAD/ WAYLAND MA 01778/ (617) 358-2721 X2815
01821 JOEL BERSON/ MS 895A/ HONEYWELL INFORMATION SYSTEMS/ 300 CONCO RD ROAD/ BILLERICA MA 01821/ (617) 667-3111
01851 GERALD NADLER/ DEPT 46/ WANG LABS/ 1 INDUSTRIAL AVE/ LOWELL MA 01851
01854 GEORGE P. CHENEY/ ELECTRICAL ENGINEERING DEPT./ UNIVERSITY OF LOWELL/ 1 UNIVERSITY AVE./ LOWELL MA 01854/ (617) 454-7811 X259
02104 DAVID R. DICK/ SOFTWARE INNOVATIONS INC./ P.O. BOX 1537/ BOSTO N MA 02104/ (617) 734-4200
02114 ROY A. WILSKER/ COMPUTER NETWORK/ MASS. STATE COLLEGE/ 150 CAU SEWAY STREET/ BOSTON MA 02114/ (617) 727-9500
02115 JENNIFER CLARKE/ COMPUTATION CENTER/ 25 RICHARDS HALL/ NORTHEA STERN U./ 360 HUNTINGTON AVE./ BOSTON MA 02115/ (617) 437-3183
02138 MICHAEL MEEHAN/ WINTHROP PUBLISHERS/ 17 DUNSTER STREET/ CAMBRI DGE MA 02138/ (617) 868-1750
02139 RONALD LEBEL/ 344/ MIT/ 545 TECHNOLOGY SQUARE/ CAMBRIDGE MA 02 139/ (617) 253-3473
02139 JACK LIEBSCHUTZ/ 329 MEMORIAL DRIVE/ CAMBRIDGE MA 02139/ (617) 494-8335
02154 RONALD V. BOSSLET/ GTE LABS INC./ 40 SYLVAN ROAD/ WALTHAM MA 0 2154/ (617) 890-8460 X338
02154 F. G. FLETCHER/ EKS CENTER/ 200 TRAPELO ROAD/ WALTHAM MA 02154 / (617) 893-3500 X157
02154 BRANKO J. GEROVAC/ EUNICE KENNEDY SHRIVER CENTER/ 200 TRAPELO ROAD/ WALTHAM MA 02154/ (617) 893-3500 X157
02154 WILL HAPGOOD/ RAYTHEON CORP./ FOUNDRY AVE./ WALTHAM MA 02154/ (617) 899-8400 X4520
02155 DAVID SOLOMONT/ COMPUTER SERVICES/ MILLER HALL/ TUFTS UNIVERSI TY/ MEDFORD MA 02155/ (617) 628-0501
02174 DAVID GRABEL/ 154 MADISON AVE./ ARLINGTON MA 02174
02174 MARK A. SWANSON/ 71 BEACON STREET/ ARLINGTON MA 02174/ (617) 6 48-4469
02181 LESLIE L. SIFTER/ HONEYWELL/ 70 WALNUT STREET/ WELLESLEY MA 02 181
02193 TERENCE M. COLLIGAN/ RIVERSIDE OFFICE PARK/ MANAGEMENT DECISIO N SYSTEMS INC./ RIVERSIDE ROAD/ WESTON MA 02193/ (617) 891-0335
02556 ANDREW SINGER/ P.O. BOX A-67/ NORTH FALMOUTH MA 02556/ (617) 5 40-1617
03031 ERIC D. FREY/ FREY ASSOCIATES INC./ CHESTNUT HILL ROAD/ AMHERS T NH 03031/ (603) 472-5185
03060 CRAIG FRANKLIN/ FUNCTIONAL AUTOMATION/ 118 NORTHEASTERN BLVD./ NASHUA NH 03060/ (603) 882-1580
03060 BILL MARSHALL/ SANDERS ASSOCIATES INC./ 24 SIMON ST./ NASHUA N H 03060
03061 JOHN VINSEL/ NCA1-3220/ SANDERS ASSOCIATES INC./ 95 CANAL STRE ET/ NASHUA NH 03061/ (603) 885-5314
03301 ANTHONY CONTI/ P.O. BOX 1201/ CONCORD NH 03301
03824 ATTENTION: R. D. BERGERON/ DEPT. OF MATH. AND COMPUTER SCIENCE / KINGSBURY HALL/ U OF NEW HAMPSHIRE/ DURHAM NH 03824/ (603) 862-2321
04103 JOHN HEATH/ DEPT. OF MATH. AND COMPUTER SCI./ UNIV. OF MAINE/ PORTLAND ME 04103/ (207) 780-4225
06095 STEVEN R. RAKITIN/ M.S. 9488-4BB/ COMBUSTION ENGINEERING INC./ 1000 PROSPECT HILL RD./ WINDSOR CT 06095/ (203) 688-1911 X2626
06268 TIM RAND/ P.O. BOX 98/ STORRS CT 06268
06504 C. B. FALCONER/ YALE - NEW HAVEN HOSPITAL/ 6016 CB/ YALE SCHOO L OF MEDICINE/ 789 HOWARD AVE./ NEW HAVEN CT 06504/ (203) 436-2603
06720 M. J. STRATFORD-COLLINS/ DEPT.310/ 40 BRISTOL STREET/ WATERBUR Y CT 06720/ (203) 756-4451 X285
06877 RICHARD ROTH/ 5 NORTH SALEM ROAD/ RIDGEFIELD CT 06877/ (203) 4 38-3954
06880 THOMAS DANBURY/ SURVEY SAMPLING INC./ 155 E. BOSTON POST RD./ WESTPORT CT 06880/ (203) 226-1321
07102 PETER ANDERSON/ COMPUTER AND INFO SCI DEPT./ NEW JERSEY INSTIT UTE OF TECHNOLOGY/ 323 HIGH STREET/ NEWARK NJ 07102/ (201) 645-5126
07801 MARTIN NICHOLS/ 100 GUY STREET/ DOVER NJ 07801/ (201) 628-9000 X777 (WORK)/ (201) 361-7180 (HOME)
08046 LARRY E. ELLISON/ 19 HUNTINGTON LANE/ WILLINGBORO NJ 08046
08108 JOHN H. DALY/ 210 HAZEL AVE./ WESTNUT NJ 08108
08540 JAMES C. EMERY/ INTERUNIVERSITY COMMUNICATIONS COUNCIL/ EDUCOM / P.O. BOX 364/ PRINCETON NJ 08540/ (609) 921-7575
08753 ROBERT C. PERLE/ 1108 KUBY DRIVE/ TOMS RIVER NJ 08753
08854 JIM STEWART/ 195B PLEASANT VIEW ROAD/ PISCATAWAY NJ 08854/ (20 1) 382-5600 (WORK)
09175 ATTN: SAM CALVIN/ COMPUTER EDUCATION/ DARMSTADT CAREER CENTER/ APO/ NEW YORK NY 09175
10007 CHARLES KUHLMAN/ CRIMINAL JUSTICE AGENCY/ 305 BROADWAY - 5TH F LOOR/ NEW YORK NY 10007/ (212) 577-0516
10016 GENE A. DAVENPORT/ JOHN WILEY AND SONS/ 605 THIRD AVENUE/ NEW YORK NY 10016/ (212) 867-9800 X246
10021 ROBERT L. SCHOENFELD/ ROCKEFELLER UNIVERSITY/ 1230 YORK AVE./ NEW YORK NY 10021/ (212) 360-1253
10021 T. K. SHARPLESS/ INVESTIGATIVE CYTOLOGY/ MEMORIAL HOSPITAL/ 12 75 YORK AVE./ NEW YORK NY 10021/ (212) 794-6007
10509 TIMOTHY P. ROBERTS/ KERN INSTRUMENTS INC./ GENEVA RD/ BREWSTER NY 10509/ (914) 2794-5095
10583 THOMAS GIVENTER/ 1250 POST ROAD/ SCARSDALE NY 10583/ (914) 472 -4035
10804 GLEN R. J. MULES/ 263 BEECHMONT DRIVE/ NEW ROCHELLE NY 10804
11418 HAROLD S. SCHECHTER/ 84-40 117 ST/ JAMAICA NY 11418/ (212) 849 -8579
11580 TIM WALSH/ 174 E. MAUJER STREET/ VALLEY STREAM NY 11580
11771 JEFF KRAVITZ/ 69 ORCHARD ST. - APT. 3H/ OYSTER BAY NY 11771/ (516) 922-7757
11794 GENE ROLLINS/ COMPUTER SCIENCE DEPT./ SUNY - STONY BROOK/ STON Y BROOK NY 11794/ (516) 246-4383
12206 SY GREENFIELD/ MIKROS SYSTEMS CORP/ 845 CENTRAL AVE./ ALBANY N Y 12206/ (518) 489-2561
12301 SANDY WARSHAW/ 4B34/K1/ GENERAL ELECTRIC/ P.O. BOX 8/ SCHENECT ADY NY 12301/ (518) 385-8192
13088 R. A. STASIOR/ 18 FORESTER ROAD/ LIVERPOOL NY 13088/ (315) 456 -7261
13346 DEBORAH BUSCH/ COMPUTER CENTER/ COLGATE UNIV./ HAMILTON NY 133 46/ (315) 824-1000 X484
13901 QUENTIN F. STOUT/ DEPT. OF MATH SCIENCES/ SUNY - BINGHAMTON/ B INGHAMTON NY 13901/ (607) 798-2147
13902 MARY DIEGERT/ MATHEMATICS DEPT./ BROOME COMMUNITY COLLEGE/ BIN GHAMTON NY 13902/ (607) 772-5000
14063 GEORGE H. GOLDEN SR./ COMPUTER CENTER/ MAYTUM HALL/ SUNY FREDO NIA/ FREDONIA NY 14063/ (716) 673-3393
14221 GILBERT BERGLASS/ NANODATA CORP./ 2457 WEHRLE DR./ WILLIAMSVIL LE NY 14221
14260 DUANE F. MARBLE/ DEPT. OF GEOGRAPHY/ SUNY-BUFFALO/ AMHERST NY 14260/ (716) 636-2264
14850 DAVID J. LEWIS/ MATHEMATICS DEPT./ ITHACA COLLEGE/ ITHACA NY 1 4850
15021 R. E. STARCK/ OCEAN AIR INTERNATIONAL INC./ R.D. #1/ BURGETTST OWN PA 15021/ (412) 681-7533
15108 TOM GRABOWSKI/ THE CHESTER ENGINEERS/ 845 FOURTH AVENUE/ CARAO POLIS PA 15108/ (412) 262-1035
15213 JJIM TSEVDOS/ CARNEGIE-MELLON UNIV./ P.O. BOX 132/ PITTSBURGH PA 15213/ (412) 665-1036
15260 ALAN M. LESGOLD/ LRDC COMPUTER FACILITY/ UNIV. OF PITTSBURGH/ 3939 O'HARA ST./ PITTSBURGH PA 15260
15701 JOHN NOLD/ COMPUTER CENTER/ G6 STRIGHT HALL/ INDIANA UNIVERSIT Y OF PA./ INDIANA PA 15701/ (412) 357-4000
15701 HOWARD E. TOMPKINS/ COMPUTER SCIENCE DEPT/ INDIANA UNIVERSITY OF PA/ INDIANA PA 15701/ (412) 357-2524
16142 JIM PECK/ R.D. #1 - ORCHARD TERRACE/ NEW WILMINGTON PA 16142
17019 E. R. BEAUREGARD/ R.D. 3 BOX 241/ DILLSBURG PA 17019/ (717) 79 0-2095 (WORK)/ (717) 766-1446 (HOME)
17557 JOSEPH MEYER/ MS 103/ SPERRY - NEW HOLLAND/ 500 DILLER AVE./ N EW HOLLAND PA 17557/ (717) 354-1798
17837 DANIEL C. HYDE/ COMPUTER SCIENCE PROGRAM/ BUCKNELL UNIVERSITY/ LEWISBURG PA 17837/ (717) 524-3743
18042 ATTN: LAFAYETTE COLLEGE/ EASTON PA 18042
19083 T. L.(FRANK) PAPPAS/ 338 FRANCIS DRIVE/ HAVERTOWN PA 19083/ (2 15) 259-1325
19117 FRED M. HOFKIN/ 8212 ASPEN WAY/ ELKINS PARK PA 19117/ (215) 88 6-4780
19141 STEPHEN A. LONGO/ PHYSICS DEPT./ LA SALLE COLLEGE/ PHILADELPHI A PA 19141/ (215) 951-1255
19174 RICHARD D. HACKATHORN/ WHARTON SCHOOL - DEPT. OF DECISION SCI/ UNIV. OF PENNSYLVANIA/ PHILADELPHIA PA 19174/ (215) 243-5149
19311 ARTIE GREEN/ HEWLETT PACKARD/ ROUTE 41/ AVONDALE PA 19311/ (21 5) 268-2281
19335 WILLIAM C. HOPKINS/ 1101 BONDSVILLE ROAD/ DOWNINGTOWN PA 19335 / (215) 269-4486
19401 DENNIS NICHOLSON/ 421 ALEXANDRA DR./ NORRISTOWN PA 19401/ (215) 539-8243
19403 JAMES A. MCGLINCHEY/ 332 WEYMOUTH RD. / PLYMOUTH TOWNSHIP/ NOR RISTOWN PA 19403
19422 J. P. M. STOFBERG/ MS B/220M/ SPERRY UNIVAC/ P.O. BOX 500/ BLU E BELL PA 19422/ (215) 542-4011
19422 PETER A. NAYLOR/ MS B/220M/ SPERRY UNIVAC/ P.O. BOX 500/ BLUE BELL PA 19422/ (215) 542-4011
19424 WILLIAM L. HAFNER/ MAIL STOP 1B-220M/ SPERRY UNIVAC/ P.O. BOX 500/ BLUE BELL PA 19424/ (215) 542-4646
19530 THOMAS PIRNOT/ MATH DEPT./ KURZTOWN STATE COLLEGE/ KURZTOWN PA 19530
19713 JAMES TRUEBLOOD/ PLATO PROJECT/ UNIV. OF DELAWARE/ 46 E. DELAW ARE AVE./ NEWARK DE 19713
19898 C. J. COTTON/ E. I. DUPONT DE NEMOURS CO./ 101 BEECH ST./ WILM INGTON DE 19898/ (302) 774-1372
20007 MARTIN BUCHANAN/ SYSTEMS CONSULTANTS INC/ 1054 31ST STREET NW/ WASHINGTON DC 20007/ (202) 342-4000
20008 BEARDSLEY RUML 2ND/ 3045 ORDWAY STREET NW/ WASHINGTON DC 20008 / (202) 244-3534
20014 THOMAS A. MARCINIAK/ 8315 N. BROOK LN. #903/ BETHESDA MD 20014 / (301) 654-7970
20016 TIMOTHY SHAW/ EPA PROJECT OFFICE/ COMNET/ 5185 MACARTHUR BLVD. / WASHINGTON DC 20016/ (202) 244-1900
20052 PETER BOCK/ DEPT OF ELEC ENGR & COMP SCI/ GEORGE WASHINGTON UN IV./ 725 23RD ST NW/ WASHINGTON DC 20052/ (202) 676-6083
20229 STEVE O'KEEFE/ 7424/ U.S. CUSTOMS DATA CENTER/ 1301 CONSTITUTI ON AVE. N.W./ WASHINGTON DC 20229/ (202) 566-5447
20234 JUSTIN C. WALKER/ SYSTEMS & SOFTWARE DIV./ A-264 BLDG. 225/ NA TIONAL BUREAU OF STANDARDS/ WASHINGTON DC 20234/ (301) 921-3491
20742 ANN MURPHY/ COMPUTER SCIENCE CENTER/ 2337 PROGRAM LIBRARY/ U O F MARYLAND/ COLLEGE PARK MD 20742/ (301) 454-4262
20760 F. T. BAKER/ IBM FEDERAL SYSTEMS DIV./ 18100 FREDERICK PIKE/ G AITHERSBURG MD 20760/ (301) 840-0111
20771 FRANK E. MCGARRY/ GODDARD SPACE FLIGHT CENTER/ NASA/ CODE 582. 1/ GREENBELT MD 20771/ (301) 982-5048
20854 ATTN: COMPUTER SCIENCE PRESS INC./ 9125 FALL RIVER ROAD/ POTOM AC MD 20854/ (301) 299-2040
20854 HORNG JYH LIANG/ CONTROL DATA CORP./ 1151 SEVEN-LOCKS ROAD/ RO CKVILLE MD 20854/ (301) 340-3883
20901 LEONARD BINSTOCK/ 820 LOXFORD TERRACE/ SILVER SPRING MD 20901/ (301) 593-3218/ (301) 496-3204
20910 JOHN LATRASH/ 1001 CPRING ST. - # 1007/ SILVER SPRING MD 20910 / (301) 588-7894
21014 BRUCE MAC ANASPIE/ 600 N. HICKORY AVE. - APT. 18/ BEL AIR MD 2 1014
21114 ROBERT W. HARRIS/ COMMUNICATIONS AND SIGNAL PROCESSING/ MAR AS SOCIATES INC./ 1702 FALLSWAY DRIVE/ CROFTON MD 21114/ (301) 261-0780
22027 CHARLES B. SHIPMAN JR./ 2205 SANDBURG ST./ DUNN LORING VA 2202 7
22030 PETER N. ROTH/ 13146 MALTESE LANE/ FAIRFAX VA 22030
22030 GENE SIMMONS/ 9514 FARMVIEW CT/ FAIRFAX VA 22030
22032 ARNOLD SHORE/ 4607 BRIAR PATCH CT./ FAIRFAX VA 22032
22044 MARK S. WATERBURY/ 2961 PATRICK HENRY DRIVE #201/ FALLS CHURCH VA 22044/ (703) 532-8119
22101 W. ASHBY BOAZ/ BLDG.BXA/2/ TRW INC./ 7600 COLSHIRE DR./ MCLEAN VA 22101/ (703) 893-2000
22209 LARRY DUBY/ 1724 N. QUINN ST. APT 305/ ARLINGTON VA 22209
22209 FRANK LINDSAY/ BUNKER RAMO CORP/ 1500 WILSON BLVD. SUITE 400/ ARLINGTON VA 22209
22209 DAVID J. MCKEE/ ADVANCED COMPUTER TECHNIQUES/ 1501 WILSON BLVD ./ ARLINGTON VA 22209/ (703) 524-8330
22301 PHILIP GAUDETTE/ SOFTWARE RESOURCES/ P.O. BOX 2015/ ALEXANDRIA VA 22301/ (703) 548-2866
22448 HARTMUT G. HUBER/ P.O. BOX 117/ DAHLGREN VA 22448/ (703) 663-8 656
22901 LINWOOD FERGUSON/ 2605C HYDROLIC RD./ CHARLOTTESVILL VA 22901/ (804) 293-7816
22903 R. LEONARD BROWN/ DAMACS/ THORNTON HALL/ UNIV. OF VIRGINIA/ CH ARLOTTESVILL VA 22903/ (804) 924-7201
23185 DOUGLAS DUNLOP/ 1502 CONWAY DRIVE - APT. 103/ WILLIAMSBURG VA 23185

23284 ANN D. SANDERSON/ OFFICE OF COMPTNG ACTIVITIES/ VIRGINIA COMM ONWEALTH UNIVERSITY/ 1015 FLOYD AVE./ RICHMOND VA 23284/ (804) 770-6339
23601 RICKY W. BUTLER/ 132 WOODS ROAD/ NEWPORT NEWS VA 23601/ (804) 596-7272/ (804) 827-2929
24502 DAVID A. HOUGH/ 223 BRYANT RD./ LYNCHBURG VA 24502/ (804) 237- 5508 (HOME)/ (804) 384-5111 X2280 (WORK)
27410 WILLIAM L. MCLAIN III/ COMP SVCS/ GUILFORD COLLEGE/ 5800 W. FR ENDLY AVE./ GREENSBORO NC 27410/ (919) 292-5511 X139
27514 STEVEN M. BELLOVIN/ DEPT. OF COMP. SCI./ U OF NORTH CAROLINA/ CHAPEL HILL NC 27514/ (919) 933-7240
29208 ROBERT L. CANNON/ DEPT. OF MATH. AND COMP. SCIENCE/ UNIV. OF S O. CAROLINA/ COLUMBIA SC 29208
30332 ATTENTION: JERRY W. SEGERS/ OFFICE OF COMPUTING SERVICES/ GEOR GIA INSTITUTE OF TECHNOLOGY/ ATLANTA GA 30332/ (404) 894-4676
30332 RICHARD LEBLANC/ SCHOOL OF INFO. AND COMP. SCI./ GEORGIA TECH/ ATLANTA GA 30332/ (404) 894-2592
32306 R. GARY LEE/ DEPT. OF MATH./ 113 LOVE BUILDING/ FLORIDA STATE U/ TALLAHASSEE FL 32306/ (904) 644-4419
32407 ANNA WATSON/ 3705 DELWOOD DRIVE/ PANAMA CITY FL 32407/ (904) 2 34-2507
32548 MICHAEL D. WEINSTOCK/ 112 KNOLLWOOD WAY/ FT. WALTON FL 32548
32601 SCOTT CRUMPTON/ 11130 N.W. 9 PLACE/ GAINESVILLE FL 32601/ (904) 375-7952
32901 T. M. CAMBRON/ TR3/1/ HARRIS ESD/ P.O. BOX 37/ MELBOURNE FL 32 901/ (305) 727-4629
33313 ERICH R. KNOBIL/ PRODUCT DEVELOPMENT/ SYSTEMS ENGINEERING LABS ./ 6901 WEST SUNRISE BLVD./ FT. LAUDERDALE FL 33313/ (305) 587-2900
35223 JEFFREY W. GRAHAM/ GRAHAM COMPUTER ENTERPRISES INC./ 3 OFFICE PARK CIR. - SUITE 106/ BIRMINGHAM AL 35223/ (205) 870-7267
35801 PEI HSIA/ 3300 O'HARA DR./ HUNTSVILLE AL 35801/ (205) 895-6088
35805 L. R. MARKER/ DEFENSE AND SPACE SYSTEMS/ TRW/ 7702 GOVERNORS D R. WEST/ HUNTSVILLE AL 35805/ (205) 837-3950
37916 ATTENTION: GORDON R. SHERMAN/ COMPUTER CENTER/ 200 STOKELY MGM T. CENTER/ U OF TENNESSEE/ KNOXVILLE TN 37916/ (615) 974-6721
38103 ROBERT M. SWEENEY/ ELLERS FANNING OAKLEY CHESTER & RIKE/ 722 F ALLS BUILDING/ MEMPHIS TN 38103/ (901) 526-7321
38138 JAMES CRAIG ZIEGLER/ 1455 LANCASTER DR./ MEMPHIS TN 38138
38163 JOE LERNER/ DEPT. OF PATHOLOGY/ UNIV. OF TENNESSEE/ 858 MADISO N AVE./ MEMPHIS TN 38163/ (901) 528-6320
38677 ATTN: DEPT. OF COMPUTER SCIENCE/ U OF MISSISSIPPI/ UNIVERSITY MS 38677
40222 JAMES W. HOLT JR./ 7705 NORWOOD DR./ LOUISVILLE KY 40222/ (502) 425-6729
43023 JIM CAMERON/ DEPT. OF MATH SCIENCE/ DENISON UNIVERSITY/ GRANVI LLE OH 43023/ (614) 587-0810 X582
43202 ROBERT RANSDELL/ 619 HARLEY DRIVE APT. 5/ COLUMBUS OH 43202
43214 RICHARD E. ADAMS/ 239 CHATHAM ROAD/ COLUMBUS OH 43214/ (614) 2 67-8068
44119 DAVID PESEC/ 21030 MILLER AVENUE/ EUCLID OH 44119/ (216) 486-8 070
44839 R. L. PALASEK/ DEPT. OF APPLIED SCIENCES/ BOWLING GREEN STATE UNIV./ 901 RYE BEACH ROAD/ HURON OH 44839
45036 ATTN: BETTE BOLLING-LIBRARIAN/ TECHNICAL INFORMATION CTR-ELECT RONICS/ CINCINNATI MILACRON INC./ LEBANON OH 45036/ (513) 494-5320
45241 FRANCIS H. BEARDEN/ DATA SYSTEMS/ CINCINNATI ELECTRONICS CORP. / 2630 GLENDALE-MILFORD ROAD/ CINCINNATI OH 45241/ (513) 563-6000
46989 R. WALDO ROTH/ COMPUTER SCIENCE DEPT/ TAYLOR UNIVERSITY/ UPLAN D IN 46989/ (317) 998-2751 X269
47119 ANDREW S. PUCHRIK/ 111Q ANDREA DR./ FLOYDS KNOBS IN 47119/ (50 2) 582-4397
47374 GEORGE A. R. SILVER/ DEPT. OF HISTORY/ EARLHAM COLLEGE/ RICHMO ND IN 47374/ (317) 962-6561
47401 ANN BARDIN/ WRUBEL COMPUTING CENTER/ HPER BUILDING/ INDIANA UN IV./ BLOOMINGTON IN 47401/ (812) 337-1911
47401 FRANK PROSSER/ COMPUTER SCIENCE DEPT./ 101 LINDLEY HALL/ INDIA NA UNIVERSITY/ BLOOMINGTON IN 47401
47904 HOWARD CUNNINGHAM/ 914 NORTH 21ST STREET/ LAFAYETTE IN 47904/ (317) 447-6403
48010 SHAUN DEVLIN/ 6854 CEDARBROOK/ BIRMINGHAM MI 48010
48105 KURT METZGER/ 478 CLOVERDALE/ ANN ARBOR MI 48105
48105 DONALD D. REDDING/ MDSI/ 4251 PLYMOUTH ROAD/ ANN ARBOR MI 4810 5/ (313) 995-6118
48106 JAMES D. ROGAN/ COMSHARE INC./ P.O. BOX 1588/ ANN ARBOR MI 481 06/ (313) 994-4800
48109 E. SWEET/ COMPUTING CENTER/ UNIV. OF MICHIGAN-NORTH CAMPUS/ 10 75 BEAL AVE./ ANN ARBOR MI 48109
48197 JOHN H. REMMERS/ DEPT. OF MATHEMATICS/ EASTERN MICHIGAN UNIV./ YPSILANTI MI 48197/ (313) 487-1290
48859 FREEMAN L. MOORE/ DEPT. OF COMPUTER SCIENCE/ 203-B PIERCE HALL / CENTRAL MICHIGAN UNIV./ MT. PLEASANT MI 48859/ (517) 774-3922
48864 ALLAN MOLUF/ 2317 KNOB HILL APT. 9/ OKEMOS MI 48864
50304 KENNETH KAPLAN/ MICROWARE SYSTEMS CORP/ P.O. BOX 954/ DES MOIN ES IA 50304/ (515) 265-6121
50701 MICHAEL RODBY/ 322 DEVONSHIRE/ WATERLOO IA 50701/ (319) 233-57 96
52242 ATTN: SERIALS DEPT./ UNIVERSITY LIBRARIES/ UNIVERSITY OF IOWA/ IOWA CITY IA 52242
52404 MICHAEL WIMBLE/ 6026 UNDERWOOD AVE. S.W./ CEDAR RAPIDS IA 5240 4/ (319) 396-5641
53201 DAVID E. ANDERSON/ JOHNSON CONTROLS INC./ 507 E. MICHIGAN AVE. / MILWAUKEE WI 53201/ (414) 276-9200
53212 ALAN M. SHERKOW/ 1211 EAST SINGER CIR. APT. #4/ MILWAUKEE WI 5 3212/ (414) 332-9533
53706 ATTN: FRIEDA S. COHN/ ACADEMIC COMPUTING CENTER/ U OF WISCONSI N/ 1210 W. DAYTON ST./ MADISON WI 53706/ (608) 262-2055
53706 MARK HORTON/ COMPUTER SCIENCE DEPT./ UNIV. OF WISCONSIN/ 1210 W. DAYTON ST./ MADISON WI 53706/ (608) 262-1079/ (608) 238-1866
53715 WILL LELAND/ 445 N. LAKE ST./ MADISON WI 53715/ (608) 257-4035
54701 GREG MORRIS/ 1705 WILSON ST./ EAU CLAIRE WI 54701/ (715) 835-6 324
55112 GERALD F. UHLIG/ 3545 OWASSO ST. APT. 110/ SHOREVIEW MN 55112/ (612) 483-9714
55113 ATTN: INFORMATION SERVICE CENTER/ SPERRY-UNIVAC/ 2276 HIGH CRE ST DRIVE/ ROSEVILLE MN 55113
55113 DAVID GARDNER/ 1730 LARPENTEUR - APT. 3C/ ST. PAUL MN 55113/ (612) 646-5479
55119 GARY BECKWITH/ 2199 GLENRIDGE AVE./ ST. PAUL MN 55119/ (612) 8 53-5235
55165 LEO J. SLECHTA/ DSD/ SPERRY UNIVAC/ BOX 3525 MS U1025/ ST. PAU L MN 55165/ (612) 456-2743
55343 CLIFFORD GERSTENHABER/ DEFENSE SYSTEMS/ MN11-2120/ HONEYWELL/ 600 SECOND ST. N.E./ HOPKINS MN 55343/ (612) 542-4940
55364 CLARENCE LEIMAN/ 5926 GUMWOOD ROAD/ MOUND MN 55364/ (612) 472- 1405
55408 ABDUL RASAQ BELLO/ P.O. BOX 8681/ MINNEAPOLIS MN 55408/ (612) 330-4106
55413 MIKE KAMRAD/ HONEYWELL AVIONICS/ 2600 RIDGWAY PKWY/ MINNEAPOLI S MN 55413/ (612) 378-5328
55413 STANLEY C. VESTAL/ MS 2340/ HONEYWELL INC./ 2600 RIDGWAY PKWY. / MINNEAPOLIS MN 55413/ (612) 378-5046
55414 THOM HOARD/ P.O. BOX 14413/ MINNEAPOLIS MN 55414/ (612) 376-62 90
55419 GENE H. OLSON/ 5149 ALDRICH AVE. S./ MINNEAPOLIS MN 55419/ (61 2) 824-9108
55423 WILLIAM J. LEE/ 7320 FIRST AVE. SO./ RICHFIELD MN 55423/ (612) 866-6284
55432 SUSAN A. PETERSON/ SOFTWARE DEVELOPMENT/ M.S. E-360/ DATA 100 CORP./ 7725 WASHINGTON AVE S./ MINNEAPOLIS MN 55432/ (612) 941-6500 X437
55435 RICHARD SCHLOTFELDT/ 7330 GALLAGHER DR. - APT. 255/ EDINA MN 5 5435/ (612) 831-6403
55435 RICHARD A. STONE/ DATA 100 CORP./ 7725 WASHINGTON AVE. SO./ MI NNEAPOLIS MN 55435/ (612) 941-6500 X479
55455 CHRISTOPHER AMLEY/ SSRFC/ 25 BLEGEN HALL/ UNIV. OF MINNESOTA/ WEST BANK/ MINNEAPOLIS MN 55455/ (612) 373-9917
55455 ATTN: SPECIAL INTERACTIVE COMPUTER LAB/ 143 SPACE SCIENCE CENT ER/ UNIV. OF MINNESOTA/ EAST BANK/ MINNEAPOLIS MN 55455/ (612) 373-5768
55455 SCOTT S. BERTILSON/ UNIVERSITY COMPUTER CENTER/ 227 EXP. ENGR. / UNIV. OF MINNESOTA/ MINNEAPOLIS MN 55455/ (612) 376-5262 (WORK)/ (612) 331-2464 (HOME)
55455 PETER YAN-TEK HSU/ COMP. SCI. DEPT./ 114 LIND HALL/ U OF MINNE SOTA/ EAST BANK/ MINNEAPOLIS MN 55455/ (612) 376-4590
55455 JOHN E. LIND/ 363 TERRITORIAL HALL/ UNIVERSITY OF MINNESOTA/ E AST BANK/ MINNEAPOLIS MN 55455/ (612) 373-6545
55901 DENNIS S. DELORME/ 1407 13TH AVENUE NE/ ROCHESTER MN 55901/ (5 07) 289-8495
55901 CLARENCE W. LYBARGER/ 3513 VILLA ROAD N.W./ ROCHESTER MN 55901 / (507) 289-0053
55901 DANIEL NICHOLSON/ 2112 17TH AVE NW/ ROCHESTER MN 55901/ (507) 289-1780 (HOME)/ (507) 286-5921 (WORK)
58102 LEONARD SHAPIRO/ MATH DEPARTMENT/ NORTH DAKOTA STATE UNIV./ FA RGO ND 58102/ (701) 237-8171
59812 ROY TOUZEAU/ COMPUTER SCIENCE DEPT./ UNIV. OF MONTANA/ MISSOUL A MT 59812/ (406) 243-2883 (WORK)/ (406) 549-8064 (HOME)
60005 CHUI FAN LIU/ 1603 S. HIGHLAND AVE./ ARLINGTON HTS IL 60005
60025 HOWARD RUTIEZEK/ 1101 ARBOR/ GLENVIEW IL 60025/ (312) 234-3400 (WORK)/ (312) 724-7731 (HOME)
60201 WILLIAM I. NOWICKI/ CSRL TECH/ B626/ NORTHWESTERN UNIV./ 2145 SHERIDAN RD/ EVANSTON IL 60201/ (312) 492-5248
60439 RICHARD D. GEORGE/ ARGONNE NATIONAL LABORATORY/ 9700 S. CASS A VENUE/ ARGONNE IL 60439
60532 JACK M. WIERDA/ DEPT. 7346/ WESTERN ELECTRIC COMPANY/ 4513 WES TERN AVE./ LISLE IL 60532/ (312) 983-3439
60601 JAMES R. KOCHANOWICZ/ DEDICATED SYSTEMS INC./ 180 N. MICHIGAN AVE./ CHICAGO IL 60601/ (312) 372-4222
60611 CHARLES MYERS/ DATALOCICS/ 325 W. HURON/ CHICAGO IL 60611/ (31 2) 266-4444
60626 DAVID J. ZOOK/ 1100 W. PRATT/ CHICAGO IL 60626
60919 MICHAEL FAY/ STELLE STATION/ CABERY IL 60919
61401 RALPH JOHNSON/ 1592 N. BROAD/ GALESBURG IL 61401/ (309) 342-30 98
61801 DONALD D. BURN/ 410 WEST ILLINOIS/ URBANA IL 61801/ (217) 367- 5362
61801 HEIDI L. NEUBAUER/ COORDINATED SCIENCES LAB/ UNIV. OF ILLINOIS - URBANA/ URBANA IL 61801/ (217) 333-4796 WORK
62026 GREG STEPHEN/ DEPT. OF MATH. STUDIES/ S. ILLINOIS UNIV. AT EDW ARDSVILLE/ EDWARDSVILLE IL 62026
62704 MIKE HARRIS/ APT. 4/ 309 WEST EDWARDS/ SPRINGFIELD IL 62704/ (217) 789-7669 (HOME)/ (217) 782-0014 (WORK)
63110 WILLIAM HOLMES/ DEPT. OF BIOCHEMISTRY/ WASHINGTON UNIV. MED. S CHOOL/ 660 S. EUCLID AVE./ ST. LOUIS MO 63110/ (314) 454-3622
63166 JILL A. HOLTZ/ PROGRAMMING SCIENCES DEPT./ BLDG. 105 LEVEL 4/E -6/ MCDONNELL DOUGLAS CORP./ P.O. BOX 516/ ST. LOUIS MO 63166
64138 STEPHEN J. WEINBERGER/ 8340 B. HILLCREST RD./ KANSAS CITY MO 6 4138/ (816) 333-9311
64151 ARDEN WOOTTON/ 5904 N.W. LENOX AVE./ KANSAS CITY MO 64151/ (81 6) 741-5822
66502 BARBARA K. NORTH/ 1516 NICHOLS ST./ MANHATTAN KS 66502/ (913) 532-6350 WORK/ (913) 537-7818 HOME
66762 FELIX DREHER/ COMPUTER SCIENCE/ PITTSBURG STATE UNIVERSITY/ PI TTSBURG KS 66762/ (316) 232-7000 X425
68005 FRED POSPESCHIL/ 3108 JACKSON ST./ BELLEVUE NE 68005/ (402) 29 1-0795
68134 EARL N. CHRISTIANSEN/ BASCO COMPANY/ 9685 AMES/ OMAHA NE 68134 / (402) 572-8911
68154 CHARLES J. GIBBONS/ SUITE 8/ APPLIED COMMUNICATIONS INC./ 1084 4 OLD MILL ROAD/ OMAHA NE 68154/ (402) 330-3732
68508 K. S. BHASKAR/ ACADEMIC COMPUTER SERVICES/ 225 NEBRASKA HALL/ UNIV. OF NEBRASKA - LINCOLN/ LINCOLN NE 68508/ (402) 472-3701
68583 D. F. COSTELLO/ ACADEMIC COMPUTING SERVICES/ UNIVERSITY OF NEB RASKA-LINCOLN/ 3835 HOLDREGE/ LINCOLN NE 68583
68588 GEORGE NAGY/ DEPT. OF COMP. SCI./ 110 FERGUSON HALL/ U OF NEBR ASKA/ LINCOLN NE 68588/ (402) 472-3200/ (402) 472-2402
69626 FRANK NUSSBAUM/ DEPT. OF MATH. SCIENCES/ LOYOLA UNIV./ 6525 N. SHERIDAN/ CHICAGO IL 69626
70118 HENRI A. LE FRIANT/ 8200 APRICOT ST./ NEW ORLEANS LA 70118/ (5 04) 865-1023
70360 RONALD P. MCRANEY/ STATION 1/ P.O. BOX 10097/ HOUMA LA 70360/ (504) 868-0453
72143 TIMOTHY M. SIMMONS/ HARDING COLLEGE/ BOX 1193 STATION A/ SEARC Y AR 72143/ (501) 268-6753
73019 ABBAS RAFII/ 219 EECS DEPT./ UNIV. OF OKLAHOMA/ 202 W. BOYD/ N ORMAN OK 73019/ (405) 325-4721
73102 REFORD BOND/ ESSAY CORP./ 100 PARK AVE./ OKLAHOMA CITY OK 7310 2
73501 GARY HUCKABAY/ DEPT. OF MATHEMATICS/ CAMERON UNIV./ LAWTON OK 73501
74104 C. BAILEY/ BAILEY AND ASSOCIATES/ 1144 S. ATLANTA/ TULSA OK 74 104/ (918) 936-0596
74171 ANTHONY BJERSTEDT/ ORAL ROBERTS UNIVERSITY/ BOX 1127/ TULSA OK 74171/ (918) 492-2011
75006 PAUL D. JOHNSON/ 2009 WEDGEWOOD/ CARROLLTON TX 75006/ (214) 24 5-7904
75080 PORTIA ISAACSON/ THE MICRO STORE/ 634 S. CENTRAL EXPY./ RICHAR DSON TX 75080/ (214) 231-1096
75207 ROB SPRAY/ MS 420-160/ ROCKWELL INTERNATIONAL/ P.O. BOX 10462/ DALLAS TX 75207/ (214) 996-2255
75229 R. N. MILLER/ INTERNATIONAL COMPUTER PRODUCTS/ 2925 MERRELL RD ./ DALLAS TX 75229/ (214) 350-6951
75229 MICHAEL SETTLE/ INTERNATIONAL COMPUTER PRODUCTS/ 2925 MERRELL RD./ DALLAS TX 75229
75231 KIM L. SHIVELEY/ 7777 MANDERVILLE LANE APT. 221/ DALLAS TX 752 31
75235 LARRY T. NOVAK/ MS 348/ TEXAS INSTRUMENTS/ P.O. BOX 35486/ DAL LAS TX 75235/ (214) 238-6904
75248 JOE COINTMENT/ 7709 QUEENS GARDEN DR./ DALLAS TX 75248/ (214) 387-0468
76010 JIM CLARKE/ P.O. BOX 517/ ARLINGTON TX 76010/ (817) 461-5429
77005 THOMAS E. SHIELDS/ SOFTWARE RESOURCES/ P.O. BOX 25210/ HOUSTON TX 77005/ (713) 521-0366
77025 JAMES PATRICK MCGEE/ P.O. BOX 20223/ HOUSTON TX 77025/ (713) 6 68-6232
77098 MIKE KERSEY/ 1850 W. MAIN #1/ HOUSTON TX 77098/ (713) 526-8658
77459 BERT INGARFIELD/ ACTION COMPUTER SERVICES/ 2202 MUSTANG SPRING S DRIVE/ MISSOURI CITY TX 77459/ (713) 437-5197
77840 JERRY R. BROOKSHIRE/ 1201 HIGHWAY 30 - APT. 153/ COLLEGE STA. TX 77840/ (713) 693-2938
77843 ANTHONY P. LUCIDO/ COMPUTING SCIENCES DEPT./ TEXAS A&M UNIV./ COLLEGE STA. TX 77843/ (713) 845-5531
78205 JON D. ROLAND/ MICRO MART/ 1015 NAVARRO/ SAN ANTONIO TX 78205/ (512) 222-1427
78284 JOHN SIGLE/ TRINITY UNIV./ P.O. BOX 237/ SAN ANTONIO TX 78284/ (512) 736-7236

