PASCAL USER'S GROUP

USER'S GROUP

PASCAL NEWSLETTER

NUMBER 7

COMMUNICATIONS ABOUT THE PROGRAMMING LANGUAGE PASCAL BY PASCALERS

FEBRUARY, 1977

TABLE OF CONTENTS

0	POLICY
1	EDITOR'S CONTRIBUTION
3	HERE AND THERE WITH PASCAL
3	News
4	Conferences
4	Books
4	Roster
8	ARTICLES
8	"Life, Liberty and the Pursuit of Unformatted Input" - D. W. Barron and J. M. Mullins
9	"Pascal Printer Plotter" - Herbert Rubenstein
17	"Yet Another Look at Code Generation for Pascal on CDC 6000 and Cyber Machines" - Lawrence A. Liddiard
24	OPEN FORUM FOR MEMBERS
27	IMPLEMENTATION NOTES
27	General Information
27	Checklist
27	Portable Pascals
29	Software Writing Tools
29	Compilers
45	ALL PURPOSE COUPON

PASCAL USER'S GROUP POLICIES

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

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

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

Cost of membership per academic year is $4 and may be sent to:
Pascal User's Group/ %Andy Mickel/University Computer Center/227 Exp Engr/
University of Minnesota/Minneapolis, MN 55455 USA/ phone: (612) 376-7290

In the United Kingdom, send £2.50 to:
Pascal Users' Group/ %Judy Mullins/Mathematics Department/The University/
SOUTHAMPTON/SO9 5NH/United Kingdom/ (telephone 0703-559122 x2387)

PASCAL NEWSLETTER POLICIES

The *Pascal Newsletter* is the official but informal publication of the User's Group. It is produced quarterly (usually September, November, February, and May). A complete membership list is printed in the November issue. Single back issues are available for $1 each. Out of print: #s 1,2,3
#4 available from George Richmond/Computing Center/U of Colorado/Boulder/80309

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

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

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

POLICY

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

- Andy Mickel, editor, John P. Strait, associate editor, December 29, 1976.

UNIVERSITY OF MINNESOTA
TWIN CITIES
University Computer Center
227 Experimental Engineering Building
Minneapolis, Minnesota 55455
(612) 376-7290

EDITOR'S CONTRIBUTION

PART I - Promoting Pascal Usage

This concerns large computer centers with many users who can be thought of as consumers of computer software. If we consider only the users (including the computer center staff) who write programs in a programming language (rather than, say, use a package such as SPSS - Statistical Package for the Social Sciences) then the problem comes down to converting users from established and less desirable languages such as FORTRAN, ALGOL-60, BASIC, and PL/I to Pascal.

William Waite's valuable guest editorial in Software, Practice and Experience Vol. 3, pp. 195-6, provides many of the guidelines. He states that the support available for a language is more important than the features in the language itself. He likens languages to "species which inhabit certain ecological niches." The analogy has been quite useful in identifying ways of promoting Pascal at the University of Minnesota (see also Editor's Contribution Pascal Newsletter #5). Waite goes on to say that only when a more efficient competitor for a given language's life support systems is introduced into the ecosystem will the given language be ousted. At many universities, then, Fortran is a very healthy species which occupies, according to Waite, "the niche created by 'scientific' computation" (perhaps numerical computation is a better term).

The specific checklist given by Waite includes:

1) Is a sufficient library of standard procedures available?
2) Can the relevant ones by extracted automatically?
3) Is there easy interface with computer system utilities such as the file manager and the sort/merge package?
4) Is there easy interface for assembler if performance measurement shows critical procedures which are not adequately optimized by the translator?
5) Will consultation and programming assistance be available if I have problems with the language?
6) Is there sufficient program preparation equipment (interactive terminals, keypunches) with the proper character sets?
7) Will the computer system provide reasonable turnaround for programs written in the language?

If we take an "advocacy" position with respect to Pascal in order to promote its use, these and other aspects come into play. At the University of Minnesota, the following proved very useful since we began using Pascal in 1972.

1) The prevailing view that a language processor should be given support only proportional to its usage cannot be tolerated. To give a language processor a fair chance, it must be actively promoted for awhile and then its acceptance evaluated. The intrinsic merit of the language and its processor should be the determining factors.
2) Proper user documentation helps promote the language. Pascal starts with Pascal User Manual and Report. In addition a local computer center should provide two documents: A) a double-sided one sheet handout describing the local Pascal facilities to satisfy the numerous requests for information made by walk-in users. On this sheet should be a date, place, author, and a description of the purpose of the sheet. The installation's Pascal facilities include references for more information (including P,UM&R and the other document described below); a description of the compiler or interpreter, its origin, reliability, commonly used options, and how to use the system in both batch and interactive modes (this includes command sequences and a description of the form of compile-time and run-time messages); and a small character set table if substitutions need to be made. B) a larger (20-30 page document) both in printed copy and in machine retrievable form which includes an introduction describing the scope of the document; information about the programming language Pascal, its history, uses, implementations, general and short description of its semantics and syntax; a history of the particular implementations (Pascal compilers) the computer center is running (current features and future developments); a description of the implementation, how it works, the specific definition of the sizes of scalar types, predefined (non-standard) identifiers, compiler options and switches, differences between this implementation and the standard; how to run programs under the implementation both in interactive and batch modes, program preparation, character sets, the commands to invoke the compiler or interpreter; guidelines, hints and cautions for effective usage, error messages, how to use software writing tools such as the cross-referencer, prettyprinter, source language editor, etc.; a detailed annotated list of references: introductory texts, reference manuals, books on applications, and sources of current information.
3) Enough people must be available for helping users with problems in their programs.
4) Publicity for Pascal to keep it constantly in the eye of the computer center user community: 2-3 week short courses in the language; articles in the computer center newsletter promoting the language as well as announcing planned changes in versions;

EDITOR'S CONTRIBUTION

living, "useful", well written, and simple example programs to show the language at its best: (e.g. a fancy calendar program - what better way to get people's interest?).

5) When converting persons remember that: don't waste time converting Fortran and assembly language programmers who are overly concerned with machine efficiency. They will persist in their habits. Pascal's strengths lie in reducing the number of runs one has to make on program development, and because Pascal is nearly as efficient in terms of machine time as Fortran, less actual computer time is used; new programmers are the best bet, the computer science department can made a great contribution by teaching the language to new programmers and using it in other parts of the curriculum; urge people to write new programs in the language rather than getting them to convert old programs - although the latter may produce converts astounded at better solutions arrived at because they were able to think more clearly and conceive of a better algorighm in a systematic language (Pascal).

After bringing Pascal from nowhere to third out of 20 languages in four years we feel that support for Pascal is sufficient to survive and "ecological counterattack" and will continue to erode Fortran's base of users as we satisfy more of William Waite's principles - particularly in the area of libraries of procedures.

PART II - Pascal and Standards

There has not been time to receive the reaction to the proposals which appeared in Newsletter #6. Formal standardization of Pascal as it is now (by an official standards organization such as ISO or ANSI which could then have economic enforcement in the marketplace) is pretty straightforward. Changing Pascal is certainly a political problem, and even deciding how to pick a committee and when and where it could meet may prove to be overwhelming. A lot of issues regarding specific changes are not clear cut. We are a loose union, not a tight band, of devotees.

What we should concentrate on is conventionalizing the few recurring extensions in the various implementations of Pascal. We can use the Newsletter for that. I cannot overemphasize my conviction that much, careful consideration was given to what features were left out of Pascal as it has evolved. We must always go back to the design goals of compactness, vehicle for portability, vehicle for teaching systematic programming, and a tool to write efficient production programs. We should not use Pascal for purposes it was not intended (such as writing an operating system). One should not misuse or break any tool. Note that other languages have been designed for those tasks (in the case of operating systems with the need to express concurrent processes we have Brinch Hansen's Concurrent Pascal, Hoare's SIMONE, and a rumor about Wirth's MODULA).

I hope my editorial in PUGN#6 did not seem too confused - I was trying to be compromising and all-encompassing and I still lack a lot of answers.

One final note: the wholeSALE bending of Pascal to make it conform to conventions of Burroughs ALGOL (as described in the report: "Burroughs Pascal: Some Implementor's Thoughts) is alarming. Why have a different language (Pascal) available which can bridge Burroughs users to software written on other machines if one adopts so many features from an existing Burrough's language: Burroughs Extended ALGOL? One might as well stick to Burroughs ALGOL. At best it's PascALGOL! Why get so upset? Implementors are not operating in a vacuum; they affect all of us on the issue of standards and portability.

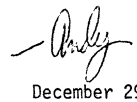
PART III - PUG and Pascal Newsletter

PUG now has 598 members in 24 countries and 44 states. We have been growing steadily at the rate of 60 members/month since we started. By June, then, we should have almost 1000 members. This will cause us financial anguish in the form of growing pains. So the result is very poor service regarding back issues. Sorry. It's hard to plan ahead with the small budget we have.

Speaking of slow distribution, as I write this I'm sure some of you have not yet received #6. This will make #8 longer when the reaction arrives. #5 and #6 were big and were mailed overseas by air so that we could get the Pascal movement back on track. We also printed nearly everything that came to our attention. This satisfied the individual urgent questions we had been receiving about newsletters and implementations. This did cost us money and that is why #7 is small and will not go by air overseas. We are happy to report that things seem to be much improved as the information has gotten out via the newsletter. We have tried to take an advocacy position with the goal of furthering Pascal. We hope that this hasn't offended anyone.

The bright area is the UK distribution center which is getting lots of members and has eased the overseas distribution load for us.

Thanks for all the compliments regarding the newsletter. We shall try to "keep up the good work." But we do need the help we requested for handling some departments of the newsletter and functions of the User's Group written in PUGN#6.



December 29, 1976

HERE AND THERE WITH PASCAL

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

NEWS (ALPHABETICAL BY LAST NAME)

Jim Fontana, Control Data Corporation, 3519 W. Warner Ave., Santa Ana, CA 92704 (PUG member): "... I would appreciate hearing from any PUG members who know where tapes and listings for any version of Concurrent Pascal and the Solo Operating System are available." (*12/20/76*)

Bill Hopkins, Dept. of Comp. and Info. Sci., University of Pennsylvania, Philadelphia, PA 19174 (PUG member): "...I am at the moment Pascal-less. Do you have any Univac (nee RCA) Spectra 70 installations on the mailing list? We're going to a 9070 Univac in the Christmas break but it will be software compatible - still Univac VMOS. I understand that the Stony Brook 360 compiler was being converted to VMOS at Georgia State, but I haven't been able to raise anyone there. Would appreciate any pointers." (*10/2/76*)

Pei Hsia, Computer Science Program, The University of Alabama, Huntsville, P.O. Box 1247, Huntsville, AL 35807 (PUG member): "...We currently have a Univac 1100/10 system and are planning to implement a Pascal in order to teach some basic programming concepts of it." (*12/6/76*)

Olivier Lecarme, I.M.A.N., Universite de Nice, Parc Valrose, Nice Cedex 06034, France (PUG member): We received the 173 page proceedings from a "Pascal Days" conference held in France during June 5th and 6th, 1975. Olivier co-organized the conference which discussed implementation, development, and applications in 7 working sessions. C. Girault was the other co-organizer. He is at the Institut de Programmation, Universite Paris VI. Other participants were: J. Cea, D. Thibault, J.L. Bouchenez, B. Lohro, C. Girault, M. de la Croix, M. Galinier, J.L. Pouzin, J. Farre, M. Gauthier, A. Tisserant, G. Terrine, P. Maurice, J.P. Partouche, G. Tassart, D. Thibault, B. Lang, D. Fortier, D. Gurtner, J.L. Nebut, R. Rousseau, G. Terrine, Nguyen Van Lu, B. Robinet, M. Dupras, C. Precetti, and J. Bezivin. (*12/13/76*)

John L. Norstad, Northwestern University, Vogelback Computing Center, 2129 Sheridan Rd., Evanston, IL 60201 (PUG member): "...I am in the process of installing release 2 of Pascal 3.4 on our 6400 at Vogelback. I'm happy to see the improvements, especially the smaller core requirements. I've heard that Larry Liddiard has been trying to reduce the size....Pascal is not receiving the attention it deserves at Northwestern. I've added a Pascal mode to our interactive text editor as a first attempt in making Pascal use more attractive. We have several students interested in the language and one course this quarter is using Wirth's Algorithms + Data Structures = Programs and running lab problems on our batch system...." (*11/22/76*)

Fleming M. Oliver, 213 Weddell #12, Sunnyvale, CA 94086 (PUG member): "Has anyone implemented Pascal on an Interdata 8/32? If so, who? I'm interested in the details." (*12/3/76*)

Ate Phung, Krefelder Str. 23, D-5100 Aachen, Germany (PUG member): "...As I am dealing with the implementation of Pascal to a Xerox Sigma 3, please send me any further informations about your experiences." (*12/8/76*)

Dean Schulz, INTEL Corporation, 3065 Bowers Avenue, Santa Clara, CA 95051 (PUG member): "...please inform me of all implementations of Pascal for microprocessors of which you are aware." (*11/29/76*)

Stephen C. Schwarm, E.I. du Pont de Nemours Co., 101 Beech St., Wilmington, DE 19898 (PUG member): "...we now have 150 persons on the DECUS (DEC Users Group) SIG Pascal mailing list! I'm preparing a newsletter which I will send shortly, and forward a copy to you as well....there is a big demand for an RSX-11 compiler. ...P4 is great for implementing! The problems with PDP-11s is that the compiler uses too much data space: 5K for the stack and 16K dynamic storage. On the other hand P-code translates well to PDP 11 code: 48K and this can be reduced to 32K with some hand optimization...." (*1/3/77*)

Manfred Seifert, Informatik III, Universitat Karlsruhe, Zirkel 2, D-7500 Karlsruhe, Germany (PUG member): "Our institute is running a PDP 11/45 and two PDP 11/34s likked together via DL-11E and DL-11B. We are using RSX-11M and RSX-11S, communication with DECNET-11 M/S. We are interested in a compiler under RSX-11M, or a compiler easily changeable to RSX support. We will use Pascal for programming IPC and decentralized control software in our local network. (*11/15/76*)

Rick Thomas, 408 Domer Ave. , Takoma Park, MD 20012 (PUG member): "...We have a Univac 1108 installation at the University of Maryland and we just installed the Pascal/1100 compiler written at Naval Undersea Center by Mike Ball. We would be especially interested in an automatic Pascal source code indenter, for displaying the structure of someone else's code...." (*12/17/76*)

University of Washington, Seattle Computer Center Newsletter, August, 1976 shows that language processor statistics on its 6400/Cyber 73 system put Pascal at #2 behind Fortran (up from 5th the previous fiscal year). There was a decline in Simula usage over the same period. The Pascal increase was 65% from last year.

Tom Zwitter, Advanced Development Div., Building B, Ohio Nuclear Inc., 6000 Cochran Rd., Solon OH 44139 (PUG member): "...I spoke with Stephen Schwarm regarding their efforts and they hope to make an RSX-11 version available..." (*11/29*)

HERE AND THERE WITH PASCAL

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

CONFERENCES

The Third Annual Computer Studies Symposium at the University of Southampton being organized by D.W. Barron and J.M. Mullins has added a speaker in the Application part of the program. Olivier Lecarme will speak on "Pascal and Portability". The symposium, entitled: Pascal, Implementation and Applications was fully described in Newsletter #6.

BOOKS AND ARTICLES

(* We really need someone to manage this section. *)

D.W. Barron reported in a letter dated 1 December, 1976 that there is a newly published book:

Introduction to PASCAL by C.A.G. Webster, Heyden, 1976. price: $11.00, £5.50, DM35.00.

ROSTER 1/4/77 (NEW MEMBERS, CHANGED OR CORRECTED ADDRESSES AND PHONES)

THOMAS G. MCGINTY
DEPT. 330
FOXBORO CO.
38 NEPONSET AVE.
FOXBORO MA 02035
(617) 543-8750 X2031

JOHN CASEY
DEPARTMENT OF MATHEMATICS
NORTHEASTERN UNIVERSITY
360 HUNTINGTON AVENUE
BOSTON MA 02115
(617) 437-2450

ROBERT E. WELLS
BOLT BERANEK AND NEWMAN INC.
50 MOULTON STREET
CAMBRIDGE MA 02138
(617) 491-1850

ROY A. WILSKER
27 BENEFIT STREET
WALTHAM MA 02154
(617) 899-6638

FRED EILENSTEIN
68 SPRING STREET
WATERTOWN MA 02172
(617) 924-2248

G. M. SHANNON
LINCOLN LAB
J-146
M.I.T.
244 WOOD STREET
LEXINGTON MA 02173
(617) 862-5500 X5719

KEN POLAKOWSKI
5D VILLAGE GREEN
BUDD LAKE NJ 07828
(201) 347-4375

RICHARD B KIEBURTZ
DEPT. OF COMPUTER SCI.
SUNY AT STONY BROOK
STONY BROOK NY 11794
(516) 246-5987

S. KAMAL ABDALI 12181
GARY S. ANDERSON 98043
ATTN: A.D.R. 38000 FRANCE
ATTN: CENTRAL LIBRARY 3001 AUSTRALIA
ATTN: COMPUTER CENTER USER SERVICES 98105
ATTN: COMPUTER UNIT CV4 7AL UNITED KINGDOM
ATTN: THE LIBRARY ISRAEL
HENRY R. BAUER III 82071
HERMAN BERG 53703
SCOTT BERTILSON 55455
JEAN BEZIVIN 35031 FRANCE
I. N. BLAVINS 5006 AUSTRALIA
C. BRON THE NETHERLANDS
J. A. CAMPBELL 2308 AUSTRALIA
JOHN CASEY 02115
JOHN E. COLLINS 55101
RICHARD CORE 94088
B. J. CORNELIUS HU6 7RX UNITED KINGDOM
LINDA E. CROLEY 94304
DENNIS DANCE 72204
DAVID DEMOREST 98124
PIERRE DESJARDINS 101 CANADA
FRED EILENSTEIN 02172
JACQUES FARRE 75230 FRANCE
EDWARD E FERGUSON 76201
TED FISHMAN 75222
DAVID C. FITZGERALD 91740
J. J. FLORENTIN WC1E 7HX UNITED KINGDOM
JIM FONTANA 92704
DAVID N. GRAY 78769
HILMAR GUTFELDT CH-3000 SWITZERLAND
C. C. HANDLEY 4000 SOUTH AFRICA
KAY A. HANSBOROUGH 87544
KAY HARRISON N2L 3G1 CANADA
KEVIN HAUSMANN 55113
S. T. HEIDELBERG 94550
RICHARD HENDRICKSON 55420
TERUO HIKITA 113 JAPAN
H.-J. HOFFMANN D-6100 GERMANY
TIMOTHY J. HOFFMANN 55455
WILLIAM C. HOPKINS 19174
PEI HSIA 35807
R. I. JOHNSON 58202
ED KATZ 70504
RICHARD B KIEBURTZ 11794
JOHN A. LAMBERT 2308 AUSTRALIA
ROBERT M. LANSFORD 91107
ROBERT A. LAWLER 55165
MIKE LEMON 61738
GEORGE LIGLER 75080
GARY LINDSTROM 84112
BRUCE LINK 87115
LEON LUKASZEWICZ 00901 POLAND
ORLANDO S. MADRIGAL 95926
LARS MAGNUSSON S-751 21 SWEDEN
CRAIG MAUDLIN 92037
KONRAD MAYER A-1150 AUSTRIA
THOMAS G. MCGINTY 02035
BRIAN MEEK W8 7AH UNITED KINGDOM
B. T. MITCHELL KY16 UNITED KINGDOM
CHARLES G. MOORE 48106
LARS G. MOSSBERG S-461 01 SWEDEN
BERNHARD NEBEL 2 GERMANY
BERGT NORDSTROM S-402 20 SWEDEN
JOHN L. NORSTAD 60201
FLEMING M. OLIVER 94086
PAUL O-BRIEN 97210
TRUMAN C. PEWITT 60439
ATE PHUNG D-5100 GERMANY
KEN POLAKOWSKI 07828
FRED W. POWELL 24401
GEORGE H. RICHMOND 80309
CLAES RICKEBY S-161 54 SWEDEN
N ROBINSON HA6 3DZ UNITED KINGDOM
DON H. ROWLAND 87109
MARK RUSTAD 55112
WILLIAM A. RUTISER 20052
HORST SANTO D-5205 GERMANY
ROSS D. SCHMIDT 55343
DEAN SCHULZ 95051
MANFRED SEIFERT D-7500 GERMANY
G. M. SHANNON 02173
DAVID ELLIOT SHAW 94301
JOHN M. SHAW 20014
J. B. SLATER NW3 7ST UNITED KINGDOM
GORDON STUART V8P 5J2 CANADA
MENACHEM SZUS ISRAEL
RAMON TAN 18104
DIDIER THIBAULT 75005 FRANCE
RICK THOMAS 20012
RON THOMAS 55425
SEVED TORSTENDAHL S-145 72 SWEDEN
EDWIN TSE H9R 1G1 CANADA
W. TYLER 95060
INDULIS VALTERS 55455
STEVEN W. WEINGART 55113
ROBERT E. WELLS 02138
ROY A. WILSKER 02154
DENIS M. WILSON AB9 2UB UNITED KINGDOM
GREGORY J. WINTERHALTER 48103
JOHN M. WOBUS 13210
ANDREW HARRIS ZIMMERMAN 94086
TOM ZWITTER 44139

