PASCAL NEWSLETTER

January 1974 Number 1

FROM THE EDITOR

This is the first issue of a newsletter sent to users and other interested parties about the programming language PASCAL. Its purpose is to keep the PASCAL community informed about the efforts of individuals to implement PASCAL on different computers and to report extensions made to the language. It will be published at infrequent intervals due to the limited manpower.

Everyone is encouraged to report to the editor any item of interest to the community. There are many examples given below. Of particular interest are reports of successful implementations on other computers that the author is willing to distribute and modifications to the CDC version to surmount problems or add features. Also, programs written in PASCAL are of interest. However, a plethora of incompatible dialects of PASCAL must be avoided.

The University of Colorado, thru the editor, is willing to support the distribution of PASCAL programs and documents subject to the limitations of its hardware. This is a CDC site with seven track industry compatible NRZI tape drives. To help defray our costs, there is a charge of 5¢ per page for documents, $5 for copying and postage for a magnetic tape from a user and $15 if the University supplies the tape (a mini reel). Send to:

George H. Richmond
University of Colorado
Computing Center
3645 Marine Street
Boulder, Colorado 80302

CURRENT PASCAL COMPILER

The current compiler is dated December 15, 1972, and now incorporates a correction written November 15, 1973. It is available on a 556 bpi SCOPE 3.2 internal binary tape. All CDC 6000 sites should be able to read the tape under any standard CDC operating system. Other tape formats can be arranged on request.

There are six files on the tape. The first two are the source and binary of PASCAL as distributed by Dr. Wirth, of Switzerland, the author of the language. The next two files are the University of Colorado version of the source and binary of PASCAL. The changes are confined to: 1) adapting PASCAL to KRONOS 2.0, 2) deleting the frequency count feature, and 3) providing alphabetic alternatives to some special characters for terminal users. The source file is an UPDATE version 1.2 sequential OLDPL of the 72/12/15 release with the 73/11/15 correction applied as a modification.

All PASCAL users are urged to use this as a base from which to modify the compiler. The fifth file is a PASCAL Manual adapted from the one written by Wilhelm Burger at the University of Texas at Austin. The last file is the text of the modsets incorporated in the OLDPL.

The following items are supplied in the documentation package.

TITLE	DATE	PAGES
The PASCAL System, Documentation, and Literature	Nov 73	4
The PASCAL Distribution Tape	Dec 73	1
Notes accompanying the PASCAL Tape of	15 Dec 73	8
A note to users of the PASCAL Language	15 Dec 72	6
Planned changes to the programming language PASCAL	Jun 72	7
Changes to the programming language PASCAL	15 Feb 72	8
The PASCAL Operating System POSYS	28 Oct 71	20
Advantages of the value parameter over the constant parameter	5 Jul 72	3
Run-Time Check Options	Dec 72	1
How to use the PASCAL 6000 System	23 Jun 72	3
The Standard Procedure WRITE and Post-Mortem Dump	23 Jun 72	4
The Programming Language PASCAL (Revised Report)	Nov 72	53
An Axiomatic Definition of the Programming Language PASCAL	Nov 72	32

This version of the compiler does not implement class variables as described in the Revised Report. The following description of them comes from the original PASCAL report.

6.2.5 Class types

A class type definition specifies a structure consisting of a class of components, all of the same type. The number of components is variable; the initial number upon declaration of a variable of class type is zero. Components are created (allocated) during execution of the program through the standard procedure new. The maximum number of components which can thus be created, however, is specified in the type of definition.

<class type> ::=class <maxnum> of <type>

<maxnum> ::= <integer>

6.2.6 Pointer types

A pointer type is associated with every variable of class type. Its values are the potential pointers to the components of that class variable (cf.7.5) and the pointer constant nil designating no component. A pointer type is said to be bound to its class variable.

<pointer type> ::= ↑<class variable>

<class variable> ::= <variable>

7.2.4 Referenced components

Components of class variable are referenced by pointers.

<referenced component> ::= <pointer variable>↑

<pointer variable> ::= <variable>

Thus, if p1 is a pointer variable which is bound to be class variable v, p1 denotes that variable and its pointer value, whereas p1↑ denotes the component of v referenced by p1.

Examples:

p1↑. father

p1↑. elder sibling↑. youngest child

10.1.2 Class component allocation procedure

new(p) allocates a new component in the class to which the pointer variable p is bound, and assigns the pointer designating the new component to p. If the component type is a record type with variants, the form

new(p,t) can be used to allocate a component of the variant whose tag field value is t. However, this allocation does not imply an assignment to the tag field. If the class is already completely allocated, the value nil will be assigned to p.