78712 JAMES L. PETERSON/ DEPT OF COMPUTER SCIENCES/ PAINTER HALL/ UNIV. OF TEXAS - AUSTIN/ AUSTIN TX 78712/ (512) 471-4353
78736 DAVID M. PHILLIPS/ 6922 THOMAS SPRINGS RD./ AUSTIN TX 78736/ (512) 471-7202
78761 ED NAYLOR/ P.O. BOX 15103/ AUSTIN TX 78761/ (512) 451-0342
78769 DAVID N. GRAY/ MS 2188/ TEXAS INSTRUMENTS/ P.O. BOX 2909/ AUSTIN TX 78769/ (512) 258-7406
78769 EDWARD C. HUMPHREY/ M.S. 2201/ TEXAS INSTRUMENTS INC./ P.O. BOX 2909/ AUSTIN TX 78769/ (512) 258-7289
80033 LYTLE JOHNSON/ 8951 W. 46TH PLACE/ WHEAT RIDGE CO 80033
80204 GERHARDT C. CLEMENTSON/ DEPT. OF COMP. AND MGMT SCIENCE/ METROPOLITAN STATE COLLEGE/ 1006 11TH STREET BOX 13/ DENVER CO 80204/ (303) 629-3122
80220 HOWARD L. TURETZKY/ HYDRA/ 1575 IVANHOE ST./ DENVER CO 80220/ (303) 333-2892
80302 DAVID PICKENS/ DEPT. 50J/ BLDG. 023/ IBM CORP./ P.O. BOX 1900/ BOULDER CO 80302/ (303) 447-5844
80306 ZHAHAI STEWART/ P.O. BOX 1637/ BOULDER CO 80306/ (303) 443-7279
80401 H. W. EGDORF/ P.O. BOX 226/ GOLDEN CO 80401/ (303) 234-3994 (WORK)
80401 HERB RUBENSTEIN/ 1036 6TH STREET/ GOLDEN CO 80401/ (303) 278-3469/ (303) 458-5900 X202 (WORK)
80523 PATRICIA R. MOHILNER/ DEPT. OF COMPUTER SCIENCE/ COLORADO STATE UNIV./ FORT COLLINS CO 80523/ (303) 491-7137
80901 GREG BOURQUE/ HEWLETT PACKARD/ P.O. BOX 2197/ COLORADO SPRIN CO 80901/ (303) 598-1900
83704 ED HIRAHARA/ 11207 MUSKET ST./ BOISE ID 83704/ (208) 376-6000 X2574/ OR X3989
84047 JOHN A. BRIGGS/ 837 EAST 6775 SOUTH/ MIDVALE UT 84047/ (801) 292-8000
84102 RICHARD W. KREUTZER/ 644 ELIZABETH ST./ SALT LAKE CITY UT 84102/ (801) 583-5202/ (801) 486-3351
84121 W. F. HAYGOOD/ COMPUTER SERVICES CO./ 7822 OAKLEDGE ROAD/ SALT LAKE CITY UT 84121
85002 S. JOHNSON/ ECS/ MARIOPA CO. COMM. COLLEGE/ P.O. BOX 13349/ PHOENIX AZ 85002
85005 ALLEN BERGLUND/ MS K-28/ P.O. BOX 6000/ PHOENIX AZ 85005/ (602) 249-7466
85712 MICHAEL A. HOUGHTALING/ UNIV. OF ARIZONA/ 3401 N. COLUMBUS APT. 17-B/ TUCSON AZ 85712/ (602) 884-4239
85721 STEVE JAY/ COMPUTER CENTER/ UNIV. OF ARIZONA/ TUCSON AZ 85721/ (602) 884-2239
86301 FULTON WRIGHT JR./ COMPUTER SERVICES/ YAVAPAI COLLEGE/ PRESCOTT AZ 86301/ (602) 778-1990
87106 REESA ABRAMS/ FALCON RESEARCH AND DEVELOPMENT/ 2350 ALAMO S.E. - #200/ ALBUQUERQUE NM 87106/ (505) 843-6101
87108 STEPHEN C. WOOD/ MICROSOFT/ 819 TWO PARK CENTRAL TOWER/ ALBUQUERQUE NM 87108/ (505) 256-3600
87545 ROBERT T. JOHNSON/ C-11 MAIL STOP 296/ LOS ALAMOS SCIENTIFIC LABORATORY/ P.O. BOX 1663/ LOS ALAMOS NM 87545/ (505) 667-5014
87545 WILLIAM M. SEIFERT/ MS 532/ LASL/ P.O. BOX 1663/ LOS ALAMOS NM 87545
87801 T. A. NARTKER/ DEPT. OF COMPUTER SCIENCE/ NEW MEXICO TECH/ SOCORRO NM 87801/ (505) 835-5126
88003 JOSEPH EINWECK/ P.O. BOX 3824/ LAS CRUCES NM 88003/ (505) 523-5377
89154 ATTN: PROGRAMMING ADVISOR/ SOUTHERN NEVADA COMPUTING FACILITY/ UNIV. OF NEVADA - LAS VEGAS/ 4505 MARYLAND PARKWAY/ LAS VEGAS NV 89154/ (702) 739-3557
89503 DARRYL KUHNS/ 1590 HILLSIDE DR./ RENO NV 89503
89507 ATTN: PROGRAMMING ADVISOR/ UNS COMPUTING CENTER/ 22 WR/ U OF NEVADA/ BOX 9068/ RENO NV 89507/ (702) 784-4008
89557 E. MAURICE BEESLEY/ DEPT. OF MATHEMATICS/ 227 SEM/ UNIV. OF NEVADA-RENO/ RENO NV 89557/ (702) 784-6773
90025 LAUREN WEINSTEIN/ 12320 TEXAS AVE. #12/ LOS ANGELES CA 90025/ (213) 826-5766
90026 R. F. TAYLOR/ 1509 SARGENT PLACE/ LOS ANGELES CA 90026/ (213) 488-0288
90070 MELVIN L. NORELL/ PROGRAMMA CONSULTANTS/ P.O. BOX 70127/ LOS ANGELES CA 90070/ (213) 243-0810
90230 LEE D. AURICH/ 5650 SUMNER WAY # 116/ CULVER CITY CA 90230/ (213) 649-4404
90250 DAVID INTERSIMONE/ SOFTWARE ENGINEERING SECTION/ TRW - CS&S/ 12911 SIMMS AVE./ HAWTHORNE CA 90250/ (213) 536-4286
90706 CARSON GERMAN/ 9615 WALNUT ST./ BELLFLOWER CA 90706/ (714) 871-3232 X2068
91103 CHARLES L. LAWSON/ JET PROPULSION LABORATORY/ MS 125/128/ CALIFORNIA INSTITUTE OF TECHNOLOGY/ 4800 OAK GROVE DR./ PASADENA CA 91103/ (213) 354-4321
91103 W. O. PAINE/ MS 83-101/ JET PROPULSION LAB./ 4800 OAK GROVE DR./ PASADENA CA 91103/ (213) 354-4284
91109 ATTN: LIBRARY/ BURROUGHS CORP./ 460 SIERRA MADRE VILLA/ PASADENA CA 91109/ (213) 351-6551 X505
91125 KARL FRYXELL/ DIVISION OF BIOLOGY/ 216-76/ CALIFORNIA INST. OF TECH./ PASADENA CA 91125/ (213) 795-6811 X2818
91311 TOM SANDERSON/ MICROSYSTEMS DIVISION/ MAIL STOP 63-02/ PERTEC COMPUTER CORP./ 20630 NORDHOFF/ CHATSWORTH CA 91311/ (213) 998-1800 X256
91330 RUSSEL J. ABBOTT/ DEPT. OF COMPUTER SCIENCE/ CALIF. STATE UNIV. - NORTHRIDGE/ 18111 NORDHOFF STREET/ NORTHRIDGE CA 91330/ (213) 885-3398
91360 ELIZABETH IBARRA/ 432 E. WILBUR RD #104/ THOUSAND OAKS CA 91360/ (805) 488-4425
91711 TONY NOE/ COMPUTING/ HARVEY MUDD COLLEGE/ CLAREMONT CA 91711/ (714) 626-8511 X2897
91789 JUDY HERRON/ MT. SAN ANTONIO COLLEGE/ 1100 NORTH GRAND AVENUE/ WALNUT CA 91789/ (714) 598-2811
92037 DAVID H. BROWN/ 5709 ABALONE PLACE/ LA JOLLA CA 92037/ (714) 293-6072
92093 ATTN: STUDENT COMPUTING COOP/ APIS DEPT/ C-014/ UNIV. OF CALIFORNIA - SAN DIEGO/ P.O. BOX 109/ LA JOLLA CA 92093/ (714) 452-4723
92093 RICHARD KAUFMAN/ INSTITUTE FOR INFO. SYSTEMS/ C-021/ UNIV. OF CALIFORNIA - SAN DIEGO/ LA JOLLA CA 92093/ (714) 452-4723
92093 KEITH ALLAN SHILLINGTON/ INSTITUTE FOR INFORMATION SYSTEMS/ UCSD MAILCODE C-021/ LA JOLLA CA 92093/ (714) 452-4723
92109 TOM KNOCHE/ 2061 REED/ SAN DIEGO CA 92109/ (714) 270-7099
92127 JOSEPH W. SMITH/ NCR/ 16550 WEST BERNARDO DR./ SAN DIEGO CA 92127/ (714) 485-2864
92138 W. E. CLARK/ DEPT. 244/ P.O. BOX 80158/ SAN DIEGO CA 92138/ (714) 455-1330 X348
92138 SCOTT T. DAVIDSON/ LOGICON/ P.O. BOX 80158/ 4010 SORRENTO VALLEY B/ SAN DIEGO CA 92138/ (714) 455-1330 X348
92277 A. PAUL CROONENBERGHS/ 74415 CACTUS DRIVE/ TWENTY-9 PALMS CA 92277
92501 DAVID H. WELCH/ P.O. BOX 207/ RIVERSIDE CA 92501/ (714) 338-4636
92506 BRUCE YALE/ 15840 SADDLEBACK RD./ RIVERSIDE CA 92506/ (714) 780-7624
92624 WILLIAM J. ARTHUR/ 26016 VIEW POINT DRIVE EAST/ CAPISTRANO BCH CA 92624/ (714) 493-5453
92634 K. W. BIXBY/ BECKMAN INSTRUMENTS/ 2500 HARBOR BLVD./ FULLERTON CA 92634/ (714) 871-4848 X1393
92634 TOM SNYDER/ BLDG 606/M136/ HUGHES AIRCRAFT CO./ P.O. BOX 3310/ FULLERTON CA 92634
92634 JAMES H. WELLS/ BLDG. 606 - MS K232/ HUGHES AIRCRAFT/ P.O. BOX 3310/ FULLERTON CA 92634
92660 ROGER SIPPL/ 1806 TOYON LANE/ NEWPORT BEACH CA 92660/ (714) 642-8977
92701 DAVID A. BEERS/ 1050 CABRILLO PARK DR. APT. 34A/ SANTA ANA CA 92701/ (714) 543-6075
92701 RONALD C. WHITES/ 1090 CABRILLO PARK DR. - APT. 64A/ SANTA ANA CA 92701/ (714) 974-0800
92705 DONALD V. MYHRA/ 14142 GERSHON PL/ SANTA ANA CA 92705/ (714) 544-5314
92713 ERIC OLSEN/ MINICOMPUTER OPERATIONS/ SPERRY UNIVAC/ 2722 MICHELSON DRIVE/ IRVINE CA 92713/ (714) 833-2400
92714 SHERRY L. CAMERON/ PERTEC COMPUTER CORP./ 17112 ARMSTRONG/ IRVINE CA 92714/ (714) 836-4592
92714 RICHARD P. SPRAGUE/ PERTEC COMPUTER CORPORATION/ 17112 ARMSTRONG AVE./ SANTA ANA CA 92714/ (714) 540-8340
92717 DENNIS J. MAINE/ ICS DEPT./ UNIV. OF CALIF. - IRVINE/ IRVINE CA 92717/ (714) 833-5233
92805 JAMES D. LETZ/ GENERAL AUTOMATION INC./ 1055 S. EAST STREET/ ANAHEIM CA 92805/ (714) 778-4800 X261
93010 MARK JUNGWIRTH/ 5408 E. HOLLY RIDGE DR./ CAMARILLO CA 93010
93017 ANDY HARRINGTON/ 218 SAN NAPOLI/ GOLETA CA 93017/ (805) 968-6934 (HOME)
93017 RON JEFFRIES/ 651 ARDMORE/ GOLETA CA 93017/ (805) 964-8964
93407 CLEMENT L. DICKEY/ COMPUTER CENTER/ CALIF. POLYTECH. STATE UNIV./ SANLUIS OBISPO CA 93407/ (805) 546-2004
93407 RICK GILLIGAN/ COMPUTER CENTER/ CALIF. POLYTECH. STATE UNIV./ SANLUIS OBISPO CA 93407/ (805) 546-2004
93454 DAVID W. SALLUME/ 945 VIA FARGO/ SANTA MARIA CA 93454/ (805) 937-4541
93940 ROGER CREAMER/ CTB/ MCGRAW HILL/ DEL MONTE RESEARCH PARK/ MONTEREY CA 93940/ (408) 649-8400
93940 LES VOGEL/ 366 VAN BUREN - APT. 4/ MONTEREY CA 93940/ (408) 375-4245
94010 JOHN R. BOGAN/ 1201 VANCOUVER AVE./ BURLINGAME CA 94010/ (415) 343-5452
94022 JAMES B. ONEY/ 1240 MONTE VERDE COURT/ LOS ALTOS CA 94022/ (415) 961-8825
94025 DAVID F. FRICK/ 920 PEGGY LANE/ MENLO PARK CA 94025/ (415) 329-1013
94040 BRUCE J. EDMUNDSON/ 575 S. RENGSTORFF AVE. - APT. 62/ MOUNTAIN VIEW CA 94040
94040 IRVINE L. MCKNIGHT/ 505 CYPRESS PT. DR. APT. 52/ MOUNTAIN VIEW CA 94040/ (415) 967-0414
94043 BYRON HALE/ 813 FARLEY ST./ MT. VIEW CA 94043
94086 ATTN: UNBOUNDED COMPUTING/ 667 TOYON AVE./ SUNNYVALE CA 94086/ (408) 247-3182
94086 STEVE MCFERRIN/ BENDIX FIELD ENGINEERING/ 155B MOFFETT PARK DR./ SUNNYVALE CA 94086
94086 JIM MERRITT/ 655 SO. FAIROAKS AVENUE APT L-216/ SUNNYVALE CA 94086/ (408) 733-6112
94086 RONALD H. PERROTT/ INSTITUTE FOR ADVANCED COMPUTATION/ P.O. BOX 9071/ SUNNYVALE CA 94086/ (408) 735-0635 X273
94087 R. STEVEN GLANVILLE/ 1531 SANDPIPER CT./ SUNNYVALE CA 94087/ (408) 241-6294
94087 CHARLES T. LEIS/ 560 MIDDLEBURY DR./ SUNNYVALE CA 94087
94104 ROBERT J. RAKER/ PACIFIC GAS & ELECTRIC CO./ 1 POST ST. - NO. 2200/ SAN FRANCISCO CA 94104
94114 LYALL MORRILL/ 705 NOE STREET/ SAN FRANCISCO CA 94114/ (415) 647-8518
94115 WILLIAM R. LLOYD/ 3190 CLAY STREET/ SAN FRANCISCO CA 94115/ (415) 556-2235
94121 ROLAND L. LEE/ 645 35TH AVE/ SAN FRANCISCO CA 94121
94134 SHARLEEN WONG/ 151 MADISON STREET/ SAN FRANCISCO CA 94134/ (415) 586-2467
94305 ATTN: LIBRARY - COPY 2/ BIN 82/ STANFORD LINEAR ACCELERATOR CTR./ P.O. BOX 4349/ STANFORD CA 94305
94305 JOHN BANNING/ MAIL DROP 88/ STANFORD LINEAR ACCELERATOR CENTER / P.O.BOX 4349/ STANFORD CA 94305/ (415) 854-3300 X2802 (OFFICE)/ (415) 325-9226 (HOME)
94305 SASSAN HAZEGHI/ P.O. BOX 4526/ STANFORD CA 94305/ (415) 854-3300 X2359
94305 SUSAN S. OWICKI/ DIGITAL SYSTEMS LABORATORY/ STANFORD UNIVERSITY/ STANFORD CA 94305
94305 CHRIS K. PHILLIPS/ P.O. BOX A-C/ STANFORD CA 94305/ (415) 493-2977/ (408) 988-1450
94305 SCOTT WAKEFIELD/ DIGITAL SYSTEMS LABORATORY/ STANFORD UNIV./ STANFORD CA 94305
94306 TIMOTHY H. JACKINS/ 585 ASHTON AVE./ PALO ALTO CA 94306/ (415) 494-0467
94545 PING K. LIAO/ 2499 CONSTELLATION DR./ HAYWARD CA 94545/ (415) 494-3942 X577 (WORK)
94550 C. W. CHILDERS/ P.O. BOX 761/ LIVERMORE CA 94550/ (415) 422-2779
94598 DOUGLAS N. JOHNSON/ LONGS DRUG STORES/ 141 N. CIVIC DRIVE/ WALNUT CREEK CA 94598/ (415) 937-1174
94609 JAMES A. STARK/ 485 34TH STREET/ OAKLAND CA 94609/ (415) 658-2566
94611 FRANK W. OECHSLI/ CHILD HEALTH & DEVELOPMENT/ UNIV. OF CALIF. - BERKELEY/ 3867 HOWE ST./ OAKLAND CA 94611/ (415) 655-7947
94611 DOUG FORSTER/ 1133 OAKLAND AVENUE/ PIEDMONT CA 94611
94618 MICHAEL L. SIEMON/ 6013 HARWOOD AVE./ OAKLAND CA 94618
94618 ERIC WOGSBERG/ COMPUTER TECHNOLOGY/ 6043 LAWTON AVE./ OAKLAND CA 94618/ (415) 653-4844
94702 RICHARD W. HAMILTON/ 1249 W. BROADWAY/ EUGENE OR 94702
94707 JOHN MEDCALF/ CONCEPT SOFTWARE/ 1842 SAN ANTONIO AVE/ BERKELEY CA 94707/ (415) 526-4035
94720 SUSAN L. GRAHAM/ COMP. SCI. DIVISION-EECS/ 511 EVANS HALL/ U OF CALIFORNIA/ BERKELEY CA 94720/ (415) 642-2059/ 642-1024 (MESSAGES)
94804 MICHAEL C. ARMSTRONG/ BADGER METER INC./ 150 E. STANDARD AVE./ RICHMOND CA 94804/ (415) 233-8220
94941 ROBERT M. BAER/ 379 COUNTRYVIEW DRIVE/ MILL VALLEY CA 94941
94941 JOHN WALKER/ MARINCHIP SYSTEMS/ 16 ST. JUDE ROAD/ MILL VALLEY CA 94941/ (415) 383-1545
94965 BRUCE WOLFE/ 71 SUNSHINE AVE./ SAUSALITO CA 94965/ (415) 332-6242
95008 DAVID J. BELL/ MICRO-SYSTEMS ENGINEERING/ 609 CRAIG AVE./ CAMPBELL CA 95008/ (408) 379-5841/ (408) 379-2498
95014 ATTN: CUPERTINO LIBRARY/ HEWLETT PACKARD/ 11000 WOLFE ROAD/ CUPERTINO CA 95014
95014 MICHAEL J. COOK/ 1658 S. STELLING/ CUPERTINO CA 95014
95014 BILL FINCH/ DATA TERMINALS DIV/ HEWLETT PACKARD CO/ 19400 HOMESTEAD RD/ CUPERTINO CA 95014/ (408) 257-7000
95014 DAVID F. OHL/ P.O. BOX 257/ CUPERTINO CA 95014/ (408) 926-9803

95014 RICHARD PALCHIK/ TYMNET/ 10261 BUBB ROAD/ CUPERTINO CA 95014/ (408) 446-6652
95014 JIM SCHULTZ/ HEWLETT PACKARD/ 11000 WOLFE RD - 42U/ CUPERTINO CA 95014/ (408) 257-7000
95030 DON GILMORE/ GILMORE ASSOCIATES/ P.O. BOX 1723/ LOS GATOS CA 95030/ (408) 395-4800
95030 C. DUDLEY WARNER/ 16345 LOS GATOS BLVD. - # 41/ LOS GATOS CA 95030
95050 JOHN CARTER SR./ 3796 PINEWOOD PLACE/ SANTA CLARA CA 95050/ (408) 988-2629
95050 BRUCE SHERRY/ 1601 WARBURTON AVE. - APT. 12/ SANTA CLARA CA 95050
95051 RONALD L. DANIELSON/ DEPARTMENT OF EECS/ UNIVERSITY OF SANTA CLARA/ SANTA CLARA CA 95051/ (408) 984-4181

