PASCAL USER'S GROUP

USER'S GROUP

PASCAL NEWSLETTER

NUMBER 5

COMMUNICATIONS ABOUT THE PROGRAMMING LANGUAGE PASCAL BY PASCALERS

SEPTEMBER, 1976

TABLE OF CONTENTS

LICY -- PASCAL USER'S GROUP AND PASCAL NEWSLETTER

ER'S GROUP POLICIES

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

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

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

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

NEWSLETTER POLICIES

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

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

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

Miscellaneous features include bibliographies, questionaires, and membership lists.

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

UNIVERSITY OF MINNESOTA
TWIN CITIES

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

EDITOR'S CONTRIBUTION

PART I - In General

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

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

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

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

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

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

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

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

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

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

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

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

PART II - Pascal at the University of Minnesota

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

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

PART III - My Concerns

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

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

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

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

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

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

- Andy

August 10, 1976

HERE AND THERE WITH PASCAL

(News from members, Conferences, New books, Applications programs, etc.)

CONFERENCES

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

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

NEW BOOKS

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

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

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

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

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

NEWS (alphabetical by last name)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Judy Mullins, Department of Mathematics, The University, Southampton United Kingdom SO9 5NH (PUG member): "I enclose ... $8 ... for two year's subscription to the Pascal Users' Group and Newsletter. This advance payment was prompted by the large banker's commission on small sums such as $4....

"Arising from this, I was wondering whether it would be useful or possible to a arrange some kind of branch of P.U.G. in the U.K. for collecting subscriptions. ... a convenient and cheaper form of membership may encourage more members in the U.K. There are certainly many institutions using Pascal now, and interest is spreading....

"As co-organizer with Prof. Barron of the Pascal Symposium for next March, I could get the thing started. Others in our group are writing a Pascal compiler for ICL's new 2970 computer, so we shall always have a vested interest in Pascal."

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

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

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

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

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

N. Solntseff, Dept. of Applied Mathematics, McMaster University, Hamilton, Ontario Canada L8S 4K1 (PUG member): "I am interested in participating in the users' group and am willing to contribute my time in any capacity.

"Incidentally, I would be happier if the title of the group were the more grammatical "Pascal Users' Group". "

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

"Will the User's Group have any affiliation with the current distribution center at the University of Colorado?"

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

"I was interested that the Pascal usage at Minnesota has exceeded that of RUN and FTN. Have you been using it in an introductory course? We plan to do so next fall, and I would appreciate any comments you have.
"It seems to me that one of the most difficult problems faced by Pascal is one that could be considered irrelevant: the ordering of multidimensional arrays. Since the language definition implies row-major order, it seems that Pascal-FORTRAN communication might be very difficult. How have you handled the problem when perverting FORTRAN library routines?"

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

Lecture Notes in Computer Science, No. 18

(*When purchasing, specify the "Springer Study Edition" - it's 35% cheaper!*)
PASCAL - USER MANUAL AND REPORT by K. Jensen and N. Wirth, Springer-Verlag, 1974, 1975, 167 pages, paperbound.

Corrections to 2nd Edition

p.51, l.16: "setop(output)" → "setop(output);"
p.56, l.-6: "fi" → "f(i)"
"g(i+1)" → "g(j+1)"
"gi" → g(j)
p.63, Fig.10a: Number sequence should be reversed.
p.69, l.23: "stricly" → "strictly"
p.77, l.18: move line 3 places to the left
p.98, l.10: append " ' ';"
p.102, l.7: last word should be "or"
l.20: "bufffer" → "buffer"
p.103, l.-6: "scaler" → "scalar"
l.-7: "char, and alfa" → "and char are listed"
p.124, l.14 and l.15: "14" → "15"
p.127, l.27: "18.A" → "4.A"
p.133, l.3: "two" → "to"
p.135, l.5: "althought" → "although"
l.30: "subtstitute" → "substitute"
p.140, l.11: "structure type" → "structured type"
p.158, delete lines -12... -8.

N. Wirth
10.May 1975

Additional Suggested Corrections

September 26, 1975 University Computer Center
University of Minnesota

Page/Line

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

ARTICLES (FORMAL SUBMITTED CONTRIBUTIONS)

Designing Data Structures by Step-wise Refinement

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

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

Abstract

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

Overview

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

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

The Queens Problem Revisited

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

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

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

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

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

Dijkstra tests whether a square is free by noting:

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

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

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

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

The Soma Cube

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

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

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

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

The Soma Cube Data Structure

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

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

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

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

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

is appropriate.

