

PASCAL NEWSLETTER

May, 1974

Number 2

FROM THE EDITOR

The second newsletter marks the release of PASCAL 2 for CDC CYBER 70 and 6000 series computers under the KRONOS or SCOPE operating systems. Interested CDC users may place orders as explained in the section PASCAL 6000 - 3.4. Also, implementations of PASCAL for other machines have become known through recent correspondence. Further information about these implementations can be obtained by writing directly to the contact given with the description of each implementation.

Please note the following important points.

1) Dr. Wirth, the author of PASCAL, is negotiating with a publisher to print a paperback edition of, "A User Manual for PASCAL" by Jensen and Wirth. People who have received a preliminary version of this manual should not make any further copies of it.

2) The University of Colorado has offered in the past a \$10 discount on orders of PASCAL from previous recipients of the package. The discount has been dropped since the new release of PASCAL is more than merely a correction to prior versions. The extra money will be used to defray the cost of this newsletter.

3) A short history of the development of PASCAL is given so that references can be made to the origin of PASCAL compilers on non-CDC computers.

4) A limited number of copies of the first edition of the newsletter are available on request from the editor.

Items of interest or requests for material can be mailed to the editor:

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

or phone: (303) 443-2211, ext. 6934.

HISTORY OF PASCAL

PASCAL is an ALGOL-like programming language with data structure facilities written by Dr. Niklaus Wirth at the Eidgenössische Technische Hochschule (ETH) in Zürich, Switzerland. The original language definition was made in November, 1970, in "The Programming Language PASCAL" published by ETH and later in Acta Informatica 1, 35-63 (1971). The last compiler of this version of PASCAL was released in August, 1972.

In November, 1972, experience gained with the original language revealed certain details of the language that should be changed. This

was done with the publishing of, "The Programming Language Pascal (Revised Report)" in November, "An Axiomatic Definition of the Programming Language Pascal" in December, and the release of an updated compiler dated December, 1972. This compiler implemented all the specifications of the Revised Report except for class variables which conformed to the definition of the original report.

↳ PACKED ARRAYS, SETS & FILES

Later, a preliminary version of the PASCAL-P compiler was developed and released to a limited number of sites. Most of the PASCAL compilers implemented for non-CDC machines are based on this compiler and are identical in the form of PASCAL compiled by the December, 1972 release for CDC machines.

May, 1974, brings the release of a completely new PASCAL compiler called PASCAL 2 for CDC machines. Details of the changes made and a description of the materials available are given in the section, PASCAL 6000 - 3.4.

Finally, the PASCAL-P compiler is being rewritten to bring it in line with standard PASCAL. This portable compiler is expected to be available in July.

PASCAL FOR NON-CDC MACHINES

Several sites have implemented PASCAL compilers for computers other than CDC 6000 series machines. The machines represented are the CII IRIS 80, CII 10070, DEC System 10, IBM 360/370, UNIVAC 1108, and XDS SIGMA 7. For further information on these projects, write the contacts given below.

The CII IRIS 80, CII 10070, and XDS SIGMA 7 share the same machine language. Mr. Didier Thibault and Mr. P. Mancel have taken the December, 1972 PASCAL compiler for the CDC machine and bootstrapped it for the CII IRIS 80. This compiler is currently being tested under control of a monitor written for the SIRIS 7 - SIRIS 8 operating system. It generates relocatable binary object code which can be linked by the general linkage editor. It uses the character set ordering as defined by PASCAL on the CDC computer. It allows procedures to be separately compiled and merged at linkage time. It allows all file management compatible with the SIRIS 7 or SIRIS 8 operating system using the S.G.F. assisted file management system distributed by the CII company. It accepts all features of the PASCAL language except the non-dynamic allocation of files.

The compiler consists of 4500 lines of PASCAL code running under control of a monitor (assembly code). The PASCAL program consists of 23,000 machine instructions, and the monitor 1000 machine instructions. It requires 40,000 thirty-two bit words to compile itself. To make this compiler available on other operating systems, the monitor has to be rewritten. This transposition would be easier if a file management system is available on the target machine.