95051 JERRY DI MASSA/ T.E.D./ SIEMENS CORP./ 1333 LAWRENCE EXP. - SU ITE 400/ SANTA CLARA CA 95051/ (408) 984-8810
95051 CECIL A. MOORE/ INTEL CORP./ 3065 BOWERS AVENUE/ SANTA CLARA C A 95051/ (408) 987-8080
95051 JOHN NAGLE/ INFORMATION SYSTEMS DESIGN/ 3205 CORONADO DR./ SAN TA CLARA CA 95051/ (408) 249-8100
95051 PAUL J. PANTANO/ TED/ SIEMENS CORP./ 1333 LAWRENCE EXP. - SUIT E 400/ SANTA CLARA CA 95051/ (408) 984-8810
95051 ED SCHOELL/ DEPT. NSAU/ NATIONAL SEMICONDUCTOR/ SANTA CLARA CA 95051
95051 MICHAEL K. STAUFFER/ SEARLE ULTRASOUND/ 2270 MARTIN AVE./ SANT A CLARA CA 95051/ (408) 984-2900
95051 MIKE TRAVIS/ SOFTWARE SUPPORT/ INTERDATA INC./ 3080 OLCOTT ST. - SUITE 125A/ SANTA CLARA CA 95051/ (408) 249-5540
95064 ALAN M. SCHLENGER/ COMPUTER CENTER/ UNIV. OF CALIF. - SANTA CR UZ/ SANTA CRUZ CA 95064
95070 HOWARD S. COLEY JR./ 18950 MCFARLAND AVE./ SARATOGA CA 95070/ (408) 374-5363
95118 MEL R. FISHER/ BUSINESS DEPT./ CALVARY COMMUNITY CHURCH/ 1175 HILLSDALE AVE./ SAN JOSE CA 95118/ (408) 269-8331
95120 ALAN DELWICHE/ LELAND HIGH SCHOOL/ 6677 CAMDEN AVE./ SAN JOSE CA 95120/ (408) 998-6290
95126 JOHN H. KILFOIL/ 1777 TOPEKA AVE./ SAN JOSE CA 95126
95129 GENE H. PRIESTMAN/ 1155-29 WEYBURN LANE/ SAN JOSE CA 95129/ (4 08) 255-3939
95819 DAMON BLOM/ 72 SANDBURG DR./ SACRAMENTO CA 95819/ (916) 455-45 02 (HOME)/ (916) 445-3890 (WORK)
95926 SHING-KIN PUN/ DEPT OF COMPUTER SCIENCE/ CALIFORNIA STATE UNIV . - CHICO/ CHICO CA 95926/ (916) 895-6442
95929 ATTN: CHICO PROJECT - CDC/ SCHOOL OF APPLIED SCIENCES/ CALIFOR NIA STATE UNIV./ CHICO CA 95929/ (916) 895-6713
95929 ORLANDO S. MADRIGAL/ DEPT. OF COMP. SCI./ CALIFORNIA ST. UNIV. - CHICO/ CHICO CA 95929/ (916) 895-6442
97005 JOHN COUSINS/ 2435 S.W. ECOLE - APT. 67/ BEAVERTON OR 97005
97005 DON TERWILLIGER/ 9100 S.W. PARKVIEW LOOP/ BEAVERTON OR 97005/ (503) 644-1933
97051 KEN LESSEY/ 50 WEST ST./ ST. HELENS OR 97051/ (503) 397-1305
97070 DON HARVEY/ P.O. BOX 367/ WILSONVILLE OR 97070
97077 ROY CARLSON/ (50-454)/ TEKTRONIX/ P.O. BOX 500/ BEAVERTON OR 9 7077/ (503) 644-0161 X5616
97077 BOB DIETRICH/ MS 60-45C/ TEKTRONIX INC./ P.O. BOX 500/ BEAVERT ON OR 97077/ (503) 682-3411 X2398
97077 MARVIN WHITE/ MS 94-344/ TEKTRONIX INC./ P.O. BOX 500/ BEAVERT ON OR 97077
97106 JOHN L. RUTIS/ RT 2 BOX 7H/ BANKS OR 97106
97213 BRIAN HANSEN/ 2426 N.E. 57TH AVE. APT #3/ PORTLAND OR 97213/ (503) 284-3537
97229 JAMES N. O'BRIEN/ 13900 SCIENCE PARK DR/ PORTLAND OR 97229/ (5 03) 641-4141
97330 DAVID SKINNER/ ALPHA OMEGA COMPUTER SYSTEMS/ P.O. BOX 392/ COR VALLIS OR 97330/ (503) 754-1911
97331 CHIP WEEMS/ DEPT. OF COMPUTER SCIENCE/ OREGON STATE UNIV./ COR VALLIS OR 97331/ (503) 754-3273
97401 ATTN: NORTHWEST MICROCOMPUTER SYSTEMS/ 121 E. 11TH ST./ EUGENE OR 97401/ (503) 485-0626
97403 ATTN: SUSAN BOWIE - DOCUMENTS ROOM/ COMPUTER CENTER/ UNIV. OF OREGON/ EUGENE OR 97403/ (503) 686-4406
97420 R. BUSH/ NORTHWEST MICRO/ 219 FITZPATRICK BLDG./ COOSBAY OR 97 420/ (503) 269-2432
98033 WILLIAM J. MARSHALL/ 11626 111TH AVE. N.E./ KIRKLAND WA 98033/ (206) 789-2000 (WORK)/ (206) 822-1329 (HOME)
98040 ROBERT EMERSON/ HONEYWELL INFORMATION SYSTEMS/ 9555 SE 36TH ST REET/ MERCER ISLAND WA 98040
98107 R. G. SHERRY/ 5828 FIRST AVE. N.W./ SEATTLE WA 98107/ (206) 78 3-0853
98507 PHIL HUGHES/ P.O. BOX 2847/ OLYMPIA WA 98507/ (206) 753-2315
98632 BUZZ HILL/ EYEDENTIFY INC./ P.O. BOX 2006/ LONGVIEW WA 98632/ (206) 423-3281
99210 FRED J. KELLER/ BUSINESS COMPUTER SYSTEMS/ DATACOMP/ P.O. BOX 1087/ SPOKANE WA 99210/ (509) 456-6908
2033 AUSTRALIA KEN ROBINSON/ DEPT. OF COMPUTER SCIENCE/ UNIVERSITY OF NEW SOU TH WALES/ P.O. BOX 1/ KENSINGTON N.S.W. 2033/ AUSTRALIA/ 663 0351
2049 AUSTRALIA DAVID POWERS/ 259A TRAFALGAR STREET/ PETERSHAM N.S.W. 2049/ AU STRALIA
2308 AUSTRALIA SIMON/ DEPT OF MATHEMATICS/ UNIV. OF NEWCASTLE/ NEWCASTLE N.S. W. 2308/ AUSTRALIA/ 68 5787
2600 AUSTRALIA ATTN: PURCHASING OFFICE/ RESEARCH SCHOOL OF PHYSICAL SCIENCES/ AUSTRALIAN NATIONAL UNIVERSITY/ P.O. BOX 4/ CANBERRA A.C.T. 2600/ AUSTRALIA/ 492143
2616 AUSTRALIA ATTN: SCHOOL OF INFORMATION SCIENCES/ CANBERRA COLLEGE OF ADVA NCED EDUCATION/ P.O. BOX NO. 1/ BELCONNEN A.C.T. 2615/ AUSTRALIA
3083 AUSTRALIA JOHN EDWARDS/ LA TROBE UNIVERSITY/ BUNDOORA VICTORIA 3083/ AUS TRALIA/ 478 3122
3165 AUSTRALIA GEOFFREY A. CLEAVE/ 18 NEIL COURT/ E. BENTLEIGH VICTORIA 3165/ AUSTRALIA
3216 AUSTRALIA P. S. EDWARDS/ 2/100 BARRABOOL ROAD/ HIGHTON VICTORIA 3216/ AU STRALIA
6009 AUSTRALIA J. S. ROHL/ DEPT. OF COMPUTER SCIENCE/ U OF WESTERN AUSTRALIA/ NEDLANDS W.A. 6009/ AUSTRALIA
A-4060 AUSTRIA KARL PRAGERSTORFER/ EDERACKERSTRASSE 11/7/ LEONDING A-4060/ AU STRIA/ (0043) 732-825392
B-2510 BELGIUM WIM STEVENS/ DRABSTRAAT 49/ MORTSEL B-2510/ BELGIUM
BRAZIL ROBERIO DIAS/ P.O. BOX 30028/ SAO PAULO/ BRAZIL
CANADA PETER GROGONO/ 73 ROXTON CRESCENT/ MONTREAL WEST QUEBEC/ CANAD A/ (514) 879-4251 (DAY)
CANADA STUART LYNNE/ 315A EVERGREEN DR./ PORT MOODY B.C./ CANADA/ (60 4) 939-2757
B2G 1C0 CANADA MAREK WIECHULA/ ST. FRANCIS XAVIER UNIV./ BOX 67/ ANTIGONISH N . SCOTIA B2G 1C0/ CANADA/ (902) 867-2275
H1G 3S5 CANADA M. MICHEL COURCHESNE/ 11471 VALADE/ MONTREAL QUEBEC H1G 3S5/ C ANADA/ (514) 324-5694/ (514) 281-8362
H3C 3J7 CANADA GUY LAPALME/ DEPT. D'INFORMATIQUE/ UNIVERSITE DE MONTREAL/ C.P . 6128 - SUCC. A/ MONTREAL QUEBEC H3C 3J7/ CANADA
H3C 3J7 CANADA DANIEL THALMANN/ DEPT D'INFORMATIQUE ET RECHERCHE/ UNIV. DE MO NTREAL/ CASE POSTALE 6128 - SUCC A/ MONTREAL QUEBEC H3C 3J7/ CANADA/ (514) 343-7477
H4T 1N1 CANADA S. MATTHEWS/ AES DATA LTD./ 570 RUE MCCAFFREY/ MONTREAL QUEBEC H4T 1N1/ CANADA/ (514) 341-5430
J1K 2R1 CANADA M. JEAN-MARIE DIRAND/ SERVICE DE L'INFORMATIQUE/ UNIVERSITE DE SHERBROOKE/ 2500 BOUL. UNIVERSITE/ SHERBROOKE QUEBEC J1K 2R1/ CANADA/ (819) 565-5575
J1K 2R1 CANADA JACQUES HAGUEL/ DEPT MATHS/ UNIVERSITE DE SHERBROOKE/ SHERBROO KE QUEBEC J1K 2R1/ CANADA/ (819) 565-3608
K1N 6N5 CANADA H. TAYLOR/ COMPUTING CENTRE/ APPLICATIONS DEPT./ U OF OTTAWA/ OTTAWA ONTARIO K1N 6N5/ CANADA/ (613) 231-5094
K7L 3N6 CANADA JACK HUGHES/ COMPUTING CENTRE/ DUPUIS HALL/ QUEEN'S UNIVERSITY / KINGSTON ONTARIO K7L 3N6/ CANADA/ (613) 547-2800/ (613) 547-2951
K9K 1J1 CANADA D. R. WESTLUND/ 1478 FIRWOOD CRESCENT/ PETERBOROUGH ONTARIO K9 K 1J1/ CANADA
L0C 1K0 CANADA JAMES D. CALLADINE/ RR #4 CONCESSION 7/ OXBRIDGE ONTARIO L0C 1 K0/ CANADA
L5L 2E9 CANADA CARLO LOCICERO/ 3501 GLEN ERIN DRIVE #401/ MISSISSAUGA ONTARIO L5L 2E9/ CANADA/ (416) 826-8640
L5N 1K7 CANADA PETER HAYNES/ CONTROL DATA CANADA LTD./ 1855 MINNESOTA COURT-S TREETSVILLE/ MISSISSAUGA ONTARIO L5N 1K7/ CANADA/ (416) 826-8640 X238
L5N 1K7 CANADA DAVID JONES/ CONTROL DATA CANADA LTD./ 1855 MINNESOTA COURT-ST REETSVILLE/ MISSISSAUGA ONTARIO L5N 1K7/ CANADA/ (416) 826-8640 X262
L8S 4K1 CANADA CHRIS BRYCE/ APPLIED MATH. COMPUTER LAB/ MCMASTER UNIVERSITY/ HAMILTON ONTARIO L8S 4K1/ CANADA/ (416) 525-9140 X4689
M3M 3B9 CANADA ATTENTION: SANDRA WRIGHT/ DEFENCE & CIVIL INST. OF ENVIRMNTL M ED/ P.O. BOX 2000/ DOWNSVIEW ONTARIO M3M 3B9/ CANADA/ (416) 633-4240 X300
M4R 1V2 CANADA TOM A. TROTTIER/ 411 DUPLEX AVE. - APT. 119/ TORONTO ONTARIO M 4R 1V2/ CANADA/ (416) 488-8802
M5H 3W6 CANADA RUSSELL JONES/ 427 ELM ROAD/ TORONTO ONTARIO M5H 3W6/ CANADA/ (416) 592-6758 (BUS)/ (416) 486-7756 (RES)
M5V 2S9 CANADA ATTENTION: G. TER HOFSTEDE/ DATA CENTRE/ THE GLOBE AND MAIL/ 4 44 FRONT ST. WEST/ TORONTO ONTARIO M5V 2S9/ CANADA
N2L 3G1 CANADA ATTN: HELEN SMITH/ COMPUTER CENTER/ 1088B M AND C/ U OF WATERL OO/ WATERLOO ONTARIO N2L 3G1/ CANADA/ (519) 885-1211 X3430
R3E 0W3 CANADA BILL WINSPUR/ COMPUTER SERVICES - HEALTH SCIENCES/ UNIVERSITY OF MANITOBA/ 753 MCDERMOT AVE./ WINNIPEG MANITOBA R3E 0W3/ CANADA/ (204) 786-3630
S7N 0W0 CANADA D. W. MACLEAN/ DEPT. OF MATHEMATICS/ UNIV. OF SASKATCHEWAN/ SA SKATOON SASK. S7N 0W0/ CANADA
T2N 1N4 CANADA BETTY CLIFFORD/ COMPUTER SERVICES/ 058 MATH. SCIENCE/ UNIV. OF CALGARY/ 2920-24 AVE. N.W./ CALGARY ALBERTA T2N 1N4/ CANADA
V4M 3T9 CANADA ROBERT M. GREEN/ ROBELLE CONSULTING LTD./ 5421 10TH AVE. - #13 0/ DELTA B.C. V4M 3T9/ CANADA/ (604) 943-8021
V8P 5J2 CANADA GORDON STUART/ TECHNICAL AND VOCATIONAL INST./ CAMOSUN COLLEGE / 1950 LANSDOWNE RD./ VICTORIA B.C. V8P 5J2/ CANADA/ (604) 592-1291 X248
Y1A 3P5 CANADA KEN SYLVESTRE/ 12 TAGISH ROAD/ WHITEHORSE YUKON Y1A 3P5/ CANAD A/ (403) 667-7372
CHILE ATTN: CECICO/ UNIVERSIDAD CATOLICA DE CHILE/ CASILLA 114-D/ SA NTIAGO/ CHILE/ 513548
CHINA JWO-WU MOU/ COMPUTER SCIENCE DEPT./ CHIAO-TUNG UNIVERSITY/ HSI NCHU TAIWAN 300/ CHINA
7 CZECHOSLOVAKIA JOSEF JINOCH/ VVC CKD PRAHA/ PRAHA 9 NA HARFE 7/ CZECHOSLOVAKI A/ 820841/664
DK-2730 DENMARK ROLF MOLICH/ DANSK DATA ELEKTRONIK/ GENERATORVEJ 6A/ HERLEV DK -2730/ DENMARK/ 45 2 84 50 11
DK-9220 DENMARK UFFE MOLLER/ DATANOMUDDANNELSEN/ LANGAGERVEJ 16/ AALBORG OST D K-9220/ DENMARK/ (08) 15 81 00
EL SALVADOR ROBERTO ARGUETA/ CENTRO DE COMPUTO/ UNIVERSIDAD DE EL SALVADOR / SAN SALVADOR/ EL SALVADOR/ 260017 X50
F-00100 FINLAND HANNU ERKIO/ DEPT. OF COMPUTER SCIENCE/ UNIVERSITY OF HELSINKI / TOOLONKATU 11/ HELSINKI 10 SF-00100/ FINLAND/ 90-440703
F-00130 FINLAND HEIKKI KASKELMA/ OY SOFTPLAN AB/ EROTTAJANKATU 9 A/ HELSINKI S F-00130/ FINLAND/ (9) 0-644306
F-02150 FINLAND ATTN:TECHNICAL RESEARCH CENTRE OF FINL/ COMPUTING SERVICE/ VUO RIMIEHENT 5/ ESPOO SF-02150/ FINLAND/ 90-4561
F-33340 FINLAND ANTTI ARVELA/ RUNKOKATU 6 A 8/ TAMPERE 34 SF-33340/ FINLAND
F-33500 FINLAND HANNU JAAKKOLA/ ITSENAISYYDENKATU 16 I 75/ TAMPERE 50 SF-33500 / FINLAND/ 931 612618
F-31450 FRANCE MICHEL GALINIER/ LA GIRAGLIA/ ESPANES F-31450/ FRANCE
F-34075 FRANCE ATTN: CENTRE DE RECHERCHE/ INFORMATIQUE ET GESTION/ UNIV. DES SCIENCES ET TECH. DU LANGUED/ AVENUE D'OCCITRAINE/ MONTPELLIER CEDEX F-34075/ FRANCE (67) 63-38-86 X339
F-35031 FRANCE ATTN: DEPT DE MATHEMATIQUE AND INFORMA/ BIBLIOTHEQUE 2 CYCLE/ C/O LE BAIL/ UNIVERSITE DE RENNES/ BP-25A/ RENNES CEDEX F-35031/ FRANCE
F-38041 FRANCE ATTN: UNIVERSITAT DE GRENOBLE/ SERVICE DE MATEMATIQUES APPLIQU EES/ CENTRE DE TRI/ BP N-53/ GRENOBLE CEDEX F-38041/ FRANCE
F-92803 FRANCE BERNARD PERRETTE/ FABRICATION DES BILLETS/ BANQUE DE FRANCE/ B .P. 89/ PUTEAUX F-92803/ FRANCE
D-1000 GERMANY ATTN: FREIE UNIVERSITAT BERLIN/ FB10-WE1/ FR ANGEWANDTE STATIS TIK/ CORRENSPLATZ 2/ BERLIN 33 D-1000/ GERMANY
D-2000 GERMANY ATTN: INSTITUT FUER INFORMATIK/ UNIVERSITAT HAMBURG/ SCHLUETER STRASSE 70/ HAMBURG 13 D-2000/ GERMANY
D-2000 GERMANY THOMAS BERNER/ HERMANN-KAUFFMANN STRASSE 35/ HAMBURG 60 D-2000 / GERMANY/ 040-2506602
D-2300 GERMANY ATTN: INSTITUT FUR INFORMATIK/ UNIVERSITAT KIEL/ OLSHAUSENSTR. 40-60/ KIEL D-2300/ GERMANY
D-4440 GERMANY KWAI-SAND LAM/ HEINRICHSTR. 7/ RHEINE D-4440/ GERMANY/ (0251) 706-3236
D-5000 GERMANY DIETRICH KREKEL/ RECHEN ZENTRUM/ UNIVERSITAT ZU KOLN/ ROBERT K OCH STR 10/ KOLN 41 D-5000/ GERMANY/ 0221/478/5587
D-5100 GERMANY PETER ALTMANN/ GREGORSTR.26/ AACHEN D-5100/ GERMANY
D-7800 GERMANY CH. SCHLIER/ FAKULTAT FUR PHYSIK DER UNIVERSITAT/ HERMANN-HERD ER STR. 3/ FRIEBURG I. BR. D-7800/ GERMANY/ 0761/203 3714
D-8000 GERMANY WERNER REMMELE/ ZT ZFE FL SAR 121/ SIEMENS AG/ OTTO-HAHN-RING 6/ MUNCHEN 83 D-8000/ GERMANY/ 089/6782-4622
HONG KONG WILLIAM M. BRACK/ UNIV. AND POLY. COMPUTER CTR. LTD./ CORE C G /FL/ HONG KONG POLYTECHNIC/ YUK CHOI ROAD / HUNG HOM/ KOWLOON/ HONG KONG
411007 INDIA S. M. VAIDYA/ REGIONAL COMPUTER CENTRE/ POONA UNIVERSITY/ POON A 411007/ INDIA
INDONESIA INDRO S. SUWANDI/ COMPUTER SCIENCE CENTER/ UNIVERSITY OF INDON ESIA/ P.O. BOX 3442/ JAKARTA/ INDONESIA/ (021) 45726
IRELAND ALAN JONES/ 2 GROVE ROAD / MALAHIDE/ DUBLIN/ IRELAND
ISRAEL JUDITH KOVETZ/ COMPUTATION CENTRE/ TEL-AVIV UNIVERSITY/ RAMAT- AVIV/ ISRAEL/ 03 420643
158 JAPAN TERUO HIKITA/ DEPT. OF MATHEMATICS/ TOKYO METROPOLITAN UNIV./ FUKAZAWA SETAGAYA-KU/ TOKYO 158/ JAPAN/ 03-717-0111
544 JAPAN HIROAKI NISHIOKA/ SHOJI-HIGASHI 1-3-7/ IKUNO-KU OSAKA 544/ JAP AN/ 06-751-4891
NEW ZEALAND CHRIS M. BISHOP/ COMPUTING CENTRE/ UNIVERSITY OF OTAGO/ P.O. B OX 56/ DUNEDIN/ NEW ZEALAND/ DUNEDIN 40109 EXT 890
NEW ZEALAND ATTN: DOCUMENTATION OFFICER/ COMPUTER CENTRE/ MASSEY UNIVERSIT Y/ PALMERSTON NORTH/ NEW ZEALAND
NORWAY OLAV NAESS/ WELHAVENSGT.65/ BERGEN/ NORWAY
PAKISTAN ADNNAN KHAN/ 222/7 BLOCK-E/ OPP. WALTON TRAINING CENTRE/ WALTO N ROAD/ LAHORE CANTT./ PAKISTAN/ 83644/ 412193
00590 POLAND MICHAL IGLEWSKI/ INSTITUTE OF COMPUTER SCIENCE/ POLISH ACADEMY OF SCIENCE/ PKIN P.O. BOX 22/ WARSZAWA 00590/ POLAND/ 200211 X2225
PORTUGAL ADELINO CARLOS DE SOUSA/ RUA JOAO PINTO RIBEIRO 7-30 ESQ./ AMA DORA/ PORTUGAL/ 937 315
SINGAPORE JACK PAGE/ PAGE-ASIA ASSOCIATES/ 279-M SELEGIE COMPLEX/ SINGAP ORE-7/ SINGAPORE/ 326102
2001 SOUTH AFRICA JUDY M. BISHOP/ COMPUTER SCI. DIV./APPLIED MATHS DEPT./ UNIV. OF THE WITWATERSRAND/ 1 JAN SMUTS AVENUE/ JOHANNESBURG 2001/ SOUTH AFRICA (11) - 394011 X8656
S-102 62 SWEDEN PATRIK WAHREN/ HUGIN HASSREGISTER AB/ BOX 4180/ STOCKHOLM S-10 2 62/ SWEDEN/ 08/24 51 00
S-350 03 SWEDEN MATS APELKRANS/ BOX 3032/ VAXJO S-350 03/ SWEDEN/ 0470/46363
CH-1007 SWITZERLAND ANA-MARIA SCHMIT/ CALCULATRICES DIGITALES/ ECOLE POLYTECHNIQUE FEDERALE/ 16 CH. DE BELLERIVE/ LAUSANNE CH-1007/ SWITZERLAND/ 021 47 26 59
CH-1224 SWITZERLAND RAYMOND MOREL/ 98 CH. DE LA MONTAGNE/ GENEVA CH-1224/ SWITZERL AND
CH-6900 SWITZERLAND HANS J. METZDORF/ HAUPTPOSTLAGERND/ LUGANO CH-6900/ SWITZERLAN D/ 0039/332/780131 X1079
CH-8006 SWITZERLAND PETER U. SCHULTHESS/ INSTITUT FUER INFORMATIK/ UNIVERSITAET ZU ERICH/ KURVENSTRASSE 17/ ZUERICH CH-8006/ SWITZERLAND
THE NETHERLANDS J. F. WILKES/ SHAPE TECHNICAL CENTRE/ POST BOX 174/ DEN HAAG/ THE NETHERLANDS
THE NETHERLANDS TOM VAN DER HOEVEN/ HAGEDOORNSWEG/ NIEBERT/ THE NETHERLANDS
UNITED KINGDOM J. AMBLER/ 21 KILKEVAN TERRACE - WHILFIELD/ DUNDEE SCOTLAND/ U NITED KINGDOM
UNITED KINGDOM R. D. FREEMAN/ EDP DEPT./ PLESSEY CO. LTD./ TITCHFIELD NEAR FA REHAM/ HAMPSHIRE ENGLAND/ UNITED KINGDOM
UNITED KINGDOM D. S. COCHRANE/ 4 ENNIS CLOSE/ HARPENDEN HERTS/ UNITED KINGDOM
UNITED KINGDOM D. J. HOWORTH/ 27 CHILTERN ROAD/ HITCHIN HERTS/ UNITED KINGDOM

APPLICATIONS

Editor: Rich Cichelli 901 Whittier Drive
Allentown, PA 18103
USA
(phone: 1-215-253-6155 work
1-215-797-9690 home)

PLEASE SUBMIT ALL PROGRAMS, SOFTWARE TOOLS, ALGORITHMS, etc. for this section to Rich. Thanks, Andy.

We decided to create a new section for printing Pascal source programs for various applications including software tools and algorithms. Additionally here, we will print news of significant applications programs written in Pascal.

Jim Miner suggested we should index each program so that they may be easily referenced for corrections and criticisms.

Arthur Sale is very enthusiastic about the algorithms section. He suggested that: we allow for 1) the provision for certification of the program by unrelated persons, with clear identification of system used; and
2) critiques of the program for a) standards conformance, or b) style, or c) algorithm, or d) output convenience and general design.

We'll number programs starting with P-1, software tools starting with S-1, and algorithms starting with A-1.

New Software Section

I am looking forward to your new software section in News, partly because I may see something really good there, but even more for the educative role I think it will play among Pascal users. I think back to the large influence that the CACM Algorithms section had on Algol stylistics and their propagation, and on the appreciation of subtle points of the FORTRAN standard.

So besides the big program things (which I shall enjoy perusing for their non-portable features), may I suggest that News could serve as a useful vehicle for procedures and algorithms of very common use? For example, what about a portable lexical analyser for reference PASCAL (as in the Report) which itself had no assumptions about wordsize or setsize beyond those likely to hold in all compilers? Or a tree-balancer? And so on. I'm therefore enclosing a little thing to stir up those other contributors you have: a procedure to produce a 'Pascal standard time and date log record'. You will see that a little machine-dependency is not harmful to communication, provided it is collected and clearly identified as such.

Arthur Sale

Arthur Sale.

** NEWS **

In PUGN # 6, John Banning wrote to solicit programs for his empirical study of Pascal programs. On 78/01/14 he wrote that he had a change in thesis topic which delayed his performing the study. He enclosed a listing of a 12000-line Pascal program which analyzes Pascal programs for style and behavior.

In PUGN # 11, Prof. David Barron announced the formation of a numerical library project. He has recently been in touch with Jan Kok and Reind P.van de Riet of Vrije Universiteit and Mathematisch Centrum in Amsterdam, The Netherlands, who have constructed a large library for Pascal on CDC-6000 series machines.

When numerical analyst Gilbert W. Stewart heard about the project, he was interested, and gave the advice that algorithms beware of the matrix storage differences between Pascal and FORTRAN (row-wise versus column-wise).

Applications

* * * * * * *

Dr. J. ROBSON/ DEPT. OF COMPUTER STUDIES/ UNIV. OF HULL/ COTTIN GHAM ROAD / N. HUMBERSIDE/ HULL, ENGLAND/ UNITED KINGDOM
ANN V. BARROW/ 19 DE VERE STREET/ LANCASTER LANCS/ UNITED KIN GDOM
J. ROSCOE/ COMPUTER DEV. DIV. - SYSTEMS ENG./ I. C. L./ WENLOC K WAY / WESTGORTON/ MANCHESTER ENGLAND/ UNITED KINGDOM/ 061 223 1301 X2589
G. F. LAPPIN/ MATH. AND COMPUTER STUDIES/ PORTADOWN TECH. COLL. LEG/ LURGAN ROAD/ CO. ARMAGH/ PORTADOWN N. IRELAND/ UNITED KINGDOM
M. A. HUSHAND/ AUTOTEST/ MARCONI INSTRUMENTS/ LONGACRES/ ST. A LBANS/ HERTS/ UNITED KINGDOM
P. CARR/ SCHOOL OF MATHEMATICS/ UNIVERSITY OF BATH/ BATH ENGLA ND BA2 7AY/ UNITED KINGDOM
JOHN ELDER/ DEPT. OF COMPUTER SCIENCE/ THE QUEENS UNIVERSITY/ BELFAST 7, IRELAND BT7 1NN/ UNITED KINGDOM
K. LANG/ THE COMPUTER CENTRE/ UNIV. OF BIRMINGHAM/ EDMS ROAD/ BIRMINGHAM ENGLAND B15 2TT/ UNITED KINGDOM/ 021-472-1301 X2233
M. L. PATRICK/ 42 CRANTHORNE AVE. / WANDSWORTH WOOD/ BIRMINGHA M ENGLAND B20 1LL/ UNITED KINGDOM
S. J. HIGGINS/ MOORLANDS SOFTWARE LTD./ NEWTHEAD/ ALSTON CUMBR IA CA9 3LP/ UNITED KINGDOM
R. B. T. BLACK/ ENGINEERING DIV./ NCR LTD./ KINGSWAY WEST/ DUN DEE SCOTLAND DD2 3XX/ UNITED KINGDOM
K. TIZARD/ DEPT. OF M.S.O.R./ UNIV. OF EXETER/ STREATHAM COUR T/ EXETER ENGLAND EX4 4PU/ UNITED KINGDOM
ROBERT G. CLARK/ COMPUTING SCIENCE DEPT./ UNIVERSITY OF STIRLI NG/ STIRLING SCOTLAND FK9 4LA/ UNITED KINGDOM/ STIRLING 3171 X2570
M. J. L. YATES/ F/0603 X66HQ/ GCHQ OAKLEY/ PRIORS ROAD/ CHELTE NHAM ENGLAND GL52 5AJ/ UNITED KINGDOM
P. R. WALEYN/ 57 BARN MEADOW LANE / GREAT BOOKHAM/ SURREY ENGL AND KT23 3EZ/ UNITED KINGDOM
ATTN: DEPT OF BIOCHEMISTRY/ ADRIAN BUILDING/ UNIVERSITY OF LEI CESTER/ LEICESTER ENGLAND LE1 7RH/ UNITED KINGDOM
H. J. ROWE/ COMPUTER LABORATORY/ LEICESTER UNIVERSITY/ LEICEST ER ENGLAND LE1 7RH/ UNITED KINGDOM/ LEIC. 50000
M. D. BEER/ COMPUTER LABORATORY/ UNIVERSITY OF LIVERPOOL/ P.O. BOX 147/ LIVERPOOL ENGLAND L69 3BX/ UNITED KINGDOM
TONY ADDYMAN/ DEPARTMENT OF COMPUTER SCIENCE/ THE UNIVERSITY/ OXFORD ROAD/ MANCHESTER ENGLAND M13 9PL/ UNITED KINGDOM/ 061-273 7121 X5546
RICHARD CLAYTON/ 20 BALDOCK ROAD/ MANCHESTER ENGLAND M20/ UNIT ED KINGDOM
E. J. LEGGE/ DEPT. OF PHYSICS/ U.M.I.S.T./ SACKVILLE STREET/ MANCHESTER ENGLAND M60 1QD/ UNITED KINGDOM
ATTN: THE LIBRARIAN/ ENGINEERING RESEARCH STATION/ BRITISH GAS CORPORATION/ KILLINGWORTH/ NEWCASTLE/TYNE ENGLAND NE99 1LH/ UNITED KINGDOM
T. M. N. IRISH/ 5 NORSE WAY SEDBURY/ CHEPSTOW GWENT NP6 7RH/ U NITED KINGDOM
P. E. OSNOM/ COMPUTER SCIENCE DEPT./ WESTFIELD COLLEGE/ KIDDER PORE AVENUE/ LONDON ENGLAND NW3 7ST/ UNITED KINGDOM/ 01-435-7141 X521
C. A. R. HOARE/ PROGRAMMING RESEARCH GROUP/ OXFORD UNIVERSITY COMPUTING LABORATORY/ 45 BANBURY ROAD/ OXFORD ENGLAND OX2 6PE/ UNITED KINGDOM
R. COLE/ DEPT. OF MATH. SCIENCES AND COMPUTING/ POLYTECHNIC OF THE SOUTH BANK/ BOROUGH ROAD/ LONDON ENGLAND SE1 0AA/ UNITED KINGDOM
M. NUNN/ 407/ CCA/ RIVERWALK HOUSE/ LONDON ENGLAND SW1P 4RT/ U NITED KINGDOM
J. R. POGH/ DEPT. OF COMPUTER STUDIES/ SHEFFIELD CITY POLYTECH NIC/ HERIOT HOUSE - POND HILL/ SHEFFIELD ENGLAND S1 1NN/ UNITED KINGDOM
ROBERT O. HARRIS/ COMPUTER CENTRE/ UNIVERSITY COLLEGE - LONDON/ 19 GORDON ST./ LONDON ENGLAND WC1A 0AH/ UNITED KINGDOM/ 01-387 0858

Applications

ALGORITHMS

A - 1 Random Number Generator

Department of Computer Studies
Bailrigg, Lancaster
Telephone Lancaster 65201 (STD 0524)

University of Lancaster

Head of Department: J. A. Llewellyn B.Sc., M.Phil., F.B.C.S., F.I.M.A.

30th November 1977.

Dear Andy,

In case anyone is interested, as I am, in using Pascal for simulation purposes, I present a Pascal random number generator based on the feedback shift register pseudorandom number generator algorithm given by Whittlesey [1], for a 16-bit word size machine.

Note that :

i) for any other word size, the constants 'pshift', 'qshift' and 'big' must be changed; (see Lewis [2])

ii) bound checking must be suppressed to allow the dual interpretation of variables as both integer and boolean (if anyone is offended by this objectionable programming trick, or if it fails to work under a particular implementation, I have a more portable version - but the price of portability is execution speed); similarly, overflow checking needs to be suppressed, but I have a remedy for this, too;

iii) our implementation has whole word logical operations for and and or, but not for not - hence the use of 'acomp' and 'bcomp'; (note - a subsequent release restricts and and or to boolean operations only)

iv) the function must initially be passed a positive non-zero integer "seed" (parameter 'x'), and will thereafter update this seed and yield a real number in the range (0,1].

The feedback shift register method has been shown by Lewis [2] to give good results, and I have thoroughly tested this version with the usual statistical tests. (Pascal procedures for these tests are available from me).

Incidentally, I would be very pleased to hear from anyone else interested in Pascal and simulation.

Brian A. E. Meekings.

References.

1. Whittlesey, J.R.B. A comparison of the correlational behaviour of random number generators for the IBM 360. Comm ACM 11,9 (Sept 1968).

2. Lewis, T.G. Distribution Sampling for Computer Simulation. D.C.Heath and Co., Lexington and Toronto (1975)

```
function random (var x: integer): real;

  const pshift = 2048;
        qshift = 16;
        big    = 32767;

  type dual = record
                case dummy: boolean of
                  true:  (i: integer);
                  false: (b: boolean)
                end;

  var  a, b, acomp, bcomp: dual;

begin

  (* exclusive or number and number shifted 4 places right *)
  a.i := x; b.i := a.i div qshift;
  acomp.i := big - a.i; bcomp.i := big - b.i;
  a.b := (a.b and bcomp.b) or (acomp.b and b.b);

  (* exclusive or number and number shifted 11 places left *)
  b.i := a.i * pshift;
  acomp.i := big - a.i; bcomp.i := big - b.i;
  a.b := (a.b and bcomp.b) or (acomp.b and b.b);

  (* convert to real result *)
  x := a.i;
  random := a.i / big
end (* random *);
```

(* Jim Miner tried out the algorithm on a PDP8 (23 + 1 bit integers). He made following points: a) the results seemed to be better if the left-shift is circular; b) one has to be careful of multiply overflow; c) the exclusive-or's are more naturally expressed as set operations, and d) a seed of zero yields a constant zero result. His version is printed below. *)

```
FUNCTION RANDOM(VAR SEED: INTEGER) : REAL;
  CONST
    PSHIFT =  65536;  (* 2 ^ 16 *)
    PMOD   = 128;     (* 2 ^ 7  *)
    QSHIFT = 64;      (* 2 ^ 6 *)
    MAXINT = 8388607; (* 2 ^ 23 - 1 *)
  VAR
    A, B:
      RECORD CASE BOOLEAN OF
        TRUE:  (I: INTEGER);
        FALSE: (S: PACKED SET OF 0..23);
      END;
BEGIN
  A.I := ABS(SEED);  B.I := A.I DIV QSHIFT; (*RIGHT SHIFT 6*)
  A.S := (A.S - B.S) + (B.S - A.S);   (* XOR *)
  B.I := A.I MOD PMOD * PSHIFT + A.I DIV PMOD;
                          (*LEFT SHIFT CIRCULAR 16*)
  A.S := (A.S - B.S) + (B.S - A.S);
  SEED := A.I;  RANDOM := A.I / (1.0 + MAXINT)
END (*RANDOM*)
```

A - 2 Timelog

DOCUMENTATION : TIMELOG

Language : Pascal

Written : A.H.J. Sale
Thursday, 1978 March 2, 3.20pm

Use

To improve the quality of production PASCAL programs by making available a standard method of recording the date and time of a run.

User documentation

Timelog is a Pascal procedure which writes on a globally declared file *output*, producing a single line which is a log-record of the date and time. It has no parameters, and is therefore used simply by including the text in the procedure declaration part of a program, and then activated by calling:

timelog

The format of the printed line is chosen to avoid all the confusion created by numeric date and time information by conflicting American, English and European conventions; in addition a measure of redundancy is included by the weekday name. See date given above as an example.

Installation

The procedure will work without modification on Burroughs B6700/7700 installations using the University of Tasmania compiler. On other systems the machine-dependent part (identified clearly in the listing) will have to be altered to acquire the necessary information. (The B6700 pre-defined procedure *timestamp* puts year/month/day/hour/minute/second information into the array parameter elements, thereby avoiding any timing glitches of separate calls.) The lower-case letters and some other characters may have to be converted to suit some systems' lexical requirements. The procedure is easily modified to handle other Indo-European languages (e.g. French) by altering the text strings.

System documentation

The procedure is straightforward. Only a few things are worth noting.

(i) Zeller's congruence is used to compute the weekday from the epoch.

(ii) The date and time are written according to ISO standard format in descending order of significance (apart from the weekday).

(iii) The minute value is printed without zero-suppression; other numeric codes are zero-suppressed, as is normal Pascal convention.

```
00010000        procedure timelog;
00010100                {*******}
00010200        {-------------------------------------------------------------}
00010300        {                                                             }
00010400        {   This procedure prints out a basic log-record on the output  }
00010500        {   file.    It avoids the well-known problems of American and  }
00010600        {   English date conventions, and the 24-hour clock confusion.  }
00010700        {                                                             }
00010800        {-------------------------------------------------------------}
00010900        var
00011000            year            : 01..99;       { two digits, 19xx assumed }
00011100            month           : 1..12;        { month number             }
00011200            day             : 1..31;        { day in month             }
00011300            hour            : 0..23;        { 24-hour clock assumed    }
00011400            minute          : 0..59;        { minutes past the hour    }
00011500
00011600            epoch           : array[0..5] of integer;
00011700                                            { required for B6700       }
00011800
00011900            adjyear         : 00..99;       { Jan & Feb are taken as   }
00012000            adjmonth        : 1..12;        { last months of prev year }
00012100            weekday         : 0..6;         { 0=Sunday, 1=Monday, etc  }
00012200            adjustedhour    : 0..12;        { conventional-clock       }
00012300
00012400        begin
00012500            { The statements between here and the next comment should   }
00012600            { be replaced by the equivalent for your system. Note the   }
00012700            { ranges of the variables documented in the declarations.   }
00012800            timestamp(epoch);
00012900            year   :=epoch[0]-1900;
00013000            month  :=epoch[1];
00013100            day    :=epoch[2];
00013200            hour   :=epoch[3];
00013300            minute :=epoch[4];
00013400            { this closes the machine-dependent part }
00013500
00013600            { compute the adjusted hour we use }
00013700            adjustedhour:=hour mod 12;
00013800            if (adjustedhour = 0) then adjustedhour:=12;
00013900
00014000            { adjust month and year information }
00014100            if (month <= 2) then begin
00014200                adjmonth:=month+10; adjyear:=year-1
00014300            end else begin
00014400                adjmonth:=month-2;  adjyear:=year
00014500            end;
00014600            { zeller's congruence }
00014700            weekday :=
00014800                (((26 * adjmonth - 2) div 10) + day + adjyear +
00014900                (adjyear div 4) + 1) mod 7;
00015000
00015100            { write the timelog out }
00015200            case weekday of
00015300                0: write(output,'Sunday');
00015400                1: write(output,'Monday');
00015500                2: write(output,'Tuesday');
00015600                3: write(output,'Wednesday');
00015700                4: write(output,'Thursday');
00015800                5: write(output,'Friday');
00015900                6: write(output,'Saturday')
00016000            end;
00016100            write(output,', ',(year+1900):4,' ');
00016200            case month of
00016300                1: write(output,'January');
00016400                2: write(output,'February');
00016500                3: write(output,'March');
00016600                4: write(output,'April');
00016700                5: write(output,'May');
00016800                6: write(output,'June');
00016900                7: write(output,'July');
00017000                8: write(output,'August');
00017100                9: write(output,'September');
00017200               10: write(output,'October');
00017300               11: write(output,'November');
00017400               12: write(output,'December')
00017500            end;
00017600            write(output,' ',day:2,', ',adjustedhour:2,':',
00017700                (minute div 10):1,(minute mod 10):1);
00017800            if (hour >= 12) then begin
00017900                writeln(output,' PM.')
00018000            end else begin
00018100                writeln(output,' AM.')
00018200            end;
00018300        end;
```

