CWI Syllabi

Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

The CWI is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Research (N.W.O).

CWI Syllabus 22

STATAL: statistical procedures in Algol 60, part 3

R. van der Horst, R.D. Gill (eds.)

Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

ISBN 90 6196 360 5
NUGI-code: 815

5. RANDOM NUMBER GENERATORS

This section contains procedures for generating pseudo random numbers from discrete (section 5.2) and continuous distributions (section 5.3). All procedures use one or more drawings from a uniform distribution on (0,1), which is generated by the procedure ASELECT (section 5.1.1). The procedures have a call-by-name variable U which has to be initialized before the first call of a random number generator (thus, at the beginning of the program). This initial value is the starting value of the whole sequence of (pseudo) random numbers used in the program and determines this sequence completely. Therefore, it is possible to generate the same sequence several times. The pseudo random numbers in each sequence are good approximations of i.i.d. drawings (samples) from the distributions considered.

In order to save execution time, the random number generating procedures do **not** test whether the actual values of the parameters satisfy conditions which are specific for the distribution considered, and thus these procedures do not give any error message. For each distribution the specific conditions are given after the description of the formal parameters. The actual values of the procedures can be tested on these conditions with the procedure TEST RANDOM (section 5.1.2).

References

[1]. A.J. van Es, C. van Putten,
The STATAL random number generator,
STATAL report 1, report SN 8/79,
Mathematical Centre, Amsterdam, 1979.

[2]. C. van Putten, I. van der Tweel,
On generating random variables,
STATAL report 2, report SN 9179,
Mathematical Centre, Amsterdam, 1979.

5.1 ELEMENTARY PROCEDURES FOR RANDOM NUMBER GENERATORS

The procedure ASELECT is a fast generator of the pseudo random numbers which are uniformly distributed on (0,1]; its source text is written in assembler language. All other random number generators are based on this procedure. ASELECT has a call-by-name variable U, which after each call contains a new value. The value is used to generate the next random number. The initial or starting value of U determines the sequence of random numbers. A new sequence starts when the program (i.e. not the procedure ASELECT) assigns a new value to U.

This section also contains the procedure TEST RANDOM, which tests whether the actual values of the parameters in a call of a random number generator satisfy some conditions. These conditions are different for each distribution and are given after the discription of the formal parameters of the procedures considered. It is advised to use TEST RANDOM during the testing of the simulation program and not when the actual simulations are performed.

TITLE: **Aselect**

AUTHORS: A.J. van Es, C. van Putten, D.T. Winter

INSTITUTE: Mathematical Centre

RECEIVED: 800512

BRIEF DESCRIPTION

This procedure generates a (pseudo) random number, uniformly distributed on the interval (0,1].

KEYWORDS

(pseudo) Random number

CALLING SEQUENCE

Heading

```
"REAL" "PROCEDURE" ASELECT (U);
"REAL" U;
"CODE" 41308;
```

Formal parameters

U: <real variable>, the value of U is used to generate a (pseudo) random number. After a call of ASELECT, U contains a new value which is used in the next call of ASELECT unless the program assigns a new value to U.

Condition: $0 \leqslant U < 1$.

DATA AND RESULTS

The generated value is assigned to the procedure indentifier ASELECT.

The call TEST RANDOM (41308, U, 0, 0, 0, 0, 0) may yield the following error message:

Errornumber 1 (if $U < 0$ or $U \geqslant 1$)

PROCEDURES USED

None

LANGUAGE

Compass

METHOD AND PERFORMANCE

A random number is generated as follows:

$U:=((A*U*2^{48}+B) \bmod 2^{48})/2^{48}$

$ASELECT:=U+2^{-48}$;

Where A and B are fixed large integers.

It follows that U satisfies $0 \leqslant U < 1$ and that the value assigned to the procedure indentifier satisfies $0 < ASELECT \leqslant 1$. This enables the user to calculate

LN(ASELECT(U)) or 1 / ASELECT(U).
(see van Es & van Putten, 1979).

REFERENCES

[1]. A.J. van Es, C. van Putten
The STATAL random number generator
STATAL report 1, report SN 8/79
Mathematical Centre, Amsterdam, 1979

[2]. D.E. Knuth
The art of computer programming, vol 2
Addison Wesley, Reading (Mass.), 1969

EXAMPLE OF USE

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
   U:=.1988;
   "FOR" I:=1 "STEP" 1 "UNTIL" 10 "DO"
   OUTPUT(61, "("Z.6D,/")", ASELECT(U))
"END"
```

Output:

```
 .460206
 .241938
 .244897
 .461863
 .336030
 .533801
 .316718
 .469128
 .015569
 .329742
```

SOURCE TEXT

The random number generator ASELECT is a generator of the linear congruential type, specially designed for STATAL and the Cyber 70 with its 48 bits integer representation. The generator is programmed in COMPASS (the CDC assembler language) for two reasons.

Firstly it is inherent to simulation studies that a generator is called upon a vast number of times, hence it is necessary that the calculations be done quickly.

Secondly since the linear congruential method involves rounding of large integers, usually machine dependent features are used to increase the speed of the calculations.

Because of the very machine dependent nature of this routine, no source text is given.
For the algorithm, see METHOD AND PERFORMANCE and A.J.van Es, C.van Putten (1979), The Statal random number generator. (REFERENCE[1])

TITLE: **Test Random**

AUTHOR: E. Opperdoes

INSTITUTE: Mathematical Centre

RECEIVED: 810101

BRIEF DESCRIPTION

The procedure tests the actual values of the parameters on the conditions specified in the section 5.1.1 and 5.2.1.1-5.3.11.1 (all random number generators). Such tests are not performed in the generators themselves in order to save execution time.

KEYWORDS

Parameter testing of random generator

CALLING SEQUENCE

Heading

```
"PROCEDURE" TEST RANDOM (CODENR, P1, P2, P3, P4, P5, P6);
"VALUE" CODENR, P1, P2, P3, P4, P5, P6;
"INTEGER" CODENR;
"REAL" P1, P2, P3, P4, P5, P6;
"CODE" 41399;
```

Formal parameters

CODENR: <integer arithmetic expression>, The code number of the procedure involved;

P1,...,P6: <arithmetic expression>, the parameters of the procedure, supplemented with dummy parameters.

DATA AND RESULTS

The following error message may appear:

Errornumber 1	(if CODENR < 41300 or CODENR > 41333)

In case of errors in the other parameters the procedure gives a message for each error. All errors cause a termination of the main program. For parameter error messages one is referred to the section in question.

PROCEDURES USED

EXIT	STATAL 11010
ALGMESS	STATAL 11017
STATAL3 ERROR	STATAL 40100

Language
Algol 60

Example of use

Program:

```
"BEGIN"
    TEST RANDOM(41300, 2, 2, 0, 0, 0, 0);
"END"
```

Output:

```
TEST RANDOM DISCOVERED THE FOLLOWING ERROR(S) IN THE PARAMETERS OF PROCEDURE SUCCES
===================================================================================

                  NUMBER OF THE ERROR:  1
PARAMETER VALUE THAT CAUSED THE ERROR:  +2.00000000000000"+000

                  NUMBER OF THE ERROR:  2
PARAMETER VALUE THAT CAUSED THE ERROR:  +2.00000000000000"+000
```

Source text

```
"CODE" 41399;
"PROCEDURE" TEST RANDOM(CODENR, P1, P2, P3, P4, P5, P6);
"VALUE" CODENR, P1, P2, P3, P4, P5, P6;
"INTEGER" CODENR; "REAL" P1, P2, P3, P4, P5, P6;
"BEGIN" "INTEGER" INDEX; "BOOLEAN" ERRORS;
    "SWITCH" PROC:=
        SUCCES, INT, UNIF, NORM, EXPON, POIS, PERM,
        SAM, ASELECT, MULTINORM, SORSAM, SORSAMREP,
        SAMREP, GAMMA, BETA, CHISQ, STAND HYPERG, HYPERG,
        CAUCHY, WEIBULL, LAPLACE, GUMBEL, LOGISTIC,
        GEOMETRIC, BIN, NEGBIN, POIS TAB,
        POIS TAB SAM, POIS HISTO, POIS SORSAM,
        BINALIAS, BINALIAS SORSAM, BIN HISTO, BINORM;


    "PROCEDURE" ERROR(CP, EN, WV); "VALUE" EN, WV;
    "STRING" CP; "INTEGER" EN; "REAL" WV;
    "BEGIN" "IF" "NOT" ERRORS "THEN"
        "BEGIN" "PROCEDURE" LAYOUT;
            FORMAT(
            "(""("TEST RANDOM DISCOVERED THE FOLLOWING ")",
            "("ERROR(S) IN THE PARAMETERS OF PROCEDURE ")",
            N,/,X("("=")"),//")", CHLENGTH(CP) + 77);
            "PROCEDURE" LIST(ITEM); "PROCEDURE" ITEM;
            ITEM(CP);
            "INTEGER" P, PP;
            SYSPARAM(61, 1, P); SYSPARAM(61, 3, PP);
            "IF" P > 0 "OR" PP > 0 "THEN"
                OUTPUT(61, "("*")");
            OUTLIST(61, LAYOUT, LIST); ERRORS:= "TRUE";
        "END";
        OUTPUT(61, "("
        "("                     NUMBER OF THE ERROR:  ")",D,/,
        "("PARAMETER VALUE THAT CAUSED THE ERROR: ")"
        ,N,//")", EN, WV);
    "END" ERROR;

    "PROCEDURE" TEST(CP, EN, PAR);
    "STRING" CP; "INTEGER" EN; "REAL" PAR;
    "IF" PAR < 0 "OR" PAR >= 1 "THEN" ERROR(CP, EN, PAR);

    "PROCEDURE" NATURAL(CP, EN, PAR);
    "STRING" CP; "INTEGER" EN; "REAL" PAR;
    "IF" PAR > ENTIER(PAR) "OR" PAR < 1 "THEN"
        ERROR(CP, EN, PAR);

    "PROCEDURE" NATURAL0(CP, EN, PAR);
    "STRING" CP; "INTEGER" EN; "REAL" PAR;
    "IF" PAR > ENTIER(PAR) "OR" PAR < 0 "THEN"
        ERROR(CP, EN, PAR);

    "PROCEDURE" POS(CP, EN, PAR);
    "STRING" CP; "INTEGER" EN; "REAL" PAR;
    "IF" PAR <= 0 "THEN" ERROR(CP, EN, PAR);

    "PROCEDURE" INTEGER(CP, EN, PAR);
    "STRING" CP; "INTEGER" EN; "REAL" PAR;
    "IF" PAR > ENTIER(PAR) "THEN" ERROR(CP, EN, PAR);

    "PROCEDURE" INT GE(CP, EN, PAR, CRIT);
    "STRING" CP; "INTEGER" EN; "REAL" PAR, CRIT;
    "IF" PAR > ENTIER(PAR) "OR" PAR < CRIT "THEN"
        ERROR(CP, EN, PAR);

    "PROCEDURE" NATURAL LE(CP, EN, PAR, CRIT);
    "STRING" CP; "INTEGER" EN; "REAL" PAR, CRIT;
    "IF" PAR > ENTIER(PAR) "OR" PAR < 1 "OR" PAR > CRIT
    "THEN" ERROR(CP, EN, PAR);

    INDEX:= CODENR - 41299; ERRORS:= "FALSE";
    "IF" INDEX < 1 "OR" INDEX > 34
    "THEN" STATAL3 ERROR("("TEST RANDOM")", 1, CODENR);
    "GOTO" PROC[INDEX];

SUCCES:
    "IF" P1 < 0 "OR" P1 > 1 "THEN"
        ERROR("("SUCCES")", 1, P1);
    TEST("("SUCCES")", 2, P2);
    "GOTO" FINISH;

INT:
    INTEGER("("RANDOM INT")", 1, P1);
    INT GE("("RANDOM INT")", 2, P2, P1);
    TEST("("RANDOM INT")", 3, P3);
    "GOTO" FINISH;
```

```
UNIF:
    "IF" P1 >= P2 "THEN" ERROR("("RANDOM UNIF")", 2, P2);
    TEST("("RANDOM UNIF")", 3, P3);
    "GOTO" FINISH;

NORM:
    POS("("RANDOM NORM")", 2, P2);
    TEST("("RANDOM NORM")", 3, P3);
    "GOTO" FINISH;

EXPON:
    POS("("RANDOM EXP")", 1, P1);
    TEST("("RANDOM EXP")", 2, P2);
    "GOTO" FINISH;

POIS:
    POS("("RANDOM POIS")", 1, P1);
    "IF" P1 > 700 "THEN" ERROR("("RANDOM POIS")", 1, P1);
    TEST("("RANDOM POIS")", 2, P2);
    "GOTO" FINISH;

PERM:
    NATURAL("("RANDOM PERM")", 2, P1);
    TEST("("RANDOM PERM")", 3, P2);
    "GOTO" FINISH;

SAM:
    NATURAL("("RANDOM SAM")", 1, P1);
    NATURAL LE("("RANDOM SAM")", 2, P2, P1);
    TEST("("RANDOM SAM")", 4, P3);
    "GOTO" FINISH;

ASELECT:
    TEST("("ASELECT")", 1, P1);
    "GOTO" FINISH;

MULTINORM:
    INTEGER("("RANDOM MULTINORM")", 2, P1);
    INT GE("("RANDOM MULTINORM")", 3, P2, P1);
    NATURAL("("RANDOM MULTINORM")", 4, P3);
    TEST("("RANDOM MULTINORM")", 8, P4);
    "GOTO" FINISH;

SORSAM:
    NATURAL("("RANDOM SORSAM")", 1, P1);
    NATURAL LE("("RANDOM SORSAM")", 2, P2, P1);
    TEST("("RANDOM SORSAM")", 4, P3);
    "GOTO" FINISH;

SORSAMREP:
    NATURAL("("RANDOM SORSAMREP")", 1, P1);
    NATURAL("("RANDOM SORSAMREP")", 2, P2);
    TEST("("RANDOM SORSAMREP")", 4, P3);
    "GOTO" FINISH;
```

```
SAMREP:
    NATURAL("("RANDOM SAMREP")", 1, P1);
    NATURAL("("RANDOM SAMREP")", 2, P2);
    TEST("("RANDOM SAMREP")", 4, P3);
    "GOTO" FINISH;

GAMMA:
    POS("("RANDOM GAMMA")", 1, P1);
    POS("("RANDOM GAMMA")", 2, P2);
    TEST("("RANDOM GAMMA")", 3, P3);
    "GOTO" FINISH;

BETA:
    POS("("RANDOM BETA")", 1, P1);
    POS("("RANDOM BETA")", 2, P2);
    TEST("("RANDOM BETA")", 3, P3);
    "GOTO" FINISH;

CHISQ:
    NATURAL("("RANDOM CHISQ")", 1, P1);
    TEST("("RANDOM CHISQ")", 2, P2);
    "GOTO" FINISH;

STAND HYPERG:
    NATURAL("("RANDOM STHYPERG")", 1, P1);
    NATURAL("("RANDOM STHYPERG")", 2, P2);
    NATURAL("("RANDOM STHYPERG")", 3, P3);
    TEST("("RANDOM STHYPERG")", 4, P4);
    "IF" P2 < P1 "OR" 2 * P2 > P3
    "THEN" ERROR("("RANDOM STHYPERG")", 5, P2);
    "GOTO" FINISH;

HYPERG:
    NATURAL("("RANDOM HYPERG")", 1, P1);
    NATURAL("("RANDOM HYPERG")", 2, P2);
    NATURAL("("RANDOM HYPERG")", 3, P3);
    TEST("("RANDOM HYPERG")", 4, P4);
    "IF" P1 > P3 "THEN" ERROR("("RANDOM HYPERG")", 1, P1);
    "IF" P2 > P3 "THEN" ERROR("("RANDOM HYPERG")", 2, P2);
    "GOTO" FINISH;

CAUCHY:
    POS("("RANDOM CAUCHY")", 2, P2);
    TEST("("RANDOM CAUCHY")", 3, P3);
    "GOTO" FINISH;

WEIBULL:
    POS("("RANDOM WEIBULL")", 2, P2);
    POS("("RANDOM WEIBULL")", 3, P3);
    TEST("("RANDOM WEIBULL")", 4, P4);
    "GOTO" FINISH;

LAPLACE:
    POS("("RANDOM LAPLACE")", 2, P2);
    TEST("("RANDOM LAPLACE")", 3, P3);
    "GOTO" FINISH;

GUMBEL:
    POS("("RANDOM GUMBEL")", 2, P2);
    TEST("("RANDOM GUMBEL")", 3, P3);
    "GOTO" FINISH;

LOGISTIC:
    POS("("RANDOM LOGISTIC")", 2, P2);
    TEST("("RANDOM LOGISTIC")", 3, P3);
    "GOTO" FINISH;

GEOMETRIC:
    "IF" P1 <= 0 "OR" P1 > 1
    "THEN" ERROR("("RANDOM GEOMETRIC")", 1, P1);
    TEST("("RANDOM GEOMETRIC")", 2, P2);
    "GOTO" FINISH;

BIN:
    NATURAL0("("RANDOM BIN")", 1, P1);
    "IF" P2 < 0 "OR" P2 > 1
    "THEN" ERROR("("RANDOM BIN")", 2, P2);
    TEST("("RANDOM BIN")", 3, P3);
    "GOTO" FINISH;

NEGBIN:
    NATURAL("("RANDOM NEGBIN")", 1, P1);
    "IF" P2 <= 0 "OR" P2 > 1
    "THEN" ERROR("("RANDOM NEGBIN")", 2, P2);
    TEST("("RANDOM NEGBIN")", 3, P3);
    "GOTO" FINISH;

POIS TAB:
    POS("("RANDOM POIS TAB")", 1, P1);
    "IF" P1 >= 256 "THEN"
        ERROR("("RANDOM POIS TAB")", 1, P1);
    TEST("("RANDOM POIS TAB")", 2, P2);
    "GOTO" FINISH;

POIS TAB SAM:
    POS("("RANDOM POISTABSAM")", 1, P1);
    "IF" P1 >= 256 "THEN"
        ERROR("("RANDOM POISTABSAM")", 1, P1);
    NATURAL("("RANDOM POISTABSAM")", 2, P2);
    TEST("("RANDOM POISTABSAM")", 4, P3);
    "GOTO" FINISH;

POIS HISTO:
    NATURAL0("("RANDOM POIS HISTO")", 2, P1);
    NATURAL("("RANDOM POIS HISTO")", 3, P2);
    NATURAL("("RANDOM POIS HISTO")", 4, P3);
    POS("("RANDOM POIS HISTO")", 5, P4);
    "IF" P4>700 "THEN"
```

```
        ERROR("("RANDOM POIS HISTO")", 5, P4);
    TEST("("RANDOM POIS HISTO")", 6, P5);
    "IF" P2 < P1
    "THEN" ERROR("("RANDOM POIS HISTO")", 3, P2);
    "GOTO" FINISH;

POIS SORSAM:
    INT GE("("RANDOM POISSORSAM")", 3, P2, P1);
    POS("("RANDOM POISSORSAM")", 4, P3);
    "IF" P3 > 700 "THEN"
        ERROR("("RANDOM POISSORSAM")", 4, P3);
    TEST("("RANDOM POISSORSAM")", 5, P4);
    "IF" P2 < P1 "THEN"
        ERROR("("RANDOM POISSORSAM")", 2, P2);
    "GOTO" FINISH;

BIN ALIAS:
    NATURALO("("RANDOM BIN ALIAS")", 1, P1);
    "IF" P2 <= 0 "OR" P2 >= 1
    "THEN" ERROR("("RANDOM BIN ALIAS")", 2, P2);
    NATURAL("("RANDOM BIN ALIAS")", 3, P3);
    TEST("("RANDOM BIN ALIAS")", 5, P4);
    "GOTO" FINISH;

BIN ALIAS SORSAM:
    NATURALO("("RANDOM BIN ALIAS SORSAM")", 1, P1);
    "IF" P2 <= 0 "OR" P2 >= 1
    "THEN" ERROR("("RANDOM BIN ALIAS SORSAM")", 2, P2);
    NATURAL("("RANDOM BIN ALIAS SORSAM")", 3, P3);
    TEST("("RANDOM BIN ALIAS SORSAM")", 5, P4);
    "GOTO" FINISH;

BIN HISTO:
    NATURALO("("RANDOM BIN HISTO")", 1, P1);
    "IF" P2 <= 0 "OR" P2 >= 1
    "THEN" ERROR("("RANDOM BIN HISTO")", 2, P2);
    NATURAL("("RANDOM BIN HISTO")", 4, P3);
    TEST("("RANDOM BIN HISTO")", 5, P4);
    "GOTO" FINISH;

BINORM:
    POS("("RANDOM BINORM")", 5, P3);
    POS("("RANDOM BINORM")", 6, P4);
    "IF" ABS(P5) > 1 "THEN"
        ERROR("("RANDOM BINORM")", 7, P5);
    TEST("("RANDOM BINORM")", 8, P6);

FINISH:
    "IF" ERRORS "THEN"
    "BEGIN" ALGMESS(
        "("** ERROR(S) DISCOVERED BY TEST RANDOM **")");
        STOP;
    "END";
"END" TEST RANDOM;
         "EOP"
```

5.2 RANDOM NUMBERS FROM DISCRETE DISTRIBUTIONS

This section contains procedures for generating one single drawing, an (unsorted) sample of size K, or a sorted sample of size K (histogram) from discrete distributions. For some distributions (discrete uniform, binomial and Poisson) there exist several procedures for slightly different or similar purposes. The differences between these procedures are explained below.

Discrete uniform distribution (section 5.2.1). The differences between the six procedures in this subsection are actually quite clear. RANDOM INT draws one single random number from an uniform distribution on the set of integers {A, A+1,...,B}. RANDOM SAM generates a random sample of size K without replacement from the set {1,...,N}, and RANDOM PERM generates a random permutation of the integers 1, 2,...,N. This procedure is equivalent to RANDOM SAM with K=N. The procedure RANDOM SORSAM draws a sorted random sample of size K without replacement from {1,...,N}. Furthermore, RANDOM SAMREP and RANDOM SORSAMREP generate an unsorted and a sorted sample of size K with replacement from {1,...,N}.

Binomial distribution (section 5.2.3). The procedure RANDOM BIN generates one single drawing from a binomial distribution, whereas RANDOM BINALIAS draws an (unsorted) sample of size K. These procedures are based on different methods; in case of one drawing RANDOM BIN is faster than RANDOM BINALIAS with K=1. RANDOM BINALSORS and RANDOM BINHISTO both generate a sorted sample of size K from a binomial distribution and deliver it in a histogram. The latter procedure is generally faster.

Poisson distribution (section 5.2.7). Both procedures RANDOM POIS and RANDOM POISTAB generate one single drawing from a Poisson distribution with expectation MU, according to different methods (see van Putten & van der Tweel, 1979). RANDOM POIS is generally faster, except when MU is a power of 2. The procedure RANDOM POISTAB generates an (unordered) sample of size K. Furthermore, the procedures RANDOM POISHISTO and RANDOM POISSORSAM both generate ordered random samples of size K. The latter procedure delivers the sample in an array of length K, whereas the former delivers it in a histogram on the integers LB, LB+1,...,UB. (Observations smaller than LB or larger than UB are counted in the classes LB and UB respectively). RANDOM POISHISTO is always faster than RANDOM POISSORSAM but has the disadvantage that it only considers observations in a restricted area (the smaller the area LB,...,UB, the more efficient RANDOM POISHISTO).

REFERENCE

[1]. C. van Putten, I van der Tweel,
On generating random variables,
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam. 1979.

Title: **Random Int**

Author: J.G. Bethlehem

Institute: Mathematical Centre

Received: 750219

Brief description

The procedure generates a random number, uniformly distributed on the set of integers {A, A+1,...,B} (see general part section 5.2.).

Keywords

Random integer

Calling sequence

Heading

```
"INTEGER" "PROCEDURE" RANDOM INT(A, B, U);
"VALUE" A, B;
"INTEGER" A, B;
"REAL" U;
"CODE" 41301;
```

Formal parameters

A: <integer arithmetic expression>, lower bound of the set of integers;

B: <integer arithmetic expression>, upper bound of the set of integers;

U: <real variable>, value used to generate a random number. After a call of RANDOM INT, U contains a new value which is used in the next call of RANDOM INT unless the program assigns a new value to U.

Conditions: A and B integers, $A \leqslant B$, $0 \leqslant U < 1$.

Data and results

The generated value is assigned to the procedure indentifier RANDOM INT.

The call TEST RANDOM (41301, A, B, U, 0, 0, 0) may yield the following error messages:

Errornumber 1	(if A is not an integer)
Errornumber 2	(if B is not an integer $\geqslant$ A)
Errornumber 3	(if U < 0 or U $\geqslant$ 1)

Procedures used

ASELECT	STATAL 41308

Language

Algol 60

METHOD AND PERFORMANCE

The generated value is equal to ENTIER (D* (B−A+1)+A), where D is a random number which is uniformly distributed on the interval [0,1).
(see van Putten & van der Tweel, 1979).

REFERENCE

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979.

EXAMPLE OF USE

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    U:= .1986;
    "FOR" I:=1 "STEP" 1 "UNTIL" 10 "DO"
    OUTPUT(61, "("ZD,/")", RANDOM INT(3, 13, U))
"END"
```

Output:

```
12
 3
10
 7
 6
10
 9
 7
 4
 6
```

SOURCE TEXT

```
"CODE" 41301;
"INTEGER" "PROCEDURE" RANDOM INT(A, B, U); "VALUE" A, B;
    "INTEGER" A, B; "REAL" U;
"BEGIN"
    ASELECT(U);
    RANDOM INT:= ENTIER( U * (B - A + 1) + A )
"END" RANDOM INT;
        "EOP"
```

TITLE: **Random Sam**

AUTHOR: J.G. Bethlehem

INSTITUTE: Mathematical Centre

RECEIVED: 760901

BRIEF DESCRIPTION

The procedure generates a random sample of size K without replacement from the set of integers {1,2,...,N}. (see general part of section 5.2).

KEYWORDS

Random sample without replacement

CALLING SEQUENCE

Heading

```
"PROCEDURE" RANDOM SAM (N, K, SAMPLE, U);
"VALUE" N, K;
"INTEGER" N, K;
"REAL" U;
"INTEGER" "ARRAY" SAMPLE;
"CODE" 41307;
```

Formal parameters

N: <integer arithmetic expression>, upper bound of the set of integers from which the random sample is taken;

K: <integer arithmetic expression>, length of the vector SAMPLE [1:K] and size of the sample;

SAMPLE: <integer array identifier>, output parameter, integer array of dimension [1:K], which at exit contains the sample;

U: <real variable>, value used to generate a random number. After a call of RANDOM SAM, U contains a new value which is used in the next call of RANDOM SAM unless the program assigns a new value to U.

Conditions: N and K integers > 0, K ⩽ N and 0 ⩽ U < 1.

DATA AND RESULTS

After a procedure call, SAMPLE[1],...,SAMPLE[K] contain a random sample without replacement from the set of integers {1, 2,...,N}. The call TEST RANDOM(41307,N, K, U, 0, 0, 0) may yield the following error messages:

Errornumber 1	(if N is not an integer > 0)
Errornumber 2	(if K is not an integer > 0 and ⩽ N)
Errornumber 4	(if U < 0 or U ⩾1)

PROCEDURES USED

ASELECT STATAL 41308

LANGUAGE

Algol 60

METHOD AND PERFORMANCE

See van Putten & van der Tweel, (1979).

REFERENCE

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

EXAMPLE OF USE

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    "INTEGER" "ARRAY" A[1:3];
    U:= .1986;
    "FOR" I:=1 "STEP" 1 "UNTIL" 10 "DO"
    "BEGIN" RANDOM SAM(5, 3, A, U);
        OUTPUT(61, "("3(D2B),/")", A)
    "END"
"END"
```

Output:

```
5  1  2
2  5  3
3  2  1
2  1  5
3  5  2
5  1  2
1  4  3
3  5  1
2  3  1
2  3  4
```

Source text

```
"CODE" 41307;
"PROCEDURE" RANDOM SAM(N, K, SAMPLE, U); "VALUE" N, K;
"INTEGER" N, K; "REAL" U; "INTEGER" "ARRAY" SAMPLE;
"BEGIN" "INTEGER" I, KK; "INTEGER" "ARRAY" B[1 : N];
    "BEGIN"
        "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO" B[I]:= I;
        "FOR" I:= 1 "STEP" 1 "UNTIL" K "DO"
        "BEGIN" ASELECT(U);
            KK:= ENTIER(U * N + 1); SAMPLE[I]:= B[KK];
            B[KK]:= B[N]; N:= N - 1;
        "END";
    "END"
"END" RANDOM SAM;
         "EOP"
```

TITLE: **Random Perm**

AUTHOR: J.G. Bethlehem

INSTITUTE: Mathematical Centre

RECEIVED: 760901

BRIEF DESCRIPTION

This procedure generates a random permutation of the integers {1,2,...,N}. (see general part of section 5.2).

KEYWORDS

Random permutation

CALLING SEQUENCE

Heading

```
"PROCEDURE" RANDOM PERM (PERM, N, U);
"VALUE" N;
"INTEGER" N;
"REAL" U;
"INTEGER" "ARRAY" PERM;
"CODE" 41306;
```

Formal parameters

PERM <integer array identifier>, output parameter, integer array of dimension [1:N] which at exit contains the permutation;

N: <integer arithmetic expression>, upper bound of the set of integers 1, 2,...,N and length of the vector PERM;

U: <real variable>, value used to generate a random number. After a call of RANDOM PERM, U contains a new value which is used in the next call of RANDOM PERM unless the program assigns a new value to U.

Conditions: N integer >0 and 0≤U<1.

DATA AND RESULTS

After a call of the procedure, PERM[1],...,PERM[N] contain a random permutation of 1,2,...,N.

The call TEST RANDOM(41306, N, U, 0, 0, 0, 0) may yield the following error messages:

Errornumber 2	(if N is not an integer >0)
Errornumber 3	(if U <0 or U≥1)

PROCEDURES USED

ASELECT STATAL 41308

LANGUAGE

Algol 60

METHOD AND PERFORMANCE

A sample without replacement of size N is taken from the set of integers {1,2,...,N}, see van Putten & van der Tweel (1979).

REFERENCE

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

EXAMPLE OF USE

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    "INTEGER" "ARRAY" A[1:3];
    U:= .1986;
    "FOR" I:=1 "STEP" 1 "UNTIL" 10 "DO"
    "BEGIN" RANDOM PERM(A, 3, U);
        OUTPUT(61, "("3(D2B),/")", A)
    "END"
"END"
```

Output:

```
2  1  3
3  1  2
3  2  1
3  1  2
2  3  1
2  3  1
1  3  2
1  3  2
2  3  1
3  2  1
```

SOURCE TEXT

```
"CODE" 41306;
"PROCEDURE" RANDOM PERM(A, N, U); "VALUE" N;
"INTEGER" N; "REAL" U; "INTEGER" "ARRAY" A;
"BEGIN" "INTEGER" I, R, X;

    "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO" A[I]:= I;
    "FOR" I:= N "STEP" -1 "UNTIL" 2 "DO"
    "BEGIN"
        ASELECT(U); R:= ENTIER( U * I + 1);
        "IF" R < I "THEN"
        "BEGIN" X:= A[I]; A[I]:= A[R]; A[R]:= X "END"
    "END"
"END" RANDOM PERM;
         "EOP"
```

TITLE: **Random Sorsam**

AUTHOR: R. Kaas

INSTITUTE: Mathematical Centre

RECEIVED: 770215

BRIEF DESCRIPTION
The procedure generates a sorted random sample without replacement from the set of integers {1,2,...,N}, (see general part of section 5.2).

KEYWORDS
Sorted random sample without replacement

CALLING SEQUENCE
Heading

```
"PROCEDURE" RANDOM SORSAM (N, K, SAMPLE, U);
"VALUE" N, K;
"INTEGER" N, K;
"INTEGER" "ARRAY" SAMPLE;
"REAL" U;
"CODE" 41310;
```

Formal parameters

N: <integer arithmetic expression>, upper bound of the set of integers from which the random sample is taken;

K: <integer arithmetic expression>, length of the vector SAMPLE[1:K] and size of the sample;

SAMPLE: <integer array indentifier>, output parameter, integer array of dimension [1:K] which at exit contains the sample;

U: <real variable>, value used to generate a random number. After a call of RANDOM SORSAM, U contains a new value which is used in the next call of RANDOM SORSAM unless the program assigns a new value to U.

Conditions: N and K integers >0, K ≤ N and 0≤U<1.

DATA AND RESULTS
After a procedure call, SAMPLE[1] ≤....≤ SAMPLE[K] contain a sorted random sample without replacement from the set of integers {1,2,...,N}.
The call TEST RANDOM(41310, N, K, U, 0, 0, 0) may yield the following error messages:

Errornumber 1	(if N is not an integer >0)
Errornumber 2	(if K is not an integer >0 and ≤ N)
Errornumber 4	(if U <0 or U≥1)

Procedures used

ASELECT	STATAL 41308

Language

Algol 60

Method and performance

For the algorithm used see Kaas (1977), program II. 6. 1., and v. Putten & v.d. Tweel (1979).

The procedure uses no extra central memory. The running time is proportional to N.

References

[1]. R. Kaas
The complexity of the drawing an ordered random sample
Report SW 51/77
Mathematical Centre, Amsterdam, 1977

[2]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

Example of use

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    "INTEGER" "ARRAY" A[1:3];
    U:= .1986;
    "FOR" I:= 1 "STEP" 1 "UNTIL" 10 "DO"
    "BEGIN" RANDOM SORSAM(5, 3, A, U);
        OUTPUT(61, "("3(D2B),/")", A)
    "END"
"END"
```

Output:

```
2  3  4
2  3  5
3  4  5
2  4  5
2  3  4
1  2  3
2  3  5
1  2  3
1  2  3
1  2  3
```

SOURCE TEXT

```
"CODE" 41310;
"PROCEDURE" RANDOM SORSAM(N, K, SAMPLE, U);
"VALUE" N, K;
"INTEGER" N, K; "INTEGER" "ARRAY" SAMPLE; "REAL" U;
"BEGIN"
    "FOR" N:= N "WHILE" N >= 1 "AND" K >= 1 "DO"
    "BEGIN" "IF" ASELECT(U) <= K / N "THEN"
        "BEGIN" SAMPLE[K]:= N; K:= K - 1 "END";
        N:= N - 1
     "END"
"END" RANDOM SORSAM;
         "EOP"
```

TITLE: **Random Samrep**

AUTHOR: R. Kaas

INSTITUTE: Mathematical Centre

RECEIVED: 770215

BRIEF DESCRIPTION

The procedure generates a random sample with replacement from the set of integers {1,2,...,N}. (see general part of section 5.2).

KEYWORDS

Random sample with replacement

CALLING SEQUENCE

Heading

```
"PROCEDURE" RANDOM SAMREP (N, K, SAMPLE, U);
"VALUE N, K;
"INTEGER" N, K;
"INTEGER" "ARRAY" SAMPLE;
"REAL" U;
"CODE" 41312;
```

Formal parameters

N: <integer arithmetic expression>, upper bound of the set of integers from which the random sample is taken;

K: <integer arithmetic expression>, length of the vector SAMPLE[1:K] and size of the sample;

SAMPLE: <integer array identifier>, output parameter, integer array of dimension [1:K], which at exit contains the sample;

U: <real variable>, value used to generate a random number. After a call of RANDOM SAMREP, U contains a new value which is used in the next call of RANDOM SAMREP unless the program assigns a new value to U.

Conditions: N and K integers >0 and 0≤ U <1.

DATA AND RESULTS

After a procedure call, SAMPLE[1],...,SAMPLE[K] contain a random sample with replacement from the integers 1,2,...,N.

The call TEST RANDOM (41312, N, K, U, 0, 0, 0) may yield the following error messages:

Errornumber 1	(if N is not an integer >0)
Errornumber 2	(if K is not an integer >0)
Errornumber 4	(if U <0 or U ≥ 1)

PROCEDURES USED

ASELECT	STATAL 41308

LANGUAGE

Algol 60

METHOD AND PERFORMANCE

See v. Putten & v.d. Tweel (1979).
The procedure uses no extra central memory. The running time is proportional to K.

REFERENCE

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

EXAMPLE OF USE

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    "INTEGER" "ARRAY" A[1:3];
    U:= .1986;
    "FOR" I:= 1 "STEP" 1 "UNTIL" 10 "DO"
    "BEGIN" RANDOM SAMREP(5, 3, A, U);
        OUTPUT(61, "("3(D2B), /")", A)
    "END"
"END"
```

Output:

```
5  1  4
2  2  4
3  3  1
2  1  3
3  4  3
5  1  3
1  5  5
3  4  1
2  3  1
2  4  5
```

SOURCE TEXT

```
"CODE" 41312;
"PROCEDURE" RANDOM SAMREP(N, K, SAMPLE, U);
"VALUE" N, K;
"INTEGER" N, K; "INTEGER" "ARRAY" SAMPLE; "REAL" U;
"BEGIN" "INTEGER" I;

    "FOR" I:= 1 "STEP" 1 "UNTIL" K "DO"
    "BEGIN" ASELECT(U); SAMPLE[I]:= ENTIER(N * U + 1)
    "END"
"END" RANDOM SAMREP;
         "EOP"
```

TITLE: **Random Sorsamrep**

AUTHOR: R. Kaas

INSTITUTE: Mathematical Centre

RECEIVED: 770215

BRIEF DESCRIPTION

The procedure generates a sorted random sample with replacement from the set of integers {1,2,...,N},
(see general part of section 5.2.).