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

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

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

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

Maxlist will be a declared constant.

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

```
positions = set of locations;
```

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

```
locations = 1..27;
```

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

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

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

How Will the Soma Program Work?

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

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

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

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

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

```
somacube := positionlist [whereat] join somacube;
```

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

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

Summary

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

(*Received 2/21/76*)

References

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

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

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

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

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

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

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

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

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

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

   generate
end.
```

Figure 1

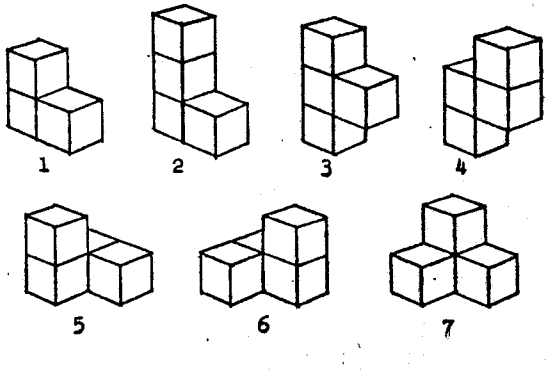

Figure 2

To the Editor:

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

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

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

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

Another Solution to the Soma Cube Puzzle
Michael N. Condict
Lehigh University

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

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

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

References

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

(*Received 3/11/76*)

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

```
CONST
   LASTCUBE= 26;
   K3321=2+9;