The bootstrap of this compiler was done using the CII IRIS 80 and a CDC machine in parallel. A simulator was not used. It took two experienced programmers 14 man months to complete.

This compiler is currently being tested prior to its distribution. People interested in receiving documentation can place their names on a mailing list by writing:

S.F.E.R./PASCAL
IRIA 15-02
B.P. 5 78150 Le Chesnay
France

Address other correspondence to:

Mr. D. Thibault
17 rue Mayet
75006 Paris
France

The DEC System 10 implementation of PASCAL was developed at the University of Hamburg, Germany by Professor H. -H. Nagel. Work began in April, 1973 with receipt of the preliminary PASCAL-P compiler from ETH. By November, this version would compile itself. As of April, 1974, everything mentioned in the revised PASCAL report of July, 1973, including I/O formats, is implemented with the exception of procedures and functions as formal arguments, arithmetic procedures (SIN, COS, EXP, LN, ARCTAN, ROUND), and GO TO leaving a procedure body (the GO TO EXIT). Work on these areas is in progress.

In particular, the following goals have been reached. The compiler generates in one pass immediately executable (no loader run) re-entrant code, generating a sharable pure part and a separate LOW-file containing data. A new feature, INITPROCEDURE, has been implemented to initialize global variables. I/O is possible to standard as well as to declared files. Files may be declared only as global variables. A standard file named TTY is introduced to allow communication with the user terminal. An optional file name may be given in READ/WRITE to use the formatting capabilities of these procedures for all files of CHAR. The actual file name may be indicated at execution time by an optional argument to RESET or REWRITE. The printable upper case ASCII character set is used as internal representation of characters. The appropriate procedures and attributes like READLN(f), WRITELN(f), EOLN(f), etc. have all been implemented. DEC line numbers are recognized, stored, and accessible with a special new procedure GETLINENR. Indexed access to PACKED ARRAY has been implemented. Constant indices are evaluated at compile time. To obtain a completely self-sufficient compiler, the re-entrant runtime support is copied out of the compiler into the user's object code file.

A preliminary version of this compiler has already been sent to several sites in the United States (Professor Terry Beyer at the University of Oregon, Dr. Donald I. Good at the University of Southern California, and Dr. Frederick A. Hosch at Louisiana State University). A bug contest has revealed several critical and about twelve minor errors in this version which have been corrected in the meantime. The compiler is currently being used in teaching undergraduate students and in small research projects. For further information write:

Professor H. -H. Nagel
Universität Hamburg
Institut für Informatik
2 Hamburg 13, Schlüterstrasse 66-72
Germany

Two sites are working on PASCAL implementations for IBM 360 and 370 series machines. Mr. Robert S. Deverill and Mr. Alfred C. Hartmann at the California Institute of Technology have a running version of the preliminary PASCAL-P compiler which will be ready soon. Also, Mr. J. M. Wells and Mr. W. Bruce Foulkes at the University of Manitoba are working on a PASCAL compiler for IBM machines which should be ready soon.

Caltech's version of PASCAL is implemented on an IBM 170/158 running under the OS/VS2 operating system. The environment will operate under any version of the OS operating system. About 270 kilobytes of memory are required to compile the compiler. The compiler uses the recursive descent parsing technique to compile PASCAL programs in a single pass. Only two files, a standard input and output file, are implemented in this version. File declarations are unimplemented, as are exit labels, formal parameter procedures and functions, and the standard functions "succ" and "pred."