KEYWORDS

Sorted random sample with replacement

CALLING SEQUENCE

Heading

```
"PROCEDURE" RANDOM SORSAMREP (N, K, SAMPLE, U);
"VALUE" N, K;
"INTEGER" N, K;
"INTEGER" "ARRAY" SAMPLE;
"REAL" U;
"CODE" 41311;
```

Formal parameters

N: <integer arithmetic expression>, upper bound of the set of integers from which the random sample is taken;

K: <integer arithmetic expression>, length of the vector SAMPLE[1:K] and size of the sample;

SAMPLE <integer array identifier>, output parameter, integer array of dimension [1:K], which at exit contains the sample;

U: <real variabel>, value used to generate a random number. After a call of RANDOM SORSAMREP, U contains a new value which is used in the next call of RANDOM SORSAMREP unless the program assigns a new value to U.

Conditions: N and K integers > 0, and 0≤U < 1.

DATA AND RESULTS

After a procedure call, SAMPLE[1] ≤,...,≤ SAMPLE[K] contain a sorted random sample with replacement from the set of integers {1,2,...,N}.
The call TEST RANDOM (41311, N, K, U, 0, 0, 0) may yield the following error messages:

Errornumber 1	(if N is not an integer > 0)
Errornumber 2	(if K is not an integer > 0)
Errornumber 4	(if U < 0 or U ≥ 1)

Procedures used

ASELECT STATAL 41308

Language

Algol 60

Method and performance

For the algorithm used see Kaas (1979), program II. 6. 4., and van Putten & van der Tweel (1979).
The procedure uses no extra central memory. The running time is proportional to N+K.

References

[1] R. Kaas
The complexity of drawing an ordered random sample
Report SW 51/77
Mathematical Centre, Amsterdam, 1977

[2] C. van Putten, I. van der Tweel
On generating random variabels
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

Example of use

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    "INTEGER" "ARRAY" A[1:3];
    U:= .1986;
    "FOR" I:= 1 "STEP" 1 "UNTIL" 10 "DO"
    "BEGIN" RANDOM SORSAMREP(5, 3, A, U);
        OUTPUT(61, "("3(D2B), /")", A)
    "END"
"END"
```

Output:

```
1  1  5
1  3  5
2  4  5
1  2  3
2  3  4
1  4  4
1  1  4
3  5  5
1  1  5
1  3  4
```

SOURCE TEXT

```
"CODE" 41311;
"PROCEDURE" RANDOM SORSAMREP(N, K, SAMPLE, U); "VALUE" N, K;
"INTEGER" N, K; "INTEGER" "ARRAY" SAMPLE; "REAL" U;
"BEGIN"
    "IF" K < N "THEN"
    "BEGIN" "REAL" Y, S; "INTEGER" G, I;
        S:= LN(N); G:= N - 1; I:= K + 1;
        "FOR" I:= I - 1 "WHILE" I > 0 "AND" G > 0 "DO"
        "BEGIN" S:= S + LN(ASELECT(U)) / I;
            Y:= EXP(S);
            "IF" Y > G "THEN" SAMPLE[I]:= G + 1 "ELSE"
            "BEGIN" "FOR" G:= G - 1 "WHILE" Y <= G "DO";
                SAMPLE[I]:= G + 1
            "END"
        "END";
        "FOR" I:= I "STEP" -1 "UNTIL" 1 "DO" SAMPLE[I]:= 1
    "END" "ELSE"
    "BEGIN" "INTEGER" F; "REAL" P, Q;
        "FOR" F:= 0 "WHILE" N > 1 "AND" K >= 1 "DO"
        "BEGIN" P:= 0; ASELECT(U); Q:= K * LN(1 - 1 / N);
            "FOR" P:= P + EXP(Q) "WHILE" P < U "DO"
            "BEGIN"
                Q:= Q + LN((K - F) / ((F + 1) * (N - 1)));
                F:= F + 1
            "END";
            "FOR" F:= F "STEP" -1 "UNTIL" 1 "DO"
            "BEGIN" SAMPLE[K]:= N; K:= K - 1 "END";
            N:= N - 1
            "END";
        "FOR" K:= K "STEP" -1 "UNTIL" 1 "DO" SAMPLE[K]:= 1
    "END"
"END" RANDOM SORSAMREP;
         "EOP"
```

TITLE: **Success**

AUTHOR: J.G. Bethlehem

INSTITUTE: Mathematical Centre

RECEIVED: 770215

BRIEF DESCRIPTION

The procedure generates a random boolean, the probability of value "TRUE" being equal to a given value P and the probability of value "FALSE" being equal to (1 − P).

KEYWORDS

Random boolean

CALLING SEQUENCE

Heading

```
"BOOLEAN" "PROCEDURE" SUCCESS(P, U);
"VALUE" P;
"REAL" P, U;
"CODE" 41300;
```

Formal parameters

P: <arithmetic expression>, probability of value "TRUE";

U: <real variable>, value used to generate a random number.
After a call of SUCCESS, U contains a new value which is used in the next call of SUCCESS unless the program assigns a new value to U.

Conditions: $0 \leq P \leq 1$ and $0 \leq U < 1$.

DATA AND RESULTS

The generated value is assigned to the procedure indentifier SUCCESS.

The call TEST RANDOM(41300, P, U, 0, 0, 0) may yield the following error messages:

Errornumber 1	(if P<0 or P>1)
Errornumber 2	(if U<0 or U≥1)

PROCEDURES USED

ASELECT	STATAL 41308

LANGUAGE

Algol 60

Method and performance

The generated value is equal to "TRUE" if D ⩽ P and equal to "FALSE" if D > P, where D is a random number which is uniformly distributed on the interval (0, 1].

Reference

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

Example of use

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
   U:= .1986;
   "FOR" I:=1 "STEP" 1 "UNTIL" 10 "DO"
   OUTPUT(61, "("D,/")",
         "IF" SUCCESS(.3, U) "THEN" 1 "ELSE" 0)
"END"
```

Output:

```
0
1
0
0
1
0
0
0
1
0
```

Source text

```
"CODE" 41300;
"BOOLEAN" "PROCEDURE" SUCCESS(P, U); "VALUE" P; "REAL" P, U;
SUCCESS:= ASELECT(U) <= P;
         "EOP"
```

TITLE: **Random Bin**

AUTHOR: C. van Putten

INSTITUTE: Mathematical Centre

RECEIVED: 810101

BRIEF DESCRIPTION

The procedure generates a random number having a binomial distribution with parameters N and P. (see general part of section 5.2).

KEYWORDS

Random binomial number

CALLING SEQUENCE

Heading

```
"INTEGER" "PROCEDURE" RANDOM BIN (N, P, U);
"VALUE" N, P;
"INTEGER" N;
"REAL" P, U;
"CODE" 41324;
```

Formal parameters

N: <integer arithmetic expression>, first parameter of the binomial distribution;

P: <arithmetic expression>, second parameter of the binomial distribution;

U: <real variable>, value used to generate a random number. After a call of RANDOM BIN, U contains a new value which is used in the next call of RANDOM BIN unless the program assigns a new value to U.

Conditions: N integer $\geqslant 0$, and $0 \leqslant P \leqslant 1, 0 \leqslant U < 1$.

DATA AND RESULTS

The generated value is asssigned to the procedure indentifier RANDOM BIN.

The call TEST RANDOM(41324, N, P, U, 0, 0, 0) may yield the following error messages:

Errornumber 1	(if N is not an integer $\geqslant$ 0)
Errornumber 2	(if P<0 or P>1)
Errornumber 3	(if U<0 or U$\geqslant$1)

PROCEDURES USED

ASELECT	STATAL 41308

Language

Algol 60

Method and performance

See v. Putten & v.d. Tweel (1979).

Reference

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

Example of use

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    U:= .1986;
    "FOR" I:= 1 "STEP" 1 "UNTIL" 10 "DO"
    OUTPUT(61, "("ZD,/")",
                RANDOM BIN(100, .18, U));
"END"
```

Output:

```
17
17
20
16
18
21
17
23
22
15
```

Source text

```
"CODE" 41324;
"INTEGER" "PROCEDURE" RANDOM BIN(N, P, U);
"VALUE" N, P; "INTEGER" N; "REAL" P, U;
"IF" P = 0 "THEN" RANDOM BIN:= 0 "ELSE"
"IF" P = 1 "THEN" RANDOM BIN:= N "ELSE"
"BEGIN" "INTEGER" I, X; "REAL" LNQ; "BOOLEAN" B;
    B:= P <= .5; LNQ:= LN("IF" B "THEN" 1 - P "ELSE" P);
    I:= X:= 0;
    "FOR" X:= X + ENTIER(LN(ASELECT(U)) / LNQ + 1)
    "WHILE" X <= N "DO" I:= I + 1;
    RANDOM BIN:= "IF" B "THEN" I "ELSE" N - I;
"END" RANDOM BIN;
"EOP"
```

TITLE: **Random Binalias**

AUTHOR: B.F. Schriever

INSTITUTE: Mathematical Centre

RECEIVED: 810101

BRIEF DESCRIPTION

The procedure generates a random sample of size K from a binomial distribution with parameters N and P, according to the Alias method (see general part of section 5.2).

KEYWORDS

Random binomial sample by Alias method

CALLING SEQUENCE

Heading

```
"PROCEDURE" RANDOM BINALIAS (N, P, K, SAMPLE, U);
"VALUE" N, P, K;
"INTEGER" N, K;
"INTEGER" "ARRAY" SAMPLE;
"REAL" U, P;
"CODE" 41330;
```

Formal parameters

N: <integer arithmetic expression>, first parameter of the binomial distribution;

P: <arithmetic expression>, second parameter of the binomial distribution;

K: <integer arithmetic expression>, length of the vector SAMPLE[1:K] and size of the sample;

SAMPLE: <integer array identifier>, output parameter, one dimensional integer array [1:K] which at exit contains the sample;

U: <real variable>, value used to generate a random number. After a call of RANDOM BINALIAS, U contains a new value which is used in the next call of RANDOM BINALIAS unless the program assigns a new value to U.

Conditions: N and K integers, $N \geq 0$, $K > 0$, $0 < P < 1$, and $0 \leq U < 1$.

DATA AND RESULTS

After a procedure call, SAMPLE[1],..,SAMPLE[K] contain the generated sample. The call TEST RANDOM(41330, N, P, K, U, 0, 0) may yield the following error messages:

Errornumber 1	(if N is not an integer $\geq$0)
Errornumber 2	(if P$\leq$0 or P$\geq$1)
Errornumber 3	(if K is not integer >0)

Errornumber 5 (if U<0 or U≥1)

PROCEDURES USED

ASELECT	STATAL 41308
LOGGAMMA	STATAL 40400

LANGUAGE
Algol 60

METHOD AND PERFORMANCE
See Kronmal & Peterson (1979).

REFERENCE

[1]. R.A. Kronmal, A.N. Peterson jr.
On the Alias method for generating random variables from a discrete distribution.
The American Statistician, 1979, p.

EXAMPLE OF USE

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    "INTEGER" "ARRAY" SAMPLE[1 : 10];
    U:= .1986;
    RANDOM BINALIAS(100, .5, 10, SAMPLE, U);
    OUTPUT(61,"("10(ZD,2B),/")", SAMPLE)
"END"
```

Output:

```
53  49  57  38  45  57  55  41  48  43
```

SOURCE TEXT

```
"CODE" 41330;
"PROCEDURE" RANDOM BIN ALIAS ( N, P, SMPLSIZE, SAMPLE, U);
"VALUE" N, P, SMPLSIZE;
"INTEGER" "ARRAY" SAMPLE;
"INTEGER" N, SMPLSIZE;
"REAL" P, U;
"BEGIN" "INTEGER" "ARRAY" L[0 : N]; "ARRAY" F[0 : N];
    "REAL" Q, HLP, PM, ONE, R;
    "INTEGER" I1, I2, I3, I4, I, J, MDS, V;
    ONE:= .999999999;
    Q:= (1 - P) / P; MDS:= ENTIER((N + 1) * P);
    F[MDS]:= HLP:= PM:=
        EXP(LOGGAMMA(N + 1) - LOGGAMMA(MDS + 1)
        - LOGGAMMA(N + 1 - MDS) - MDS * LN(Q) +
        N * LN(1 - P)) * (N + 1);
    I1:= I2:= I3:= I4:= MDS;
    "FOR" I:= MDS - 1 "STEP" -1 "UNTIL" 0 "DO"
    "BEGIN" F[I]:= HLP:= HLP * Q * (I + 1) / (N - I);
        "IF" F[I] < ONE "THEN"
        "BEGIN" F[I2]:= F[I2] + F[I] - 1; L[I]:= I2;
            "IF" F[I2] < ONE "THEN" I2:= I2 + 1
        "END"
        "ELSE" I1:= I2:= I
    "END";
    HLP:= PM;
    "FOR" I:= MDS + 1 "STEP" 1 "UNTIL" N "DO"
    "BEGIN" F[I]:= HLP:= HLP * (N - I + 1) / (I * Q);
        "IF" F[I] < ONE "THEN"
        "BEGIN" F[I4]:= F[I4] + F[I] - 1; L[I]:= I4;
            "IF" F[I4] < ONE "THEN" I4:= I4 - 1
        "END"
                        "ELSE" I3:= I4:= I
    "END";
    I:= I1 - 1;
    "FOR" I:= I + 1 "WHILE" I < I2 "DO"
    "BEGIN" F[I2]:= F[I2] + F[I] - 1; L[I]:= I2;
        "IF" F[I2] < ONE "THEN" I2:= I2 + 1
    "END";
    I:= I3 + 1;
    "FOR" I:= I - 1 "WHILE" I > I4 "DO"
    "BEGIN" F[I4]:= F[I4] + F[I] - 1; L[I]:= I4;
        "IF" F[I4] < ONE "THEN" I4:= I4 - 1
    "END";

    N:= N + 1;
    "FOR" I:= 1 "STEP" 1 "UNTIL" SMPLSIZE "DO"
    "BEGIN" R:= ASELECT(U); R:= U * N; V:= ENTIER(R);
        "IF" R < V + F[V] "THEN" SAMPLE[I]:= V
                          "ELSE" SAMPLE[I]:= L[V]
    "END"
"END" RANDOM BIN ALIAS;
         "EOP"
```

TITLE: **Random Binalsors**

AUTHOR: B.F. Schriever

INSTITUTE: Mathematical Centre

RECEIVED: 810101

BRIEF DESCRIPTION

The procedure generates a sorted sample of size K from a binomial distribution with parameters N and P according to the Alias method. It is presented in a histogram (see general part of section 5.2).

KEYWORDS

Sorted random binomial sample by Alias method

CALLING SEQUENCE

Heading

```
"PROCEDURE" RANDOM BINALSORS (N, P, K, SAMPLE, U);
"VALUE" N, P, K;
"INTEGER" N,K;
"INTEGER" "ARRAY" SAMPLE;
"REAL" P, U;
"CODE" 41331;
```

Formal parameters

N: <integer arithmetic expression>, first parameter of the binomial distribution;

P: <arithmetic expression>, second parameter of the binomial distribution;

K: <integer arithmetic expression>, size of the sample;

SAMPLE: <integer array identifier>, output parameter, integer array of dimension [0:N] which at exit contains the histogram;

U: <real variable>, value used to generate a random number. After a call of RANDOM BINALSORS, U contains a new value which is used in the next call of RANDOM BINALSORS unless the program assigns a new value to U.

Conditions: N and K integers, N≥0, K>0, 0<P<1, and 0≤U<1.

DATA AND RESULTS

After a procedure call, SAMPLE[I] contains the number of times the value I is observed in the sample, I=0,...,N.

The call TEST RANDOM(41331, N, P, K, U, 0, 0) may yield the following error messages:

Errornumber 1	(if N is not an integer ≥0)
Errornumber 2	(if P≤0 or P≥1)
Errornumber 3	(if K is not an integer >0)

Errornumber 5 (if `U<0` or `U>=1`)

Procedures used

```
ASELECT            STATAL 41308
LOGGAMMA           STATAL 40400
```

Language

Algol 60

Method and performance

See Kronmal & Peterson (1979).

Reference

[1]. R.A. Kronmal, A.V. Peterson jr.
On the Alias method for generating random variables from a discrete distribution.
The American Statistican, 1979, p.

Example of use

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
   "INTEGER" "ARRAY" HIST[0 : 10];
   U:= .1986;
   RANDOM BINALSORS(10, .5, 100, HIST, U);
   "FOR" I:= 0 "STEP" 1 "UNTIL" 10 "DO"
   OUTPUT(61,"("2(ZD,2B),/")", I, HIST[I])
"END"
```

Output:

```
 0  0
 1  1
 2  4
 3 12
 4 19
 5 28
 6 17
 7 14
 8  2
 9  3
10  0
```

Source text

```
"CODE" 41331;
"PROCEDURE" RANDOM BIN ALSORS(N, P, SAMPLESIZE, SAMPLE, U);
"VALUE" N, P, SAMPLESIZE;
"INTEGER" "ARRAY" SAMPLE;
"INTEGER" N, SAMPLESIZE;
"REAL" P, U;
"BEGIN" "INTEGER" "ARRAY" L[0 : N]; "ARRAY" F[0 : N];
    "REAL" Q, HLP, PM, ONE, R;
    "INTEGER" I1, I2, I3, I4, I, J, MDS, V, M;
    ONE:= .999999999; M:= N + 1;
    Q:= (1 - P) / P; MDS:= ENTIER(M * P);
    F[MDS]:= HLP:= PM:=
        M * EXP(LOGGAMMA(M) - LOGGAMMA(MDS + 1)
        - LOGGAMMA(M - MDS) - MDS * LN(Q) + N * LN(1 - P));
    I1:= I2:= I3:= I4:= MDS; SAMPLE[MDS]:= 0;
    "FOR" I:= MDS - 1 "STEP" -1 "UNTIL" 0 "DO"
    "BEGIN" F[I]:= HLP:= HLP * Q * (I + 1) / (N - I);
        SAMPLE[I]:= 0;
        "IF" HLP < ONE "THEN"
        "BEGIN" R:= F[I2]:= F[I2] + HLP - 1; L[I]:= I2;
            "IF" R < ONE "THEN" I2:= I2 + 1
        "END"
        "ELSE" I1:= I2:= I
    "END";
    HLP:= PM;
    "FOR" I:= MDS + 1 "STEP" 1 "UNTIL" N "DO"
    "BEGIN" F[I]:= HLP:= HLP * (M - I) / (I * Q);
        SAMPLE[I]:= 0;
        "IF" HLP < ONE "THEN"
        "BEGIN" R:= F[I4]:= F[I4] + HLP - 1; L[I]:= I4;
            "IF" R < ONE "THEN" I4:= I4 - 1
        "END"
        "ELSE" I3:= I4:= I
    "END";
    I:= I1 - 1;
    "FOR" I:= I + 1 "WHILE" I < I2 "DO"
    "BEGIN" R:= F[I2]:= F[I2] + F[I] - 1; L[I]:= I2;
        "IF" R < ONE "THEN" I2:= I2 + 1
    "END";
    I:= I3 + 1;
    "FOR" I:= I - 1 "WHILE" I > I4 "DO"
    "BEGIN" R:= F[I4]:= F[I4] + F[I] - 1; L[I]:= I4;
        "IF" R < ONE "THEN" I4:= I4 - 1
    "END";
    "COMMENT" R TAKES ON ANOTHER MEANING;
    "FOR" I:= 1 "STEP" 1 "UNTIL" SAMPLESIZE "DO"
    "BEGIN" R:= ASELECT(U);
       R:= U * M;
       V:= ENTIER(R);
       "IF" R < V + F[V] "THEN" SAMPLE[V]:= SAMPLE[V] + 1
                         "ELSE"
       "BEGIN" J:= L[V]; SAMPLE[J]:= SAMPLE[J] + 1
```

```
        "END"
    "END"
"END" RANDOM BIN ALSORS;
          "EOP"
```

TITLE: **Random Binhisto**

AUTHOR: C. van Putten

INSTITUTE: Mathematical Centre

RECEIVED: 810101

BRIEF DESCRIPTION

The procedure generates a sorted random sample of size K from a binomial distribution with parameters N and P. It is presented in a histogram.
(see general part of section 5.2).

KEYWORDS

Sorted random binomial sample

CALLING SEQUENCE

Heading

```
"PROCEDURE" RANDOM BINHISTO ( N, P, SAMPLE, K, U);
"VALUE" N, P, K;
"INTEGER" N, K;
"INTEGER" "ARRAY" SAMPLE;
"REAL" P, U;
"CODE" 41332;
```

Formal parameters

N: <integer arithmetic expression>, first parameter of the binomial distribution;

P: <arithmetic expression>, second parameter of the binomial distribution;

K: <integer arithmetic expression>, size of the sample;

SAMPLE: <integer array identifier>, output parameter; integer array of dimension [0:N] wich at exit contains the histogram;

U: <real variable>, value used to generate a random number. After a call of RANDOM BINHISTO, U contains a new value which is used in the next call of RANDOM BINHISTO unless the program assigns a new value to U.

Conditions: N and K integers, N ⩾0, K>0, 0<P<1, and 0⩽U<1.

DATA AND RESULTS

After a procedure call, SAMPLE[I] contains the number of times the value I is observed in the sample, I=0,...,N.

The call TEST RANDOM(41332, N, P, K, U, 0, 0) may yield the following error messages:

Errornumber 1	(if N is not an integer ⩾0)
Errornumber 2	(if P⩽0 or P⩾1)
Errornumber 4	(if K is not an integer >0)

Errornumber 5 (if U<0 or U≥1)

Procedures used

ASELECT STATAL 41308

Language

Algol 60

Method and performance

See v. Putten & van der Tweel (1979).

Reference

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

Example of use

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    "INTEGER" "ARRAY" HIST[0 : 10];
    U:= .1986;
    RANDOM BINHISTO(10, .5, HIST, 100, U);
    "FOR" I:= 0 "STEP" 1 "UNTIL" 10 "DO"
    OUTPUT(61,"("2(ZD,2B),/")", I, HIST[I])
"END"
```

Output:

```
 0   0
 1   1
 2   3
 3  12
 4  20
 5  26
 6  21
 7  11
 8   5
 9   1
10   0
```

SOURCE TEXT

```
"CODE" 41332;
"PROCEDURE" RANDOM BIN HISTO(N, P, SAMPLE, SIZE, U);
"VALUE" N, P, SIZE;
"INTEGER" N, SIZE;
"REAL" P, U;
"INTEGER" "ARRAY" SAMPLE;
"BEGIN" "INTEGER" I, M, COUNT;
    "REAL" PROB, SUMPROB, Q, S, LN P DIV Q, R, T, LN PROB;
    "BOOLEAN" REVERSED;

    REVERSED:= P > .5;
    "IF" REVERSED "THEN" "BEGIN" Q:= P; P:= 1 - P "END"
                 "ELSE" Q:= 1 - P;
    LN P DIV Q:= LN(P / Q); T:= 0; LN PROB:= N * LN(Q);
    SUMPROB:= PROB:= EXP(LN PROB); S:= 1 - PROB;
    R:= LN(S); I:= COUNT:= 0;
    "FOR" M:= SIZE "STEP" -1 "UNTIL" 1 "DO"
    "BEGIN" T:= T + LN(ASELECT(U)) / M;
      L:"IF" T > R "THEN" COUNT:= COUNT + 1
                   "ELSE"
        "IF" I < N - 1 "THEN"
        "BEGIN"
            SAMPLE["IF" REVERSED "THEN" N - I "ELSE" I]:=
                COUNT;
            COUNT:= 0; I:= I + 1;
            LN PROB:= LN PROB + LN P DIV Q +
                LN(( N - I + 1 ) / I);
            SUMPROB:= SUMPROB + EXP(LN PROB);
            S:= 1 - SUMPROB;
            R:= LN(S); "GOTO" L
        "END"
        "ELSE"
        "BEGIN" "IF" REVERSED "THEN"
            "BEGIN" SAMPLE[1]:= COUNT; SAMPLE[0]:= M "END"
            "ELSE"
            "BEGIN" SAMPLE[N - 1]:= COUNT;
                SAMPLE[N]:= M
            "END";
            "GOTO" OUT
        "END"
    "END";
    SAMPLE["IF" REVERSED "THEN" N - I "ELSE" I]:= COUNT;
    "IF" REVERSED "THEN"
    "BEGIN" "FOR" I:= I + 1 "STEP" 1 "UNTIL" N "DO"
        SAMPLE[N - I]:= 0
    "END"
    "ELSE" "FOR" I:= I + 1 "STEP" 1 "UNTIL" N "DO"
                         SAMPLE[I]:=0;
    OUT:
"END" RANDOM BIN HISTO;
         "EOP"
```

TITLE: **Random Geometric**

AUTHOR: C. van Putten

INSTITUTE: Mathematical Centre

RECEIVED: 810101

BRIEF DESCRIPTION

The procedure generates a random number having a geometric distribution with parameter P.

KEYWORDS

Random geometric number

CALLING SEQUENCE

Heading

```
"INTEGER" "PROCEDURE" RANDOM GEOMETRIC (P, U);
"VALUE" P;
"REAL" P, U;
"CODE" 41323;
```

Formal parameters

P: <arithmetic expression>, parameter of the geometric distribution;

U: <real variable>, value used to generate a random number. After a call of RANDOM GEOMETRIC, U contains a new value which is used in the next call of RANDOM GEOMETRIC unless the program assigns a new value to U.

Conditions: 0<P≤1 and 0≤U <1.

DATA AND RESULTS

The generated value is assigned to the procedure identifier RANDOM GEOMETRIC.

The call TEST RANDOM(41323, P, U, 0, 0, 0, 0) may yield the following error messages:

Errornumber 1	(if P≤ or P>1)
Errornumber 2	(if U<0 or U≥1)

PROCEDURES USED

ASELECT	STATAL 41308

LANGUAGE

Algol 60

METHOD AND PERFORMANCE

If P<1, the generated value is equal to ENTIER(LN(D)/LN(1−P)+1), where D is a random number which is uniformly distributed on (0,1].
If P=1, The generated value equals 1. See v. Putten & v.d. Tweel (1979).

REFERENCE

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

EXAMPLE OF USE

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
   U:= .1986;
   "FOR" I:= 1 "STEP" 1 "UNTIL" 10 "DO"
   OUTPUT(61, "("ZD,/")",
              RANDOM GEOMETRIC(0.5, U));
"END"
```

Output:

```
 1
 5
 1
 2
 2
 1
 1
 2
 4
 2
```

SOURCE TEXT

```
"CODE" 41323;
"INTEGER" "PROCEDURE" RANDOM GEOMETRIC(P, U);
"VALUE" P; "REAL" P, U;
RANDOM GEOMETRIC:= "IF" P = 1 "THEN" 1 "ELSE"
                   ENTIER(LN(ASELECT(U)) / LN(1 - P) + 1);
"EOP"
```

TITLE: **Random Sthyperg**

AUTHOR: C. van Putten

INSTITUTE: Mathematical Centre

RECEIVED: 810101

BRIEF DESCRIPTION

The procedure generates a random number having a hypergeometric distribution with parameters N,R and NN. It is assumed that the parameters are standardized; i.e. N ≤ R ≤ NN−R ≤ NN−N ≤ NN.

KEYWORDS

Random standardized hypergeometric number.

CALLING SEQUENCE

Heading

```
"INTEGER" "PROCEDURE" RANDOM  STHYPERG (N, R, NN, U);
"VALUE" N, R, NN;
"INTEGER" N, R, NN;
"REAL" U;
"CODE" 41316;
```

Formal parameters

N <integer arithmetic expression>, first parameter of the hypergeometric distribution, the size of the sample taken from the population;

R: <integer arithmetic expression>, second parameter of the hypergeometric ditribution, the number of elements in the population with a given property;

NN <integer arithmetic expression>, third parameter of the hypergeometric distribution, the number of elements in the population;

U: <real variable>, value used to generate a random number. After a call of RANDOM STHYPERG, U contains a new value which is used in the next call of RANDOM STHYPERG unless the program assigns a new value to U.

Conditions: N,R,NN integers >0,N≤R≤ NN−R≤ NN−N≤NN, and 0≤U<1.

DATA AND RESULTS

The generated value is assigned to the procedure identifier RANDOM STHYPERG.
The call TEST RANDOM(41316, N, R, NN, U, 0, 0) may yield the following error messages:

Errornumber 1	(if N is not an integer >0)
Errornumber 2	(if R is not an integer >0)
Errornumber 3	(if NN is not an integer >0)
Errornumber 4	(if U<0 or U≥1)
Errornumber 5	(if N>R or R>NN−R)

PROCEDURES USED

```
ASELECT                    STATAL 41308
```

PROCEDURES USED

Algol 60

METHOD AND PERFORMANCE

See v. Putten & van der Tweel (1979).

REFERENCE

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

EXAMPLE OF USE

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    U:= .1986;
    "FOR" I:= 1 "STEP" 1 "UNTIL" 10 "DO"
    OUTPUT(61, "("2ZD,/")", RANDOM STHYPERG(10, 15, 50, U));
"END"
```

Output:

```
  3
  3
  2
  2
  1
  4
  3
  4
  4
  3
```

SOURCE TEXT

```
"CODE" 41316;
"INTEGER" "PROCEDURE" RANDOM STHYPERG(N, R, NN, U);
"VALUE" N, R, NN; "INTEGER" N, R, NN; "REAL" U;
"BEGIN" "INTEGER" A, I;
  A:= 0;
  "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
  "BEGIN" "IF" ASELECT (U) * NN <= R "THEN"
    "BEGIN" A:= A + 1; R:= R - 1 "END";
    NN:= NN - 1;
    "IF" R = NN "THEN"
    "BEGIN" A:= A + N - I; "GOTO" OUT "END"
  "END";

  OUT: RANDOM STHYPERG:= A

"END" RANDOM STHYPERG;
         "EOP"
```

Title: **Random Hyperg**

Author: C. van Putten

Institute: Mathematical Centre

Received: 810101

Brief description

The procedure generates a random number having a hypergeometric distribution with parameters N, R and NN.

Keywords

Random hypergeometric number

Calling sequence

Heading

```
""INTEGER" "PROCEDURE" RANDOM HYPERG (N, R, NN, U);
"VALUE" N, R, NN;
"INTEGER" N, R, NN;
"REAL" U;
"CODE" 41317;
```

Formal parameters

N: <integer arithmetic expression>, first parameter of the hypergeometric distribution, the size of the sample taken from the population;

R: <integer arithmetic expression>, second parameter of the hypergeometric distribution, the number of elements in the population with a given property;

NN: <integer arithmetic expression>, third parameter of the distribution, the number of elements in the population;

U: <real variable>, value used to generate a random number.
After a call of RANDOM HYPERG, U contains a new value which is used in the next call of RANDOM HYPERG unless the program assigns a new value to U.

Conditions: N, R, NN integers > 0, N ≤ NN, R≤NN, and 0≤U<1.

Data and results

The generated value is assigned to the procedure identifier RANDOM HYPERG.
The call TEST RANDOM(41317, N, R, NN, U, 0, 0) may yield the following error messages:

Errornumber 1	(if N > NN or not an integer >0)
Errornumber 2	(if R > NN or not an integer >0)
Errornumber 3	(if NN is not an integer >0)
Errornumber 4	(if U<0 or U≥1)

Procedures used

ASELECT	STATAL 41308

Language

Algol 60

Method and performance

After interchanging the roles of N, R, NN−N and NN−R, the same method as in section 5.2.5.1 is applied; see v. Putten & v.d. Tweel (1979).

Reference

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

Example of use

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    U:= .1986;
    "FOR" I:= 1 "STEP" 1 "UNTIL" 10 "DO"
    OUTPUT(61, "("2ZD,/")", RANDOM HYPERG(18, 12, 60, U));
"END"
```

Output:

```
  4
  3
  2
  2
  4
  3
  5
  5
  4
  3
```

Source text

```
"CODE" 41317;
"INTEGER" "PROCEDURE" RANDOM HYPERG(N, R, NN, U);
"VALUE" N, R, NN; "INTEGER" N, R, NN; "REAL" U;
"BEGIN" "INTEGER" AUX, A, I, NN1;
    "BOOLEAN" CHROW, CHCOL, BAUX;

    "COMMENT" CHROW EN CHCOL INDICATE WHICH INTERCHANGES
              ARE REQUIRED TO GET THE A TOP LEFT IN THE
              2 X 2 - TABLE ;
    CHROW:= CHCOL:= "FALSE"; A:= 0; NN1:= NN;
    "IF" N * 2 > NN "THEN"
    "BEGIN" N:= NN - N; CHROW:= "TRUE" "END";
    "IF" R * 2 > NN "THEN"
    "BEGIN" R:= NN - R; CHCOL:= "TRUE" "END";
    "IF" N > R "THEN"
    "BEGIN" AUX:= N; N:= R; R:= AUX;
            BAUX:= CHROW; CHROW:= CHCOL; CHCOL:= BAUX;
    "END";

    "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
    "BEGIN" "IF" ASELECT(U) * NN <= R "THEN"
        "BEGIN" A:= A + 1; R:= R - 1 "END";
        NN:= NN - 1;
        "IF" R = NN "THEN"
        "BEGIN" A:= A + N - I; R:= R + I - N;
          "GOTO" OUT
        "END"
  "END";

    "COMMENT" R CONTAINS THE NUMBER OF RED BALLS OUTSIDE
              THE SAMPLE ;
OUT:
    "IF" CHROW "THEN" "BEGIN" A:= R; N:= NN1 - N "END";
    RANDOM HYPERG:= "IF" CHCOL "THEN" N - A "ELSE" A

"END" RANDOM HYPERG;
         "EOP"
```

TITLE: **Random Negbin**

AUTHOR: C. van Putten

INSTITUTE: Mathematical Centre

RECEIVED: 810101

BRIEF DESCRIPTION

The procedure generates a random number having a negative binomial distribution with parameters K and P.

KEYWORDS

Random negative binomial number.

CALLING SEQUENCE

Heading

```
"INTEGER" "PROCEDURE" RANDOM NEGBIN (K, P, U);
"VALUE" K, P;
"INTEGER" K;
"REAL" P, U;
"CODE" 41325;
```

Formal parameters

K: <integer arithmetic expression>, first parameter of the negative binomial distribution, required number of successes;

P: <arithmetic expression>, second parameter of the negative binomial distribution, the probability of success in a single experiment;

U: <real variable>, value used to generate a random number. After a call of RANDOM NEGBIN, U contains a new value which is used in the next call of RANDOM NEGBIN unless the program assigns a new value to U.

Conditions: K integer >0,0<P ≤1, and 0≤U<1.

DATA AND RESULTS

The generated value is assigned to the procedure identifier RANDOM NEGBIN.

The call TEST RANDOM(41325, K, P, U, 0, 0, 0) may yield the following error messages:

Errornumber 1	(if K is not an integer >0)
Errornumber 2	(if P≤0 or P>1)
Errornumber 3	(if U<0 or U≥1)

PROCEDURES USED

ASELECT	STATAL 41308

Language

Algol 60

Method and performance

The generated value is equal to the sum of K independent random geometric numbers (section 5.2.4.1.); see v. Putten & v.d. Tweel (1979)

Reference

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

Example of use

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    U:= .1986;
    "FOR" I:= 1 "STEP" 1 "UNTIL" 10 "DO"
    OUTPUT(61, "("ZD,/")",
                 RANDOM NEGBIN(10, .5, U));
"END"
```

Output:

```
21
22
21
16
19
22
23
19
18
18
```

Source text

```
"CODE" 41325;
"INTEGER" "PROCEDURE" RANDOM NEGBIN (K, P, U);
"VALUE" K, P; "INTEGER" K; "REAL" P, U;
"BEGIN" "IF" P = 1 "THEN" RANDOM NEGBIN:= K "ELSE"
    "BEGIN" "INTEGER" I, S; "REAL" LNQ;
        LNQ:= LN(1 - P); S:= K;
        "FOR" I:= 1 "STEP" 1 "UNTIL" K "DO"
        S:= S + ENTIER(LN(ASELECT(U)) / LNQ);
        RANDOM NEGBIN:= S
    "END"
"END" RANDOM NEGBIN;
         "EOP"
```

TITLE: **Random Pois**

AUTHOR: J.G. Bethlehem

INSTITUTE: Mathematical Centre

RECEIVED: 750219

BRIEF DESCRIPTION

The procedure generates a random number having a Poisson distribution with parameter MU≤700 (see general part part of section 5.2).

KEYWORDS

Random Poisson number

CALLING SEQUENCE

Heading

```
"INTEGER" "PROCEDURE" RANDOM POIS (MU, U);
"VALUE" MU;
"REAL" MU, U;
"CODE" 41305;
```

Formal parameters

MU: <arithmetic expression>, parameter of the Poisson distribution;

U: <real variable>, value used to generate a random number. After a call of RANDOM POIS, U contains a new value which is used in the next call of RANDOM POIS unless the program assigns a new value to U.

Conditions: 0<MU≤700.

DATA AND RESULTS

The generated value is assigned to the procedure identifier RANDOM POIS.
The call TEST RANDOM(41305, MU, U, 0, 0, 0, 0) may yield the following error messages:

Errornumber 1	(if MU≤0 or MU>700)
Errornumber 2	(if U<0 or U≥1)

PROCEDURES USED

ASELECT	STATAL 41308

LANGUAGE

Algol 60

Method and performance

The generated value is the smallest integer for which the cumulative distribution function of the Poisson distribution with parameter MU is greater than or equal to a random number D having a uniform distribution on the interval (0, 1).
See v. Putten & v.d. Tweel (1979).

In order to avoid overflow MU is restricted to MU≤700.