An error in code generation was recently found and corrected as shown below. It occured when an integer variable I and a real variable X are used in expressions like I+1+X.

Procedure SIMPLEEXP before fix:

```
PROCEDURE SIMPLEEXP ;
VAR LATTR :  ATTR ; LADOPCL  :  SHRTINT ; LFB,BT1,BT2 : BOOLEAN ;
BEGIN LFG := FALSE ;
  IF NO = 7 THEN {ADDOP}
  BEGIN IF CL = 2 THEN LFG := TRUE ELSE IF CL = 3 THEN ERROR(51) ;
    INSYMBOL ;
```

```
  END ;
  TERM ;
  IF LFG∨(NO = 7) THEN
  BEGIN WITH LATTR DO
    BEGIN TYPTR := GATTR.TYPTR ; KIND :=LVAL ; CTERM :=0 ;
      IF TYPTR ≠ NIL THEN
      BEGIN TRANSFER(GATTR,RP) ;
        IF LFG THEN
        BEGIN GEN15(13B,0,0,0) ;
          IF (TYPTR↑.FORM = NUMERIC)∨(TYPTR = REALPTR) THEN
          GEN15(37B,RP,0,RP) ELSE ERROR(50)
        END
      END

  END ;
  WHILE NO = 7 DO
  BEGIN LADOPCL := CL ; INSYMBOL ; TERM ;
    IF (LATTR.TYPTR ≠ NIL)∧(GATTR.TYPTR ≠ NIL) THEN
```

Procedure SIMPLEEXP after fix:

```
PROCEDURE SIMPLEEXP ;
VAR LATTR : ATTR ; LADOPCL : SHRTINT ; LFG,BT1,BT2 : BOOLEAN ;
BEGIN LFG := FALSE ;
  IF NO = 7 THEN {ADDOP}
  BEGIN IF CL = 2 THEN LFG := TRUE ELSE IF CL = 3 THEN ERROR(51) ;
    INSYMBOL ;
  END ;
  TERM ;
  IF LFG∨(NO = 7) THEN
  BEGIN WITH LATTR DO
    BEGIN TYPTR := GATTR.TYPTR ; KIND := LVAL ; CTERM :=0 ;
      IF TYPTR ≠ NIL THEN
      BEGIN TRANSFER(GATTR,RP) ;
        IF LFG THEN
        BEGIN GEN15(13B0,0,0) ;
          IF (TYPTR↑.FORM = NUMERIC)∨(TYPTR = REALPTR) THEN
          GEN15(37B,RP,0,RP) ELSE ERROR(50)
        END
      END
    END ;
    WHILE NO = 7 DO
    BEGIN LADOPCL := CL ;
      IF LATTR.KIND = LVAL THEN
         IF LATTR.CTERM ≠ 0 THEN
            BEGIN GEN30(71B,0,0,LATTR.CTERM) ; GEN15(36B,RP,RP,0);
               LATTR.CETERM := 0
            END;
         INSYMBOL; TERM;
      IF (LATTR.TYPTR ≠ NIL)∧(GATTR.TYPTR ≠ NIL) THEN
```

COST OF THE PASCAL PACKAGE

There is a charge of $30 the first time a PASCAL tape and documentation package is sent. Deduct $10 if a tape is supplied and deduct $10 if you have previously received PASCAL from us.

FORTHCOMING VERSIONS OF THE COMPILER

An entirely new compiler called PASCAL 2 is under development. It implements the language PASCAL as defined in the Revised Report (Nr. 5) with a few extensions (see below). It is implemented for the operating system SCOPE 3.4 and will be available in two versions:

1) For the CDC scientific 64-character set, and
2) For the ASCII 64-character set (with CDC's ordering).

This implies that no explicit line control characters (eol) are available; instead, ends of lines in textfiles are generated and recognized by additional standard procedures and functions. This, unfortunately, requires changes in most existing programs - although they may be clerical only. Hence, the distinct name PASCAL 2 was chosen.

The (other) principal new characteristics of the compiler are:

1) It generates relocatable binary object code which can be loaded by the standard CDC loader.

2) It allows procedures (and functions) to be separately compiled and merged at load-time.

3) It provides a facility to use subroutines written in other languages. In these cases, the standard FTN calling sequence will be generated.

4) It introduces packed arrays. They can be treated like regular arrays, but will be allocated with as many components as possible packed into each word. Accordingly, access to individual elements will be slower.