SOFTWARE TOOLS

S - 1 Compare

```
{*      COMPARE - Compare two text files and report their differences.
 *
 *      Copyright (C) 1977, 1978
 *      James F. Miner
 *      Social Science Research Facilities Center
 *      University of Minnesota
 *
 *      General permission to make fair use in non-profit activities
 *      of all or part of this material is granted provided that
 *      this notice is given. To obtain permission for other uses
 *      and/or machine readable copies write to:
 *
 *              The Director
 *              Social Science Research Facilities Center
 *              25 Blegen Hall
 *              269 19th Ave. So.
 *              University of Minnesota
 *              Minneapolis, Minnesota  55455
 *              U S A
 }


{*      Compare is used to display on "Output" the differences
 *      between two similar texts ("Filea" and "Fileb"). Notable
 *      characteristics are:
 *
 *      - Compare is line oriented. The smallest unit of comparison
 *        is the text line (ignoring trailing blanks).  The present
 *        implementation has a fixed maximum line length.
 *
 *      - By manipulating a program parameter, the user can affect
 *        Compare's sensitivity to the "locality" of differences.
 *        More specifically this parameter, "Minlinesformatch",
 *        specifies the number of consecutive lines on each file
 *        which must match in order that they be considered as
 *        terminating the prior mismatch.  A large value of
 *        "Minlinesformatch" tends to produce fewer but larger
 *        mismatches than does a small value.  The value six appears
 *        to give good results on Pascal source files but may be
 *        inappropriate for other applications.
 *
 *        If compare is to be used as a general utility program,
 *        "Minlinesformatch" should be treated as a program
 *        parameter of some sort.  It is declared as a constant here
 *        for portability's sake.
 *
 *      - Compare employs a simple backtracking search algorithm to
 *        isolate mismatches from their surrounding matches.  This
 *        requires (heap) storage roughly proportional the the size
 *        of the largest mismatch, and time roughly proportional to
 *        the square of the size of the mismatch for each mismatch.
 *        For this reason it may not be feasible to use Compare on
 *        files with very long mismatches.
 *
 *      - To the best of the author's knowledge, Compare utilizes
 *        only features of Standard Pascal.
 }


program compare(filea, fileb, output);

  const
    version = '1.2p  (78/03/01)';
    linelength = 120;               { MAXIMUM SIGNIFICANT INPUT LINE LENGTH }
    minlinesformatch = 6;           { NUMBER OF CONSECUTIVE EQUIVALENT }
                                    { LINES TO END A MIS-MATCH }

  type
    linepointer = ^line;
    line =                          { SINGLE LINE BUFFER }
      packed record
        nextline : linepointer;
        length : 0..linelength;
        image : packed array [1..linelength] of char
      end;

    stream =                        { BOOKKEEPING FOR EACH INPUT FILE }
      record
        cursor, head, tail : linepointer;
        cursorlineno, headlineno, taillineno : integer;
        endfile : boolean
      end;

  var
    filea, fileb : text;
    a, b : stream;
    match : boolean;
    endfile : boolean;              { SET IF END OF STREAM A OR B }

    templine :                      { USED BY READLINE }
      record
        length : integer;
        image : array [0..linelength] of char
      end;

    freelines : linepointer;        { FREE LIST OF LINE BUFFERS }

    same : boolean;                 { FALSE IF NO MIS-MATCHES OCCUR }


  procedure comparefiles;

    function endstream(var x : stream) : boolean;
    begin { ENDSTREAM }
      endstream := (x.cursor = nil) and x.endfile
    end;  { ENDSTREAM }

    procedure mark(var x : stream);

      { CAUSES BEGINNING OF STREAM TO BE POSITIONED BEFORE }
      { CURRENT STREAM CURSOR.  BUFFERS GET RECLAIMED, LINE }
      { COUNTERS RESET, ETC. }

      var
        p : linepointer;

    begin { MARK }
      with x do
        if head <> nil then
          begin
```

```
121           while head <> cursor do { RECLAIM BUFFERS }
122             begin
123               with head^ do
124                 begin  p := nextline;
125                   nextline := freelines;  freelines := head
126                 end;
127               head := p
128             end;
129           headlineno := cursorlineno;
130           if cursor = nil then
131             begin  tail := nil;  taillineno := cursorlineno  end
132         end
133     end;  { MARK }
134
135     procedure movecursor(var x : stream;  var filex : text);
136
137       { FILEX IS THE INPUT FILE ASSOCIATED WITH STREAM X.  THE }
138       { CURSOR FOR X IS MOVED FORWARD ONE LINE, READING FROM X }
139       { IF NECESSARY, AND INCREMENTING THE LINE COUNT.  ENDFILE }
140       { IS SET IF EOF IS ENCOUNTERED ON EITHER STREAM. }
141
142       procedure readline;
143         var
144           newline : linepointer;
145           c, c2 : 0..linelength;
146       begin { READLINE }
147         if not x.endfile then
148           begin
149             c := 0;
150             while not eoln(filex) and (c < linelength) do
151               begin  c := c + 1; templine.image[c] := filex^; get(filex)  end;
152             readln(filex);
153             while templine.image[c] = ' ' do c := c - 1;
154             if c < templine.length then
155               for c2 := c+1 to templine.length do templine.image[c2] := ' ';
156             templine.length := c;
157             newline := freelines;
158             if newline = nil then new(newline)
159             else  freelines := freelines^.nextline;
160             pack(templine.image, 1, newline^.image);
161             newline^.length := c;
162             newline^.nextline := nil;
163             if x.tail = nil then
164               begin  x.head := newline;
165                 x.taillineno := 1;  x.headlineno := 1
166               end
167             else
168               begin x.tail^.nextline := newline;
169                 x.taillineno := x.taillineno + 1
170               end;
171             x.tail := newline;
172             x.endfile := eof(filex);
173           end
174       end;  { READLINE }
175
176     begin { MOVECURSOR }
177       if x.cursor <> nil then
178         begin
179           if x.cursor = x.tail then readline;
180           x.cursor := x.cursor^.nextline;
181           if x.cursor = nil then endfile := true;
182           x.cursorlineno := x.cursorlineno + 1
183         end
184       else
185         if not x.endfile then { BEGINNING OF STREAM }
186           begin
187             readline; x.cursor := x.head;
188             x.cursorlineno := x.headlineno
189           end
190         else  { END OF STREAM }
191           endfile := true;
192     end;  { MOVECURSOR }
193
194     procedure backtrack(var x : stream;  var xlines : integer);
195
196       { CAUSES THE CURRENT POSITION OF STREAM X TO BECOME THAT }
197       { OF THE LAST MARK OPERATION.  I.E., THE CURRENT LINE   }
198       { WHEN THE STREAM WAS MARKED LAST BECOMES THE NEW CURSOR. }
199       { XLINES IS SET TO THE NUMBER OF LINES FROM THE NEW CURSOR }
200       { TO THE OLD CURSOR, INCLUSIVE. }
201
202     begin { BACKTRACK }
203       xlines := x.cursorlineno + 1 - x.headlineno;
204       x.cursor := x.head;  x.cursorlineno := x.headlineno;
205       endfile := endstream(a) or endstream(b)
206     end;  { BACKTRACK }
207
208     procedure comparelines(var match : boolean);
209
210       { COMPARE THE CURRENT LINES OF STREAMS A AND B, RETURNING }
211       { MATCH TO SIGNAL THEIR (NON-) EQUIVALENCE.  EOF ON BOTH STREAMS }
212       { IS CONSIDERED A MATCH, BUT EOF ON ONLY ONE STREAM IS A MISMATCH }
213
214     begin { COMPARELINES }
215       if (a.cursor = nil) or (b.cursor = nil) then
216         match := endstream(a) and endstream(b)
217       else
218         begin
219           match := (a.cursor^.length = b.cursor^.length);
220           if match then
221             match := (a.cursor^.image = b.cursor^.image)
222         end
223     end;  { COMPARELINES }
224
225     procedure findmismatch;
226     begin { FINDMISMATCH }
227       { NOT ENDFILE AND MATCH }
228       repeat  { COMPARENEXTLINES }
229         movecursor(a, filea); movecursor(b,fileb);
230         mark(a); mark(b);
231         comparelines(match)
232       until endfile or not match;
233     end;  { FINDMISMATCH }
234
235     procedure findmatch;
236       var
237         advanceb : boolean; { TOGGLE ONE-LINE LOOKAHEAD BETWEEN STREAMS }
238
239       procedure search(var x : stream; { STREAM TO SEARCH }
240                        var filex : text;
241                        var y : stream; { STREAM TO LOOKAHEAD }
242                        var filey : text);
243
244         { LOOK AHEAD ONE LINE ON STREAM Y, AND SEARCH FOR THAT LINE }
245         { BACKTRACKING ON STREAM X. }
```

```
246
247       var
248         count : integer;  { NUMBER OF LINES BACKTRACKED ON X }
249
250       procedure checkfullmatch;
251         { FROM THE CURRENT POSITIONS IN X AND Y, WHICH MATCH, }
252         { MAKE SURE THAT THE NEXT MINLINESFORMATCH-1 LINES ALSO }
253         { MATCH, OR ELSE SET MATCH := FALSE.  }
254         var
255           n : integer;
256           savexcur, saveycur : linepointer;
257           savexline, saveyline : integer;
258       begin { CHECKFULLMATCH }
259         savexcur := x.cursor;  saveycur := y.cursor;
260         savexline := x.cursorlineno;  saveyline := y.cursorlineno;
261         comparelines(match);
262         n := minlinesformatch - 1;
263         while match and (n <> 0) do
264           begin  movecursor(x, filex);  movecursor(y, filey);
265             comparelines(match);  n := n - 1
266           end;
267         x.cursor := savexcur;  x.cursorlineno := savexline;
268         y.cursor := saveycur;  y.cursorlineno := saveyline;
269       end;  { CHECKFULLMATCH }
270
271     begin { SEARCH }
272       movecursor(y, filey);  backtrack(x, count);
273       checkfullmatch;  count := count - 1;
274       while (count <> 0) and not match do
275         begin
276           movecursor(x, filex);  count := count - 1;
277           checkfullmatch
278         end
279     end;  { SEARCH }
280
281     procedure printmismatch;
282       var
283         emptya, emptyb : boolean;
284
285       procedure writetext(p, q : linepointer);
286       begin { WRITETEXT }
287         writeln;
288         while (p <> nil) and (p <> q) do
289           begin  write(' * ');
290             if p^.length = 0 then writeln
291             else writeln(p^.image : p^.length);
292             p := p^.nextline
293           end;
294         if p = nil then writeln(' *** eof ***');
295         writeln
296       end;  { WRITETEXT }
297
298       procedure writelineno(var x : stream);
299         var
300           f, l : integer;
301       begin { WRITELINENO }
302         f := x.headlineno;  l := x.cursorlineno - 1;
303         write('line');
304         if f = l then write(' ', f:1)
305         else write('s ', f:1, ' to ', l:1);
306         if x.cursor = nil then write(' (before eof)');
307       end;  { WRITELINENO }
308

309       procedure printextratext(var x : stream; xname : char;
310                                var y : stream; yname : char);
311       begin { PRINTEXTRATEXT }
312         write(' extra text on file', xname, ', ');
313         writelineno(x);  writeln;
314         if y.head = nil then
315           writeln(' before eof on file', yname)
316         else
317           writeln(' between lines ', y.headlineno-1:1, ' and ',
318                   y.headlineno:1, ' of file', yname);
319         writetext(x.head, x.cursor)
320       end;  { PRINTEXTRATEXT }
321
322     begin { PRINTMISMATCH }
323       writeln(' **********************************');
324       emptya := (a.head = a.cursor);
325       emptyb := (b.head = b.cursor);
326       if emptya or emptyb then
327         if emptya then printextratext(b, 'b', a, 'a')
328         else printextratext(a, 'a', b, 'b')
329       else
330         begin
331           writeln(' mismatch:');  writeln;
332           write(' filea, ');  writelineno(a);  writeln(':');
333           writetext(a.head, a.cursor);
334           write(' fileb, ');  writelineno(b);  writeln(':');
335           writetext(b.head, b.cursor)
336         end
337     end;  { PRINTMISMATCH }
338
339   begin { FINDMATCH }
340     { NOT MATCH }
341     advanceb := true;
342     repeat
343       if not endfile then advanceb := not advanceb
344       else advanceb := endstream(a);
345       if advanceb then search(a, filea, b, fileb)
346         else search(b, fileb, a, filea)
347     until match;
348     printmismatch;
349   end;  { FINDMATCH }
350
351 begin { COMPAREFILES }
352   match := true;  { I.E., BEGINNINGS-OF-FILES MATCH }
353   repeat
354     if match then findmismatch else begin same := false; findmatch end
355   until endfile and match;
356   { MARK(A);  MARK(B);    MARK END OF FILES, THEREBY DISPOSING BUFFERS }
357 end;  { COMPAREFILES }
358
359 procedure initialize;
360
361   procedure initstream(var x : stream; var filex : text);
362   begin { INITSTREAM }
363     with x do
364       begin
365         cursor := nil;  head := nil;  tail := nil;
366         cursorlineno := 0; headlineno := 0;  taillineno := 0
367       end;
368     reset(filex);  x.endfile := eof(filex);
369   end;  { INITSTREAM }
370
```

```
371
372   begin { INITIALIZE }
373     initstream(a, filea); initstream(b, fileb);
374     endfile := a.endfile or b.endfile;
375     freelines := nil;
376     templine.length := linelength;
377     templine.image[0] := 'x';  { SENTINEL }
378   end; {INITIALIZE}
379
380
381 begin {COMPARE}
382   initialize;
383   page(output);
384   writeln('     compare.  version ', version);
385   writeln;
386   writeln(' match criterion = ', minlinesformatch:1, ' lines.');
387   writeln;
388   if a.endfile then writeln(' filea is empty.');
389   if b.endfile then writeln(' fileb is empty.');
390   if not endfile then
391     begin  same := true;
392       comparefiles;
393       if same then writeln(' no differences.')
394     end
395 end.  { COMPARE }
```

```
(* The following output from Compare was generated on a CDC Cyber 74.  The data used was the
   source text of the Compare program itself, modified in 3 places by performing a change, a
   deletion, and an insertion.  The original source is "File A". *)


    COMPARE.  VERSION 1.2P  (78/03/01)

MATCH CRITERION = 6 LINES.

**********************************
MISMATCH:

FILEA, LINE 101:

*   PROCEDURE COMPAREFILES;

FILEB, LINE 101:

*   PROCEDURE COMPAREFOOLS;

**********************************
EXTRA TEXT ON FILEA, LINE 131
BETWEEN LINES 130 AND 131 OF FILEB

*               BEGIN  TAIL := NIL;  TAILLINENO := CURSORLINENO  END

**********************************
EXTRA TEXT ON FILEB, LINE 162
BETWEEN LINES 162 AND 163 OF FILEA

*             GARBAGE;
```


PERFORMANCE MEASUREMENT OF PASCAL PROGRAMS USING AUGMENT AND ANALYZE

- Andy Mickel 78/03/14.
University Computer Center
University of Minnesota
Minneapolis, MN 55455

What AUGMENT and ANALYZE Do

Suppose you want to examine the execution efficiency of your Pascal program--perhaps to make improvements to those parts which take the most computer time.

AUGMENT and ANALYZE are designed to obtain rough measures of such execution times, particularly for large Pascal programs. Unlike other kinds of performance measurement, AUGMENT and ANALYZE assume the PROCEDURE and FUNCTION to be the smallest unit of a program to be monitored. This is a satisfactory assumption because well-written Pascal programs produced by stepwise refinement naturally are composed of proper-sized procedures and functions.

The general principle used by these programs is that the value of the non-standard Pascal function CLOCK (which returns the elapsed processing time in milliseconds) can be sampled at procedure or function entry and exit. When the the expression (exittime - entrytime), is evaluated the time spent within the particular procedure or function can be ascertained.

AUGMENT is the program which inserts the necessary CLOCK-sampling code into your Pascal source program for every procedure and function entry and exit. It thus causes your program to capture timing information and to write it out to a file.

Next you compile and execute your program, which actually produces the file of dynamic timing measurements.

ANALYZE then reads the timing file produced, and writes a report, which gives the name of each procedure or function, the number of times it was called, and the execution time it consumed for all calls and per call.

AUGMENT and ANALYZE therefore provide a nearly machine-independent method for gathering performance-measurement data about a Pascal program. Most Pascal implementations have the required CLOCK function which returns the elapsed processor time in milliseconds.

It is sometimes necessary to exclude the monitoring of excessively called procedures and functions in large programs. A feature of AUGMENT allows you to specify any number of names to be excluded.

How to Use AUGMENT and ANALYZE

Under CDC 6000/Cyber 70,170 operating systems, AUGMENT and ANALYZE are control statements. Thus the following 3 batch commands do the job:

```
AUGMENT(the file name of your Pascal source program)
PASCAL(INTER/L-,G+)
ANALYZE.
```

The program headings for AUGMENT and ANALYZE are:

```
AUGMENT(INPUT, EXCEPT, INTER, INTER2, OUTPUT)
```

where:

INPUT is the textfile containing the Pascal source program to be AUGMENTed.

```
    EXCEPT is the textfile containing a list of names (one to a line
           with no leading blanks) of procedures and functions to be
           excluded from measurement.  EXCEPT can be an empty file in which
           case no procedures or functions will be excluded.
    INTER  is the textfile on which the AUGMENTed version of
           the Pascal source program is written.
    INTER2 is the binary file on which only the names of each
           procedure and function in the Pascal source program
           is written for use by ANALYZE.
    OUTPUT is the textfile on which error messages are written
           if problems occur during AUGMENTing.  A report is written
           on OUTPUT verifying which procedures or functions were
           excluded, if any.

           The error messages are:
            *TOO MANY PROCEDURES AND FUNCTIONS TO AUGMENT.
                 (A limit of 2000 is imposed.)
            *"BEGIN" EXPECTED.
            *"END" EXPECTED.
                 (There's something wrong with the statement part
                  of the Pascal source program which is being
                  AUGMENTed; it began with some reserved symbol
                  other than "begin" or there weren't enough
                  "END"s to match "BEGIN"s.)
            *"PROGRAM" EXPECTED.
                 (AUGMENT couldn't find "PROGRAM" as the first
                  reserved symbol in the Pascal source program.
                  Possibly the INPUT file was empty.)
            *UNDECLARED LABEL.
                  AUGMENT couldn't find a label referred to by a
                  GOTO statement.


ANALYZE(OUTPUT, INTER2, TIMING)

where:
    OUTPUT is the textfile on which the performance measurement
           report is written or alternatively the error message:
            *TIMING FILE EMPTY.
    INTER2 is the binary file on which the names of each procedure
           and function in the Pascal source program was written by
           AUGMENT.
    TIMING is the binary file containing the dynamic timing
           measurements resulting from execution of the AUGMENTed
           Pascal program.

     Note:  The identifier "TIMING" is added to the Pascal source
program by AUGMENT and must not appear in any procedure or function
which is to be monitored.  When you use AUGMENT and ANALYZE, it is
probably a good idea to consider the file names INTER, INTER2, and
TIMING reserved.

     In summary, there are four steps to the performance measurement
process:

1) [Pascal source program] -> AUGMENT -> INTER and INTER2
                              *******

2)                 INTER -> PASCAL Compiler -> [Pascal binary program]
                            ***************
3) [input data for
    Pascal program] -> Pascal binary program -> TIMING and [results
                       **********************    from Pascal program]

4)       TIMING and INTER2 -> ANALYZE -> [performance measurement report]
                              *******
```

EXAMPLE

Below are a test program, its AUGMENTed version, and the performance measurement report:

The source of the test program:

```
PROGRAM TEST(OUTPUT);
  LABEL  5;
  VAR   N: INTEGER;

  PROCEDURE A;

    PROCEDURE B;
    BEGIN  N := N + 1;
      IF ODD(N DIV 2) THEN A ELSE B
    END (*B*) ;

  BEGIN (*A*)
    N := N + 1;
    IF N > 200 THEN GOTO 5;
    B
  END (*A*) ;
BEGIN N := 0;  A;  5:  END.
```

The AUGMENTed version of the test program:

```
PROGRAM TEST(OUTPUT,TIMING);
  LABEL   5;
  VAR
TIMING:FILE OF PACKED RECORD I:0..2000;T:0..99999999;M:0..2 END;
   N: INTEGER;

  PROCEDURE A;

    PROCEDURE B;
    BEGIN
WITH TIMING^ DO BEGIN I:=   3;T:=CLOCK;M:=0 END;PUT(TIMING);
  N := N + 1;
      IF ODD(N DIV 2) THEN A ELSE B
    ;
WITH TIMING^ DO BEGIN I:=   3;T:=CLOCK;M:=1 END;PUT(TIMING)
END
 (*B*) ;

  BEGIN
WITH TIMING^ DO BEGIN I:=   1;T:=CLOCK;M:=0 END;PUT(TIMING);
 (*A*)
    N := N + 1;
    IF N > 200 THEN  BEGIN
WITH TIMING^ DO BEGIN I:=   2;T:=CLOCK;M:=2 END;PUT(TIMING);
GOTO 5 END
;
    B
  ;
WITH TIMING^ DO BEGIN I:=   2;T:=CLOCK;M:=1 END;PUT(TIMING)
END
 (*A*) ;
BEGIN REWRITE(TIMING);
WITH TIMING^ DO BEGIN I:=   1;T:=CLOCK;M:=0 END;PUT(TIMING);
 N := 0;  A;  5:  ;
WITH TIMING^ DO BEGIN I:=   1;T:=CLOCK;M:=1 END;PUT(TIMING)
END
.
```

The report from ANALYZE:

PERFORMANCE MEASUREMENT SUMMARY FOR PASCAL PROGRAM: TEST

MODULE NAME	CALLS TIMES CALLED	CALLS PERCENT OF TOTAL	EXECUTION TIME (MILLISECONDS) AVERAGE PER CALL	EXECUTION TIME (MILLISECONDS) MODULE TOTAL	EXECUTION TIME (MILLISECONDS) PERCENT OF TOTAL
A	52	25.490	0.15	8	27.586
B	151	74.020	0.13	20	68.966
TEST	1	0.490	1.00	1	3.448
TOTALS	204	100.000	0.14	29	100.000

From the summary provided by AUGMENT and ANALYZE, you can identify which procedures and functions to improve for greater execution efficiency. In general, it pays to concentrate on procedures and functions which are frequently called and take a significant amount of the execution time of the total program. Procedures and functions which have a large average execution time per call, but which are only called a few times are not worth worrying about.

If one or more procedures or functions seem to dominate the results, it might be a good idea to monitor the program with these modules excluded from measurement. Use the except feature provided by AUGMENT.

History

AUGMENT and ANALYZE were conceived originally under the names PROFILE and PRINRES in 1975-1976 by S. Matwin and M. Missala, of the Polish Academy of Sciences Computer Centre, PKiN, Warwaw, Poland. The goal of the project was to build a simple tool to measure very large programs -- such as the Pascal compiler itself. A paper describing their successful work entitled: "A Simple, Machine Independent Tool for Obtaining Rough Measures of Pascal Programs," appeared in SIGPLAN Notices (11:8) August, 1976, pages 42-45.

Their successful implementation was on CDC machines using Pascal-6000.

In 1976, Richard J. Cichelli of Lehigh University Mathematics Department and the American Newspaper Publishers Association Research Institute, obtained the programs and documented and distributed them to the Pascal community in the United States.

In 1977, Herb Rubenstein and Andy Mickel of the University of Minnesota Computer Center, modified the programs for coding style and to increase portability, fixed bugs, and improved the performance of the programs themselves. We also removed several limitations (the built-in restrictions regarding the use of non-local GOTOs within procedures and functions as well as the monitoring of procedures named NEXTCH).

The programs are now supported with the Pascal-6000 system which is distributed to CDC installations around the world.

(* Note: In the programs listings following,
empty comments denote lines with possible
or outright machine dependencies. *)

```
{*      AUGMENT - AUGMENT PASCAL PROGRAMS WITH CODE TO GATHER
*                EXECUTION TIME PERFORMANCE MEASUREMENTS.
*
*       S. MATWIN   AND
*       M. MISSALA              1975.
*       POLISH ACADEMY OF SCIENCES COMPUTER CENTRE.
*       PKIN, WARSAW POLAND.
*
*       REFERENCE:  "A SIMPLE MACHINE INDEPENDENT
*                    TOOL FOR OBTAINING ROUGH MEASURES
*                    OF PASCAL PROGRAMS."
*                    SIGPLAN NOTICES, 1976 AUGUST, PP. 42-45.
*
*
*       MODIFIED, GENERALIZED, AND RENAMED
*       FROM "PROFILE" TO "AUGMENT" BY:
*       A. B. MICKEL            77/08/04.
*       H. U. RUBENSTEIN        77/06/01.
*       UNIVERSITY OF MINNESOTA COMPUTER CENTER
*       MINNEAPOLIS, MN 55455 USA.
*
*       THE NAMES AND ORGANIZATIONS GIVEN HERE MUST NOT BE
*       DELETED IN ANY USE OF THIS PROGRAM.
*
*       SEE THE PTOOLS WRITEUP (UNDER MEASURE) FOR
*       EXTERNAL DOCUMENTATION.
*
**      AUGMENT (INTERNAL DOCUMENTATION).
*
*       AUGMENT INSERTS CODE TO CREATE A TIMING FILE IN THE PROGRAM
*       HEADER, DECLARATION PART, AND STATEMENT PART OF THE PROGRAM
*       TO BE MONITORED.  CODE IS ALSO INSERTED IN THE STATEMENT
*       PART OF EACH PROCEDURE AND FUNCTION TO WRITE CLOCK
*       MEASUREMENTS (AT ENTRY, EXIT, OR GOTOENTRY) TO THE TIMING
*       FILE WHEN THE PROGRAM IS EXECUTED.
*
*       AUGMENT MUST PARSE A SUBSET OF PASCAL AND THEREFORE HAS A
*       LEXICAL ANALYZER.  THE TIMING FILE IS PROCESSED BY THE
*       COMPANION PROGRAM CALLED ANALYZE.
}

{$R-,T-,P-,U+ }

program augment(input, except, inter, inter2, output);

label
    13 {EXIT FOR PROGRAM ERRORS};

const
          beginsy = 1;
           casesy = 2;
            endsy = 3;
{}       externsy = 4;
{}      fortransy = 5;
{}      forwardsy = 6;
           funcsy = 7;
           gotosy = 8;
          labelsy = 9;
            procsy = 10;
        programsy = 11;
            varsy = 12;
```

```
65        maxmodules = 2000;
66             llmax = 120 { LINE LENGTH MAX };
67             llmin = 72  { LINE LENGTH MIN };
68          alfaleng = 10;
69
70 type
71              alfa = packed array [1 .. alfaleng] of char;
72          codetype = (entry, exit, gotoentry, declare);
73           symbols = beginsy .. varsy;
74           namenode = record
75                         name: alfa;
76                         link: ^ namenode
77                      end;
78         modulecnt = 0 .. maxmodules;
79          labelptr = ^ labelnode;
80         labelnode = record
81                         labl: 0 .. 9999;
82                         declaredin: modulecnt;
83                         next: labelptr
84                      end { LABELNODE };
85
86 var
87              idlen,              {IDENTIFIER LENGTH}
88           lastidlen: 0 .. alfaleng;
89                 sy: symbols;
90               chbuf: array [1 .. alfaleng] of char;
91          identifier: alfa;
92              number: 0 .. 9999;
93                 key: array [symbols] of alfa;
94            badnames,
95       readinglabels: boolean;
96          linelength,
97              colcnt: integer;
98                  ch: char;
99               inter: text       {AUGMENTED PROGRAM FILE};
100             except: text       {FILE OF EXCEPTED MODULE NAMES};
101             inter2: file of alfa {FILE OF ALL MODULE NAMES};
102            badlist: ^ namenode {LIST OF EXCEPTED MODULE NAMES};
103              count: modulecnt  {RUNNING COUNT OF MODULES};
104
105
106 procedure nextch;
107
108    begin
109       get(input);
110       ch := input^;
111       colcnt := colcnt + 1;
112       while (not eoln(input) and (colcnt > linelength)) do
113          get(input);
114       if eoln(input) then
115          colcnt := 0
116    end { NEXTCH };
117
118
119 procedure advance;
120
121    begin
122       if eoln(input)
123       then
124          writeln(inter)
125       else
126          write(inter, ch);
127       nextch
128    end {ADVANCE};
129
130
```

```
131 procedure readid;
132 { GLOBAL : CHBUF,CH,IDENTIFIER,IDLEN,LASTIDLEN }
133
134    begin { READID }
135       idlen := 0;
136       repeat
137          if idlen < alfaleng then
138             begin
139                idlen := idlen + 1;
140                chbuf[idlen] := ch
141             end {IF};
142          nextch
143 {}    until not (ch in ['a' .. 'z', '0' .. '9']);
144       if idlen >= lastidlen
145       then
146          lastidlen := idlen
147       else
148          repeat
149             chbuf[lastidlen] := ' ';
150             lastidlen := lastidlen - 1
151          until lastidlen = idlen;
152       pack(chbuf, 1, identifier)
153    end {READID};
154
155
156 procedure writeid;
157
158    var
159       { GLOBAL : CHBUF,IDLEN }
160       i: integer;
161
162    begin {WRITEID}
163       i := 1;
164       while i <= idlen do
165          begin
166             write(inter, chbuf[i]);
167             i := i + 1
168          end {WHILE}
169    end {WRITEID};
170
171
172 procedure comment;
173
174    begin
175       advance;
176       repeat
177          while ch <> '*' do
178             advance;
179          advance
180       until ch = ')';
181       advance
182    end {COMMENT};
183
184
185 procedure stdcomment;
186
187    begin
188       repeat
189          advance
190       until ch = '}';
191       advance
192    end {STDCOMMENT};
193
194
195 procedure scan;
196
```

```
{ FIND NEXT IDENTIFIER (OR NUMBER IF READINGLABELS).
  SKIP STRINGS AND COMMENTS. }

  label
    21;

{ GLOBAL : IDENTIFIER,CH,NUMBER,BADNAMES,READINGLABELS }


  function nokey(id: alfa): boolean;

    var
      { GLOBAL : SY,KEY }
      i, j: integer;

    begin { BINARY SEARCH }
      i := beginsy;
      j := varsy;
      repeat
        sy := (i + j) div 2;
        if key[sy] <= id then
          i := sy + 1;
        if key[sy] >= id then
          j := sy - 1;
      until i > j;
      nokey := key[sy] <> id
    end { NOKEY };


  begin { SCAN }
    while not eof(input) do
      begin
        while not eoln(input) do
          begin
            if ch = ' '
            then
              advance
            else
{}            if ch in ['a' .. 'z']
              then
                begin
                  readid;
                  readinglabels := false;
                  if nokey(identifier) and not badnames
                  then
                    writeid
                  else
                    goto 21 { EXIT ON KEY OR EXCEPTED ID }
                end {IF}
              else
                if ch in ['0' .. '9']
                then
                  if readinglabels
                  then
                    begin
                      read(number);
                      ch := input^;
                      goto 21 { EXIT ON LABEL }
                    end {IF}
                  else
                    repeat
                      advance
{}                  until not (ch in ['a' .. 'z', '0' .. '9'])
                else
                  if ch = ''''
                  then
                    begin
                      repeat
                        advance
                      until ch = '''';
                      advance
                    end {IF}
                  else
                    if ch = '('
                    then
                      begin
                        advance;
                        if ch = '*' then
                          comment
                      end {IF}
                    else
                      if ch = '{'
                      then
                        stdcomment
                      else
                        advance
          end {WHILE};
        writeln(inter);
        nextch
      end {WHILE};
  21:
  end { SCAN };


procedure complmodule(lastl: labelptr);

{ PROCESS THE BLOCK OF A PROGRAM, PROCEDURE, OR FUNCTION TO FIND
  THE APPROPRIATE CODE INSERTION POINTS.  LASTL IS THE HEAD OF THE
  LIST OF LABELS WHOSE SCOPE APPLIES TO THE BLOCK.  COMPLMODULE
  MUST PARSE LABEL, VAR, PROCEDURE, AND FUNCTION DECLARATIONS, AS
  WELL AS GOTO STATEMENTS AND THE COMPOUND STATEMENT FORMING THE
  STATEMENT PART OF EACH MODULE. }

  var
    { GLOBAL : IDENTIFIER,KEY,SY,CH,READINGLABELS,NUMBER,COUNT }
               name: alfa;
              depth: integer;
             params: boolean;
                 l: labelptr;
        gotolabel: 0 .. 9999;
          looking: boolean;
              tag: modulecnt;


  procedure insertnewtext(code: codetype);

    begin
      case code of
        entry:
          begin
            write(inter, 'with timing^ do begin ');
            write(inter, 'i:=', tag:4, ';');
{}          write(inter, 't:=clock;m:=0' {ENTRY});
            writeln(inter, ' end;put(timing);');
          end {ENTRY};
        exit:
          begin
            writeln(inter, ';');
            write(inter, 'with timing^ do begin ');
            write(inter, 'i:=', tag:4, ';');
{}          write(inter, 't:=clock;m:=1' {EXIT});
            writeln(inter, ' end;put(timing)');
```

```
329                end {EXIT};
330             gotoentry:
331                begin
332                   write(inter, 'with timing^ do begin ');
333                   write(inter, 'i:=', 1^.declaredin:4, ';');
334 {}                write(inter, 't:=clock;m:=2' {GOTOENTRY});
335                   writeln(inter, ' end;put(timing);');
336                end {GOTOENTRY};
337             declare:
338                begin
339                   writeln(inter, 'var ');
340                   write(inter, 'timing:file of packed record ');
341                   write(inter, 'i:0..2000;');
342                   write(inter, 't:0..99999999;');
343                   write(inter, 'm:0..2');
344                   writeln(inter, ' end;');
345                end {DECLARE}
346          end { CASE CODE OF }
347       end { INSERTNEWTEXT };
348
349
350    function nameok: boolean;
351
352    { CHECK PROCEDURE OR FUNCTION NAME AGAINST LIST OF NAMES TO BE
353       EXCLUDED. }
354
355       var
356          { GLOBAL : BADLIST,NAME }
357                   n: ^ namenode;
358             looking: boolean;
359
360       begin { NAMEOK }
361          n := badlist;
362          looking := true;
363          while (n <> nil) and looking do
364             begin
365                looking := n^.name <> name;
366                n := n^.link
367             end {WHILE};
368          nameok := looking
369       end {NAMEOK};
370
371
372    begin {COMPLMODULE}
373 {}    while not (ch in ['a' .. 'z']) do
374          if ch = '('
375          then
376             begin
377                advance;
378                if ch = '*' then
379                   comment
380             end
381          else
382             if ch = '{'
383             then
384                stdcomment
385             else
386                advance;
387       readid;
388       name := identifier;
389       writeid;
390       tag := count;
391       params := false;
392       while not params and (ch <> ';') do
393          if ch = '('
394          then
395             begin
396                advance;
397                if ch = '*'
398                then
399                   comment
400                else
401                   params := true
402             end {IF}
403          else
404             if ch = '{'
405             then
406                stdcomment
407             else
408                advance;
409       if params
410       then
411          while ch <> ')' do { READ THROUGH PARAMETER LIST }
412             if ch = '{'
413             then
414                stdcomment
415             else
416                if ch = '('
417                then
418                   begin
419                      advance;
420                      if ch = '*' then
421                         comment
422                   end {IF}
423                else
424                   advance;
425       if tag = 1 { MAIN PROGRAM }
426       then
427          write(inter, ',timing)')
428       else
429          write(inter, ch);
430       nextch;
431       scan;
432
433 {}    if sy in [forwardsy, externsy, fortransy]
434 {}    then
435 {}       writeid
436 {}    else
437          begin
438             count := count + 1;
439             if count = maxmodules then
440                begin
441                   writeln(' *too many procedures and',
442                           ' functions to augment.');
443                   goto 13
444                end;
445             write(inter2, name);
446             if sy = labelsy { LABEL DECLARATION }
447             then
448                begin { READ LOCAL LABELS }
449                   writeid;
450                   readinglabels := true;
451                   scan;
452                   repeat
453                      new(l);
454                      l^.labl := number;
455                      l^.declaredin := tag;
456                      write(inter, number: 1);
457                      l^.next := lastl;
458                      lastl := l;
459                      scan
460                   until not readinglabels
```

```
461            end {IF};
462
463         while sy in [casesy, endsy] do { TYPE DECLARATION }
464            begin
465               writeid;
466               scan
467            end {WHILE};
468         if tag = 1 { MAIN PROGRAM }
469         then
470            insertnewtext(declare)
471         else
472            if not (sy in [beginsy, funcsy, procsy]) then
473               writeid;
474
475         if sy = varsy
476         then
477            begin
478               scan;
479               while sy in [casesy, endsy] do
480                  begin { CASESY,ENDSY IN AN ANONYMOUS TYPE }
481                     writeid;
482                     scan
483                  end {WHILE}
484            end {IF};
485
486         while sy in [funcsy, procsy] do
487            begin
488               writeid;
489               complmodule(lastl)
490            end {WHILE};
491
492         if sy = beginsy { STATEMENT PART }
493         then
494            begin
495               depth := 1;
496               writeid;
497               if tag = 1 { MAIN PROGRAM }
498               then
499                  writeln(inter, ' rewrite(timing);')
500               else
501                  writeln(inter);
502               if nameok then
503                  insertnewtext(entry)
504            end {IF}
505         else
506            begin
507               writeln(' *''begin'' expected.');
508               goto 13
509            end {ELSE};
510
511         repeat { LOOK FOR LAST ENDSY }
512            scan;
513            if sy = gotosy
514            then
515               begin { CHECK AGAINST LOCAL LABELS }
516                  readinglabels := true;
517                  scan;
518                  gotolabel := number;
519                  readinglabels := false;
520                  looking := true;
521                  l := lastl;
522                  while (l <> nil) and looking do
523                        if l^.labl = gotolabel
524                        then
525                           looking := false
526                        else
527                           l := l^.next;
528                  if looking
529                  then
530                     begin
531                        writeln(' *undeclared label ', gotolabel: 1);
532                        goto 13
533                     end {IF}
534                  else
535                     begin
536                        if l^.declaredin <> tag then
537                           begin { EXIT GOTO }
538                              writeln(inter, 'begin');
539                              if nameok then
540                                 insertnewtext(gotoentry)
541                           end {IF};
542                        write(inter, 'goto ', gotolabel: 1);
543                        if l^.declaredin <> tag then
544                           writeln(inter, ' end')
545                     end {ELSE}
546               end {IF}
547            else
548               if sy in [beginsy, casesy]
549               then
550                  begin
551                     depth := depth + 1;
552                     writeid
553                  end {IF}
554               else
555                  if sy = endsy
556                  then
557                     begin
558                        depth := depth - 1;
559                        if depth <> 0 then
560                           writeid
561                     end {IF}
562                  else
563                     begin
564                        writeln(' *''end'' expected.');
565                        goto 13
566                     end {ELSE}
567            until depth = 0;
568            if nameok then
569               insertnewtext(exit);
570            writeln(inter, 'end');
571         end {ELSE};
572      scan
573   end {COMPLMODULE};
574
575
576 procedure readbadnames;
577
578    var
579       { GLOBAL : BADLIST,CHBUF }
580       n: ^ namenode;
581       i: 1 .. alfaleng;
582
583
584    begin { READBADNAMES }
585       writeln('the following procedures will not be augmented' :50);
586       writeln;
587       repeat
588          new(n);
589          for i := 1 to alfaleng do
590             if eoln(except)
591             then
592                chbuf[i] := ' '
```

```
593             else
594                 read(except, chbuf[i]);
595           readln(except);
596           pack(chbuf, 1, n^.name);
597           n^.link := badlist;
598           badlist := n;
599           writeln(n^.name :25);
600           writeln
601       until eof(except)
602     end { READBADNAMES };
603
604
605 begin { MAIN PROGRAM }
606     rewrite(inter);
607     rewrite(inter2);
608     reset(except);
609     ch := input^;
610     count := 1;
611     colcnt := 1;
612     linelength := llmax;
613     lastidlen := alfaleng;
614 {}  linelimit(inter, maxint);
615     key[beginsy  ] := 'begin     ';
616     key[casesy   ] := 'case      ';
617     key[endsy    ] := 'end       ';
618 {}  key[externsy ] := 'extern    ';
619 {}  key[fortransy] := 'fortran   ';
620 {}  key[forwardsy] := 'forward   ';
621     key[funcsy   ] := 'function  ';
622     key[gotosy   ] := 'goto      ';
623     key[labelsy  ] := 'label     ';
624     key[procsy   ] := 'procedure ';
625     key[programsy] := 'program   ';
626     key[varsy    ] := 'var       ';
627     badlist := nil;
628     if not eof(except) then
629         readbadnames;
630     scan;
631     readinglabels := false;
632     if sy = programsy
633     then
634         begin
635             write(inter, 'program');
636             complmodule(nil)
637         end {IF}
638     else
639         writeln(' *''program'' expected.');
640 13:
641 end { AUGMENT }.
```

```
  1 {*        ANALYZE - ANALYZE AND SUMMARIZE EXECUTION TIME
  2 *                  PERFORMANCE MEASUREMENTS FROM AN AUGMENTED
  3 *                  PASCAL PROGRAM.
  4 *
  5 *         S. MATWIN   AND
  6 *         M. MISSALA        1975.
  7 *         POLISH ACADEMY OF SCIENCES COMPUTER CENTRE.
  8 *         PKIN, WARSAW POLAND.
  9 *
 10 *         MODIFIED, GENERALIZED, AND RENAMED
 11 *         FROM "PRINRES" TO "ANALYZE" BY:
 12 *         A. B. MICKEL      77/11/18.
 13 *         H. U. RUBENSTEIN  77/05/15.
 14 *         UNIVERSITY OF MINNESOTA COMPUTER CENTER
 15 *         MINNEAPOLIS, MN 55455 USA.
 16 *
 17 *         THE NAMES AND ORGANIZATIONS GIVEN HERE MUST NOT BE
 18 *         DELETED IN ANY USE OF THIS PROGRAM.
 19 *
 20 *         SEE THE PTOOLS WRITEUP (UNDER MEASURE) FOR
 21 *         EXTERNAL DOCUMENTATION.
 22 *
 23 *
 24 **        ANALYZE (INTERNAL DOCUMENTATION).
 25 *
 26 *         ANALYZE READS TWO FILES.  INTER2 IS THE FILE CONTAINING
 27 *         THE MODULE (PROCEDURE AND FUNCTION) NAMES WHICH ARE USED
 28 *         WHEN THE RESULTS ARE SORTED AND WRITTEN OUT.  TIMING IS
 29 *         THE FILE CONTAINING THE EXECUTION TRACE OF THE PROGRAM
 30 *         BEING MONITORED.
 31 *
 32 *         WITHIN ANALYZE, THE PROCEDURE NAMED PROCESSBODY DOES THE
 33 *         ACTUAL ANALYSIS BY DETERMINING EVERY TIME INTERVAL:
 34 *
 35 *                       TIME[EXIT] - TIME[ENTRY].
 36 *
 37 *         EVERY GOTOEXIT FROM A PROCEDURE IS CONSIDERED TO BE A
 38 *         SPECIAL KIND OF PROCEDURE ENTRY, SO THAT ALL PROCEDURES
 39 *         WHICH, UP TO THAT TIME HAVE BEEN ENTERED BUT NOT NORMALLY
 40 *         EXITED, ARE ALL EXITED BY THE GOTOENTRY.  SEE THE COMPANION
 41 *         PROGRAM CALLED AUGMENT.
 42 }
 43
 44
 45 {$R-,T-,P-,U+}
 46
 47
 48 program analyze(output, inter2, timing);
 49
 50 label
 51     13;
 52
 53 const
 54     alfaleng = 10;
 55     maxnames = 2000;
 56
 57 type
 58         alfa = packed array [1 .. alfaleng] of char;
 59     tagrange = 1 .. maxnames;
 60  measurement = packed record
 61                           tag: tagrange;
 62                          time: 0 .. 99999999;
 63                          mark: (entry, exit, gotoentry)
 64                        end;
 65      counter = record
 66                          count: integer;
 67                           name: alfa;
 68                     timespent: integer
 69                end;
 70
 71 var
 72        timing: file of measurement;
 73        inter2: file of alfa;
 74       modules: array [tagrange] of counter;
 75        maxtag,
 76           tag: tagrange;
 77      progtime: integer;
 78     totaltime,
 79    totalcalls: integer;
 80
 81
 82 procedure sort(min, max: tagrange);
 83
```

```
{ QUICKSORT WITH BOUNDED RECURSION DEPTH }
{ REQUIRES MIN <= MAX }

   var
         low,
        high: integer;
      midkey: alfa;
        temp: counter;

   begin
      repeat {PICK SPLIT POINT}
         midkey := modules[(min + max) div 2].name;
         low := min;
         high := max;
         repeat {PARTITION}
            while modules[low].name < midkey do
               low := low + 1;
            while modules[high].name > midkey do
               high := high - 1;
            if low <= high then
               begin
                  temp := modules[low];
                  modules[low] := modules[high];
                  modules[high] := temp;
                  low := low + 1;
                  high := high - 1
               end;
         until low > high;

         {RECURSIVELY SORT SHORTER SUB-SEGMENT}
         if high - min < max - low
         then
            begin
               if min < high then
                  sort(min, high);
               min := low
            end
         else
            begin
               if low < max then
                  sort(low, max);
               max := high
            end
      until max <= min
   end {SORT};


procedure processbody;

{ PROCESS TIMING FILE OF DYNAMIC MEASUREMENTS. }

   var
      moduletag: tagrange;
      moduletime: integer;

   begin
      moduletag := timing^.tag;
      moduletime := - timing^.time;
      get(timing);
      while timing^.mark = entry do
         begin
            moduletime := moduletime + timing^.time;
            processbody;
            moduletime := moduletime - timing^.time;
            if (timing^.mark = gotoentry) <= (timing^.tag = moduletag)
               then { ONLY ADVANCE THE TIMING FILE IF A GOTOENTRY
                      IS NOT ENCOUNTERED OR IF A GOTOENTRY IS
                      ENCOUNTERED INTO THE CURRENT MODULE. }
                  get(timing)
         end;
      moduletime := moduletime + timing^.time;
      totalcalls := totalcalls + 1;
      with modules[moduletag] do
         begin
            count := count + 1;
            timespent := timespent + moduletime
         end
   end {PROCESSBODY};


begin {MAIN PROGRAM}
   reset(inter2);
   tag := 1;
   while not eof(inter2) do
      begin
         with modules[tag] do
            begin
               read(inter2, name);
               count := 0;
               timespent := 0;
            end;
         tag := tag + 1
      end;
   maxtag := tag - 1;
   reset(timing);
   if eof(timing) then
      begin
         writeln(' *timing file empty.');
         goto 13
      end;
   progtime := timing^.time;
   totalcalls := 0;

   processbody;

   totaltime := timing^.time - progtime;
   page(output);
   writeln;
   writeln;
   writeln(' performance measurement summary for pascal program:  ',
            modules[1].name,'.');
   writeln;
   writeln('execution time': 62);
   writeln('calls': 27, '(milliseconds)': 35);
   writeln('module': 9, 'times': 13, 'percent': 11, 'average': 15,
      'module': 10, 'percent': 11);
   writeln('name': 8, 'called': 15, 'of total': 11, 'per call': 15,
      'total': 8, 'of total': 13);
   writeln(' ----------', '------': 12, '--------': 11,
      '--------': 15, '------': 9, '--------': 12);
   if maxtag > 1 then
      sort(1, maxtag);
   for tag := 1 to maxtag do
      with modules[tag] do
         begin
            write(name: 11, count: 12,
               ((count * 100) / totalcalls): 11:3);
            if count = 0
            then
               write('----': 15)
            else
               write((timespent / count): 15:2);
```

```
216             writeln(timespent: 9,
217                ((timespent * 100) / totaltime): 12:3)
218           end;
219     writeln(' ==========', '======': 12, '========': 11,
220        '========': 15, '======': 9, '========': 12);
221     writeln('totals': 9, totalcalls: 14, '100.000': 11,
222        (totaltime / totalcalls): 15:2, totaltime: 9, '100.000': 12);
223 13:
224 end {ANALYZE}.
```