Reference

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

Example of use

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    U:= .1986;
    "FOR" I:=1 "STEP" 1 "UNTIL" 10 "DO"
    OUTPUT(61, "("ZD,/")", RANDOM POIS(5, U))
"END"
```

Output:

```
 7
 1
 6
 4
 4
 6
 5
 4
 2
 4
```

Source text

```
"CODE" 41305;
"INTEGER" "PROCEDURE" RANDOM POIS(MU, U); "VALUE" MU;
"REAL" MU, U;
"BEGIN" "INTEGER" I; "REAL" SUM, TERM, W;
    ASELECT(U); "IF" U = 0 "THEN" ASELECT(U);
    W:= U * EXP(MU); SUM:= TERM:= 1; I:= 0;
    "FOR" I:= I + 1 "WHILE" SUM < W "DO"
    "BEGIN" TERM:= TERM * MU / I; SUM:= SUM + TERM "END";
    RANDOM POIS:= I - 1
"END" RANDOM POIS;
        "EOP"
```

TITLE: **Random Poistab**

AUTHOR: B.F. Schriever

INSTITUTE: Mathematical Centre

RECEIVED: 800429

BRIEF DESCRIPTION

The procedure generates a random number having a Poisson distribution with parameter MU<256, using a table search method. (see general part of section 5.2).

KEYWORDS

Random Poisson number by table search method.

CALLING SEQUENCE

Heading

```
"INTEGER" "PROCEDURE" RANDOM POISTAB (MU, U);
"VALUE" MU;
"REAL" MU, U;
"CODE" 41326;
```

Formal parameters

MU: <arithmetic expression>, expectation of the Poisson distribution;

U: <real variable>, value used to generate a random number.
After a call of RANDOM POISTAB, U contains a new value which is used in the next call of RANDOM POISTAB unless the program assigns a new value to U.

Conditions: 0<MU<256, and 0≤U<1.

DATA AND RESULTS

The generated value is assigned to the procedure identifier RANDOM POISTAB. The call TEST RANDOM(41326, M, U, 0, 0, 0, 0) may yield the following error messages:

Errornumber 1	(if MU≤ or MU≥256)
Errornumber 2	(if U<0 or U≥1)

PROCEDURES USED

ASELECT STATAL 41308

LANGUAGE

Algol 60

Method and performance

The parameter MU is decomposed according to $MU=\Sigma_i b_i * 2^i + R$, where $b_i \in \{0,1\}$ and $0 \leq R < 2$. For each i such that $b_i = 1$, a realisation from a Poisson distribution with expectation 2^i is generated using a table. If $R>0$, a random number having a Poisson distribution with expectation R is generated by the multiplication method (c.f. section 5.2.7.1).
The sum of all these random numbers is assigned to the procedure identifier. (see v. Putten & v.d. Tweel, 1979).

Reference

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

Example of use

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
   U:= .1986;
   "FOR" I:= 1 "STEP" 1 "UNTIL" 10 "DO"
   OUTPUT(61, "("ZD,/")",
                RANDOM POISTAB(10, U));
"END"
```

Output:

```
11
10
 8
10
 5
 8
 9
10
 7
 7
```

Source text

```
"CODE" 41326;
"INTEGER" "PROCEDURE" RANDOM POIS TAB(MU, U);
"VALUE" MU; "REAL" MU, U;
"BEGIN"
    "ARRAY" TAB[1 : 235]; "INTEGER" S;

    "INTEGER" "PROCEDURE" POISS MUL(MU, U); "REAL" MU, U;
    "BEGIN" "INTEGER" N; "REAL" P;
        N:= -2; P:= EXP(MU);
        "FOR" N:= N + 1 "WHILE" P > 1
        "DO" P:= P * ASELECT(U);
        POISS MUL:= N
    "END" POISS MUL;

    "INTEGER" "PROCEDURE"
    POISS TAB(MU, FIRST, LAST, INDEXMU);
    "INTEGER" MU, FIRST, LAST, INDEXMU;
    "BEGIN" "INTEGER" N; "REAL" V, P, CUMPROB;
        V:= ASELECT(U);
        "IF" V <= TAB[INDEXMU - 1] "THEN"
        "BEGIN" N:= INDEXMU;
            "FOR" N:= N - 1 "WHILE" V <= TAB[N - 1]
                            "AND" N > FIRST + 1 "DO";
            "IF" N > FIRST + 1 "THEN"
                POISS TAB:= MU - INDEXMU + N
            "ELSE"
            "BEGIN" P:= TAB[FIRST];
                CUMPROB:= TAB[FIRST + 1] - P;
                N:= MU - INDEXMU + FIRST + 2;
                "FOR" N:= N - 1
                "WHILE" V <= CUMPROB "AND" N > 0 "DO"
                "BEGIN" P:= P * N / MU;
                    CUMPROB:= CUMPROB - P
                "END";
                POISS TAB:= N
            "END"
        "END"
        "ELSE"
        "BEGIN" N:= INDEXMU - 1;
            "FOR" N:= N + 1
            "WHILE" V > TAB[N] "AND" N < LAST "DO";
            "IF" N < LAST "THEN"
                POISS TAB:= MU - INDEXMU + N
            "ELSE"
            "BEGIN" P:= TAB[LAST]; CUMPROB:= TAB[LAST - 1];
                N:= MU - INDEXMU + LAST - 2;
                "FOR" N:= N + 1
                "WHILE" V > CUMPROB "AND" N < 500 "DO"
                "BEGIN" P:= MU * P / (N + 1);
                    CUMPROB:= CUMPROB + P
                "END";
                POISS TAB:= N
            "END"
        "END";
    "END" POISS TAB;

    S:=0;
    "IF" MU >= 128 "THEN"
    "BEGIN"
        TAB[164]:= +3.8120098644196"-004;
        TAB[165]:= +1.3778499806580"-003;
        TAB[166]:= +1.8861179625800"-003;
        TAB[167]:= +2.5568221036718"-003;
        TAB[168]:= +3.4328438389742"-003;
        TAB[169]:= +4.5654780017880"-003;
        TAB[170]:= +6.0152497301878"-003;
        TAB[171]:= +7.8525841978626"-003;
        TAB[172]:= +1.0158258823963"-002;
        TAB[173]:= +1.3023563213680"-002;
        TAB[174]:= +1.6550091693334"-002;
        TAB[175]:= +2.0849097839967"-002;
        TAB[176]:= +2.6040350545319"-002;
        TAB[177]:= +3.2250447239579"-002;
        TAB[178]:= +3.9610561840180"-002;
        TAB[179]:= +4.8253632196854"-002;
        TAB[180]:= +5.8311023157355"-002;
        TAB[181]:= +6.9908735255956"-002;
        TAB[182]:= +8.3163263368645"-002;
        TAB[183]:= +9.8177242115765"-002;
        TAB[184]:= +1.1503504281428"-001;
        TAB[185]:= +1.3379850793956"-001;
        TAB[186]:= +1.5450302118123"-001;
        TAB[187]:= +1.7715411259094"-001;
        TAB[188]:= +2.0172478801841"-001;
        TAB[189]:= +2.2815374982277"-001;
        TAB[190]:= +2.5634464241404"-001;
        TAB[191]:= +2.8616641308912"-001;
        TAB[192]:= +3.1745482822364"-001;
        TAB[193]:= +3.5001513015227"-001;
        TAB[194]:= +3.8362576440121"-001;
        TAB[195]:= +4.1804305387204"-001;
        TAB[196]:= +4.5300664952492"-001;
        TAB[197]:= +4.8824554908055"-001;
        TAB[198]:= +5.2348444863617"-001;
        TAB[199]:= +5.5845017842780"-001;
        TAB[200]:= +5.9287797391494"-001;
        TAB[201]:= +6.2651734660467"-001;
        TAB[202]:= +6.5913734436441"-001;
        TAB[203]:= +6.9053102641889"-001;
        TAB[204]:= +7.2051902121720"-001;
        TAB[205]:= +7.4895208295190"-001;
        TAB[206]:= +7.7571261164337"-001;
        TAB[207]:= +8.0071514939891"-001;
        TAB[208]:= +8.2390590905622"-001;
        TAB[209]:= +8.4526142874066"-001;
        TAB[210]:= +8.6478647530928"-001;
    TAB[211]:= +8.8251134027939"-001;
    TAB[212]:= +8.9848868335102"-001;
    TAB[213]:= +9.1279008134522"-001;
    TAB[214]:= +9.2550243511782"-001;
    TAB[215]:= +9.3672437499986"-001;
    TAB[216]:= +9.4656278804712"-001;
    TAB[217]:= +9.5512956947603"-001;
    TAB[218]:= +9.6253867773887"-001;
    TAB[219]:= +9.6890354926668"-001;
    TAB[220]:= +9.7433490630374"-001;
    TAB[221]:= +9.7893897054709"-001;
    TAB[222]:= +9.8281607727832"-001;
    TAB[223]:= +9.8605966983778"-001;
    TAB[224]:= +9.8875564287422"-001;
    TAB[225]:= +9.9098199480108"-001;
    TAB[226]:= +9.9280874510004"-001;
    TAB[227]:= +9.9429807018454"-001;
    TAB[228]:= +9.9550461202516"-001;
    TAB[229]:= +9.9647591614842"-001;
    TAB[230]:= +9.9725295944702"-001;
    TAB[231]:= +9.9787073300120"-001;
    TAB[232]:= +9.9835885037734"-001;
    TAB[233]:= +9.9874215727394"-001;
    TAB[234]:= +9.9904132363226"-001;
    TAB[235]:= +2.9916635832062"-004;
    S:= S + POISS TAB (128,164,235,198); MU := MU - 128
"END";
"IF" MU >= 64 "THEN"
"BEGIN"
    TAB[112]:= +5.4212965801156"-004;
    TAB[113]:= +1.4170798450894"-003;
    TAB[114]:= +2.2431821811069"-003;
    TAB[115]:= +3.4727298440162"-003;
    TAB[116]:= +5.2611628082474"-003;
    TAB[117]:= +7.8047119129324"-003;
    TAB[118]:= +1.1343562841191"-002;
    TAB[119]:= +1.6162423679673"-002;
    TAB[120]:= +2.2587571464318"-002;
    TAB[121]:= +3.0979601223857"-002;
    TAB[122]:= +4.1721399316074"-002;
    TAB[123]:= +5.5201302804350"-002;
    TAB[124]:= +7.1791953251461"-002;
    TAB[125]:= +9.1825946244196"-002;
    TAB[126]:= +1.1556993793930"-001;
    TAB[127]:= +1.4319931009360"-001;
    TAB[128]:= +1.7477573541280"-001;
    TAB[129]:= +2.1022996735015"-001;
    TAB[130]:= +2.4935187845342"-001;
    TAB[131]:= +2.9178920575189"-001;
    TAB[132]:= +3.3705568820359"-001;
    TAB[133]:= +3.8454839110379"-001;
    TAB[134]:= +4.3357311667811"-001;
    TAB[135]:= +4.8337601249961"-001;
    TAB[136]:= +5.3317890832110"-001;
    TAB[137]:= +5.8221560574534"-001;
    TAB[138]:= +6.2976634264158"-001;
    TAB[139]:= +6.7518794206484"-001;
    TAB[140]:= +7.1793768269851"-001;
    TAB[141]:= +7.5758961603988"-001;
    TAB[142]:= +7.9384281223770"-001;
    TAB[143]:= +8.2652174965546"-001;
    TAB[144]:= +8.5556969402680"-001;
    TAB[145]:= +8.8103638498250"-001;
    TAB[146]:= +9.0306163121444"-001;
    TAB[147]:= +9.2185650799904"-001;
    TAB[148]:= +9.3768377265976"-001;
    TAB[149]:= +9.5083890172840"-001;
    TAB[150]:= +9.6163285378473"-001;
    TAB[151]:= +9.7037732127340"-001;
    TAB[152]:= +9.7737289526433"-001;
    TAB[153]:= +9.8290026236828"-001;
    TAB[154]:= +9.8721430498600"-001;
    TAB[155]:= +9.9054079567918"-001;
    TAB[156]:= +9.9307526477874"-001;
    TAB[157]:= +9.9498357092430"-001;
    TAB[158]:= +9.9640370573030"-001;
    TAB[159]:= +9.9744840259906"-001;
    TAB[160]:= +9.9820818214000"-001;
    TAB[161]:= +9.9875454046156"-001;
    TAB[162]:= +9.9914306193467"-001;
    TAB[163]:= +3.8852147311420"-004;
    S:= S + POISS TAB (64,112,163,136); MU := MU - 64
"END";
"IF" MU >= 32 "THEN"
"BEGIN"
    TAB[ 74]:= +7.3173973442918"-004;
    TAB[ 75]:= +1.3916672870287"-003;
    TAB[ 76]:= +2.7690597283076"-003;
    TAB[ 77]:= +5.2177574016927"-003;
    TAB[ 78]:= +9.3418797989703"-003;
    TAB[ 79]:= +1.5940475634614"-002;
    TAB[ 80]:= +2.5995478812736"-002;
    TAB[ 81]:= +4.0620937980915"-002;
    TAB[ 82]:= +6.0969402910554"-002;
    TAB[ 83]:= +8.8100689483409"-002;
    TAB[ 84]:= +1.2282873629667"-001;
    TAB[ 85]:= +1.6557094775913"-001;
    TAB[ 86]:= +2.1622838356649"-001;
    TAB[ 87]:= +2.7412259591775"-001;
    TAB[ 88]:= +3.3800586471912"-001;
    TAB[ 89]:= +4.0614801810724"-001;
    TAB[ 90]:= +4.7648830547560"-001;
    TAB[ 91]:= +5.4682859284397"-001;
    TAB[ 92]:= +6.1503735635269"-001;
    TAB[ 93]:= +6.7923383965502"-001;
    TAB[ 94]:= +7.3792776724572"-001;
    TAB[ 95]:= +7.9010014732632"-001;
    TAB[ 96]:= +8.3522220577444"-001;
```

```
        TAB[ 97]:= +8.7321972867810"-001;
        TAB[ 98]:= +9.0439718336828"-001;
        TAB[ 99]:= +9.2933914712043"-001;
        TAB[100]:= +9.4880604565870"-001;
        TAB[101]:= +9.6363796835452"-001;
        TAB[102]:= +9.7467567826769"-001;
        TAB[103]:= +9.8270310365908"-001;
        TAB[104]:= +9.8841149504852"-001;
        TAB[105]:= +9.9238254992812"-001;
        TAB[106]:= +9.9508624686743"-001;
        TAB[107]:= +9.9688871149363"-001;
        TAB[108]:= +9.9806583124952"-001;
        TAB[109]:= +9.9881918789330"-001;
        TAB[110]:= +9.9929188225802"-001;
        TAB[111]:= +4.7269436471777"-004;
        S:= S + POISS TAB (32,74,111,91); MU := MU - 32
    "END";
    "IF" MU >= 16 "THEN"
    "BEGIN"
        TAB[ 46]:= +9.8334736138684"-004;
        TAB[ 47]:= +1.3837850247629"-003;
        TAB[ 48]:= +4.0060446551275"-003;
        TAB[ 49]:= +9.9997809531040"-003;
        TAB[ 50]:= +2.1987253549056"-002;
        TAB[ 51]:= +4.3298315941861"-002;
        TAB[ 52]:= +7.7396015770348"-002;
        TAB[ 53]:= +1.2699267006634"-001;
        TAB[ 54]:= +1.9312154246098"-001;
        TAB[ 55]:= +2.7451092386977"-001;
        TAB[ 56]:= +3.6752735976553"-001;
        TAB[ 57]:= +4.6674489138768"-001;
        TAB[ 58]:= +5.6596242300982"-001;
        TAB[ 59]:= +6.5934362924244"-001;
        TAB[ 60]:= +7.4234914589364"-001;
        TAB[ 61]:= +8.1224852833677"-001;
        TAB[ 62]:= +8.6816803429126"-001;
        TAB[ 63]:= +9.1077337216136"-001;
        TAB[ 64]:= +9.4175907243052"-001;
        TAB[ 65]:= +9.6331434218297"-001;
        TAB[ 66]:= +9.7768452201794"-001;
        TAB[ 67]:= +9.8688143711232"-001;
        TAB[ 68]:= +9.9254107717040"-001;
        TAB[ 69]:= +9.9589493794557"-001;
        TAB[ 70]:= +9.9781142981709"-001;
        TAB[ 71]:= +9.9886880464275"-001;
        TAB[ 72]:= +9.9943273788310"-001;
        TAB[ 73]:= +5.6393324035520"-004;
        S:= S + POISS TAB (16,46,73,58); MU := MU - 16
    "END";
    "IF" MU >= 8 "THEN"
    "BEGIN"
        TAB[ 26]:= +2.6837010232201"-003;
        TAB[ 27]:= +3.0191636511226"-003;
        TAB[ 28]:= +1.3753967744003"-002;
        TAB[ 29]:= +4.2380111991685"-002;
        TAB[ 30]:= +9.9632400487048"-002;
        TAB[ 31]:= +1.9123606207963"-001;
        TAB[ 32]:= +3.1337427753641"-001;
        TAB[ 33]:= +4.5296080948703"-001;
        TAB[ 34]:= +5.9254734143764"-001;
        TAB[ 35]:= +7.1662425872706"-001;
        TAB[ 36]:= +8.1588579255860"-001;
        TAB[ 37]:= +8.8807599898154"-001;
        TAB[ 38]:= +9.3620280326350"-001;
        TAB[ 39]:= +9.6581929820625"-001;
        TAB[ 40]:= +9.8274300960210"-001;
        TAB[ 41]:= +9.9176898901322"-001;
        TAB[ 42]:= +9.9628197871878"-001;
        TAB[ 43]:= +9.9840573858022"-001;
        TAB[ 44]:= +9.9934963185198"-001;
        TAB[ 45]:= +9.4389327175133"-004;
        S:= S + POISS TAB (8,26,45,34); MU := MU - 8
    "END";
    "IF" MU >= 4 "THEN"
    "BEGIN"
        TAB[ 12]:= +1.8315638888734"-002;
        TAB[ 13]:= +1.8315638888734"-002;
        TAB[ 14]:= +9.1578194443672"-002;
        TAB[ 15]:= +2.3810330555355"-001;
        TAB[ 16]:= +4.3347012036671"-001;
        TAB[ 17]:= +6.2883693517987"-001;
        TAB[ 18]:= +7.8513038703040"-001;
        TAB[ 19]:= +8.8932602159742"-001;
        TAB[ 20]:= +9.4886638420715"-001;
        TAB[ 21]:= +9.7863656551200"-001;
        TAB[ 22]:= +9.9186775720306"-001;
        TAB[ 23]:= +9.9716023387948"-001;
        TAB[ 24]:= +9.9908477085272"-001;
        TAB[ 25]:= +1.9245369732436"-003;
        S:= S + POISS TAB (4,12,25,17); MU := MU - 4
    "END";
    "IF" MU >= 2 "THEN"
    "BEGIN"
        TAB[  1]:= +1.3533528323661"-001;
        TAB[  2]:= +1.3533528323661"-001;
        TAB[  3]:= +4.0600584970984"-001;
        TAB[  4]:= +6.7667641618306"-001;
        TAB[  5]:= +8.5712346049856"-001;
        TAB[  6]:= +9.4734698265630"-001;
        TAB[  7]:= +9.8343639151939"-001;
        TAB[  8]:= +9.9546619447376"-001;
        TAB[  9]:= +9.9890328103215"-001;
        TAB[ 10]:= +9.9976255267175"-001;
        TAB[ 11]:= +8.5927163959755"-004;
        S:= S + POISS TAB (2,1,11,4); MU := MU - 2
    "END";
    "IF" MU > 0 "THEN" S:= S + POISS MUL ( MU,U );
FINISH:  RANDOM POIS TAB := S
```

```
"END" RANDOM POIS TAB;
          "EOP"
```

TITLE: **Random Poistabsam**

AUTHOR: B.F. Schriever

INSTITUTE: Mathematical Centre

RECEIVED: 800414

BRIEF DESCRIPTION

The procedure generates a sample of size K from a Poisson distribution with expectation MU<256, using a table search method (see general part of section 5.2).

KEYWORDS

Random Poisson sample by table search method.

CALLING SEQUENCE

Heading

```
"PROCEDURE" RANDOM POISTABSAM (MU, K, SAMPLE, U);
"VALUE" MU, K;
"INTEGER" K;
"INTEGER" "ARRAY" SAMPLE;
"REAL" MU, U;
"CODE" 41327;
```

Formal parameters

MU: <arithmetic expression>, expectation of the Poisson distribution;

K: <integer arithmetic expression>, length of the vector SAMPLE[1:K] and size of the sample;

SAMPLE: <integer array identifier>, output parameter, integer array of dimension [1:K], which at exit contains the sample;

U: <real variable>, value used to generate a random number. After a call of RANDOM POISTABSAM, U contains a new value which is used in the next call of RANDOM POISTABSAM unless the program assigns a new value to U.

Conditions: 0<MU<256, K integer >0, and 0≤U<1.

DATA AND RESULTS

After a procedure call, SAMPLE[1],...,SAMPLE[K] contain the generated sample. The call TEST RANDOM(41327, MU, K, U, 0, 0, 0) may yield the following error messages:

Errornumber 1	(if MU≤0 or MU≥256)
Errornumber 2	(if K is not an integer >0)
Errornumber 3	(if U<0 or U≥1)

Procedures used

ASELECT STATAL 41308

Language

Algol 60

Method and performance

Each drawing SAMPLE[I] is generated according to the table search method used in section 5.2.7.2.

Example of use

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    "INTEGER" "ARRAY" SAMPLE[1 : 10];
    U:= .1986;
    "FOR" I:= 1 "STEP" 1 "UNTIL" 10 "DO"
    "BEGIN" RANDOM POISTABSAM(10, 10, SAMPLE, U);
        OUTPUT(61, "("10(ZD2B),/")", SAMPLE)
    "END"
"END"
```

Output:

```
12   5  11   9   8  12   9   9   4  10
17  13   9   4  10  10   6   8  13  16
 8  13  13  11  14  12   3   9   7  13
11  11   7  13   8   2  12  11   6  11
 9  12   9  10  10  11   8  10   7   8
11  17  12   8   7  10   2  14  14   8
 6  13   8   7   8  15  10  10  14  16
10   9  13  14  11   9  14  10   7  11
14   8  11  13  11   8  11  15   9   8
 9   8  10   4  14   9   5  12   6   8
```

Source text

```
"CODE" 41327;
"PROCEDURE" RANDOM POISTABSAM(MU, SAMPLESIZE, S, U);
"VALUE" MU, SAMPLESIZE;
"INTEGER" "ARRAY" S;
"REAL" MU, U;
"INTEGER" SAMPLESIZE;
"BEGIN"
    "ARRAY" TAB[1 : 235]; "INTEGER" K;

    "PROCEDURE" SMPL POISS MUL(MU, SAMPLESIZE, S, U);
    "INTEGER" "ARRAY" S; "INTEGER" SAMPLESIZE; "REAL" MU, U;
    "BEGIN" "INTEGER" K, N; "REAL" P;
        "FOR" K:= 1 "STEP" 1 "UNTIL" SAMPLESIZE "DO"
        "BEGIN" N:= -2; P:= EXP(MU);
        "FOR" N:= N + 1 "WHILE" P > 1 "DO"
            P:= P * ASELECT(U);
        S[K]:= S[K] + N
    "END";
"END" SMPL POISS MUL;

"PROCEDURE" SMPL POISS TAB
(MU, SAMPLESIZE, S, FIRST, LAST, INDEXMU);
"INTEGER" "ARRAY" S;
"INTEGER" MU, SAMPLESIZE, FIRST, LAST, INDEXMU;
"BEGIN" "INTEGER" N, K; "REAL" V, P, CUMPROB;
    "FOR" K:= 1 "STEP" 1 "UNTIL" SAMPLESIZE "DO"
    "BEGIN" V:= ASELECT(U);
        "IF" V <= TAB[INDEXMU - 1] "THEN"
        "BEGIN" N:= INDEXMU;
            "FOR" N:= N - 1 "WHILE" V <= TAB[N - 1]
                            "AND" N > FIRST + 1 "DO";
            "IF" N > FIRST + 1 "THEN"
                S[K]:= S[K] + MU - INDEXMU + N
            "ELSE"
            "BEGIN" P:= TAB[FIRST];
                CUMPROB:= TAB[FIRST + 1] - P;
                N:= MU - INDEXMU + FIRST + 2;
                "FOR" N:=N - 1 "WHILE" V <= CUMPROB
                "AND" N > 0 "DO"
                "BEGIN" P:= P * N / MU;
                    CUMPROB:= CUMPROB - P
                "END";
                S[K]:= S[K] + N
            "END"
        "END"
        "ELSE"
        "BEGIN" N:= INDEXMU - 1;
            "FOR" N:= N + 1 "WHILE" V > TAB[N]
            "AND" N < LAST "DO";
            "IF" N < LAST "THEN"
                S[K]:= S[K] + MU - INDEXMU + N
            "ELSE"
            "BEGIN" P:= TAB[LAST];
                CUMPROB:= TAB[FIRST + 1];
                N:= MU - INDEXMU + LAST - 2;
                "FOR" N:= N + 1 "WHILE" V > CUMPROB
                "AND" N < 500 "DO"
                "BEGIN" P:= MU * P / (N + 1);
                    CUMPROB:= CUMPROB + P
                "END";
            S[K]:= S[K] + N
            "END"
        "END"
    "END"
"END" SMPL POISS TAB;

"FOR" K:= 1 "STEP" 1 "UNTIL" SAMPLESIZE "DO" S[K]:= 0;
"IF" MU >= 128 "THEN"
```

```
"BEGIN"
    TAB[164]:= +3.8120098644196"-004;
    TAB[165]:= +1.3778499806580"-003;
    TAB[166]:= +1.8861179625800"-003;
    TAB[167]:= +2.5568221036718"-003;
    TAB[168]:= +3.4328438389742"-003;
    TAB[169]:= +4.5654780017880"-003;
    TAB[170]:= +6.0152497301878"-003;
    TAB[171]:= +7.8525841978626"-003;
    TAB[172]:= +1.0158258823963"-002;
    TAB[173]:= +1.3023563213680"-002;
    TAB[174]:= +1.6550091693334"-002;
    TAB[175]:= +2.0849097839967"-002;
    TAB[176]:= +2.6040350545319"-002;
    TAB[177]:= +3.2250447239579"-002;
    TAB[178]:= +3.9610561840180"-002;
    TAB[179]:= +4.8253632196854"-002;
    TAB[180]:= +5.8311023157355"-002;
    TAB[181]:= +6.9908735255956"-002;
    TAB[182]:= +8.3163263368645"-002;
    TAB[183]:= +9.8177242115765"-002;
    TAB[184]:= +1.1503504281428"-001;
    TAB[185]:= +1.3379850793956"-001;
    TAB[186]:= +1.5450302118123"-001;
    TAB[187]:= +1.7715411259094"-001;
    TAB[188]:= +2.0172478801841"-001;
    TAB[189]:= +2.2815374982277"-001;
    TAB[190]:= +2.5634464241404"-001;
    TAB[191]:= +2.8616641308912"-001;
    TAB[192]:= +3.1745482822364"-001;
    TAB[193]:= +3.5001513015227"-001;
    TAB[194]:= +3.8362576440121"-001;
    TAB[195]:= +4.1804305387204"-001;
    TAB[196]:= +4.5300664952492"-001;
    TAB[197]:= +4.8824554908055"-001;
    TAB[198]:= +5.2348444863617"-001;
    TAB[199]:= +5.5845017842780"-001;
    TAB[200]:= +5.9287797391494"-001;
    TAB[201]:= +6.2651734660467"-001;
    TAB[202]:= +6.5913734436441"-001;
    TAB[203]:= +6.9053102641889"-001;
    TAB[204]:= +7.2051902121720"-001;
    TAB[205]:= +7.4895208295190"-001;
    TAB[206]:= +7.7571261164337"-001;
    TAB[207]:= +8.0071514939891"-001;
    TAB[208]:= +8.2390590905622"-001;
    TAB[209]:= +8.4526142874066"-001;
    TAB[210]:= +8.6478647530928"-001;
    TAB[211]:= +8.8251134027939"-001;
    TAB[212]:= +8.9848868335102"-001;
    TAB[213]:= +9.1279008134522"-001;
    TAB[214]:= +9.2550243511782"-001;
    TAB[215]:= +9.3672437499986"-001;
    TAB[216]:= +9.4656278804712"-001;
```

```
        TAB[217]:= +9.5512956947603"-001;
        TAB[218]:= +9.6253867773887"-001;
        TAB[219]:= +9.6890354926668"-001;
        TAB[220]:= +9.7433490630374"-001;
        TAB[221]:= +9.7893897054709"-001;
        TAB[222]:= +9.8281607727832"-001;
        TAB[223]:= +9.8605966983778"-001;
        TAB[224]:= +9.8875564287422"-001;
        TAB[225]:= +9.9098199480108"-001;
        TAB[226]:= +9.9280874510004"-001;
        TAB[227]:= +9.9429807018454"-001;
        TAB[228]:= +9.9550461202516"-001;
        TAB[229]:= +9.9647591614842"-001;
        TAB[230]:= +9.9725295944702"-001;
        TAB[231]:= +9.9787073300120"-001;
        TAB[232]:= +9.9835885037734"-001;
        TAB[233]:= +9.9874215727394"-001;
        TAB[234]:= +9.9904132363226"-001;
        TAB[235]:= +2.9916635832062"-004;
        SMPL POISS TAB (128,SAMPLESIZE,S,164,235,198);
        MU:= MU - 128
    "END";
    "IF" MU >= 64 "THEN"
    "BEGIN"
        TAB[112]:= +5.4212965801156"-004;
        TAB[113]:= +1.4170798450894"-003;
        TAB[114]:= +2.2431821811069"-003;
        TAB[115]:= +3.4727298440162"-003;
        TAB[116]:= +5.2611628082474"-003;
        TAB[117]:= +7.8047119129324"-003;
        TAB[118]:= +1.1343562841191"-002;
        TAB[119]:= +1.6162423679673"-002;
        TAB[120]:= +2.2587571464318"-002;
        TAB[121]:= +3.0979601223857"-002;
        TAB[122]:= +4.1721399316074"-002;
        TAB[123]:= +5.5201302804350"-002;
        TAB[124]:= +7.1791953251461"-002;
        TAB[125]:= +9.1825946244196"-002;
        TAB[126]:= +1.1556993793930"-001;
        TAB[127]:= +1.4319931009360"-001;
        TAB[128]:= +1.7477573541280"-001;
        TAB[129]:= +2.1022996735015"-001;
        TAB[130]:= +2.4935187845342"-001;
        TAB[131]:= +2.9178920575189"-001;
        TAB[132]:= +3.3705568820359"-001;
        TAB[133]:= +3.8454839110379"-001;
        TAB[134]:= +4.3357311667811"-001;
        TAB[135]:= +4.8337601249961"-001;
        TAB[136]:= +5.3317890832110"-001;
        TAB[137]:= +5.8221560574534"-001;
        TAB[138]:= +6.2976634264158"-001;
        TAB[139]:= +6.7518794206484"-001;
        TAB[140]:= +7.1793768269851"-001;
        TAB[141]:= +7.5758961603988"-001;
    TAB[142]:= +7.9384281223770"-001;
    TAB[143]:= +8.2652174965546"-001;
    TAB[144]:= +8.5556969402680"-001;
    TAB[145]:= +8.8103638498250"-001;
    TAB[146]:= +9.0306163121444"-001;
    TAB[147]:= +9.2185650799904"-001;
    TAB[148]:= +9.3768377265976"-001;
    TAB[149]:= +9.5083890172840"-001;
    TAB[150]:= +9.6163285378473"-001;
    TAB[151]:= +9.7037732127340"-001;
    TAB[152]:= +9.7737289526433"-001;
    TAB[153]:= +9.8290026236828"-001;
    TAB[154]:= +9.8721430498600"-001;
    TAB[155]:= +9.9054079567918"-001;
    TAB[156]:= +9.9307526477874"-001;
    TAB[157]:= +9.9498357092430"-001;
    TAB[158]:= +9.9640370573030"-001;
    TAB[159]:= +9.9744840259906"-001;
    TAB[160]:= +9.9820818214000"-001;
    TAB[161]:= +9.9875454046156"-001;
    TAB[162]:= +9.9914306193467"-001;
    TAB[163]:= +3.8852147311420"-004;
    SMPL POISS TAB (64,SAMPLESIZE,S,112,163,136);
    MU:= MU - 64
"END";
"IF" MU >= 32 "THEN"
"BEGIN"
    TAB[ 74]:= +7.3173973442918"-004;
    TAB[ 75]:= +1.3916672870287"-003;
    TAB[ 76]:= +2.7690597283076"-003;
    TAB[ 77]:= +5.2177574016927"-003;
    TAB[ 78]:= +9.3418797989703"-003;
    TAB[ 79]:= +1.5940475634614"-002;
    TAB[ 80]:= +2.5995478812736"-002;
    TAB[ 81]:= +4.0620937980915"-002;
    TAB[ 82]:= +6.0969402910554"-002;
    TAB[ 83]:= +8.8100689483409"-002;
    TAB[ 84]:= +1.2282873629667"-001;
    TAB[ 85]:= +1.6557094775913"-001;
    TAB[ 86]:= +2.1622838356649"-001;
    TAB[ 87]:= +2.7412259591775"-001;
    TAB[ 88]:= +3.3800586471912"-001;
    TAB[ 89]:= +4.0614801810724"-001;
    TAB[ 90]:= +4.7648830547560"-001;
    TAB[ 91]:= +5.4682859284397"-001;
    TAB[ 92]:= +6.1503735635269"-001;
    TAB[ 93]:= +6.7923383965502"-001;
    TAB[ 94]:= +7.3792776724572"-001;
    TAB[ 95]:= +7.9010014732632"-001;
    TAB[ 96]:= +8.3522220577444"-001;
    TAB[ 97]:= +8.7321972867810"-001;
    TAB[ 98]:= +9.0439718336828"-001;
    TAB[ 99]:= +9.2933914712043"-001;
    TAB[100]:= +9.4880604565870"-001;
```

```
        TAB[101]:= +9.6363796835452"-001;
        TAB[102]:= +9.7467567826769"-001;
        TAB[103]:= +9.8270310365908"-001;
        TAB[104]:= +9.8841149504852"-001;
        TAB[105]:= +9.9238254992812"-001;
        TAB[106]:= +9.9508624686743"-001;
        TAB[107]:= +9.9688871149363"-001;
        TAB[108]:= +9.9806583124952"-001;
        TAB[109]:= +9.9881918789330"-001;
        TAB[110]:= +9.9929188225802"-001;
        TAB[111]:= +4.7269436471777"-004;
        SMPL POISS TAB (32,SAMPLESIZE,S,74,111,91);
        MU:= MU - 32
    "END";
    "IF" MU >= 16 "THEN"
    "BEGIN"
        TAB[ 46]:= +9.8334736138684"-004;
        TAB[ 47]:= +1.3837850247629"-003;
        TAB[ 48]:= +4.0060446551275"-003;
        TAB[ 49]:= +9.9997809531040"-003;
        TAB[ 50]:= +2.1987253549056"-002;
        TAB[ 51]:= +4.3298315941861"-002;
        TAB[ 52]:= +7.7396015770348"-002;
        TAB[ 53]:= +1.2699267006634"-001;
        TAB[ 54]:= +1.9312154246098"-001;
        TAB[ 55]:= +2.7451092386977"-001;
        TAB[ 56]:= +3.6752735976553"-001;
        TAB[ 57]:= +4.6674489138768"-001;
        TAB[ 58]:= +5.6596242300982"-001;
        TAB[ 59]:= +6.5934362924244"-001;
        TAB[ 60]:= +7.4234914589364"-001;
        TAB[ 61]:= +8.1224852833677"-001;
        TAB[ 62]:= +8.6816803429126"-001;
        TAB[ 63]:= +9.1077337216136"-001;
        TAB[ 64]:= +9.4175907243052"-001;
        TAB[ 65]:= +9.6331434218297"-001;
        TAB[ 66]:= +9.7768452201794"-001;
        TAB[ 67]:= +9.8688143711232"-001;
        TAB[ 68]:= +9.9254107717040"-001;
        TAB[ 69]:= +9.9589493794557"-001;
        TAB[ 70]:= +9.9781142981709"-001;
        TAB[ 71]:= +9.9886880464275"-001;
        TAB[ 72]:= +9.9943273788310"-001;
        TAB[ 73]:= +5.6393324035520"-004;
        SMPL POISS TAB (16,SAMPLESIZE,S,46,73,58) ;
        MU:= MU - 16
    "END";
    "IF" MU >= 8 "THEN"
    "BEGIN"
        TAB[ 26]:= +2.6837010232201"-003;
        TAB[ 27]:= +3.0191636511226"-003;
        TAB[ 28]:= +1.3753967744003"-002;
        TAB[ 29]:= +4.2380111991685"-002;
        TAB[ 30]:= +9.9632400487048"-002;
        TAB[ 31]:= +1.9123606207963"-001;
        TAB[ 32]:= +3.1337427753641"-001;
        TAB[ 33]:= +4.5296080948703"-001;
        TAB[ 34]:= +5.9254734143764"-001;
        TAB[ 35]:= +7.1662425872706"-001;
        TAB[ 36]:= +8.1588579255860"-001;
        TAB[ 37]:= +8.8807599898154"-001;
        TAB[ 38]:= +9.3620280326350"-001;
        TAB[ 39]:= +9.6581929820625"-001;
        TAB[ 40]:= +9.8274300960210"-001;
        TAB[ 41]:= +9.9176898901322"-001;
        TAB[ 42]:= +9.9628197871878"-001;
        TAB[ 43]:= +9.9840573858022"-001;
        TAB[ 44]:= +9.9934963185198"-001;
        TAB[ 45]:= +9.4389327175133"-004;
        SMPL POISS TAB (8,SAMPLESIZE,S,26,45,34);
        MU:= MU - 8
    "END";
    "IF" MU >= 4 "THEN"
    "BEGIN"
        TAB[ 12]:= +1.8315638888734"-002;
        TAB[ 13]:= +1.8315638888734"-002;
        TAB[ 14]:= +9.1578194443672"-002;
        TAB[ 15]:= +2.3810330555355"-001;
        TAB[ 16]:= +4.3347012036671"-001;
        TAB[ 17]:= +6.2883693517987"-001;
        TAB[ 18]:= +7.8513038703040"-001;
        TAB[ 19]:= +8.8932602159742"-001;
        TAB[ 20]:= +9.4886638420715"-001;
        TAB[ 21]:= +9.7863656551200"-001;
        TAB[ 22]:= +9.9186775720306"-001;
        TAB[ 23]:= +9.9716023387948"-001;
        TAB[ 24]:= +9.9908477085272"-001;
        TAB[ 25]:= +1.9245369732436"-003;
        SMPL POISS TAB (4,SAMPLESIZE,S,12,25,17);
        MU:= MU - 4
    "END";
    "IF" MU >= 2 "THEN"
    "BEGIN"
        TAB[  1]:= +1.3533528323661"-001;
        TAB[  2]:= +1.3533528323661"-001;
        TAB[  3]:= +4.0600584970984"-001;
        TAB[  4]:= +6.7667641618306"-001;
        TAB[  5]:= +8.5712346049856"-001;
        TAB[  6]:= +9.4734698265630"-001;
        TAB[  7]:= +9.8343639151939"-001;
        TAB[  8]:= +9.9546619447376"-001;
        TAB[  9]:= +9.9890328103215"-001;
        TAB[ 10]:= +9.9976255267175"-001;
        TAB[ 11]:= +8.5927163959755"-004;
        SMPL POISS TAB (2,SAMPLESIZE,S,1,11,4); MU:= MU - 2
    "END";
    "IF" MU > 0 "THEN" SMPL POISS MUL ( MU,SAMPLESIZE,S,U );
"END" RANDOM POISTABSAM;
          "EOP"
```

TITLE: **Random Poissorsam**

AUTHOR: C. van Putten

INSTITUTE: Mathematical Centre

RECEIVED: 800401

BRIEF DESCRIPTION

The procedure generates a sorted sample of size K=UB−LB+1 from a Poisson distribution with expectation MU≤700 (see general part of section 5.2).

KEYWORDS

Sorted random Poisson sample

CALLING SEQUENCE

Heading