S. KAMAL ABDALI
DEPT. OF MATHEMATICAL SCIENCES
RENSSELAER POLYTECHNIC INSTITUTE
TROY NY 12181
(518) 270-6558

JOHN M. WOBUS
453 WESTCOTT ST. APT. 1
SYRACUSE NY 13210
(315) 472-4923

RAMON TAN
2345 UNION ST.
ALLENTOWN PA 18104
(215) 434-5432

WILLIAM C. HOPKINS
DEPT. OF COMP. AND INFO. SCI.
U OF PENNSYLVANIA
PHILADELPHIA PA 19174
(215) 243-8549

RICK THOMAS
408 DOMER AVENUE
TAKOMA PARK MD 20012
(301) 565-2678

JOHN M. SHAW
BLDG 36 / ROOM 2A29
NATIONAL INSTITUTES OF HEALTH
BETHESDA MD 20014
(301) 496-3204

WILLIAM A. RUTISER
CAAC
THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON DC 20052
(202) 676-6525

FRED W. POWELL
INNOVATIVE MANAGEMENT SYSTEMS
865 MIDDLEBROOK AVENUE
STAUNTON VA 24401

PEI HSIA
COMPUTER SCIENCE PROGRAM
U OF ALABAMA AT HUNTSVILLE
P.O. BOX 1247
HUNTSVILLE AL 35807

TOM ZWITTER
ADVANCED DEVELOPMENT DIV.
BUILDING B
OHIO NUCLEAR INC.
6000 COCHRAN RD.
SOLON OH 44139

GREGORY J. WINTERHALTER
5148 DEXTER RD.
ANN ARBOR MI 48103

CHARLES G. MOORE
NETWORK SERVICES INC.
175 JACKSON PLAZA
ANN ARBOR MI 48106
(313) 426-2620

HERMAN BERG
108 E. DAYTON
MADISON WI 53703
(608) 251-1910

JOHN E. COLLINS
BLDG 235 F247
3M CENTER
ST. PAUL MN 55101
(612) 736-0778

MARK RUSTAD
585 HARRIET AVE #213
ST. PAUL MN 55112

KEVIN HAUSMANN
MECC
2520 W. BROADWAY
LAUDERDALE MN 55113
(612) 376-1119

STEVEN W. WEINGART
MS 4753
SPERRY-UNIVAC
2276 HIGHCREST DRIVE
ROSEVILLE MN 55113
(612) 633-6170 X3748

ROBERT A. LAWLER
MS U2M23
UNIVAC PARK
P.O. BOX 3525
ST. PAUL MN 55165
(612) 456-3107

ROSS D. SCHMIDT
MN 11-2120
HONEYWELL INC.
600 2ND ST. NO.E.
HOPKINS MN 55343
(612) 542-6741

RICHARD HENDRICKSON
CRAY RESEARCH INC.
7850 METRO PARKWAY SUITE 213
MINNEAPOLIS MN 55420
(612) 854-7472

RON THOMAS
DATA 100 CORPORATION
7725 WASHINGTON AVE. S.
MINNEAPOLIS MN 55425
(612) 941-6500

SCOTT BERTILSON
UNIVERSITY COMPUTER CENTER
227 EXP. ENGR.
U OF MINNESOTA
MINNEAPOLIS MN 55455
(612) 376-5262 (WORK)
(612) 729-0059 (HOME)

TIMOTHY J. HOFFMANN
UNIVERSITY COMPUTER CENTER
227 EXP. ENGR.
U OF MINNESOTA
MINNEAPOLIS MN 55455
(612) 373-6957 (HOME)
(612) 376-5262 (WORK)

INDULIS VALTERS
MISRC
93 BLEGEN HALL
U OF MINNESOTA
WEST BANK
MINNEAPOLIS MN 55455
(612) 341-4430 (HOME)

R. I. JOHNSON
COMP. SCI. DEPT.
U OF NORTH DAKOTA
BOX 3181 UNIVERSITY STATION
GRAND FORKS ND 58202
(701) 777-4107

JOHN L. NORSTAD
VOGELBACK COMPUTING CENTER
NORTHWESTERN UNIVERSITY
2129 SHERIDAN RD.
EVANSTON IL 60201
(312) 492-5369

TRUMAN C. PEWITT
APPLIED MATH DIVISION
BLDG. 221
ARGONNE NATIONAL LABORATORY
9700 SOUTH CASS AVENUE
ARGONNE IL 60439
(312) 739-7711

MIKE LEMON
168 WEST THIRD STREET
EL PASO IL 61738
(309) 527-4342

ED KATZ
COMPUTER SCIENCE DEPT.
U OF SOUTHWESTERN LOUISIANA
BOX 4-4330 USL STATION
LAFAYETTE LA 70504
(318) 233-6840
(318) 233-6767

DENNIS DANCE
COMPUTER SCIENCE DEPT.
UNIVERSITY OF ARKANSAS AT LITTLE ROCK
33RD AND UNIVERSITY
LITTLE ROCK AR 72204
(501) 569-3252

GEORGE LIGLER
1000 W. SPRING VALLEY RD. APT. 263
RICHARDSON TX 75080
(214) 231-0825

TED FISHMAN
TEXAS INSTRUMENTS
P.O. BOX 6015 (MS 295)
DALLAS TX 75222
(214) 349-3028

EDWARD E FERGUSON
1222 AUSTIN AVE
DENTON TX 76201
(214) 238-4092

DAVID N. GRAY
MS 2188
TEXAS INSTRUMENTS
P.O. BOX 2909
AUSTIN TX 78769
(512) 258-5121

GEORGE H. RICHMOND
COMPUTING CENTER
UNIVERSITY OF COLORADO
3645 MARINE STREET
BOULDER CO 80309
(303) 492-8131

HENRY R. BAUER III
COMPUTER SCIENCE DEPT.
UNIVERSITY OF WYOMING
BOX 3682
LARAMIE WY 82071
(307) 766-5134

GARY LINDSTROM
COMPUTER SCIENCE DEPT.
U OF UTAH
SALT LAKE CIT UT 84112
(801) 581-8224

DON H. ROWLAND
5805 TORREON DR.
ALBUQUERQUE NM 87109
(505) 821-9207 (HOME)
(505) 264-9149 (OFFICE)

BRUCE LINK
DIVISION 1712
SANDIA LABORATORIES
ALBUQUERQUE NM 87115

KAY A. HANSBOROUGH
2377B 45TH ST.
LOS ALAMOS NM 87544
(505) 662-9369 (HOME)
(505) 667-6034 (OFFICE)

ROBERT M. LANSFORD
BURROUGHS CORPORATION
3620 GREENHILL ROAD
PASADENA CA 91107
(213) 351-0206

DAVID C. FITZGERALD
652 S. CULLEN
GLENDORA CA 91740
(213) 335-6055

CRAIG MAUDLIN
3161 -F- VIA ALICANTE DR.
LA JOLLA CA 92037
(714) 452-8716

JIM FONTANA
CONTROL DATA CORPORATION
3519 W. WARNER AVE.
SANTA ANA CA 92704
(714) 754-4102

FLEMING M. OLIVER
213 WEDDELL APT. 12
SUNNYVALE CA 94086

ANDREW HARRIS ZIMMERMAN
550 NORTH FAIR OAKS AVE. APT. 14
SUNNYVALE CA 94086

RICHARD CORE
PO BOX 61628
SUNNYVALE CA 94088
(408) 735-8400 X233

DAVID ELLIOT SHAW
STRUCTURED SYSTEMS CORP.
427 EMBARCADERO ROAD
PALO ALTO CA 94301
(415) 321-8111

LINDA E. CROLEY
BNR INC.
3174 PORTER DR.
PALO ALTO CA 94304
(415) 494-3942 X40 OR 61

S. T. HEIDELBERG
DIVISION 8323
SANDIA LABORATORIES
LIVERMORE CA 94550
(415) 455-2179

DEAN SCHULZ
INTEL CORPORATION
3065 BOWERS AVENUE
SANTA CLARA CA 95051
(408) 246-7501

W. TYLER
200 SEABORG PLACE
SANTA CRUZ CA 95060
(408) 925-0206

ORLANDO S. MADRIGAL
DEPARTMENT OF COMPUTER SCIENCE
CALIFORNIA STATE UNIVERSITY AT CHICO
CHICO CA 95926
(916) 895-6442

PAUL O-BRIEN
P.O. BOX 10572
PORTLAND OR 97210
(503) 244-7538

GARY S. ANDERSON
JOHN FLUKE MFG. CO. INC.
P.O. BOX 43210 M.S. 16
MOUNTLAKE TER WA 98043
(206) 774-2211 X353

ATTN: COMPUTER CENTER USER SERVICES
UNIVERSITY OF WASHINGTON
3737 BROOKLYN AVE. N.E. RM 15
SEATTLE WA 98105

DAVID DEMOREST
M/S 8M-71
BOEING COMPUTER SERVICES
P.O. BOX 24346
SEATTLE WA 98124
(206) 244-6923
(206) 773-2019

J. A. CAMPBELL
MATHEMATICS DEPT.
NEWCASTLE UNIVERSITY
N.S.W. 2308
AUSTRALIA

JOHN A. LAMBERT
COMPUTING CENTRE
UNIVERSITY OF NEWCASTLE
N.S.W. 2308
AUSTRALIA

ATTN: CENTRAL LIBRARY
FLOOR 1 CASEY WING
ROYAL MELBOURNE INSTITUTE OF TECHNOLOG
376-392 SWANSTON STREET
MELBOURNE VICTORIA 3001
AUSTRALIA

I. N. BLAVINS
KATHLEEN LUMLEY COLLEGE
FINNIS STREET
NORTH ADELAID S.A. 5006
AUSTRALIA

KONRAD MAYER
REICHSAPFELG 13/8
VIENNA A-1150
AUSTRIA

EDWIN TSE
525 DELMAR ST.
POINTE CLAIRE QUEBEC H9R 1G1
CANADA
(514) 697-1320

KAY HARRISON
COMPUTER CENTER
1088B M AND C
U OF WATERLOO
WATERLOO ONTARIO N2L 3G1
CANADA

GORDON STUART
TECHNICAL AND VOCATIONAL INST.
CAMOSUN COLLEGE
1950 LANSDOWNE RD.
VICTORIA B.C. V8P 5J2
CANADA
(604) 592-1281 X248

PIERRE DESJARDINS
INFORMATIQUE
UNIVERSITE DE MONTREAL
C.P. 6128
MONTREAL QUEBEC 101
CANADA
(514) 343-6463

JEAN BEZIVIN
DEPARTEMENT DE MATHEMATIQUES & INFORMA
UNIVERSITE DE RENNES
RENNES CEDEX 35031
FRANCE
36.48.15

ATTN: A.D.R.
CHAMBRE DE COMMERCE ET D-INDUSTRIE
6 BLD GAMBETTA
GRENOBLE 38000
FRANCE

DIDIER THIBAULT
17 RUE GAY-LUSSAC
PARIS 75005
FRANCE
527 16 85

JACQUES FARRE
T 55.65
INSTITUT DE PROGRAMMATION
4 PLACE JUSSIEU
PARIS CEDEX 05 75230
FRANCE
336 25 25 X58 77

ATE PHUNG
KREFELDER STR. 23
AACHEN D-5100
GERMANY

HORST SANTO
GMD-IPES
POSTFACH 1240
ST.AUGUSTIN 1 D-5205
GERMANY

H.-J. HOFFMANN
FACHBEREICH INFORMATIK
TECHNISCHE HOCHSCHULE
STEUBENPLATZ 12
DARMSTADT D-6100
GERMANY

MANFRED SEIFERT
INFORMATIK III
UNIVERSITAT KARLSRUHE
ZIRKEL 2
KARLSRUHE D-7500
GERMANY
0721/608-3982

BERNHARD NEBEL
STEGLITZER STR. 17F
HAMBURG 70 2
GERMANY
040/664911

ATTN: THE LIBRARY
MINISTRY OF DEFENCE
P.O.BOX 962
HAIFA
ISRAEL

MENACHEM SZUS
ART AND SCIENCE
BEZALEL ACADEMY OF ART AND DESIGN
10 SHMUEL HANAGID ST.
JERUSALEM
ISRAEL

TERUO HIKITA
DEPT. OF INFO. SCI.
U OF TOKYO
TOKYO 113
JAPAN
03-812-2111 X2947

LEON LUKASZEWICZ
COMPUTATION CENTRE
POLISH ACADEMY OF SCIENCE
WARSZAWA PKIN 00901
POLAND
200211 X2225

C. C. HANDLEY
DEPT. OF COMPUTER SCIENCE
UNIVERSITY OF DURBAN-WESTVILLE
P/BAG X54001
DURBAN 4000
SOUTH AFRICA
821211 X138

SEVED TORSTENDAHL
TOMTBERGAV 279
NORSBORG S-145 72
SWEDEN

CLAES RICKEBY
HEDEBYVAGEN 5
BROMMA S-161 54
SWEDEN
08/37 65 37

BERGT NORDSTROM
DEPARTMENT OF COMPUTER SCIENCES
CHALMERS INSTITUTE OF TECHNOLOGY
GOTEBERG S-402 20
SWEDEN

LARS G. MOSSBERG
VOLVO FLYGMOTOR AB
BOX 136
TROLLHATTEN S-461 01
SWEDEN

LARS MAGNUSSON
INSTITUTE OF TECHNOLOGY
UPPSALA UNIVERSITY
BOX 534
UPPSALA S-751 21
SWEDEN
018-10 04 70

HILMAR GUTFELDT
RESEARCH AND DEVELOPMENT-DEPT. 82
HASLER LTD.
BELPSTRASSE 23
BERNE 14 CH-3000
SWITZERLAND
031 65 21 11

C. BRON
DEPT. OF ELECTRICAL ENGINEERING
TECHNISCHE HOGESCHOOL TWENTE
POSTBUS 217
ENSCHEDE
THE NETHERLANDS
(031) 53 894451

DENIS M. WILSON
DEPARTMENT OF COMPUTING SCIENCE
UNIVERSITY OF ABERDEEN
KING-S COLLEGE
OLD ABERDEEN SCOTLAND AB9 2UB
UNITED KINGDOM

ATTN: COMPUTER UNIT
COMPUTER CENTER
UNIVERSITY OF WARWICK
COVENTRY ENGLAND CV4 7AL
UNITED KINGDOM
(0203) 24011 X2754

N ROBINSON
1 THE FAIRWAY
NORTHWOOD MIDDLESEX
LONDON ENGLAND HA6 3DZ
UNITED KINGDOM

B. J. CORNELIUS
DEPT. OF COMP. STUDIES
UNIVERSITY OF HULL
HULL ENGLAND HU6 7RX
UNITED KINGDOM

B. T. MITCHELL
COMPUTING LABORATORY
UNIVERSITY OF ST. ANDREWS
NORTH HAUGH ST. ANDREWS
FIFE SCOTLAND KY16
UNITED KINGDOM

J. B. SLATER
COMPUTER UNIT
WESTFIELD COLLEGE
KIDDERPORE AVENUE
LONDON ENGLAND NW3 7ST
UNITED KINGDOM

J. J. FLORENTIN
DEPARTMENT OF COMPUTER SCIENCE
BIRKBECK COLLEGE
MALET STREET
LONDON ENGLAND WC1E 7HX
UNITED KINGDOM

BRIAN MEEK
COMPUTER UNIT
QUEEN ELIZABETH COLLEGE
CAMPDEN HILL ROAD
LONDON ENGLAND W8 7AH
UNITED KINGDOM

ARTICLES

(FORMAL SUBMITTED CONTRIBUTIONS)

LIFE, LIBERTY AND THE PURSUIT OF UNFORMATTED INPUT

D.W. BARRON AND J.M. MULLINS

University of Southampton

IN PUGN#5, Eisenberg presents three examples which, he claims, demonstrate the necessity for formatted input. This note attempts to demolish those claims.

Example 1 is concerned with survey analysis. 75 candidates are rated on a scale 1 to 5, and the observations from each individual taking part in the survey are punched as a contiguous stream of 75 digits. Formatted input allows this to be read as 75 integers: extracting the integer values by reading characters and using ord(ch) - ord('0') is said to "detract from the exercise ... the student is not interested in the use of "ord" or strings...."

The student needs to realise that dealing with this sort of data is messy, so that in future he can tell people designing surveys to prepare their data in a more palatable manner, instead of producing unreadable and inherently error prone sequences of digits. (Anyone designing a survey ought to consult the person responsible for analysing the results before the data is punched. They can then be told that redundancy in the form of separators allows error checking.)