PROGRAMS

P - 1 Printme

(* Nearly every programming language has to have a program which can reproduce its source text as output. In 1976, Pascal enthusiast, John Strait of the University of Minnesota, (who incidentally maintains the CDC-6000 compiler for the world), wrote such a program, and it is presented below. When Urs Ammann, of E.T.H., Zurich, (who incidentally authored the CDC-6000 compiler), saw this program, he said he had written a shorter one using a case statement. We have not seen it, but we would like to. *)

```
PROGRAM PRINTME(OUTPUT);
 (*JPS 76/05/26.*)
CONST FIRSTHALF = 9;
 SECONDHALF = 10;
 LENGTH = 22;
 Q = '''';
VAR I,J: INTEGER;
 IMAGE: ARRAY[0..LENGTH] OF
  PACKED ARRAY[1..40] OF CHAR;
BEGIN (* PRINTME *)
IMAGE[ 0] := 'PROGRAM PRINTME(OUTPUT);                ';
IMAGE[ 1] := ' (*JPS 76/05/26.*)                      ';
IMAGE[ 2] := 'CONST FIRSTHALF = 9;                    ';
IMAGE[ 3] := ' SECONDHALF = 10;                       ';
IMAGE[ 4] := ' LENGTH = 22;                           ';
IMAGE[ 5] := ' Q = ''''''''';                             ';
IMAGE[ 6] := 'VAR I,J: INTEGER;                       ';
IMAGE[ 7] := ' IMAGE: ARRAY[0..LENGTH] OF             ';
IMAGE[ 8] := '  PACKED ARRAY[1..40] OF CHAR;          ';
IMAGE[ 9] := 'BEGIN (* PRINTME *)                     ';
IMAGE[10] := 'FOR I := 0 TO FIRSTHALF DO              ';
IMAGE[11] := ' WRITELN(IMAGE[I]);                     ';
IMAGE[12] := 'FOR I := 0 TO LENGTH DO                 ';
IMAGE[13] := ' BEGIN WRITE(''IMAGE['',I:2,''] := '',Q);   ';
IMAGE[14] := ' FOR J := 1 TO 40 DO                    ';
IMAGE[15] := '  IF IMAGE[I][J] = Q                    ';
IMAGE[16] := '   THEN WRITE(Q,Q)                      ';
IMAGE[17] := '   ELSE WRITE(IMAGE[I][J]);             ';
IMAGE[18] := ' WRITELN(Q,'';'')                         ';
IMAGE[19] := ' END;                                   ';
IMAGE[20] := 'FOR I := SECONDHALF TO LENGTH DO        ';
IMAGE[21] := ' WRITELN(IMAGE[I])                      ';
IMAGE[22] := 'END (*PRINTME*).                        ';
FOR I := 0 TO FIRSTHALF DO
 WRITELN(IMAGE[I]);
FOR I := 0 TO LENGTH DO
 BEGIN WRITE('IMAGE[',I:2,'] := ',Q);
 FOR J := 1 TO 40 DO
  IF IMAGE [I][J] = Q
   THEN WRITE(Q,Q)
   ELSE WRITE(IMAGE[I][J]);
 WRITELN(Q,';')
 END;
FOR I := SECONDHALF TO LENGTH DO
 WRITELN(IMAGE[I])
END (*PRINTME*).
```

Articles

Extensions to PASCAL for Separate Compilation

Richard J. LeBlanc
School of Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332

The lack of features in PASCAL to allow procedures and functions to be compiled separately can be of considerable inconvenience in the development of large programs. This weakness is particularly evident when modifications are being made only to limited parts of a program. Modifications of this sort are common, for example, in the maintenance or extension of a PASCAL compiler.

The extensions described below allow the creation of a global environment, separate compilation of routines using that environment, and additions to the environment without requiring recompilation of existing routines and declarations. Four kinds of modules are recognized to implement these features:

1) Declaration modules are used to create an environment.
2) Routine modules are used to provide the bodies of routines declared in an environment.
3) Environment extension modules are used to make extensions to an environment.
4) Main program modules are used to compile the main program body.

Declaration modules

```
<declaration module> ::=
     <declaration heading> <declaration block> .
<declaration heading> ::=
     declarations (<file identifier>) ;
<declaration block> ::= <constant definition part>
     <type definition part> <variable declaration part>
     <routine heading list>
<routine heading list> ::= <empty> |
     <routine heading> {, <routine heading>}
<routine heading> ::= <procedure heading> |
     <function heading>
```

Within the module may be declarations of constants, types and variables just as in a standard main program. Following these declarations come the headings of routines that are to be part of the environment. These headings are identical to the headings in normal routine declarations.

Any identifier defined in a declaration module may be referenced in any other module compiled using the environment created from the declarations. This mechanism allows routines compiled separately to call each other and to use the same global constants, types and variables. Compilation of a declaration module creates

a description of an environment. This description is used to do all type checking at compile-time, just as if separately compiled routines had been compiled as a standard program.

Routine modules

```
<routine module> ::=
     <routine module head> <routine module body> .
<routine module head> ::= <environment head> routines
     ( <identifier> {, <identifier>} );
<environment head> ::= environment (<file identifier>) ;
<routine module block> ::= <constant definition part>
     <type definition part> <variable declaration part>
     <routine declaration part>
```

The file in the environment head specifies the environment in which the module is to be compiled. The list of identifiers in the heading tells the compiler which of the routines defined in the module are to match declarations in the environment and thus are to be callable from outside of the module. Routine modules may also contain declarations of constants, type, variables and local routines. The variables so declared are statically allocated and thus retain their values between calls to the routines in the module. This makes routine modules useful for limiting access to data structures to only those routines that need to manipulate and reference them.

A routine module defines a new name scope, so identifiers used in the global environment may be redefined within a module. When a routine declared in the global environment is defined in a routine module, the declaration of its parameters is repeated and the types must match those specified in the environment declaration. (Since the parameter names are not relevant to type checking, they need not match those in the declaration.)

Environment extension modules

```
<environment extension module> ::=
     <extend heading> <declaration block> .
<extend heading> ::=
     extend (<file identifier>, <new file identifier>) ;
```

Environment extension modules may add any kind of declaration to the environment, but cannot change any existing ones. The environment description from the old environment file is expanded to describe the extended environment and is written as the new environment file.

Main program modules

```
<main module> ::=
     <environment head> <program heading> <block> .
```

Main program modules look exactly like standard PASCAL programs except that the heading is prefixed by an environment heading to supply an environment file specification. If any routine modules have been compiled in environments produced by extending earlier environment declarations, the main program module must be compiled in the last of the extended environments. Only a linear succession of environments may be used to compile the modules that make up a program.

Experience using separate compilation

These extensions have been implemented in the UW-PASCAL compiler [1,2,3] developed at the University of Wisconsin - Madison for Univac 1100 series machines. Experience using the extensions for further development of that compiler has shown them to be of considerable utility and to provide significant economic advantages. In particular, having the separate compilation features has made it possible to modify and test the compiler within a short time, even during periods of very heavy demand on system resources. Previously, recompilation of the compiler was practical only during off-peak hours.

The UW-PASCAL compiler has also been used by students in a compiler writing course, who made considerable use of the separate compilation features. These students found these extensions to be among the most useful aspects of the compiler. However, no significant reduction in total computing costs was observed in comparison to previous experience using ALGOL and SIMULA 67 compilers. The cost of keeping environment files and relocatable code generated by compilation of routine modules apparently offset the savings in compilation costs. The students' compilers were about 2000-3000 lines long or about 20% of the length of the UW-PASCAL compiler. Some work may be done to determine the program size at which separate compilation provides definite economic advantages in addition to its contributions to convenience and modularity.

References

[1] UW-PASCAL Refernece Manual, Madison Academic Computing Center, 1977.

[2] Charles N. Fischer and Richard J. LeBlanc, "A Diagnostic Compiler for the Programming Language PASCAL", USE Fall Conference Technical Papers, Lake Buena Vista Florida, October 1976.

[3] ------, "Efficient Implementation and Optimization of Run-time Checking in PASCAL", SIGPLAN Notices 12, 3, March 1977.

This paper describes work supported by the Madison Academic Computing Center of the University of Wisconsin - Madison.

(* Received 78/02/03 *)

Articles

What Are Pascal's Design Goals?

Robert D. Vavra
March 13, 1978

As a long-time reader of Pascal News (PN), I have enjoyed the many articles in which people have discussed various features which could be added to Pascal, but I have been unable to take much of the discussion seriously. In arguing for or against some particular feature, writers have rarely invoked Pascal's design goals in support of their arguments. Such failure to build a proper foundation for one's arguments might be acceptable in casual conversation, but not in a serious discussion.

If a discussion about a language feature is to be taken seriously (by me, at least), the writer must demonstrate that it is firmly based on Pascal's design goals. It is not enough to support a proposed feature by saying that it is easy to use or implement, nor to reject a proposed feature by saying that it is only a "favourite feature". A writer should weigh a proposed feature against each of Pascal's design goals by pointing out which goals favor it and which do not, and should discuss why the tradeoff is desirable.

All of this presupposes that Pascal's design goals are well-understood and generally accepted. In fact, Pascal's design goals are rarely mentioned in PN, so I suspect that they are not well-understood. Further, I think that much of the debate over various language features is really a debate over what Pascal's design goals should be. This article attempts to remedy this situation by summarizing what Wirth's design goals for Pascal originally were, and by starting a discussion of what Pascal's design goals should now be.

In both the original and revised reports [1,2], Wirth's stated design goals are suitably modest:

* To make available a language suitable to teach programming as a systematic discipline.
* To allow development of implementations which are both reliable and efficient.

In [3], Wirth stated the following design goals:

* To make available a notation in which the fundamental concepts and structures of programming are expressible in a systematic, precise, and appropriate way.
* To make available a notation which takes into account the various new insights concerning systematic methods of program development.
* To demonstrate that a language with a rich set of flexible data and program structuring facilities can be implemented by an efficient and moderately sized compiler.
* To demonstrate that the use of a machine-independent language with flexible data and program structures for the description of a compiler leads to an increase of its readability, verifiability and consequently its reliability, and that this gain need not be offset by any loss in efficiency.
* To gain more insight into the methods of organizing large programs and managing software projects.
* To obtain a home-made tool which can easily be adapted to other needs.

In [4], Wirth stated the following design goals:

* To permit clarity and rigour of description by using a small number of fundamental concepts, thereby making program verification easier.
* To have a wide range of applicability through proximity to actual computer structure, rather than through a host of features collected from various fields of usage.
* To promote both compile- and run-time efficiency by omitting features which require multi-pass compilation or elaborate run-time support.
* To promote reliability and efficiency of compilers by providing a language which is simply and regularly structured, thereby allowing the compilers to be simply and regularly structured.
* To promote machine independence (portability) by extending the definitional capabilities of the language to such a degree of generality that machine dependent entities (types and operations) may appear as special cases selectable by means of predefined names, and whose use presumably enhances the efficiency of programs executed on the particular system in which they are defined.

In [5], Hoare and Wirth summarize all these goals as:

* To make available a general purpose language efficiently implementable on many computers and sufficiently flexible to be able to serve in many areas of application.

Interested readers can find further details in each of the referenced papers.

As a starting point for a discussion of what Pascal's design goals should now be, I suggest the following list:

General purpose	Pascal should be usable in almost any application area or almost any computer system. There should be reasonably easy ways to manipulate both numeric and non-numeric data, to make use of pre-existing software subsystems (either in libraries or in the operating system), and to implement new general-purpose software subsystems in Pascal.
Introductory	Pascal should be usable by beginning programmers in an introductory programming course. It should be possible for them to write simple programs without needing detailed knowledge of large portions of the language, e.g. I/O.
Low level	Pascal's features should be close to those supported by current computer systems. Any higher-level abstractions (e.g. lists, strings, i/o) should be supported by writing new types and procedures in Pascal.

Portable	Software written in Pascal should be reasonably easy to move from one computer system to another. It should be easy for programmers to isolate implementation dependencies in a few places within programs. There should be a minimal number of situations in which the actions of a program are "undefined" or "defined by the implementation".
Small	Pascal should include a minimal number of fundamental concepts.
Systematic	Pascal should encourage the programmer think systematically by allowing him or her to concentrate on a small section of the program at a time. Pascal should minimize the number of special rules which must be learned, instead relying on general rules which apply in all situations.

Obviously, much more needs to be said about Pascal's design goals. For starters, important design goals may need to be added (e.g. reliability, replacement for fortran). Additionally, each of the design goals needs to be more fully explained (e.g. what does "thinking systematically" really mean). Finally, the implications of the entire set of design goals need to be explored (e.g. are the present extension mechanisms powerful enough to allow the language to be general purpose, low level, and small?).

I hope to find the time to write more about Pascal's design goals, and I encourage others (especially those who are proposing language features) to do the same. I look forward to a continuing dialogue in PN on this important topic.

References

1. N. Wirth, The Programming Language Pascal. Acta Informatica 1 (1971).
2. K. Jensen and N. Wirth, PASCAL User Manual and Report. Springer-Verlag (1974).
3. N. Wirth, The Desigr of a PASCAL Compiler. Software - Practice and Experience vol. 1 (1971).
4. N. Wirth, The Programming Language Pascal and its Design Criteria. Infotech State of the Art Report No. 7, High Level Languages (1972).
5. C. Hoare and N. Wirth, An Axiomatic Definition of the Programming Language PASCAL, Acta Informatica 2 (1973).

(* Received 78/03/20 *)

PASCAL ENVIRONMENT INTERFACE

T. Noodt
University of Oslo

Opening comments

I am at present working on a Pascal implementation for the Nord 10, running an interactively oriented operating system. (The Nord is a 16-bit Norwegian-made computer. It comes in two variants with 48-bit and 32-bit reals, respectively.) The Pascal Report does not say too much about how to interface a compiler to a computer system and its users. To further complicate matters, what it does say about this relates to a batch system, and is worthless or unusable in an interactive system.

I think that the design of the Pascal environment is fairly important, and that a certain unification would be of value. Below I have schetched a few thoughts about this, and also make some proposals.

Why bother

A language is often judged on the way a particular implementation interfaces to its environment, i.e.

1) what tools are available to a user for the construction, compilation, and execution of a program, and
2) what are the interfaces between the implementation and other systems on the computer (particularly the operating system).

Examples of such interfaces are: - What the available options are, and in what way they can be set or reset. - How a specific file is associated with a file name within a program.

Pascal implementors and fans have chosen to step off the FORTRAN, BASIC, and PL/I highways to enjoy the much nicer view from the Pascal path. The implementors being such rugged individualists, there probably are as many different Pascal environment interfaces as there are Pascal implementations. Implementors are inventing and re-inventing interface features, giving different names to the same feature, or implementing the same feature in a slightly different way.

Since all the big computer vendors soon will become Pascalers (do you doubt it?), the situation will become worse. In vendor A's Pascal implementation, all the extra-language "nice features" will be totally incompatible with those of vendor B. Goodbye portability.

So what

All Pascal implementations need a minimal environment interface, and often a broader interface is highly desirable. There is no reason why this interface should be a completely new one for every new implementation. A little effort can give a lot towards unification, at least in future implementations.

I would like to see a discussion about what such an interface ought to contain, ending up with someone making a recommendation list of features which seem necessary or highly desirable, with the heading:

> If you want to include a feature from the list below in your Pascal implementation, it is recommended that you adhere to the specifications stated for that feature.

To initiate a discussion about these matters, I will present my tentative list for the extra-language features in the Pascal implementation I am working on at present.

Options

Option	Effect	Default
L	List program	on
M	List symbolic object code (MAC)	off
Rn	Reals will occupy n words	3
Sn	Sets will occupy n words	8
T	Generate run-time test on indexing, sub-range assignments etc.	on
U	Convert all lower-case letters outside strings to upper case	off

Conditional compilation

The compiler will conditionally skip parts of the source text, depending on the value of flags which can be set by the user. Source lines containing commands to the compiler must have the character $ in position 1.

```
flag -> identifier
```

Command	Effect
$SET flag	flag gets value true
$RESET flag	flag gets value false
$IF flag	Include succeeding source lines if flag is true
$ELSE flag	End of $IF of same flag. Include succeeding lines if flag is false.
$END flag	End of $IF or $ELSE of same flag.

A flag which has not been assigned a value, will have the value false.

Multiple source files

The compiler command
    $INCLUDE filename
will include the content of that file at this point in the source text. The command may be used recursively.

Command processor

The command processor is a part of the compiler which accepts commands specifying parameters for a compilation.

SET flag
RESET flag
    Set and reset conditional compilation flags.

OPTIONS option-list
    Set or reset options according to option-list, which has the same syntax as if it appeared within a Pascal comment.

COMPILE sourcefile, listfile, objectfile
    listfile and objectfile are optional. The L-option is turned off if listfile is left out.

File access

The procedure OPEN enables association between a specific file and a Pascal file variable at runtime.

```
procedure OPEN(F: filetype; NAME: string;
               var STATUS: integer) . . .
```

F is associated with the file with name NAME. The file is opened, and status for this operation is left in STATUS. The parameters NAME and STATUS are optional. If NAME is not present, the system wil enquire the user to specify the file. If STATUS is not present and an error occurs, the job will be aborted if in batch mode or if NAME was specified, otherwise the user will be asked to respecify the file name.

```
procedure CLOSE(F: filetype) . . .
```

The file is closed, and F is disassociated from the file.

Standard procedures

The following standard procedures will be implemented.

```
procedure TIME(var HOUR, MIN, SEC: integer) . . .

procedure DATE(var YEAR, MONTH, DAY: integer) . . .

function TUSED: real . . .
(* gives accumulated CPU time in seconds *)
```

Interactive programs

Several difficult problems arise in the design and running of interactive programs. OPEN and CLOSE take care of run-time association of files.

Another nasty problem is that of reading data from a terminal. If the data does not have the correct syntax, it is of course not acceptable to abort the program. The CERN Pascal implementation has solved this problem in the following manner:

There is a standard procedure
SETINTERACTIVE
which when called, will make all error exits from READ save an error status instead of aborting the program. This status can be read with the function COMPLETION.

Closing comments

Some of the specifications above are vague, partly because I do not feel that a long, detailed document is necessary at this stage. Also, your own interpretation or evaluation of a feature may be as good, or better, than mine.

I invite criticism and comments on the features I have described above. You are also invited to add or subtract features.

However, I am not looking for an environment interface list which is as long as possible. Think ecologically, and do not let the environment pollute Pascal!

March 1978 (* Received 78/03/20 *)

SUBRANGES AND CONDITIONAL LOOPS

*Judy M. Bishop
Computer Science Division
University of the Witwatersrand
Johannesburg
2001 SOUTH AFRICA

The subrange facility in Pascal is an aid to runtime security for fixed boundary constructs such as counting (for) loops and array subscripts. The relevant types can be precisely and naturally defined and the compiler can minimise the amount of runtime checking required. However, an index which increases under program control, as in a conditional (while) loop, presents a problem. This note discusses the problem and presents a solution in terms of a naming convention.

THE PROBLEM

Consider the definitions

```
type  index = min .. max;                    [D1]

var   i : index;
```

and the conditional loop

```
i := min;
while (i <= max) and condition do            [D2]
begin
     (* something *)
     i := succ(i)
end;
```

If the condition remains true, an attempt will eventually be made to set i to succ(max) - a quite normal way of triggering the end of the loop. However, because i's type was precisely defined, succ(max) does not exist! Rewriting the loop with the tests at the end gives a similar error with respect to pred(min), i.e.

```
i := pred(min);                              [D3]
repeat  i := succ(i);
        (* something *)
until  (i = max) or condition;
```

Even without a compiler or program verifier run, it is obvious that these loops are inconsistant with the definitions [D1]. If the loop had only the test on i , then the for statement is the appropriate construct and the undefined nature of the final value is taken care of by the compiler. At least, it should be, but only the B6700 does this. According to Sale's Pascal Compatability Report, the various final values are

* Previously Judy M. Mullins
Computer Studies Group
Southampton University
ENGLAND.

undefined - B6700
max - CDC6000, Univac 1100
succ(max) - Dec-10, ICL1900, ICL2900

Do these options apply equally to succ? Like Sale, I think they should not: an implementation should have a detectable undefined value and succ(max) should yield it. This does not solve the original problem which was to allow a succ(max) for the purposes of controlling a loop.

Typically, this is achieved by weakening the type definition to one of

```
type  index = min .. succmax;                    [D4]
```
or
```
type  index = predmin .. max;
```
or
```
type  index = predmin .. succmax;
```

This compromise raises its own problem: the subscript type of a corresponding array declaration such as

```
var  a : array[index] of item;
```

expects the original index type as defined in [D1] and not one of [D4]'s extended ones.

A SOLUTION

In teaching programming to undergraduate students at Southampton, we made subranges "compulsory". (This can be done by omitting to mention the predefined types integer and real.) We were also blessed with a security-conscious compiler. In order to avoid untold "out of range" errors and general disillusionment in Pascal, we developed the following convention:

1. Subranges are defined over the genuine, natural range of elements, typically that which would be used as an array subscript. E.g.

```
type  months = 1 .. 12;                          [D5]
      money  = minint .. maxint;
var   balances : array[months] of money;
```

2. If an index is required for this subrange, its type is given the same name but prefixed by x- (for "extra") or z- (for "zero"). E.g.

```
const dec = 12;                                  [D6]
type  xindex = min .. succmax
      xmonths = 1 .. 13;
      zmonths = 0 .. dec;
```

3. For an ennumerated type, the extra or zero element in the list is named according to the type with the appropriate prefix. E.g.

```
type  xmonths = (jan, feb, mar, apr, may, jun,   [D7]
        jul, aug, sep, oct, nov, dec, extramonth);
      months = jan  .. dec;
```

EXAMPLE

```
function deficit (since : months) : boolean;
  var  m : xmonths;  negative : boolean;
  begin
    negative := false;   m := since;
    while not negative and (m <= dec) do
    begin
      negative := balances[m] < 0;
      m := succ(m)
    end;
    deficit := negative
  end; (* deficit *)
```

Deficit will work with the definitions in [D5] plus either those in [D6] or [D7]. It could be called by

```
if deficit ((*since the*) 6 (*th month*))then ...
```
or by
```
if deficit ((*in*) dec))then ...
```

This is made possible by keeping max (in this case dec) symbolic and by making use of succ instead of +1 . In all compilers known to me, succ(m) is equivalent to m+1 , that is, it does not invoke a function call.

COMMENTS

1. Single letter prefixes do, admittedly, hinder readability, but were chosen so as to impinge the least on the length of the original name.
2. A single letter was not considered necessary for the dummy ennumerated element since clashes here would be few.
3. The possibility exists for relating a subrange with its extensions in a more formal way. Language designers and pedagogues may care to investigate this.

ACKNOWLEDGEMENTS

Mike Rees, Martin Must and others in the Computer Studies Coffee Room helped convince me that the easy way out i.e. declare one type as the union of all possible variations, should be avoided. The students of Computer Studies II, 1977, proved our solution to be both workable and effective.

The Pascal Compatability Report can be obtained from

Professor A.H.J. Sale
Information Science
University of Tasmania
Box 252C
Hobart
Tasmania 7001.

(* Received 78/02/28 *)

John Nagle
Information Systems Design
3205 Coronado Drive
Santa Clara, CA 95051

(408)-249-8100

Since quite a number of extensions to Pascal have been proposed, I thought that it would be desirable to propose a few deletions to keep the size of the language down. With the goal in mind of keeping Pascal a simple, elegant, and useful language requiring a minimum of run-time machinery, I propose a few simple changes in the direction of simplicity.

1. Get rid of GOTO and LABEL

Need I say more?

2. Get rid of the FORWARD declaration

The FORWARD declaration is a means by which a simple one-pass compiler can find out about calls to procedures not yet defined. Since the compiler can perfectly well figure this out under its own power by capturing the procedure definition at its first call, we should not need to saddle the programmer with this responsibility. At the end of the compilation we can list all undefined procedures. This implementation has the additional advantage that the compiler will not complain so much when we compile programs which are not yet complete, yet will even crosscheck the calls of unwritten procedures for compatibility. This is in keeping with the 'top-down' coding philosophy.

3. Get rid of the 'FOR loop variable problem'

When we get rid of GOTO, we simplify the semantics of the FOR loop by guaranteeing that the index variable of the FOR is always meaningless outside of the loop. Given this simplification, why not simply declare the FOR index variable automatically at the FOR statement AS LOCAL TO THE LOOP. The programmer need not declare it at all, nor should he, since it has no meaning outside the loop. The type of the variable is implicit in the type of the value assigned to it in the FOR statement. This gets rid of the ambiguity associated with the present definition, simplifies

register allocation in the compiler, and is fully compatible with existing correct programs.

4. Get rid of the CASE statement ambiguity

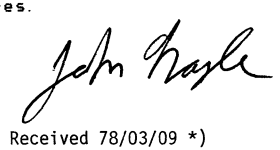
The action of the CASE statement when the CASE selector is out of range is 'undefined'. In one implementation, nothing happens in such cases. In another, the error is always detected and fatal. One of these two actions should be made standard. I would go for making it fatal (and perhaps installing a case name of 'other' to allow the program to capture these cases).

5. Get rid of 'column 1 forms control'

I am aware that this is not a part of the language but of an implementation. However, it is a very undesirable extension as it builds an implementation dependence into programs unnecessarily. The procedure 'page(<file>)' is defined as indicating page ejection. I suggest we use it. Blank lines may be inserted by calls to WRITELN without parameters. If it is desirable to handle the printing of blank lines in some special way to improve printing speed, the WRITE routine should take care of thes detail by itself.

6. Get rid of the proposed 'formatted read'

Various parties have noted the very real need for some means of dealing with fixed-format input. Extending READ by adding a formatting facility has been proposed. I see this as an unnecessarily complex approach to the problem. The problem is that of dealing with fixed-format input records. We have a very nice record facility in Pascal; we simply cannot read records easily. I propose that it be permitted to READ any structure of fixed length all of whose base types are CHAR. This form of READ should not be permitted to cross record boundaries (EOLN=TRUE) and the record being read into should be filled with an agreed-upon fill character (probably space, but some persons may want to use an ASCII NUL). This facility also provides the ability to read strings (arrays of CHAR) which is currently an annoying lack. Conversion of numeric fields to values of type INTEGER and REAL can be handled by suitable procedures. It would be desirable to define and standardize such procedures.



(* Received 78/03/09 *)

Open Forum for Members

The University of Tasmania

Postal Address: Box 252C, G.P.O., Hobart, Tasmania, Australia 7001

Telephone: 23 0561. Cables 'Tasuni' Telex: 58150 UNTAS

Mr. A. Mickel,
Editor, PASCAL News,
Computing Centre,
University of Minnesota,
Minneapolis, Minnesota
U.S.A.

18th January, 1978

Dear Sir,

PASCAL in Australia

May I clarify a few points that have arisen from my letters in #9/10? Firstly the cost of Australian subscription has been criticised for its size ($A10.00) against the US fee ($US4.00). Our original budgetting was based on the following quoted figures:

Cost/copy (based on issue #8)	$2.80	
Postage/copy (in Australia)	.70	(higher in NZ, Malaysia)
	3.50	

I suspected the cost/copy to be high, and we struck a rate of $2.50/copy as reasonable, giving a subscription of $10 (4 copies per year). This makes no allowance for othermail (reminders), the necessity to overprint to fill back-orders and possible surplus stock, and higher postage to overseas countries. It must be remembered that the print run in Australia is small, and the postage rates are simply fiendish.

Subsequently, issues 9/10 have been coalesced, and by the time next year comes we shall know better what it really costs. I hope it is less. Preliminary revisions of the costs have shown cost/copy at about $1.00 and postage in Australia has dropped to 60c, making a fee of $7.00 possible. I'll give you a figure for 1978/9 after we have processed #11.

The second point I have been taken to task over is a statement of mine that our first-year course switch to PASCAL was "a first for reactionary Australia". Since this seems to have been misunderstood, PASCAL News readers may be interested to know that PASCAL has been in use in CDC universities for some time, notably the Universities of Adelaide, Sydney, New South Wales (and Melbourne). In some cases as an introductory language, in others as a later language.

However, in none of these, nor in any of the other universities that I know, has the combination of a full undergraduate first-year course combined with the use of PASCAL as a first language. However, I don't know everything that goes on in all of the 20 universities spread over the continent, so I apologize if I'm wrong. My clear impression is that FORTRAN still dominates the Australian scene, with COBOL and BASIC hovering around as well.

Yours sincerely,

Arthur Sale

Arthur Sale,
Professor of Information Science.

Tema

Società per azioni con sede in Roma
Capitale L. 500.000.000 i. v.
Trib. di Roma Reg. soc. n. 1450/72
CCIAA Roma 374468 Bologna 211702

Direzione e uffici:
40122 Bologna / via Marconi 29/1
telefono 267285 (5 linee)

PASCAL USER'S GROUP
c/o Andy Mickel
UCC: 227 Exp. Engr.
University of Minnesota
Minneapolis, MN 55455
U.S.A.

riferimenti da citare nella risposta

026/77/0343/GS/ls

Bologna 11/11/1977

Dear Mr. Mickel,

I wish to submit a few considerations to the Forum of PUG members and PASCAL implementers:

Prologue:

"The new language PASCAL is my own favourite programming language, but only time will tell if it can fall beneath its academic roots"
"Teaching the fatal desease" - R. Holt - SIGPLAN Notice (May 1973).
I work in a Software Consulting Agency that deals with the following main subjects:

- mathematical models;
- management information systems;
- basic software and process control.

Our applications must work on user's defined environments so we design and code programs for different computers. I emphasize: we design and code, we do not teach programming.

Aim:

We think that PASCAL is a very good language (we use it, as design tool, since 1972, and as coding language since 1975) but we also think that a few PASCAL features ought to be improved and a few ones ought to be introduced, on our opinion, in order to provide a good use in commercial applications or, that is almost the same, in day-to-day programming. "You Know that 80% of commercial application programming is done in COBOL" and there is not an "other language that is a practical alternative for commercial work" - Call for Paper, First European Conference on pragmatic programming & sensible software - 1978, J.Weinberg - With this letter we wish add ourselves to the chorus of other letters on the same matter that we have read on Pascal Newsletters.

Improvements:

1 - Enumerated scalar types: many commercial applications, and not only, use very much this kind of type, and their subranges, but variables of these types are incommunicable via standard Input/Output and external files. In order to have a clear programming is quite useless their inner representation, it would be better their identifiers.
A possibile solution is to use their names (with common restrinctions) as strings in I/O, this is implementable if we do a pre-processor of the compiling stage that is not very heavy compared with the time and space used by other common compilers.