```
"PROCEDURE" RANDOM POISSORSAM (SAMPLE, LB, UB, MU, U);
"VALUE " LB, UB, MU;
"INTEGER" LB, UB;
"INTEGER" "ARRAY" SAMPLE;
"REAL" MU, U;
"CODE" 41329;
```

Formal parameters

MU: <arithmetic expression>, expectation of the Poisson distribution;

LB: <integer arithmetic expression>, smallest index of the array SAMPLE;

UB: <integer arithmetic expression>, largest index of the array SAMPLE;

SAMPLE: <integer array indentifier>, output parameter, integer array of dimension [LB:UB], which at exit contains the sample;

U: <real variable>, value used to generate a random number. After a call of RANDOM POISSORSAM, U contains a new value which is used in the next call of RANDOM POISSORSAM unless the program assigns a new value to U.

Conditions: 0<MU≤700, LB, UB integers, UB>LB>0, and 0≤U<1.

DATA AND RESULTS

After a procedure call, SAMPLE[LB] ≤SAMPLE[LB+1] ≤....≤ SAMPLE[UB] contain the sample.

The call TEST RANDOM(41329, LB, UB, MU, U, 0, 0) may yield the following error messages:

Errornumber 1 (if the array segment SAMPLE[LB],...,SAMPLE[UB] is not declared in the main program)

Errornumber 2 (if LB is not an integer >0)

Errornumber 3	(if UB is not an integer >LB)
Errornumber 4	(if MU≤0 or MU>700)
Errornumber 5	(if U <0 or U≥1)

Procedures used

ASELECT	STATAL 41308
OPEN SCRATCH	STATAL OSCR
CLOSE SCRATCH	STATAL CSCR

Language

Algol 60

Method and performance

An ordered sample from the Poisson distribution is generated using the method described in v. Putten & v.d. Tweel (1979). The procedure uses a scratch file.

Reference

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

Example of use

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    "INTEGER" "ARRAY" SAMPLE[1 : 10];
    U:= .1986;
    RANDOM POISSORSAM(SAMPLE, 1, 10, 3, U);
    OUTPUT(61,"("10(ZD,2B),/")", SAMPLE)
"END"
```

Output:

```
 0   1   2   2   2   3   3   3   4   7
```

Source text

```
"CODE" 41329;
"PROCEDURE" RANDOM POISSORSAM(SAMPLE, LB, UB, MU, U);
"VALUE" LB, UB, MU; "INTEGER" LB, UB;
"REAL" MU, U; "INTEGER" "ARRAY" SAMPLE;
"BEGIN"
    "INTEGER" ARG, SCR, ADDRESS, SAMPLESIZE, I, LBS, UBS;
    "REAL" LNMU, LNOBS, SUMLNARG, CUMPROBARG, LNCUMPROBARG;

    LBS:= LOWERBOUND(SAMPLE, 1);
    UBS:= UPPERBOUND(SAMPLE, 1);
    "IF" LBS > LB "THEN"
    STATAL ERROR("("RANDOM POISSORSAM")", 1, LBS);
    "IF" UBS < UB "THEN"
    STATAL ERROR("("RANDOM POISSORSAM")", 1, UBS);

    SAMPLESIZE:= UB - LB + 1;
    LNOBS:= LN(ASELECT(U)) / SAMPLESIZE; LNMU:= LN(MU);
    SCR:= OPEN SCRATCH("("SC41329")");
    CUMPROBARG:= 1; SUMLNARG:= 0; ARG:= 0; ADDRESS:= 1;
    LNCUMPROBARG:= -MU;
    "FOR" ARG:= ARG + 1 "WHILE" LNCUMPROBARG < LNOBS "DO"
    "BEGIN" STORE ITEM(SCR, ADDRESS, LNCUMPROBARG);
        SUMLNARG:= SUMLNARG + LN(ARG);
        CUMPROBARG:=CUMPROBARG + EXP(LNMU * ARG - SUMLNARG);
        LNCUMPROBARG:= LN(CUMPROBARG) - MU;
    "END";
    "IF" ARG = 1 "THEN"
    "BEGIN"
        "FOR" I:= LB "STEP" 1 "UNTIL" UB "DO" SAMPLE[I]:= 0;
        "GOTO" END PROC
    "END";
    "IF" SAMPLESIZE > ARG "THEN"
    "BEGIN" ADDRESS:= ADDRESS - 1; I:= UB;
        "FOR" ARG:= ARG - 1 "STEP" -1 "UNTIL" 1 "DO"
        "BEGIN" FETCH ITEM(SCR, ADDRESS, LNCUMPROBARG);
            ADDRESS:= ADDRESS - 2;
        NEXT OBS:
            "IF" LNOBS > LNCUMPROBARG "THEN"
            "BEGIN" SAMPLE[I]:= ARG; I:= I - 1;
                "IF" SAMPLESIZE = 1 "THEN" "GOTO" END PROC;
                SAMPLESIZE:= SAMPLESIZE - 1;
                LNOBS:= LNOBS + LN(ASELECT(U)) / SAMPLESIZE;
                "GOTO" NEXT OBS;
            "END";
        "END";
        "FOR" I:= I "STEP" -1 "UNTIL" LB "DO" SAMPLE[I]:= 0;
    "END" "ELSE"
    "BEGIN" SAMPLE[UB]:= ARG:= ARG - 1;
        ADDRESS:= ADDRESS - 1;
        FETCH ITEM(SCR, ADDRESS, LNCUMPROBARG);
        ADDRESS:= ADDRESS - 2;
        I:= UB - 1;
        "FOR" SAMPLESIZE:= SAMPLESIZE - 1
        "STEP" -1 "UNTIL" 1 "DO"
        "BEGIN" LNOBS:= LNOBS + LN(ASELECT(U)) / SAMPLESIZE;
    NEXT CLASS:
            "IF" LNOBS > LNCUMPROBARG "THEN"
            "BEGIN" SAMPLE[I]:= ARG; I:= I - 1 "END"
            "ELSE" "IF" ARG = 1 "THEN"
            "BEGIN" "FOR" I:= I "STEP" -1 "UNTIL" LB "DO"
                SAMPLE[I]:= 0;
                "GOTO" END PROC
            "END" "ELSE"
            "BEGIN" ARG:= ARG - 1;
                FETCH ITEM(SCR, ADDRESS, LNCUMPROBARG);
                ADDRESS:= ADDRESS - 2;
                "GOTO" NEXT CLASS;
            "END"
        "END";
    "END";
END PROC: RETURN(SCR);
"END" RANDOM POIS SORSAM;
         "EOP"
```

TITLE: **Random Poishisto**

AUTHOR: C. van Putten

INSTITUTE: Mathematical Centre

RECEIVED: 800401

BRIEF DESCRIPTION

The procedure generates a sorted random sample of size K from a Poisson distribution with expectation MU. It is presented in a histogram (see general part of section 5.2.).

KEYWORDS

Sorted random Poisson sample in histogram

CALLING SEQUENCE

Heading

```
"PROCEDURE" RANDOM POISHISTO (SAMPLE, LB, UB, K, MU, U);
"VALUE" LB, UB, K, MU;
"INTEGER" LB, UB, K;
"INTEGER" "ARRAY" SAMPLE;
"REAL" MU, U;
"CODE" 41328;
```

Formal parameters

MU: <arithmetic expression>, expectation of the Poisson distribution;

LB: <integer arithmetic expression>, smallest index of the array SAMPLE;

UB: <integer arithmetic expression>, largest index of the array SAMPLE;

K: <integer arithmetic expression>, size of the sample;

SAMPLE: < integer array identifier> output parameter, integer array of dimension [LB:UB], which at exit contains the histogram;

U: <real variable>, value used to generate a random number. After a call of RANDOM POISHISTO, U contains a new value which is used in the next call of RANDOM POISHISTO unless the program assigns a new value to U.

Conditions: 0<MU≤700, LB, UB, K integers >0,UB> LB, and 0≤U<1.

DATA AND RESULTS

After a procedure call, SAMPLE [I] contains the number of times the value I is observed in the sample, I=LB+1,...,UB−1. SAMPLE[LB] contains the number of times a value ≤LB is observed, and SAMPLE[UB] contains the number of times a value ≥UB is observed.

The call TEST RANDOM(41328, LB, UB, K, MU, U, 0) may yield the following error messages:

Errornumber 2	(if LB is not an integer >0)
Errornumber 3	(if UB is not an integer >LB)
Errornumber 4	(if K is not an integer >0)
Errornumber 5	(if MU<0 or MU>700)
Errornumber 6	(if U<0 or U>1)

Procedures used

ASELECT	STATAL 41308

Language

Algol 60

Method and performance

See v. Putten & v.d. Tweel (1979).

Reference

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

Example of use

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    "INTEGER" "ARRAY" HIST1, HIST2[0 : 9];
    "FOR" I:=0 "STEP" 1 "UNTIL" 9 "DO"
    "BEGIN" HIST1[I]:= 0; HIST2[I]:= 0 "END";
    U:= .1986;
    RANDOM POISHISTO(HIST1, 0, 9, 100, 3, U);
    U:= .1986;
    RANDOM POISHISTO(HIST2, 0, 6, 100, 3, U);
    "FOR" I:=0 "STEP" 1 "UNTIL" 9 "DO"
    OUTPUT(61,"("3(ZD,2B),/")", I, HIST1[I], HIST2[I])
"END"
```

Output:

```
0  4  4
1 17 17
2 21 21
3 25 25
4 16 16
5  9  9
6  7  8
7  0  0
8  0  0
9  1  0
```

Source text

```
"CODE" 41328;
"PROCEDURE"
RANDOM POIS HISTO(OBS, LB, UB, SAMPLE SIZE, MU, U);
"VALUE" LB, UB, SAMPLE SIZE, MU;
"INTEGER" LB, UB, SAMPLE SIZE;
"REAL" MU, U; "INTEGER" "ARRAY" OBS;
"BEGIN" "INTEGER" ARG, MAXARG, COUNTARG, LB1;
    "REAL" LNMU, LNOBS, SUMLNARG, CUMPROBARG, LNCUMPROBARG;
    "ARRAY" LNCUMPROB[LB + 1 : UB - 1];

    LNOBS:= LN(ASELECT(U)) / SAMPLE SIZE; LNMU:= LN(MU);
    CUMPROBARG:= 1; SUMLNARG:= 0;
    "FOR" ARG:= 1 "STEP" 1 "UNTIL" LB "DO"
    "BEGIN" SUMLNARG:= SUMLNARG + LN(ARG);
        CUMPROBARG:= CUMPROBARG + EXP(LNMU * ARG - SUMLNARG)
    "END";
    ARG:= LB; LNCUMPROBARG:= LN(CUMPROBARG) - MU;
    "IF" LNCUMPROBARG >= LNOBS "THEN"
    "BEGIN" OBS[LB]:= SAMPLE SIZE; MAXARG:= LB;
        "GOTO" END PROC
    "END";
    "FOR" ARG:= ARG + 1
    "WHILE" ARG < UB "AND" LNCUMPROBARG < LNOBS "DO"
    "BEGIN" LNCUMPROB[ARG]:= LNCUMPROBARG;
        SUMLNARG:= SUMLNARG + LN(ARG);
        CUMPROBARG:=CUMPROBARG + EXP(LNMU * ARG - SUMLNARG);
        LNCUMPROBARG:= LN(CUMPROBARG) - MU
    "END";
    "IF" LNCUMPROBARG >= LNOBS "THEN"
    "BEGIN" ARG:= ARG - 1;
        LNCUMPROBARG:= LNCUMPROB[ARG]
    "END";
    MAXARG:= ARG;
    "IF" SAMPLE SIZE <= ARG - LB "THEN"
    "BEGIN" COUNTARG:= 1;
        "FOR" SAMPLE SIZE:= SAMPLE SIZE - 1
        "STEP" -1 "UNTIL" 1 "DO"
        "BEGIN" LNOBS:=LNOBS + LN(ASELECT(U)) / SAMPLE SIZE;
    NEXT CLASS:
            "IF" LNOBS > LNCUMPROBARG "THEN"
                COUNTARG:= COUNTARG + 1
            "ELSE" "IF" ARG = LB + 1 "THEN"
            "BEGIN" OBS[ARG]:= COUNTARG; ARG:= LB;
                OBS[LB]:= SAMPLE SIZE; "GOTO" END PROC;
            "END" "ELSE"
            "BEGIN" OBS[ARG]:= COUNTARG; COUNTARG:= 0;
                ARG:= ARG - 1;
                LNCUMPROBARG:= LNCUMPROB[ARG];
                "GOTO" NEXT CLASS
            "END"
        "END";
        OBS[ARG]:= COUNTARG;
    "END" "ELSE"
    "BEGIN" "PROCEDURE" EXHAUST ARG;
        "BEGIN" COUNTARG:= 0;
        NEXT OBS: "IF" LNOBS > LNCUMPROBARG "THEN"
            "BEGIN" COUNTARG:= COUNTARG + 1;
                "IF" SAMPLE SIZE = 1 "THEN"
                "BEGIN" OBS[ARG]:= COUNTARG;
                    "GOTO" END PROC
                "END";
                SAMPLE SIZE:= SAMPLE SIZE - 1;
                LNOBS:=LNOBS + LN(ASELECT(U)) / SAMPLE SIZE;
                "GOTO" NEXT OBS;
            "END";
            OBS[ARG]:= COUNTARG;
        "END" EXHAUST ARG;

        "IF" ARG = UB "THEN" EXHAUST ARG;
        LB1:= LB + 1;
        "FOR" ARG:= ARG - 1 "STEP" -1 "UNTIL" LB1 "DO"
        "BEGIN" LNCUMPROBARG:= LNCUMPROB[ARG];
            EXHAUST ARG
        "END";
        OBS[LB]:= SAMPLE SIZE;
    "END";
END PROC:
    "FOR" ARG:= ARG - 1 "STEP" -1 "UNTIL" LB,
          MAXARG + 1 "STEP" 1 "UNTIL" UB
    "DO" OBS[ARG]:= 0;
"END" RANDOM POIS HISTO;
"EOP"
```

5.3 RANDOM NUMBERS FROM CONTINUOUS DISTRIBUTIONS

This section contains procedures for generating one single drawing from continuous distribution. For the multivariate normal distribution a procedure is given which generates an (unordered) sample of size K. In order to generate samples of other ditributions, repeated call's of procedures for one single drawing should be used. To generate ordered samples use the fact that an ordered sample from an exponential distribution has spacings (i.e. distances between successive elements) which are independent and exponentially distributed with known parameters, See v. Putten & v.d. Tweel (1979).

REFERENCE

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979.

TITLE: **Random Unif**

AUTHOR: J.G. Bethlehem

INSTITUTE: Mathematical Centre

RECEIVED: 750219

BRIEF DESCRIPTION

This procedure generates a random number, uniformly distributed on the interval (A, B].

KEYWORDS

Random uniform number

CALLING SEQUENCE

Heading

```
"REAL" "PROCEDURE" RANDOM UNIF(A, B, U);
"VALUE" A, B;
"REAL" A, B, U;
"CODE" 41302;
```

Formal parameters

A: <arithmetic expression>, lower bound of the interval;

B: <arithmetic expression>, upper bound of the interval;

U: <real variable>, value used to generate a random number.
After a call of RANDOM UNIF, U contains a new value which is used in the next call of RANDOM UNIF unless the program assigns a new value to U.

Conditions: A<B and 0⩽U<1.

DATA AND RESULTS

The generated value is assigned to the procedure identifier RANDOM UNIF.
The call TEST RANDOM(41302, A, B, U, 0, 0, 0) may yield the following error messages:

Errornumber 2	(if B⩽A)
Errornumber 3	(if U<0 or U⩾1)

PROCEDURES USED

ASELECT STATAL 41308

LANGUAGE

Algol 60

Method and performance

The generated value is equal to D * (B−A)+A, where D is a random number which is uniformly distributed on the interval (0, 1]. See v. Putten & v.d. Tweel (1979).

Reference

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

Example of use

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    U:= .1986;
    "FOR" I:=1 "STEP" 1 "UNTIL" 10 "DO"
    OUTPUT(61, "("ZD.6D,/")", RANDOM UNIF(3, 13, U))
"END"
```

Output:

```
11.355616
 3.343306
 9.574724
 6.769831
 5.801691
 9.724809
 8.520579
 7.073558
 3.952506
 6.353785
```

Source text

```
"CODE" 41302;
"REAL" "PROCEDURE" RANDOM UNIF(A, B, U); "VALUE" A, B;
"REAL" A, B, U;
RANDOM UNIF:= ASELECT(U) * (B - A) + A;
        "EOP"
```

TITLE: **Random Norm**

AUTHOR: J.G. Bethlehem

INSTITUTE: Mathematical Centre

RECEIVED: 750219

BRIEF DESCRIPTION

The procedure generates a random number having a normal distribution with mean MU and standard deviation SIGMA.

KEYWORDS

Random normal number

CALLING SEQUENCE

Heading

```
"REAL" "PROCEDURE" RANDOM NORM (MU, SIGMA, U);
"VALUE" MU, SIGMA;
"REAL" MU, SIGMA, U;
"CODE" 41303;
```

Formal parameters

MU: <arithmetic expression>, mean of the normal distribution;

SIGMA: <arithmetic expression>, standard deviation of the normal distribution;

U: <real variable>, value used to generate a random number. After a call of RANDOM NORM, U contains a new value which is used in the next call of RANDOM NORM unless the program assigns a new value to U.

Conditions: SIGMA >0 and 0⩽U<1.

DATA AND RESULTS

The generated value is assigned to the procedure identifier RANDOM NORM.

The call TEST RANDOM(41303, MU, SIGMA, U,0, 0, 0) may yield the following error messages:

Errornumber 2	(if SIGMA ⩽0)
Errornumber 3	(if U<0 or U⩾1)

PROCEDURES USED

ASELECT	STATAL 41308
PHINV	STATAL 41501

Language
Algol 60

Method and performance
The generated value is equal to MU+SIGMA * PHINV (D), where PHINV is the inverse of the standard normal cumulative distribution function and D is a random number which is uniformly distributed on the interval (0,1).

Example of use

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    U:= .1986;
    "FOR" I:=1 "STEP" 1 "UNTIL" 10 "DO"
    OUTPUT(61, "("+ZD.6D,/")", RANDOM NORM(10, 3, U))
"END"
```

Output:

```
+12.929136
 +4.538073
+11.216724
 +9.059758
 +8.252982
+11.340322
+10.392587
 +9.296944
 +6.072707
 +8.724672
```

Source text

```
"CODE" 41303;
"REAL" "PROCEDURE" RANDOM NORM(MU, SIGMA, U);
"VALUE" MU, SIGMA; "REAL" MU, SIGMA, U;
"BEGIN" "REAL" R;

    R:= ASELECT(U); "IF" R = 1 "THEN" R:= ASELECT(U);
    RANDOM NORM:= MU + SIGMA * PHINV(R);
"END" RANDOM NORM;
          "EOP"
```

TITLE: **Random Binorm**

AUTHOR: E. Opperdoes

INSTITUTE: Mathematical Centre

RECEIVED: 760901

BRIEF DESCRIPTION

The procedure generates two random numbers having normal distributions with mean and standard deviation (MU1, SIGMA1) and (MU2, SIGMA2), respectively, and with correlation RHO.

KEYWORDS

Random bivariate normal numbers

CALLING SEQUENCE

Heading

```
"PROCEDURE" RANDOM BINORM (X1, X2, MU1, MU2, SIGMA1, SIGMA2, RHO, U);
"VALUE" MU1, MU2, SIGMA1, SIGMA2, RHO;
"REAL" MU1, MU2, SIGMA1, SIGMA2, RHO, U;
"CODE" 41333;
```

Formal parameters

X1: <real variable>, output parameter, real variable which at exit contains the first random number;

X2: <real variable>, output parameter, real variable which at exit contains the second random number;

MU1: <arithmetic expression>, mean of the normal distribution of the first random number;

MU2: <arithmetic expression>, mean of the normal distribution of the second random number;

SIGMA1: <arithmetic expression>, standard deviation of the distribution of the first random number;

SIGMA2: <arithmetic expression>, standard deviation of the distribution of the second random number;

RHO: <arithmetic expression>, correlation of the distributions of the two random numbers;

U: <real variable>, value used to generate a random number. After a call of RANDOM BINORM, U contains a new value which is used in the next call of RANDOM BINORM unless the program assigns a new value to U.

Conditions: SIGMA1, SIGMA2 >0, $-1 \leq$ RHO ≤ 1, and $0 \leq$ U < 1.

Data and results

After a procedure call, X1 and X2 contain the two random numbers. The call TEST RANDOM(41333, MU1, MU2, SIGMA1, SIGMA2, RHO, U) may yield the following error messages:

Errornumber 5	(if SIGMA1 $\leqslant$0)
Errornumber 6	(if SIGMA2 $\leqslant$0)
Errornumber 7	(if RHO <−1 or RHO>1)
Errornumber 8	(if U<0 or U$\geqslant$1)

Procedures used

ASELECT   STATAL 41308

Language

Algol 60

Method and performance

The random numbers X1 and X2 are generated by X1=MU1+SIGMA1 * V1 and X2=MU2+SIGMA2*(RHO+V1+SQRT(1−RHO^{2})*V2), where V1 and V2 are two independent standard normal distributed random numbers. These are generated by V1=SQRT(−2*LN(D))* COS(2*π*D) and V2=SQRT (−2*LN(D))* SIN(2*π*D),
where D is a random number uniformly distributed on (0, 1].
See v. Putten & v.d. Tweel (1979).

Reference

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

Example of use

Program:

```
"BEGIN" "INTEGER" I; "REAL" U, X1, X2;
   U:= .1986;
   "FOR" I:= 1 "STEP" 1 "UNTIL" 10 "DO"
   "BEGIN" RANDOM BINORM(X1, X2, 0, 10, 1, 1, 1, U);
           OUTPUT(61,"("2(-ZD.6D,2B),/")", X1, X2)
   "END"
"END"
```

Output:

```
 -2.230507    7.769493
 -1.167340    8.832660
  0.874865   10.874865
 -0.430590    9.569410
  0.832780   10.832780
  1.194067   11.194067
  0.402234   10.402234
  0.304066   10.304066
  1.159442   11.159442
  0.193551   10.193551
```

Source text

```
"CODE" 41333;
"PROCEDURE"
RANDOM BINORM(X1, X2, MU1, MU2, SIGMA1, SIGMA2, RHO, U);
"VALUE" MU1, MU2, SIGMA1, SIGMA2, RHO;
"REAL" X1, X2, MU1, MU2, SIGMA1, SIGMA2, RHO, U;
"BEGIN" "REAL" ANGLE, RADIUS, V1, V2;

    ANGLE:= 6.2831853071796 * ASELECT(U);
    RADIUS:= SQRT(-2 * LN(ASELECT(U)));
    V1:= RADIUS * SIN(ANGLE); V2:= RADIUS * COS(ANGLE);
    X1:= MU1 + SIGMA1 * V1;
    X2:= MU2 + SIGMA2 * (RHO * V1 + SQRT(1 - RHO * RHO)
        * V2);
"END" RANDOM BINORM;
        "EOP"
```

TITLE: **Random Multinorm**

AUTHOR: E. Opperdoes

INSTITUTE: Mathematical Centre

RECEIVED: 760901

BRIEF DESCRIPTION
The procedure generates a random sample of size K=UB−LB+1 from a multivariate normal distribution with given mean vector MU and covariance matrix SIGMA.

KEYWORDS
Random multivariate normal sample

CALLING SEQUENCE
Heading

```
"PROCEDURE" RANDOM MULTINORM (SAMPLE, LB, UB, N, MU, SIGMA, DECOMPOSED, U);
"VALUE" LB, UB, N, MU, DECOMPOSED;
"INTEGER" LB, UB, N;
"BOOLEAN" DECOMPOSED;
"ARRAY" SAMPLE, MU, SIGMA;
"REAL" U;
"CODE" 41309;
```

Formal parameters

SAMPLE: <array identifier>, output parameter, array of dimension [LB:UB, 1:N] which at exit contains the sample;

LB: <integer arithmetic expression>, smallest index of the array SAMPLE;

UB: <integer arithmetic expression>, largest index of the array SAMPLE;

N: <integer arithmetic expression>, dimension of the multivariate normal distribution, upper bound of the second index of SAMPLE;

MU: <array identifier>, containing the mean vector of the multivariate normal distribution in MU[1],...,MU[N];

SIGMA: <array identifier>, vector containing either the lower triangular part (diagonal included) of the covariance matrix, row by row, in SIGMA[1], SIGMA[2],..., SIGMA[N *(N+1) / 2], or a lower triangular matrix, row by row, which is the result of a Choleski decomposition of the covariance matrix;

DECOMPOSED: <boolean expression>, if SIGMA contains the covariance matrix, then the value of DECOMPOSED should be "FALSE".
If SIGMA contains the result of a Choleski decomposition of the covariance marix then the value of DECOMPOSED should be "TRUE";

U: <real variable>, value used to generate a random number. After a call of RANDOM MULTINORM, U contains a new value which is used in the next call of RANDOM MULTINORM unless the program assigns a new value to U.

Conditions: LB, UB and N integers, LB ≤ UB, N≥1, covariance matrix positive definite (if DECOMPOSED = "FALSE").

DATA AND RESULTS

The generated sample is assigned to the elements of array SAMPLE, each row SAMPLE[I, 1],...,SAMPLE[I, N] contains one drawing of the multivariate normal distribution, I=LB,...,UB.

The call TEST RANDOM(41309, LB, UB, N, U, 0, 0) may yield the following error messages:

Errornumber 2	(if LB is not an integer)
Errornumber 3	(if UB is not an integer ≥LB)
Errornumber 4	(if N is not an integer >0)
Errornumber 8	(if U<0 or U≥1)

In a call of RANDOM MULTINORM with DECOMPOSED = "FALSE" the following error message may appear:

Errornumber 6	(if the covariance matrix is not positive definite)

The precision of the Choleski decomposition which is performed when DECOMPOSED is "FALSE" is 10^{-14}.

PROCEDURES USED

VECVEC	NUMAL 34010
CHLDEC1	NUMAL 34311
STATAL3 ERROR	STATAL 40100
ASELECT	STATAL 41308
PHINV	STATAL 41501

LANGUAGE

Algol 60

METHOD AND PERFORMANCE

If DECOMPOSED ="FALSE", a Choleski decomposition is performed on the covariance matrix SIGMA. This produces a lower triangular matrix Q such that SIGMA=QQ'.

If DECOMPOSED ="TRUE", SIGMA should contain such a matrix which multiplied with its transpose yields the convariance matrix of the normal distribution.

For each drawing of the multivariate normal distribution, a sample of size N is generated from a standard normal distribution. The drawing from the multivariate distribution is obtained by multiplying the vector with N i.i.d. normal samples with the matrix Q and adding this to the vector MU.

See v. Putten & v.d. Tweel (1979).

Reference

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

Example of use

Program:

```
"BEGIN" "REAL" "ARRAY" SAMPLE[1:10, 1:3], MEAN[1:3],
   COV[1:6]; "REAL" U;
   INARRAY(60, MEAN);
   INARRAY(60, COV);
   U:=.1986;

   RANDOM MULTINORM
   (SAMPLE, 1, 10, 3, MEAN, COV, "FALSE", U);
   OUTPUT(61, "("10(3(-2ZD.4D, 2B),/)")", SAMPLE)
"END"
```

Input:

```
 1.0   1.0  1.0
 1.0
 0.8   1.0
-0.2  -0.5  1.0
```

Output:

```
   1.1283      1.4540      0.9667
   2.3718      1.9395      1.2319
   1.5993      0.9331      0.3497
   2.3998      2.4048      0.3536
  -0.1202     -0.2981      1.4508
   1.6074      1.6458      0.6243
  -0.9869      0.2326      2.0387
  -0.0729     -0.0139     -0.6821
   0.7113     -0.0486      1.0381
  -0.5371     -0.2350      2.0529
```

Source text

```
"CODE" 41309;
"PROCEDURE" RANDOM MULTINORM(SAMPLE, LOW, UPP, DIMENSION,
            MEAN VECTOR, COV DEC, DECOMPOSED, U);
"VALUE" LOW, UPP, DIMENSION, MEAN VECTOR, DECOMPOSED;
"ARRAY" SAMPLE, MEAN VECTOR, COV DEC;
"INTEGER" LOW, UPP, DIMENSION;
"BOOLEAN" DECOMPOSED;
"REAL" U;
"BEGIN" "INTEGER" SAMPLING, SHIFT, I;
    "REAL" RADIUS, ANGLE, TWO PI, UNIF01;
    "BOOLEAN" DIMENSION ODD;
    "ARRAY" AUX[2 : 3], X[1 : DIMENSION];


    "IF" "NOT" DECOMPOSED "THEN"
    "BEGIN" AUX[2]:= "-14; CHLDEC1(COV DEC, DIMENSION, AUX);
        "IF" AUX[3] ^= DIMENSION "THEN"
        STATAL3 ERROR("("RANDOM MULTINORM")", 6, AUX[3])
    "END";

    "COMMENT" INITIALIZE CONSTANTS;
    TWO PI:= 6.28318 53071 796;
    DIMENSION ODD:= (DIMENSION // 2) * 2 ^= DIMENSION;

    "COMMENT" GENERATE THE SAMPLE;
    "FOR" SAMPLING:= LOW "STEP" 1 "UNTIL" UPP "DO"
    "BEGIN" "COMMENT" GENERATE STANDAARD-NORMAL SAMPLE;
        "FOR" I:= 2 "STEP" 2 "UNTIL" DIMENSION "DO"
        "BEGIN" RADIUS:= SQRT(-2 * LN(ASELECT(U)));
            ANGLE:= TWO PI * ASELECT(U);
            X[I - 1]:= RADIUS * SIN(ANGLE);
            X[I]:= RADIUS * COS(ANGLE)
        "END";
        "IF" DIMENSION ODD "THEN"
        "BEGIN" UNIF01:= ASELECT(U);
            "IF" UNIF01 = 1 "THEN" UNIF01:= ASELECT(U);
            X[DIMENSION]:= PHINV(UNIF01)
        "END";

        "COMMENT" COMPUTE THE REQUIRED MULTINORMAL SAMPLE;
        SHIFT:= 0;
        "FOR" I:= 1 "STEP" 1 "UNTIL" DIMENSION "DO"
        "BEGIN" SAMPLE[SAMPLING, I]:=
            VECVEC(1, I, SHIFT, X, COV DEC) +
                                       MEAN VECTOR[I];
            SHIFT:= SHIFT + I
        "END"
    "END" SAMPLING;
"END" RANDOM MULTINORM;
         "EOP"
```

Title: **Random Chisq**

Author: C. van Putten

Institute: Mathematical Centre

Received: 810101

Brief description

The procedure generates a random number having a χ^2-distribution with DF degrees of freedom.

Keywords

Random chi-square number

Calling sequence

Heading

```
"REAL" "PROCEDURE" RANDOM CHISQ (DF, U);
"VALUE" DF;
"REAL" U;
"INTEGER" DF;
"CODE" 41315;
```

Formal parameters

DF: <integer arithmetic expression>, degrees of freedom of the distribution.

U: <real variable>, value used to generate a random number. After a call of RANDOM CHISQ, U contains a new value which is used in the next call of RANDOM CHISQ unless the program assigns a new value to U.

Conditions: DF integer >0, and 0<U<1.

Data and results

The generated value is assigned to the procedure identifier RANDOM CHISQ.

The call TEST RANDOM(41315, DF, U, 0, 0, 0, 0) may yield the following error messages:

Errornumber 1	(if DF is not an integer >0)
Errornumber 2	(if U<0 or U≥1)

Procedures used

RANDOM GAMMA	STATAL 41313

Language

Algol 60

Method and performance

The generated value is equal to a random number having a gamma distribution with parameters ALPHA=DF/2 and SIGMA=2.
See v. Putten & v.d. Tweel (1979).

Reference

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

Example of use

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    U:= .1986;
    "FOR" I:= 1 "STEP" 1 "UNTIL" 10 "DO"
    OUTPUT(61, "("2ZD.6D,/")", RANDOM CHISQ(11, U));
"END"
```

Output:

```
  9.384233
 13.810072
  9.770222
  8.860550
 10.344511
 13.648579
 17.138998
 10.305416
 18.855723
 11.627909
```

Source text

```
"CODE" 41315;
"REAL" "PROCEDURE" RANDOM CHISQ(DF,U);
"VALUE" DF; "REAL" U; "INTEGER" DF;
RANDOM CHISQ:= RANDOM GAMMA(DF / 2, 2, U);
            "EOP"
```

TITLE: **Random Exp**

AUTHOR: J.G. Bethlehem

INSTITUTE: Mathematical Centre

RECEIVED: 750219

BRIEF DESCRIPTION
The procedure generates a random number having an exponential distribution with parameter LAMBDA.

KEYWORDS
Random exponential number

CALLING SEQUENCE
Heading

```
"REAL" "PROCEDURE" RANDOM EXP(LAMBDA, U);
"VALUE" LAMBDA; "REAL" LAMBDA, U;
"CODE" 41304;
```

Formal parameters

LAMBDA <arithmetic expression>, parameter of the exponential distribution;

U: <real variable>, value used to generate a random number. After a call of RANDOM EXP, U contains a new value which is used in the next call of RANDOM EXP unless the program assigns a new value to U.

Conditions: LAMBDA >0 and 0≤U<1.

DATA AND RESULTS
The generated value is assigned to the procedure identifier RANDOM EXP.
The call TEST RANDOM(41304, LAMBDA, U, 0, 0, 0, 0) may yield the following error messages:

Errornumber 1	(if LAMBDA ≤0)
Errornumber 2	(if U<0 or U≥1)

PROCEDURES USED
ASELECT STATAL 41308

LANGUAGE
Algol 60

Method and performance

The generated value is equal to −LN(D)/ LAMBDA, where D is a random number which is uniformly distributed on the interval (0,1].
See v. Putten & v.d. Tweel (1979).