Example 2 "Your Ph.D. Advisor" has produced a card deck in which "columns 11 to 70 contain 20 3-digit numbers...".

See the remarks on example 1 above. If your Ph.D. Advisor really produces data like this, he ought not to be allowed to advise Computer Science Ph.D. students. We are asked to note that omitting blanks "saves keypunching time and storage space, both of which are the equivalent of money". Balderdash. Do the sums, and compare the cost with the time of the (expensive) programmer sorting out the resultant mess. (Incidentally, anyone who tells us to keypunch real numbers in a contiguous stream to avoid "wasting time typing spaces" will be told, politely but firmly, to go get his head examined.)

Example 3 is presented as a situation where formatted input "is a near necessity". The problem is to extract the three fields from the 7-digit code on a pair of jeans, where the first three digits code the style, the next two code the color, and the last two digits code the material. Who needs formatted input? How about:

```
var code, style, color, material : integer;
read(code);
material := code mod 100;
color    := (code div 100) mod 100;
style    := code div 10000;
```

This method works as long as we can read the largest code, 7070436, which means we need at least 24-bit arithmetic. Doing it this way in a class has the advantage that the student will understand mod and div at the end of it, as well as learning something about positional coding systems, the representation of integers, and the significance of maxint.

If we want to avoid the maxint restriction, we have to use the dreaded 'ord'. But is it so terrible? If we have defined type digit = 0..9 and we are prepared to take the accuracy of the input on trust (we shouldn't, but formatted input schemes do), all we need is

```
function nextdigit : digit;
var ch : char;
begin
read (ch);
nextdigit := ord(ch) - ord('0')
end;
```

If the student is dealing with these kind of codes it will do him no harm to use ord, since in any case he ought to appreciate the difference between the character '3' and the number 3.

ARTICLES

(FORMAL SUBMITTED CONTRIBUTIONS)

Finally, if we want to be fancy we can define

```
type field = (onecolumn, twocolumns, threecolumns);
and     function formattedread (f : field): integer;
        begin
        case f of
          onecolumn : formattedread := nextdigit;
          twocolumns : formattedread := 10* nextdigit + nextdigit;
          threecolumns : formattedread := 100* nextdigit +
                                          10* nextdigit + nextdigit
          end;
        end;
```

OK, this only does what formatted input would have done. But we can amend nextdigit to catch input errors, and educationally it is far better to learn to do it this way than it is to learn how to construct format statements.

There is only one moral to be drawn from the defence of formatted input:

WARNING: FORTRAN CAN IMPAIR YOUR JUDGEMENT

(*Received 11/29/76*)

PRNTPLT - PASCAL PRINTER PLOTTER

```
(*  a set of subprograms for producing two-dimensional  *)
(*          (X,Y) plots on ordinary printers            *)
                                  (* by Herb Rubenstein, Research Assistant
User                                    University Computer Center
Callable
Procedures:   SCLINITIAL          University of Minnesota *)
              SCLWINDOW
              SCLPLOT
              SCLPRINT

Language:     PASCAL 6000.

Computer
System:       KRONOS 2.1 CDC CYBER 74.

Location:     University of Minnesota.

Programmer:   Herbert Rubenstein.

Documentor:   same.

Date:         September, 1976.      (revised - December, 1976)

Contents:     A.)  General Description.
              B.)  Using the PRINTER PLOTTER.
              C.)  Programming Examples.
              D.)  User Hints.
              E.)  Notes.
```

A. General Description:

This PRINTER PLOTTER consists of four PASCAL callable procedures: SCLINITIAL, SCLWINDOW, SCLPLOT, and SCLPRINT. A high speed line-printer, 1004 remote job entry terminal, or a 132 column hard-copy interactive terminal (such as a DECwriter) is used for output. The plots are X-Y graphs scaled to fit a single sheet of printer-paper. X values run horizontally while Y values are vertical. Axis labels and axes are automatically set up. The user need only provide data points, titles, and plotting characters. Nearly any range of data is suitable. Plots overlayed on one another (multiplots) can be generated just as easily as single plots. Using SCLWINDOW, (expansions) can be made. That is, a tiny piece of a plot can literally be blown-up and made into a new plot. The PASCAL PRINTER PLOTTER was created primarily because the printer-plotting routines, available in the FORTRAN library do not offer the features and simplicity described herein. Besides, they are not compatible with PASCAL anyway.

B. Using the PRINTER PLOTTER:

The PASCAL PRINTER PLOTTER is easy to use. Only a few declarations are necessary. Four separate procedures must be considered: SCLINITIAL, SCLWINDOW, SCLPLOT, and SCLPRINT.

Single plots are made by calling the basic procedures once, first SCLINITIAL, then SCLPLOT, and finally SCLPRINT. Multiplots are made by repetitively calling SCLPLOT. The first call establishes the scale, and all further calls will then be handled according to that scale. Any points lying outside the boundaries established by either SCLPLOT (first call), or a call to SCLWINDOW will be ignored.

SCLWINDOW acts as if it were the first call to SCLPLOT except there is no plotting. SCLWINDOW provides the expansion facility by allowing the user to specify ranges of X and Y values directly.

B.1. Variable and Type Declarations:

Certain declarations are required by the PASCAL PRINTER PLOTTER. The user should insert them into his MAIN PROGRAM. The most important is an ARRAY of 700 words which SCLPLOT uses as a large work area. Since the four procedures also communicate through this ARRAY, it is very important that it not be altered by the user. Identifiers used in the following illustration are only suggested. In other words, the large work area, which will be referred to as IMAGE, could just as well be called XYZ.

Example B.1

```
PROGRAM TESTPLOTTER(INPUT,OUTPUT);
CONST;
   NUM=50;  (*MAX NUMBER OF DATA POINTS TO BE PLOTTED*)
TYPE
   SCRATCH = ARRAY [1..700] OF INTEGER;
   ARDATA  = ARRAY [1..NUM]  OF REAL;
   CH60    = PACKED ARRAY [1..60] OF CHAR;
VAR
        X : ARDATA;  (*ARRAY OF X-VALUES*)
        Y : ARDATA;  (*ARRAY OF CORRESPONDING Y-VALUES*)
        N : INTEGER; (*NUMBER OF DATA POINTS TO BE PLOTTED*)
    IMAGE : SCRATCH; (*WORK AREA FOR PASCAL PRINTER PLOTTER*)
    TITLE : CH60;    (*TITLE OF THE PLOT  *)
     XLEG : CH60;    (*HORIZONTAL HEADING FOR THE X-AXIS*)
     YLEG : CH60;    (* VERTICAL  HEADING FOR THE Y-AXIS*)
       CH : CHAR;    (*PLOTTING CHARACTER*)
```


Note:

It is important that SCRATCH, ARDATA, and CH60, or any other names used for these, be defined exactly as shown above. SCLINITIAL, SCLWINDOW, SCLPLOT, and SCLPRINT expect this and cannot function otherwise. Also, the variables: TITLE, XLEG, YLEG, CH, and N are optional. This depends upon the particular form of PROCEDURE usage. See section B.3 and example B.3.3.

B.2. Procedure Declarations:

The user should also declare the four procedures: SCLINITIAL, SCLWINDOW, SCLPLOT, and SCLPRINT in the MAIN PROGRAM. Notice that IMAGE, PRFILE, X, and Y are the only variable parameters. X1, X2, Y1, Y2, N, CH, TITLE, XLEG, and YLEG are value parameters. Value parameters can be specified as constants, directly in a PROCEDURE call. This can help eliminate extra variables and also can be an aid to readability.

PROCEDURE declarations must appear as shown below. Again, identifiers can be anything convenient, however positioning is extremely important. For example, in SCLPLOT, the parameters: IMAGE , N , X , Y , CH must appear in that order. X , Y , N , CH , IMAGE is syntactically incorrect.

Example B.2

```
PROCEDURE SCLINITIAL(VAR IMAGE:SCRATCH);EXTERN;

PROCEDURE SCLWINDOW(VAR IMAGE:SCRATCH;
                        X1,X2,Y1,Y2:REAL);EXTERN;

PROCEDURE SCLPLOT(VAR IMAGE:SCRATCH;
                      N:INTEGER;
                        VAR X,Y:ARDATA;
                              CH:CHAR);EXTERN;

PROCEDURE SCLPRINT(VAR IMAGE:SCRATCH;VAR PRFILE:TEXT;
                        TITLE,XLEG,YLEG:CH60);EXTERN;
```

Note:

PROCEDURE declarations must be made under the default compiler X option, that is, X4 for PASCAL 6000.

B.3. Typical PROCEDURE Usage:

The following are four examples of typical PROCEDURE usage. The first three are single plots. The fourth is a multiplot. The second is an expansion of the first using SCLWINDOW. The third shows how convenient it can be to substitute constants directly as ACTUAL PARAMETERS, thus eliminating extra variables. Notice that TITLE, XLEG, and YLEG are actually strings. Strings are defined in PASCAL as PACKED ARRAYS OF CHAR. Also, note that SCLPRINT writes the graph on the file, OUTPUT. A different external file, other than OUTPUT, can be used if desired.

Example B.3.1

```
SCLINITIAL(IMAGE);                          (* INITIALIZE *)
SCLPLOT(IMAGE,N,X,Y,CH);                    (*    PLOT    *)
SCLPRINT(IMAGE,OUTPUT,TITLE,XLEG,YLEG); (*   PRINT    *)
```

Example B.3.2

```
SCLINITIAL(IMAGE);                          (* INITIALIZE *)
SCLWINDOW(IMAGE,X1,X2,Y1,Y2);               (*   EXPAND   *)
SCLPLOT(IMAGE,N,X,Y,CH);                    (*    PLOT    *)
SCLPRINT(IMAGE,OUTPUT,TITLE,XLEG,YLEG); (*   PRINT    *)
```

Example B.3.3

```
SCLINITIAL(IMAGE);
SCLPLOT(IMAGE,47,X,Y,"P"); (*CONSTANTS ARE SUBSTITUTED FOR N AND CH*)
SCLPRINT(IMAGE,OUTPUT,
    ".....................THIS IS THE TITLE.....................",
    ".....................THIS IS THE X-LEGEND.....................",
    ".....................THIS IS THE Y-LEGEND.....................");
(*CONSTANTS ARE SUBSTITUTED FOR TITLE, XLEG, AND YLEG*)
```

Example B.3.4

```
SCLINITIAL(IMAGE);
SCLPLOT(IMAGE,N,X,Y,"A");  (*       ESTABLISH   SCALING        *)
SCLPLOT(IMAGE,N,X,Y,"B");  (*ALL FURTHER  REFERENCES TO SCLPLOT
        .                    ARE DEPENDENT  UPON THE SCALE JUST
        .                    ESTABLISHED.                      *)
        .
        .
        .
        .
SCLPLOT(IMAGE,N,X,Y,"Y");
SCLPLOT(IMAGE,N,X,Y,"Z");
SCLPRINT(IMAGE,OUTPUT,TITLE,XLEG,YLEG):  (*PRINTS THE MULTIPLOT*)
```


C. Programming Examples:

Three different programming examples will be given at the end of this writeup. The first is an example of a single plot. It shows what happens if the data is too large or small to have properly displayed axis labels. A scale factor message is printed beside each axis affected. The second is a multiplot. Finally, the third is an expansion of the second showing where B, C, and D intersect. The range of X-values is from 4.5 to 5.5 while the range of Y-values is from 0.4 to 0.6.

Notice that when using SCLWINDOW for expansion purposes, the more data points provided, the more resolution possible. That is, if the user generates more X and Y values, (by making delta X smaller), more expansions are possible. Expansions of expansions are easily obtained in this manner. If the user is running out of data points after repetitively expanding a plot, then decreasing delta X and generating more points will help.

In examples C.2 and C.3 be sure to notice how Y is declared as an ARRAY of ARRAYs of Y-values. This makes overplotting convenient in a looping situation. Do not forget that it is either the first call to SCLPLOT, or a single call to SCLWINDOW which determines the overall scale. Any calls thereafter are treated according to that scale.

Do not call SCLWINDOW more than once. Nothing will happen except time will be wasted. For the same reason, do not call SCLWINDOW if SCLPLOT has previously been called. Also, if either the X-values or the Y-values for SCLWINDOW are accidentally given as being equal, no scaling will occur. The first SCLPLOT call would then provide the scaling. The parameters following IMAGE can be in any numerical order as long as the first two are X-values and the second two are Y-values.

D. Hints, Cautions, Errors:

There is one mistake a user could make which can cause unpredictable results, that is, failure to call SCLINITIAL.

The usual syntax precautions apply here as well as in any PASCAL program. Make sure that the parameters are of the right TYPE and see that they are properly positioned. Do not forget that the variable parameters are: IMAGE, PRFILE, X, and Y, and that the value parameters are: X1, X2, Y1, Y2, N, CH, TITLE, XLEG, and YLEG. Also, if the user is manipulating compiler options, the X option must be set to X4 for PROCEDURE declarations.

If for each call to SCLPLOT, N is less than or equal to zero, and either SCLWINDOW is called with the X-values or the Y-values being equal, or is not called at all, then when SCLPRINT is called a message will be printed: "EMPTY GRAPH".

Normally, when calling SCLPRINT, the OUTPUT file is used. See - Typical Procedure Usage. However, a different external file could just as well be used provided that it is properly declared in the program heading. Also, to obtain extra copies of a graph, call SCLPRINT repetitively.

In reference to SCLPLOT, as long as N agrees with the number of X and Y values and also if N is not less than or equal to zero, everything will work properly. It should go without saying that any N passed to SCLPLOT should lie within the range of INTEGER numbers as defined by PASCAL 6000.

E. Notes:

The external routines described herein require approximately 2300 octal words of central memory.

The scaling algorithm was obtained from: Dixon, W, J. and R. A. Kronmal, "The Choice of Origin and Scale for Graphs," JOURNAL OF THE ACM, 12,2 (april, 1965) pages 259-261.

This reference gives an excellent method for choosing graph scales which centers the information in the graphical frame and provides divisions which are simple numbers. A few changes have been made to handle special cases (e.g., when all values are on a single X or Y line) and to re-scale to fit PASCAL field specifications. The plot image of numerically rounded points is built up in the scratch array and is then printed.

J5 MINN SCLPLOT, by M. Frisch, revised February 1971, describing a FORTRAN printer plotting routine, was also used as a reference.

****

A Copy of This Writeup Can be Obtained in Two Ways:

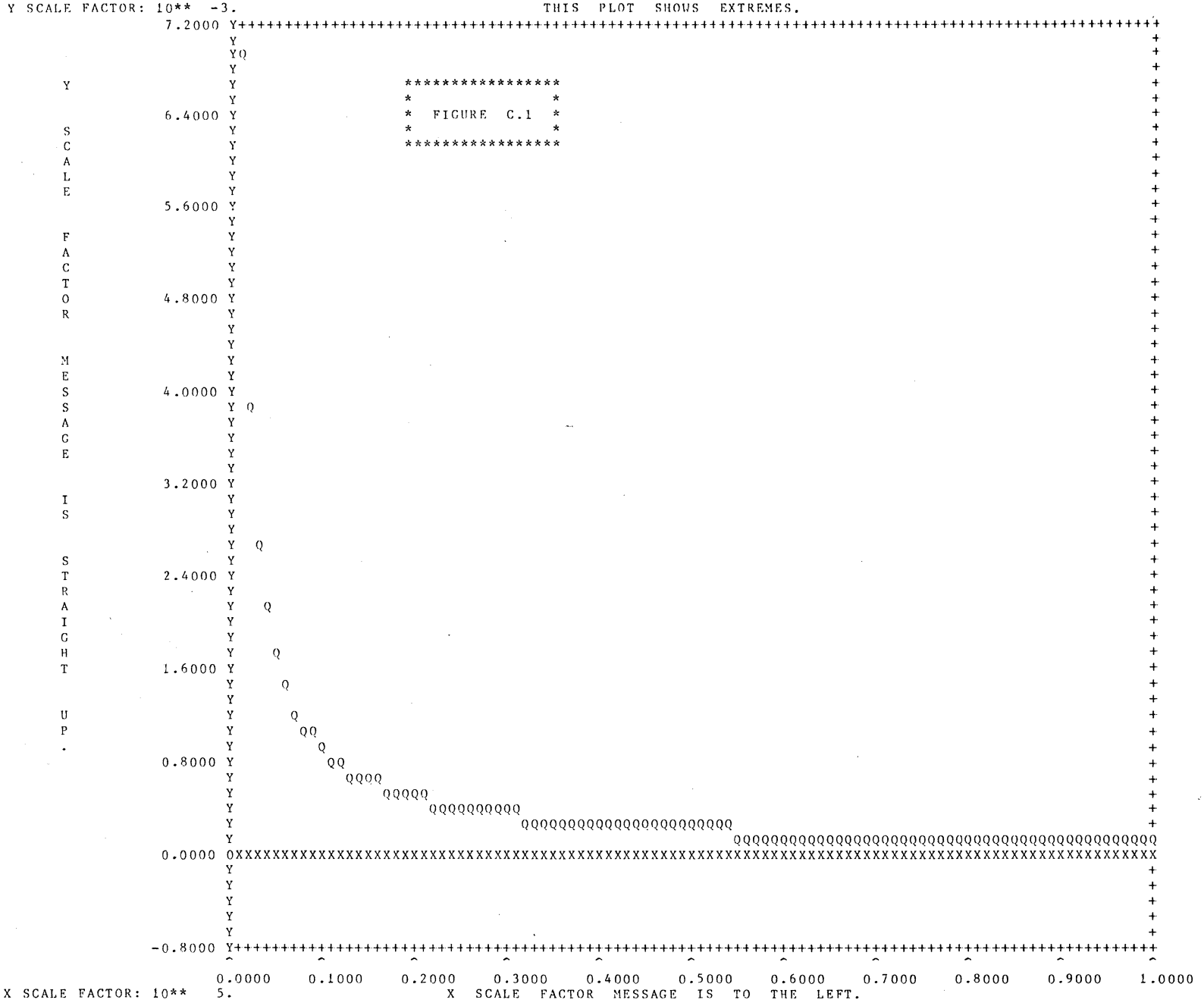
Check room 140 Experimental Engineering.
Look for: -PASCAL PRINTER PLOTTER-

Use the CYBER 74 or 6400 machine.
Execute the control statement: WRITEUP(PASCLIB=PRNTPLT)

****

(*Received 12/19/76*)