2 - Strings: we like the way of defining a string - packed array [1..n] of char - but, in practical programming, strings must be of varying

lenght - with 'n' as maximum lenght - With formatted input, it would be better to read the complete string in the same way as it is possible to write it (also without format). Pack and Unpack are useful procedures to process a character in a string but they are not sufficient to use strings in programs, one must provide the complete set of operators.
We think that a good way of re-design this data type is to introduce string (n) as base data type.

3 - Formatted input: we have read the proposed solutions of the quarrel in Pascal Newsletters, but is not a practical day-to-day solution. On the other without formatted input is difficult to use PASCAL in commercial applications because a blank in a (commercial) card is as important as a character.

4 - Case statement: we enphasize the utility to have:

```
case expression of
     value 1:
        :
     value n:
otherwise:
```

in addition to actual <case statement>

5 - Interlanguage communication: the "slogan" must be: "we have to manage the transition between old nad new systems". But more than this often it happens that one have to use existing packages (data-base, linear-programming) but not also, many times we have to interface a system that, for many reasons, must not be re-done. So it is very important, if we want to introduce the PASCAL Language, to have a construct to manage communication with FORTRAN, COBOL, PL/1.

Epilogue

We think it is very good to have a:

"- sparse, simple language;
- general purpose language;
- vehicle for portable software;
- tool for systematic programming;
- etc. "

but also that, if software workers do not use it, it means that we have not proposed a real alternative "to dinosaur languages" and to actual programming style.

Sincerely

(Dr. Giuseppe Selva)

Open Forum for Members

J. E. POURNELLE AND ASSOCIATES
12051 LAUREL TERRACE STUDIO CITY, CALIFORNIA 91604 (213) 762-2256

2 Feb 1978

Dear Mr. Mickel,

I have located a PRIMER of PASCAL and although I haven't obtained it yet--the bookstore is very slow-- it promises to solve most of my bibliographic problems.

In your package with the back issues of the newsletter you noted that you have a good PASCAL for a Z-80 system with disks.

My system is a Cromemco Z-2 with 48 K RAM and 16 K ROM, iCom disks with FDOS-3; CPM is also available. Also have TARBEL cassette interface.

I would be very interested in obtaining a source code for PASCAL that I could get running. I do not expect to get this for nothing, and I am prepared to pay reasonable costs and fees; but I do need to know from whom I can obtain it.

I enclose (1) a stamped self-addressed envelope for your convenience in replying, and (2) a check for $10 as a donation to the user's group, in the hopes that will provide an incentive for you to take a moment to reply.

As the former President of a writer's association I know very well how volunteer and unpaid work eats up one's professional time and I recall from my grad student days just how little time is available; still, I do hope you'll be able to give me the information on how to acquire PASCAL.

From all Ihear, including from friends out at JPL who work in programming the MARINER and PIONEER probes, PASCAL is very powerful and indeed probably better than some of the much-touted and much better known languages. I am about to write a whole mess of software and I would like to have the option of getting that done in PASCAL rather than FORTRAN or any of the 3 BASICS I have.

Thank you again,

Jerry Pournelle

Joe Celko

Box 11023
Atlanta, GA
30310
USA

Date: 24th Feb 78

To: Pascal News
University Computer Center
University of Minnesota
Mineeapolis, MN
55455

Gentlemen:

RE: Mr. R. A. Fraley's piece in the last Newsletter on suggested extensions.

I miss dynamic arrays, being an Algol, and I'd love to have real honest strings, too.

However, I don't miss Common, and I don't think we need to have Modules, either.

Common can be dismissed as a horror that allowed each subroutine in FORTRAN to cut up the shared data as it wished (thus Subroutine A was working on four integer, that Subroutine B thought were two Reals, etc.). It was as big a burden as a long parameter list, and it should die with FORTRAN.

There is a nice feature in COBOL, PL/I and some other languages for COPY or INCLUDE verbs that allow the placement of text into the program from other files.

Since this is an operating system-compile time interface we could do the include at the source code level, the intermediate code level, or the binary code level. I favor the idea of doing it as source code, since optimizing work can be done better with constants being plugged in by actual user calls (there was a Student paper in CACM with some timing studies on this technique last year, that would be of interest), and since we must already have a good text file system.

This would mean writting an extension to the compiler you now have that would recognize

⟨include statement⟩::= include ⟨external file id⟩

convert it into a comment, and append the proper file(s).

A library would be defined as a procedure or function, a group of procedures or functions, or a group of procedures or functions preceded by a group of declarations. A crippled compiler could check to see that no executable statements were on file. By having the declarations in front of the procedures or functions we would get the shared variables via the existing Global conventions of Pascal, and it would cost us only a few extra lines in a compiler. Portability would not be affected. Optimizing would be possible (for example, deleting items that are not called by the program).

Yours,

Joe Celko

LEIBNIZ-RECHENZENTRUM
DER BAYERISCHEN AKADEMIE DER WISSENSCHAFTEN
BARER STRASSE 21 D-8000 MÜNCHEN 2

Pascal User's Group
c/o Andy Mickel
UCC: 227 Exp. Engr.
University of Minnesota
Minneapolis, MN 55455
(612) 376-7290

München, den 23.02.1978/HW-la
Telefon (089) 21 05/84 84
21 05/
Telex: 05/24 634

Dear Andy,

enclosed you find a list of wishes of an enthusiastic PASCAL-user. I have condensed it from discussions with some of my collegues who tried PASCAL for various problems.
These wishes are not "filtered" by implementation considerations, but are propositions to make a very attractive language more usable.

As I am preparing a PASCAL course for the users of our computing center, I would be very glad to receive a short note from you indicating the new features of release 3 of the compiler and the approximate date of distribution.

Thank you in advance.

Yours sincerely,

H. Weber

(Hellmut Weber)

A PASCAL-user's viewpoint:

1. Extensions which I consider as necessary

1.1 Dynamic array bounds, for example as describing by WIRTH and further explained by CONDICT in SIGPLAN letters.
(It is ridiculous to have to write a separate inner-product procedure for every vector length. The development of procedure libraries would be drastically simplified.)

1.2 Procedures which are parameters of other procedures must permit var-parameters
(You don't have always one single value as result!)

1.3 Formated i/o as an alternative for the existing READ and WRITE procedures.
(It seems to be the only posibility on convince people who write production programs, and until now write them in FORTRAN or COBOL, whether you like it or not.)

1.4 Structured Constants,
especially constant arrays
(Are needed if one wants to write library routines which are to be called very often by one program, e.g. Serées expansions).

2. Extensions which would make programming in PASCAL much more comfortable:

2.1 Break_character

2.2 All characters of a identifier should be significant.
(The possibility of suggestive and self-documenting names is severly by the 8-char-rule.
Regard

Numberofpossibilities and
Numberofrealizations

and compare it with

Number_of_possibilities and
Number_of_realizations

Even the CDC character set has still unused characters.)

2.3 Alphanumeric labels
(can indicate the condition of their use:
error_in_input_data:)

2.4 More standard functions
as y^x, $\log_{10}x$

3. "Extensions" to be included in a sort of "standard" implement tation:

3.1 Possibilities for interactive use!

3.2 Extended file access:

3.2.1 Possibility to append an element to a file after having read the file.

3.2.2 Random access
(KNUDSEN (ETH) described a sort of minimal version in PN.)

4. Features every implementation should include:

4.1 Simple possibility to generatelibrary routines
(e.g. Compiler-option which causes the compiler to assume the dummy main program and to suppress the code generation for it. Non-standard type declarations needed for the library routine had to be placed after the "switching on" of this compiler option.)

4.2 Possibility to compile several programs (or a program and separate libray routines à la 4.1) in one compiler run.

4.3 Debugging aids:
(there are no upper limits to the ingenuity of the compiler writer, but there should be at least a dump showing all variables, of structured variables I would accept the bit pattern. See for example the description of H.H. NAGEL in PN#4).

4.4 Compiler identification and statistics (Important for documentation and not much work for the implementor.)

5. Especially for CDC-users:
Alternative characters for the important but dangerous characters ':' and '^' .

Release 2 allows '%' instead of ':' (as proposed by VIM) but I didn't see the fact documented).

A COMMENTARY ON PASCAL NEWS No. 11

A.H.J. Sale
University of Tasmania.

This report is a commentary on the contents of issue 11 of News; it may be of interest to PASCALlers in assessing where PASCAL is going and the value of the NEWS.

Page 4: David Barron's proposal for algorithms is excellent! Even if inter-program linkages aren't provided in a PASCAL version, source inclusion always is. I offer any help I can give regarding portability (alas, I am no numerical analyst). Can I urge others to help too?

Pages 8-13: possibly news, abstracts, etc, are the most useful part of News that appears regularly. For me anyway.

Pages 33-34: a good summary of a serious 'compatibility' problem. For a long time I have liked SRTCCO (in the author's terms), but this is something a standard must resolve before I make a change. There are problems in SRTCCO in letting adjustable arrays in, aren't there? It would be interesting to know why the change to SRTCC1 was made in later CDC compilers.

Pages 34-40: A contrast between a tale of woe and rosy future. It is good to see such a lot going on. Anyway, sometime soon it may be possible to present all these 'standard' compilers with a large suite of validation programs to exercise the claims a bit. It's being worked on in the UK by Brian Wichmann, and here by me.

Pages 40-41: Suggestions for compilers and my reactions: (1) I suppose ok, the comment idea is best, but you'd better look out if I find anyone allowing nested comments. (2) Yes; but possibly a good separate cross-referencer would do as well. (3) Must have this; we do. (4) I suppose so; it hadn't occured to me as a problem as I'd rather remove the packed symbols. (5) Not that we have it, but it is obvious that Kempton isn't a teacher of programming! Do whatever you want if you've got your own version about setting such defaults. They're nothing to do with PASCAL. (6) No comment needed. (7) God help us, why? Cannot input forms be identical to output forms and those in the language too? I regard things like 1. or .4 as quite abnormal; most scientists don't write them that way anyway. (8) What's wrong with Wirth's suggestion of writeln? (9) Some compilers do this now. (10) Possibly pre-defining more constants is a bit of a sledgehammer; a prologue to portable programs can invite users to change a const part ot suit their machine.

Pages 41-48: The set of Fraley extensions is too large to comment on. Most of them fall far, far beyond any current ideas of standardization of PASCAL which must include its warts as well as its beauty. I'll only comment on one semantic ambiguity at the end. (1) This is a violation of the principle of environmental independence of procedures; probably an oversight in PASCAL-P. The ICL and Tasmania B6700 do it properly; the compiler is marginally simpler!

Pages 48-53: What a welter of desire to change PASCAL! Can I re-iterate what Andy said: basic PASCAL is not up for grabs so that everyone can add their favourite feature. The Revised Report needs tidying up, yes, after all it was written for communication not as a standard. (Writing a standard calls for the same intellectual effort as proving programs correct.) But not wholesale revision. Only a very few 'agreed extensions' seem worthy of writing down; one that appeared in these pages was allowing of a wider class of type-changing procedures. This might be worth it if it stops people misusing variant records for the purpose (a very difficult thing to detect).

Pages 66-80: Standards. It seems to me now that we could tidy up the loose ends if only all these efforts don't get in each other's way. It'd be a disaster if everyone started modifying compilers to fit into different views. I'm adopting the policy of still fixing bugs and things the current Report is quite clear on, but waiting a bit on any extensions until I see something all are agreed on. It is only a pity this effort is two years or so too late to forestall the variety. I'm delighted to see the bugs in PASCAL-P being brought out; I find I can detect the ancestry of compilers by the results they give to some test programs. And of course errors shouldn't be propagated.

Pages 83-103: Implementation notes: usually not comment-worthy. I feel I must say something about the lousy decisions people are making in the lexical area. Why on earth substitute # for <>, or abbreviate procedure to proc? It is just pointless. Alternately, if the compiler is meant to be used in-house alone, why publicise it in News. Still, enough of gripes. I applaud the setting up of the ICL clearing house. PASCAL now has a canonic implementation for CDC Cybers, Decsystem 10s, UNIVAC 110s, and ICL gear. The IBM and PDP-11 situations are the most worrying, though I think the rumors I get suggest that the AAEC version (see p94) of IBM is quite good (if you keep away from the extensions).

Overview: If this sounded critical in places, please take in it context. I think News is growing up, and the standard of news and contribution is getting much better. I'm glad we are going to have an 'Algorithms Section', and I hope we'll have critical contributions on what's published there. A lot of the portable software I send off for isn't, and requires work to repair the defects which should have been done by the designer. On extensions, I'd really suggest that everyone stop writing them into News since we have too much already. Checklists of troublespots would be welcome, but no new ideas please. For my piece of mind alone, if not Andy's.

Thanks

It is now two years since Andy took up the task of News editor, and just so he doesn't despair and think the task thankless, I'd like to express all Pascallers thanks to you Andy, for the tremendous job you've done. I don't know how you find the time, really I don't. Anyway, thanks.

Arthur Sale

THE UNIVERSITY OF KANSAS / LAWRENCE, KANSAS 66045

Department of Computer Science
18 Strong Hall
913 864-4482

27 February 1978

Pascal User's Group
c/o Andy Mickel
University Computer Center: 227 EX
208 SE Union Street
University of Minnesota
Minneapolis, MN 55455

Dear Andy,

Shame on you! I expect that the only reason you included R. A. Fraley's papers on "Suggested Extensions to Pascal," in PN #11, was to scare the living begabbers out of us. Congratulations. You were terrifyingly successful. By the end of the second paper I was quaking in my chair (actually, I was expecting to see such things as proposals for long and short int, packed decimal, on condition constructs, sort, and all of the rest of Fortran (ugh!), COBOL (ugh! ugh!), and PL/I (ugh! ugh! ugh!) to be included. Tighten your belt and stand by your guns, I'm sure there's more to come.

Sincerely,

Greg

Gregory F. Wetzel
Research Assistant

eric small & associates, inc.
consultants in broadcast technology
680 beach street, suite 365, san francisco, california 94109 telephone (415) 441-0666

March 6, 1978

Pascal User's Group
C/O Andy Mickel
University Computer Center:
227EX
208 SE Union St.
Univ. of Minn.
Minn, MN. 55455

Dear Andy,

We are looking for a PASCAL programer to join our merry band.

The ideal candidate will be a recent graduate. He or she will have had a lot of PASCAL experience. Exposure to broadcasting operations or engineering would be very desirable, possibly college radio.

Seven people comprise ESA. Dave Rowland, one of the implementors of ESA pascal, is our lead software engineer. We are developing a line of LSI-11 based products for the radio and TV braodcast market.

Sincerely,

Eric Small
President

TO: Andy Mickel
Editor,
PASCAL NEWS

March 8, 1978

I would like to address certain misconceptions which may have been generated by Professor Arthur Sale's letter in PN#11 (page 75), titled "Unimplementable Features -- Warning".

Professor Sale expressed concern about extensions which some PASCAL implementors have added to their implementations. He claims that these extensions are "not implementable on the Burroughs B6700 system and possibly on other computers." Not only is this claim false in the general case (we are, after all, dealing with computers as powerful as a Turing machine), but there exist relatively simple implementations of two of the three extensions which he uses as examples.

(1) "Passing pointer values as addresses, even in-stack"

Professor Sale's observations in this case are correct, if over stated. It would be inconvenient to implement pointers to non-heap variables on a B6700 system while realizing the advantages of its architecture.

However, there is a more important reason why this extension is not advisable -- the "up-level pointer problem" (and the related "dangling reference problem"). This reason alone, apart from any implementation difficulties, is sufficient to reject such an extension to PASCAL.

(2) "Returning function values of all kinds except files"

Professor Sale has misstated the facts about the B6700 RETN operator. He claims that only operand values (with zero tag) can be returned without causing an "Invalid Operand" interrupt. In fact, however, the B6700 will allow a word with any tag to be returned from a function via the RETN operator. In particular, a data descriptor can be returned quite easily by this method. However, there are other reasons (notably, software conventions) why this approach is not a wise implementation choice.

There is, however, a very clean and simple way of implementing a function return of a descriptor-based type. The most recent release of the B6700 Extended ALGOL compiler (version III.0) includes the new data type "STRING". Values of type STRING can be returned from a typed procedure. The implementation technique for this descriptor-based type may easily be adapted to arrays or records. The B6700 PL/I compiler has used a similar technique for several years, although the new ALGOL STRING implementation is somewhat simpler.

Besides the availability of a suitable implementation, returning such data values from functions seems to be a reasonable language feature. After all, records and arrays may be assigned values and may be passed as parameters by value. There is no fundamental difference between these uses of data values and their use as the returned type of a function.

(3) "Allowing pointers to file-types and the use of new(file)"

While I have argued on grounds independent of implementation that pointers should not be allowed to reference non-heap variables and that records and arrays should be allowed to be returned from a function, I will not take sides on the issue of files in the heap. I will only point out that the restriction that file descriptors must reside in the program stack is a software convention, not a hardware restriction. Furthermore, if the heap is marked as a "dope vector", no more than a few lines of changes to the operating system are required to make files whose descriptor resides in the heap behave normally with respect to being properly closed and deallocated at end-of-task.

Bob Jardine

Bob Jardine
Mission Viejo, California

LINCOLN COMPUTING FACILITY
225 NEBRASKA HALL
LINCOLN, NEBRASKA 68508

THE UNIVERSITY OF NEBRASKA
COMPUTER NETWORK

TELEPHONE (402) 472-3701

March 10, 1978

Dear Mr. Mickel:

I maintain PASCAL at the University of Nebraska in Lincoln, and have decided I need my very own personal copy of PASCAL NEWS, so I am enclosing a check for membership for 1977-78.

Looking at the different mutually incompatible versions of PASCAL serves to emphasize one point strongly--that PASCAL needs standardized extensibility. To each his own (idiosyncracies, environments, needs, etc.). If extension mechanisms are developed for PASCAL, as they have for ALGOL 68, a standard PASCAL implementation could be defined as one whose correct programs ran correctly (even if inefficiently) on another standard PASCAL when enclosed by a prelude or environment extension/redefinition block and called as a procedure. Such mechanisms should also enable character set redefinition, reserved word redefinition, etc.

Ideas anyone?

Sincerely,

K.V.S. Bhaskar
Academic Computing Services

Oslo, March 15, 1978

Dear Andy,

I am presently working on a new implementation of Pascal for the Nord-10. The implementation is based on the TRUNC compiler. I would like to communicate with others who have done or plan to do likewise.

Somehow I must interface the compiler to the environment in which it is to be used. This means that I have to invent a set of "features" to go with the implementation - features which already have been invented a lot of times by others.

Trying to turn some of the stones in this field started a train of thoughts, some of which are refelcted in the enclosed article. My hope is that the article will provoke a discussion about how the field ought to be plowed.

Yours sincerely,

Terje Noodt

Tektronix
COMMITTED TO EXCELLENCE

Tektronix, Inc.
P.O. Box 500
Beaverton, Oregon 97077

Phone: (503) 644-0161
TWX: 910-467-8708

March 16, 1978

Pascal News
Andy Mickel
University Computer Center: 227Ex
208 S. E. Union Street
University of Minnesota
Minneapolis, MN 55455

Dear Mr. Mickel:

Interest in Pascal has been growing among software engineers at Tektronix for some time. Within Tektronix, wide use is made of many dialects of Pascal for various programming purposes. However, Tektronix does not currently have any products incorporating Pascal programming capabilities. Tektronix will not offer such products unless and until we can do so within our requirements for utility and quality.

Because of the many (somewhat incompatible) dialects of Pascal currently in use and the possibility of Pascal's application to some future products, Tektronix has recently engaged in a study of Pascal extensions. The results of that study are not yet available, but will be made available to the summer workshop proposed by Ken Bowles in Pascal News #11. We expect to offer these results for publication in Pascal News #13.

Sincerely,

TEKTRONIX, INC.

Don Terwilliger

Dr. Don Terwilliger
Manager, Computer Research
Tektronix Laboratories

DT:jlk

State of Florida
Department Of
HIGHWAY SAFETY AND MOTOR VEHICLES
NEIL KIRKMAN BUILDING TALLAHASSEE 32304

RALPH DAVIS
EXECUTIVE DIRECTOR

COL. J. ELDRIGE BEACH, DIRECTOR
DIVISION OF FLORIDA HIGHWAY PATROL
JOHN D. CALVIN, DIRECTOR
DIVISION OF MOTOR VEHICLES
CLAY W. KEITH, DIRECTOR
DIVISION OF DRIVER LICENSES
AUDRY CARTER, JR., DIRECTOR
DIVISION OF ADMINISTRATIVE SERVICES

March 16, 1978

Mr. Andy Mickel
Editor, Pascal News
University of Minnesota
University Computing Center
227 Experimental Engineering Building
Minneapolis, Minnesota 55455

Dear Andy,

It is such a temptation to detail all my opinions, just because they are mine, even though others have already said the same. Luckily I am pressed for time and the resistance is relatively easy; I think each of the following is in some way new.

1. I enjoy reading Arthur Sale's prolific comments - he is one who often states my opinions. I do object to one aspect of his contributions: his consistent referral to the University of Tasmania's Pascal compiler as "The B6700 compiler". I use a different Pascal compiler on the B6700 (produced by Kenneth Bowles' group at UCSD), know of still another, and hear rumors of one or two others. I expect the Tasmania compiler is a very good one, but it is not the only one.

2. I agree with Arthur Sale's conclusions that certain non-standard features should be avoided. I do not agree with his reasons. These features are not unimplementable on the B6700, as he claims, or the difficulties as horrible as he puts forth. The proper reasons for not implementing these features deal with the language itself, not with a particular implementation. The difficulties encountered on the B6700 are most valuable when used to give insight on future machine design.

3. Formal procedures and functions should be completely specified; the lack thereof is merely a bad holdover from ALGOL60. (I suspect the lack of specification is one reason so many compiler writers omit this feature.) Declaration of procedure types as suggested by George Richmond (PN#8 p 13) leads to such questions as

    Are procedure variables allowed?
    Should procedures be declared in the VAR section?
    Why does a procedure have an initial value (the body) when other variables not?

ad nauseum. These problems should be left to ALGOL68. Therefore the specification should be in-line only. To do so, change the definition of <formal parameter section> to read

::= <parameter group> |
var <parameter group> |
<procedure heading> |
<function heading>

This generates an extra semicolon, so the definitions of procedure and function declaration and heading must be altered to take this into account. This affects pp 112,155 and 159 of the PUM&R; also affected is the program on p 79. Restriction 2 on p 83 can then be dropped. This usage assumes that the type compatibility checking is, in the terminology of Desjardins (PN #11 p 33), SRTCC1; otherwise no two procedures would ever be compatible as types.

4. Standard methods for data transformation are needed, particularly for conversion between character and integer or real; these methods may be functions or procedures or statements. This issue has been much discussed under the guise of formatted I/O. I believe that embedding the transformation of activity into "formatted I/O" unnecessarily complicates the I/O part of the language and unnecessarily restricts the conversion features.

I cannot let Barron & Mullins' argument (PN # p 8) pass unnoticed. Packed data is necessary at times, though formatted I/O is not. My agency handles about 10000 title activity transactions per day, with about 30 fields each.

10000 transactions/day x 30 separators/transactions
= 30000 keystrokes/day

≅ 30 key entry stations&operators

≅ $30,000/month

= $360,000/year to use separators.

5. There have been many proposals for extending Pascal's I/O, but usually with no mention of the overall I/O facility which results. Pascal I/O needs improvement, but suggestions should be limited to proposals for a simple, consistent and complete I/O facility, never for isolated features.

Sincerely,

C Edward Reid

C. EDWARD REID
Kirkman Data Center

CER:jem

Pascal User´s Group, c/o Andy Mickel
University Computer Center: 227 EX
208 SE Union Street
University of Minnesota
Minneapolis, MN 55455 USA

SIEMENS

Ihre Zeichen und Ihre Nachricht vom

Unsere Zeichen
Re

München, Otto-Hahn-Ring 6
2.1.1978

Dear Andy,

thanks for Pascal News 9 and 10. We finally have completed our work on a version 0 of our portable Pascal-System (cross-version). The first implementation is now on an MDS800 (Intel 8080 microprocessor), using the ISIS II operating system.

Just some short notes about the history of the project. We decided to implement a portable programming system for the different computers in the Siemens product-field, i.e. 32bit, 16bit and 8bit machines. We came upon Pascal after an implementation of a self-invented language, whose syntax and parts of its data-concept were mainly influenced by Pascal. This language was but experimental and could not be used as base for a programming system.

Our main goals were portability of user-programs as well as portability of the system itself. We think we now have reached both, the first by implementing Standard-Pascal and no dialect at all, the second by using just high-level languages for implementation (mostly Pascal, of course).

Our plans for the near future are a resident version and system-dependent features as a code-generator, generating some form of threaded code. Also a machine-independent dialogue-system has to be developed.

Please do note our new address.

Sincerely

Werner Remmele

Adresse:
Siemens AG, Zentralbereich Technik
Zentrale Forschung und Entwicklung
Forschungslaboratorien

Bearbeiter
Remmele

(089)
6782- 4622
Vermittlung 6782-1

528384

Postfach 83 27 29, D-8000 München 83

SIEMENS AKTIENGESELLSCHAFT
Zentrale Forschung und Entwicklung · Forschungslaboratorien Leitung: Prof. Dr. Walter Heywang

Andy Mickel
Editor, Pascal News

Dear Andy:

Enclosed is my membership form for PUG. You're doing a great job. Keep up the good work!

PASCAL is indeed catching on within Computer Science departments, but, despite the numerous examples mentioned in PN, most other groups I have seen are reluctant to use PASCAL in place of more available and familiar tools. In particular, PASCAL will never replace BASIC or FORTRAN as long as these languages provide features that are sorely lacking in PASCAL. In particular:

FORTRAN has static variables, external compilations, initialization of variables (in particular, arrays), procedures with flexible sized array parameters, STOP and RETURN statements, formatted input with error detection under user control, and large libraries of applications packages. PASCAL does not. In particular, complex numbers would not be missed if a standard subroutine package were available.

BASIC is, despite its lack of power, an extremely friendly language to beginners. Most idiosyncracies are hidden from the user - only one numeric data type, arbitrary length character strings, general FOR loop. Interactive programs are natural. The notion of "One line = one statement" is much easier on the beginner than PASCAL's relatively complicated set of syntax rules. BASIC's editor is very easy to learn.

I have many gripes about PASCAL - mostly concerning features that have been left off. Despite it's goal of being systematic, PASCAL has formatted output (but no formatted input.) It can read and write integers, reals, and characters, (but not enumerated types or records, and to add to the confusion, you can write booleans and packed arrays of characters but not read them!) PASCAL has constants of type integer, real, boolean, character, and set, (but no const declarations of type set, and no record or array constants or constructors.) Functions can return integers, reals, characters, booleans, pointers, enumerated types, and subranges (but not records, sets, or arrays.) There are numerous places where a type identifier is allowed but a type may not be constructed. Semicolons come after most statements (but not before end and never before else.) I also feel strongly that stop, return, exit, and next statements are necessary to promote structured programming.

An else or otherwise clause should be available in case statements. I don't buy Wirth's argument about unstructured programming. He left the goto statement in, so it is quite possible to write poor programs. Give a user enough rope and he can hang himself or climb a cliff. Many members have pointed out the need for a default, so I won't repeat the arguments. Different implementations, unfortunately, use else, otherwise, and <> as the default label. I feel the choice should be otherwise. The problem with else is that the preceding statement might be an if - then. The following illustrates the ambiguity:

```
case C of
    'A': writeln('blah');
    'B': if d = e then writeln('blah blah')
    else writeln('error')
end
```

The writeln('error') might be an else for the if or for the

case. (I hope implementors who use else have looked at this problem!) The usual kludge is to require a semicolon before the else (how confusing and inconsistant!) but when a user forgets to put the semicolon in, it is possible to get a syntactically correct program, producing an obscure bug that could go undetected for months. (This problem was pointed out by Charlie Fischer.) The case against <> as just another label is that it adds nothing to the language: Fraley's example on page 46 of PN #11 is unchanged if you simply remove the semicolon! Of course, using otherwise requires adding another reserved word, but I feel it is the most reasonable solution. The semantics of using an out of range selector with no otherwise should be defined.

Walt Brainerd proposed a loop construction for FORTRAN that solves the "exit in middle of loop" problem (SIGPLAN Dec. 77). Such a constructon can be PASCAL-ized and modified to reflect my own biases as is shown in the first syntax diagram. The semantics are that loop ... end cause repetition, and the various other parts give ways to get out of the loop. The for part has the same meaning as PASCAL's for statement: vary the index from the initial value until the final value and then quit. The while part also has the usual semantics:

```
if not <expression> then exit
```

The flag part lists one or more identifiers which should be declared as boolean variables. Entering the loop sets all the variables to false. An exit statement naming a variable sets that variable to true and jumps out of the loop. It is possible to jump out of more than one level of loop by naming a variable in the outer loop's flag part. If no variable is named, none is set, and the innermost loop is exited. The next statement behaves just like the exit except that rather than jumping out of the loop, the remainder of the loop body is skipped and the next execution of the loop begins (after any appropriate incrementing and testing of for and while parts,) and the boolean named is not set to true. (The only purpose of a variable in a next statement is to specify more than one level of loop.) If all three parts are left out of the loop header, an infinite loop results (presumably containing an exit statement somewhere.) Assigning true to one of the flag variables has no effect, is bad style, and might be prohibited.

This construction has a number of advantages. It includes the power of PASCAL's for, while, and repeat statements into one construct. It also has the power of being able to exit or resume one or more levels of loops from any point in the middle. In addition, when you get out of a loop, you can test the boolean variables to see what caused loop termination. Consider for example binary search:

```
l := 1; u := n;
loop while (l <= u) flag found do
    mid := (l+u) div 2;
    if x < A[mid]
        then u := mid-1
        else if x > A[mid]
                then l := mid+1
                else exit found
end;
if found then writeln('Found at',mid)
         else writeln('Not found');
```

Another example, finding prime numbers:

```
loop for p := 2 to n flag potential_prime do
    loop for d := 2 to trunc(sqrt(p)) do
        if p mod d = 0 then next potential_prime
    end;
    write(p)
end;
```

The second example above seems to be one case where PASCAL really needs a step option in the for loop, since it is only necessary to check the odd numbers and divisors. What is so all fired important that makes increments other than 1 and -1 against the spirit of PASCAL?

This construction is powerful enough to replace for, while, and repeat loops. A lone for or while part on the loop statement gives you the for and while loops, and a while part on the loop end gives you the repeat ... until construction.

In principle, the while clause is unnecessary, since a conditional exit at the beginning or end of the body will have the same effect. I argue that the while construction provides additional readability. The keyword flag is perhaps not ideal, Brainerd used until, which would only cause confusion in PASCAL. Another keyword, such as conditions, could be substituted.

I'm not suggesting throwing away PASCAL's looping constructions and replacing them with the loop statement. Clearly there is already too much investment in existing programs and compilers, and too little to gain. However, there are several points to learn from. PASCAL's looping constructs, even though far better than what is available in many other languages, still leave much to be desired. Future languages, including a possible PASCAL II, might include it. Alternatively, it might be possible to include the exit and next statements and the flag part in "standardized" extensions. The flag part could be optionally inserted before the do in for and while loops, and after the repeat in that loop. (See the second set of syntax diagrams.)

Work at the University of Wisconsin is currently in progress toward the design and implementation of a PASCAL based Artificial Intelligence language called TELOS. This language is a superset of PASCAL (with two exceptions: goto out of procedures, and passing procedures as parameters, are disallowed) and has numerous extensions (including a clean way to achieve the effect of passing procedure names as parameters.) The language includes features found in many other AI languages, with a special emphasis on the PASCAL philosophy of structured programming, readable code, and detection of errors at compile time.

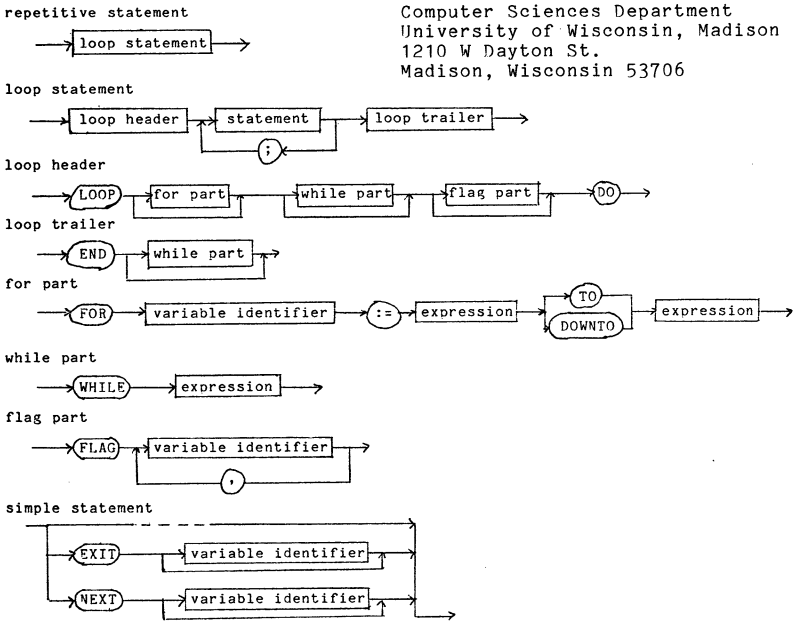
The language includes capsules (programmer defined data types with their own local operations); coroutines and other synchronous process manipulation facilities; events and handlers for them; an associative data base (referenced with patterns) that can hold objects of any user defined data type, including records, capsules, and the like; different contexts of the data base (so you can make a tentative modification to the data base and see how it compares); multi type pointers; record, array, and pattern constructors; modular compilation; and miscellaneous minor PASCAL extensions (including an otherwise in a case, flexibly sized array types, and functions returning any definable type.) The above loop construction is not part of TELOS, since an effort has been made to avoid cluttering the language up with extra features that can be gotten with existing features, and the event mechanism will provide the same power.

A TELOS implementation is currently under development based on Charlie Fischer's UNIVAC 1100 PASCAL compiler, which currently has most of the data base features implemented. Work is also

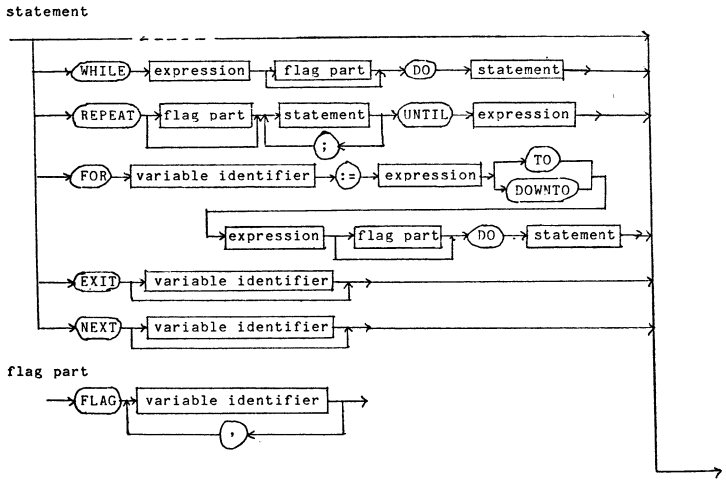
beginning on a portable TELOS interpreter based on the P Compiler.

Mark Horton
Computer Sciences Department
University of Wisconsin, Madison
1210 W Dayton St.
Madison, Wisconsin 53706

Syntax Diagram 1



Syntax Diagram 2



Westinghouse Electric Corporation Defense Group

Defense and Electronics Systems Center
Systems Development Division

Baltimore-Washington International Airport
Box 746 M.S. 451
Baltimore Maryland 21203

April 11, 1978

Mr. Andy Mickel
Pascal User's Group
University Computer Center 227EX
208 SE Union Street
University of Minnesota
Minneapolis, MN 55455

Dear Mr. Mickel:

I have been aware of PASCAL for several years. Recent interest by Department of Defense in PASCAL as a base for a DoD's Common Programming Language Effort has stimulated my interest. I am deeply involved in the DoD world of software and its unique problems. Possibly PUG can help with one unique problem, ie. The government requires detailed specifications for everything, including software. Further, the government requires acceptance test to be sure specifications are met. Is it possible for PUG to develop an acceptance test for PASCAL compilers? Don't answer too quickly. An acceptance test, that might satisfy government standards, requires the following:

1) a detailed, unambiguous specification.

2) A test against every item in the specifications.

As a case in point, Rome Air Development Center has a JOVIAL J73/1 compiler validation (acceptance tests) against MIL-STD-1589, that has over 20,000 source JOVIAL statements in 28 source modules. The JOVIAL validation is compiled and executed. The results of the execution are several thousand "TEST PASSED" or "TEST FAILED" messages with appropriate comments about the language feature being tested.