Reference

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

Example of use

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
   U:= .1986;
   "FOR" I:=1 "STEP" 1 "UNTIL" 10 "DO"
   OUTPUT(61, "("ZD.6D,/")", RANDOM EXP(.5, U))
"END"
```

Output:

```
 0.359302
 6.743436
 0.838705
 1.951110
 2.544724
 0.793563
 1.188205
 1.796137
 4.702487
 2.184991
```

Source text

```
"CODE" 41304;
"REAL" "PROCEDURE" RANDOM EXP(LAMBDA, U); "VALUE" LAMBDA;
"REAL" LAMBDA, U;
RANDOM EXP:= -LN(ASELECT(U)) / LAMBDA;
         "EOP"
```

TITLE: **Random Logistic**

AUTHOR: C. van Putten

INSTITUTE: Mathematical Centre

RECEIVED: 810101

BRIEF DESCRIPTION
The procedure generates a random number having a logistic distribution with location parameter LOC and scale parameter SCALE.

KEYWORDS
Random logistic number

CALLING SEQUENCE
Heading

```
"REAL" "PROCEDURE" RANDOM LOGISTIC (LOC, SCALE, U);
"VALUE" LOC, SCALE;
"REAL" LOC, SCALE, U;
"CODE" 41322;
```

Formal parameters
LOC: <arithmetic expression>, location parameter of the distribution;
SCALE: <arithmetic expression>, scale parameter of the distribution;
U: <real variable>, value used to generate a random number.
After a call of RANDOM LOGISTIC, U contains a new value which is used in the next call of RANDOM LOGISTIC unless the program assigns a new value to U.
Conditions: SCALE>0, and 0≤U<1.

DATA AND RESULTS
The generated value is assigned to the procedure identifier RANDOM LOGISTIC.
The call TEST RANDOM(41322, LOC, SCALE, U, 0, 0, 0) may yield the following error messages:

Errornumber 2	(if SCALE ≤0)
Errornumber 3	(if U<0 or U≥0)

PROCEDURES USED
ASELECT STATAL 41308

LANGUAGE
Algol 60

Method and performance

The generated value is equal to LOC−SCALE*LN(1/D−1), when D is a random number which is uniformly distributed on (1,0].
See v. Putten & v.d. Tweel (1979).

Reference

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

Example of use

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    U:= .1986;
    "FOR" I:= 1 "STEP" 1 "UNTIL" 10 "DO"
    OUTPUT(61, "("-2ZD.6D,/")",
               RANDOM LOGISTIC(0, 1, U));
"END"
```

Output:

```
  1.625568
 -3.336784
  0.652050
 -0.502373
 -0.943623
  0.719427
  0.208989
 -0.374907
 -2.251146
 -0.683958
```

Source text

```
"CODE" 41322;
"REAL" "PROCEDURE" RANDOM LOGISTIC (LOC, SCALE, U);
"VALUE" LOC, SCALE; "REAL" LOC, SCALE, U;
"BEGIN" "REAL" V;
    V:=ASELECT(U); "IF" V = 1 "THEN" V:= ASELECT(U);
    RANDOM LOGISTIC:= LOC - SCALE * LN (1 / V - 1)
"END" RANDOM LOGISTIC;
          "EOP"
```

TITLE: **Random Gamma**

AUTHOR: C. van Putten

INSTITUTE: Mathematical Centre

RECEIVED: 810101

BRIEF DESCRIPTION
The procedure generates a random number having a gamma distribution with shape and scale parameters ALPHA and SIGMA.

KEYWORDS
Random gamma number

CALLING SEQUENCE
Heading

```
"REAL" "PROCEDURE" RANDOM GAMMA (ALPHA, SIGMA, U);
"VALUE" ALPHA, SIGMA;
"REAL" ALPHA, SIGMA, U;
"CODE" 41313;
```

Formal parameters

ALPHA: <arithmetic expression>, shape parameter of the gamma distribution;

SIGMA: <arithmetic expression>, scale parameter of the gamma distribution;

U: <real variable>, value used to generate a random number. After a call of RANDOM GAMMA, U contains a new value which is used in the next call of RANDOM GAMMA unless the program assigns a new value to U.

Conditions: ALPHA, SIGMA >0, and 0⩽U<1.

DATA AND RESULTS
The generated value is assigned to the procedure identifier RANDOM GAMMA.
The call TEST RANDOM(41313, ALPHA, SIGMA, U, 0, 0, 0) may yield the following error messages:

Errornumber 1	(if ALPHA⩽0)
Errornumber 2	(if SCALE ⩽0)
Errornumber 3	(if U<0 or U⩾1)

PROCEDURES USED
ASELECT STATAL 41308

LANGUAGE
Algol 60

METHOD AND PERFORMANCE
The random number is generated according to the rejection method.
See v. Putten & v.d. Tweel (1979).

REFERENCE

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

EXAMPLE OF USE

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    U:= .1986;
    "FOR" I:= 1 "STEP" 1 "UNTIL" 10 "DO"
    OUTPUT(61, "("2ZD.6D,/")", RANDOM GAMMA(6.8, 21, U));
"END"
```

Output:

```
123.954855
174.883625
128.486894
117.773322
135.193981
173.060565
211.984181
134.738707
230.800622
150.039688
```

SOURCE TEXT

```
"CODE" 41313;
"REAL" "PROCEDURE" RANDOM GAMMA(ALPHA, SIGMA, U);
"VALUE" ALPHA, SIGMA; "REAL" ALPHA, SIGMA, U;
"BEGIN" "IF" ALPHA < 1 "THEN"
    "BEGIN" "REAL" U1, U2, V1, V2, W, Y;
 NEXT1: U1:= ASELECT(U); U2:= ASELECT(U);
        V1:= U1 ** (1 / ALPHA);
        V2:= U2 ** (1 / (1 - ALPHA));
        W:= V1 + V2;
        "IF" W > 1 "THEN" "GOTO" NEXT1 "ELSE" Y:= V1 / W;
        RANDOM GAMMA:= -LN(ASELECT(U)) * Y * SIGMA
    "END" "ELSE"
    "BEGIN" "REAL" A, B, C, U1, U2, V, X;
```

```
        A:= 1 / SQRT(ALPHA * 2 - 1); B:= ALPHA - LN(4);
        C:= ALPHA + SQRT(ALPHA * 2 - 1);
 NEXT2: U1:= ASELECT(U); U2:= ASELECT(U);
        "IF" U2 = 1 "THEN" U2:= ASELECT(U);
        V:= A * LN(U2 / (1 - U2)); X:= ALPHA * EXP(V);
        "IF" B + C * V - X < LN(U1 * U2 * U2) "THEN"
            "GOTO" NEXT2
        "ELSE" RANDOM GAMMA:= X * SIGMA
    "END"

"END" RANDOM GAMMA;
         "EOP"
```

TITLE: **Random Beta**

AUTHOR: C. van Putten

INSTITUTE: Mathematical Centre

RECEIVED: 810101

BRIEF DESCRIPTION

The procedure generates a random number having a beta distribution with parameters ALPHA1 and ALPHA2.

KEYWORDS

Random beta number

CALLING SEQUENCE

Heading

```
"REAL" "PROCEDURE" RANDOM BETA (ALPHA1, ALPHA2, U);
"VALUE" ALPHA1, ALPHA2;
"REAL" ALPHA1, ALPHA2, U;
"CODE" 41314;
```

Formal parameters

ALPHA1: <arithmetic expression>, first shape parameter of the beta distribution;

ALPHA2: <arithmetic expression>, second shape parameter of the beta distribution;

U: < real variable> value used to generate a random number. After a call of RANDOM BETA, U contains a new value which is used in the next call of RANDOM BETA unless the program assigns a new value to U.

Conditions: ALPHA1, ALPHA2>0 and 0≤U<1.

DATA AND RESULTS

The generated value is assigned to the procedure identifier RANDOM BETA.

The call TEST RANDOM(41314, ALPHA1, ALPHA2, U, 0, 0, 0) may yield the following error messages:

Errornumber 1	(if ALPHA1 ≤0)
Errornumber 2	(if ALPHA2 ≤0)
Errornumber 3	(if U<0 or U≥1)

PROCEDURES USED

ASELECT	STATAL 41308

Language

Algol 60

Method and performance

The generated value is equal to
(D1*1/ALPHA1)/(D1**1/ALPHA1+D2**1/ALPHA2),
where (D1, D2) is the first pair, in a sequence of random numbers uniformly distributed on (0,1], such that D1**1/ALPHA1+D2 **1/ALPHA2≤1.
See v. Putten & v.d. Tweel (1979).

Reference

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

Example of use

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    U:= .1986;
    "FOR" I:= 1 "STEP" 1 "UNTIL" 10 "DO"
    OUTPUT(61, "("2ZD.6D,/")", RANDOM BETA(5, 5, U));
"END"
```

Output:

```
  0.742113
  0.490507
  0.458454
  0.623171
  0.502997
  0.437990
  0.362342
  0.466458
  0.605676
  0.500535
```

SOURCE TEXT

```
"CODE" 41314;
"REAL" "PROCEDURE" RANDOM BETA(ALPHA1, ALPHA2, U);
"VALUE" ALPHA1, ALPHA2; "REAL" ALPHA1, ALPHA2, U;
"BEGIN" "REAL" U1, U2, V1, V2, W;
NEXT:
    U1:= ASELECT(U); U2:= ASELECT(U);
    V1:= U1 ** (1 / ALPHA1); V2:= U2 ** (1 / ALPHA2);
    W:= V1 + V2;
    "IF" W > 1 "THEN" "GOTO" NEXT
               "ELSE" RANDOM BETA:= V1 / W;
"END" RANDOM BETA;
"EOP"
```

TITLE: **Random Cauchy**

AUTHOR: C. van Putten

INSTITUTE: Mathematical Centre

RECEIVED: 810101

BRIEF DESCRIPTION

The procedure generates a random number having a Cauchy distribution with location parameter (median) LOC and scale parameter SCALE.

KEYWORDS

Random Cauchy number

CALLING SEQUENCE

Heading

```
"REAL" "PROCEDURE" RANDOM CAUCHY (LOC, SCALE, U);
"VALUE" LOC, SCALE;
"REAL" LOC, SCALE, U;
"CODE" 41318;
```

Formal parameters

LOC: <arithmetic expression>, location parameter of the Cauchy distribution;

SCALE: <arithmetic expression>, scale parameter of the Cauchy distribution;

U: <real variable>, value used to generate a random number. After a call of RANDOM CAUCHY, U contains a new value which is used in the next call of RANDOM CAUCHY unless the program assigns a new value to U.

Conditions: SCALE>0, and 0⩽U<1.

DATA AND RESULTS

The generated value is assigned to the procedure identifier RANDOM CAUCHY. The call TEST RANDOM(41318, LOC, SCALE, U, 0, 0, 0) may yield the following error messages:

Errornumber 2	(if SCALE ⩽0)
Errornumber 3	(if U<0 or U⩾1)

PROCEDURES USED

ASELECT STATAL 41308

LANGUAGE

Algol 60

Method and performance

The generated value is equal to LOC−SCALE* COS(π*D)/SIN(π*D), where D is a random number which is uniformly distributed on (0,1].
See v. Putten & v.d. Tweel (1979).

Reference

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

Example of use

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    U:= .1986;
    "FOR" I:= 1 "STEP" 1 "UNTIL" 10 "DO"
    OUTPUT(61, "("-ZD.6D,/")", RANDOM CAUCHY(0, 1, U));
"END"
```

Output:

```
  1.760396
 -9.235918
  0.539459
 -0.406933
 -0.826378
  0.601967
  0.165018
 -0.299557
 -3.241467
 -0.568815
```

Source text

```
"CODE" 41318;
"REAL" "PROCEDURE" RANDOM CAUCHY (LOC, SCALE, U);
"VALUE" LOC, SCALE; "REAL" LOC, SCALE, U;
"BEGIN" "REAL" U1, PI;
  U1:= ASELECT (U); "IF" U1 = 1 "THEN" U1 := ASELECT (U);
  PI := 3.1415925358979;
  RANDOM CAUCHY:=
      LOC - SCALE * COS (PI * U1) / SIN (PI * U1)

"END" RANDOM CAUCHY;
        "EOP"
```

TITLE: **Random Weibull**

AUTHOR: C. van Putten

INSTITUTE: Mathematical Centre

RECEIVED: 810101

BRIEF DESCRIPTION
The procedure generates a random number having a Weibull distribution with parameters LOC, SCALE and ALPHA.

KEYWORDS
Random Weibull number

CALLING SEQUENCE
Heading

```
"REAL" "PROCEDURE" RANDOM WEIBULL (LOC, SCALE, ALPHA, U);
"VALUE" LOC, SCALE, ALPHA;
"REAL" LOC, SCALE, ALPHA, U;
"CODE" 41319;
```

Formal parameters

LOC: <arithmetic expression>, location parameter of the Weibull distribution;

SCALE: <arithmetic expression>, scale parameter of the Weibull distribution;

ALPHA: <arithmetic expression>, shape parameter of the Weibull distribution;

U: <real variable>, value used to generate a random number. After a call of RANDOM WEIBULL, U contains a new value which is used in the next call of RANDOM WEIBULL unless the program assigns a new value to U.

Conditions: SCALE, ALPHA>0, and 0⩽U<1.

DATA AND RESULTS
The generated values is assigned to the procedure identifier RANDOM WEIBULL. The call TEST RANDOM(41319, LOC, SCALE, ALPHA, U, 0, 0) may yield the following error messages:

Errornumber 2	(if SCALE ⩽0)
Errornumber 3	(if ALPHA ⩽0)
Errronumber 4	(if U<0 or U ⩾1)

PROCEDURES USED

ASELECT	STATAL 41308

LANGUAGE

Algol 60

METHOD AND PERFORMANCE

The generated value is equal to LOC + SCALE*((−LN(D)) **(1/ALPHA)) where D is a random number which is uniformly distributed on (0,1].
See v. Putten & v.d. Tweel (1979).

REFERENCE

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

EXAMPLE OF USE

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    U:= .1986;
    "FOR" I:= 1 "STEP" 1 "UNTIL" 10 "DO"
    OUTPUT(61, "("-ZD.6D,/")",
               RANDOM WEIBULL(1, 1.0, 2, U));
"END"
```

Output:

```
  1.423853
  2.836224
  1.647574
  1.987702
  2.127990
  1.629906
  1.770780
  1.947665
  2.533377
  2.045225
```

SOURCE TEXT

```
"CODE" 41319;
"REAL" "PROCEDURE" RANDOM WEIBULL(LOC, SCALE, ALPHA, U);
"VALUE" LOC, SCALE, ALPHA; "REAL" LOC, SCALE, ALPHA, U;
RANDOM WEIBULL:=
    LOC + SCALE * (-LN(ASELECT(U))) ** (1 / ALPHA);
"EOP"
```

TITLE: **Random Laplace**

AUTHOR: C. van Putten

INSTITUTE: mathematical Centre

RECEIVED: 810101

BRIEF DESCRIPTION

The procedure generates a random number having a Laplace distribution with mean LOC and scale parameter SCALE (which is proportional to the standard deviation).

KEYWORDS

Random Laplace number

CALLING SEQUENCE

Heading

```
"REAL" "PROCEDURE" RANDOM  LAPLACE (LOC, SCALE, U);
"VALUE LOC, SCALE;
"REAL" LOC, SCALE, U;
"CODE" 41320;
```

Formal parameters

LOC: <arithmetic expression>, location parameter of the Laplace distribution;

SCALE: <arithmetic expression>, scale parameter of the Laplace distribution;

U: <real variable>, value used to generate a random number. After a call of RANDOM LAPLACE, U contains a new value which is used in the next call of RANDOM LAPLACE unless the program assigns a new value to U.

Conditions: SCALE > 0 and $0 \leqslant$ U < 1

DATA AND RESULTS

The generated value is assigned to the procedure identifier RANDOM LAPLACE.

The call TEST RANDOM(41320, LOC, SCALE, U, 0, 0, 0) may yield the following error messages:

Errornumber 2	(if SCALE $\leqslant 0$)
Errornumber 3	(if U < 0 or U $\geqslant 1$)

PROCEDURES USED

ASELECT	STATAL 41308

Language

Algol 60

Method and performance

The generated value is equal to `LOC+SCALE*LN(2*D)` when `D` ⩽ `1/2`, and equals `LOC-SCALE*LN(2*(1-D))` when `D > 1/2`, where `D` is a random number which is uniformly distributed on (0, 1]. See v. Putten & v.d. Tweel (1979).

Reference

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

Example of use

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    U:= .1986;
    "FOR" I:= 1 "STEP" 1 "UNTIL" 10 "DO"
    OUTPUT(61, "("-2ZD.6D,/")",
                 RANDOM LAPLACE(100, 100, U));
"END"
```

Output:

```
 211.207199
-167.857065
 137.825578
  71.759230
  42.078511
 142.306175
 110.994404
  79.507890
 -65.809651
  60.065177
```

Source text

```
"CODE" 41320;
"REAL" "PROCEDURE" RANDOM LAPLACE (LOC, SCALE, U);
"VALUE" LOC, SCALE; "REAL" LOC, SCALE, U;
"BEGIN" ASELECT(U); "IF" U = 0 "THEN" ASELECT(U);
    RANDOM LAPLACE:=
    "IF" U <= 0.5 "THEN" LOC + SCALE * LN(2 * U)
    "ELSE" LOC - SCALE * LN(2 * (1 - U));

"END" RANDOM LAPLACE;
        "EOP"
```

TITLE: **Random Gumbel**

AUTHOR: C. van Putten

INSTITUTE: Mathematical Centre

RECEIVED: 810101

BRIEF DESCRIPTION
The procedure generates a random number having a Gumbel distribution with parameters LOC and SCALE.

KEYWORDS
Random Gumbel number

CALLING SEQUENCE
Heading

```
"REAL" "PROCEDURE" RANDOM GUMBEL (LOC, SCALE, U);
"VALUE" LOC, SCALE;
"REAL" LOC, SCALE, U;
"CODE" 41321;
```

Formal parameters
LOC: <arithmetic expression>, location parameter of the Gumbel distribution;
SCALE: <arithmetic expression>, scale parameter of the Gumbel distribution;
U: <real variable>, value used to generate a random number.
After a call of RANDOM GUMBEL, U contains a new value which is used in the next call of RANDOM GUMBEL unless the program assigns a new value to U.
Conditions: SCALE > 0 and $0 \leqslant$ U < 1.

DATA AND RESULTS
The generated value is assigned to the procedure identifier RANDOM GUMBEL.
The call TEST RANDOM(41321, LOC, SCALE, U, 0, 0, 0) may yield the following error messages:

Errornumber 2	(if SCALE $\leqslant 0$)
Errornumber 3	(if U < 0 or U $\geqslant 1$)

PROCEDURES USED

ASELECT	STATAL 41308

LANGUAGE
Algol 60

Method and performance

The generated value is equal to LOC*SCALE+LN(−LN(D)), where D is a random number which is uniformly distributed in (0,1]. See v. Putten & v.d. Tweel.

Reference

[1]. C. van Putten, I. van der Tweel
On generating random variables
STATAL report 2, report SN 9/79
Mathematical Centre, Amsterdam, 1979

Example of use

Program:

```
"BEGIN" "INTEGER" I; "REAL" U;
    U:= .1986;
    "FOR" I:= 1 "STEP" 1 "UNTIL" 10 "DO"
    OUTPUT(61, "("-2ZD.6D,/")",
                 RANDOM GUMBEL(0, 1, U));
"END"
```

Output:

```
  1.716738
 -1.215422
  0.869043
  0.024749
 -0.240875
  0.924369
  0.520704
  0.107509
 -0.854944
 -0.088464
```

Source text

```
"CODE" 41321;
"REAL" "PROCEDURE" RANDOM GUMBEL (LOC, SCALE, U);
"VALUE" LOC, SCALE; "REAL" LOC, SCALE, U;
"BEGIN" "REAL" V;
    V:= ASELECT (U); "IF" V = 1 "THEN" V := ASELECT(U);
    RANDOM GUMBEL:= LOC - SCALE * LN (- LN (V))
"END" RANDOM GUMBEL;
        "EOP"
```

6. TABLES AND PICTURES

This section contains procedures for tabulating and plotting data on a line printer or plotter. The graphical techniques in this section display data in pair-charts, probability plots, two-dimensional scatter plots and in histograms. The procedures which produce tables or pictures via a line printer have a parameter CHN to identify the channel via which the output must be written to file. This channel should be defined in the main program (c.f. CDC−ALGOL 60 r.m., version 5, pp. 8-16). The graphical procedures PLOTPC and PLOTDIST write their output to a so called graph file. This graph file is generated by these procedures and can be send to the plotter by using the command PLOTGF. Alternatively, one can send a graph file to a line printer by using the command PRINTGF or inspect it at a graphical terminal by using GRIMAS. (c.f. SARA publikatie 11).

REFERENCES

[1]. Algol 60 reference manual, version 5, Control Data Corporation, 1979.

[2]. Graphics, SARA publikatie 11, Stichting Academisch Rekencentrum Amsterdam, 1980.

TITLE: **Tabulate**

AUTHOR: H. Elffers

INSTITUTE: Mathematical Centre

RECEIVED: 740130

BRIEF DESCRIPTION
The procedure tabulates a function of two variables on a lattice of arguments.

KEYWORDS
Table of a function of two variables

CALLING SEQUENCE
Heading

```
"PROCEDURE" TABULATE (TEXT, F, PRER, POSTR, LOWY, STEPY, UPPY, PREY,
POSTY, TEXTY, LOWX, STEPX, UPPX, PREX, POSTX, TEXTX, CHN, LINES, POS);
"VALUE" PRER, POSTR, LOWY, STEPY, UPPY, PREY, POSTY, LOWX, STEPX, UPPX,
PREX, POSTX, CHN, LINES, POS;
"INTEGER" PRER, POSTR, PREY, POSTY, PREX, POSTX, CHN, LINES, POS;
"REAL" LOWY, STEPY, UPPY, LOWX, STEPX, UPPX;
"STRING" TEXT, TEXTY, TEXTX;
"REAL" "PROCEDURE" F;
"CODE" 40200;
```

Formal parameters

TEXT: <string>, identifying text, heading of the table;

F: <real procedure>, declaring the function to be tabulated. The name of the procedure is an actual parameter, no Jensen's device is used;

PRER,POSTR,PREY,POSTY,PREX,POSTX:
<integer arithmetic expression>, defining the respective formats of the numbers to be printed, PRE(POST) denotes the number of places to the left (right) of the decimal point. R denotes the function values, X denotes the values of the first argument, to be printed along the horizontal axis, Y denotes the values of the second argument, to be printed along the vertical axis;

LOWY,STEPY,UPPY,LOWX,STEPX,UPPX:
<arithmetic expression>, the respective values of the arguments. LOW denotes the first value of an argument, STEP the step value and UPP the last value. X denotes the first argument, Y the second argument;

TEXTX: <string>, label for the first argument;

TEXTY: <string>, label for the second argument;

CHN: <integer arithmetic expression>, channel number via which the output is written to file;

LINES: <integer arithmetic expression>, number of lines per page;

POS: <integer arithmetic expression>, number of positions per line;

Calling sequence F

Heading

```
"REAL" "PROCEDURE" F(X,Y);
"VALUE" X,Y;"REAL" X,Y;
```

Formal parameters

X: <arithmetic expression>, first argument;
Y: <arithmetic expression>, second argument.

Data and results

TABULATE generates a table of values of the function F(X, Y) for
X= LOWX, LOWX+STEPX, LOWX+STEPX*2,...,UPPX and
Y= LOWY, LOWY+STEPY, LOWY+STEPY*2,...,UPPY.
The procedure takes care of a neat layout.

The following error messages may appear:

Errornumber 1	(if, given length of text, there is no room for LINES on channel CHN)
Errornumber 3	(if PRER <0)
Errornumber 4	(if POSTR <0)
Errornumber 5	(if PRER+POSTR=0 or PRER+POSTR >24)
Errornumber 6	(if STEPY=0)
Errornumber 7	(if SIGN ((UPPY−LOWY)*STEPY)<0)
Errornumber 8	(if PREY+POSTY >24)
Errornumber 12	(if STEPX=0)
Errornumber 13	(if SIGN((UPPX−LOWX)*STEPX)<0)
Errornumber 14	(if PREX+POSTX>24)
Errornumber 17	(if CHN ⩽0 or CHN=60)
Errornumber 18	(if no vertical boundary is given on channel CHN)
Errornumber 19	(if no horizontal boundary is given on channel CHN)
Errornumber 20	(if, given POS, there is no room for the number of columns)

Procedures used

STATAL3 ERROR STATAL 40100

Language

Algol 60

Method and performance

The format parameters and the range parameters are checked. The layout is prepared by the checking of the size parameters. After the final error checking the table is produced.

Example of use

Program:

```
"BEGIN" "REAL" "PROCEDURE" F(X,Y);
        "VALUE" X,Y; "REAL" X,Y;
        F:= X*X - Y*Y;
  CHANNEL(71,"("E")",61);
  TABULATE("("TABLE OF X*X - Y*Y")", F, 1, 2,
  -1, 0.4, 1, 1, 1, "("Y-VALUE")",
  -1, 0.4, 1, 1, 1, "("X-VALUE")",
  71, 40, 60);
"END"
```

Output:

```
TABLE OF X*X - Y*Y

X-VALUE:  -1.0  -0.6  -0.2   0.2   0.6   1.0 I
Y-VALUEI=====================================I
 -1.0  I  0.00  0.64  0.96  0.96  0.64  0.00 I
 -0.6  I -0.64  0.00  0.32  0.32  0.00 -0.64 I
 -0.2  I -0.96 -0.32  0.00 -0.00 -0.32 -0.96 I
  0.2  I -0.96 -0.32  0.00  0.00 -0.32 -0.96 I
  0.6  I -0.64  0.00  0.32  0.32  0.00 -0.64 I
  1.0  I  0.00  0.64  0.96  0.96  0.64  0.00 I
=============================================

=============================================
```

Source text

```
"CODE" 40200;
"PROCEDURE" TABULATE(MAIN TEXT,FUNCTION,PRE R,POST R,
              FROM V,WITH V,TO V,PRE V,POST V,TEXT V,
              FROM H,WITH H,TO H,PRE H,POST H,TEXT H,
         CHANNEL,NUMBER OF LINES,NUMBER OF POSITIONS);
    "VALUE" PRE R,POST R,FROM V,WITH V,TO V,PRE V,POST V,
            FROM H,WITH H,TO H,PRE H,POST H,
            CHANNEL,NUMBER OF LINES,NUMBER OF POSITIONS;
    "STRING" MAIN TEXT,TEXT V,TEXT H;
    "REAL""PROCEDURE" FUNCTION;
    "INTEGER" PRE R,POST R,PRE V,POST V,PRE H,POST H,
              CHANNEL,NUMBER OF LINES,NUMBER OF POSITIONS;
    "REAL" FROM V,WITH V,TO V,FROM H,WITH H,TO H;
    "COMMENT" PROVISIONAL VERSION, NOTHING IS GUARANTEED;
    "COMMENT" USERS COMMENT SEE DESCRIPTION;
    "COMMENT" SUFFIX V STANDS FOR VERTICAL ARGUMENT,
                     H            HORIZONTAL ARGUMENT,
                     R            RESULT,
                     L            LENGTH,
              PREFIX I STANDS FOR INTEGER CODED EQUIVALENT
                                  OF THE VARIABLE AFTER
                                  THE PREFIX;
```

```
"BEGIN" "INTEGER" LOCAL,LEFTCOLUMNL,TEXTHL,TEXTVL,
                ARG VL,RIGHTARGVBLANK,LEFTARGVBLANK,
                TEXTHBLANK,RIGHTTEXTVBLANK,LEFTTEXTVBLANK,
                COLUMNL,ARGHL,RESULTL,ARGHBLANK,
                RESULTBLANK,NUMBER OF COLUMNS,ACTUAL LENGTH,
                LASTH,FIRSTH,ITOV,ITOH;

    "REAL" MAX RESULT;

    "PROCEDURE" TEXT AND H LAYOUT;
    "IF" POST H=0
    "THEN" FORMAT("("*,N,//,XB,N,"(":")",X(XB-XZD),
           B,"("I")",/")",
           TEXTHBLANK,NUMBER OF COLUMNS,ARGHBLANK,PREH-1)
    "ELSE" FORMAT("("*,N,//,XB,N,"(":")",X(XB-XZD.XD),
           B,"("I")",/")",TEXTHBLANK,NUMBER OF COLUMNS,
           ARGHBLANK,PREH-1,POST H);

    "PROCEDURE" ARG V LAYOUT;
    "IF" POST V = 0
    "THEN" FORMAT("("/,XB-XZDXB,"("I")"")",
           LEFTARGVBLANK,PRE V-1,RIGHTARGVBLANK)
    "ELSE" FORMAT("("/,XB-XZD.XDXB,"("I")"")",
           LEFTARGVBLANK,PRE V-1,POST V,RIGHTARG V BLANK);

    "PROCEDURE" TEXT V LAYOUT;
    FORMAT("("XB,N,XB,"("I")",X(N),"("I")"")",
           LEFTTEXTVBLANK,RIGHTTEXTVBLANK,
           ACTUAL LENGTH-LEFTCOLUMNL-2);

    "PROCEDURE" TEXT V LIST(ITEM);"PROCEDURE" ITEM;
    "BEGIN""INTEGER" POS;
        ITEM(TEXT V);
        "FOR" POS:= LEFTCOLUMNL+3 "STEP" 1
        "UNTIL" ACTUAL LENGTH "DO"
        ITEM("("=")");
    "END" TEXT V LIST;

    "PROCEDURE" RESULT OVERFLOW(RESULT);
    "VALUE" RESULT; "REAL" RESULT;
    "BEGIN" OUTPUT(CHANNEL,"("//,
        "("RESULT DOES NOT FIT IN GIVEN FORMAT, RESULT = ")"
        ,N,//,
        "("* THE PROCEDURE  TABULATE  IS LEFT HERE *")",
        ///")", RESULT);
        "GOTO" EXIT TABULATE;
    "END" RESULT OVERFLOW;

    "PROCEDURE" RESULT LAYOUT;
    "IF" POST R=0
    "THEN" FORMAT("("X(XB-XZD),B,"("I")"")",
                  NUMBER OF COLUMNS,RESULTBLANK,PRE R-1)
    "ELSE" FORMAT("("X(XB-XZD.XD),B,"("I")"")",
           NUMBER OF COLUMNS,RESULTBLANK,PRE R-1,POST R);
```

```
"PROCEDURE" UNDERLINE LAYOUT;
FORMAT("("/,X(N),/")",ACTUAL LENGTH);

"PROCEDURE" UNDERLINE LIST(ITEM);"PROCEDURE" ITEM;
"BEGIN" "INTEGER" POS;
   "FOR" POS:= 1 "STEP" 1 "UNTIL" ACTUAL LENGTH "DO"
   ITEM("("=")");
"END" UNDERLINE LIST;


"PROCEDURE" GROUP OF COLUMNS(FIRST H,LAST H);
"VALUE" FIRST H,LAST H; "INTEGER" FIRST H,LAST H;
"BEGIN""INTEGER" FIRST V,LAST V;

    "PROCEDURE" TEXT AND H LIST(ITEM); "PROCEDURE" ITEM;
    "BEGIN""INTEGER" RUN H;
        ITEM(MAIN TEXT); ITEM(TEXT H);
        "FOR" RUN H:= FIRST H "STEP" 1 "UNTIL" LAST H
        "DO" ITEM(RUN H*WITH H+FROM H);
    "END" TEXT AND H LIST;

    "PROCEDURE" PAGE(FIRST V,LAST V);
    "VALUE" FIRST V,LAST V; "INTEGER" FIRST V,LAST V;
    "BEGIN" "INTEGER" RUN V;

        "PROCEDURE" ARG V LIST(ITEM); "PROCEDURE" ITEM;
        ITEM(WITH V*RUN V+FROM V);

        "PROCEDURE" RESULT LIST(ITEM); "PROCEDURE" ITEM;
        "BEGIN""INTEGER" RUN H;
            "REAL" RESULT;
            "FOR" RUN H:= FIRST H "STEP" 1
            "UNTIL" LAST H "DO"
            "BEGIN" RESULT:=
                FUNCTION(RUN V * WITH V+FROM V,
                RUN H * WITH H + FROM H);
                "IF" ABS(RESULT) > MAX RESULT "THEN"
                    RESULT OVERFLOW(RESULT)
                "ELSE" ITEM( RESULT);
            "END";
        "END" RESULTLIST;

        OUTLIST(CHANNEL,TEXT AND H LAYOUT,
        TEXT AND H LIST);
        OUTLIST(CHANNEL,TEXT V LAYOUT,TEXT V LIST);
        "FOR" RUN V:= FIRST V "STEP" 1 "UNTIL" LAST V
        "DO"
        "BEGIN"
            OUTLIST(CHANNEL,ARG V LAYOUT,ARG V LIST);
            OUTLIST(CHANNEL,RESULTLAYOUT,RESULT LIST);
        "END";

        OUTLIST(CHANNEL,UNDERLINE LAYOUT,
        UNDERLINE LIST);
    "END" PAGE;

    LAST V:= -1;
    "FOR" FIRST V:= LAST V+1 "WHILE" FIRST V<= ITOV "DO"
    "BEGIN" LAST V:= LAST V + NUMBER OF LINES;
        "IF" LAST V > ITOV "THEN" LAST V:= ITOV;
        PAGE(FIRST V,LAST V);
    "END" FOR;

"END" GROUP OF COLUMNS;

"PROCEDURE" INFER FORMAT(PRE,POST,FROM,WITH,TO,ERROR);
    "VALUE" FROM,WITH,TO,ERROR;
    "INTEGER" PRE, POST, ERROR; "REAL" FROM, WITH, TO;
"BEGIN"
    "INTEGER""PROCEDURE" PREDIGITS(A,B);
    "VALUE" A,B; "REAL" A,B;
    "BEGIN" "IF" A<B "THEN" A:= B;
        PREDIGITS:= "IF" A<=1 "THEN" 1 "ELSE"
                  -ENTIER(-LN(A) * .43429448190352);
    "END" PREDIGITS;

    "INTEGER" "PROCEDURE" POSTDIGITS(A);
    "VALUE" A; "REAL" A;
    "BEGIN""INTEGER" I; I:= 0;
        "FOR" A:= A, A * 10 "WHILE" I < 14 "DO"
        "IF" ABS(A - ENTIER(A)) < "-13
        "THEN" "GOTO" EXIT LOOP
        "ELSE" I:= I + 1;
    EXIT LOOP:
        POSTDIGITS:= I;
    "END" POSTDIGITS;

    "INTEGER" "PROCEDURE" MAX(A,B,C);
    "VALUE" A,B,C;"INTEGER" A,B,C;
    "BEGIN" "IF" B>A "THEN" A:= B;
        "IF" C>A "THEN" A:= C;
        MAX:= A;
    "END" MAX;

    "INTEGER" PRENEED, POSTNEED;

    FROM:= ABS(FROM); WITH:= ABS(WITH); TO:= ABS(TO);
    PRENEED:= PREDIGITS(FROM,TO);
    POSTNEED:=MAX(POSTDIGITS(FROM),POSTDIGITS(WITH),
                                   POSTDIGITS(TO));
    "IF" POST = 0 & PRE = 0 "THEN"
    "BEGIN" PRE:= PRENEED; POST:= POSTNEED "END"
    "ELSE"
    "BEGIN" "IF" POST < POSTNEED
        "THEN" MESSAGE(4+ERROR, POSTNEED);
        "IF" POST < 0 "THEN" POST:= 0;
        "IF" PRE < PRENEED "THEN"
        "BEGIN" MESSAGE(5 + ERROR, PRE);
```

```
            PRE:= PRENEED
        "END";
    "END";

CHECK ON MAXIMUM FORMAT:
    "IF" PRE + POST > 24 "THEN"
    "BEGIN" "IF" PRE > PRENEED ! POST > POSTNEED "THEN"
        "BEGIN" MESSAGE(ERROR + 21, PRE + POST);
            PRE:= PRENEED;
            "IF" POSTNEED < POST "THEN" POST:=POSTNEED;
        "END";
        "IF" PRE + POST > 24 "THEN"
            MESSAGE(ERROR+22, PRE + POST);
    "END";

"END" INFER FORMAT;

"PROCEDURE" MESSAGE(NUMBER,SCAPEGOAT);
"VALUE" NUMBER,SCAPEGOAT;
    "INTEGER" NUMBER,SCAPEGOAT;
"BEGIN"

    "SWITCH" ERROR:= E1,E2,E3,E4,E5,E6,E7,E8,E9,E10,
                     E11,E12,E13,E14,E15,E16,E17,E18,
                     E19,E20,E21,E22,E23,E24,E25;

    "PROCEDURE" NON FATAL(TEXT,PRINT SCAPEGOAT);
    "STRING" TEXT; "BOOLEAN" PRINT SCAPEGOAT;
    "BEGIN"OUTPUT(CHANNEL,"("//,"("        MESSAGE: ")",
        N")",TEXT);
        "IF" PRINT SCAPEGOAT "THEN"
            OUTPUT(CHANNEL,"("B-6ZD")", SCAPEGOAT);
        "GOTO" EXIT MESSAGE;
    "END" NON FATAL;

    "IF" NO ERRORS "THEN"
    "BEGIN" OUTPUT(CHANNEL,"("///,
        "(" * MESSAGE FROM PROCEDURE TABULATE ,")",
        "(" CALLED WITH MAINTEXT: ")",N,"(" *")"")",
                   MAINTEXT);
        NO ERRORS:= "FALSE";
    "END";

    "GOTO" ERROR[NUMBER];

E1: STATAL3 ERROR("("TABULATE")", 3, SCAPEGOAT);
E2: STATAL3 ERROR("("TABULATE")", 4, SCAPEGOAT);
E3: STATAL3 ERROR("("TABULATE")", 5, SCAPEGOAT);
E4: NON FATAL("("INFORMATION LOST BEHIND COMMA,
    NEEDED POST V =")", "TRUE");
E5: NON FATAL("("PRE V IS TOO SMALL, PRE V =")",
              "TRUE");
E6: NON FATAL(
    "("INFORMATION LOST BEHIND COMMA, NEEDED POST H =")"
```