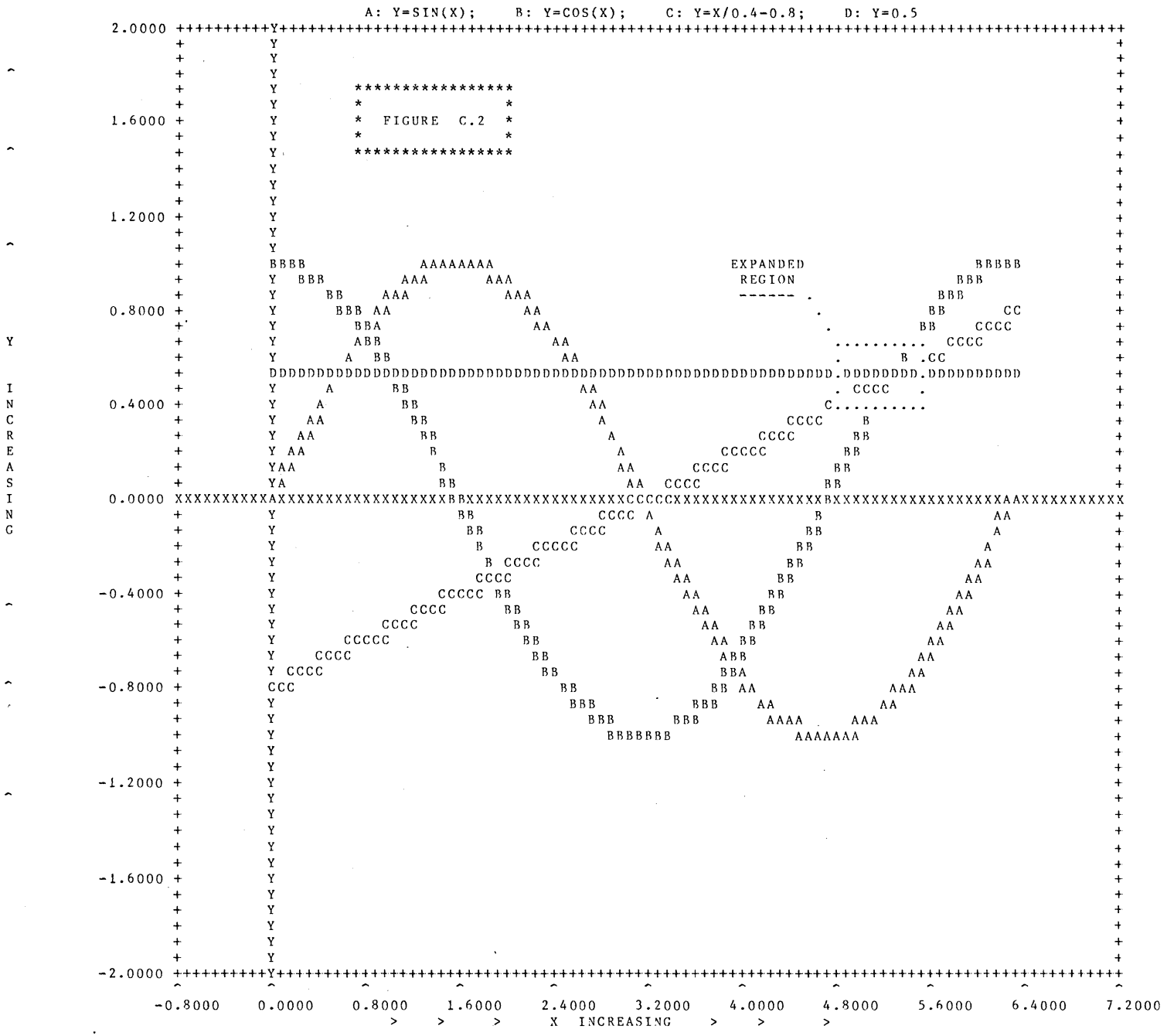

```
PASCAL COMPILER - E.T.H. ZURICH, SWITZERLAND - PASCAL CYBER V2.0 - 76/08/26.
UNIVERSITY OF MINNESOTA       (76/07/21)

000006 (*                          EXAMPLE C.1                              *)
000006 (*                          -----------                              *)
000006
000006
000006 PROGRAM SINGLEPLOT(OUTPUT);
000235 CONST MAX = 200; (* MAX NO. OF POINTS *)
000235 TYPE
000235    SCRATCH= ARRAY[1..700]OF INTEGER;
000235    ARDATA=ARRAY[1..MAX] OF REAL;
000235    CH60=PACKED ARRAY[1..60]OF CHAR;
000235 VAR
000235    X:ARDATA;
000401    Y:ARDATA;
000545    I:INTEGER;
000546    N:REAL;
000547    IMAGE:SCRATCH;
002043    TITLE,XLEG,YLEG:CH60;
002065
002065 (***************************)
002065
002065 PROCEDURE SCLINITIAL(VAR REC:SCRATCH); EXTERN;
000004
000004 (*****************************************)
000004
000004 PROCEDURE SCLWINDOW(VAR IMAGE:SCRATCH;X1,X2,Y1,Y2:REAL);EXTERN;
000010
000010 (*************************************************************)
000010
000010 PROCEDURE SCLPLOT(VAR REC:SCRATCH;N:INTEGER;VAR X,Y:ARDATA;CH:CHAR);
000010                EXTERN;
000010
000010 (*************************************************************)
000010
000010 PROCEDURE SCLPRINT(VAR REC:SCRATCH;VAR F:TEXT;TITLE,XLEG,YLEG:CH60);
000010                         EXTERN;
000031
000031 (*************************************************************)
000031
000031 BEGIN(*MAIN PROGRAM*)
002065 TITLE:="              THIS  PLOT  SHOWS  EXTREMES.                  ";
000021  XLEG:="    X  SCALE  FACTOR  MESSAGE  IS  TO  THE  LEFT.           ";
000024  YLEG:="    Y  SCALE  FACTOR  MESSAGE  IS  STRAIGHT  UP.            ";
000027 N:=1000.0;
000030 FOR I:=1 TO 100 DO
000031    BEGIN (*GENERATE 100 DATA POINTS*)
000033    X[I]:=N;
000037    Y[I]:=LN(X[I])/X[I];
000055    N:=N+1000.0
000055    END;   (*GENERATE 100 DATA POINTS*)
000060 SCLINITIAL(IMAGE);
000062 SCLPLOT(IMAGE,100,X,Y,"Q");
000071 SCLPRINT(IMAGE,OUTPUT,TITLE,XLEG,YLEG)
000073 END(*MAIN PROGRAM*).
```

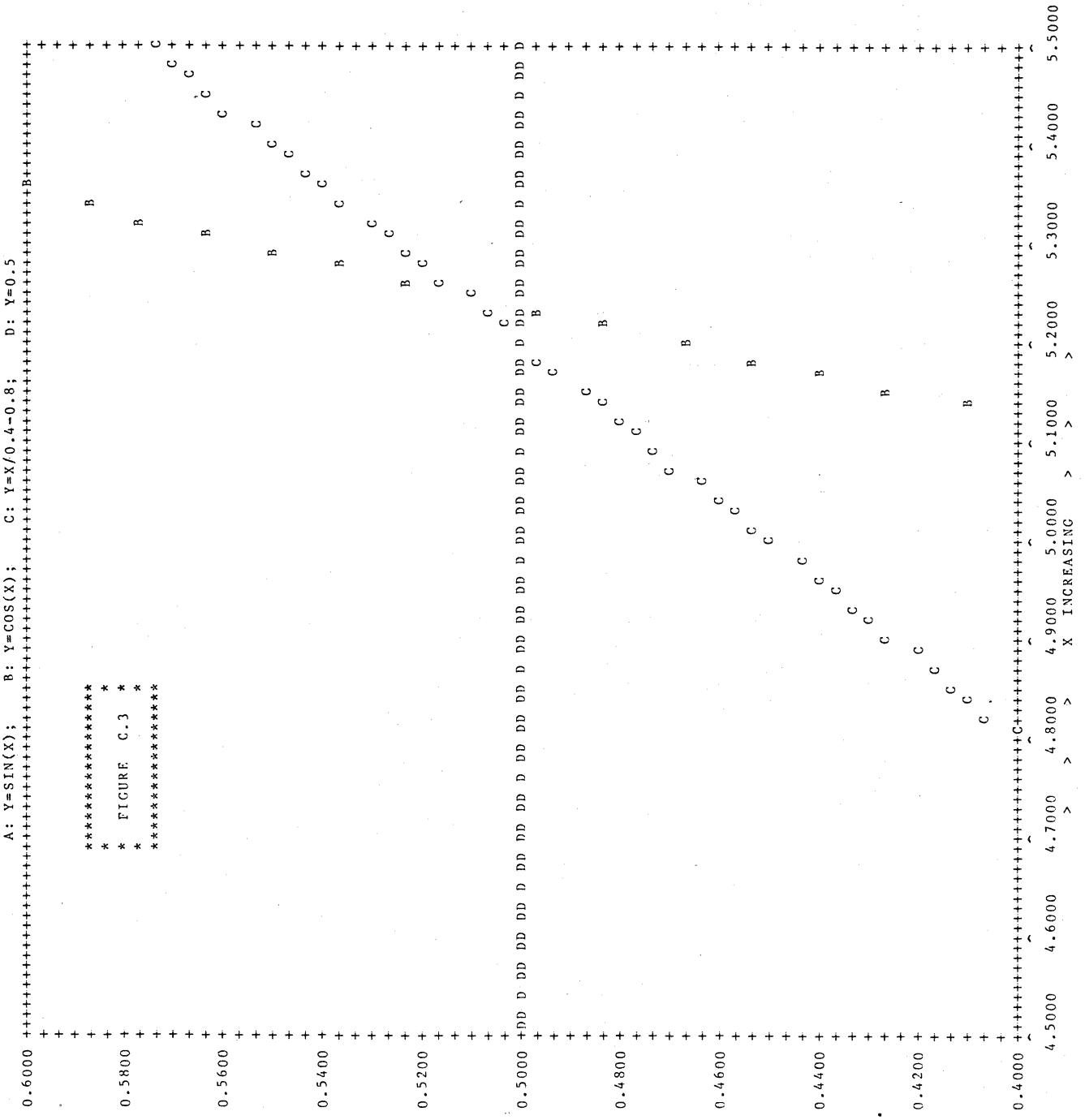

Y SCALE FACTOR: 10** -3.
THIS PLOT SHOWS EXTREMES.
7.2000
6.4000
5.6000
4.8000
4.0000
3.2000
2.4000
1.6000
0.8000
0.0000
-0.8000
SCALE FACTOR MESSAGE IS STRAIGHT UP.
FIGURE C.1
0.0000
0.1000
0.2000
0.3000
0.4000
0.5000
0.6000
0.7000
0.8000
0.9000
1.0000
X SCALE FACTOR: 10** 5.
X SCALE FACTOR MESSAGE IS TO THE LEFT.

```
PASCAL COMPILER - E.T.H. ZURICH, SWITZERLAND - PASCAL CYBER V2.0 - 76/08/26.
UNIVERSITY OF MINNESOTA       (76/07/21)

000006 (*                          EXAMPLE C.2                              *)
000006 (*                          -----------                              *)
000006
000006
000006
000006
000006 PROGRAM MULTIPLOTTING(OUTPUT);
000235 CONST MAX=400; (* MAX NO. OF POINTS *)
000235 TYPE
000235    SCRATCH= ARRAY[1..700]OF INTEGER;
000235    ARDATA=ARRAY[1..MAX] OF REAL;
000235    CH60=PACKED ARRAY[1..60]OF CHAR;
000235 VAR
000235    X:ARDATA;
001055    Y:ARRAY[1..4] OF ARDATA;
004155    I:INTEGER;
004156    IMAGE:SCRATCH;
005452
005452 (***************************)
005452
005452 PROCEDURE SCLINITIAL(VAR REC:SCRATCH); EXTERN;
000004
000004 (*******************************************)
000004
000004 PROCEDURE SCLWINDOW(VAR IMAGE:SCRATCH;X1,X2,Y1,Y2:REAL);EXTERN;
000010
000010 (*************************************************************)
000010
000010 PROCEDURE SCLPLOT(VAR REC:SCRATCH;N:INTEGER;VAR X,Y:ARDATA;CH:CHAR);
000010                EXTERN;
000010
000010 (*********************************************************)
000010
000010 PROCEDURE SCLPRINT(VAR REC:SCRATCH;VAR F:TEXT;TITLE,XLEG,YLEG:CH60);
000010                         EXTERN;
000031
000031 (*********************************************************)
000031
000031 BEGIN(*MAIN PROGRAM*)
005452 FOR I:=1 TO MAX DO
000016    BEGIN (*GENERATE DATA POINTS*)
000020    X[I]:=0.005*I*3.14159258367;
000027    Y[1,I]:=SIN(X[I]);
000042    Y[2,I]:=COS(X[I]);
000056    Y[3,I]:=(X[I]/4.0)-0.8;
000066    Y[4,I]:=0.5
000070    END;    (*GENERATE DATA POINTS*)
000073 SCLINITIAL(IMAGE);
000075 FOR I:=1 TO 4 DO SCLPLOT(IMAGE,MAX,X,Y[I],CHR(I));
000114 SCLPRINT(IMAGE,OUTPUT,
000115       "A: Y=SIN(X);    B: Y=COS(X);    C: Y=X/0.4-0.8;    D: Y=0.5 ",
000115       "   >    >     >     X  INCREASING    >    >      >           ",
000116       "   ^    ^     ^     Y  INCREASING    ^    ^      ^           ")
000116 END(*MAIN PROGRAM*).
```



```
PASCAL COMPILER - E.T.H. ZURICH, SWITZERLAND - PASCAL CYBER V2.0 - 76/08/26.
UNIVERSITY OF MINNESOTA       (76/07/21)

000006 (*                          EXAMPLE C.3                              *)
000006 (*                          -----------                              *)
000006
000006
000006
000006
000006 PROGRAM USINGSCLWINDOW(OUTPUT);
000235 CONST MAX=400; (* MAX NO. OF POINTS *)
000235 TYPE
000235    SCRATCH= ARRAY[1..700]OF INTEGER;
000235    ARDATA=ARRAY[1..MAX] OF REAL;
000235    CH60=PACKED ARRAY[1..60]OF CHAR;
000235 VAR
000235    X:ARDATA;
001055    Y:ARRAY[1..4] OF ARDATA;
004155    I:INTEGER;
004156    IMAGE:SCRATCH;
005452
005452 (***************************)
005452
005452 PROCEDURE SCLINITIAL(VAR REC:SCRATCH); EXTERN;
000004
000004 (*******************************************)
000004
000004 PROCEDURE SCLWINDOW(VAR IMAGE:SCRATCH;X1,X2,Y1,Y2:REAL);EXTERN;
000010
000010 (*************************************************************)
000010
000010 PROCEDURE SCLPLOT(VAR REC:SCRATCH;N:INTEGER;VAR X,Y:ARDATA;CH:CHAR);
000010                EXTERN;
000010
000010 (*********************************************************)
000010
000010 PROCEDURE SCLPRINT(VAR REC:SCRATCH;VAR F:TEXT;TITLE,XLEG,YLEG:CH60);
0Q0010                         EXTERN;
000031
000031 (*********************************************************)
000031
000031 BEGIN(*MAIN PROGRAM*)
005452 FOR I:=1 TO MAX DO
000016    BEGIN (*GENERATE DATA POINTS*)
000020    X[I]:=0.005*I*3.14159258367;
000027    Y[1,I]:=SIN(X[I]);
000042    Y[2,I]:=COS(X[I]);
000056    Y[3,I]:=(X[I]/4.0)-0.8;
000066    Y[4,I]:=0.5
000070    END;    (*GENERATE DATA POINTS*)
000073 SCLINITIAL(IMAGE);
000075 SCLWINDOW(IMAGE,5.5,4.5,0.4,0.6);
000103 FOR I:=1 TO 4 DO SCLPLOT(IMAGE,MAX,X,Y[I],CHR(I));
000122 SCLPRINT(IMAGE,OUTPUT,
000123       "A: Y=SIN(X);    B: Y=COS(X);    C: Y=X/0.4-0.8;    D: Y=0.5 ",
000123       "   >    >     >     X  INCREASING    >    >      >           ",
000124       "   ^    ^     ^     Y  INCREASING    ^    ^      ^           ")
000124 END(*MAIN PROGRAM*).
```


A: Y=SIN(X); B: Y=COS(X); C: Y=X/0.4-0.8; D: Y=0.5
FIGURE C.2
EXPANDED REGION
Y INCREASING
X INCREASING
2.0000
1.6000
1.2000
0.8000
0.4000
0.0000
-0.4000
-0.8000
-1.2000
-1.6000
-2.0000
-0.8000 0.0000 0.8000 1.6000 2.4000 3.2000 4.0000 4.8000 5.6000 6.4000 7.2000


A: Y=SIN(X); B: Y=COS(X); C: Y=X/0.4-0.8; D: Y=0.5
FIGURE C.3
Y INCREASING
X INCREASING
0.6000
0.5800
0.5600
0.5400
0.5200
0.5000
0.4800
0.4600
0.4400
0.4200
0.4000
4.5000
4.6000
4.7000
4.8000
4.9000
5.0000
5.1000
5.2000
5.3000
5.4000
5.5000

YET ANOTHER LOOK AT CODE GENERATION FOR PASCAL ON CDC 6000 AND CYBER MACHINES

BY

LAWRENCE A. LIDDIARD
ASSOCIATE DIRECTOR/SYSTEMS AND OPERATIONS
UNIVERSITY COMPUTER CENTER

DECEMBER, 1976

In the 1974-1975 school year the Computer Science Department at the University of Minnesota decided that more extensive use of PASCAL would be made in their programming language courses. Since much of this use would be interactive, one concern of the University Computer Center was that PASCAL required the most memory for compilation (approx. 55000_8) of the interactive languages available (for comparison note that BASIC and APL*CYBER require approximately 25000_8 and MNF FORTRAN 44000_8) on our Instructional Time Sharing CDC 6400. Several years experience in running a large volume (202 maximum simultaneous users and 350,000 runs in April 1975 see reference [5]) time sharing service had shown that a successful time sharing service on CDC 6000 machines required that each user be limited to at most 54000_8 words of memory. (In addition to a restriction on memory for each user there are requirements for enough mass storage channels and peripheral processors . . . but that is another story.) Since the PASCAL level 9 compiler required 42121_8 to load and in addition allocated buffers, "stack" and "heap" space, I decided that a reduction in PASCAL load size could be accomplished by rethinking certain PASCAL code generations in the area of procedure calls, constant loads, case statement jumps and/or by combining common procedures (+, , divide, mod, in-line functions). In addition when Urs Ammann of ETH, Zurich was informed of the project, he suggested several core reduction ideas that were implemented by John Strait of our staff in late 1975. In a compiler-compiler the generation of faster and shorter code helps not only the user, but also since PASCAL 6000 compiles itself this will be reflected in a smaller and faster compiler.

In trying to analyze where core reductions can be made in a compiler such as PASCAL, it is worthwhile obtaining several tables that help the

core reducer to concentrate on the essentials of the language (or program) that are amenable to reduction techniques. The first such table is the count of the static number of procedure calls ordered in descending use.

In PASCAL 6000 with a load length of approximately 18,000-60 bit words (43130_8) there are approximately 2300 static (meaning physically present) procedure calls distributed among the approximately 140 procedures that constitute the compiler. The top 10% of the called procedures are listed in the following table with an additional break down into three main areas: code generation, symbol input and error message.