My group will soon be getting a PASCAL compiler for the UNIVAC 1110. Since the compiler may be used on government contracts, it would be a great help if an acceptance test was available.

Jon S. Squire, Manager
Operational Software

JSS:j

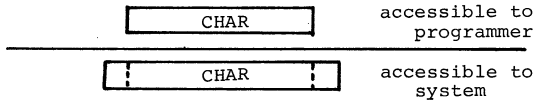
University of the Witwatersrand, Johannesburg
DEPARTMENT OF APPLIED MATHEMATICS
1 Jan Smuts Avenue, Johannesburg, 2001, South Africa
Telephone 39-4011, Telegrams 'University', Telex 8-7330 SA

The Editor,
Pascal News.

telephone ext
your reference
our reference JB/SW
date 7 April 1978

Dear John,

Thanks for your note - the question of predefined types also requiring a succ(max) facility had not occurred to me when I wrote "Subranges and Conditional Loops". The convention I suggest only works for genuine subranges, not full ranges such as integer, char and boolean. I thought long and hard of the possibility of letting these types be subranges of underlying and hidden ranges. These ranges would be one bigger on either side than the subrange we see but these "fringe" elements would not be accessible. Diagrammatically, we want



The idea would be to let a program declare

```
var  ch : 0.. succharmax
```

and write

```
ch: = charmin;
while (ch <= charmax) and condition do
begin
     (* something *)
     ch := succ (ch)
end;
```

The trouble is that the fringe elements are accessible: if succ(ch), when ch=charmax, is a valid expression then there is no way of stopping a program from writing it out - which would be invalid. Furthermore, there may be severe implementation problems since these types have a "fully packed" property i.e.they are usually represented in the exact number of bits required for max.

This leads me to realize that the predefined types, namely char, boolean and integer, are ranges and have a different nature to the subranges that we build on top of them. For the first time I feel some sympathy with Haberman and his "Critical Comments"!

...../2

To obtain the full effect of the above program in standard Pascal requires a boolean i.e.

```
var ch : char;
  ch := charmin;
  indexended  := false;
  while not indexended and condition do
  begin
       (* something *)
       if ch = charmax then indexended  := true
                       else ch := succ (ch)
  end;
```

This use of booleans is similar to that required to simulate sequential conjunction. I must admit that I don't like it and wonder if one day we'll have a "Booleans Considered Harmful" article.

I would be very pleased to hear if other Pascal people have thought about this problem and have alternative views to mine.

Enclosed are some membership forms - dollars are coming separately by Postal Money Order.

Best wishes,

Judy.

JUDY BISHOP

Encl.

(* Note: This letter is in reply to a letter sent on 78/03/08 from John Strait to Judy:

"Belated congratulations to you and Nigel! We received your card--you two make a handsome couple.
Andy let me read your article "Subranges and Conditional Loops" which he received yesterday. I have a question: What do you do with a pre-defined type which cannot be redeclared (e.g. CHAR) or one with special meanings (e.g. BOOLEAN). I ran into this problem last week with CHAR. Aside from this problem, I found your solution interesting/elegant."

*)

PASCAL STANDARDS

Editor: Tony Addyman Department of Computer Science
The University
Manchester M13 9PL
United Kingdom
(phone: 44-61-273-7121 x5546)

PLEASE DIRECT ALL INQUIRIES, LETTERS, ETC. ABOUT Pascal Standards to Tony. Thanks, Andy.

Jim Miner and I would like to bring you up to date on recent standards developments. First, Ken Bowles at the University of California, San Diego, has failed to keep us informed about his proposed summer workshop. We have no news since last issue!

Beginning this January, Jørgen Steensgaard-Madsen of the Datalogisk Institut, University of Copenhagen, Denmark, has done all Pascalers a favor by initiating work on the difficult task of conventionalizing extensions--thus answering Pierre Desjardins's good question in the last issue of PUGN. Jørgen is working in cooperation with Niklaus Wirth and several implementors: Jeff Tobias and Gordon Cox (IBM 370) in Australia, H. H. Nagel (DEC 10), in Germany, Olivier Lecarme (CII IRIS) in France, John Strait (CDC 6000) in the USA, Arthur Sale (B6700) in Australia, Ken Bowles (DEC PDP-11 and micros) in the USA, and Jim Welsh (ICL 1900) in the UK.

Olivier Lecarme published letters from Niklaus Wirth in the Bulletin No. 3 for the French Working Group on Pascal in March. The hope was expressed that this is hopefully the final work done in this area and that progress could be made if the number of people were kept small and the range of topics to be considered kept limited.

Jørgen is in contact with Tony Addyman who continues making progress on an ISO standard with his 10-(so far we at PUGN don't know who they all are)-person BSI working group called DPS/13/4. Tony now expects to have a draft document by the end of September.

In the course of our correspondence with Jørgen and Niklaus, we discovered another standardization effort begun by Justin S. Walker at the National Bureau of Standards (NBS) within the U.S. Government. He coincidentally (?) joined PUG 2 weeks later. We sent a personal letter to him trying to determine just what he is doing, and he did not answer. Hmmmmm.

Below is news from Tony and DPS/13/4: an Attention List #2.

Following that are several letters. Charles Fischer of the University of Wisconsin and Richard LeBlanc of Georgia Institute of Technology have stated very clearly some widely-held concerns over standards. Jim and I wholeheartedly agree with them.

Bob Vavra has written an outstanding and timely article and letter on design goals.

Arthur Sale has issued a revised version of his "Pascal Compatibility Report" [Department of Information Science Report No R78-3, May, 1978] which we described last time in this space. It now includes many more implementations.

Arthur is working with Brian Wichmann, of the National Physical Laboratory, in the United Kingdom on a set of Pascal programs to do: 1) Validity Checks - Does the compiler accept standard code, normal, or wierd? 2) Quality Checks: How does the compiler cope with error and error recovery? and 3) Compatibility Checks: How does the compiler cope in the undefined areas?

- Andy and Jim

April 7, 1978

Dear Andy,

I enclose a Pascal syntax written in EBNF. Would it be of any interest to the Newsletter?

Best regards,

Niklaus

Prof. Niklaus Wirth

ETH EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE ZÜRICH
Institut für Informatik

PASCAL syntax NW 12.3.78

(Extended BNF: cf. Comm.ACM 20, 11, p. 822, Nov. 1977)

```
identifier = letter {letter | digit}.
IdentList = identifier {"," identifier}.
UnsignedInteger = digit {digit}.
UnsignedReal = UnsignedInteger ["." digit {digit}] ["E" ScaleFactor].
sign = "+" | "-".
ScaleFactor = [sign] UnsignedInteger.
UnsignedNumber = UnsignedInteger | UnsignedReal.
String = "'" character {character} "'".

ConstantDefinition = identifier "=" constant.
ConstantIdentifier = identifier.
constant = [sign] (UnsignedNumber | ConstantIdentifier) | string.

TypeDefinition = identifier "=" type.
type = SimpleType | StructuredType | PointerType.
SimpleType = TypeIdentifier | ScalarType | SubrangeType.
TypeIdentifier = identifier.
ScalarType = "(" IdentList ")".
SubrangeType = constant ".." constant.
StructuredType = [PACKED] (ArrayType | RecordType | SetType | FileType).
ArrayType = ARRAY "[" SimpleType {"," SimpleType} "]" OF type.
RecordType = RECORD FieldList END.
FieldList = [FixedPart] [VariantPart].
FixedPart = RecordSection {";" RecordSection}.
RecordSection = [IdentList ":" type].
VariantPart = CASE [identifier ":"] TypeIdentifier OF variant {";" variant}.
variant = [CaseLabelList ":" "(" FieldList ")" ].
CaseLabelList = constant {"," constant}.
SetType = SET OF SimpleType.
FileType = FILE OF type.
PointerType = "^" TypeIdentifier.

VariableDeclaration = IdentList ":" type.
variable = identifier {index | "." identifier | "↑"}.
index = "[" expression {"," expression} "]".

expression = SimpleExpression [relation SimpleExpression].
relation = "=" | "<>" | "<" | "<=" | ">" | ">=" | IN.
SimpleExpression = ["+" | "-"] term {AddOperator term}.
AddOperator = "+" | "-" | OR.
term = factor {MulOperator factor}.
MulOperator = "*" | "/" | DIV | MOD | AND.
factor = variable | UnsignedConstant | FunctionDesignator | set |
         "(" expression ")" | NOT factor.
set = "[" [element {"," element}] "]".
element = expression [".." expression].
FunctionDesignator = identifier [ActualParameterList].
UnsignedConstant = UnsignedNumber | string | ConstantIdentifier | NIL.

statement = [label ":"] UnlabelledStatement.
UnlabelledStatement = SimpleStatement | StructuredStatement.
SimpleStatement = [AssignmentStatement | ProcedureStatement | GotoStatement].
AssignmentStatement = variable ":=" expression.
ProcedureStatement = identifier [ActualParameterList].
ActualParameterList = "(" expression {"," expression} ")".
GotoStatement = GOTO label.
label = UnsignedInteger.
```

```
StructuredStatement = Compoundstatement | ConditionalStatement |
                      RepetitiveStatement | WithStatement.
CompoundStatement = BEGIN statement {";" statement} END.
ConditionalStatement = IfStatement | CaseStatement.
IfStatement = IF expression THEN statement [ELSE statement].
CaseStatement = CASE expression OF case {";" case} END.
case = [CaseLabelList ":" statement].
RepetitiveStatement =  WhileStatement | RepeatStatement | ForStatement.
WhileStatement = WHILE expression DO statement.
RepeatStatement = REPEAT statement {";" statement} UNTIL expression.
ForStatement = FOR identifier ":=" ForList DO statement.
ForList = expression (TO | DOWNTO) expression.
WithStatement = WITH variable {"," variable} DO statement.

ProcedureDeclaration = ProcedureHeading block.
ProcedureHeading = PROCEDURE identifier [FormalParameterList] ";".
FunctionDeclaration = FunctionHeading block.
FunctionHeading = FUNCTION identifier [FormalParameterList] ":"
                  TypeIdentifier ";".
FormalParameterList = "(" FormalParameterSection
                      {";" FormalParameterSection} ")".
FormalParameterSection = [VAR | FUNCTION] IdentList ":" TypeIdentifier |
                         PROCEDURE IdentList.

block = [LabelDeclarationPart] [ConstantDefinitionPart] [TypeDefinitionPart]
        [VariableDeclarationPart] ProcedureAndFunctionDeclarationPart
        StatementPart.
LabelDeclarationPart = LABEL label {"," label} ";".
ConstantDefinitionPart = CONST ConstantDefinition ";" {ConstantDefinition ";"}.
TypeDefinitionPart = TYPE TypeDefinition ";" {TypeDefinition ";"}.
VariableDeclarationPart = VAR VariableDeclaration ";" {VariableDeclaration ";"}.
ProcedureAndFunctionDeclarationPart =
                          {ProcedureDeclaration ";" | FunctionDeclaration ";"}.
StatementPart = CompoundStatement.

program = ProgramHeading block "." .
ProgramHeading = PROGRAM identifier "(" IdentList ")" ";".

SYMBOLS
-------
+ - * / := . , ; : " = <> < <= > >= ( ) [ ] ↑ ..
(*  *)


Keywords
--------

AND ARRAY BEGIN CASE CONST DIV DO DOWNTO ELSE END
FILE FOR FUNCTION GOTO IF IN LABEL MOD NIL NOT
OF OR PACKED PROCEDURE PROGRAM RECORD REPEAT
SET THEN TO TYPE UNTIL VAR WHILE WITH


Predeclared identifiers
-----------------------

ABS ARCTAN BOOLEAN CHAR CHR COS DISPOSE EOF EOLN
EXP GET INPUT INTEGER LN NEW ODD ORD OUTPUT PRED
PUT READ READLN REAL RESET REWRITE ROUND SIN SQR
SQRT SUCC TEXT TRUNC WRITE WRITELN
```

PROFESSOR OF COMPUTER SCIENCE
T. KILBURN, C.B.E., M.A., Ph.D.,
D.Sc., F.I.E.E., F.B.C.S., F.R.S.
ICL PROFESSOR OF COMPUTER ENGINEERING
D. B. G. EDWARDS, M.Sc., Ph.D., M.I.E.E.
PROFESSOR OF COMPUTING SCIENCE
F. H. SUMNER, Ph.D., F.B.C.S.
PROFESSOR OF COMPUTER PROGRAMMING
D. MORRIS, Ph.D.

Recd 78/2/21

DEPARTMENT OF COMPUTER SCIENCE
THE UNIVERSITY
MANCHESTER
M13 9PL

Please note our **new** telephone number
061-273-7121

Andy Mickel
PUG
etc.

061-273-7121 X5546
6 February 1978

Dear Andy

This letter will serve several purposes. These are:

1. To tell you my new phone number for the roster.
2. To give and all others at PUG central the latest Attention List. As I said in my call, don't be alarmed by some of the items I still operating on the same basis as before.
3. I want to include several paragraphs from the beginning of Pascal News as an appendix to the textbook. This will serve to advertise PUG by describing Pascal News, giving the central and regional addresses etc.. If space permits I would like to include a copy of the All-Purpose-Coupon. Will this be OK? This does not need a reply. I will call you. If you cannot be around, you can always leave a simple yes/no answer.

4. A reminder to all PUG members that any contributions to the standardisation effort will be gratefully received by myself and other members of DPS/13/4.
5. As the enclosed material is rather bulky, you may print it or not as you see fit. Hopefully I will be able to send you some more up-to-date information before the next issue is due to by printed.

Yours

Tony

A M ADDYMAN

PROFESSOR OF COMPUTER SCIENCE
T. KILBURN, C.B.E., M.A., Ph.D.,
D.Sc., F.I.E.E., F.B.C.S., F.R.S.
ICL PROFESSOR OF COMPUTER ENGINEERING
D. B. G. EDWARDS, M.Sc., Ph.D., M.I.E.E.
PROFESSOR OF COMPUTING SCIENCE
F. H. SUMNER, Ph.D., F.B.C.S.
PROFESSOR OF COMPUTER PROGRAMMING
D. MORRIS, Ph.D.

DEPARTMENT OF COMPUTER SCIENCE
THE UNIVERSITY
MANCHESTER
M13 9PL

Telephone: 061-273 5466

1st February 1978

TO: Members of DPS/13/4, the Swedish Technical Committee on Pascal and all Correspondants.

May I first apologise for the delay in the production of Attention List #2. I (wrongly) decided to keep altering the list to include new material. Had I realised that it would take such a long time to produce the list, I would have issued an incomplete list earlier.

It is my hope that the list contains all the doubts and problems concerning Pascal which have been brought to my attention. If this is not so, then the list will be updated.

Since it is a long time since DPS/13/4 last held a meeting, and several of its members are very active, I am suggesting that a meeting be held in late February or early March. The meeting cannot be called too soon because the Swedish Technical Committee will need time to arrange for their representative(s) to be present.

Progress is being made in several related areas:

1. BSI is proposing the creation of an ISO project for Pascal.
2. Brian Wichmann is endeavouring to create a suite of Pascal test programs. This is an encouraging move for DPS/13, who collectively believe that validation suites for compilers should be provided wherever possible.
3. Prof. N. Wirth is optimistic concerning our copyright problems. Springer-Verlag have not yet replied to my letter.

To avoid further delaying this attention list, the following items will be sent separately later in the month:

1. A list of names (and where appropriate addresses and telephone numbers) of all people actively involved in the standardization effort.
2. A list of other people who are being kept informed of progress.
3. A selection of the guidelines and rules from BS:0, Part 3, which concerns the way in which the working group should operate.
4. A summary of the relevant parts of "Guidelines for Approving Standards", which is part of a document presented by the United Kingdom to ISO at the Hague meeting of the programming languages sub-committee. Although this was not accepted by ISO, it may be necessary for BSI to adhere to its own requirements.
5. Sections and sub-sections of the report which individual members of the working group are requested to study. Any member of the group is, of course, quite free to study and make suggestions concerning any part of the report.

A.M. Addyman
Convenor of DPS/13/4.

March 23, 1978

ACADEMIC COMPUTING CENTER
THE UNIVERSITY OF WISCONSIN - MADISON

1210 WEST DAYTON STREET
MADISON, WISCONSIN 53706
608-262-1166

Mr. Andy Mickel
Editor, Pascal News
Computer Center
University of Minnesota
Minneapolis, Minnesota 55455

Dear Andy:

As an implementor of a PASCAL compiler as well as a "firm believer" in PASCAL's merit as a programming language, I feel compelled to comment on Ken Bowles' recent proposal (PASCAL News #11) to convene a workshop to standardize PASCAL extensions. The value of standardizing the extensions all implementors (including this one!) seems to add to PASCAL is unquestionable. However, Bowles' approach seems to me to be rather suspect. If this standard is to have any real value, it must have broad-based support in the PASCAL user community.

Why then should the effort to produce this standard be dominated by organizations with a large monitary investment in PASCAL with the gratuitous inclusion of "a small number of academic experts" to placate the rest of us? Are we to believe the opinions of the average PASCAL devotee are less important than those of industrial and governmental organizations? Such an idea strikes me as rather odd given the fact that PASCAL has succeeded not because of these organizations, but in spite of them.

Even if the composition of Bowles' workshop is made more equitable and broad-based, I have very serious reservations about any language designed by committee. It can be very strongly argued that PASCAL's simplicity and elegance derives directly from the fact that it was designed by one man. Why not then adher to this principle?

Bowles' workshop should by all means meet (although with a more broad-based composition). Rather than drafting a specific set of language extensions, however, it should draft a set requirement that an extended PASCAL should meet. Where necessary, differences in emphasis or opinion should be included--all concerned parties must have a say in what they feel is important. These requirements should then be forwarded to a very small group of acknowledged language design experts (Nicklaus Wirth would, of course, be ideal) who would produce a single set of specific language changes consonant with the "spirit" of PASCAL, the state of the art, and the overall requirements given them. This set of changes would then be widely distributed, discussed and debated, but accepted or rejected as a whole. If they are rejected, (say by vote of the PUG membership) then we must acknowledge that no standardization is, at present, possible. If they are approved, we should accept them as the one and only definition of extended PASCAL.

I realize, of course, that my opinion of how standardization should be done is just one man's viewpoint. However, the principle that everyone should have a voice in what kind of extensions should be included while limiting to a very few experts the decision of exactly how these extensions are to be specified seems a sound one. If we are to produce a standard, let us make every effort to ensure it is something we can live with.

Sincerely,

Charles N Fischer

Charles N. Fischer

GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL OF INFORMATION AND COMPUTER SCIENCE • ATLANTA, GEORGIA 30332 • (404) 894-3152

April 10, 1978

Mr. Andy Mickel
Editor Pascal News
University Computer Center: 227 EX
208 SE Union Street
University of Minnesota
Minneapolis, MN 55455

Dear Andy:

I wish to make a few comments about Ken Bowles' proposal (PASCAL News #11) for a workshop to develop a set of standardized PASCAL extensions. As an experienced PASCAL user and an implementor of a PASCAL compiler, I certainly agree that there are areas in which PASCAL could be improved as a systems programming language. There is no doubt that some standardization of extensions would be quite valuable to both users and implementors. However, I have some reservations about the process Bowles has proposed to develop these extensions.

Having had some experience in designing programming languages, I am quite concerned that a set of extensions produced by a large committee might not be consistent with the simplicity that is one of the most attractive characteristics of PASCAL. This simplicity probably results from the fact that PASCAL was designed by one man. It might also be noted that while the DoD "Ironman" project mentioned by Bowles included input from a great number of people in identifying goals, the actual language design work is being done by small groups. I think it more appropriate that any large workshop produce a statement of requirements rather than a "finished" language design.

The fact that attendance at the workshop will be restricted to a certain group of PASCAL users is also of concern to me. If the language to be produced by the workshop would end up being used by only the participants, this would not be objectionable. However, any extended PASCAL standard adopted by a group of users with considerable economic influence is likely to become a de facto standard. It is not acceptable for the PASCAL user community to have so little influence in such an effort. Further, there are apparently other standardization efforts under way. These should certainly not be ignored.

If Bowles wishes his workshop to produce a systems implementation language designed by and for industrial firms and government agencies, the language should be given a name that does not contain "PASCAL" so that there will be no confusion as to its nature. If the workshop is to make a more valuable contribution toward the standardization of PASCAL extensions, a broader group of participants is necessary and more care must be taken to insure that the resulting language reflects the "spirit of PASCAL" and is acceptable to PASCAL users in general.

Sincerely,

Richard J. LeBlanc

Richard J. LeBlanc,
Assistant Professor

741 Terrace Dr.
Roseville MN 55113 USA
March 30, 1978

Dear Andy,

In your recent article on Pascal Standards (PN 11 page 65), you and Jim point out that many people are suggesting (and implementing) changes and extensions to Pascal, but few are using Pascal's design goals to evaluate their changes or extensions (at least few are doing it publicly). By referring readers to the design goals listed on the back cover of PN, you imply that Pascal's design goals are well-understood and generally accepted. I disagree on both counts.

* The ten-line description of Pascal's design goals is adequate for the back cover of PN, but it is too vague for use in judging proposed language features. For example, "general purpose but not all-purpose" is a very nice phrase which is intuitively appealing, but it gives little or no guidance to someone who is attempting to evaluate a proposed feature. If I didn't know you, I would be tempted to suggest that you are purposely fostering confusion in this area to keep up the volume of provocative language proposals in PN.

* You have been exhorting people to justify their proposals in terms of Pascal's design goals since PUG was formed (PN 5 page 2). The fact that few people have done so seems to indicate that many disagree with or do not understand the stated design goals (or else they don't read your editorials).

I agree with you that any future development of Pascal, beginning with the Standardized Extensions that you call for, must be guided by a clearly-defined set of design goals. Discussion of Pascal's design goals has been remarkably absent from most material appearing in PN. I enclose an article which opens such a discussion.

I am pessimistic about our ability to standardize anything beyond the Revised Report (which Addyman seems to have well in hand), without institutionalizing Pascal to a very great degree. (Doing language design by committee is very difficult, but doing it in the pages of PN by a committee-of-the-whole is unthinkable.) The Simula Standards Group (as described by Palme in Software Practice and Experience 1976 pages 405-409) seems to be successful because it has a representative from each of the eight Simula implementations. A Pascal standards committee consisting of a representative from each Pascal implementation would not be a viable design group, and a committee which is any less representative would have difficulty achieving widespread adoption of its standards.

Even so, I am optimistic about the future. It will not be the end of the world (or of Pascal) if we fail to standardize. Pascal's greatest weakness (a multiplicity of incompatible implementations) is a direct result of its greatest strength (simplicity and clarity of programming). The Pascal compilers that I have seen are sparsely documented (to say the least), but many users have been able to modify them to accept new features. Can you imagine as many people doing that to Fortran or Cobol compilers?

Sincerely,

Bob

Bob Vavra

Implementation Notes

CHECKLIST

Please note the new Checklist entry number 0: DATE/VERSION. In preparation for next year, we encourage implementors, users, or anyone else to submit new or revised checklist reviews for their favorite implementations.

Pascal Implementations Checklist

```
0.  DATE/VERSION
      (* Last checklist changes; version name or number, if any. *)

1.  DISTRIBUTOR/IMPLEMENTOR/MAINTAINER
      (* Names, addresses, phone numbers. *)

2.  MACHINE
      (* Manufacturer, model/series and equivalents. *)

3.  SYSTEM CONFIGURATION
      (* operating system, minimum hardware, etc. *)

4.  DISTRIBUTION
      (* cost, magnetic tape formats, etc. *)

5.  DOCUMENTATION
      (* In form of supplement to Pascal User Manual and Report?
         Machine retrievable? *)

6.  MAINTENANCE POLICY
      (* How long? Accept bug reports? Future development plans. *)

7.  STANDARD
      (* Implements full standard? Why not? What is different? *)

8.  MEASUREMENTS
      (* -compilation speed (in characters/sec. please; this is a
            meaningful measurement for compilation speed);
         -compilation space (memory required at compilation time);
         -execution speed;
         -execution space (the memory required at execution time;
            compactness of object code produced by the compiler);
      ** Try to compare these measurements to the other language
            processors on the machine, e.g., FORTRAN. *)

9.  RELIABILITY
      (* stability of system (poor, moderate, good, excellent);
         how many sites are using it?
         when was the system first released to these sites? *)

10. DEVELOPMENT METHOD
      (* Compiler or interpreter? Developed from Pascal-P / hand-
         coded from scratch/bootstrapped/cross-compiled/etc.? What
         language? Length in source lines? Effort to implement in
         person-months? Previous experience of implementors? *)

11. LIBRARY SUPPORT
      (* Libraries of subprograms available? Facilities for
         external and FORTRAN (or other language) procedures
         available? Easily linked? Separate compilation available?
         Automatic copy of text from library into source program
         available? Symbolic dumps available? *)
```

PORTABLE PASCALS

Pascal P4 -- Bug Reports

Due to a fit of oversight, I forgot to print in issue #11 Chris Jacobi's Updates 1 and 2 to P4. They appear below. Also, there were a couple of errors in the bug list in issue #11. Bob Fraley caught one (see his letter below). The other error was in bug number (17.), in which the fix should have read:

Replace PASCP.2826 with:
```
        THEN ERROR(131);
```
Replace PASCP.2831 with:
```
        ERROR(131);
```
- Jim Miner

HEWLETT hp PACKARD

3500 Deer Creek Road. Palo Alto. California 94304. Telephone 415 494-1444, TWX 910 373 1267

March 22, 1978

Mr. Jim Miner
25 Blegen Hall
University of Minnesota
West Bank
Minneapolis, Minnesota 55455

Dear Jim,

The fix for set declaration error checking is incorrect. In particular, if any error occurs, LSP is not set and therefore FSP is set to a bad value. To maximize error checking of uses of this type, I would suggest the following fix:

Replace PASCP 1275 by:
```
   IF LSP1 = REALPTR THEN
     BEGIN ERROR(114); LSP1 := NIL END
   ELSE IF LSP1 = INTPTR THEN
     BEGIN ERROR(169); LSP1 := NIL END
   ELSE
     BEGIN GETBOUNDS (LSP1, LMIN, LMAX);
       IF (LMIN < SETLOW) OR (LMAX > SETHIGH) THEN
         ERROR(169);
     END;
```

This will build a "SET" type node, checking the use of variables which have this type. (Alternatively, set LSP: = NIL before PASCP 1271, and remove "LSP1: = NIL" from line PASCP 1273).

The correction to field list, allowing ";" before the "END" of a record definition, is incomplete. In particular, the syntax allows null field entries (multiple ";" in a row). The full fix is:

Replace PASCP 1077 by
```
   WHILE SY = SEMICOLON DO
```
Change PASCP 1079 to:
```
   IN FSYS + [IDENT, CASESY, SEMICOLON]
```
Change P 150 to
```
   IN FSYS + [SEMICOLON]
```

There is another error in P4 which causes an infinite loop when a comment is not closed.

Replace PASCP 509 by

```
UNTIL (CH = ')') OR EOF (INPUT);
```

Replace PASCP 507 by

```
WHILE (CH <>'*') AND NOT EOF (INPUT) DO NEXTCH:
```

Sincerely yours,

Bob Fraley
Hewlett-Packard Laboratories
Electronics Research Laboratory

RAF/hma

(* Thanks, Bob! *)

Pascal P4 — UPDATE1 and UPDATE2

Both of these updates are dated January, 1977. They were issued by Chris Jacobi of ETH, Zuerich, and we printed them in issue #8.

UPDATE1:

```
Replace BOOT.4 by:
        FOR I := ORDMINCHAR TO ORDMAXCHAR DO SOP[CHR(I)] := NOOP;

Replace P.477 by:
        LOAD;  GENLABEL(LCIX);

Insert after P.479:
        GENUJPXJP(57(*UJP*),LCIX);

Replace P.147 by:
        BEGIN ALIGN(LSP1,DISPL);

Replace P.424 by:
        LOCPAR := LOCPAR + PTRSIZE;
        ALIGN(PARMPTR,LOCPAR);

Insert after line PASCP.3200:
        ALIGN(PARMPTR,LLC1);

Replace P.531 by:
        IF IDTYPE^.FORM > POWER THEN

Insert after PASCP.3204:
        IF VKIND = ACTUAL THEN
        BEGIN

Insert after PASCP.3207:
        END;
```

UPDATE2:

```
Replace P.122 by:
        FLC := L + K - (K + L) MOD K

Replace P.528 by:
        CSTPTRINX := 0;
        TOPNEW := LCAFTERMARCKSTACK;
        TOPMAX := LCAFTERMARCKSTACK;
```

Implementation Notes

FEATURE IMPLEMENTATION NOTES

INTERIM REPORT - IMPLEMENTATION OF SETS - 1

This report addresses results of set implementation tests on three compilers, and a personal estimation of optimal treatment (not yet achieved anywhere so far).

Professor A.H.J. Sale,
University of Tasmania.

interpretation	CDC6000	B6700	ICL 1900	optimal
set of char	compile error	compile error	allowed, run-check	allowed, correctly
set constructor with value > setmax	allowed	compile error	compile error	compile-error if genuine, however limit should be big, say 256.
set-type constraints	0..58	0..47	0..47, but only checked at runtime.	no limits, except range in reasonable size (256 bits?)

Note: In the ICL compiler, sets may be declared of any subrange type, and the run-time system will be correct as long as no element with a representation outside 0..47 is involved. If this occurs, an "index error" is raised. (I believe this to be liable to lead to undetected and hibernating bugs.)

MACHINE-DEPENDENT IMPLEMENTATIONS

Burroughs B1700

PASCAL ON BURROUGHS B1700

Dear Mr. Mickel,
we have developed a Pascal System for the Burroughs B1700/B1800 Series. The System like many others is derived from the Pascal-P Compiler developed by Wirth and Amman at the ETH-Zuerich.
A preliminary Version has been distributed to several european Universities about a year ago. The System is also the subject of a PhD Thesis in german.
Unlike other B1700 Pascal systems, ours is implemented on top of the B1700 SDL-S Language

which also serves as the Basis for the master control program and the utility software. The system runs on MCP-Release 6.0 and higher and is particularly suitable for small machine configurations.

In order to remain compatible with the standard SDL-Architecture only emulation of realarithmetic is provided.

Current Projects include addition of mathematical functions and the design of an "ideal" Pascal architecture.

The system has recently been redesigned and we will gladly distribute it to universities sending us a tape. (We would appreciate tapes in a reusable box. Installations should also indicate if they have use of SDL- and MIL-Compilers). Unfortunately we can not guarantee an errorfree system but we will eventually fix errors made known to us.

Another Pascal-system was produced at our installation by Mr. K. Haeusermann. It uses a separate interpreter which emulates the hypothetical stack computer by Wirth and Ammann.

Pascal systems for the B1700 have also been developed at many other universities (Karlsruhe, Newcastle, Dublin, Edinburgh....).

Yours sincerely

Peter U. Schulthess
Institut fuer Informatik
Universitaet Zuerich
Kurvenstrasse 17
8006 ZUERICH
============
SWITZERLAND

Burroughs B4700 (Fredonia)

George Golden, Sr. (Computer Center; SUNY Fredonia; Fredonia, NY 14063; 716/673-3393) wrote on 78/4/10: "We are trying to get Pascal running on the Burroughs B4700. It runs! But it takes too much core."

Burroughs B6700 (Tasmania)

The PASCAL compiler for Burroughs B6700/B7700 systems written at the University of Tasmania is now available for distribution. To acquire a copy, fill out the attached forms and send to:

PASCAL Support,
Department of Information Science,
University of Tasmania,
Box 252C, G.P.O.,
HOBART, 7001

The compiler is distributed on 9-track magnetic tape, (but 7-track is also available) and an installation manual is supplied, together with two copies of the user-documentation. At present this comprises:

* Report R77-1 - a supplement to Wirth's User Manual.
* Report R77-3 - a Reference Manual similar to B6700 Algol's.
* a PASCAL Language Card.
* a PASCAL System Card.

Further copies of the user-documentation may be available at production cost.

The charge for the system is Australian $100 annually, and will be invoiced to you when you receive the tape. The tape remains our property, and should be returned when you have copied its contents so that future releases can be mailed to you. The service will cover:

* mailing and processing costs,
* extensions and revisions, and
* the costs of an FTR-reporting service and maintenance.

Each installation will be issued with FTR-forms similar to those used by Burroughs for use in corresponding with us, and we will attempt to do a professional job in maintenance of the system.

The Tasmania B6700 PASCAL compiler is a true compiler for the B6700 or B7700 computer systems: it generates executable code-files which are accepted by the operating system. Its compilation and execution performance is within a 20% margin of comparable compilers in the B6700 system for average programs. The current version generates LINEINFO in the code-file, but does not generate BINDINFO, so PASCAL programs cannot yet be linked to other code-files. The compiler itself is written in B6700 Algol, as are most of the extra trinsic procedures it uses.

Objectives of this project were to develop a compiler which enforced compliance with the standard definition of PASCAL as far as possible by utilizing the special features of the B6700 system, while making it a fully integrated member of the B6700 compiler set. These targets have been largely met, and a wide variety of checks are available to the user-programmer; probably to a higher degree than most other PASCAL compilers. However, file attributes, record-oriented formatted i/o, random-access i/o, and compiler options, are provided in a way that will ease the learning problems of existing B6700 programmers. The compiler permits use in a very similar manner to the well-know compilers (Algol, FORTRAN, COBOL, etc).

The compiler has been stable in code for some time, reflecting its basic integrity. However new features are added from time to time, and notified to recipients as patches or as new version releases. The Department accepts FTR notices, and will attempt to fix those which warrant such attention. Some modifications have taken place as a result of user feedback. The compiler was especially designed so as not to generate dangerous code to the MCP, and no system crashes have been attributed to it since the first few months of testing, and then only three!

User-level documentation is provided for the compiler in the form of cards and reference manuals. The standard of these is similar to that of Burroughs' manuals and cards. Systems documentation is more sparse, and consists of some implementation notes, the compiler itself (a microfiche listing is provided), and a report on aspects of the language.

The compiler is in daily use by students at the University of Tasmania.

I must apologize to those of you who wrote enquiring about the availability of our B6700 PASCAL compiler earlier and did not receive a prompt reply. The end of the academic year and a number of important decisions interfered to prevent us making the compiler available as soon as we would have liked.

To cut a long story short, the B6700 PASCAL compiler developed at the University of Tasmania is now available from us. There are three conditions:

(1) each recipient must agree not to disclose the compiler to other parties, and must agree not to supply copies to other institutions.

(2) an annual fee of $100 (Australian) is required to cover mailing, processing, and other maintenance charges, payable to "The University of Tasmania".

The compiler has been operational in a student environment at the University of Tasmania for a year and has proved stable and reliable; it has been released on a restricted basis to two other sites for about eight months with similar results. The compiler is provided with a Reference Manual and a Supplement to the User Manual (of Jensen & Wirth), and with ready-reference cards. Recipients are granted copyright permission to reproduce these for their own purposes, and in some cases additional copies may be ordered from the University of Tasmania. The service provided includes the provision of updated versions of the compiler at intervals, and the maintenance of an FTR-service similar to that of Burroughs.

If you want further information before ordering the compiler, please write and we can send you documentation and listings. If you want a copy, please arrange for the non-disclosure notice (FORM A) to be signed by a responsible officer of your institution and the computing centre manager (if applicable), and forward it with the supplementary information (FORM B) to the address given.

Yours sincerely,

A.H.J. Sale

Professor A.H.J. Sale,
Department of Information Science.

CII 10070, IRIS 80 (Paris)

0. DATE/VERSION. 78/02/21.

1. Distributor/implementor/maintainer:

implementor:	distributor/maintainer:
D.Thibault 17, rue Gay-Lussac F-75005-Paris	P.Maurice Université Paul Sabatier Informatique 118,route de Narbonne F-31077-Toulouse-cedex (61)53-11-20(300)

2. Machine: CII-10070,CII-HB-IRIS 80,XDS-Sigma 7

3. System configuration: Siris7,Siris8.Easily available on other systems: adaptation of run-time routines and perhaps of the code-generation phase of the compiler.

4. Distribution: source programs (Pascal and assembly code), object programs and load modules available on magnetic tape (9 tracks,1600bpi);send a mini-tape to distributor; mailing cost only.

5. Documentation: user manual, in french (sept. 75);separate papers describe extensions and differences with the User Manual and Report (K.Jensen,N.Wirth); not machine retrievable.

6. Maintenance policy: bug reports are encouraged;announcements of releases are sent to users, together with listings of modifications (errors and/or extensions).Release 5 has been issued in Jan. 78.