```
                  ,"TRUE");
    E7: NON FATAL("("PRE H IS TOO SMALL, PRE H =")",
                "TRUE");
    E8: STATAL3 ERROR("("TABULATE")", 19, SCAPEGOAT);
    E9: NON FATAL(
        "("VERTICAL RANGE TRUNCATED ON INTEGER CODING")",
                  "FALSE");
    E10: NON FATAL(
         "("HORIZONTAL RANGE TRUNCATED ON INTEGER CODING")",
                   "FALSE");
    E11: STATAL3 ERROR("("TABULATE")", 6, SCAPEGOAT);
    E12: STATAL3 ERROR("("TABULATE")", 12, SCAPEGOAT);
    E13: STATAL3 ERROR("("TABULATE")", 7, SCAPEGOAT);
    E14: STATAL3 ERROR("("TABULATE")", 13, SCAPEGOAT);
    E15: STATAL3 ERROR("("TABULATE")", 1, SCAPEGOAT);
    E16: NON FATAL(
         "("TOO GREAT NUMBER OF LINES =")", "TRUE");
    E17: STATAL3 ERROR("("TABULATE")", 18, SCAPEGOAT);
    E18: NON FATAL(
         "("TOO GREAT NUMBER OF POSITIONS =")",
                "TRUE");
    E19: NON FATAL("("TOO GREAT NUMBER OF COLUMNS ASKED :")"
         ,"TRUE");
    E20: STATAL3 ERROR("("TABULATE")", 20, SCAPEGOAT);
    E21: NON FATAL(
"("SPECIFIED/INFERRED PRE V+POST V > 24, PRE V+POST V =")"
               ,"TRUE");
    E22: STATAL3 ERROR("("TABULATE")", 8, SCAPEGOAT);
    E23: NON FATAL(
"("SPECIFIED/INFERRED PRE H+POST H > 24, PRE V+POST V =")"
               ,"TRUE");
    E24: STATAL3 ERROR("("TABULATE")", 14, SCAPEGOAT);
    E25: STATAL3 ERROR("("TABULATE")", 17, SCAPEGOAT);

    EXIT MESSAGE:

    "END" MESSAGE;

    "BOOLEAN" NO ERRORS;



INITIALIZE:
    NO ERRORS:= "TRUE";

HANDLING OF FORMAT PARAMETERS:
    "IF" PRE R + POST R = 0 "THEN" MESSAGE(3,0);
    "IF" PRE R < 0 "THEN" MESSAGE(1,PRE R) "ELSE"
    "IF" PRE R = 0 "THEN" PRE R := 1;
    "IF" POST R < 0 "THEN" MESSAGE(2,POST R);
    "IF" PRE R + POST R > 24
    "THEN" MESSAGE(3, PRE R + POST R);
    MAX RESULT:= 10 ** PRE R - .5 * 10 ** (-POST R);
    INFER FORMAT(PRE V, POST V, FROM V, WITH V, TO V, 0);
    INFER FORMAT(PRE H, POST H, FROM H, WITH H, TO H, 2);

CHECK OF RANGE PARAMETERS:
    "IF" WITH V = 0 "THEN" MESSAGE (11,0);
    "IF" WITH H = 0 "THEN" MESSAGE(12,0);
    "IF" (TO V - FROM V) * WITH V < 0 "THEN" MESSAGE(13,0);
    "IF" (TO H - FROM H) * WITH H < 0 "THEN" MESSAGE(14,0);


PREPARATION OF LAYOUT:

  CHECK OF SIZE PARAMETERS:
    "IF" CHANNEL <= 0 "OR" CHANNEL = 60
    "THEN" MESSAGE(25,CHANNEL);
    NUMBER OF COLUMNS:= 0;
    SYSPARAM(CHANNEL,5,LOCAL);
    "IF" LOCAL = 0 "THEN"
    "BEGIN" MESSAGE(8, CHANNEL); LOCAL:= 135 "END" "ELSE"
    LOCAL:= LOCAL - 1;
    "IF" NUMBER OF POSITIONS = 0 "THEN"
        NUMBER OF POSITIONS:= LOCAL
    "ELSE"
    "IF" NUMBER OF POSITIONS < 0 "THEN"
    "BEGIN" NUMBER OF COLUMNS:= - NUMBER OF POSITIONS;
        NUMBER OF POSITIONS:= LOCAL;
    "END"
    "ELSE"
    "IF" NUMBER OF POSITIONS > LOCAL "THEN"
    "BEGIN" MESSAGE(18,NUMBER OF POSITIONS);
        NUMBER OF POSITIONS:= LOCAL;
    "END";

    SYSPARAM(CHANNEL, 7, LOCAL);
    "IF" LOCAL = 0 "THEN"
    "BEGIN" MESSAGE(17, CHANNEL); LOCAL:= 60 "END";
    LOCAL:= LOCAL +
        ENTIER(-CHLENGTH(MAIN TEXT)/NUMBER OF POSITIONS)-5;
    "IF" LOCAL < 1 "THEN" MESSAGE(15,0);
    "IF" NUMBER OF LINES < 1 "THEN" NUMBER OF LINES:= LOCAL
    "ELSE" "IF" NUMBER OF LINES > LOCAL & LOCAL > 0 "THEN"
    "BEGIN" MESSAGE(16, NUMBER OF LINES);
        NUMBER OF LINES:= LOCAL;
    "END";

  EQUALIZING TEXT AND NUMBER FORMATS IN LEFT COLUMN:
    LEFTCOLUMNL:= TEXTHL:= CHLENGTH(TEXT H);
    TEXTVL:= CHLENGTH(TEXT V);
    "IF" TEXT VL > LEFTCOLUMNL "THEN" LEFTCOLUMNL:= TEXT VL;
    ARGVL:= ("IF" POST V = 0 "THEN" 1 "ELSE" POST V + 2)
            + PRE V;
    "IF" ARGVL >= LEFTCOLUMNL "THEN" LEFTCOLUMNL:= ARGVL+1;
    RIGHTARGVBLANK:= LEFTCOLUMNL - ARGVL;
    LEFTARGVBLANK:= RIGHTARGVBLANK "DIV" 2;
    RIGHTARGVBLANK:= RIGHTARGVBLANK - LEFTARGVBLANK;
    TEXTHBLANK:= LEFTCOLUMNL - TEXTHL;
    RIGHTTEXTVBLANK:= LEFTCOLUMNL - TEXTVL;
    LEFTTEXTVBLANK:= RIGHTTEXTVBLANK "DIV" 2;
    RIGHTTEXTVBLANK:= RIGHTTEXTVBLANK - LEFTTEXTVBLANK;

  EQUALIZING NUMBER FORMATS IN OTHER COLUMNS:
    COLUMNL:= ARGHL:=
    ("IF" POST H = 0 "THEN" 2 "ELSE" POST H + 3) + PRE H;
    RESULTL:=
    ("IF" POST R = 0 "THEN" 2 "ELSE" POST R + 3) + PRE R;
    "IF" RESULTL > COLUMNL "THEN" COLUMNL:= RESULTL;
    ARGHBLANK:= COLUMNL - ARGHL + 1;
    RESULTBLANK:= COLUMNL - RESULTL + 1;

  DETERMINING OVERALL LENGTH:
    "IF" NUMBER OF COLUMNS = 0 "THEN"
    NUMBER OF COLUMNS:=
    (NUMBER OF POSITIONS - 3 - LEFTCOLUMNL) "DIV" COLUMNL;
    "IF" NUMBER OF COLUMNS = 0 "THEN"
    MESSAGE(20,NUMBER OF POSITIONS);
    ACTUAL LENGTH:=
    NUMBER OF COLUMNS * COLUMNL + LEFTCOLUMNL + 3;
    "IF" ACTUAL LENGTH > NUMBER OF POSITIONS
    & NUMBER OF COLUMNS > 0 "THEN"
    "BEGIN" MESSAGE(19, NUMBER OF COLUMNS);
        NUMBER OF POSITIONS:= 0;
        "GOTO" CHECK OF SIZE PARAMETERS;
    "END";

INTEGER CODING OF RANGE:
    "IF" WITH H = 0 ! WITH V = 0 "THEN"
    "GOTO" FINISHING OF ERROR CHECKS;
    ITOV:= ENTIER((TO V - FROM V)/WITH V);
    "IF" FROM V + ITOV * WITH V^= TO V "THEN" MESSAGE( 9, 0);
    ITOH:= ENTIER((TO H - FROM H)/WITH H);
    "IF" FROM H + ITOH * WITH H^= TO H "THEN" MESSAGE(10, 0);

FINISHING OF ERROR CHECKS:
    "IF"^NO ERRORS "THEN"
    "BEGIN" LOCAL:= ACTUAL LENGTH;
        OUTPUT(CHANNEL,"("///,
        "("TABLE IS PRODUCED WITH THE ")",
        "("(POSSIBLY CHANGED) PARAMETERS:")"
        ,/,"("   PRE V             :")",B9ZD,/
        ,"("   POST V           :")",B9ZD,/
        ,"("   PRE H            :")",B9ZD,/
        ,"("   POST H           :")",B9ZD,/
        ,"("   NUMBER OF LINES   :")",B9ZD,/
        ,"("   NUMBER OF POSITIONS:")",B9ZD,/")",
        PRE V,POST V, PRE H,POST H,NUMBER OF LINES,
        NUMBER OF POSITIONS);
        ACTUAL LENGTH:= NUMBER OF POSITIONS;
        OUTLIST(CHANNEL,UNDERLINE LAYOUT,UNDERLINE LIST);
        ACTUAL LENGTH:= LOCAL;
    "END";


PRODUCTION OF TABLE:
    LAST H:= -1;
    "FOR" FIRST H:= LASTH + 1 "WHILE" FIRST H <= ITOH "DO"
    "BEGIN" LASTH:= LASTH + NUMBER OF COLUMNS;
        "IF" LASTH > ITOH "THEN"
        "BEGIN" LASTH:= ITOH;
            NUMBER OF COLUMNS:= LAST H - FIRST H + 1;
            ACTUAL LENGTH:=
            NUMBER OF COLUMNS * COLUMNL +LEFTCOLUMNL+3;
        "END";
        GROUP OF COLUMNS(FIRST H, LAST H);
    "END" FOR;


EXIT TABULATE:
    OUTLIST(CHANNEL,UNDERLINE LAYOUT,UNDERLINE LIST);
    SYSPARAM(CHANNEL,7,LOCAL);
    "IF" LOCAL > 0 "THEN" SYSPARAM(CHANNEL,4,LOCAL-1) "ELSE"
         OUTPUT(CHANNEL,"("///")");
    SYSPARAM(CHANNEL,5,LOCAL);
    "IF" LOCAL > 0 "THEN" SYSPARAM(CHANNEL,2,LOCAL-1);

"END" TABULATE;
         "EOP"
```

TITLE: **Triangle**

AUTHORS: R. Wiggers, E. Opperdoes

INSTITUTE: Mathematical Centre

RECEIVED: 740601

BRIEF DESCRIPTION

The procedure prints the lower triangular part of a square matrix.

KEYWORDS
Printing a lower triangular matrix

CALLING SEQUENCE

Heading

```
"PROCEDURE" TRIANGLE (CHN, TEXT, MATRIX, NROW, ROWNR);
"VALUE" CHN, NROW;
"INTEGER" CHN, NROW;
"STRING" TEXT;
"ARRAY" MATRIX;
"INTEGER" "ARRAY" ROWNR;
"CODE" 40201;
```

Formal parameters

CHN: <integer arithmetic expression>, channel number via which the output is written to file;

TEXT : <string>, identifying text, to be printed on top of each page;

MATRIX: <array identifier>, one-dimensional array MATRIX [1:U], where U=NROW*(NROW+1)/2, which contains the lower triangular part row by row including the diagonal elements;

NROW: <integer arithmetic expression>, number of rows, the sign indicating the print format used;

ROWNR: <integer array identifier>, one-dimensional array ROWNR [1:NROW], which contains integer labels for the rows of the matrix;

DATA AND RESULTS

The lower triangular part of the matrix includes the diagonal and must be given row by row in a one-dimensional array. NROW may be negative. If so, another output format is used, (see method and performance).
The following error messages may appear:

Errornumber 1	(if CHN ≤0 or CHN=60)
Errornumber 4	(if NROW=0)

Procedures used

```
STATAL ERROR              STATAL 40100
```

Language

Algol 60

Method and performance

In the case that NROW>0, the output is in floating point format +.4D"+DDB and on each page 11 columns and 50 rows are printed. If NROW<0, the output is in fixed format −Z.4D2B and 13 columns and 50 rows are printed on each page. If the lower triangular part of the matrix consists of more columns (and rows) than can be printed on one page, it is printed in parts with each part on a new page.

The fixed format is suitable for printing a correlation matrix. The procedure prints the value 4*"+15/9 as ------; this value can be assigned to e.g. correlations which cannot be computed, and a neat output is obtained.

Example of use

Program:

```
"BEGIN" "INTEGER" I; "INTEGER" "ARRAY" R[1:4];
  "ARRAY" A[1:10];

  "FOR" I:=1 "STEP" 1 "UNTIL" 4 "DO" R[I]:= 2 * I;
  INARRAY(60, A);

  CHANNEL(71,"("E")",61);
  TRIANGLE(71, "("MATRIX IN FIXED FORMAT")", A, -4, R);
  TRIANGLE(71, "("MATRIX IN FLOATING-POINT FORMAT")",
           A, 4, R)
"END"
```

Input:

```
 .23252
 .69019  .21350
 .74750  .10000  .16823
 .00783  .11232  .54339  .13678
```

Output:

```
      MATRIX IN FIXED FORMAT

           2        4        6        8

2       .2325
4       .6902    .2135
6       .7475    .1000    .1682
8       .0078    .1123    .5434    .1368

           2        4        6        8


      MATRIX IN FLOATING-POINT FORMAT

           2          4          6          8

2     +.2325"+00
4     +.6902"+00 +.2135"+00
6     +.7475"+00 +.1000"+00 +.1682"+00
8     +.7830"-02 +.1123"+00 +.5434"+00 +.1368"+00

           2          4          6          8
```

SOURCE TEXT

```
"CODE" 40201;
"PROCEDURE" TRIANGLE (CHANNEL, TEXT, MATRIX, SIZE, VARNR);
"VALUE" CHANNEL, SIZE;
"INTEGER" CHANNEL, SIZE; "STRING" TEXT;
"INTEGER" "ARRAY" VARNR; "ARRAY" MATRIX;
"BEGIN"
    "INTEGER" IMIN, COUNT, IMAX, K, I, J, MAXI, PJ, PAGE,
        NK; "BOOLEAN" CORRM; "REAL" X;

    "INTEGER" "PROCEDURE" LINENUMBER;
    "BEGIN" "INTEGER" L; SYSPARAM (CHANNEL, 3, L);
        LINENUMBER:= L + 1
    "END" LINENUMBER;
    "PROCEDURE" NEW;
    "BEGIN" OUTPUT (CHANNEL, "("*")"); NAME; NUMBER "END";

    "PROCEDURE" CARRIAGE (N); "VALUE" N; "INTEGER" N;
    "BEGIN" "INTEGER" I, M;
        SYSPARAM (CHANNEL, 3, I);
        "IF" N < - 1 "OR" N = 1 "OR" N > 31
        "THEN" OUTPUT (CHANNEL, "("/")")
        "ELSE"
        "BEGIN" M:= N + I - 1;
            "IF" N = - 1 "OR" M > 59 "THEN"
                OUTPUT (CHANNEL, "("*")")
            "ELSE"
            "BEGIN" SYSPARAM (CHANNEL, 4, M);
                OUTPUT (CHANNEL, "("/")")
            "END"
        "END"
    "END" CARRIAGE;

    "PROCEDURE" NUMBER;
    "BEGIN" PAGE:= PAGE + 1;
        OUTPUT (CHANNEL, "("5B,"("PAGE:")",BZD,/")", PAGE);
    "END" NUMBER;

    "PROCEDURE" NAME; OUTPUT (CHANNEL, "("8B,N")", TEXT);

    "PROCEDURE" HEADING;
    "BEGIN" "INTEGER" I;
        "IF" CORRM "THEN" OUTPUT (CHANNEL, "("5B")")
                   "ELSE" OUTPUT (CHANNEL, "("3B")");
        "FOR" I:= IMIN "STEP" 1 "UNTIL" IMAX
        "DO" "IF" CORRM
             "THEN" OUTPUT (CHANNEL, "("8ZD")", VARNR [I])
             "ELSE" OUTPUT (CHANNEL, "("10ZD")", VARNR [I])
    "END";

    "IF" CHANNEL <= 0 "OR" CHANNEL = 60 "THEN"
        STATAL3 ERROR("("TRIANGLE")", 1, CHANNEL);
    "IF" SIZE = 0 "THEN"
        STATAL3 ERROR("("TRIANGLE")", 4, SIZE);
    SYSPARAM(CHANNEL,1,I);
    "IF" I > 0 "THEN" OUTPUT(CHANNEL, "("/")");
    CORRM:= SIZE < 0; SIZE:= ABS (SIZE);
    PAGE:= 0; NK:= SIZE + SIZE // 10;
    IMIN:= 1; COUNT:= "IF" CORRM "THEN" 13 "ELSE" 11;
    "IF" LINENUMBER > 1 "THEN"
    "BEGIN" "IF" SIZE <= COUNT "AND" LINENUMBER < 53 - NK
        "THEN" OUTPUT (CHANNEL, "("3/")")
        "ELSE" OUTPUT (CHANNEL, "("*")");

    "END" "ELSE"
    "BEGIN" SYSPARAM(CHANNEL, 1, I);
        "IF" I > 0 "THEN" OUTPUT(CHANNEL, "("/")")
    "END";
    NAME;
    "IF" SIZE <= COUNT "AND" LINENUMBER < 56 - NK
    "THEN" OUTPUT (CHANNEL, "("/")")
    "ELSE" NUMBER;
    IMAX:= "IF" SIZE > COUNT "THEN" COUNT "ELSE" SIZE;
AGAIN:
    K:= IMIN - 1 - (IMIN - 1) // 50 * 50;
    CARRIAGE (K + K // 10 + 1); HEADING;
    OUTPUT (CHANNEL, "("/")");
    "FOR" J:= IMIN "STEP" 1 "UNTIL" SIZE "DO"
    "BEGIN" OUTPUT (CHANNEL, "("/,2ZD,5B")", VARNR [J]);
        MAXI:= "IF" J < IMAX "THEN" J "ELSE" IMAX;
        PJ:= (J - 1) * J / 2;
    "FOR" I:= IMIN "STEP" 1 "UNTIL" MAXI
    "DO"
    "BEGIN" X:= MATRIX [PJ + I];
      "IF" CORRM "THEN"
      "BEGIN" "IF" X = .9999"9 "THEN"
        OUTPUT (CHANNEL, "(""("   ----   ")"")")
        "ELSE" OUTPUT (CHANNEL, "("-Z.4DBB")", X)
      "END"
      "ELSE" OUTPUT (CHANNEL, "("+.4D"+DDB")", X)
    "END";
    "IF" J // 10 * 10 = J "THEN"
    "BEGIN" OUTPUT (CHANNEL, "("/")");
      "IF" J // 50 * 50 = J "AND" J < SIZE "THEN"
      "BEGIN" OUTPUT (CHANNEL, "("/")"); HEADING; NEW;
        OUTPUT (CHANNEL, "("/")"); HEADING;
        OUTPUT (CHANNEL, "("/")");
      "END";

    "END"
  "END";
  CARRIAGE("IF" SIZE // 10 * 10 = SIZE "THEN" 1 "ELSE" 2);
  HEADING; "IF" IMAX = SIZE "THEN" "GOTO" READY;
  IMIN:= IMIN + COUNT; IMAX:= IMAX + COUNT;
  "IF" IMAX > SIZE "THEN" IMAX:= SIZE; NEW; "GOTO" AGAIN;
READY:
"END" TRIANGLE;
    "EOP"
```

TITLE: **Rectangle**

AUTHORS: R. Wiggers, E. Opperdoes

INSTITUTE: Mathematical Centre

RECEIVED: 750201

BRIEF DESCRIPTION
The procedure prints (a rectangular part of) a matrix.

KEYWORDS
Printing a matrix

CALLING SEQUENCE
Heading
```
"INTEGER" "PROCEDURE" RECTANGLE (CHN, TEXT, MATRIX, LR, UR, LC, UC,
COLNR, FORMAT, ROWNR, PAGE);
"VALUE" CHN, LR, UR, LC, UC, FORMAT, PAGE;
"INTEGER" CHN, LR, UR, LC, UC, FORMAT, PAGE;
"STRING" TEXT;
"INTEGER" "ARRAY" ROWNR;
"REAL" "ARRAY" MATRIX, COLNR;
"CODE" 40202;
```
Formal parameters

CHN: <integer arithmetic expression>, channel number via which the output is written to file;

TEXT: <string>, identifying text, to be printed on top of each page;

MATRIX: <array identifier>, to contain the matrix;

LR: <integer arithmetic expression>, smallest index of rows to be printed;

UR: <integer arithmetic expression>, largest index of rows to be printed;

LC: <integer arithmetic expression>, smallest index of columns to be printed;

UC: <integer arithmetic expression>, largest index of columns to be printed;

COLNR: <array identifier>, a one-dimensional array COLNR[LC:UC], which contains labels for the columns of matrix;

FORMAT: <integer arithmetic expression>, code for printing format of the elements of COLNR;

ROWNR: <integer array identifier>, a one-dimensional array ROWNR [LR:UR], which contains integer labels for the rows of the matrix;

PAGE: <integer arithmetic expression>, starting value of the page numbering.

Data and results

If FORMAT=2, the elements of COLNR are printed in format 11ZD, and if FORMAT=1, the format +.4D"+2D2B is used. For other values of FORMAT, no column labels are printed. The number of the next page to be printed is assigned to the procedure identifier RECTANGLE.
The following error messages may appear:

Errornumber 1	(if CHN ≤0 or CHN=60)
Errornumber 4	(if LR >UR)
Errornumber 6	(if LC >UC)

Procedures used

STATAL3 ERROR   STATAL 40100

Language

Algol 60

Method and performance

On each page 11 columns and 50 rows are printed. If the matrix cannot be printed on one page, it is printed in parts, with each part on a new page.

Example of use

Program:

```
"BEGIN" "INTEGER" I, J; "INTEGER" "ARRAY" R[1:12];
  "REAL" "ARRAY" M[1:12, 1:3], C[1:3];
  "FOR" J:=1, 2, 3 "DO" C[J]:= J * 10;
  "FOR" I:=1 "STEP" 1 "UNTIL" 12 "DO" R[I]:= 3 * I;
  INARRAY(60, M);
  CHANNEL(71,"("E")",61);
  RECTANGLE(71, "("EXAMPLE")", M, 1, 12, 1, 3, C, 1, R, 1)
"END"
```

Input:

```
14.189  137.27  1990.1
153.27  2851.9  9626.7
4.2349  704718  73.731
2.6158  2.3072  15.024
190.32  50.274  2137.8
757614  2.2112  40.362
22.834  27.018  2551.9
33.425  7116.5  18.020
4.0250  9422.1  18.520
33.848  70.628  146.23
4.1123  159.23  15.942
673.26  2.7701  2500.1
```

Output:

```
        EXAMPLE

        +.1000"+02  +.2000"+02  +.3000"+02

  3     +.1419"+02  +.1373"+03  +.1990"+04
  6     +.1533"+03  +.2852"+04  +.9627"+04
  9     +.4235"+01  +.7047"+06  +.7373"+02
 12     +.2616"+01  +.2307"+01  +.1502"+02
 15     +.1903"+03  +.5027"+02  +.2138"+04
 18     +.7576"+06  +.2211"+01  +.4036"+02
 21     +.2283"+02  +.2702"+02  +.2552"+04
 24     +.3342"+02  +.7116"+04  +.1802"+02
 27     +.4025"+01  +.9422"+04  +.1852"+02
 30     +.3385"+02  +.7063"+02  +.1462"+03

 33     +.4112"+01  +.1592"+03  +.1594"+02
 36     +.6733"+03  +.2770"+01  +.2500"+04
```

Source text

```
"CODE" 40202;
"INTEGER" "PROCEDURE" RECTANGLE (CHANNEL, TEXT, MATRIX,
    LOWVERT, UPPVERT,LOWHOR, UPPHOR, COLNR,
    HEADINGCODE, ROWNR, PAGE);
"VALUE" LOWVERT, LOWHOR, PAGE, HEADINGCODE,
    UPPVERT, UPPHOR, CHANNEL;
"INTEGER" LOWVERT, LOWHOR, UPPVERT, UPPHOR,
    HEADINGCODE, PAGE, CHANNEL;
"REAL" "ARRAY" MATRIX, COLNR; "STRING" TEXT;
"INTEGER" "ARRAY" ROWNR;
"BEGIN" "INTEGER" I, J, L, U, NK, MM;

    "PROCEDURE" NEW;
    "BEGIN" OUTPUT (CHANNEL, "("*")"); NAME; NUMBER "END";

    "PROCEDURE" NUMBER;
    "BEGIN" OUTPUT (CHANNEL, "("5B,"("PAGE:")",BZD,/")",
        PAGE);
        PAGE:= PAGE + 1
    "END" NUMBER;

    "PROCEDURE" NAME; OUTPUT (CHANNEL, "("8B,N")", TEXT);

    "PROCEDURE" HEADING;
    "IF" HEADINGCODE = 1 "THEN"
    "BEGIN" "INTEGER" J;
        OUTPUT (CHANNEL, "("/,8B")");
        "FOR" J:= L "STEP" 1 "UNTIL" U
        "DO" OUTPUT (CHANNEL, "("+.4D"+DD,2B")", COLNR [J]);
        OUTPUT (CHANNEL, "("/")")
    "END"
    "ELSE"
    "IF" HEADINGCODE = 2 "THEN"
    "BEGIN" "INTEGER" J;
        OUTPUT (CHANNEL, "("/,3B")");
        "FOR" J:= L "STEP" 1 "UNTIL" U
        "DO" OUTPUT (CHANNEL, "("11ZD")", COLNR [J]);
        OUTPUT (CHANNEL, "("/")")
    "END";

    "INTEGER" "PROCEDURE" LINENUMBER;
    "BEGIN" "INTEGER" L; SYSPARAM (CHANNEL, 3, L);
        LINENUMBER:= L + 1
    "END" LINENUMBER;

    "IF" CHANNEL <= 0 "OR" CHANNEL = 60 "THEN"
        STATAL3 ERROR("("RECTANGLE")", 1, CHANNEL);
    "IF" LOWVERT > UPPVERT "THEN"
        STATAL3 ERROR("("RECTANGLE")", 4, LOWVERT);
    "IF" LOWHOR > UPPHOR "THEN"
        STATAL3 ERROR("("RECTANGLE")", 6, LOWHOR);
    SYSPARAM(CHANNEL,1,I);
    "IF" I > 0 "THEN" OUTPUT(CHANNEL, "("/")");
    NK:= UPPVERT - LOWVERT + 1; NK:= NK + NK // 10;
    MM:= UPPHOR - LOWHOR + 1;
    "IF" LINENUMBER > 1 "THEN"
    "BEGIN" "IF" MM < 11 "AND"
             LINENUMBER <=
             ("IF" HEADINGCODE = 1 "OR" HEADINGCODE = 2
              "THEN" 54 "ELSE" 56) - NK
        "THEN" OUTPUT (CHANNEL, "("3/")")
        "ELSE" OUTPUT (CHANNEL, "("*")");

    "END";
    NAME;
    "IF" MM < 11 "AND"
         LINENUMBER <=
         ("IF" HEADINGCODE = 1 "OR" HEADINGCODE = 2
          "THEN" 57 "ELSE" 59) - NK
    "THEN" OUTPUT (CHANNEL, "("/")")
    "ELSE" NUMBER;
    L:= LOWHOR;
IN: U:= "IF" UPPHOR - L > 9 "THEN" L + 9 "ELSE" UPPHOR;
    HEADING;
    "FOR" I:= LOWVERT "STEP" 1 "UNTIL" UPPVERT "DO"
    "BEGIN" OUTPUT (CHANNEL, "("/,2ZD,5B")", ROWNR [I]);
        "FOR" J:= L "STEP" 1 "UNTIL" U
        "DO"
        OUTPUT (CHANNEL, "("+.4D"+DD,BB")", MATRIX [I, J]);
        "IF" I // 10 * 10 = I "THEN"
        "BEGIN" OUTPUT (CHANNEL, "("/")");
            "IF" I // 50 * 50 = I "AND" I < UPPVERT "THEN"
            "BEGIN" NEW; HEADING "END";
        "END"
    "END";
    "IF" U < UPPHOR "THEN"
    "BEGIN" NEW; L:= U + 1; "GOTO" IN "END";
    RECTANGLE:= PAGE;
"END" RECTANGLE;
         "EOP"
```

TITLE: **Printpc**

AUTHOR: R. Kaas

INSTITUTE: Mathematical Centre

RECEIVED: 740701

BRIEF DESCRIPTION

A pairchart is printed via a lineprinter. A pairchart is a diagram representing the rank properties of two samples. Wilcoxon's, Wald-Wolfowitz', Ansari-Bradley's, Kolmogorov-Smirnov's and other test statistics are easily computed using a pairchart.

KEYWORDS

Pairchart via a lineprinter

CALLING SEQUENCE

Heading

```
"PROCEDURE" PRINTPC (CHN, XI, I, LX, UX, YJ, J, LY, UY, SORTED, CODE);
"VALUE" CHN, CODE, LX, UX, LY, UY;
"INTEGER" I, J, LX, UX, LY, UY, CHN;
"REAL" XI, YJ, CODE;
"BOOLEAN" SORTED;
"CODE" 47001;
```

Formal parameters

CHN: <integer arithmetic expression>, channel number via which the output is written to file;

XI: <arithmetic expression>, observations of the first sample, depending on the Jensen parameter I;

I: <integer variable>, index of the first sample, Jensen parameter for XI;

LX: <integer arithmetic expression>, smallest index of the first sample;

UX: <integer arithmetic expression>, largest index of the first sample;

YJ: <arithmetic expression>, observations of the second sample, depending on Jensen parameter J;

J: <integer variable>, index of the second sample, Jensen parameter for YJ;

LY: <integer arithmetic expression>, smallest index of the second sample;

UY: <integer arithmetic expression>, largest index of the second sample;

SORTED: <boolean expression>, indicating whether the samples are sorted in non-decreasing order or not;

CODE: <arithmetic expression>, identification number of the problem.

Data and results

If the smallest sample size (i.e. MIN(UX−LX+1, UY−LY+1)) is larger than 44, the samples are too large for a pairchart over a lineprinter and the execution of the program is terminated. In this case one is advised to use the procedure PLOTPC, which can handle any sample size.
The following error messages may appear:

Errornumber 1	(if CHN ⩽0 or CHN=60)
Errornumber 4	(if LX >UX)
Errornumber 8	(if LY >UY)

Procedures used

VEC QSORT	STATAL 11020
STATAL3 ERROR	STATAL 40100

Language

Algol 60

Method and performance

A cumulative frequency table of the pooled sample is made. In two arrays printing orders are collected. The pairchart is printed from left to right and from top to bottom according to these orders.

Reference

[1]. D. Quade,
The pairchart,
Statistica Neerlandica, 27, (1973), pp. 29-45.

Example of use

Program:

```
"BEGIN"
    "ARRAY" X, Y[1:7]; "INTEGER" J, K;
    INARRAY(60,X); INARRAY(60,Y);
    CHANNEL(71, "("E")", 61);
    PRINTPC(71, X[J], J, 1, 7, Y[K], K, 1, 7, "TRUE", 1)
"END"
```

Input:

```
1  2  7  9  10  12
2  3  4  6   8  10  11  12
```

Output:

```
PAIRCHART WITH PROBLEM:        1


+    +    +    +    +    +    +    +
                                   I
                                   I
+    +    +    +    +    +    +----+
                              I    I
                              I    I
+    +    +    +    +----+----+----+
                    I    I
                    I    I
+    +    +    +    +----+    +    +
                    I
                    I
+    +    +    +----+    +    +    +
               I
               I
+----+----+----+    +    +    +    +
I
I
+    +    +    +    +    +    +    +
I
I
+    +    +    +    +    +    +    +
```

SOURCE TEXT

```
"CODE" 47001;
"PROCEDURE" PRINTPC(KN, XI, IX, XLOW, XUP, YI, IY, YLOW,
                                    YUP, SORTED, CODENR);
    "VALUE" KN, XLOW, XUP, YLOW, YUP, CODENR, SORTED;
    "REAL" XI, YI, CODENR; "BOOLEAN" SORTED;
    "INTEGER" KN, IX, XLOW, XUP, IY, YLOW, YUP;
"BEGIN" "INTEGER" N, M, MN, A, B, NMAX; "BOOLEAN" WSL;

    "PROCEDURE" FREQTAB(A, LOW, UPP, FR); "VALUE" LOW, UPP;
        "ARRAY" A; "INTEGER" "ARRAY" FR; "INTEGER" LOW, UPP;
    "BEGIN" "INTEGER" IO, I; "REAL" T; I:= LOW;
        "FOR" IO:= I "WHILE" I <= UPP "DO"
        "BEGIN" T:= A[I];
            "FOR" I:= I + 1 "WHILE" ("IF" I <= UPP
                                     "THEN" A[I]=T
                                     "ELSE" "FALSE") "DO";
            FR[IO]:= I - IO
        "END"
    "END" FREQTAB;

    "PROCEDURE" FORMAT;

    "BEGIN" "INTEGER" MAX, MIN; MIN:= "IF" M<N "THEN" M
                                               "ELSE" N;
        MAX:= MN-MIN;
        "IF" MAX<27
        "THEN" "BEGIN" A:= 4; WSL:= N>M; N:= MIN "END""ELSE"
        "IF" MAX<45
        "THEN" "BEGIN" A:= 2; WSL:= N>M; N:= MIN "END""ELSE"
        "IF" MIN<27
        "THEN" "BEGIN" A:= 4; WSL:= M>N; N:= MAX "END""ELSE"
        "IF" MIN<45
        "THEN" "BEGIN" A:= 2; WSL:= M>N; N:= MAX "END""ELSE"
        "BEGIN" OUTPUT( 61, "(" ///,
"("SAMPLES TOO LARGE FOR PAIRCHART OVER LINEPRINTER")" ")");
            "GOTO" EXIT
        "END";
        M:= MN-N; B:= A/2
    "END" FORMAT;

    "PROCEDURE" STRPL;
    "IF" A=4 "THEN" OUTPUT( KN, "(""("----+")"")")
             "ELSE" OUTPUT( KN, "(""("--+")"")");

    "PROCEDURE" BPL;
    "IF" A=4 "THEN" OUTPUT( KN, "(""("    +")"")")
             "ELSE" OUTPUT( KN, "(""("  +")"")");


    "IF" KN <= 0 "OR" KN = 60 "THEN"
        STATAL3 ERROR("("PRINTPC")", 1, KN);
    "IF" XLOW > XUP "THEN"
        STATAL3 ERROR("("PRINTPC")", 4, XLOW);
    "IF" YLOW > YUP "THEN"
        STATAL3 ERROR("("PRINTPC")", 8, YLOW);
    N:= XUP-XLOW + 1; M:= YUP-YLOW + 1; MN:= M + N;
    NMAX:= "IF" N>M "THEN" N "ELSE" M;
    FORMAT;

    "BEGIN" "ARRAY" XOBS[1:N], YOBS[1:M];
        "INTEGER" J, K, N1, N2, Q1, Q2, FX, FY, I, SGN, Q;
        "INTEGER" "ARRAY" XFR[1:N], YFR[1:M],
                          PRX[1:N,1:2], PRY[1:M,1:2];

        IX:= XLOW; IY:= YLOW; I:= 0;
        "IF" WSL "THEN"
        "BEGIN" "FOR" I:= I + 1 "WHILE" IX<= XUP "DO"
            "BEGIN" YOBS[I]:= XI; IX:= IX + 1 "END"; I:= 0;

            "FOR" I:= I + 1 "WHILE" IY<= YUP "DO"
            "BEGIN" XOBS[I]:= YI; IY:= IY + 1 "END"
        "END" "ELSE"
        "BEGIN" "FOR" I:= I + 1 "WHILE" IX<= XUP "DO"
            "BEGIN" XOBS[I]:= XI; IX:= IX + 1 "END"; I:= 0;
            "FOR" I:= I + 1 "WHILE" IY<= YUP "DO"
            "BEGIN" YOBS[I]:= YI; IY:= IY + 1 "END"
        "END";

        "IF" "NOT" SORTED "THEN"
        "BEGIN" VEC QSORT(XOBS, 1, N);
                VEC QSORT(YOBS, 1, M) "END";
        FREQTAB( XOBS, 1, N, XFR);
        FREQTAB( YOBS, 1, M, YFR);

        "FOR" K:= 1 "STEP" 1 "UNTIL" N "DO" PRX[K,2]:= MN;
        "FOR" K:= 1 "STEP" 1 "UNTIL" M "DO" PRY[K,2]:= MN;
        J:= K:= 0;
NEXTOBS:
        SGN:= SIGN( XOBS[K + 1] - YOBS[J + 1]);
        FX:= XFR[K + 1]; FY:= YFR[J + 1];