TABLE 1

PROCEDURE NAME	STATIC COUNT
ERROR	387
GEN15	338
INSYMBOL	214
GEN30	154
COMPTYPES	136
DECREFX	124
NEEDX	116
LOAD	77
SKIP	53
NOOP	45
CLEARREGS	40
EXPRESSION	37
NEXTCH	36
OPERATION	25
TOTAL	1782 = 77% of 2300 calls

CODE GENERATION				SYMBOL INPUT		ERROR MESSAGE	
NAME	COUNT	WORDS /CALL	CORE REQUIRED	NAME	COUNT	NAME	COUNT
GEN15	338	* 4 =	1352	INSYMBOL	214	ERROR	387
GEN30	154	* 4 =	616	SKIP	53		
COMPTYPES	136	* 4.5 =	612	NEXTCH	36		
DECREFX	124	* 3 =	372				
NEEDX	116	* 3.5 =	406				
LOAD	77	* 3 =	231				
NOOP	45	* 1 =	45				
CLEARREGS	40	* 1 =	40				
OPERATION	25	* 3.5 =	87	* 1 word/call =		*1½ word/call=	
TOTALS	1055		3761		303		580
% of 2300 calls: 46%				13%		17%	
% of load core:			21%		2%		3%

Except for EXPRESSION these procedures fall into the three groups mentioned above and account for 76% of the static procedure calls and 25.5 per cent of the total loaded length of PASCAL 6000. Thus, a large savings can be obtained for each word saved in a static procedure call (2300 * n words saved).

Another place to look for central memory reductions is in the prologue and exit of each procedure. Since there are 140 procedures in the PASCAL 6000, each word saved represents 140 * n (214_8) additional cells saved. Candidates for this reduction are the procedure name word, the word containing the lengths of the executable and total procedure code, the three words for procedure initialization, and the 1 1/2 words for procedure exit. The first two are needed for the current POST MORTEM dump, but could be eliminated if the POST MORTEM dump would obtain these words from the LGO file or would build a specific POST MORTEM file at load time for possible use when errors occurred. The procedure initialization and exit can be reduced to one word and three quarters respectively by use of common entry and exit routines at a probable 8% slow down in compilation speed due to the slowness of NEXTCH (actually 25% of compilation time on a CYBER 74 is currently spent in NEXTCH, and by improving this routine the 8% compilation slow down can be more than compensated for by the NEXTCH speed up).

One final way to look for central memory reduction in the compiler is to look at a table of the largest PASCAL 6000 procedures. This uses the theory that it always pays to look at the fattest routines. When looking at this table the core reducer should look for similar functionality that can be combined into one routine or broken out into a simple subroutine. In addition the longest routines are examined for exactly what makes them long, with a look to improving code production that can reduce the length (in PASCAL it may be the nesting depth of procedure calls from that routine, a poorly done CASE statement, or a rethinking of code generation). Finally, the reverse of top down step wise refinement (integration) can sometimes achieve good savings as was obtained by combining the procedures ROUNDF, ABSF, SQRF, TRUNFC, ODDP, ORDF, CHRF, PREDSUCCF, CARDF, EXPOF, into the routine STDINLINEFUNCS.

TABLE 2

PASCAL 6000 PROCEDURES ORDERED BY LENGTH IN PASCAL2

COMPILER PROCEDURE NAME	LENGTH		
	PASCAL 2	PASCAL 2 MODIFIED	
CALLNONSTANDARD	1266_8	1034_8	
STORE	1244_8	1153_8	
LOAD	1200_8	1146_8	
TERM	1027_8	516_8	benefits from depth of nesting change by not loading static link
EXPRESSION	1025_8	634_8	
FACTOR	1006_8	533_8	
BODY	754_8	674_8	
TYP	712_8	605_8	
INSYMBOL	655_8	566_8	
WRITE	603_8	436_8	
UNPACK	535_8	411_8	
FORSTATEMENT	523_8	415_8	
FIELDLIST	515_8	414_8	
INDEXCODE	504_8	414_8	
CASESTATEMENT	503_8	426_8	
PACK	473_8	352_8	
PARAMETERLIST	472_8	375_8	
SIMPLEEXPRESSION	461_8	220_8	("+" and "-" code combined)
PROCEDUREDECLARATION	427_8	404_8	
STDINLINEFUNCS [as separate routines]	[1350_8]	352_8	

Not all of the code reduction features were viable or even desirable; but since the application of each reduction was applied to the previous one, the following table briefly describes the change, the octal length at the PASCAL compiler after the change and the octal (decimal) savings

CHANGE		PASCAL OCTAL LOAD LENGTH	SAVINGS
0	stock level 9 compiler	43121	
1	avoid extra SBi Xj commands when loading a base address	42703	216_8(142)
2	avoid extra Bxi Xj commands at procedure calls	42666	15_8(13)
3	use 2 jump commands/word in CASE statements	42504	162_8(114)
4	correct inefficient case statement in the procedure "STATEMENT"	42462	22_8(18)
5	use common ENTRY/EXIT routines	41577	663_8(435)
6	make the "RJTOEXT" procedure reasonable	41435	142_8(98)
7	use a single procedure for the in line functions (ODD..CARD) called STDINLINEFUNCS	40600	635_8(413)
8	eliminate extra stack manipulation	40441	137_8(95)
9	use MXi mask and LXi shift for constants	40410	31_8(25)
10	eliminate jumps after procedures that end a CASE or THEN	40017	371_8(249)
11	use common code for "+" and "-"	37707	110_8(72)
12	use X5 = MX5 number-1 of static indirects to load if static link not available	35541	2146_8(1126)
13	use RJ procedure rather than SX7 return, JP procedure	34162	1357_8(751)
	TOTAL		6737_8(3551)

Examples of the code generated in the application of these principles are given in the following pages for changes 3, 5, 9, 10, 12 and 13. Note also that the application of change 13 disallows 10.

Change 3

Example of the CASE statement

From procedure OPTIONS

```
    PASCAL                          CODE GENERATION
16  CASE  CHI  OF       .16   SAI   CHI                SB3   x1
20  'B'             .17   JP    B3+.113
27  'E'         'B'.20    (* code for case 'B' *)
    'L'         'E'.27    (* code for case 'E', etc. *)
    'P'             .113
    'T'             .114  JP    .144       (*'A'*)
    'U'             .115  JP    .20        (*'B'*)
    'X'             .116  JP    .144       (*'C'*)
```



```
    PASCAL                          CODE GENERATION (continued)
    END             .117  JP    .144       (*'D'*)
144 IF              .120  JP    .27        (*'E'*)
                    .
                    .
                    .
                    .142  JP    .144       (*'W'*)
                    .143  JP    .106       (*'X'*)
                    .144
```

If 2 jumps per word are used

```
13  CASE            .13   SAI   CHI  SX1 X1-2
15  'B'             .14   LX1   59   SB3 X1     JP B3+.112
24  'E'         'B'.15    (*etc.*)
    .               .111  JP    .126       (*'A'*)
    .               .112  PL X1,.15         (*'B'*)  JP .126 (*'C'*)
    .               .113  PL X1,.126        (*'D'*)  JP .24  (*'E'*)
    END             .114  PL X1,.126        (*'F'*)  JP .126 (*'G'*)
126 IF              .
                    .
                    .
                    .124  PL X1,.126        (*'V'*)  JP .126 (*'W'*)
                    .125  PL X1,.103        (*'X'*)  NO       NO
```

Saves: 11 cells

Costs: loss of rememberance of CHI in X1 and slower execution speed.

single jump/word 48 cycles for SA1 CH1; SB3 X1; NO; JP B3 + .113; JP.20

vs 61 cycles upper SA1 CH1; SX1 X1-2; LX1 59; SB3 X1; JP B3 + .112; PL X1,.15

vs 66 cycles lower jump taken.

(Note the SX1 X1-2 is not needed if only positive case labels are allowed).

Change 5

The current entry code of

```
                SA6 B6              (save static link) in stack
                SX6 B5              save old STACK BASE
                LX6 18
                BX7 X7+X6           + return address in STACK + 1
                SB5 B6              set new STACK BASE
3 1/2 words     SA7 B5+B1
                SA6 B6+needed       set B6 to NEXT
                SB7 B6+100
                SA0 5               set return address
                GE  B7,B4,HEAPCOLLISION
```

```
  is changed to 1 word
              SB7 needed
              RJ  P.ENTRB           (see code for 12 and 13)

the current exit code of
              SA1 B5+B1
              SB6 B5                reset NEXT
  1 3/4 words SB7 X1                set return address
              LX1 42
              SB5 X1                restore old STACK BASE
              JP  B7
  is changed to 3/4 word
              SA1 B5+B1
              JP  P.EXIT
  where P.EXIT SB7 X1
              LX1 42
              SB6 B5
              SB5 X1
              JP  B7
```

For a CYBER 74 the change in P.EXIT ratio in cycle speed is 33/26 = 1.27; for the 6400 the ratio is 63/51 = 1.24.

Change 9

Examples of the MXi and LXi commands for constants.

In PASCAL the very useful powerset often uses constants that have contiguous bit sequences. For the CDC 6000 machines it is usually best to have code productions that are two 15-bit commands rather than a single 30-bit command, since the 30-bit command often will not fit into the current word causing a non-useful NO command to be produced. The MASK and SHIFT commands of the CDC 6000 allow any constant of the form $(2^n-1)*2^m$ to be generated by MXi n; LXi m+n. If the constant value is greater than 2^{17} this is always the best method since it takes (on a 6400) the same or less time and saves the 60-bit word holding that constant. If the constant is less than 2^{17} and there is only 15-bits left in the current word, this method will avoid the useless NO command.

E X A M P L E S

a) PROCEDURE WITHSTATEMENT (COMP 6147, 6148)

```
IF CBDFSPL <> 0 THEN                CURRENT PASCAL
   BEGIN NEEDX(0,7,1)               ZR  X7,.143  BX0 X6  NO
                                    SX1 7  SX2 B5+20
                                    SX6 B7  SX7 .120   NO
                                    EQ  NEEDX
```



```
                                    MODIFIED PASCAL
                                    ZR  X7,.127  BX0 X6  MX1 3
                                    LX1  3  SX2 B5+20  MX5 1
                                    RJ  NEEDX
```

b) PROCEDURE OPTIONS (COMP 407)

```
'B' IF CH IN ['1'..'9']             CURRENT  SA1 B2+474  SA2 162
                                    MODIFIED SA1 B2+474  MX2 9  LX2 37
```

c) PROCEDURE GEN30 (COMP 2070)

```
CBUF = CBUF*1000B*1000B+777777B+FK  CURRENT  BX5 X6  LX5 9  LX5 9  NO
                                             SA1 217
CBUF = CBUF*1000000B+777777B+FK     MODIFIED BX5 X6  LX5 18  MX1 18  LX1 18
```

Change 10

Example from Procedure LOAD (COMP 2792-2796)

```
     PASCAL                              CODE GENERATION
     IF SVAL = 0 then GEN15(13B,1,1,1)   .120 IX4 X2-X3  NZ X4,.125  NO
     ELSE                                .121 SX0 13B  BX2 X1  BX3 X1
125  IF SVAL = 1 then GEN15(76B,1,1,0)   .122 SX6 B7   SX7 .124
     ELSE                                .123 JP GEN15
131  IF SVAL = 2 then GEN15(76B,1,1,1)   .124 JP .141
                                         .125 IX4  X2-X0  NZ X4,.131  NO
                                         .126 SX0  76B  SX2 B1  SX6 B7
                                         .127 SX7 .130  JP GEN15
                                         .130 EQ .141
                                         .131 SX4 B1+B1  IX5 X2-X4  NZ X5,.136
                                         .132 SX0 76B  SX2 B1  BX3 X2
                                         .133 SX6 B7  SX7 .135
                                         .134 JP  GEN15
                                         .135 JP  .141
                                         .136
```

Rule 1

If the termination of a THEN clause is a procedure call and there is no other clause ending at the same point before an ELSE; more efficient code can be generated by eliminating the standard JP to terminal IF point by making the return address of that procedure call go to the terminal IF point. The above example code would produce the following, saving three words and the corresponding jump times:

```
.120 IX4 X2-X3  NZ X4,.124  NO
.121 SX0 13B  BX2 X1  BX3 X1
.122 SX6 B7  SX7 .141
.123 JP GEN15
.124 IX4 X2-X0  NZ X4,.127  NO
.125 SX0 76B  SX2 B1  SX6 B7
.126 SX7 .141  JP GEN15
.127 SX4 B1+B1  IX5 X2-X4  NZ X5,.133
.130 SX0 76B  SX2 B1  BX3 X2
.131 SX6 B7  SX7 .141
.132 JP GEN15
.133
```

Rule 2

If the termination of an individual CASE is a procedure call and there is no other clause ending at the same point (i.e. more than one address that refers to such a point) then the JP to the terminal CASE point can be eliminated by making the return address and that procedure call go the terminal case point. The coding produced is similar to that in the THEN-ELSE.

Changes 12 and 13

Example from the procedure FACTOR.

Note that this modification depended on Change 5, the use of a common ENTRY point routine.

Previous CODE GENERATED FOR DECREFX(I) and FACTOR(FSYS):

```
          DECREFX(I)   COMP 5200            FACTOR(FSYS)   COMP 5115
          SA1 B5+12  ]
          BX0 X1     ] I                                   SA1 B5+3
                                                           BX0 X1
3 ½ words SA2 B5     ] follow                2 ½ words      SA2 B5
          SA2 X2     | static                              BX6 X5
          SA2 X2     | link                                SX7 *+2
          SA2 X2     |                                     EQ  FACTOR
          BX6 X2     ]
          SX7 *+2    ] set return address
          EQ  DECREFX] transfer to routine
```

Using changes 12 and 13:

```
          SA1 B5+12  ]
          BX0 X1     ] I                     1 ½ words      SA1 B5+3
                                                           BX0 X1
1 ½ words MX5 4      ] if base not in B register          SA5 B5    ]base in B
          RJ  DECREFX] transfer to routine                 RJ  FACTOR]register
```


P.ENTRB

is an example of the common ENTRY routine where the static link is stored on the stack and a RJ ROUTINE is used rather than the SX7 RETURN JP ROUTINE.

X5 = STATIC LINK (IF X5 > 0) else MX5 LEVEL-PFLEV-1

where LEVEL and PFLEV are the LEVELS of the caller and called procedure respectively.

```
P.ENTRB  DATA  0
         PL  X5, P.ENT2
         BX6 X5
         SA5 B5
  +      SA5 X5          ]
         LX6 1           ] static link to X5
         NG  X6,*        ]

P.ENT2   BX6 X5          ] store static
         SA6 B6          ] link on stack

         SA5 P.ENTRB     ]
         AX5 30          ] obtain
         SA5 X5-2        ] RETURN
         AX5 30          ] ADDRESS
         SX7 X5          ]

         BX6 B5          ] store B5 and
         LX6 18          ] RETURN ADDRESS
         BX7 X7+X6       ] on stack
         SB5 B6          ] set B5
         SA7 B6+B1       ] to BASE
         SB6 B6+B7       ] set B6 to NEXT
         SB7 B6+100      ] exit if no STACK
         LT  B7,B4,P.ENTRB] HEAPCOLLISION
```

This change was the most controversial since although it gave dramatic reduction in central memory, it changed the manner of procedure calls from that documented in reference [3] and definitely slowed down compilation by 8% on the CYBER 74. One way to use the core reduction and slower procedure linkage would be to have the PASCAL compiler internally use the RETURN JUMP method of procedure calls while generating the forms SXi return and JP procedure for user execution binary. This would mean that three assemblies of PASCAL would be required to update to a new PASCAL rather than the two assemblies required presently.

Comparisons

Some comparisons with PASCAL 1 described in [1] are interesting.

(1) by instruction length		PASCAL 2 Modified to Save Core		
1971 PASCAL 1				
15,925	48.7% long instructions (30-bit)	12,959	32.7%	
9,385	28.7% short instructions (15-bit)	19,676	49.6%	
7,456	22.8% padding instruction (NOOP)	7,010	17.7%	executable 2564 (6.5%)
32,766	100% = $12,173_{10}$ words	39,645	100%	= $13,151_{10}$ words
(2) by instruction type				
	27.6% fetch and store	8,536	21.5%	
	15.0% load literal	4,085	10.3%	
	3.5% arithmetic	1,810	4.6%	
	14.4% logical and shift and mask	12,523	31.6%	
	6.2% base address register	776	2.0%	
	10.5% jump and subroutine calls	4,905	12.4%	
	22.8% (NOOPS)	7,010	17.7%	
32,766	100.0%	39,645	100.1%	

Reference [3] shows that register remembrances, BXi X6,7 rather than NOOP's and more efficient code generation were one aim in the design of PASCAL2. This manifests in PASCAL2 in increased subroutine calls (from 10.5% to 12.4%) to produce the more efficient code and in the reduction of fetch and store commands (from 27.6% to 21.5%). In addition to [3]'s replacement of NOOP's with logical commands the generation of MX5 level; RJ PROCEDURE compared with PASCAL's SX7 RETURN EQ PROCEDURE caused increased mask commands and a corresponding decrease in LOAD LITERALS (SXi value).

Note that in PASCAL1 15.3% of the loaded space (NOOP commands) is not used compared with 13.3% in the modified PASCAL. The designer of the CDC 6000 machines has recognized this loss in his latest machine the CRAY 1. In that machine JUMPS are to any 16-bit portion of the 64-bit word rather than to the top most portion of the word as is done in a CDC 6000 instruction. Thus the 4,446 non executable NOOPS (of 7010 total) could be eliminated if the CDC 6000 machines had such a feature allowing 8.5% of the loaded compile space to be saved.

Our last way to shorten the compiler is to rewrite the code generation part which currently comprises about one-fourth of the total length. The current code generation scheme seems to have two main deficiencies. The target computer code that PASCAL generates is spread throughout rather than gathered in functional groupings in the compiler. Second, for

"simple ability" machine instructions of micro and CDC 6000 computers, a procedure call for each command of the numerous code generation routines insures that the compiler will be fairly lengthy. In order to avoid this excess length a macro skeleton (i.e. simple) language is designed that a very short macro interpreter can expand to required computer commands, register allocations, decisions and code generation operations.

As an example consider the PASCAL code

```
357(8)  BEGIN NEEDX(0,7,K); GEN15(10B,K,I,0)          COMP 5324
        DECREFX(I); GEN15(21B,K,0,LREC.EXP)           COMP 5325
        NEEDX(0,7,I); GEN15(13B,I,I,I);               COMP 5326
        GEN15(36B,K,K,I); DECREFX(I)                  COMP 5327
414(8)  END                                           COMP 5328
        (414(8) - 357(8) = 35(8) = 29 cells)
```

rewritten in a macro skeleton similar to that used by E. J. Mundstock when we wrote the MNF compiler. In those skeletons the basic 15-bit command of the CDC 6000 was broken down into its four component parts of 6-bit operation code and 3-bit I, J and K register fields. To each of these fields an additional bit was added to signify a relative register or pseudo command if set to 1, else an absolute register or command.