5) It introduces so-called segmented files; each segment corresponds to a logical record (in CDC terminology).

6) External files may be passed to the program as parameters in a program heading.

We hope to be able to release the new compiler by May, 1974 along with a User Manual.

The PASCAL-P system is a compiler which generates code (so-called P-code) for a simple, hypothetical stack computer. This computer is described in the form of a PASCAL program as a loader and interpreter of P-code.

The PASCAL-P system was developed in order to simplify the implementation of PASCAL on other machines. The method to proceed is in general (i.e., without access to a CDC 6000 computer) the following:

1) Program the loader interpreter in any (probably assembly) language for the target machine M . PASCAL-programs can now be executed interpretively on M , since the PASCAL-P compiler is available in P-code.

2) Rewrite the PASCAL-P compiler by replacing the P-code generators by routines generating code for machine M.

3) Interpretive compilation of the modified compiler then yields a compiler in P-code which generates M-code.

4) Recompilation of the modified compiler by itself then results in a PASCAL compiler in the form of M-code and generating M-code.

The PASCAL-P system is available under the same conditions as the CDC compiler. It is delivered in source form (a PASCAL program) and in P-code form (coded as a string of characters).

Note: We recommend that this system be ordered only by people seriously considering to implement PASCAL on their computer.

OTHER PASCAL COMPILERS

Mr. Wilhelm Burger at the University of Texas at Austin has made extensive modifications to PASCAL and has written a manual that is available in machine readable form. The University of Colorado has adapted this manual so that it corresponds to the version we release. Mr. Burger's PASCAL incorporates the following extensions. It allows external procedures, segmentation of programs into overlays which can be called sequentially, and partial compilation of declarations and procedures to allow them to be used later. Compiler options can be set on the control card. Several new convenient procedures and functions are defined such as a random number generator, clock, memory dump, and tracing of procedure calls.

A tape of Mr. Burger's PASCAL system can be obtained from the University of Colorado or one can write directly to:

Mr. Wilhelm F. Burger
The University of Texas at Austin
Department of Computer Science
Austin, Texas 78712

An IBM 370 version of PASCAL that runs interpretively at speeds comparable to PL/I for student jobs is available. For further information, write directly to:

Mr. Al Hartmann
Mail Code 286-80
California Institute of Technology
Pasadena, California 91109

MODIFICATIONS TO PASCAL

The University of Colorado has developed and distributes with the PASCAL system the following mods which are of interest to all PASCAL users.

FUNCTION	DATE	PAGES
Adapt POSYS to KRONOS 2.0	30 Jan 73	5
Remove frequence count feature	30 Jan 73	1
Print PMD message on OUTPUT in addition to dayfile	6 Feb 73	2
Add alphabetic tokens equivalent to special symbols	30 Jan 73	1

In addition, a 23 page copy can be made of the following mods from Professor Hellmut Golde at the University of Washington in Seattle, dated November, 1973.

Incorporate cross reference program
Abort after too many compilation errors
Delete frequency count feature
Implement 64 character set for SCOPE 3.4
Adapt POSYS to SCOPE 3.4
Require EOR between program and data on all files
Yield error on attempt to read past EOF
Print PASCAL error message on abort
Accept alphanumeric tokens for special symbols
Add line numbers to compiler listings
Scan only 72 columns of source

OTHER DOCUMENTATION

Copies can be made from the following items in our files.

Author and Title	DATE	PAGES
N. Wirth, ETH, "Program Development by Step-wise Refinement"	Jan 71	24
N. Wirth, ETH, "The Design of a PASCAL Compiler," submitted to Software-Practice and Experience	Jul 71	26
N. Wirth, Stanford University, "On PASCAL, Code Generation, and the CDC 6000 Computer"	Feb 72	40
N. Wirth, ETH, "The Programming Language PASCAL"	Aug 72	61
A. Mickel, University of Minnesota, "PASCAL at the University of Minnesota"	Sep 72	16

Author and Title	Date	Pages
N. Wirth, ETH, "The Programming Language PASCAL (Revised Report)"	Nov 72	53
W. Burger, University of Texas at Austin, "PASCAL Manual," a complete manual for their local version	Jul 73	68
H. Golde, University of Washington, "PASCAL-W, Users Manual," a list of changes to the revised report for their local version	Sep 73	24
K. Jensen and N. Wirth, ETH, "A User Manual for PASCAL," a preliminary copy to be released with the next version of the compiler	Oct 73	87