TYPE
    CONFIGSETNOS= 0..28;
      CUBETYPES= (EDGE, CORNER, FACE, CENTER);
    COMBINATIONS= 0..11;
 PIECECONFIG... AT= 0000..K3321;
   AMTSFORCONFIG= 0..50;
          HOLES= 0..24;
          SIDES= 1..6;
           AXES= (Z,Y,X);
          CUBES= 0..LASTCUBE;
       CUBESPI= -1..27;
         PIECES= 1..7;
      PIECESPI= 1..8;
     POSITIONS= 0..145;
    PIECECUBES= 1..6;
 PIECEPOSITIONS= SET OF 0..58;
        SHIFTS= -26..LASTCUBE;
     THEPIECES= ARRAY [PIECES] OF
                RECORD
                   PACKED: ARRAY [POSITIONS] OF PIECEPOSITIONS;
                   UNPACKED: ARRAY [PIECECUBES] OF CUBES;
                   PRINTLIST: PIECEPOSITIONS;
                   CONFIGCOMBINATION:
                      ARRAY [COMBINATIONS] OF CONFIGSETNOS;
                   NAME: CHAR;
                   FIRSTCONFIG,
                   CURRENTCONFIG,
                   LASTCONFIG: CONFIGSETNOS;
                END /*RECORD*/;
      THECUBES= ARRAY [CUBESPI] OF
                RECORD
                   ALLOWEDSHIFT: SET OF 0..15;
                   ROTATEABOUT: ARRAY [AXES] OF CUBES;
                END /*RECORD*/;
VAR
   FIRSTPOSITION,
    LASTPOSITION: ARRAY [CONFIGSETNOS] OF POSITIONS;
      NUMBEROF: ARRAY [CUBETYPES,CONFIGSETNOS] OF 0..4;
     COMBNUMBER: COMBINATIONS;
        COUNTOF: ARRAY [CUBETYPES] OF 0..14;
      CUBETYPE: CUBETYPES;
    MAXALLOWED: ARRAY [CUBETYPES] OF 0..12;
 SOLUTIONNUMBER: INTEGER;
      TESTSHAPE,
  SOLUTIONSHAPE: PIECEPOSITIONS;
          CUBE: CUBESPI;
         PIECE: PIECESPI;
          SOMA: THEPIECES;
          CHS: THECUBES;
     HOLESIZE: HOLES;
     HOLECUBE: ARRAY[HOLES] OF CUBESPI;
       TYPEOF: ARRAY [CUBES] OF CUBETYPES;
  ROTATEVALUES: ARRAY [AXES,CUBES] OF CUBES;
   PIECEVALUES: ARRAY [PIECES,PIECECUBES] OF CUBES;
    NAMEVALUES: ARRAY [PIECES] OF CHAR;
    ADJACENTTO: ARRAY [CUBESPI] OF
                RECORD
                   NUMBERADJSIDES: SIDES;
                   NEIGHBORING: ARRAY [SIDES] OF CUBESPI;
                   CUBESURROUNDED: PIECEPOSITIONS;
                END;
VALUE
   TYPEOF=
      (CORNER,EDGE,CORNER, EDGE,FACE,EDGE, CORNER,EDGE,CORNER,
       EDGE,FACE,EDGE, FACE,CENTER,FACE, EDGE,FACE,EDGE,
       CORNER,EDGE,CORNER, EDGE,FACE,EDGE, CORNER,EDGE,CORNER);
   ROTATEVALUES=
    ( 6, 3, 3, 7, 4, 1, 8, 5, 2,
     15,12, 9,16,13,10,17,14,11,
     24,21,18,25,22,19,26,23,20,
      2,11,20, 5,14,23, 8,17,26,
      1,10,19, 4,13,22, 7,16,25,
      0, 9,18, 3,12,21, 6,15,24,
     19,19,28, 9,10,11, 0, 1, 2,
     21,22,23,12,13,14, 3, 4, 5,
     24,25,26,15,16,17, 6, 7, 8);
   PIECEVALUES=
    ( 0, 1, 4, 7,
      9,18,19,21,
     10,11,20,23,
     10,11,13,20,
      1, 3, 4, 7,
      0, 1, 4, 5,
      1, 3, 4, 4);
   NAMEVALUES= ('L', 'V', 'S', 'R', 'T', 'Z', 'V');
   ADJACENTTO=
      /*CUBE  # ADJ        NEIGHBORS          FILLER      */
      /*-----  -----  ------------------   --------------*/
      ( -1 ,   0,                         0,0,0,0,0,0,0,
      (  3 ,   3,              1, 3, 9,    0,0,0,0,0,
      (  1 ,   4,           0, 2, 4,10,    0,0,0,
      (  2 ,   3,           1,    5,11,    0,0,0,0,
      (  3 ,   4,        0,    4, 6,12,    0,0,0,
      (  4 ,   5,        1, 3, 5, 7,13,    0,0,
      (  5 ,   4,        2, 4,    8,14,    0,0,0,
      (  6 ,   3,        3,    7,   15,    0,0,0,0,
      (  7 ,   4,        4, 6, 8,   16,    0,0,0,
      (  8 ,   3,        5, 7,      17,    0,0,0,0,
      (  9 ,   4,     8,       10,12,19,    0,0,0,
      ( 10 ,   5,     1,    9,11,13,19,    0,0,
      ( 11 ,   4,     2,   10,   14,20,    0,0,0,
      ( 12 ,   5,     3, 9,    13,15,21,    0,0,
      ( 13 ,   6,     4,10,12,14,16,22,    0,
      ( 14 ,   5,     5,11,13,   17,23,    0,0,
      ( 15 ,   4,     6,12,   16,   24,    0,0,0,
      ( 16 ,   5,     7,13,15,17,   25,    0,0,
      ( 17 ,   4,     8,14,16,      26,    0,0,0,
      ( 18 ,   3,     9,       19,21,      0,0,0,0,
      ( 19 ,   4,    10,   18,20,22,       0,0,0,
      ( 20 ,   3,    11,   19,   23,       0,0,0,0,
      ( 21 ,   4,    12,19,   22,24,       0,0,0,
      ( 22 ,   5,    13,19,21,23,25,       0,0,
      ( 23 ,   4,    14,20,22,   26,       0,0,0,
      ( 24 ,   3,    15,21,   25,          0,0,0,0,
      ( 25 ,   4,    16,22,24,26,          0,0,0,
      ( 26 ,   3,    17,23,25,             0,0,0,0,
      ( 27 ,   1,    27,                   0,0,0,0,0,0,0);
```

```
PROCEDURE PRINTSET(WORD: PIECEPOSITIONS);
   VAR I: 0..58;
   BEGIN
      WRITE('(');
      IF WORD ISNT [] THEN
      BEGIN
         I:=9;  WHILE NOT (I IN WORD) DO I:=I+1;
         WRITE(I:2);
         FOR I:=I+1 TO 58 DO IF I IN WORD THEN WRITE(',',I:2);
      END /*IF[]*/;
      WRITE(')');
   END /*PRINTSET*/;


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


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


   PROCEDURE ROTATE(W:AXES);
      VAR CUBE: 1..6;
      BEGIN
         FOR CUBE:=1 TO 4 DO WITH SOMA[PIECE] DO
            WITH ONE[UNPACKED[CUBE]] DO
               UNPACKED[CUBE]:=ROTATEABOUT[W];
      END /*ROTATE*/;


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


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