MACRO Language Example for the previous PASCAL code

```
P:=LREC.EXP
MACRO(MOD1)
```

			pseudo command	command	pseudo register I	register I	pseudo register J	register J	pseudo register K	register K
			1	7	1	3	1	3	1	3 = 20 bits
where MOD1	NEEDX(0,7,K)		1	1	0	0	0	7	1	2
	GEN15(10B,K,I,0)	1st word	0	10	1	2	1	0	0	0
	DECREFX(I)		1	0	1	0	0	0	0	0
	GEN15(21B,K,0,P)		0	21	1	2	0	0	1	7
	NEEDX(0,7,I)	2nd word	1	1	0	0	0	7	1	0
	GEN15(13B,I,I,I)		0	13	1	0	1	0	1	0
	GEN15(36B,K,K,I)		0	36	1	2	1	2	1	0
	DECREFX(I)	3rd word	1	0	1	0	0	0	0	0
	ENDMACRO		1	2	0	0	0	0	0	0

pseudo commands: DECREFX=0; NEEDX=1; ENDMACRO=2; IFDEBUG=3; ENDIF=4; LOAD=5
pseudo register of ATTR: I=0; J=1; K=2; L=3; M=4; N=5; O=6; P=7

Thus approximately 29 cells are replaced by 6 cells which compares favorably with that reduction obtained in the MNF compiler where 1900 pseudo commands occupied 709 words and required a MACRO interpreter of 350 words. If the pseudo commands of MNF were done as normal procedure calls it would have taken approximately 1900 * 2 3/4 = 5225 words in the MNF compiler. Thus the total saving is 4000 words or an 80% reduction in that code generation portion of the MNF compiler. Applying this to the PASCAL compiler with approximately 4000 words for 1055 code generation procedures calls and assuming a MACRO interpreter of 500 words and 100 calls to the MACRO interpreter we can estimate a $2500\text{-}3000_{10}$ word reduction in the loaded length of the PASCAL compiler.

Note that the implementation of this assumes a VALUE declaration initialization of packed records which is currently not available in PASCAL2. The actual implementation of the MACRO skeleton is not as simple as my example but reference [4] or the MNF compiler listing give additional pseudo commands and implementation techniques.

Conclusions

PASCAL 2 is amenable to several different methods of compiler length reduction. As a fellow compiler writer (although since MNF is written in machine language it may be compared with the last of the dinosaurs speaking to Homo sapiens), I would rather see the full language specification and one standard compiler, than to see small subsets such as PASCAL-S. For this reason I think it essential to improve PASCAL2 and with the reductions discussed in this article it should be possible to obtain load lengths of approximately 30_8k for the full language rather than the current 44_8k on a CDC 6000 (i.e. a reduction by one-third).

References

1. N. Wirth, "The Design of a PASCAL Compiler," Software Practice and Experience, Oct-Dec 1971.
2. N. Wirth, "On 'PASCAL' Code Generation and the CDC 6000 Computer," Stanford, February 1972.
3. Urs Ammann, "On Code Generation in a PASCAL Compiler," ETH, April 1976.
4. David Gries, Computer Construction for Digital Computers, "17.5 Computing Code Generation," John Wiley & Sons, Inc., 1971, pages 363-366.
5. M. M. Skow, "MERITSS 1971-1975," University Computer Center, University of Minnesota, Technical Report 75003, October 1975.

(*Received 12/29/76*)

OPEN FORUM FOR MEMBERS

(SHORT, INFORMAL CORRESPONDENCE)

The University of Southampton

Computer Studies

Professor D W Barron

November 18, 1976.

Dear Andy,

Reading "A Primer on PASCAL" by Conway Gries and Zimmerman, my eye was caught by the fact that every time they print a line in their programs they start it with an explicit blank. After doing this without explanation for many pages, they eventually explain about carriage control characters, a la FORTRAN. Unfortunately, they have some justification from §14 of the REPORT: "...The first character on each line of printfiles may be interpreted as a printer control character..."

To my mind it is deplorable that a clean language should be soiled by such a hangover from the bad old days. In most of the "primitive" languages, I/O is relegated to a sub-language of a wholly adhoc nature, e.g. FORTRAN format statements and carriage control characters. In PASCAL we have got away from this: we have "writeln" and "page", and if the system has to put a '1' in front of the print line to suit the peculiarities of the line printer, that is the system's affair. I don't want to put it there myself, because my terminal will actually print the '1' - it expects ASCII code FF to signal a page feed. If everybody agrees that "page" means "move to head of form", our programs are more likely to be transportable, so why the special provision for carriage control characters in the Report?

Yours sincerely,

David

D.W. Barron.

Mr. Andy Mickel,
Pascal Users Group.

Department of Mathematics, The University, Southampton, SO9 5NH. Tel: 0703 559122 Ext: 700 Telex: 47661

UNIVERSITY OF MINNESOTA
TWIN CITIES

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

December 9, 1976

Prof. N. Wirth
Xerox Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304

Dear Niklaus,

Pascal Newsletter #6 is finally in the mail. There are two articles by G. Michael Schneider and Richard J. Cichelli of particular interest concerning needs for formal standardization of Pascal. I feel that it is very important for the User's Group to solicit your reactions and your responses to these articles because of your unique contribution to the development of the language.

I therefore warmly invite you to submit for publication to the Newsletter your opinions on the subject of these articles. I hope that this will lead to a continuing discussion on this important matter.

Sincerely,

Andy

Andy Mickel, Editor

UNIVERSITY OF MINNESOTA
TWIN CITIES

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

December 10, 1976

Christian Jacobi
Institut fuer Informatik
E.T.H. - Zentrum
CH-8092 ZURICH
Switzerland

Dear Chris,

How are you?

We are curious to know the results of the Pascal-P questionaire so that they may be printed in the Pascal Newsletter. Could you please help us all out?

Also, how many sites are on your distribution list? We have a rough idea from George Richmond's implementors list in Newsletter #4, but I'm sure that PUG members would find this information interesting. Also how many Pascal-6000 sites do you distribute to?

Be sure to keep us up to date regarding prices and the distribution policy printed in the Newsletter for both Pascal 6000 and Pascal-P.

Thanks a lot!

Sincerely,

Andy

Andy Mickel, Editor

OPEN FORUM FOR MEMBERS

(SHORT, INFORMAL CORRESPONDENCE)

1101 New Hampshire Avenue NW
Washington DC 20037
2 jan 77

Dear Andy -

Since the purpose of a language is to communicate among people, it seems clear that the standards should be set by the users; a committee for PASCAL ought to be set up. Niklaus Wirth has done a magnificent job in creating PASCAL, but those of us who are to use it should have a voice in shaping it as a communications tool.

To reconcile Wirth's legitimate concerns for the purity and economy of the language with the needs of the users for expanded utility, e.g. runtime allocation of array bounds, perhaps the following suggestion may be appropriate:

1) Let there be a STANDARD PASCAL, whose main purpose (and perhaps only purpose) is to mediate the implementation of PASCAL compilers. This language should be very close to the language set forth in the REPORT & USER MANUAL.

2) Let everyone who wishes to expand the language provide a compiler, written in STANDARD PASCAL, that will implement his expansion.

These two principles, which might appropriately be called the Vademecum Principles, would allow for standardization and variety. Students could be taught in STANDARD PASCAL [and there is no doubt that the economy of SP is greatly valuable for teaching purposes] and could use it for much of their programming work. Us peons that do the practical work could earn our freedom from the standards by providing an identifiable compiler, written in SP without compromise, that would implement whatever ideas we want to bring to the marketplace.

I detest the premonition that I have: that the developing countries will be shackled to Fortran* as they emerge into the computing world. The PUG should do everything it can to achieve unity, and not allow the virtues of PASCAL to be drowned in a sea of discordant proposals.

Happy New Year!

Arthur Brown

Arthur Brown

* Or [ugh!] COBOL or [ugh! ugh!] PL 1.

UNIVERSITY OF MINNESOTA
TWIN CITIES

Social Sciences Research Facilities Center
25 Blegen Hall
Minneapolis, Minnesota 55455

January 3, 1977

Dear Andy,

After the great "standards outcry" of #6, I want to contribute my two cents worth.

First, I fully believe that Standard Pascal, as defined by Prof. Wirth in the latest Report, is defendable in terms of the two objectives stated there: a good teaching tool, and reliable and efficient implementation. It is not surprising then that Standard Pascal should be frozen by its author. I think we must accept the fact that any politically accepted changes or extensions leave us with a different language (perhaps a "Standard Production Pascal").

Since Pascal is being used for purposes not encompassed by Prof. Wirth's objectives (e.g., applications, production, and even commercial processing), there is a need to reassess the language. It is important however to distinguish clearly between any new version of Pascal and the Standard Pascal in the Report. To do any less is inconsiderate of a creative person whose work will remain relevant to further development.

Second, as an applications-oriented user, I do see the need for extensions (and possibly changes). The resulting language must have a "political standard" (e.g., ANSI), and thus a standards committee is inevitable. A relatively centralized body is necessary for effective discussion and debate on detailed issues, as well as for providing a formal mechanism for language maintenance as Mike Schneider discussed in #6. What I fear in the formation of a committee is the power of deciding what the "standard"will be. I quote the word standard because of the difference between official policy and what we fondly refer to as the real world. We want the standard to be "good", and we want the standard to be enforced. I don't believe that the formation of a committee of benevolent despots will assure either outcome.

A "bad" standard should not be enforced, and hopefully with enough intelligent and independent-minded users and implementors a "bad" standard cannot be enforced. On the other hand, users will create their own non-standards if the official position does not account for real or perceived need for change. There is ample evidence to show that Pascal is both portable and mutable, suggesting that supply will grow to meet demand whether officially standard or not.

Regardless of how a standards committee may be established, I think that it should accept suggestions for changes and extensions from users, and if necessary should shape the suggestions into fully developed proposals for consideration and decision by the entire user community. Under no circumstances should a committee have the authority to decide what will and will not be officially standard. Its purpose should be an organized facility to handle the mechanics of standardization, and to filter or reshape fragmented or poorly thought-out suggestions (i.e., quality control).

Each suggestion considered by the committee must be weighed against many criteria, and these deliberations should be included in the resulting proposal. Each proposal should also include complete definitions, examples, implementation recommendations, proof techniques, and perceived impact on the entire language.

One of the few points on which I disagree with Richard Cichelli (#6) deals with committee representation of implementors. As I see it, one of the prime motivations for a committee's existence is to assure that officially standard features are feasible -- i.e., implementable on (nearly?) all machines. My experience suggests that users do not appreciate in all cases the implementation difficulties inherent in their suggestions. What better way is there to represent implementors than through such a committee? The committee should combine the skills of the language designer and the language feature designer (C.A.R. Hoare, "Hints on Programming Language Design") with those of the implementor.

The representation of users is of course very important. The Pascal Newsletter provides an excellent forum both for the dissemination of proposals, and for the reactions and views of users. I doubt that users can be better represented than by having the fate of each proposal be determined by a general vote.

Finally, I'm afraid that we as users are most vulnerable right now because of the lack of any formal mechanism for controlling change in the language. As serious users of Pascal we have an investment in code. I hope that all currently active implementors are considering very carefully the costs of deviating from the current standard (User Manual and Report). Incompatible implementations can only have the effect of devaluating our investments.

Optimistically, James F. Miner

GENERAL INFORMATION (1/3/77)

We must again thank Tim Bonham and all the many who have responded to his letters. Things seem to be on track. This is what we have to print. If you don't find what you are looking for, check newsletters #5 and #6. Note! New implementors: We don't know who you are, so please keep us informed; use the checklist below.

CHECKLIST* * * * * * * * * * * * * * *

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

* * * * * * * * * * * * * * *

PASCAL-P

Pascal-P is a "portable" compiler for Pascal generating code for a hypothetical "stack machine". It comes on tape as a kit and may be used to bootstrap compilers onto real computer systems. Not much has changed with Pascal-P distribution (we have yet to receive a copy of George Richmond's new order form). Please refer to PUGNs #5 and #6 especially #6 for ordering information.

IMPLEMENTATION NOTES

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

Reports we have, seem to be very favorable for Pascal P4 (see Stephen Schwarm in the Here and There section and the PDP-8 news described herein). See the Open Forum section for a request sent to Chris Jacobi for the results of the Pascal-P questionaire which appeared in #5. The following persons replied to Tim Bonham's letters that they wish to be removed from the implementors list in Newsletter #4:

Gordon Stuart at Camosun College, Victoria, B.C. Canada
Richard O'Bryant at Texas Instruments, Dallas, Texas
Bob Vavra, Sperry-Univac, Roseville, Minnesota

PASCAL TRUNK COMPILER

(We have sent a request to Urs Ammann of ETH, Zurich for more information.)

PASCALJ

UNIVERSITY OF COLORADO
DEPARTMENT OF ELECTRICAL ENGINEERING
BOULDER, COLORADO 80309

November 19, 1976

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

Dear Mr. Bonham:

I enclose a copy of our latest price list and status of the project. As you will note, we do not believe that PASCALJ is yet a viable product. Currently we are completing a set of macros which resolve all stack operations and indexing in terms of a single accumulator and simple index register. These macros will be modified to handle multiple registers, with a projected completion date of February, 1977. At that time another evaluation of the portability of the system will be made.

Sincerely yours,

William M. Waite

William M. Waite
Professor

WMW mjr
enc

IMPLEMENTATION NOTES

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

DEPARTMENT OF ELECTRICAL ENGINEERING

UNIVERSITY OF COLORADO

Boulder, Colorado 80309 USA

1 September 1976

PASCALJ

Since our February distribution of PASCALJ we have concentrated upon cleaning up the compiler itself and resolving some small anomalies in the definition of Janus. This work is reflected in the current distribution.

We have received feedback from several groups who have attempted implementation of PASCALJ via a full bootstrap. This feedback indicates that such an implementation is very time-consuming with current tools, and may well lie outside the reach of most people. Our own experience indicates that this view is substantially correct: PASCALJ is not really portable at the present time.

During the next months we plan to concentrate upon improvement of the bootstrap. Our basic approach is to provide tested tools which will map Janus code into that of a simple machine which is easily realizeable but not perfectly efficient. We hope that by February we will have brought the bootstrap within reach of most potential implementors.

One recurring theme in the letters we have received is that the syntax of Janus is too strongly biased towards STAGE2 processing. This is a valid criticism. We have derived an LL(1) grammar for Janus, in which the basic symbols can be extracted by a simple lexical analyzer. Examples of changes are:

DISP3	becomes	DISP(3)
AINT 5	becomes	A INT 5

The September compiler does not incorporate these changes; we hope to release them in February also.

- Software Engineering Group

A. IDENTIFICATION

Program name: PASCALJ

Authors: B.W. Pavenel, C.B. Mason

Date: 1 September 1976

B. GENERAL DESCRIPTION

PASCALJ is a compiler which translates the high-level language PASCAL to the intermediate language Janus. This compiler was originally written in PASCAL, and used to translate itself to Janus. The original PASCAL program is included in the Janus text as comments, and may be extracted by selecting only those lines which have a period in the first character position. All other lines should be ignored during extraction, and the periods in position 1 should be replaced by spaces.

PASCAL is defined in the following book, which is available from its publisher (Springer-Verlag):

Jensen, K., Wirth, N., PASCAL User Manual and Report
1975, Second Edition.

It is necessary to indicate the character set of the host computer on which PASCALJ is to be implemented, as the compiler must provide a mapping from this character set to its own internal representations. We are providing several standard character sets from which one should be chosen; should some other character set be required please send a detailed collating sequence for it.

Character sets available:

ASCII (full set, 96- or 64- character subset)
EBCDIC
CDC display code

C. DOCUMENTATION

SEG-76-1	A Preliminary Definition of Janus	5 oz	$4.00
SEG-76-2	PASCALJ Implementation Notes	2 oz	2.25
SEG-76-1	Janus Memory Mapping: The J1 Abstraction	2 oz	2.25

D. TEXT (Includes STAGE2)

7-track magnetic tape (1200-foot reel)	1.75 lb	28.00
9-track magnetic tape (1200-foot reel)	1.75 lb	39.00

(Deduct $7.00 from tape cost if you supply a 1200-foot reel. We will accept longer reels, but we must charge more for postage. Overseas orders add cost of postage and spcify type of shipping.)

CONCURRENT PASCAL (no new information - see Newsletter #6)

MODULA (maybe more information in the next newsletter)

SOFTWARE WRITING TOOLS

Mike Ball (distributor and maintainer of a Univac 1100 Pascal compiler) phoned recently to advocate the formation of a clearing house for Pascal Software Writing Tools similar to what we've requested in the Editor's Contributions in #6 and this issue. He confesses that his 3 compilers keep him too busy to perform this service and that it will have to be someone else. He would like to contribute some programs to the pool (and I'm sure that other people would want to do likewise). One of these programs is a "source code configurator". Mike cited the problems of character sets and a coordinated standard for external and internal documentation. (* Perhaps we also need a simple editor to facilitate updates for these programs. *)

BURROUGHS B-6700/B-7700

Last newsletter a vague reference was made to a technical report about A. H. J. Sale's implementation in Hobart, Tasmania, Australia. The full reference is: "Burroughs PASCAL - Some Implementor's Thoughts", Arthur Sale, Department of Information Science Report 76-2, University of Tasmania, 14 pages.

In its contents are sections on: Principles; Design Decisions; Lexical Tokens; Syntactic Features; Machine Dependency; Run-Time Actions; Compiler Options; Input and Output; Regrets; Closing Remarks; and References.

CII IRIS 80, 10070

In a letter dated November 11 to Tim Bonham, Didier Thibault, 8 rue d'Auteuil, 75016 Paris, France, wrote: "...joined is the description and the details about the status of our project: Pascal compiler for CII Iris 80, 10070 computers. All this information may be of interest to PUG members. You may publish it in the next Pascal newsletter. I am sending a copy of your letter to the maintenance group in Toulouse so that they may get in touch with you...."

PASCAL COMPILER FOR CII I0070, CII IRIS 80 and XDS SIGMA 7 COMPUTERS

IMPLEMENTOR Didier THIBAULT
I7 rue GAY-LUSSAC
75005 PARIS FRANCE tel: 527 I6 85

MAINTENER & DISTRIBUTER
Pierre MAURICE
UER d'informatique- université Paul SABATIER
II8 route de Narbonne
3I077 TOULOUSE FRANCE tel:(6I) 53 II 20 ext.300

MACHINES CII I0070 , CII IRIS 80 , XDS SIGMA 7

OPERATING SYSTEM SIRIS 7 & SIRIS 8 (CII operating systems)
easily available on other operating systems,see implementation

METHOD OF DISTRIBUTION with magnetic tape.Contact distributer

DOCUMENTATION AVAILABLE
User's manual in French:
SFER PASCAL_ Le Langage de programmation PASCAL
compilateur pour les ordinateurs CII I0070,IRIS 80
système d'exploitation SIRIS 7-8 sept.I975

MAINTENANCE POLICY maintenance available since july 74 until july 77

STANDARD PASCAL standard with PASCAL user manual & report (Springer verlag N° I8)
and:
-separate compilation of PASCAL programs
-symbolic post mortem dump output of variables & procedure in case of abort at execution
-'value' feature for initialisation of variables
-'packed' variables implemented

IMPLEMENTATION This is a full compiler generating object code for the linkage editor.The compiler consists of

a MONITOR: written in assembly language(size: 2kwords of 32 bits) It links the PASCAL program to the operating system and controls the execution of the PASCAL program.All operating system dependancies are located in this monitor.To get the compiler available on other operating systems,the rewriting of this monitor is neccessary.

a COMPILER program: written in PASCAL language itself,it consists of 4800 lines of PASCAL program.It is a one pass compiler with top-down syntax analysis,separate compilation of PASCAL programs,symbolic post mortem dump output,and many specific options.