For further information write:

Mr. Alfred C. Hartmann
California Institute of Technology
Information Science 286-60
Pasadena, California 91109

The PASCAL compiler at the University of Manitoba had its genesis in 1971 at ETH in Zurich, Switzerland. An early paper, "A Pascal Compiler for the IBM 360/370 Computers," was presented last fall at the Third Manitoba Conference on Numerical Mathematics; reprints should be available by now. The first object programs should be running soon and detailed documentation will be available later this year. For further information write:

Professor J. M. Wells
University of Manitoba
Department of Computer Science
Winnipeg, Canada R3T 2N2

The Univac 1108 implementation of PASCAL was done at the Technical University of Norway by Professor Tore Amble and two of his degree students, Mr. Terje Molster and Mr. Vernar Sundvor. The compiler is based on the preliminary version of the PASCAL-P compiler. All code generated is in the form of subroutine calls, so efficiency of compiled programs is not very high. Packed records, packed arrays, files (except for standard files), formal procedures, and formal functions were not implemented. For further information write:

Professor Tore Amble
Computing Centre
Technical University of Norway
Department of SIN TEF
N-7034 Trondheim-NTH
Norway

by N. Wirth

An entirely new compiler for the CDC 6000 series of computers has been under development at ETH Zurich for the last 10 months. As predicted last fall and announced in the first issue of the NEWSLETTER, it is released in May 1974. An important development is the definition of a Standard PASCAL: in the interest of portability of programs, we wish to make a clear distinction between Pascal and Pascal-like languages, as several of these have already been proposed. The new compiler adheres to this Standard, and includes some additional facilities clearly labelled as extensions (3.5--3.6). This Standard also includes the definition of a program representation in terms of the ASCII character set.

The new compiler is designed for use under the CDC SCOPE 3.4 operating system with its 64-character set. The decision to adapt PASCAL to the ASCII set and to character sets without explicit control characters has made necessary some changes in the definition of the language. Of particular importance is the decision to eliminate the gol character. It was felt that this change is in the interest of making Pascal less dependent on particular character sets and actual representations of textfiles.

A summary of the changes and innovations of Pascal 6000-3.4 compared to Pascal 6000-3.2 is presented below in an informal, descriptive style. It is divided into the following parts:

1. Notation (representation, character sets)
2. Differences between Pascal 6000-3.4 and Pascal 6000-3.2
3. New facilities of Pascal 6000-3.4
4. New predefined procedures and functions
5. Control statements (SCOPE 3.4)

The new compiler generates relocatable binary code that can be loaded by the standard loader of the operating system. Before execution, the generated code must be linked by the loader with a set of subroutines for input/output handling. Each program operates on files that are declared as formal parameters in its heading, and are substituted with actual files that can be specified in the EXECUTE statement of the control statement record.

Besides some new features, the major advantage of the new compiler is its improved code which makes compiled programs more efficient and more compact. The price for the expanded capabilities is a larger size of the compiler: for average programs, a field length of 60000 (octal) is needed.

The Pascal 6000-3.4 compiler can be ordered from

Ms. Kathleen Jensen
Institut fur Informatik
Clausiusstr. 55
8006 Zurich
Switzerland

The charge for a minitape, tape handling, postage, and documentation is SFr. 100; if a tape is supplied by the requestor, the charge is SFr. 75. (Please send minitapes only!)

In the USA and Canada, orders must be directed to

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

The charge for a minitape, tape handling, postage (North American continent), and documentation is \$30; if a tape is supplied by the requestor, the charge is \$20.

The system is available in two versions, namely for use with the ASCII character set (CDC-defined collating sequence) or with the CDC scientific character set. When ordering, please specify

PASCAL 6000-3.4 ASCII or
PASCAL 6000-3.4 CDC

Along with the system, the following documentation is provided:

1. A User-Manual. (copying prohibited, as we are currently in contact with a publisher who might possibly be able to provide this manual along with the Revised Report in a moderately priced paperback edition.)
2. A description of the contents of the tape with instructions on how to install the Pascal system.

Note: the system also runs under SCOPE 3.2, 3.3, and 3.4 with the 63-character set, with the only restriction being that the % character cannot be used.

1. Notation

- 1.1 Set union is denoted by + and set intersection by * (instead of \vee and \wedge).

- 1.2 The symbols in the left-hand column may be substituted for those in the right-hand column.

until now	new
~	not
^	and
v	or
/	<>
\	<=
:	>=
{ and }	(* and *)

Note: ^ and v denote Boolean operations and cannot be used for set operations.