7. Standard:
   - not implemented:
     - type T= <type identifier>
     - record ... case <type id> of ...
       (tag field is mandatory)
     - structured types of files.
   - extensions:
     - separate compilation
     - VALUE part for global variable initialization
     - heap management through NEW/DISPOSE or NEW/RESET
     - standard files TTYIN,TTYOUT used for interactive applications programming
     - compiler options (source listing,run-time checks, post mortem dump, pseudo-assembler listing of generated code.

8. Measurements:
   - compilation space: minimum 32K words;
     40K words to compile the compiler.
   - compilation speed: ≃ 2100c/s (Fortran: ≃ 1300c/s)
   - execution speed: programs from N.Wirth(ETH Zürich,March 76):

	Pascal run-time checks	Pascal no checks	Fortran
palindromes	4260ms	3860ms	2970ms
powers of two	1530ms	1470ms	3867ms
prime numbers	1900ms	1700ms	941ms
count characters in a file	5100c/s	5800c/s	5100c/s

9. Reliability: good;used since 1974 in ≃25 installations, mainly for teaching programming and compiler writing, and also for the development of large system software projects.

10. Development method: fully bootstrapped from Amman's CDC compiler;generates code for the CII link editor;all operating system dependencies are located in a monitor (≃2000 lines of assembly code), which must be linked with user programs. The compiler takes advantage of the separate compilation system: it consists of four overlayed modules (≃8500 'pretty-printed' Pascal lines).
The bootstrap process took about 2 man/years, to produce a compiler for the first version of the language(Wirth 71);adaptation to standard took about 6 man/months.

11. Library support: a system library contains the standard Pascal functions SIN, COS,... and the Pascal monitor (see 10).
Separate compilation allows using private libraries, written in Pascal or in any other language;interfacing with other languages requires a knowledge of the compiler.
Programs are manipulated under control of a 'Pascal programming system', which provides the users with powerful editing functions, ranging from source inclusion to program transformations.Also provided are interactive debugging at compile and execution time, and library management.The system is entirely written in Pascal (≃22000 lines).

Commodore 6502.

Formerly MOSTEK. See DEC LSI-11 (San Diego).

Computer Automation LSI-2, 4

Bob Hutchins (Computer Automation; 18651 Von Karman; Irvine, CA 92713; 714/833-8830x335) wrote on 78/3/1: "We just recently brought up sequential Pascal on out new 16-bit minicomputer series, the Naked Mini-4 series. It runs on all models including the NM-4/10 which sells for $645 including CPU, 4K RAM, and 4 I/o ports. As far as I know, this is the lowest priced minicomputer system that supports Pascal. Our Pascal is based on sequential Pascal supplied by Brinch Hansen. It is supplied at a one time fee of $900 including compiler, interpreter, and documentation."

Minicomputer News reported on page 2 of their Jan. 5, 1978, issue that "Pascal software [on the LSI-4 line], formerly priced at $900, will be offered without charge."

Data General ECLIPSE (San Bernardino)

MEDICAL DATA CONSULTANTS (714) 825-2683

March 10, 1978

1894 Commercenter West, Suite 302, San Bernardino, CA 92408

Dear Andy,

We have spent the last several months in a reconsideration of our entire PASCAL endeavor. As we reported previously, we have developed a new version of Data General compatible PASCAL which is significantly faster than our previous version, but which continues to use a 64-bit data path, is fully RDOS compatible and easily modifiable and extendable. We had previously intended to take this version to market as a low priced, but profit making venture, as reported in the February PASCAL NEWS.

As part of our continuing PASCAL development we now have a preliminary implementation of a PASCAL compiler which produces code that executes at the speed of that provided by Data General's Optimizing FORTRAN 5. We expect, however, the full development of this product will take 6-12 months.

We have decided, therefore, to release our current improved version of Data General PASCAL for a reproduction cost of $100.00 on 800 BPI, 9 track magnetic tape. This includes executable object code, source code and machine readable documentation.

Please find attached a standard description of the product.

Sincerely,

Ted C. Park
Director, Systems Development

DISTRIBUTOR/IMPLEMENTER/MAINTAINER

Ted C. Park
Director, Systems Development
Medical Data Consultants
1894 Commercenter West
Suite 302
San Bernardino
CA 92408

MACHINE

Data General - any ECLIPSE-line computer

SYSTEM CONFIGURATION

ECLIPSE must have FPU or EAU
Minimum of 16K words user memory
RDOS REV 6.1 or 6.2
FORTRAN 5 (any recent revision)

DISTRIBUTION

System supplied on 9-track 800 BPI tape in RDOS 'dump' format.
The cost is $100.00 to cover our mailing and duplicating costs.

DOCUMENTATION

User must obtain his own copy of the Pascal Users Manual and Report.
It is recommended that the user obtain an implementation kit from the University of Colorado.
Documentation and operating procedures are supplied on the tape.

MAINTENANCE POLICY

Bug reports are welcome but no formal commitment for support can be made at this time. To date, no bugs have been reported.

STANDARD

PASCAL P4 subset

MEASUREMENTS

Compilation Speed:	50 chars/sec (including blanks and comments)
Word Size:	64 bits
Real Arithmetic:	Uses 64 bits
Integer Arithmetic:	Uses 32 bits
Execution Speed:	Fairly slow (since it is interpreted!)
Minimum Memory Needed:	16K words
Virtual Memory Required:	A contiguous disc file of 524,288 bytes

RELIABILITY

Version 1 exists in at least 10 sites, we believe no bugs exist. Version 2 is primarily the same as Version 1 except with improved operating procedures, faster compiles and executions, and increased capability. As such we anticipate few, if any, bugs.

DEVELOPMENT METHOD

Developed from PASCAL-P4. P-CODE assembler and interpreter written in assembly language. All programs are extremely modular and well documented so that any changes wished by the user should be easy to incorporate.

LIBRARY SUPPORT

No Data General libraries are needed to run the system nor is it possible to use any if desired.

Data General NOVA, ECLIPSE (Columbia)

Rhintek, Inc.

Computer Engineering — *P.O. Box 220, Columbia, Maryland 21045*

March 8,1978

Dear Andy,

RHINTEK, Inc. is making available its PASCAL compiler to other Data General NOVA/ECLIPSE users. This compiler is used by RHINTEK as an application and system programming language and will continue to receive support and enhancements by us. We are using the compiler on a NOVA 3/D running Rev. 6.10 mapped RDOS. However, we are cleaning up the code and expect the compiler to be able to run under unmapped RDOS on a 32k NOVA within a few weeks.

Below is the checklist information on our PASCAL compiler for Data General NOVA (or equivilent) computers.

DISTRIBUTOR/IMPLEMENTOR/MAINTAINER -- RHINTEK, INC; Box 220; Columbia, Md. 21045 (301)

MACHINE -- Data General NOVA or ECLIPSE minicomputers or equivalents.

SYSTEM CONFIGURATION -- Mapped RDOS system or 32K unmapped RDOS with minimum operating system. The current revision of Data General RDOS will be supported but the compiler should work with older levels.

DISTRIBUTION -- 9 track magnetic tape, 800 BPI, 7.5 inch tape in the RDOS dump format. Price for a single user license is $975. Multi-use, OEM's, and educational licenses will be handled on a separate basis.

DOCUMENTATION -- The package includes source code, binary code, and ready to run demo programs. Instructions for executing the compiler are included; the operational information can be obtained from the books by Per Brinch Hansen and Al Hartman.

MAINTENANACE POLICY -- Updates for one year and notification of substantial enhancements as long as interest is shown. We will maintain a users group and encourage bug reports and suggestions.

STANDARD -- Based on Sequential PASCAL written by Per Brinch Hansen and Al Hartman. The current version lacks: "file, goto, label, and packed" reserved words and sqr, sin, cos, arctan, ln, exp, sqrt, eof, eoln, odd, and round built in functions. This is a seven pass Sequential PASCAL compiler written in PASCAL and generating code for a hypothetical 'stack' machine. The code is interpreted by a program written in NOVA assembly language.

MEASUREMENTS -- The compiler compiles source code at the rate of 200 line/min. This is about one-half of the rate of the PDP 11/45 but five to ten times the speed of the other compilers on the NOVA. The compiler will compile itself in about 30 minutes total.

RELIABILITY -- good

DEVELOPMENT METHOD -- The virtual machine interpreter was coded in NOVA assembly language and then the compiler was modified along with interpreter into its present form.

LIBRARY SUPPORT -- There is no library support as yet. The operating programs support program swapping or chaining with only minor effort as this is used with the compiler.

Sincerely,

Rainer McCown

Rainer McCown, President
Rhintek, Inc.

DEC PDP-11 (Berkeley)

A package of UNIX software is available from the Computer Science Division of the University of California at Berkeley. This package includes the instructional Pascal system which has been in use at Berkeley this past year and the standard Berkeley editor ex, an extension of the standard UNIX editor ed which offers many new and improved features. The Pascal system requires separate I/D space to run (an 11/45 or 11/70); ex will run without separate I/D but requires a full load of user core (64 bytes).

UNIX Pascal is designed for interactive instructional use. It produces interpretive code, providing fast translation at the expense of slower execution speed. An execution profiler and Wirth's cross reference program are also available with the system. The systems supports full Pascal, with the exception of procedure and function names as parameters. The language accepted is very close to 'standard' Pascal, with only a small number of extensions for the UNIX system. (An option restricts the implementation to the standard.)

The UNIX Pascal User's Manual gives a list of sources relating to the UNIX system, the Pascal language, and the UNIX Pascal system. Basic usage examples are provided for the Pascal interpreter components pi, px, and pxp. Errors commonly encountered in these programs are discussed. Details are given of special considerations due to the interactive implementation. A number of examples are provided including many dealing with input/output. An appendix supplements Wirth's Pascal Report to form the full definition of the UNIX implementation of the language.

Source code, binaries and machine readable versions of all documentation are included with the tape. The Pascal system and the ex text editor are distributed under a license agreement; UC Berkeley is thus the sole source for this software. The software is distributed only to UNIX licensees and only for non-commercial purposes. A copy of the cover page of the UNIX license agreement is an acceptable form of proof of license.

The distribution tape is a standard "tp" format, 800 BPI magnetic tape. A 1200 foot reel is the minimum and preferred size. There is a one time $50 charge ($65 for overseas airmail) for a copy of this tape. This charge includes the costs of preparing the tape, mailing costs, and the costs of distributing future updates and corrections to the programs and documentation on the tape. These updates and corrections will be distributed at regular intervals as their volume and severity warrants. Also included with the tape are high quality copies of the UNIX Pascal User's Manual and the Ex Reference Manual which require a phototypesetter to produce. It is also possible to obtain a copy of the documentation without getting a copy of the tape. The $5 charge for this copy may be deducted from the tape charge if you later decide that you want a tape. If you prefer, you may send an additional $10 and we will purchase a tape on which to send you the software.

To receive a copy of the license agreement (which must be signed before you can receive the tape) write to:

Berkeley UNIX Software Distribution
c/o William N. Joy
Computer Science Division
Department of EECS
Evans Hall
University of California, Berkeley
Berkeley, California 94720

Questions about this tape can be directed to William Joy at the address above or at (415) 642-4948. Messages can be left at the Computer Science Division office phone (415) 642-1024.

DEC PDP-11 (Stockholm)

0. DATE/VERSION. 77/12/22.

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. Seved Torstendahl, TN/X/Tds, Telefon AB LM Ericsson, S-126 25 Stockholm, Sweden, tel 08-719 4909.

2. MACHINE. Running on PDP-11 model 35 and up and generates code for all PDP-11's. Crosscompiler on DEC-10 generates code for all PDP-11's. The compilers generate code for floating point hardware and extended arithmetic if option switches are set.

3. SYSTEM CONFIGURATION. RSX-11M or IAS. (DEC-10 crosscompiler under TOPS-10). PDP-11 with memory management and a user partition of at least 28 Kwords, preferrably 32 Kwords. It would be an easy task to replace the RSX interfacing routines with new ones interfacing DOS or RT-11. We don't plan to do that work.

4. DISTRIBUTION. The compilers are available at $50, plus $10 if we supply the tape (600 feet). The distribution set contains source and object modules of the compilers and the runtime library, command files for compiler generation and maintenance, user manual and compiler generation instructions. The compiler will be distributed in one or more of the following formats, indicate which you want:
- three DECtapes in PDP 11 DOS format (DEC10 and PDP11 users)
- one 9-track magnetic tape in DEC 10 format (DEC10 users)
- one 9-track magnetic tape in industry compatible format (users of DEC10 and other computers)
- one 9-track magnetic tape in DOS format (PDP11 users).

5. DOCUMENTATION. A machine retrievable user manual complementing the PUM&R book is included on the distribution tape.

6. MAINTENANCE. No responsibility, but if errors are found reports will be distributed to known users. Error reports and improvement suggestions accepted.

7. STANDARD. Restrictions:
- packed data structures are only implemented for character arrays (always packed, two char's/word) and for boolean arrays (packing optional, one boolean/bit). The procedures pack and unpack are not implemented.
- only local jumps are allowed.
- a pair of procedures, mark and release, to allocate and deallocate dynamic storage.

Extensions:
- function results can be of nonscalar type,
- arrays with unspecified bounds (but specified index-structure) can be used as formal parameters to procedures, allowing differently declared variables or constants as actual parameters,
- a string parameter type has been introduced in which one-dimensional character arrays or substrings thereof may be passed as parameters. Such strings and their constituent characters are considered as "read only",
- procedures may be compiled separately,
- separately compiled procedures can be accessed through a declaration with the procedure block replaced by "extern",
- most option selectors ((*$M+*) etc.) are settable by switches in the MCR command line (version 5 December -77).

8. MEASUREMENTS. The compiler requires a 32 Kwords partition (at least 26 Kwords for small programs). Compilation speed is about 300 char's per second. In a 64 Kwords partition using PLAS under RSX-11M increases speed to about 3000 characters per second.

9. RELIABILITY. Excellent. The compiler is now in heavy use at five sites, and is distributed to 17. Only minor errors have been found since July -77. First version released April -77. Latest version December -77.

10. METHOD OF DEVELOPMENT. The crosscompiler for PDP-11 running on DEC-10 produced by Bron et al was used as input. It is written in Pascal and developped from Pascal-P. This compiler was modified to generate object code linkable under RSX/IAS and to give access to the file system of RSX/IAS. When the crosscompiler was finished it compiled itself and transferred to PDP-11. The implementation effort until now is about 7 manmonths.

11. LIBRARY SUPPORT. Separate compilation allowed. Possible to use external procedures written in FORTRAN (or assembler). The December -77 version also gives: Automatic copy of text from library into source program ("include"); execution frequency measurements; execution trace; option selectors ($R- etc) settable by switches in the MCR command line. Next version (spring -78) will also include a symbolic post-mortem dump and an interactive source-level debugging package (mainly copied from DEC-10 Hamburg compiler).

DEC PDP-11 (OMSI) (formerly ESI)
---

0. DATE/VERSION. 77/12/26; "OMSI Pascal-1" (formerly "ESI Pascal").

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. Oregon Minicomputer Software, Inc. (OMSI); 4015 SW Canyon Road; Portland, OR 97221; 503/226-7760. Implementors: John Ankcorn, Don Baccus, and Dave Rowland.

2. MACHINE. Any model Digital Equipment Corp. PDP-11 or LSI-11.

3. SYSTEM CONFIGURATION. Minimum of 16K words. Operates under RT-11, RSTS/E, or RSX.

4. DISTRIBUTION. Compiler, support module, cross referencer, text editor and instruction manual available for $1500 ($995 for educational use). Available on 9 track 800 bpi magnetic tape, or DEC cartridge disk.

5. DOCUMENTATION. Over 70 page machine retrievable instruction manual. Currently (76/11/02) working on more.

6. MAINTENANCE. One year of unlimited fixes and updates, followed by annual subscription service. (* Reported by users that "vendor seems to be responsive in terms of support". *)

7. STANDARD. Full standard plus extensions: additional features for real-time hardware control; separate compilation of procedures; Macro (assembler) code in line insertion; actual core addresses of variables can be fixed (giving access to external page I/O addresses at the Pascal level.

8. MEASUREMENTS.
compilation speed--About 3500 characters /second, on the PDP-11 model 05.
compilation space--very economical-it can compile 3000 line programs in 28K on PDP-11/40. No overlays are used in the system.
execution speed--about twice as fast as the DEC FORTRAN IV and many times faster than DEC BASIC. A worst-case 'number-cruncher' example ran at 40% faster than the DEC original FORTRAN.
execution space--very economical-much of the space improvement over DEC FORTRAN is due to the smaller support module for Pascal.

9. RELIABILITY. Excellent--far better than DEC FORTRAN. In use since 75/11. Over 60 installations, and growing steadily.

10. DEVELOPMENT METHOD. Single pass recursive descent compiler written in Macro-11. Hand-coded based on University of Illnois bootstrap (with extensive changes) in about two person-years of effort. First compiler written by both implementors. Compiler translates source into Macro-11 which is then assembled and linked to the support module for execution.

11. LIBRARY SUPPORT. Separate compilation of procedures with load-time insertion and linkage is implemented.

DEC VAX-11/780
---

0. DATE/VERSION. 78/03/27.

1. Implementor/Distributor/Maintainer.

Implementor: Professor Hellmut Golde
Department of Computer Science
University of Washington
Seattle, WA 98195
Tel. 543-9264 (Area Code 206)

2. Machine: DEC VAX-11/780 in native mode.

3. System Configuration: DEC VAX-11/780 under the VMS Operating System.

10. Development Method:

The compiler will be derivative of the CDC 6000/CYBER compiler.
The compiler will be transported to the VAX system via cross-compilation.

Hewlett Packard HP-2100, 21MX (Trieste)

Paolo Sipala (Istituto di Elettrotecnica; Universita di Trieste; Via Valerio, 10; 34127 TRIESTE; Italy) wrote on 78/03/20:

I have recently completed a Pascal-S compiler/interpreter for the HP 2100/21MX computer, running under DOS-IIIB. I enclose the Documentation Form accompanying the submission of the program to the Hewlett-Packard Software Center, Contributor Section (11000 Wolfe Rd.; Cupertino, CA 95014), through which the program should be available for distribution in the near future.

To summarize the data in the form, the system requires a 11K main core area (so it might fit into a 16K system, if the resident DOS modules are kept to a minimum, but 24K is more comfortable); there are separate versions for non-EAU, EAU, and floating point options machines. It is not noticeably slower than the standard compilers while compiling, and not worse than the standard interpreter (Basic) while interpreting. It has been subjected to rather limited testing (a few dozens programs from the Pascal Manual) and is being now offered to students here for their exercises.

Until the program becomes available through HP Software Contributors Center, I might send a copy of the program to those who request it by enclosing the price of the mailing (the weight is about 2 lbs.).

Hewlett Packard HP-3000
---

0. DATE/VERSION. 78/04/15.

1. DISTRIBUTOR: The system is available in the HP-3000 Contributed Library, Volume 4. Contact your local sales office, or write:
Hewlett-Packard Company
Contributed Software
P.O. Box 61809
Sunnyvale, CA 94088

IMPLEMENTOR: Robert A. Fraley
Hewlett-Packard Laboratories
3500 Deer Creek Rd.
Palo Alto, CA 94304

MAINTAINER: Maintenance is not provided, but errors may be mailed to the implementor.

2. MACHINE: HP-3000.

3. SYSTEM CONFIGURATION: MPE.

4. DISTRIBUTION: The system will be available through the HP-3000 Contributed Library in June, 1978.

5. DOCUMENTATION: Sparse machine-readable documentation is included.

6. MAINTENANCE: None. Error reports may be sent to the implementor, and may be fixed in later releases. Full file support and separate procedure compilation may be available in a future release.

7. STANDARD: Falls short of the standard due to the sorry state of the P-compiler. Measures are being taken to improve the P-compiler.

8. MEASUREMENTS: No specific measurements made. Some improvement will be available in a future release. The compiler is somewhat awkward to use, due to the P-code intermediate. Compilation and link-edit of the compiler operates at 125 lines per minute.

9. RELIABILITY: Good. Currently in use at nine installations.

10. DEVELOPMENT METHOD: Bootstrapped from a P-compiler by Grant Munsey, Jeff Eastman, and Bob Fraley. Compiles to HP-3000 machine code.

11. LIBRARY SUPPORT: None yet.

IBM 360/370 (Australia)

Cox/Tobias letter(s).

AUSTRALIAN ATOMIC ENERGY COMMISSION

NUCLEAR SCIENCE AND TECHNOLOGY BRANCH

RESEARCH ESTABLISHMENT, NEW ILLAWARRA ROAD, LUCAS HEIGHTS

TELEGRAMS: ATOMRE, SYDNEY
TELEX: 24562
TELEPHONE: 531-0111
IN REPLY PLEASE QUOTE: JMT.mwb

ADDRESS ALL MAIL TO:
AAEC RESEARCH ESTABLISHMENT
PRIVATE MAIL BAG, SUTHERLAND 2232
N.S.W. AUSTRALIA

13 March, 1978.

Dear Andy,

Just a note to let you know the current status of Pascal 8000 for IBM360/370 computers.

We are currently distributing version 1.2 of the system. The differences between 1.2 and our earlier 1.1 distribution include a few bug fixes (there were some installation problems on VS1), and a few new features, such as the inclusion of the characters _, [,], &, | and ¬. We are very happy with the reliability of the system, - this too has now gone from very good to excellent. We very much enjoy the reports received from Hal Perkins at Cornell University. His letters to us are somewhat overwhelming (average - 10 pages), and we really appreciate his feedback. We only wish more sites would drop us a note as to their progress.

We have now shipped a system to Judy Bishop at the University of the Witwatersrand, and we enjoy corresponding with her. We hope that Pascal 8000 will meet all of her expectations, and we look forward to hearing her comments on the system.

Judy passed on your thoughts of setting up an American distribution centre; we somehow feel that this may cause more problems than it is worth. We cannot understand why people in the U.S. are afraid to contact us directly - perhaps they doubt that Australia has an adequate postal system (no, letters are not delivered in the pouches of kangaroos). We had some delays in the processing of orders earlier in the year, mainly due to our deciding to drop the non-disclosure agreement. We have since established a rather smooth distribution setup and have involved another person to handle the answering of correspondence, mailing of tapes, etc. We answer all initial enquiries with an order form and a copy of our Reference Manual; on receipt of the order form, we despatch the system and invoice the organisation for $A100 at a later date. The time from our receiving an order form to despatching a system should be no longer than five days, i.e. the system should be in the recipient's hands three to four weeks after they post their order, provided there are no unforeseen delays. We feel that this is not an unreasonable period of time.

We see a number of problems arising if we were to establish alternative distribution centres - who supplies and copies the tapes, who prints the manuals, who fixes the bugs, who answers technical questions, who supplies the updates, and so on. We are, however, willing to hear of any strong arguments supporting such a centre.

We now have 40 Pascal 8000 sites operational; those on Version 1.1 automatically received the updates to bring them up to 1.2. We anticipate more orders as a result of our dropping the non-disclosure agreement. We are planning a Version 2, but cannot anticipate its release.

We have sent a copy of our latest Reference manual to you under separate cover to add to your undoubtedly desk-high pile of manuals. We hope you find it of interest.

And finally, let us say how much we appreciate your efforts in the Pascal Users Group, and your words of encouragement for Pascal 8000.

Best regards,

Gordon

Jeff

Gordon Cox

Jeffrey Tobias
Systems Design Section

Intel 8080 (Ann Arbor)

Jim Rogan (Comshare; Wolverine Tower; 3001 S. State St.; P.O. Box 1588; Ann Arbor, Michigan 48106; 313/994-4800) wrote on 78/2/16 that Comshare "can currently cross-compile [Pascal] source for the Sigma 9 and an INTEL 8080 machine."

The following is an overview of COMSHARE'S PASCAL compiler system. It is presented and outlined with respect to a "package" that could be delivered, from which you could implement the system on your machine.

I. History

Comshare's PASCAL compiler was originally a bootstrapped version of the portable Pascal 'P' compiler. The impetus for the compiler project was to provide the company programmers with a state-of-the-art language from which they could write readable, easily maintainable, efficient programs. Along with these objectives, machine independent programs were sought and this feature was designed into the compiler system. It was decided that the portable PASCAL compiler, with some major modifications would be a reasonable base to start from.

II. PASCAL Language Modifications

In areas where the language definitions were found undesirable or inadequate, modifications were made. The areas primarily effected were the I/O and scoping structure. In brief, the standard INPUT and OUTPUT files were eliminated along with the GET and PUT operations. They were replaced with 'FILE' declaration types, OPEN and CLOSE primitives. The READ and WRITE statements were modified and binary file operators were added.

Also, the scoping mechanism was eliminated (ie. all procedures are considered on the same "level") because it was contrary to structured programming principles, allowing for pathelogical data references, etc. All the basic language statements, control structures and the declaration sections are the same or enhanced.

Note: Since the language has become an off-color PASCAL, the name has been changed to PASTEL.

III. Operational Characteristics

A. System design

Comshare's PASTEL compiler is a three phase (pass) language processor system. The first phase is a machine independent phase, the second and third are target machine dependent. The process basically consists of translating a source program into a machine independent intermediate form of code for a hypothetical stack computer (see "The PASCAL <P> Compiler: Implementation Notes", NORI, AMMANN, JENSEN, WIRTH). Then, for any given machine, a code generator for it converts the intermediate code into hard machine code.

The first phase (compiler) has three functions: to syntatically analyze the source program; to translate the program into a form of "assembler-like" intermediate instructions (P-codes) and directives; and finally to perform the static and dynamic data allocation.

The second phase also has three discrete functions: to translate the P-codes into a form suitable for code emission (triples) and optimization; optimization, and the emission of the machine instructions themselves.

The third phase is necessary for portability purposes. It is simply running the target machines assembler over the generated symbolic instruction to produce a loader compatible relocatable binary object file. The process can be viewed as follows:

Please note that for the ultimate "production" compiler, one would want to eliminate the third phase by adding a module to the code generator to emit relocatable binary directly. The emission of the symbolic meta symbol could then be an "optional" feature for the compiler to aid in analysis and debugging of the systems you apply this language to.

IV. Compiler Specifications and Limits

Aside from our current, and most highly recommended compiler, we have available two predecessors from which it evolved. A list of pertinent facts relating to each version follows. All timing estimates are based relative to our XDS 1968 FORTRAN compiler which is a one pass processor written in a low-level language.

A. PASCAL THREADED -CODE INTERPRETER

This version implements the language essentially as described in Jensen and Wirth's User Guide / Report.

- uses the ETH character set.
- no external procedures.
- generates macro's that are assembled into threaded calls to runtime interpreter.
- very limited I/O facilities.
- do not know the specific core requirements but I'm sure it's no problem.
- runs at approx. 10.0 times the speed of FORTRAN.

B. PASCAL COMPILER

This is a "real" compiler in the sense that all interpreter functions were eliminated and replaced with a code generation phase. The general enhancements are as follows:

- uses EBCDIC character sets.
- augmented P-code set.
- I/O still limited but faster.
- full set of data types.
- stack machine operations are simulated in registers where possible.
- maximum core requirement is approx. 40K words (??).
- language complement is very close to "standard" PASCAL.
- runs at approx. 3.0 times the speed of FORTRAN.

C. CURRENT PASTEL COMPILER

This compiler is very close to our version of a finished product. It has a lot of enhancements in the areas of usability, efficiency and machine independence. It contains user-oriented features, a new and optimizing code generator and cross-compile abilities working for a Sigma-9 and an INTEL 8080 micro computer. Its language and feature descriptions can be reviewed in the enclosed preliminary reference manual. They are highlighted by:

- compiler option recognition.
- language processor control program.

- full complement of I/O facilities that are very efficient.
- external non-PASTEL procedure linkage.
- dynamic arrays.
- static and dynamic data allocation.
- packed data structures and data allocation options.
- a manual.
- very good documentation (in English) of the internals.
- 'LOOP' statement.

COMPILER:

source language is PASTEL and is approx. 6600 lines of code + comments; object size is 31K words;

CODE GENERATOR:

source language is PASTEL and is approx. 2900 lines of code + comments; object size is 17K words;

RUNTIME:

source language is meta-symbol with a little PASTEL and is relatively small in size; requires 1.8K words of core for code + storage buffers.

- runs at approx. 1.5 times the speed of FORTRAN.
- good testing procedures for releases of new versions.

V. Implementation Considerations (MACHINE X)

1. Must develop a code generator targeted for your specific machine. This would basically involve modifying the code emission routines within our "skeleton" code generation phase processor.

2. A runtime must be developed to support the emitted calls for I/O and a few miscellaneous functions. The runtime is approximately 90% I/O routines interfacing with the operating system, 6% house keeping routines and the remainder consists of miscellaneous system functions to support language features. These routines could be written in PASTEL and developed concurrently with the code generator using COMSHARE timesharing services or, could be done on your given system in any language desired.

3. The compiler "process controller" will need some minor changes to do the appropriate subprocess start-up, termination and communication control.

4. Modifying the code generator mechanisms to incorporate the new procedure calling protocol for interfacing with non-PASTEL languages.

VI. Implementation Considerations (XEROX SIGMA 9)

1. The compiler and code generator can be directly assembled by the meta-symbol processor, since they are coded in PASTEL.

2. The runtime will need some modifications for interfacing with CPV. These changes should be strictly limited to the I/O interface. Our system does not have a 'DCB' concept and it would be necessary to install these into the runtime to do the physical data transfers. All the other code is in the Sigma-9 instruction set and PASTEL.

3. The process controller will need some re-writes to do the appropriate subprocess startup, termination and communication.

4. Modifying the code generator to incorporate the new procedure calling protocol for interfacing with external, non-PASTEL languages.

Intel 8080 (Munich)
---

1. Implementors:
   D. Krall, W. Remmele, U. Weng
   Siemens AG
   ZT ZFE FL SAR 121
   Otto-Hahn-Ring 6
   D-8000 München 83
   Germany

2. Machine:
   Intel MDS800 (under ISIS II) with 8080 processor;
   Host-machine: Siemens 4004/151 (or any with a Pascal-system)

3. System configuration:
   64 K Byte RAM, Floppy Disk, Console;
   A possibility to transport the intermediate code from the host-computer to the MDS.

4. Distribution:
   No final decisions made yet - contact implementors.

5. Documentation:
   A manual is available (written in German). Updating is done with each new version.

6. Maintenance:
   No final decisions made yet.

7. Standard:
   No changes to the Standard. The attribute packed is ignored.
   Current restriction: No functions as parameters.
   Extension: external procedures.

8. Measurements:
   No measuring has been done yet.

9. Reliability:
   Seems to be excellent: No known errors.

10. Development method:
    Compiler derived from ETH's P4; new Assembler, Linkage-Editor and Interpreter. A resident version for the MDS800 is in work.

Interdata 7/16, 8/32

Marinchip Systems

computer hardware and software
16 Saint Jude Road
Mill Valley, Ca. 94941
(415) 383-1545

March 22, 1978

Timothy M. Bonham
D605/1630 S. Sixth Street
Minneapolis, MN 55454

Dear Tim:

The many extended conversations that went on at the Computer Faire resulted in some scrambled information being received. The Interdata 7/16 Pascal compiler that I have a copy of is the cross-compiler for the Univac 1100 that was done by Mike Ball of the Naval Ocean Systems Center (formerly Naval Undersea Center) in San Diego. His compiler is a version of the Hartmann / Brinch Hansen compiler with the interpretive code generation pass removed and three phases added which generate Interdata machine code. He has both the Sequential and Concurrent compilers running (with common code generators), and an Interdata kernel for Concurrent Pascal. The compiler was written with "source code configuration" statements in it so that either a Univac or an Interdata version can be generated by processing a common source with a Pascal program. As of the time I got a copy of the compiler (about a year ago), only the cross-version was running, and the bootstrapping to the 7/16 was not yet complete. I have not talked with Mike to find out whether the compiler is yet running on the 7/16 itself. I do know that the Univac version was producing workable 7/16 code.

I understand that Mike now has the Interdata 8/32 version compiling itself on the 8/32. Apparently the 8/32 version is extended beyond the original 7/16 design, and may be moved back down to the 16 bit series. In any case, the person to contact about all this stuff is Mike, not me. (Mike is a PUG member, and his address is listed in the roster).

I got a copy of Mike's compiler in the hopes of using it as a base to build a true compiler for the T.I. 9900 machines I am building. At present, we are taking a hydra-headed approach to Pascal. We are looking at the UCSD Pascal, and also at bootstrapping the original Concurrent Pascal via the interpretive code. Once we have a workable interpretive Pascal, we will do the true compiler if we feel the need.

I hope this information has been of use. I will send in an implementation checklist for my 9900 Pascal as soon as it is running.

Sincerely,

John Walker

Northwest Microcomputer Systems 85/P

Northwest Microcomputer Systems (121 East 11th; Eugene, OR 97401; 503/458-0626) is marketing an Intel 8085A based system which supports UCSD Pascal -- see DEC LSI-11 (San Diego). Hardware includes two floppy disks (1 megabyte), 54K bytes of 450ns static RAM, a keyboard, 24 by 80 char CRT, 2 serial ports, and several parallel ports. The price is $7495. Also included is the CP/M operating system.

Prime P-400

0. DATE/VERSION. 78/03/01. GEORGIA TECH PRIME 400 PASCAL COMPILER.

1. IMPLEMENTOR/DISTRIBUTOR:
   Professor Richard J. LeBlanc
   School of Information and Computer Science
   Georgia Institute of Technology
   Atlanta, Georgia 30332

2. MACHINE: PRIME 400

3. SYSTEM CONFIGURATION: PRIMOS IV Operating System, 64V mode, 128K bytes minimum.

4. DISTRIBUTION: A first release of the compiler should be available by July 1978. Further details are not yet finalized.

5. DOCUMENTATION: None yet available beyond PASCAL-P documentation.

6. MAINTENANCE POLICY: Error reports from users will be encouraged. Details concerning distribution of corrections and updates not yet finalized.

7. LANGUAGE IMPLEMENTED: PASCAL-P subset of Standard PASCAL.

8. MEASUREMENTS: Not yet available.

9. RELIABILITY: Not yet available. (It is intended that this implementation project will eventually result in a highly diagnostic and very reliable compiler.)

10. DEVELOPMENT METHOD: The code generation parts of the PASCAL-P4 compiler are currently being rewritten to generate PMA calls to interpreter routines. This will then be assembled and linked with those routines, producing a "threaded code" interpretive program. The compiler will be bootstrapped to the PRIME using PASCAL-6000 on a CDC CYBER 70.

11. LIBRARY SUPPORT: None yet available. Support for external procedures written in PASCAL, FORTRAN and PMA will be an early addition to the compiler.

12. FURTHER DEVELOPMENT: As soon as this first version is available, work will begin on adding code generators to produce directly executable code. At the same time, implementation of full PASCAL will be under development. Many of the diagnostic features currently found in the UW-PASCAL compiler for UNIVAC 1100 machines will also be included.

INDEX TO IMPLEMENTATION NOTES

POLICY: PASCAL USER'S GROUP (78/04/15)

Purposes: Pascal User's Group (PUG) tries to promote the use of the programming language Pascal as well as the ideas behind Pascal. PUG members help out by sending information to Pascal News, the most important of which is about implementations (out of the necessity to spread the use of Pascal).

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

Membership: Anyone can join PUG: particularly the Pascal user, teacher, maintainer, implementor, distributor, or just plain fan. Memberships from libraries are also encouraged.

See the ALL PURPOSE COUPON for details.

FACTS ABOUT Pascal, THE PROGRAMMING LANGUAGE:

Pascal is a small, practical, and general purpose (but not all-purpose) programming language possessing algorithmic and data structures to aid systematic programming. Pascal was intended to be easy to learn and read by humans, and efficient to translate by computers.

Pascal has met these design goals and is being used quite widely and successfully for:

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

Pascal is a leading language in computer science today and is being used increasingly in the world's computing industry to save energy and resources and increase productivity.

Pascal implementations exist for more than 62 different computer systems, and the number increases every month. The Implementation Notes section of Pascal News describes how to obtain them.

The standard reference and tutorial manual for Pascal is:

Pascal - User Manual and Report (Second, study edition)

by Kathleen Jensen and Niklaus Wirth

Springer-Verlag Publishers: New York, Heidelberg, Berlin

1978 (corrected printing), 167 pages, paperback, $6.90.

Introductory textbooks about Pascal are described in the Here and There Books section of Pascal News.

The programming language Pascal was named after the mathematician and religious fanatic Blaise Pascal (1623-1662). Pascal is not an acronym.

Pascal User's Group is each individual member's group. We currently have more than 1923 active members in more than 35 countries. This year Pascal News is averaging more than 150 pages per issue.

Return to:

University Computer Center
227 Experimental Engineering Building
208 Southeast Union Street
University of Minnesota
Minneapolis, Minnesota 55455 USA

return postage guaranteed

address correction requested