The compiler is fully bootstrapped so that any user may adapt it easily to it specific need (tables sizes, specific features)

a PASCAL LIBRARY used by the linkage editor

MEMORY REQUIREMENTS to run the PASCAL system:

minimum 30 K word in the overlay version of the system
45 K word without overlay

COMPILATION SPEED (measurement on CII IRIS 80 mono-processor)

1800 PASCAL lines/mn
2400 car/sec.
(FORTRAN speed: 1200 car/sec.)

EXECUTION SPEED: dependant on the program profile

	FORTRAN	PASCAL
matrix multiplication	I	I.6
recursive program	I	0.3
character count on file	I	0.2

RELIABILITY OF THE COMPILER The actual release is release 3.This compiler has been tested over 2 years in 25 different computing centers. The reliability is good-excellent.

METHOD OF DEVELOPPEMENT:

starting date: end 1972
first release: may 1974
Bootstrapped from CDC PASCAL compiler in a computing center having both CDC 6400 & CII IRIS 80 machines.
effort: 14 man x month with a team of 2 persons.
experience of implementors:experience as Stanford graduate students on compiler design and optimisation.

bibliography:D. Thibault , P. Mancel , Transport d'un compilateur PASCAL écrit en PASCAL d'un CDC 6400 à un IRIS 80. Thèse de docteur ingenieur-Sept. 1974- université Paris VI.
(micro-fighe available at IRIA documentation,domaine de Voluceau Rocquencourt 78150 Le Chesnay FRANCE)

UNIVERSITÉ PAUL SABATIER
118, route de Narbonne, 118
31077 - TOULOUSE - CEDEX - FRANCE
U.E.R. D'INFORMATIQUE

P. MAURICE

TOULOUSE, LE 22 novembre 19 76
TÉLÉPHONE 52-12-12

T. BONHAM
PASCAL Implementations
University Computer Center
227 Experimental Engineering Building
University of Minnesota
MINEAPOLIS,MN 55455
U.S.A

Dear Pr. BONHAM,

In addition to the letter by D. THIBAULT dealing with the PASCAL/IRIS-80 compiler, I give you the following information :

I am a member of the IRIA's project SFER which has, among other works, maintained, developped and distributed the PASCAL compiler since July 74th and will go on until january 78 ; it is used by about 30 centers mainly in the fields of software teaching and development.

The compiler, including all source programs -PASCAL and assembler code- is freely available ; you must just send a tape (mini, if possible) to the above address.

Transpositions to others systems can be an easy task ; it depends on the binary code compatibility. The CII-IRIS 80 machine uses the same machine langage as the XDS-Sigma 7. A successful transposition has been made by M. TAKEICHI, TOKYO University, to an XDS-Sigma 7, running under the BPM monitor. — TAKEICHI

Bug reports are encouraged and reflected into the following releases of the system.

The compiler accepts standard PASCAL (ref. "User Manual and Report" by K. JENSEN and N. WINTH, Springer Verlag 75) with some additionnal features - separate compiling, compile time initializations, extensions to read and write procedures for use in an interactive environnement - and perhaps some minor differences - standard PASCAL is sometimes somewhat evasive and I have no access to a CDC computer.

Another IRIA 's project develops a PASCAL subsystem (under Time-sharing system SIRIS 8 TS) designed to provide the users with powerful commands for editing, management (store and retreive) and exploitation of PASCAL programs. The first release of the system may take place at the beginning of 1977.

Please, contact me for further information,

Your truly,

P. MAURICE

P.S. I sent my contribution to PUG a couple of months ago and did not receive any newsletter issue

DIGITAL EQUIPMENT (DEC) PDP-8

UNIVERSITY OF MINNESOTA
TWIN CITIES

Social Sciences Research Facilities Center
25 Blegen Hall
Minneapolis, Minnesota 55455

Prerelease Implementation Notice. January 2, 1977

1. Implementors: John T. Easton 612/373-7525
   James F. Miner 612/373-9916
   Jonathon R. Gross
   Address correspondence to:
   Pascal
   SSRFC
   25 Blegen Hall
   University of Minnesota
   Minneapolis, Minnesota 55455
2. Machine: Digital Equipment Corporation PDP8/e.
3. Developed under OS/8 version 3. Hardware required:
   - KE8-E (EAE with mode B instruction set).
   - RK8-E disk, or other direct access mass storage device with at least 131K 12-bit words (e.g., DF32 or RF08).
   - 32K of core/RAM is highly recommended. Minimum is 16K.
4, 5, 6. Distribution, documentation, and maintenance details are not yet available. Hopefully more information will be available in the next Pascal Newsletter.
7. Emphasis has been placed on close adherence to the Pascal User Manual and Report.
8. As with most languages on the PDP8, Pascal makes use of an interpreter (a modification of P-code) written in PAL8. The compiler and assembler are written in Pascal. All standard procedures are written in PAL8.

   Execution is roughly comparable to Fortran IV (F4). I/O seems to be faster than Fortran, while computation seems slower. Unfortunately Pascal compilation is much slower than F4. We hope to make some improvements in this area before the first release.

   Because of the design of the system, the implementation is not suitable for real-time applications. On the other hand, the implementation does provide 131K words of virtual memory for code and store (expandable to 262K).

   Standard Pascal types are implemented as follows:

Boolean	1 word	
Char	1 word	(ASCII 32..127)
Integer	2 words	
Real	4 words	(uses equivalent of DEC 23-bit package; fourth word required for alignment.)
Set	8 words	[0..95]

9. The system has been fairly reliable.
10. We have spent about 9 man months so far. It has been useful to cross-compile from the University of Minnesota's CDC Cyber machines.

DIGITAL EQUIPMENT (DEC) PDP-11

EN-1029

E. I. DU PONT DE NEMOURS & COMPANY
INCORPORATED

WILMINGTON, DELAWARE 19898

ENGINEERING DEPARTMENT

November 10, 1976

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

Dear Mr. Bonham:

In answer to your query of October 22, 1976, see attached "PASCAL for PDP-11 As We See It".

The hardware we have available on this site for the creation of distributions is as follows:

PDP-11/50 with FP11B 64K Core

RK03
TU56 DECTAPE
TA11 DECCASSETTE
PC11 Paper Tape

PDP-11/10 25K Core

RK05
T803

Very truly yours,

ENGINEERING R&D DIVISION

Stephen C. Schwarm

S. C. Schwarm, Development Engineer
Engineering Physics Laboratory (Beech Street)

Craig E. Bridge

C. E. Bridge, Engineer
Engineering Physics Laboratory (Beech Street)
E. I. Du Pont de Nemours & Company, Inc.
Wilmington, Delaware 19898

SCS/CEB:pjm
Atch

(* Stephen Schwarm in a recent phone call confirmed the scattered news we've been receiving that the U.S. government is interested in Pascal on PDP-11s. He also reports that C. Bron's cross compiler (from a DEC-10 to PDP-11) has excellent documentation. See the Here and There section for more news. *)

PASCAL FOR PDP-11 AS WE SEE IT

Our installation has as many problems in getting a working "standard" PASCAL compiler up on our machines as any. All of our efforts so far have been aimed at transporting the work of others to our site. We have no original work available yet.

The efforts involved at this site have been to:

I. Bring up Per Brinch Hansen's SOLO/CONCURRENT and SEQUENTIAL PASCAL compiler on his minimum machine (- nine track magtape). This compiler works reasonably if you are willing to suffer along under the SOLO executive (poor editor/single user). The variable scopes are also limited to GLOBAL and LOCAL, thus voiding some of the capabilities of nested procedures. These compilers are tied extremely closely to SOLO and are nonstandard. We have developed the following to aid others in using SOLO:

   A. BOOTSOLO: copies nine track SOLO distribution magtape to SOLO RK03/05 disk cartridge. (Same as distributed SOLO bootstrap but runs on PDP-11/04/10/15/20/34/40/45/50/60/70 with 16K core, console terminal, nine track magtape, RK03/05 disk.)

   B. RSOLO: copies files from SOLO RK03/05 disk to DOS V9.20C (possibly V4?) RK03/05 disk. Runs under DOS/BATCH-11 V9.20C (V4?) on 16K PDP-11/04/10/15/20/34/40/45/50/55/60/70 with 16K core, two RI03/05 disk drives, console terminal.

   C. READOS: copies files from DOS/BATCH-11 V9.20C (possibly DOS-11 V4?) disk to SOLO disk. Runs under SOLO and PDP-11/45/50/55/60/70 with 48K core, floating point unit, two RK03/05 disk drives, console terminal.

II. Bring up University of Illinois PASCAL compiler under MINITAUR operating system (bastard son of DOS-11 V4?). This system is nonstandard, it works well, but is ugly inside. Runs on PDP-11/04/10/15/20/34/40/45/50/55/60/70 in 28K core under MINITAUR. DOS/BATCH-11 V9.20C can read/write files onto MINITAUR file structured device (not the other way however). System comes up in one evening.

   Documentation is scant, however, and the command syntax for batch streams is entirely different from DOS/BATCH-11 which came in DOS-11 V8.0 and after.

III. Bring up DECUS PASCAL-11 from Lucien Feiereisen, et al (PASCAL to JANUS through Stage 2 macros to MACRO-11).

11/10/76

Came up in one afternoon. This implementation has a rather weak stomach. It will not digest anything but small syntactically correct (as per this implementation as opposed to "standard") programs. Most errors end up as F342 (addressing exceptions) without hope of further diagnosis. This particular implementation runs on a PDP-11/34/40/45/50/55/60/70 under DOS V9.20C with 64K core.

This implementation has given DEC a sour taste on PASCAL (much to our dismay).

IV. Bring up a PCODE-2 to MACRO-11 translator in hopes of bootstrapping the PCODE PASCAL compiler. This approach got bogged down in the problem of knowing what size the stack elements that were pushed, popped and operated on. The word and byte alignment problem was the straw that broke PCODE-2's back. We have just recently received the PCODE-3 and 4 release which promises to solve these problems. Work will continue in this direction.

V. Bringing up several of the PDP-10 PASCAL compilers (all of which work well). Don't wish to elaborate here since newsletter 5 covers this well.

S. C. Schwarm/C. E. Bridge:pjm
11/10/76

TECHNISCHE HOGESCHOOL TWENTE

ONDERAFDELING DER TOEGEPASTE WISKUNDE

Timothy Bonham,
University of Minnesota,
University Computer Center,
227 Experimental Engineering Building,
Minneapolis - Minnesota 55455.

ONDERWERP: KENMERK: TW76/INF/793p. ENSCHEDE, 14.11.1976.

Dear Mr. Bonham,

With regard to your letter of October 25th, it appears that several of your questions have been answered in PASCAL Newsletter 4, dated July 76. Questions unanswered are the following.

6. Maintenance policy.
   We intend to correct reported errors for the next few years. This is however a private enterprise, and no absolute guarantees can be given. Error reports and updates will be sent at irregular instants to those who have received a copy of the compiler, and have not stated their lack of interest.
9. Reliability : good (what measure ?)
10. Method of development : adaptation from P-compiler. See for further details publication in Software, Practice and Experience.

Yours sincerely,

C. Bron.
Associate Professor of Computer Science.

ENSCHEDE - DRIENERLO - POSTBUS 217 - TELEFOON: 05420 - 99111 - TELEX 44200

The following contains a summary of the PASCAL-cross-compiler for the PDP 11 as of November 1976. The reader is expected to be familiar with the publication in "Software, Practical Experience, Vol. 6, 109-116 (1976).
With regard to the definition of the language PASCAL the following restrictions hold:
Files are not implemented, apart from the standard files input and output which are always linked to the keyboard/printer-device.
Packed data structures are only implemented for: one-dimensional character arrays (packing always takes place), and for one-dimensional boolean arrays (packing optional, one boolean per bit).
There is no procedure DISPOSE to delete individual dynamically created structures, but there is a MARK/RELEASE pair which suffices when allocation and deallocation takes place in nested fashion.
Only local jumps are allowed.
The following extensions have been implemented:

- function results can be of non-scaler type,
- arrays with unspecified bounds (but specified index-structure) can be passed as arguments to procedures,
- several standard procedures have been added, notably two procedures to obtain and set the value of specified memory-locations. These procedures are intended for communication with the device registers. Undiscriminated use of these procedures may endanger the implementation,
- procedures may be declared in the outer program block to be associated with specified interrupt-sources. Apart from the level of declaration there are no restrictions on interrupt procedures. Of course interrupt procedures cannot be called in the conventional sense.
- A string parameter type has been introduced in which one dimensional character arrays or substrings thereof may be passed as actual parameters. Such strings and their constituent characters are considered as "read only". The most important application is in passing strings to procedures which may in turn pass on these strings to "write", which up till now was the only procedure to accept strings of unspecified length.

Current Experience.

Experience gained so far shows the implementation quite fit for real time applications.
Recently a number of PASCAL routines have been developped and implemented for the implementation of parallel processes. We mention: queue-operations, process switching, starting of independent processes, P- and V-operations, monitors (à la Hoare) and conditional critical regions, device-drivers, interrupt handlers, asynchronous I/O. These PASCAL-routines have been based on three routines that were written in PDP 11 assembler and consisted of appr. 50 words. These routines performed: register switching, interrupt enabling and disabling, and a bit of address manipulation to make the address of a procedure available to the PASCAL environment. These three routines will soon be implemented in the form of standard-procedures.

Future Developments.

A full implementation of the file-concept to run under RT 11 is expected to be completed shortly. This implementation will only need minor revisions in order to become operational under other DEC operating systems.
In a next phase the development of a simple multiuser operating system centred around a file system (fully implemented in PASCAL) is foreseen.

C. Bron.
November 12th. 1976.
Twente U. of Technology,
P.O.Box 217,
Enschede.
The Netherlands.

PATTERN ANALYSIS & RECOGNITION CORP.

ON THE MALL
ROME, N. Y. 13440
TEL. 315.336.8400
315.724.4072

14 December 1976

PASCAL User's Group
% Andy Mickel
University Computing Center
227 Exp. Engr.
University of Minnesota
Minneapolis, Minnesota 55455

Dear Andy:

We recently received your letter to David Bennett here at PAR, asking about the status of our PASCAL compiler implementation efforts. Here it is:

1) We attempted to construct one from the ill-fated PASCAL P compiler kit, but:

   a) The interpreter subroutines plus threaded code would not fit in 32000 words, our maximum allowed program size, and system overlay facilities were not applicable to interpreter code which looks like data.

   b) The P assembly language was not very suitable. It required tag fields with data and had several other major faults.

   c) My bosses weren't ready to spring for the newer version of the kit after being burned with the first kit.

2) I analyzed what was lacking in the kit and decided it was not nearly as portable as it could be if it translated PASCAL into some language for which everybody already has a translation to machine language (you guessed it -- FORTRAN).

3) I spent 3 months thinking up methods by which PASCAL might be translated efficiently into FORTRAN. Then I began writing this compiler

in PASCAL, translating it by hand into its equivalent FORTRAN and debugging the FORTRAN. I am about halfway (with about 3-5 months of work remaining) towards having a full PASCAL compiler (including recursive procedures) which will produce reasonably efficient FORTRAN.

This is a solo effort on my part, with two goals:

1) Obtain the first PASCAL compiler (to my knowledge) which runs under RSX-11D on the PDP 11/45.

2) Produce a truly portable PASCAL compiler (i.e., one that can be set up with no more work than transporting the compiler source in PASCAL and the compiler object code in FORTRAN to another computer.

Here's a list of information about the project in the same order you requested it.

1) Implementor - Michael N. Condict
   PAR Corporation
   On The Mall
   Rome, N.Y. 13440

   Phone - (315) 336-8400

2) Machine - DEC, PDP 11/45

3) Operating Sys. - RSX-11D

   Minimal Config.- Same as for RSX

4) Method of Distribution - None for at least 5 months.

5) Documentation - Not yet.

6) Maintenance - Not yet.

7) Standard PASCAL? - Yes, probably with extensions.

8) Type of Program - PASCAL to FORTRAN translator, expected to be about 5000 FORTRAN source lines excluding comments, and about 3000 PASCAL source lines. Will be one pass and thus run rings around FORTRAN compiler.

9) Reliability - Will not be distributed until it is.

10) Method of Development - Initial version of each procedure was written in PASCAL then hand translated into a "pretty" FORTRAN preserving all identifiers by use of the PARAMETER statement (allows symbolically named constants). The FORTRAN program (currently 2500 source lines and half finished) was debugged after each addition to it (I am using both top down development and top down debugging by writing "stub" procedures, later replaced with real ones). When the compiler is finished or can compile itself, it will be restored to its original PASCAL in a massive, manual, inverse translation, and run through itself, thus completing the bootstrap.

Man Months - I expect this project to require between 6 and 9 man-months (with 1 man devoting half his time). It has currently consumed about 4 man months.

Experience of Implementor - I built a compiler for a subset of PASCAL as a class project once, but this is already the largest program I have ever attempted.

I hope this answers all questions you may have about our implementation efforts. We are still interested in learning of any existing full PASCAL compiler which runs under RSX on the PDP 11/45, if you are aware of one.

Sincerely,

PAR Corporation

Michael N. Condict

MNC/med

FOXBORO Fox-1

The Foxboro Company Foxboro, Massachusetts, U.S.A. 02035 • Telephone (617) 543-8750

30 November 1976

PASCAL User's Group
c/o Andy Mickel
University Computer Center
227 Experimental Engineering Building
Minneapolis, Minnesota 55455

Dear Andy:

Enclosed you will find my check for $4.00 for membership in the PASCAL User's Group. This is the third (and hopefully not the last) membership application you will receive from The Foxboro Company.

We have recently received inquiries from Tim Bonham and yourself concerning the 'state' of PASCAL at our installation. Unfortunately, there isn't an awful lot of information we can give you. The best we can do is bring you up to date on what's gone on here.

Bob Matherne, who, incidentally, is no longer with our company, was primarily responsible for our interest in PASCAL. He acquired a copy of the PASCAL-P system and spent a significant amount of time developing a P-CODE interpreter (in FORTRAN) on our FOX 1 product system. You may not be familiar with the FOX 1, but it's a minicomputer system primarily intended for industrial process control applications.