        "IF" SGN=-1 "THEN"
        "BEGIN"
            "FOR" Q:= 1 "STEP" 1 "UNTIL" FX
            "DO" PRX[K + Q,1]:= J;
            K:= K + FX
        "END" "ELSE" "IF" SGN=1 "THEN"
        "BEGIN"
            "FOR" Q:= 1 "STEP" 1 "UNTIL" FY
            "DO" PRY[J + Q,1]:= K;
            J:= J + FY
        "END" "ELSE"
        "BEGIN" "FOR" Q:= 1 "STEP" 1 "UNTIL" FX "DO"
            "BEGIN" PRX[K + Q,1]:= J;
                    PRX[K + Q,2]:=J + FY "END";
            "FOR" Q:= 1 "STEP" 1 "UNTIL" FY "DO"
            "BEGIN" PRY[J + Q,1]:= K;
                    PRY[J + Q,2]:=K + FX "END";
            J:= J + FY; K:= K + FX;
        "END";
        "IF" K<N & J<M
        "THEN" "GOTO" NEXTOBS "ELSE" "IF" K<N "THEN"
        "BEGIN"
        "FOR" Q:=K+1 "STEP" 1 "UNTIL" N
        "DO" PRX[Q,1]:= M "END"
        "ELSE" "IF" J<M "THEN"
        "FOR" Q:= J + 1 "STEP" 1 "UNTIL" M
        "DO" PRY[Q,1]:= N;
        OUTPUT( KN, "("*")");
        "IF" CODENR > 0 "THEN"
        OUTPUT( KN, "(""("PAIRCHART WITH PROBLEM:")", 7ZD,
             ///")", CODENR );
        SYSPARAM( KN, 8, 66);

        "FOR" Q:= N "STEP" -1 "UNTIL" 1 "DO"
        "BEGIN" OUTPUT( KN, "("/, "("+")"")");
            "FOR" K:= 1 "STEP" 1 "UNTIL" M "DO"
            "IF" PRY[K,1]=Q ! PRY[K,2]=Q
            "THEN" STRPL "ELSE" BPL;
            "FOR" K:= 1 "STEP" 1 "UNTIL" B "DO"
            "BEGIN" SYSPARAM( KN, 2, PRX[Q,1]*(A + 1));
                OUTPUT( KN, "(""("I")"")");
                "IF" PRX[Q,2]<MN "THEN"
                "BEGIN" SYSPARAM( KN, 2, PRX[Q,2]*(A + 1));
                    OUTPUT( KN, "(""("I")"")")
                "END"
            "END"
        "END";
LAST LINE:
        OUTPUT( KN, "(" /, "("+")" ")");
        "FOR" K:= 1 "STEP" 1 "UNTIL" M "DO"
        "IF" PRY[K,1]=0 "THEN" STRPL "ELSE" BPL;

        SYSPARAM( KN, 8, 60);
    "END";
EXIT:
"END" PRINTPC;
         "EOP"
```

TITLE: **Plotpc**

AUTHOR: R. Kaas

INSTITUTE: Mathematical Centre

RECEIVED: 740701

BRIEF DESCRIPTION

A pairchart is written to a graphfile which can be plotted on a plotter. A pairchart is a diagram representing the rank properties of two samples. Wilcoxon's, Wald-Wolfowitz' Ansari-Bradley's, Kolmogorov-Smirnov's and other test statistics are easily computed using a pairchart.

KEYWORDS

Pairchart via a plotter

CALLING SEQUENCE

Heading

```
"PROCEDURE" PLOTPC (GRFILE, XI, I, LX, UX, YJ, J, LY, UY, SORTED, CODE);
"VALUE" CODE, LX, UX, LY, UY;
"INTEGER" I, J, LX, UX, LY, UY;
"REAL" XI, YJ, CODE;
"BOOLEAN" SORTED;
"STRING" GRFILE;
"CODE" 47002;
```

Formal parameters

GRFILE: <string>, name of the file on which the plot must be written as a maingraph. If the string GRFILE is empty, then the name of this file is "GRFILE".

XI: <arithmetic expression>, observations of the first sample, depending on the Jensen parameter I;

I: <integer variable>, index of the first sample, Jensen parameter for XI;

LX: <integer arithmetic expression>, smallest index of the first sample;

UX: <integer arithmetic expression>, largest index of the first sample;

YJ: <arithmetic expression>, observations of the second sample, depending on the Jensen parameter J;

J: <integer variable>, index of the second sample, Jensen parameter for YJ;

LY: <integer arithmetic expression>, smallest index of the second sample;

UY: <integer arithmetic expression>, largest index of the second sample;

SORTED: <boolean expression>, indicating whether the samples are

	sorted in non-decreasing order or not;
CODE:	<arithmetic expression>, identification number of the problem. If CODE≤0, the identification is suppressed.

DATA AND RESULTS

The pairchart is written as a main graph to a graphfile. In subsequent calls of PLOTPC, the same name may be choosen for this file. The pairchart is plotted in a grid, if the largest sample size is larger than 50, the grid is omitted.
The following error messages may appear:

Errornumber 4	(if LX >UX)
Errornumber 8	(if LY >UY)

PROCEDURES USED

PLOT	CALCOMP
PLOTS	CALCOMP
FACTOR	CALCOMP
SYMBOL	CALCOMP
NUMBER	CALCOMP
GRID	CALCOMP
RECT	CALCOMP
DASHP	CALCOMP
VEC QSORT	STATAL 11020
STATAL3 ERROR	STATAL 40100

LANGUAGE

Algol 60

METHOD AND PERFORMANCE

A cumulative frequency table of the pooled sample is made. The pairchart is plotted from the left-lower to the right-upper corner.

REFERENCES

[1]. D. Quade,
The pairchart,
Statistica Neerlandica, 27, (1973), pp. 29-45

[2]. *Graphics, SARA publikatie 11*
Stichting Academisch Rekencentrum Amsterdam, Amsterdam, 1977

Example of use

Program:

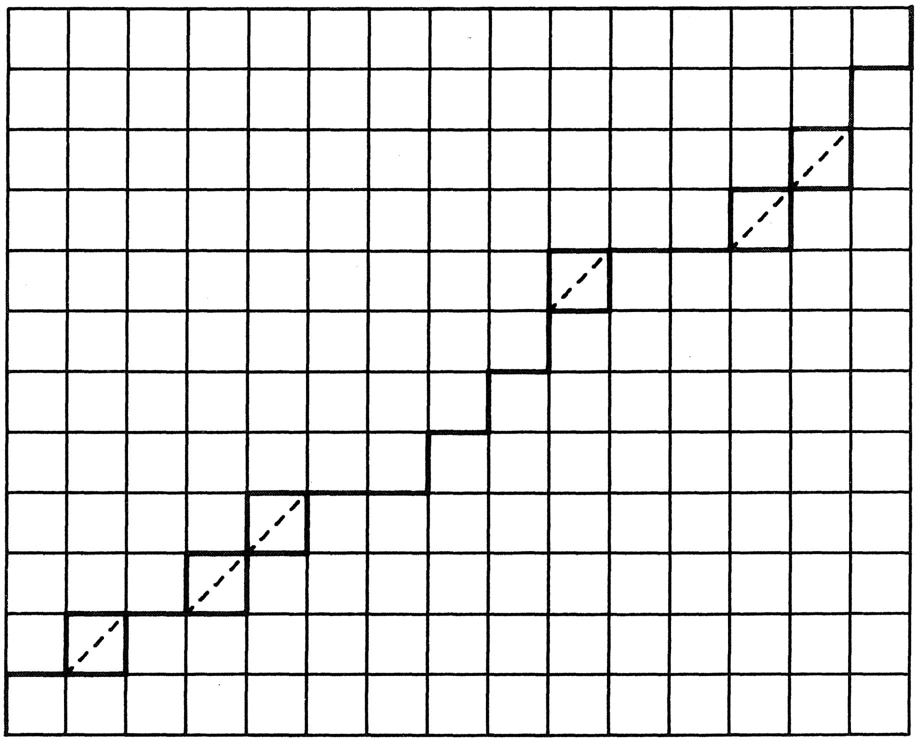
```
"BEGIN"
    "ARRAY" X[1:12], Y[1:15]; "INTEGER" J, K;
    INARRAY(60, X); INARRAY(60, Y);
    PLOTPC("(""")", X[J], J, 1, 12, Y[K], K, 1, 15, "TRUE", 1)
"END"
```

Input:

```
1  4  8  10  14  16  19  20  25  27  30  32
2  4  6   8  10  11  12  15  18  20  22  24  25  27  31
```

Output:

```
PAIRCHART WITH PROBLEM: 1
```



SOURCE TEXT

```
"CODE" 47002;
"PROCEDURE" PLOTPC(GRFILE, XI, IX, XLOW, XUP, YI, IY, YLOW,
                YUP, SORTED, CODENR);
    "VALUE" XLOW, XUP, YLOW, YUP, CODENR, SORTED;
    "INTEGER" XLOW, XUP, YLOW, YUP, IX, IY;
    "BOOLEAN" SORTED;
    "REAL" XI, YI, CODENR; "STRING" GRFILE;
"BEGIN" "INTEGER" N, M, NMAX, I, J, K, SGN, FX, FY;
    "ARRAY" XOBS[1:XUP - XLOW + 1], YOBS[1:YUP - YLOW + 1];
    "INTEGER" "ARRAY" XFR[1:XUP-XLOW+1], YFR[1:YUP-YLOW+1];
    "REAL" EPS, DASH, TEPS, FACT, LINETHICKNESS;

    "PROCEDURE" HORTHICK(A, B, C);
         "VALUE" A, B, C;
         "REAL" A, B, C;
    "BEGIN" "REAL" AME, BME, BPE, CPE;
        AME:= A - EPS; BME:= B - EPS;
        BPE:= B + EPS; CPE:= C + EPS;
        PLOT(AME, BPE, 3); PLOT(CPE, BPE, 2);
        PLOT(CPE, BME, 3); PLOT(AME, BME, 2);
    "END" HORTHICK;

    "PROCEDURE" VERTTHICK(A, B, C);
         "VALUE" A, B, C;
         "REAL" A, B, C;
    "BEGIN" "REAL" AME, BME, CPE, APE;

        AME:= A - EPS; BME:= B - EPS; CPE:= C + EPS;
        APE:= A + EPS;
        PLOT(AME, BME, 3); PLOT(AME, CPE, 2);
        PLOT(APE, CPE, 3); PLOT(APE, BME, 2);
    "END" VERTTHICK;

    "PROCEDURE" FREQTAB(A, LOW, UPP, FR);
         "VALUE" LOW, UPP;
         "ARRAY" A;
         "INTEGER" "ARRAY" FR;
         "INTEGER" LOW, UPP;
    "BEGIN" "INTEGER" IO, I; "REAL" T;
        "FOR" I:= LOW, I "WHILE" I <= UPP "DO"
        "BEGIN" T:= A[I]; IO:= I;
            "FOR" I:= I + 1 "WHILE" ("IF" I <= UPP
                                     "THEN" A[I]=T
                                     "ELSE" "FALSE") "DO";
            FR[IO]:= I - IO
        "END"
    "END" FREQTAB;

    LINETHICKNESS:= 0.015;

    "IF" XLOW > XUP
    "THEN" STATAL ERROR("("PLOTPC")", 4, XLOW);
```

```
"IF" YLOW > YUP
"THEN" STATAL ERROR("("PLOTPC")", 8, YLOW);
N:= XUP-XLOW+1; M:= YUP-YLOW+1;
NMAX:= "IF" N>M "THEN" N "ELSE" M;
IX:= XLOW; IY:= YLOW; I:= 0;
"FOR" I:= I+1 "WHILE" IX<= XUP "DO"
"BEGIN" XOBS[I]:= XI; IX:= IX+1 "END";
I:= 0;
"FOR" I:= I+1 "WHILE" IY<= YUP "DO"
"BEGIN" YOBS[I]:= YI; IY:= IY+1 "END";
"IF" "NOT" SORTED "THEN"
"BEGIN" VEC QSORT(XOBS, 1, N);
        VEC QSORT(YOBS, 1, M) "END";
FREQTAB( XOBS, 1, N, XFR);
FREQTAB( YOBS, 1, M, YFR);

K:= EQUIV(GRFILE);
"IF" K = EQUIV("("")")
"THEN" K:= EQUIV("("GRFILE")");
PLOTS(0, 0, K * 4096);
FACT:= 12 / NMAX; FACTOR(FACT);
"IF" CODENR > 0 "THEN"
"BEGIN" SYMBOL(0, N + 1.5, 0.5, 80, 0, -1);
    "FOR" K:= 65, 73, 82, 67, 72, 65, 82, 84, 32,
              87, 73, 84, 72, 32,
              80, 82, 79, 66, 76, 69, 77, 58, 32
    "DO"
    SYMBOL(999, 999, 0.5, K, 0, -1);
    NUMBER(999, 999, 0.5, CODENR, 0,
        "IF" CODENR = ENTIER(CODENR)
        "THEN"  -1 "ELSE" 4);
"END";
PLOT(0, 0, 3);

K:= J:= 0; EPS:= LINETHICKNESS / FACT;
DASH:= SQRT(2) / 8; TEPS:= 2 * EPS;
"IF" NMAX <= 50 "THEN"
"BEGIN" GRID(0, 0, 1, 1, M, N);
NEXT:
    SGN:= SIGN(XOBS[K + 1] - YOBS[J + 1]);
    FX:= XFR[K + 1]; FY:= YFR[J + 1];

    "IF" SGN = 1 "THEN"
    "BEGIN" HORTHICK(J, K, J + FY); J:= J + FY "END"
    "ELSE"
    "IF" SGN = -1 "THEN"
    "BEGIN" VERTTHICK(J, K, K + FX); K:= K + FX "END"
    "ELSE"
    "BEGIN"
        RECT(J + EPS, K + EPS, FY - TEPS, FX - TEPS,
                                                0, 3);
        RECT(J - EPS, K - EPS, FY + TEPS, FX + TEPS,
                                                0, 3);
        K:= K + FX; J:= J + FY; DASHP(J, K, DASH);
        "END";

        "IF" K = N "THEN" HORTHICK(J, N, M) "ELSE"
        "IF" J = M "THEN" VERTTHICK(M, K, N)
                   "ELSE" "GOTO" NEXT;
    "END" "ELSE"
    "BEGIN" RECT(0, 0, N, M, 0, 3);
NEXT1:
        SGN:= SIGN(XOBS[K + 1] - YOBS[J + 1]);
        FX:= XFR[K + 1]; FY:= YFR[J + 1];

        "IF" SGN = 1 "THEN"
        "BEGIN" J:= J + FY; PLOT(J, K, 2) "END" "ELSE"

        "IF" SGN = -1 "THEN"

        "BEGIN" K:= K + FX; PLOT(J, K, 2) "END" "ELSE"

        "BEGIN" RECT(J, K, FY, FX, 0, 3);
            K:= K + FX; J:= J + FY; DASHP(J, K, DASH);
        "END";

        "IF" K < N "AND" J < M "THEN" "GOTO" NEXT1
                               "ELSE" PLOT(M, N, 2);
    "END";
"END" PLOTPC;
"EOP"
```

TITLE: **Scatterprint**

AUTHOR: H. Elffers

INSTITUTE: Mathematical Centre

RECEIVED: 740501

BRIEF DESCRIPTION
The procedure produces a two dimensional scatterdiagram via a lineprinter.

KEYWORDS
Scatterdiagram via a lineprinter

CALLING SEQUENCE
Heading

```
"PROCEDURE" SCATTERPRINT (XI, XMIN, XMAX, XTEXT, YI, YMIN, YMAX, YTEXT,
I, L, U, SCALE, LAYOUT, CHN, CODE);
"VALUE" XMIN, XMAX, YMIN, YMAX, L, U, SCALE, LAYOUT, CHN, CODE;
"REAL" XI, XMIN, XMAX, YI, YMIN, YMAX;
"INTEGER" I, L, U, SCALE, LAYOUT, CHN, CODE;
"STRING" XTEXT, YTEXT;
"CODE" 47000;
```

Formal parameters

XI: <arithmetic expression>, first coordinate of the observations, depending on the Jensen parameter I;

YI: <arithmetic expression>, second coordinate of the observations, depending on the Jensen parameter I;

XMIN: <arithmetic expression>, lower bound for first coordinates;

XMAX: <arithmetic expression>, upper bound for first coordinates;

YMIN: <arithmetic expression>, lower bound for second coordinates;

YMAX: <arithmetic expression>, upper bound for second coordinates;

XTEXT: <string>, label for the first coordinate;

YTEXT: <string>, label for the second coordinate;

I: <integer variable>, Jensen parameter for XI and YI;

L: <integer arithmetic expression>, smallest index for I;

U: <integer arithmetic expression>, largest index for I;

SCALE: <integer arithmetic expression>, to specify the length of the units along the axes;

LAYOUT: <integer arithmetic expression>, to specify the numbering along the axes;

CHN: <integer arithmetic expression>, channel number via which the output is written to file;

CODE: <integer arithmetic expression>, indentification number of the problem.

DATA AND RESULTS

SCATTERPRINT produces a two dimensional scatterdiagram of the observations (XI, YI), for I=L, L+1,...,U on a lattice of P−9 horizontal and PP−6 vertical positions, numbered 0,1,...,P−10 and 0,1,...,PP−7 respectively. Here P and PP are the boundaries of horizontal and vertical control, to be specified on the channel card (see CDC, ALGOL 60 r.m. version 5). The default values are P=136 and PP=60.

The diagram is made as large as possible under the restrictions set by the parameters XMIN, XMAX, YMIN, YMAX and SCALE.

Linear transformations F and G are constructed such that

(1) F (XMIN) = G (YMIN) = 0.

(2) F (XMAX) ⩽ P−10 and G (YMAX) ⩽ PP−7.

(3) F (XMAX) and G (YMAX) are maximal under the restrictions (4).

In this determination the parameters XMIN, YMIN, XMAX, YMAX are possibly altered. Thus, if XMIN equals XMAX the procedure assigns MIN{XI: L⩽ I ⩽U} to XMIN and MAX{XI: L ⩽ I ⩽U} to XMAX. Analogously, if YMIN equals YMAX the minimum and maximum of the YI are assigned to YMIN and YMAX.

Possibly these values are rounded off according to (5).

A picture is then drawn with an X-axis from F(XMIN) to F(XMAX) and an Y-axis from G(YMIN) to G(YMAX), in which the points (F(XI), G(YI)) are drawn, after rounding off to the nearest lattice point.

The axes are provided with numbering in the original units and with various explications of used units etc. It is emphasized that points whose coordinates do not fall between the thus altered parameters XMIN, XMAX or YMIN, YMAX are not represented in the picture.

These extreme values, however, need not to be reached.

The parameter SCALE can take the values 0, 1, 10, 11, 100, 101, 110. It is decomposed into COMPAR * 100 + XSCALE * 10 + YSCALE. These are used to choose an unit for the axes, i.e.

F(1)−F(0) and G(1)−G(0).

(4) For both axes there are two possibilities:

- No restrictions on the unit, asked for by specification XSCALE=0 (YSCALE = 0). This gives the biggest picture, the units are not, however, nice numbers.
- The unit is bound to be an integer or the reciprocal of an integer.

(5) As a side effect the values of XMIN (YMIN) and XMAX (YMAX) are rounded off to integers. This is asked for by specifying XSCALE=1 (YSCALE=1). By specifying COMPAR=1 the scales of both axes become comparable (i.e. 1 cm on the X-axis represents the same difference in X as 1 cm on the Y-axis in Y. On account of different size of width of position and height of a line, this results in unequal units for both axes. Note that this feature is printer dependent). It is not

possible to ask for comparability and nice units on both axes at the same time.
It is strongly recommended to use the comparability feature only when X and Y are in reality measured on the same scale, and their ranges obey RANGE(X)/3⩽RANGE(Y)⩽RANGE(X), otherwise very odd shaped pictures might be obtained. Select COMPAR=0 if comparability is not asked for. Selection of XSCALE=1 or YSCALE=1 diminishes the size of the picture somewhat.
The parameter layout is decomposed into XLAYOUT * 10 +YLAYOUT.
XLAYOUT and YLAYOUT are the number of digits used to print the numbering along the axes.
All numbering is without decimal point, whose position is separately given, if necessary. For the numbering of the X-axis a maximum of 8 digits may be used, for the Y-axis 5 digits.
Specifying of XLAYOUT=0 (YLAYOUT=0) results in rounding off the number to be printed to an integer, if possible in the tolerated number of digits.
Specification of LAYOUT<0 gives maximum number of digits on both axes.
The last digit of the numbering of the horizontal axis is placed at the lattice point which stands for this number.

The following restrictions must be obeyed:

A) XMIN ⩽ XMAX and YMIN ⩽ YMAX. if not, a message is given and SCATTER PRINT terminates.
B) When XMIN equals XMAX and YMIN equals YMAX, repeated evaluations of XI (YI) for the same I must give the same results. If not, a not intended scatterdiagram may be produced, without any alarm, or an ALGOL runtime error may occur.
C) P⩾20 and PP⩾10.
If not, a message is given and SCATTERPRINT terminates.
D) P⩾45+ MAX (25, LENGTH(XTEXT),LENGTH(YTEXT)).
If not, titles below the picture cause page overflow.

SCATTERPRINT prints the picture on a separate page on the specified channel. Each observation is represented by an asterisk. When more observations fall on the same lattice point the total number of them is recorded instead of an asterisk, unless this number exceeds 9, in which case a question mark is printed.
Afterwards, the first free position is the first position on the next page. Observations falling outside the picture are signalled.

Advice:
If almost nothing is known of the scatterdiagram to be produced, a safe choice for the parameters is XMIN=YMIN=XMAX=YMAX=SCALE=0 and LAYOUT=−1.

To avoid interference of the picture and the sara vignet on a lineprinter print, the horizontal control boundary of the output channel can be set, using the system procedure SYSPARAM, smaller than 120.

The following error message may appear:
Errornumber 14 (if CHN ≤0 or CHN =60)

Procedures used

STATAL3 ERROR STATAL 40100

Language

Agol 60

Method and performance

An array of P*PP positions is declared.
The running time depends on the number of observations, the evaluation time of each observation and on the differences (XMIN−XMAX) and (YMIN−YMAX).
The print of a picture takes about 2 central processor seconds.

Reference

[1]. ALGOL 60 reference manual, version 5,
Control Data Corporation, 1979.

Example of use

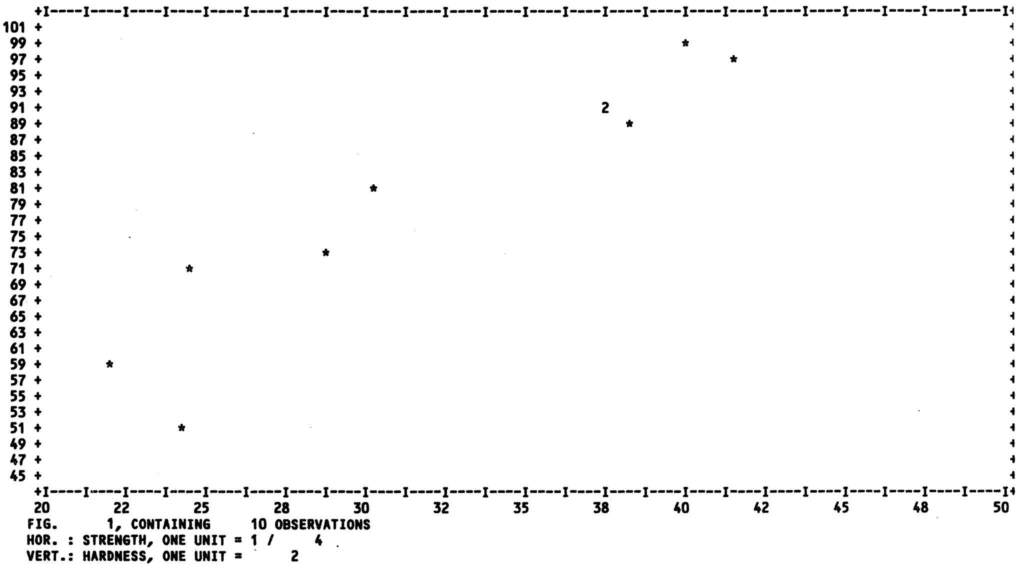
Program:

```
"BEGIN"
    "INTEGER" J; "ARRAY" H, S[1:10];
    INARRAY(60, S); INARRAY(60, H);;
    CHANNEL(71, "("E")", 61);
    SCATTERPRINT(S[J], 20, 50, "("STRENGTH")", H[J], 45, 100,
        "("HARDNESS")", J, 1, 10, 11, 23, 71, 1)
"END"
```

Input:

```
22.0  38.2  24.5  40.1  28.7  37.5  37.6  24.2  30.3  41.4
58.3  88.1  71.2  98.0  72.2  90.1  91.9  50.3  81.1  97.1
```

Output:



```
    +I----I----I----I----I----I----I----I----I----I----I----I----I----I----I----I----I----I----I----I----I----I----I----I----I+
101 +                                                                                                                         +
 99 +                                                                                *                                        +
 97 +                                                                                      *                                  +
 95 +                                                                                                                         +
 93 +                                                                                                                         +
 91 +                                                                      2                                                  +
 89 +                                                                         *                                               +
 87 +                                                                                                                         +
 85 +                                                                                                                         +
 83 +                                                                                                                         +
 81 +                                         *                                                                               +
 79 +                                                                                                                         +
 77 +                                                                                                                         +
 75 +                                                                                                                         +
 73 +                                   *                                                                                     +
 71 +                  *                                                                                                      +
 69 +                                                                                                                         +
 67 +                                                                                                                         +
 65 +                                                                                                                         +
 63 +                                                                                                                         +
 61 +                                                                                                                         +
 59 +        *                                                                                                                +
 57 +                                                                                                                         +
 55 +                                                                                                                         +
 53 +                                                                                                                         +
 51 +                 *                                                                                                       +
 49 +                                                                                                                         +
 47 +                                                                                                                         +
 45 +                                                                                                                         +
    +I----I----I----I----I----I----I----I----I----I----I----I----I----I----I----I----I----I----I----I----I----I----I----I----I+
    20        22        25        28        30        32        35        38        40        42        45        48        50
   FIG.      1, CONTAINING     10 OBSERVATIONS
   HOR. : STRENGTH, ONE UNIT = 1 /     4
   VERT.: HARDNESS, ONE UNIT =      2
```

Source text

```
"CODE" 47000;
"PROCEDURE" SCATTERPRINT(XI, XMIN, XMAX, XTEXT, YI, YMIN,
   YMAX, YTEXT, I, LB, UB, SCALE, LAYOUT, CHANNEL, NUMBER);
    "VALUE" XMIN, XMAX, YMIN, YMAX, LB, UB, NUMBER, SCALE,
            LAYOUT, CHANNEL;
    "REAL" XI, XMIN, XMAX, YI, YMIN, YMAX;
    "INTEGER" I, LB, UB, NUMBER, SCALE, LAYOUT, CHANNEL;
    "STRING" XTEXT, YTEXT;

"BEGIN" "REAL" XFAC, YFAC, RATIO, LN10INV;
"INTEGER" J, K, LOCAL, EXPX, EXPY, XLAYOUT, YLAYOUT,
          DEPTH, WIDTH, DEPTH1, WIDTH1, ERRORS;
"BOOLEAN" XINTSCALE, YINTSCALE, COMPAR;


"PROCEDURE" NEWPAGE IF NECESSARY(CHANNEL);
    "VALUE" CHANNEL; "INTEGER" CHANNEL;
"BEGIN" "INTEGER" RHOPRIME, PPRIME, RHO;
    SYSPARAM(CHANNEL,3,RHOPRIME);
    SYSPARAM(CHANNEL,7,PPRIME);

    "IF" PPRIME = RHOPRIME "THEN" OUTPUT(CHANNEL,"("/")")
    "ELSE" "IF" RHOPRIME = 0 "THEN"
    "BEGIN" SYSPARAM(CHANNEL,1,RHO);
        "IF" RHO > 0 "THEN" OUTPUT(CHANNEL,"("*")");
    "END"
    "ELSE" OUTPUT(CHANNEL,"("*")");
"END" NEWPAGE IF NECESSARY;
```

```
"PROCEDURE" INFER(ZI, ZMIN, ZMAX); "REAL" ZI, ZMIN, ZMAX;
"BEGIN" I:= LB;
    ZMIN:= ZMAX:= ZI;
    "FOR" I:= LB + 1 "STEP" 1 "UNTIL" UB "DO"
    "IF" ZI > ZMAX "THEN" ZMAX:= ZI "ELSE"
    "IF" ZI < ZMIN "THEN" ZMIN:= ZI;
"END" INFER;

"INTEGER" "PROCEDURE" BLOW UP(Z); "VALUE" Z; "REAL" Z;
BLOW UP:= "IF" Z < "-200 "THEN" -200
          "ELSE" LN(Z) * LN10INV + .50001;

"PROCEDURE" ROUND(FAC); "REAL" FAC;
FAC:= "IF" FAC < 1 "THEN" -1 / ENTIER(-1 / FAC)
                   "ELSE" ENTIER(FAC);

INITIALISATION:
    LN10INV:= .4342944819;
    RATIO:= .6; "COMMENT" RATIO IS PRINTERDEPENDENT;
    "IF" CHANNEL <= 0 "OR" CHANNEL = 60 "THEN"
        STATAL3 ERROR("("SCATTERPRINT")", 14, CHANNEL);
    SYSPARAM(CHANNEL,5,WIDTH); SYSPARAM(CHANNEL,7,DEPTH);
    WIDTH:= WIDTH - 10; WIDTH1:= WIDTH - 1;
    DEPTH:= DEPTH - 6; DEPTH1:= DEPTH - 1;
    ERRORS:= 0;

NEWPAGE IF NECESSARY(CHANNEL);
"IF" WIDTH < 10 ! DEPTH < 10 "THEN"
"BEGIN" OUTPUT(CHANNEL,
    "("5/"(" LACK OF SPACE FOR PROCEDURE  SCATTERPRINT")",
        "(" NUMBER ")",-5ZD, "(" ON CHANNEL ")",-5ZD,
        /"(" CHECK HORIZONTAL AND VERTICAL CONTROL ON")",
        "(" THIS CHANNEL")",
        /"(" MINIMA ARE: HORIZONTAL 20 POSITIONS,")"/,
         "("            VERTICAL   16 LINES")"//")",
        NUMBER,CHANNEL);
    "GOTO" ABANDONING SCATTERPRINT;
"END";

DERIVATION OF SCALE WISHES:
    COMPAR:= SCALE > 99;
    SCALE:= SCALE - SCALE // 100 * 100;
    YINTSCALE:= SCALE ^= SCALE // 2 * 2;
    XINTSCALE:= "IF" COMPAR & YINTSCALE "THEN" "FALSE"
                "ELSE" SCALE // 10 > 0;

DETERMINATION OF EXTREMA:
    "IF" XMIN = XMAX "THEN" INFER(XI, XMIN, XMAX);
    "IF" YMIN = YMAX "THEN" INFER(YI, YMIN, YMAX);
    "IF" XINTSCALE "THEN"
    "BEGIN" XMIN:= ENTIER(XMIN);
            XMAX:= -ENTIER(-XMAX) "END";
    "IF" YINTSCALE "THEN"
    "BEGIN" YMIN:= ENTIER(YMIN);
```

```
            YMAX:= -ENTIER(-YMAX) "END";

"IF" XMIN >= XMAX ! YMIN >= YMAX "THEN"
"BEGIN" OUTPUT(CHANNEL,
    "("5/"("ERROR IN PROCEDURE  SCATTERPRINT NUMBER: ")",
    -10ZD, /34B"("XMIN: ")",N,/34B"("XMAX: ")",N,
     /34B"("YMIN: ")",N,/34B"("YMAX: ")",N,3/")",
    NUMBER,XMIN,XMAX,YMIN,YMAX);
    "GOTO" ABANDONING SCATTERPRINT;
"END";

DERIVATION OF SUBSCRIPTION WISHES:
    "IF" LAYOUT < 0 "THEN"
    "BEGIN" XLAYOUT:= 8; YLAYOUT:= 5 "END"
    "ELSE"
    "BEGIN" XLAYOUT:= LAYOUT // 10;
        YLAYOUT:= LAYOUT - XLAYOUT * 10;
        "IF" YLAYOUT > 5 "THEN" YLAYOUT:= 5;
        "IF" XLAYOUT > 8 "THEN" XLAYOUT:= 8
    "END";

TRANSFORMATION FACTORS:
    XFAC:= WIDTH1 / (XMAX - XMIN);

    YFAC:= DEPTH1 / (YMAX - YMIN);
    "IF" YINTSCALE "THEN" ROUND(YFAC);
    "IF" XINTSCALE "THEN" ROUND(XFAC);
    "IF" COMPAR "THEN"
    "BEGIN" "IF" XFAC * RATIO < YFAC
            "THEN" YFAC:= XFAC * RATIO
            "ELSE" XFAC:= YFAC / RATIO;
        "IF" YINTSCALE "THEN"
        "BEGIN" ROUND(YFAC); XFAC:= YFAC / RATIO "END"
        "ELSE"
        "IF" XINTSCALE "THEN"
        "BEGIN" ROUND(XFAC); YFAC:= XFAC * RATIO "END"
    "END";

DEPTH1:= (YMAX - YMIN) * YFAC;
WIDTH1:= (XMAX - XMIN) * XFAC;

ACTUAL SUBSCRIPTION LAYOUT:
    LOCAL:= BLOW UP(ABS(YMAX));
    EXPY:= BLOW UP(ABS(YMIN));
    "IF" EXPY < LOCAL "THEN" EXPY:= LOCAL;

    EXPY:= "IF" YLAYOUT = 0
           "THEN" ("IF" EXPY < 6 "THEN" 0 "ELSE" 5 - EXPY)
           "ELSE" YLAYOUT - EXPY;
    LOCAL:= BLOW UP(ABS(XMAX));
    EXPX:= BLOW UP(ABS(XMIN));
    "IF" EXPX < LOCAL "THEN" EXPX:= LOCAL;
    EXPX:= "IF" XLAYOUT = 0
           "THEN" ("IF" EXPX < 9 "THEN" 0 "ELSE" 8 - EXPX)
```

```
            "ELSE" XLAYOUT - EXPX;

"BEGIN" "INTEGER" "ARRAY" POINT[0:WIDTH1,0:DEPTH1];

    "PROCEDURE" UP AND DOWN;
    "BEGIN"
        "PROCEDURE" UADLAYOUT;
        FORMAT("("7B"("+I")",X("("----I")"),X("("-")"),
              "("+")"/")", WIDTH1 // 5,
              WIDTH1 - WIDTH1 // 5 * 5);

        "PROCEDURE" UADDUMMY(ITEM); "PROCEDURE" ITEM; ;

        OUTLIST(CHANNEL, UADLAYOUT, UADDUMMY);
    "END" UP AND DOWN;

    "PROCEDURE" BODY;
    "BEGIN"
        "PROCEDURE" BODYLAYOUT;
        FORMAT("("X(-4ZDB"("+")",S/)")", DEPTH1 + 1);

        "PROCEDURE" BODYLIST(ITEM); "PROCEDURE" ITEM;
        "BEGIN" "INTEGER" B, D; "REAL" EXPO;
            EXPO:= 10 ** EXPY;
            "FOR" D:= DEPTH1 "STEP" -1 "UNTIL" 0 "DO"
            "BEGIN" ITEM((D / YFAC + YMIN) * EXPO);
                "FOR" B:= 0 "STEP" 1 "UNTIL" WIDTH1 "DO"
                OUTCHARACTER(CHANNEL,"(" *23456789?")",
                                         POINT[B,D] + 1);
                ITEM("("+")");
            "END"
        "END" BODYLIST;

        OUTLIST(CHANNEL, BODYLAYOUT, BODYLIST);


    "END" BODY;

    "PROCEDURE" COUNT(HOR,VERT);
    "VALUE" HOR,VERT; "INTEGER" HOR,VERT;
    "BEGIN" "INTEGER" REPETITION;
      "IF" HOR < 0 ! VERT < 0 ! HOR > WIDTH1 ! VERT > DEPTH1
        "THEN"
        "BEGIN" ERRORS:= ERRORS + 1;
            "IF" ERRORS = 1 "THEN"
            OUTPUT(CHANNEL,
                  "("5/"("* THE FOLLOWING OBSERVATIONS, ")",
                   "("INTENDED TO BE INCLUDED IN THE ")"
                   , "("NEXT SCATTERPRINT FALL OUTSIDE")",
                   "(" SPECIFIED RANGE *")"//,
                   "("ERRORNUMBER  JENSENVARIABLE I")",
                   21B"("XI")",21B"("YI")"//")");
            OUTPUT(CHANNEL,"("+9ZDBB,+14ZDBB,N,N,/")",
                   ERRORS,I,XI,YI);
    "END" "ELSE"
    "BEGIN" REPETITION:= POINT[HOR, VERT];
        "IF" REPETITION < 10 "THEN"
        POINT[HOR, VERT]:= REPETITION + 1
    "END"

"END" COUNT;

"INTEGER" "PROCEDURE" TRUE(BOOLEAN);
    "VALUE" BOOLEAN; "BOOLEAN" BOOLEAN;
TRUE:= "IF" BOOLEAN "THEN" 1 "ELSE" 0;

"PROCEDURE" DOWN;
"BEGIN"
    "PROCEDURE" DOWNLAYOUT;
    "BEGIN" HLIM(1, WIDTH1 + 10);
        FORMAT("("-7ZD,X(B-7ZD),
            /"("FIG.")"-5ZD,
            "(", CONTAINING ")"5ZD"(" OBSERVATIONS")",
            X("(", COMPARABLE SCALES")"),
            /"("HOR. : ")",N ")",
            WIDTH1 // 10, TRUE(COMPAR));
    "END" DOWNLAYOUT;