The above table defines a unique, context-independent correspondence between those Pascal symbols which are not available in the international standard of ISO (ASCII) and the ASCII character set. Hence there is a standard representation for a Pascal program in the ASCII character set.

2. Differences between PASCAL 6000-3.4 and PASCAL 6000-3.2

2.1 End of lines in textfiles

The control character gol, which marked the end of a line, has been eliminated. Instead, the following textfile operators are able to recognize and generate line endings:

eoln(f) a predicate function, evaluated while reading a textfile, which indicates whether the end of the current line in the textfile f has been reached. Suppose the buffer variable f↑ is positioned at the character x and that the procedure "get(f)" (or "read") is called in order to access the next character. If x had been the last character in the line, then f↑ = ' ' (blank), and the value of eoln(f) = true. The next call of get(f) (or read) accesses the first character of the next line, and eoln(f) = false (provided the next line is not empty).

writeln(f) a standard procedure that terminates the current line when writing the textfile f.

readln(f) a standard procedure that skips to the beginning of the next line; the buffer variable f↑ is equal to the first character of the new line.

The usual program schema for sequential reading of a textfile f follows: (x is a variable of type char; P denotes the processing of the (next) character.)

```

reset(f);
while -eof(f) do
  begin beginline;
    while -eoln(f) do
      begin
        read(f,x); {read from the textfile f and
                    assign to x; see section 3.1}
        P(x)
      end;
    endline: readln(f)
  end
end

```

A line ending is represented by a blank. Notice that the following schema can be used when it is not necessary to recognize line endings--i.e. when no special action is required upon encountering an end of a line:

```

reset(f);
while -eof(f) do
  begin read(f,x); P(x)
  end
end

```

The following abbreviations may be used:

abbreviated form	expanded form
writeln(f,x1, ..., xn)	begin write(f,x1,...,xn); writeln(f) end
readln(f,x1, ..., xn)	begin read(f,x1,...,xn); readln(f) end

Note: The first parameter names the relevant textfile (see section 3.1); when it is not of type text, then the file "input" is assumed by reading and the file "output" by writing. Hence,

writeln	stands for	writeln(output)
and		
readln	stands for	readln(input)

2.2 The program heading

PASCAL 6000-3.4 requires the specification of a program heading. The form is:

```

program p(x1,x2, ..., xn);

```

where p is the name of the program and $x1...xn$ are formal file parameters ($n \geq 1$). $x1...xn$ are available to the program, but also exist outside of the program; hence, they are called external files (as opposed to local files).

The program name has to be declared within the program, but can be used in the control statements:

```
EXECUTE p(f1, ...,fn).
```

where f1...fn are file names, i.e. the actual parameters corresponding to the formal parameters x1...xn.

The following rules hold:

- a) The program heading must contain the formal parameter "output".
- b) As with any other variable, the files denoted by the names x1...xn must be declared as file variables in the main program. The exception occurs with the files "input" and "output" which are automatically predeclared as:

```
var input,output: text;
```
- c) If any actual parameter fi in the EXECUTE statement is left empty, the corresponding formal parameter xi in the program heading is then assumed as the actual "logical file name".
- d) Only one "logical record" will be read from the actual file INPUT--i.e., the next EOR mark appears as EOF in a Pascal program.
- e) If a file xi is to be opened for reading only, then this must be indicated by an asterisk following the file parameter in the program heading. (This is necessary, if actual files are Permanent Files with Read Permission only.)

Note: Rules a)--e) are specifically for the CDC implementation. A consequence of rule c) is that a program with a program heading:

```
program standard(input,output);
```

can be called simply with the control statement:

```
EXECUTE .STANDARD.
```

or even

```
EXECUTE .
```

when the standard SCOPE files INPUT and OUTPUT are intended. Note that the file specifications [IN] and [OUT] of PASCAL 6000-3.2 are eliminated.

2.3 The label declaration part

Every label must be declared. Consequently, the symbol exit is eliminated from the goto statements. If a label L (an