At the time that Bob was implementing the interpreter he was beginning some research in the area of System Implementation Languages. He was particularly interested in control structures and data typing capabilities and chose to use PASCAL as an experimental vehicle in these areas. About halfway through the development of the interpreter Bob was joined by a second programmer, Jim Pownell, who completed the interpreter, and who has since become responsible for its maintenance.

Unfortunately, when Bob left the company the project he had been working on came to a close. Since then parts of his project have been included into other projects that are going on now. The use of PASCAL has not been identified as a major part of any of these projects, however.

Our specific implementation of the PASCAL-P system seems to be pretty much the standard PASCAL, as it was distributed. The only change we made was the inclusion of a STOP statement which can be used to terminate execution at any point in a program. Also, because the FOX 1 is a 24 bit/word machine, the size of a set was restricted to 48 members.

FOXBORO

The interpreter Bob and Jim developed does have a few interesting aspects, though. Approximately 14K of memory is used to execute the interpreter, including 14K words of 'stack,' 12K words of P-CODE, and 19K words of code for the interpreter itself. Of course there is a lot of overlaying involved in placing all of this into 14K, and our implementation is, therefore, fairly slow. But since our primary use of PASCAL is as a research tool we can live with this.

We do have a few 'enhancements' built into our interpreter. For example, we have a trace facility that can be controlled through a code in a PASCAL program. This was intended primarily for debugging the interpreter, but has also proven useful in debugging programs written in PASCAL. The output of the trace is variable, based on the type of instruction being interpreted. Basically the output consists of the operation code, operand fields, stack pointer, program counter, and top four locations of the stack. As with most trace facilities, output can be turned on and off at the whim of the programmer.

Another optional output of our interpreter is what we call a 'profile.' This consists of two lists. The first indicates the number of P-CODE instructions, by type, and the percentage of code that occurances of each instruction type constitute. The second list shows, by instruction type, the number of times an instruction of that type was 'executed,' and the percentage of 'execution time' spent in instructions of that type.

It was intended that this profile serve two purposes. First of all, if we ever got around to optimizing our interpreter we would have some idea of most frequently used instructions, etc. Secondly, we could project actual execution time of PASCAL programs on either the FOX 1 or some other system by building tables of instruction execution time. At this time we have done nothing more than produce profiles, however.

We have, at present, four PASCAL programmers (or should I say programmers who can write PASCAL programs) at our installation. I wish I could tell you we use PASCAL heavily, but that's not the case. The higher level language supported by our product systems happens to be FORTRAN and so we are somewhat locked into it. Other than the occasional program one of us might write we are, in fact, doing very little with PASCAL.

I hope the above helps to clarify our present position. Should any change in the status of PASCAL at Foxboro occur I'll let you know. Until then, we hope to keep abreast of what's happening with PASCAL through your newsletter.

Sincerely,

Thomas G. McGinty

Thomas G. McGinty
Computer Section
Corporate Research

pad

IBM System 360/370

STANFORD UNIVERSITY

Stanford Linear Accelerator Center

Mail Address
SLAC, P. O. Box 4349
Stanford, California 94305

December 20, 1976

Timothy Bonham
PASCAL Implementations
University Computer Center
227 Experimental Engineering Building
University of Minnesota
Minneapolis, Minn. 55455

Dear Tim:

Your letter of inquiry about the SLAC PASCAL Compiler was (eventually) directed to me and I apologize for the (very) long delay in the process of responding.

Our PASCAL Compiler for the IBM 360/370 system is based on the P-compiler (the P2, May 74 version) and, with a few minor extensions, implements exactly the same language.

During the bootstrap process, the output of the P-compiler was translated by a set of IBM Assembler-H Macros into 360/370 Assembly language and then into object code. The resulting code was overinflated and fairly inefficient. Currently, we have a post processor (a so-called P-Translator) which is a 2500 line PASCAL program that translates the output of the P-compiler into either 360/370 assembly languge or directly produces a standard OS object module.

In addition to fixing some of the obvious bugs (and some not so obvious) in the P2 Compiler, we had to modify the compiler and the syntax of the P-Instructions in order to specify the type of the operand(s) of some instructions (in particular Load, Store type operations) before a reasonable translator to 370 code could be written. In other respects, the Syntax Analyzer is almost as it was in the P2 compiler.

The P-Translator is a one-pass, no lookahead translator which could be incorporated into the compiler, or ideally, communicate with it in coroutine fashion but, for simplicity, we run it as a post processor to the P-compiler with somewhat increased I/O overhead. The current version of the P-Translator (like the P-Compiler) does not provide run time checking of indices or subranges.

Timothy Bonham Page two of two pages

Some statistics:

	P-Compiler	Post Processor
lines of source code	4000	2500
# of P-Instructions	15000	9500
bytes of 370 instructions	76000*	52000*
time to compile the compiler	10 sec.+	5 sec.+ (total 15 sec)
processing rate	400 lines/sec.	800 (source) lines/sec.

*this includes the I/O interface and conversion routines
+timed on the 370/168 with 16k cache memory.

The Compiler/Post-processor is capable of compiling (bootstrapping) itself in a 130k byte region, 24k of which is returned to the operating system for I/O buffers.

The entire system is available to the public (as is) and we are working on further optimization of the object code.

Once again, sorry for the delay in our response. Please feel free to write me if you need more detailed information.

Sincerely,

Sassan Hazeghi
Computation Research Group

SH/hlc

STAUNTON, VIRGINIA 24401 COMPUTER CENTER

IBM 1130

Mary Baldwin College

December 10, 1976

Mr. Timothy Bonham
Pascal User's Group
University Computer Center
227 Experimental Engineering Building
University of Minnesota
Minneapolis, Minnesota 55455

Dear Tim:

Indeed I am interested in an IBM 1130 implementation of PASCAL but little has been done so far. The intent is still very much alive, but the project remains in a planning stage. I am leaving Mary Baldwin College at the end of the month, so please note my change of address below. If you would, please see that it also gets to the PASCAL User's Group. I will still be using an IBM 1130 Computing system and I still intend to work on the PASCAL project, so I'll let you know when something is available. My new address is:

Innovative Management Systems
865 Middlebrook Avenue
Staunton, Virginia 24401

Sincerely,

Fred W. Powell
Director

cc: Mr. James M. Moloney
Mr. Joe C. Roberts

INTERDATA 8/32

Mike Ball at the Naval Undersea Center, San Diego reports that Paul Fisher at Kansas State University has a slow version of Brinch Hansen Pascal on the Interdata 8/32 which ignores lower case characters. Mike is currently implementing Pascal himself on the 8/32, changing Brinch Hansen's compiler to a 9-pass highly optimizing compiler.

Mike passed on the observation that it took Brinch Hansen's com iler took three times as long to bootstrap but had far fewer errors than the P-compiler. He would like to know other people's implementation experiences. (* See the CHECKLIST, item 10. *)

UNIVAC 90/70 (*No known implementations; see Here and There under Hopkins.*)

UNIVAC 1100 SERIES

Mike Ball of the Naval Undersea Center, San Diego, reports that he has received many more requests than he expected for his Univac 1100 Pascal compiler. He has talked to Fischer and LeBlanc at the University of Wisconsin, Madison,WI about their compiler. They are using his to hlp them write a diagnostic, checkout compiler based on LR(k) scanner with variant record and pointer checking. This should be useful for students. They are also concentrating on code optimization.

We received the following more complete news of the 1100 implementation from DIKU, University of Copenhagen. It appears that for this compiler, the source code is not released. However, the documentation appears good; a nice (unfortunately faint) 19 page machine retrievable manual was sent to us entitled: "A Pascal Compiler for Univac 1100 machines" by J. Steensgaard-Madsen and Henrik Snog at the Datalogisk Institut, University of Copenhagen. It is in the form of a supplement to Pascal User Manual and Report. This compiler has 6-bit characters as standard type CHAR but does supply an additional type ASCII. ALFA then is a predefined type of packed array[1..12] of CHAR and there is another predefined type HALFA which is packed array[1..6] of CHAR (or 6-6 bit characters per 36 bit word). Set sizes are only 72 elements as compared to 144 in Mike Ball's compiler.

DIKU DATALOGISK INSTITUT KØBENHAVNS UNIVERSITET
SIGURDSGADE 41, DK-2200 KØBENHAVN, DANMARK, TLF. (01) TA 94 66

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

November 11, 1976

JSM/HG

Dear Timothy.

Thank you for your letter of 25 Oct. Contrary to your expectations I have not received Newsletter no. 5 and neither has our library.

To assist you in collecting information on Pascal implementation I enclose a short characterization of our Pascal system, a Request Form and our Users Guide, which should be considered a supplement to the book "Pascal User Manual and Report" by K. Jensen and N. Wirth.

Yours sincerely

J. Steensgaard-Madsen

J. Steensgaard-Madsen

encl.

DIKU DATALOGISK INSTITUT KØBENHAVNS UNIVERSITET
SIGURDSGADE 41, DK-2200 KØBENHAVN, DANMARK, TLF. (01) TA 9466

Request for Pascal 1100.

Requestors name and institution.

Distribution policy.

The Pascal system is distributed, in relocatable form only, to university computing centers.
We aim to build a distribution tree. Will you assist in this and forward a copy of the system to (say) at most 5 Pascal 1100 users on request from Datalogisk Institut? yes / no.
If so, how much will you have to bill for administration and magnetic tape assuming that the users are situated at your continent? _____
If distributed from Datalogisk Institute the charge will be Dkr. 200.

Conditions for distribution:

A. The system will be made available free of charge to all users of the computer center, at which the system is installed.
B. The system must not be distributed further without written approval from Datalogisk Institut, University of Copenhagen.
C. After installation the availability will be announced locally by the requestor, who will also provide a guide for obtaining the Pascal 1100 User Manual provided along with the system.
D. The system is accepted in relocatable form, without promise for maintenance. If errors are detected these will be reported to Datalogisk Institut through a local contact person, who is supposed to be the requestor initially.

Parameters.
Magnetic tape recording (without label) using
___ tracks and density _____ bpi.
EXEC-8 level : _____ .

A signed copy of this form should be sent to

J. Steensgaard-Madsen.
DIKU
Sigurdsgade 41
DK-2200 Copenhagen N.
Denmark.

An unmodest characteristic of a Pascal compiler for UNIVAC 1100.

Pascal 1100A :
An extreemly fast compile-and-go batch system.
Pascal 1100S :
A compiler generating comprehensible assembler code. A true extension of version A.

Common characteristics :

Deviations from Standard Pascal.
A. Any TEXT variable F will after RESET(F) fulfill:
EOF(F) = FALSE and EOLN(F) = TRUE
(This allows a program to open a dialog with a user).
B. Parameters of formal procedures and functions may be of any kind, but specification is required.

Restrictions.
A. File components containing files should not be used.
B. Standard procedures cannot be passed as actual parameters.
C. The value of ORD(X) for X belonging to the base type of a set must be in the interval 0..71.
D. The first operation (in the dynamic sense) on any file must be REWRITE.

Extensions.
A. A scheme for exhaustive specification of parameters.
B. Constant definitions according to the syntax
<identifier> : <type> = <values>;
C. An OTHERWISE clause in a CASE statement.
D. A LOOP ... EXIT ... END statement.

Machine dependent facilities.
A. Handling of both Fieldata and ASCII characters and files.
B. REAL is double precision always.
C. Dynamic association of external files.
D. Smooth cooperation with the operating system.

Pascal 1100S features.
A. External procedures and functions may be written in Pascal or (ASCII) Fortran.
B. Inclusion of assembler code is possible.

Compiler performance.
A. Both compilers requires a bit more than 40K words to execute.
B. Compilation speed is roughly 100 lines per SUP second (SUP = Standard Unit of Processing, defined by UNIVAC).
C. Compiler reliability is excellent.
D. Compiled programs run efficiently compared to other processors.

UNIVERSITY OF MINNESOTA
TWIN CITIES

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

19 November 1976

Jorgen Steensgaard-Madsen
Datalogisk Institut
Kobenhavns Universitet
Sigurdsgade 41
DK-2200 Kobenhavn
DENMARK

Dear Jorgen,

Thank you for the information on your Univac 1100 Pascal. We received it November 15, too late to be printed in Pascal Newsletter #6 (which is 90 pages). It will definitely appear in #7.

Your letter indicates that there was some confusion on our part that I would like to clear up. First, we received the subscription from your library in mid-October. The mailing of issue #5 has been delayed because we have been negotiating with Judy Mullins at the University of Southampton, England about European mailing. This issue has now been air mailed directly to your library, but the library will receive further issues from Southampton.

Second, we sent letters similar to the one you received to over 80 other known implementors. We wrote two versions of the letter, one for PUG members, and one for non-members. We have no record of your membership, so we apparently sent you the wrong version. Please accept my apologies for this mistake.

I would like to take this opportunity to personally invite you to join the Pascal User's Group. This is an informal group of users, teachers, implementors and just plain fans of Pascal. We now have 528 members in 22 countries. We have published and mailed issue #5 (64 pages). Issue #6 is now at the printer. I am enclosing a membership coupon.

Thank you again for the Pascal 1100 information. It is important to us that we keep the Implementation Notes section current and correct. Your documentation will certainly help in Newsletter #7.

Sincerely,

John P. Strait

John P. Strait

Enclosures.

XEROX SIGMA 6/SIGMA 9

PASCAL - SIGMA (2.0)

NOTES ON THE DISTRIBUTION PACKAGE

Distribution includes a magnetic tape containing the latest system release and a package of documentation. All programs are commented in English but documentation is in French (except for the "Program description").

Distribution costs, which are $250.00, do not imply any responsibility or maintenance service on the part of the distributor, implementor or the Université de Montréal. Distribution costs may be paid by sending a check (payable to Pierre Desjardins) to the address below.

PIERRE DESJARDINS
Université de Montréal
Informatique
C.P. 6128, MONTREAL 101
Québec, CANADA

Release: September 1976
Implementor: Pierre Desjardins
Distributor: Pierre Desjardins

COMPILER CHARACTERISTICS

1. The compiler is a 6220 lines Pascal-Sigma program (Pascal-Sigma corresponds to Wirth and Jensen's "Pascal: user manual" together with the restrictions cited below);

2. it was obtained by cross-compilation on a CDC CYBER74, using the Nov.1972 version of Pascal from Zurich;

3. without any prior knowledge of Sigma machines, the entire programming system was completed in 18 man months;

4. reliability is considered "good to excellent";

5. language restrictions are as follows:
   - sets are limited to 32 elements,
   - standard procedures (functions) are not allowed as actual parameters,
   - array, record and file types cannot have file type components,
   - string constants cannot be defined in the const part of the declarations.

6. non-standard extensions to the language include:
   - keyed files,
   - ghost variables

7. peak code size is 25K;

8. self-compilation requires 35K (peak) of core and executes at the rate of
   - $\approx$ 600 lines/min. on Sigma 6 (BPM/BTM)
   - $\approx$ 1200 lines/min. on Sigma 9 (CP-V)

9. generated code takes the form of a file of relocatable object modules (one for every procedure (function)) in "Xerox Standard Object Language".

TAPE STRUCTURE AND FORMAT

Tape format is: 9 tracks, 800 bpi, binary.

The tape is structured much like a standard Xerox processor distribution tape (labelled tape in account :SYSGEN). As specified in the "Program Description", file $$CONTENT displays the contents of the tape.

MAINTENANCE POLICY

As specified earlier, distribution costs do not include "on-demand" maintenance service. However, bug reports are welcomed so that once they have been seen to and a few of them have been accumulated, update sheets could be sent to Pascal-Sigma users.

Bug reports should be as explicit as possible as to the nature of the bug and the environment in which it occured.

DOCUMENTATION

Except for the "Program Description" document, all other documents are written in French.
Here is a list of all documents provided in the destribution package together with a breif description of their use:

1. "Program Description", contains information related to the installation and maintenance of Pascal-Sigma.

2. "Manuel d'utilisation....", is the user's manual. It gives information related to the particularities of Pascal-Sigma: restrictions, extensions and use under BPM/BTM.

3. "METAPASC: .....", will be of interest to people expecting to use external procedures or functions, written in Meta-symbol, inside a Pascal program. METAPASC provides a set of macro-procedures destined to releive the user of interface details: declaration of parameters and local variables, parameter passing mechanisms (generation of data-segments), retreiving from (storing in) the data-segment by symbolic reference, ...

4. "Pascal 2 - Sigma: un système de programmation Pascal", provides a description of the "functional structure" of the Pascal-Sigma compiler: the major logical components of the compiler are explained together with the way by which they communicate information. This document was written with the following intention: to provide a description of the major parts of a large program (the compiler) and a global view of its stategy, without annoying the reader with too many low-level details (which he can easily investigate in the source text once he understands the structure).

XEROX SIGMA 7

DEPARTMENT OF COMPUTER SCIENCE
COLLEGE OF ARTS AND SCIENCES

BOX 3682, UNIVERSITY STATION
PHONE 307 766-5190

THE UNIVERSITY OF WYOMING
LARAMIE, WYOMING 82071

November 9, 1976

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

Dear Sir:

I am responding to Timothy Bonham's letter inquiring about the Pascal efforts in the Computer Science Department at the University of Wyoming. I will respond in order to the ten points in the letter.

1. I am the only one here who has supervised any students who are interested in implementing Pascal. Mainly student projects have been involved.
   Implementor: Henry R. Bauer, III (307) 766-5134
   Computer Science Dept.
   University of Wyoming
   Box 3682
   Laramie, WY 82071

2. We have had two projects.
   Project 1: Xerox Sigma 7 implementation of Pascal
   Project 2: Transfer of Brinch Hansen's Concurrent Pascal to the Texas Instruments 980A
3. Operating Systems
   Project 1: CP-V
   Project 2: Stand-alone written in PL980 for a machine with 16K of 16-bit word memory with 2 AED floppy disk drives
4. No distribution.
5. No documentation.
6. No maintenance.
7. Implementations incomplete.
8. Project 1: A. One product is an interpretter for the Pascal
9. P-code compiler in Metasymbol macros. Running size is about
10. 40K and therefore not too practical. Speed is slow. Currently the project is shelved but recent student interest may lead us to dusting it off.
    B. A second product is to be a bootstrap of a Paris version of Pascal for the CII-IRIS 80 and 10070 computers using a neighboring CDC6400. Project was dropped when the University Computer center ordered a Pascal from another University; their compiler has not yet arrived.
    Project 2: The concurrent Pascal version is not completely operational but no current efforts are underway.

I hope that this information updates your records on our Pascal efforts. I am willing to answer any inquiries others may have.

Yours sincerely,

Henry R. Bauer, III
Assistant Professor
of Computer Science

PASCAL USER'S GROUP

USER'S

GROUP

ALL PURPOSE COUPON

****

Clip, photocopy, or reproduce, etc. and mail to:

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

(phone: (612) 376-7290)

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

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

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

/ / You messed up my address. See below.

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

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

/ / None of the above. _____

_____

_____

_____

Other comments:

From: name _____

address _____

_____

_____

_____

phone _____

date _____

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

return to:

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

return postage guaranteed