    "PROCEDURE" DOWNLIST(ITEM); "PROCEDURE" ITEM;
    "BEGIN" "INTEGER" B; "REAL" EXPO;
        EXPO:= 10 ** EXPX;
        "FOR" B:= 0 "STEP" 10 "UNTIL" WIDTH1 "DO"
        ITEM((B / XFAC + XMIN) * EXPO);
        ITEM(NUMBER); ITEM(UB - LB + 1 - ERRORS);
        ITEM(XTEXT);
        "IF" EXPX ^= 0 "THEN"
        "BEGIN" FORMAT("(""(" * "")"-XZD")",
                    -ENTIER(-LN(ABS(EXPX)) * LN10INV));
            ITEM(EXPX);
        "END";
        FORMAT("("N,N,X(5ZD),X(.7D"+3D),/"("VERT.: ")"
        ,N")", TRUE(XINTSCALE & XFAC > "-7),
        TRUE(^(XINTSCALE & XFAC > "-7)));
        ITEM("(", ONE UNIT =")");
        "IF" XINTSCALE & XFAC > 1 "THEN"
        "BEGIN" ITEM("(" 1 /")"); ITEM(XFAC) "END"
        "ELSE"
        "BEGIN" ITEM("(" ")"); ITEM(1/XFAC) "END";
        ITEM(YTEXT);
        "IF" EXPY ^= 0 "THEN"

        "BEGIN" FORMAT("(""(" * "")"-XZD")",
                    -ENTIER(-LN(ABS(EXPY)) * LN10INV));
            ITEM(EXPY);
        "END";
        FORMAT("("N,N,X(5ZD),X(.7D"+3D),/")",
        TRUE(YINTSCALE & YFAC > "-7),
        TRUE(^(YINTSCALE & YFAC > "-7)));
```

```
            ITEM("(", ONE UNIT =")");
            "IF" YINTSCALE & YFAC > 1 "THEN"
            "BEGIN" ITEM("(" 1 /")"); ITEM(YFAC) "END"
            "ELSE"
            "BEGIN" ITEM("(" ")"); ITEM(1/YFAC) "END";
        "END" DOWNLIST;

        OUTLIST(CHANNEL, DOWNLAYOUT, DOWNLIST);

    "END" DOWN;

"FOR" J:= 0 "STEP" 1 "UNTIL"  WIDTH1  "DO"
"FOR" K:= 0 "STEP" 1 "UNTIL"  DEPTH1  "DO" POINT[J,K]:= 0;
"FOR" I:= LB "STEP" 1 "UNTIL" UB "DO"
COUNT((XI - XMIN) * XFAC, (YI - YMIN) * YFAC);

        NEWPAGE IF NECESSARY(CHANNEL);
        UP AND DOWN;
        BODY;
        UP AND DOWN;
        DOWN;

"END";

ABANDONING SCATTERPRINT:
    NEWPAGE IF NECESSARY(CHANNEL);

"END" SCATTERPRINT;
         "EOP"
```

TITLE: **Histo**

AUTHOR: J. Bethlehem

INSTITUTE: Mathematical Centre

RECEIVED: 750601

BRIEF DESCRIPTION

The procedure produces a histogram via a lineprinter.

KEYWORDS

Histogram via a lineprinter

CALLING SEQUENCE

Heading

```
"PROCEDURE" HISTO (CHN, TEXT, FREQ, L, U); "VALUE" CHN, L, U;
"INTEGER" CHN, L, U;
"STRING" TEXT;
"INTEGER" "ARRAY" FREQ;
"CODE" 47003;
```

Formal parameters

CHN: <integer arithmetic expression>, channel number via which the output is written to file;

TEXT: <string>, identifying text, heading of the histogram;

FREQ: <integer array identifier>, one-dimensional array with frequencies, FREQ[I] contains the number of times item I is observed;

L: <integer arithmetic expression>, smallest index of FREQ;

U: <integer arithmetic expression>, largest index of FREQ;

DATA AND RESULTS

The procedure draws a picture with U−L+1 lines; line I has length FREQ[I], I=L,...,U.

The following error messages may appear:

Errornumber 1	(if CHN ≤0 or CHN =60)
Errornumber 3	(if some FREQ[I] <0)
Errornumber 4	(if L >U)

PROCEDURES USED

STATAL3 ERROR	STATAL 40100

LANGUAGE

Algol 60

Method and performance

The total number of observations and the largest frequency, `MAX`, are computed.
If `MAX` ⩾100, the scale is transformed.

Example of use

Program:

```
"BEGIN"
    "INTEGER" "ARRAY" A[0:15];
    ININTARRAY(60, A);
    CHANNEL(71, "("E")", 61);
    HISTO(71, "("EXAMPLE")", A, 0, 15)
"END"
```

Input:

```
3  17  28  39  44  53  65  73  67  60  49  38  27  16  2  1
```

Output:

```
TOTAL NUMBER OF ITEMS:      582


             EXAMPLE

             0        10        20        30        40        50        60        70        80        90        100
             +---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+

      0      +---
             :
             +---------+-------
             :
             +---------+---------+--------
             :
             +---------+---------+---------+---------
             :
             +---------+---------+---------+---------+----
             :
      5      +---------+---------+---------+---------+---------+---
             :
             +---------+---------+---------+---------+---------+---------+-----
             :
             +---------+---------+---------+---------+---------+---------+---------+---
             :
             +---------+---------+---------+---------+---------+---------+-------
             :
             +---------+---------+---------+---------+---------+---------+
             :
     10      +---------+---------+---------+---------+---------
             :
             +---------+---------+---------+--------
             :
             +---------+---------+-------
             :
             +---------+------
             :
             +--
             :
     15      +-
             :
```

Source text

```
"CODE" 47003;
"PROCEDURE" HISTO(CH, TEXT, A, LOW, UPP);
"VALUE" CH, LOW, UPP;
"INTEGER" CH, LOW, UPP;
"STRING" TEXT;
"INTEGER" "ARRAY" A;
"BEGIN"
    "INTEGER" I, J, AI, AI10, SUM, MAX, F, FACTOR,
              PAGEHEIGHT, PAGEWIDTH;

    "PROCEDURE" LAYOUT;
    FORMAT("("3/,20B,N,3/,20B,"("0")",10(9ZD),/,20B,"("+")",
              10("("---------+")"),2/")");

    "PROCEDURE" LIST(ITEM); "PROCEDURE" ITEM;
    "BEGIN" "INTEGER" I;
        ITEM(TEXT);
        "FOR" I:= 1 "STEP" 1 "UNTIL" 10 "DO"
        ITEM(I * 10 * FACTOR)
    "END" LIST;

    "IF" CH <= 0 "OR" CH = 60
    "THEN" STATAL3 ERROR("("HISTO")", 1, CH);
    "IF" LOW > UPP
    "THEN" STATAL3 ERROR("("HISTO")", 4, UPP - LOW);
    SUM:= MAX:= 0;
    "FOR" I:= LOW "STEP" 1 "UNTIL" UPP "DO"
    "BEGIN" AI:= A[I]; "IF" AI > MAX "THEN" MAX:= AI;
        "IF" AI > 0 "THEN" SUM:= SUM + AI "ELSE"
        "IF" AI < 0
        "THEN" STATAL3 ERROR("("HISTO")", 3, AI);
    "END";
    FACTOR:= "IF" MAX > 100 "THEN"
    2 ** ENTIER(LN((MAX - 1) / 100) / LN(2) + 1) "ELSE" 1;
    SYSPARAM(CH, 7, PAGEHEIGHT); SYSPARAM(CH, 8, 66);
    SYSPARAM(CH, 5, PAGEWIDTH); SYSPARAM(CH, 6, 136);
    SYSPARAM(CH, 3, I); SYSPARAM(CH, 1, J);
    "IF" I > 0 "OR" J > 0 "THEN" OUTPUT(CH, "("*")");
    OUTPUT(CH, "(""("TOTAL NUMBER OF ITEMS:")",7ZD,/")",
                                                      SUM);
    "IF" FACTOR > 1 "THEN"
    OUTPUT(CH, "(""("ONE MARK REPRESENTS")",3ZD,
                               "(" ITEMS")"")",FACTOR);
    OUTLIST(CH, LAYOUT, LIST);
    "FOR" I:= LOW "STEP" 1 "UNTIL" UPP "DO"
    "BEGIN" AI:= ENTIER(A[I] / FACTOR + .5);
        AI10:= AI // 10;
        "IF" I // 5 * 5 = I
        "THEN" OUTPUT(CH, "("-13ZD5B")", I)
        "ELSE" OUTPUT(CH, "("20B")");
        OUTPUT(CH, "(""("+")"")");
        "FOR" J:= 1 "STEP" 1 "UNTIL" AI10 "DO"
        OUTPUT(CH, "(""("---------+")"")");
        AI10:= 10 * AI10 + 1; "IF" AI >= AI10 "THEN"
        "FOR" J:= AI10 "STEP" 1 "UNTIL" AI "DO"
        OUTPUT(CH, "(""("-")"")");
        OUTPUT(CH, "("/,20B,"(":")",/")")
    "END";
    SYSPARAM(CH, 8, PAGEHEIGHT); SYSPARAM(CH, 6, PAGEWIDTH)
"END" HISTO;
         "EOP"
```

TITLE: **Probprint**

AUTHOR: E. Opperdoes

INSTITUTE: Mathematical Centre

RECEIVED: 750310

BRIEF DESCRIPTION

Probprint prints a probability plot. Probability plotting is a graphical technique for comparing an empirical distribution function with a given theoretical distribution function.

KEYWORDS

Probability plot via lineprinter

CALLING SEQUENCE

Heading

```
"PROCEDURE" PROBPRINT (CHN, OBS, L, U, TEXTH, SORTED, X, INVX, CDFX,
TEXTV, NPOINTS, A, B, CODE);
"VALUE" CHN, L, U, NPOINTS, A, B, CODE;
"INTEGER" CHN, L, U, NPOINTS, CODE;
"BOOLEAN" SORTED;
"REAL" X, INVX, CDFX, A, B;
"STRING" TEXTH, TEXTV;
"REAL" "ARRAY" OBS;
"CODE" 47004;
```

Formal parameters

CHN: <integer arithmetic expression>, channel number via which the output is written to file;

OBS: <array identifier>, observations of the sample, at exit containing the sorted sample;

L: <integer arithmetic expression>, smallest index of the sample;

U: <integer arithmetic expression>, largest index of the sample;

TEXTH: <string>, text to describe the horizontal axis (empirical distribution);

SORTED: <boolean variable>, indicating whether the observations are sorted in non-decreasing order or not;

X: <real variable>, argument of the theoretical cumulative distribution function, Jensen parameter for INVX and CDFX;

INVX: <arithmetic expression>, inverse of the theoretical cumulative distribution function, depending on X as argument;

CDFX: <arithmetic expression>, theoretical cumulative ditribution function, depending on X as argument;

TEXTV: <string>, text to describe the vertical axis (theoretical distribution);

NPOINTS: <integer arithmetic expression>, number of points to be printed;

A: <arithmetic expression>, indicating the type of plot, see method and performance;

B: <arithmetic expression>, indicating the type of plot, see method and performance;

CODE: <integer arithmetic expression>, identification number of the problem.

Data and results

After a call of PROBPRINT a probability plot is printed on one page. The parameter OBS is also used as an output parameter and contains at exit the sorted observations. SORTED obtains the value "TRUE".

If the empirical cumulative distribution function of the sample resembles the theoretical one, a straight line occurs in the probability print. **For a more detailed plot, use** PLOTDIST **(section 6.9.).**

The following error messages may appear:

Errornumber 1	(if CHN ≤0 or CHN =60)
Errornumber 4	(if L >U)
Errornumber 11	(if NPOINTS ≤0)
Errornumber 14	(if CODE <0)

Procedures used

VEC QSORT	STATAL 11020
STATAL3 ERROR	STATAL 40100
SCATTERPRINT	STATAL 47000

Language

Algol 60

Method and performance

Probprint prints a probability plot, consisting of the points (EINV(P), TINV(P)), for all P specified in the cases 1, 2, or 3 below. Here EINV is the inverse of the empirical cumulative distribution function of the sample, and TINV the inverse of the theoretical cumulative distribution function, given in INVX. The values of P are determined by the values of A and B. As follows:

Case 1: A = B.

P =K/(NPOINTS+1), for K=1, 2,...,NPOINTS. In this case the plotted values are real quantiles and the probability print becomes a quantile plot (cf. Wilk and Gnanadesikan, 1968)

Case 2: A < B.

P=T(A+(K−1)*(B−A)/NPOINTS), for K=1, 2,...,NPOINTS. T is the theoretical cumulative distribution function, given in CDFX. In this case the theoretical quantiles are equidistant.

Case 3: A > B.
P=E(OBS(K)) for K=1,...,N. Here OBS(1)<OBS(2) <···<OBS(N) are the ordered distinct observations, and E is the empirical cumulative distribution function of the sample. In this case the plotted emperical quantiles are the discontinuity points of the empirical cumulative distribution function.

In case 1 CDFX may have a dummy value, in case 3 both CDFX and NPOINTS may have dummy values.

The procedure needs about 55000B+4*(U−L+1) CM-words.

Reference

[1]. M.B. Wilk and R. Gnanadesikan,
Pobability plotting methods for the analysis of data,
Biometrika 55, (1968), pp. 1-17.

Example of use

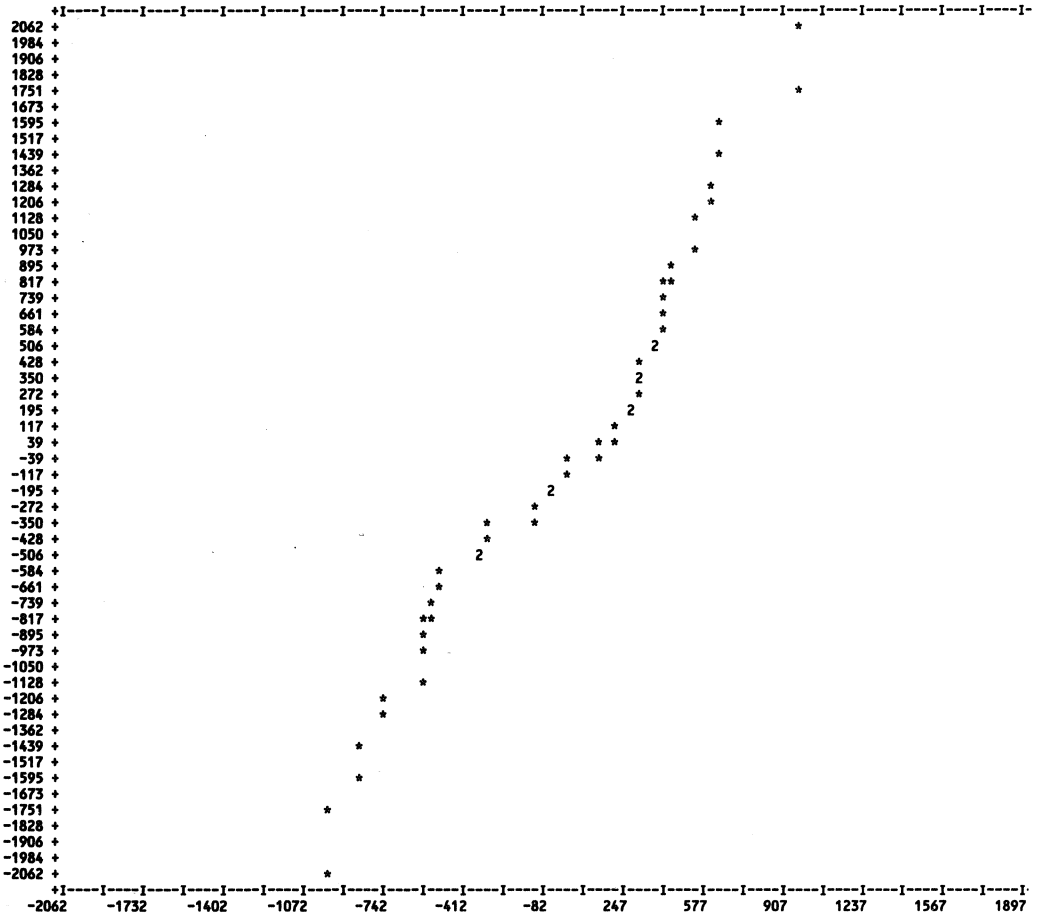
Program:

```
"BEGIN"
    "REAL" X; "BOOLEAN" SORTED; "ARRAY" S[1:25];
    INARRAY(60,S);
    SORTED:= "FALSE";
    CHANNEL(71, "("E")", 61);
    PROBPRINT(71, S, 1, 25, "("RANDOM SAMPLE")", SORTED, X,
     PHINV(X), PHI(X), "("NORMAL QUANTILES")", 50, 0, 0, 1)
"END"
```

Input:

```
 .29  -.73   .56   .43   .02
-.85   .41   .63  -.55   .96
 .30  -.58   .42  -.50  -.32
 .23  -.11  -.05   .39   .16
-.58   .31  -.97  -.36   .61
```

Output:



```
FIG.      1, CONTAINING     50 OBSERVATIONS
HOR. : RANDOM SAMPLE * "  3, ONE UNIT = .3299066"-001
VERT.: NORMAL QUANTILES * "  3, ONE UNIT = .7780817"-001
```

Source text

```
"CODE" 47004;
"PROCEDURE" PROBPRINT (CHANNEL, OBS, L, U, TEXT OBS, SORTED,
    X, CDFINV X, CDF X, TEXT TH, NPOINTS, MIN, MAX, NUMBER);
"VALUE" CHANNEL, L, U, NPOINTS, MIN, MAX, NUMBER;
"INTEGER" CHANNEL, L, U, NPOINTS, NUMBER; "BOOLEAN" SORTED;
"REAL" X, CDFINV X, CDF X, MIN, MAX; "REAL" "ARRAY" OBS;
"STRING" TEXT OBS, TEXT TH;
"BEGIN" "INTEGER" SUM, DIFF U, NOBS, POINTER, K;
    "REAL" LAST, NEXT, MIN XY, MAX XY;

    "IF" CHANNEL <= 0 "OR" CHANNEL = 60 "THEN"
        STATAL3 ERROR("("PROBPRINT")", 1, CHANNEL);
    "IF" L > U "THEN" STATAL3 ERROR ("("PROBPRINT")", 4, U);
```

```
"IF" NPOINTS <= 0 "THEN"
    STATAL3 ERROR ("("PROBPRINT")", 11, NPOINTS);
"IF" NUMBER < 0 "THEN"
    STATAL3 ERROR ("("PROBPRINT")", 14, NUMBER);
"BEGIN" "REAL" "ARRAY" A, F [L : U];

    "REAL" "PROCEDURE" ECDFINV (ALFA);
    "VALUE" ALFA; "REAL" ALFA;
    "BEGIN" POINTER:= POINTER - 1;
        "FOR" POINTER:= POINTER + 1
        "WHILE" F [POINTER] < ALFA "DO";
        ECDFINV:= A [POINTER]
    "END" ECDFINV;

    "IF" "NOT" SORTED "THEN"
    "BEGIN" VEC QSORT (OBS, L, U);
            SORTED:= "TRUE" "END";
    NOBS:= U - L + 1; SUM:= 0; A [L]:= LAST:= OBS [L];
    DIFF U:= L;
    "FOR" K:= L + 1 "STEP" 1 "UNTIL" U "DO"
    "BEGIN" NEXT:= OBS [K]; SUM:= SUM + 1;
        "IF" NEXT > LAST "THEN"
        "BEGIN" F [DIFF U]:= SUM / NOBS;
            DIFF U:= DIFF U + 1;
            A [DIFF U]:= LAST:= NEXT
        "END"
    "END";
    F [DIFF U]:= 1;
    POINTER:= L; DIFF U:= DIFF U - 1;
    "IF" MIN = MAX "THEN"
    "BEGIN" "INTEGER" K, J;
       "REAL" "ARRAY" H, V [1 : NPOINTS];
        K:= NPOINTS + 1;
        "FOR" J:= 1 "STEP" 1 "UNTIL" NPOINTS "DO"
        "BEGIN" X:= J / K; V [J]:= CDFINV X;
            H [J]:= ECDFINV (X)
        "END";
        MIN XY:= "IF" V [1] < H [1]
                 "THEN" V [1] "ELSE" H [1];
        MAX XY:= "IF" V [NPOINTS] > H [NPOINTS]
                 "THEN" V [NPOINTS] "ELSE" H [NPOINTS];
        SCATTERPRINT(H [J], MIN XY, MAX XY, TEXT OBS,
             V [J], MIN XY, MAX XY, TEXT TH, J, 1,
             NPOINTS, 0, 44, CHANNEL, NUMBER)
    "END" "ELSE"
    "IF" MIN < MAX "THEN"
    "BEGIN" "INTEGER" K; "REAL" STEP;
        "REAL" "ARRAY" O, P [1 : NPOINTS];
        STEP:= (MAX - MIN) / NPOINTS; MIN:= MIN - STEP;
        "FOR" K:= 1 "STEP" 1 "UNTIL" NPOINTS "DO"
        "BEGIN" X:= P [K]:= MIN + K * STEP;
            O [K]:= ECDFINV (CDF X)
        "END";
        MIN XY:= "IF" O [1] < P [1]
```

```
                     "THEN" O [1] "ELSE" P [1];
            MAX XY:= "IF" O [NPOINTS] > P [NPOINTS]
                     "THEN" O [NPOINTS] "ELSE" P [NPOINTS];
            SCATTERPRINT(O [K], MIN XY, MAX XY, TEXT OBS,
                 P [K], MIN XY, MAX XY, TEXT TH, K, 1,
                 NPOINTS, 0, 44, CHANNEL, NUMBER)
        "END" "ELSE"
        "BEGIN" "INTEGER" J; "REAL" "ARRAY" T [L : DIFF U];
            "FOR" J:= L "STEP" 1 "UNTIL" DIFF U "DO"
            "BEGIN" X:= F [J]; T [J]:= CDFINV X "END";
            MIN XY:= "IF" A [L] < T [L]
                     "THEN" A [L] "ELSE" T [L];
            MAX XY:= "IF" A [DIFF U] > T [DIFF U]
                     "THEN" A [DIFF U] "ELSE" T [DIFF U];
            SCATTERPRINT(A [J], MIN XY, MAX XY, TEXT OBS,
                 T [J], MIN XY, MAX XY, TEXT TH, J, L,
                 DIFF U, 0, 44, CHANNEL, NUMBER)
        "END"
    "END"
"END" PROBPRINT;
         "EOP"
```

TITLE: **Plotdist**

AUTHORS: A.J. van Es, C. van Putten, I. van der Tweel

INSTITUTE: Mathematical Centre

RECEIVED: 830301

BRIEF DESCRIPTION

A probability plot of requested type and size is plotted via a plotter. On request the plot contains a confidence band or an estimated straight line for reference. Enlargements of parts of the plot are possible.

KEYWORDS

Empirical distribution function, probability plot, confidence band

CALLING SEQUENCE

Heading

```
"PROCEDURE" PLOTDIST (GRFILE, V, LV, UV, TYPE, LB, UB, MODE, PART, SIZE,
OPTION, BETA, SORTED, PAR, IDENT);
"VALUE" V, LV, UV, LB, UB, MODE, PART, SIZE, OPTION, BETA, SORTED, PAR;
"INTEGER" LV, UV, MODE, PART, SIZE, OPTION;
"REAL" LB, UB, BETA, PAR;
"BOOLEAN" SORTED;
"ARRAY" V;
"STRING" GRFILE, TYPE, IDENT;
"CODE" 47005;
```

Formal parameters

GRFILE: <string>, name of the file on which the plot must be written as a maingraph. If the string GRFILE is empty then the name of this file is "GRFILE". In subsequent calls of PLOTDIST the same value may be chosen for GRFILE;

V: <array identifier>, V[LV],...,V[UV] is a vector containing the sample;

LV: <integer arithmetic expression>, smallest index of the sample array;

UV: <integer arithmetic expression>, largest index of the sample array;

TYPE: <string>, type of probability plot. TYPE should contain one of the following identifiers: UNIFORM, NORMAL, EXP1, EXP2, LAPLACE, GUMBEL, CAUCHY, WEIBULL2, WEIBULL3;

LB, UB: <real arithmetic expression>, lower and upper bound, respectively, for the possible enlargement. The plot contains the empirical distribution function and confidence band (when requested) in those arguments for which the empirical distribution function is greater then or equal to LB and less then or equal to UB. If no enlargement is desired LB should be taken equal to 0 and UB equal to 1;

MODE: <integer arithmetic expression>, if MODE=−1 the jumps of the empirical distribution function are connected by straight lines. If MODE>0 symbols with integer representation mode (see table 1) are plotted at the jumps;

PART: <integer arithmetic expression>, if PART=0 all jumps of the empirical distribution function are plotted. If PART>0 the empirical distribution function is plotted in PART+1 equidistant arguments;

SIZE: <integer arithmetic expression>, size of the plot. There are two possibilities for the parameter size. If SIZE equals 0,1,2,3,4 or 5 the format of the plot is "report format" (i.e. the plot fits on a page of a Mathematical Centre report), A1, A2, A3, A4 or A5, respectively. The other possibility is that SIZE equals a nonnegative integer of six digits, of which the first three indicate the width of the plot, while the last three indicate the height (in mm). The minimum size allowed is 100100. The height of the plot should be less than or equal to 725 mm;

OPTION: <integer arithmetic expression>, indicating whether a confidence band or a straight line for reference has to be plotted;
OPTION =11: Both the band and the line are plotted
OPTION =10: Only the band is plotted
OPTION =01: Only the line is plotted
OPTION =00: Neither one is plotted;

BETA: <real arithmetic expression>, confidence level of the confidence band. BETA should equal 0.9, 0.95 or 0.99;

SORTED: <boolean expression>, indicating whether the sample is sorted (in a non-decreasing or non-increasing order). In case of a sorted sample SORTED should be "TRUE", otherwise SORTED should be "FALSE";

PAR: <real arithmetic expression>, containing information about the theoretical distribution function in case TYPE equals WEIBULL2 or WEIBULL3;

IDENT: <string>, identifying text to appear below the plot. The maximum number of characters allowed depends on the size of the plot, as indicated by SIZE. If W and H denote the width and height of the plot (in mm), respectively, then it is 0.5*W if H≤210 and 0.5* W*210/H otherwise.

Table 1 - integer representation of some symbols (for the complete table see SARA publicatie 11, Graphics, CALCOMP, p.8)

2	Δ
3	+
4	×
11	*

DATA AND RESULTS

The empirical distribution function of the sample V[LV],...,V[UV] is computed and plotted in a probability plot, the type of which is indicated by the parameter TYPE. A detailed description of the method and performance of plotdist for the various types can be found in v. Es & v. Putten (1983). It suffices to give a description of the types EXP1, EXP2, WEIBULL2 and WEIBULL3 since all other values of TYPE refer to distributions of the given type with the usual parameters.

EXP1 and EXP2:

The plots of type EXP1 and EXP2 are both exponential probability plots which differ only in the way the straight line (if requested) is computed. In a plot of type EXP1 the threshold parameter is assumed zero and the line runs through the origin, while for plots of type EXP2 the treshold parameter is estimated.

WEIBULL2 and WEIBULL3:

The cumulative distribution function of the Weibull distribution is F(X)=1-EXP(-((X-LOC)/SCALE) **C).

WEIBULL2:

When LOC is known to be equal to 0,PAR should have the value 0. Then the empirical distribution function of the sample -LN(V[LV]),...,-LN(V[UV]) is plotted in a Gumbel probability plot.

When the value of C is known, PAR should be equal to C (and hence PAR>0). Then the emperical distribution function of V[LV],...,V[UV] is plotted in a Weibull (fixed C) probability plot.

WEIBULL3:

When the value of LOC is known, PAR should be made equal to LOC. Then the emperical distribution function of -LN(V[LV]-LOC),...,-LN(V[UV]-LOC) is plotted in a Gumbel probability plot.

When PAR equals 0, all three parameters are assumed unknown. In this case the location parameter LOC is estimated and then the procedure is the same as above.

Error messages are written on file OUTPUT via channel 61.

Procedures used

AXIS	CALCOMP
NUMBER	CALCOMP
PLOT	CALCOMP
PLOTS	CALCOMP
SCALE	CALCOMP
SYMBOL	CALCOMP
VECINDQSORT	STATAL 11021
LOGGAMMA	STATAL 40400
PHI	STATAL 41500
PHIINV	STATAL 41501
ZEROIN	NUMAL 34150

Language

Algol 60

References

[1] A.J. van Es & C. van Putten
Probability plots
Statal Report 3
Mathematical Centre, Amsterdam, 1983

[2] Graphics
SARA publikatie 11
Stichting Academisch Rekencentrum Amsterdam, 1980

Example of use

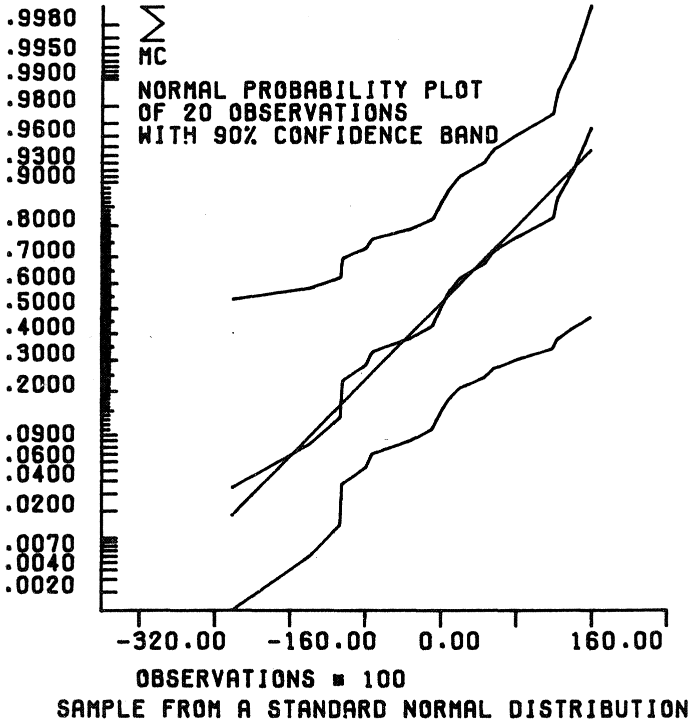
Program:

```
"BEGIN"
    "ARRAY" V[1:20];
    INARRAY(60, V);
    PLOTDIST("("")", V, 1, 20, "("NORMAL")", 0, 1, -1, 0,
        100100, 11, 0.90, "FALSE", 0,
        "("SAMPLE FROM A STANDARD NORMAL DISTRIBUTION")");
"END"
```

Input:

```
-0.80   1.58   0.02   0.83
-1.05   0.20  -1.07   0.09
 1.39   1.18  -0.73  -0.04
-0.10  -1.40  -2.22  -1.05
 0.56  -0.34   1.23   0.45
```

Output:


.9980
.9950
.9900
.9800
.9600
.9300
.9000
.8000
.7000
.6000
.5000
.4000
.3000
.2000
.0900
.0600
.0400
.0200
.0070
.0040
.0020
Σ
MC
NORMAL PROBABILITY PLOT
OF 20 OBSERVATIONS
WITH 90% CONFIDENCE BAND
-320.00
-160.00
0.00
160.00
OBSERVATIONS * 100
SAMPLE FROM A STANDARD NORMAL DISTRIBUTION

Source text

```
"CODE"47005;
"PROCEDURE" PLOTDIST(GRFILE,V,LV,UV,TYPE,LB,UB,MODE,PART,
                     SIZE,OPTION, BETA,SORTED,PAR,IDENT);
"VALUE" V,LV,UV,LB,UB,MODE,PART,SIZE,OPTION,BETA,SORTED,PAR;
"INTEGER" LV,UV,MODE,PART,SIZE,OPTION;
"REAL" LB,UB,BETA,PAR;
"BOOLEAN" SORTED;
"ARRAY" V;
"STRING" TYPE, IDENT, GRFILE;
"BEGIN"
  "INTEGER" I, NUV, ILV, INUV, SUB;
  "REAL" XMIN, XMAX, MIN, MAX, YMIN, YMAX, FACTOR, MEAN,
       STDDEV, DUNIT1, DUNIT2, YFACTOR, YFDUNIT1, YFDUNIT2,
       SWIDTH, CHARHEIGHT, XSCALE, YSCALE, XSHIFT, YSHIFT,
       XCONT, YCONT, YPOS, SIZEX, SIZEY, DELTA;
  "BOOLEAN" UNIFORM, NORMAL, EXP1, EXP2, GUMBEL, LAPLACE,
     CAUCHY, WEIBULL2, WEIBULL3, SPECIAL, CONFBAND, ESTLINE;
  "INTEGER" "ARRAY" IND [LV : UV];
  "ARRAY" F [LV-1 : UV], HL, HU [LV : UV], X[LV : UV];

  "INTEGER" "PROCEDURE" ENT(X); "VALUE" X; "REAL" X;
  ENT := "IF" X>=-1 "AND" X<0 "THEN" -1 "ELSE" ENTIER(X);

  "COMMENT" ALGOL5 RETURNS 0 FROM ENTIER(X) IF X LIES
            BETWEEN 0 AND -1/4 ;

  "REAL" "PROCEDURE" OWNDIST(X); "VALUE" X; "REAL" X;
  "CODE" 41598;
  "REAL" "PROCEDURE" OWNINV(PROB);
  "VALUE" PROB;
  "REAL" PROB;
  "CODE" 41599;

    "PROCEDURE" PLOT(X, Y); "VALUE" X, Y; "REAL" X, Y;
    PLOT CALCOMP(X CE(X), Y CE(Y), 2);

    "PROCEDURE" PLUP(X, Y); "VALUE" X, Y; "REAL" X, Y;
    PLOT CALCOMP(X CE(X), Y CE(Y), 3);

    "PROCEDURE" PLTEXT(X, Y, TEXT); "VALUE" X, Y;
    "REAL" X, Y; "STRING" TEXT;
    "BEGIN" "INTEGER" I, C, L;
        L:= LENGTH(TEXT);
        "IF" L > 0 "THEN"
        "BEGIN" "PROCEDURE" IEQU(I, C); "VALUE" I;

            "INTEGER" I, C;
            "BEGIN" STRINGELEMENT(TEXT, I,
            "(" !"#$ &'()*+,-./0123456789:;<=>?@")"
            "("ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_")", C);
                C:= C + 31;
            "END" IEQU;
```

```
          X:= "IF" X = XCONT "THEN" 999 "ELSE" XCE(X);
          Y:= "IF" Y = YCONT "THEN" 999 "ELSE" YCE(Y);
          IEQU(1, C); SYMBOL(X, Y, CHARHEIGHT, C, 0, -1);
          "FOR" I:= 2 "STEP" 1 "UNTIL" L "DO"
          "BEGIN" IEQU(I, C);
              SYMBOL(999, 999, CHARHEIGHT, C, 0, -1)
          "END"
      "END"
  "END" PLTEXT;

  "PROCEDURE" PLFRAME(X1, Y1, X2, Y2, XCM, YCM);
  "VALUE" X1, Y1, X2, Y2, XCM, YCM;

  "REAL" X1, Y1, X2, Y2, XCM, YCM;
  "BEGIN"
      XSCALE:= XCM / (X2 - X1); YSCALE:= YCM / (Y2 - Y1);
      XSHIFT:= -X1 * XSCALE; YSHIFT:= -Y1 * YSCALE;
      XCONT:= (999 - XSHIFT) / XSCALE;
      YCONT:= (999 - YSHIFT) / YSCALE;
  "END" PLFRAME;

  "REAL" "PROCEDURE" X CE(X); "VALUE" X; "REAL" X;
  X CE:= XSCALE * X + XSHIFT;

  "REAL" "PROCEDURE" Y CE(Y); "VALUE" Y; "REAL" Y;
  Y CE:= YSCALE * Y + YSHIFT;

"REAL" "PROCEDURE" DIST(X); "VALUE" X; "REAL" X;
"IF" UNIFORM               "THEN"
  DIST:= X "ELSE"
"IF" NORMAL                 "THEN"
  DIST:= PHI(X) "ELSE"
"IF" EXP1 "OR" EXP2         "THEN"
  DIST:= 1 - EXP(-X) "ELSE"

"IF" GUMBEL                 "THEN"
  DIST:= EXP(-EXP(-X)) "ELSE"
"IF" LAPLACE                "THEN"
  DIST:= "IF" X <= 0 "THEN" EXP(X) / 2 "ELSE"
         1 - EXP(-X) / 2 "ELSE"
"IF" CAUCHY                 "THEN"
  DIST:= 0.318309886 * ARCTAN(X) + 0.5 "ELSE"
"IF" WEIBULL2 "AND" PAR > 0 "THEN"
  DIST:= 1 - EXP(-X ** PAR) "ELSE"
"IF" SPECIAL "THEN" DIST:= OWNDIST(X);


"REAL" "PROCEDURE" DISTINV(PROB);
"VALUE" PROB; "REAL" PROB;
"IF" UNIFORM               "THEN"
  DISTINV:= PROB "ELSE"
"IF" NORMAL                 "THEN"
  DISTINV:= PHINV(PROB) "ELSE"
"IF" EXP1 "OR" EXP2         "THEN"
```

```
  DISTINV:= -LN(1 - PROB) "ELSE"
"IF" GUMBEL                 "THEN"

  DISTINV:= -LN( -LN(PROB)) "ELSE"
"IF" LAPLACE                "THEN"
  DISTINV:= "IF" 0 < PROB "AND" PROB <= 0.5
            "THEN" LN(PROB * 2)
            "ELSE" -LN(2 - PROB * 2) "ELSE"
"IF" CAUCHY                 "THEN"
  "BEGIN" PROB := (PROB - 0.5) * 3.14159654;
          DISTINV := TAN(PROB);
  "END" "ELSE"
"IF" WEIBULL2 "AND" PAR > 0 "THEN"
  DISTINV:= (-LN(1 - PROB)) ** (1 / PAR) "ELSE"
"IF" SPECIAL "THEN" DISTINV:= OWNINV(PROB);


"PROCEDURE" VERTAXIS(FROM,TO,OTHER);
"VALUE" FROM,TO,OTHER;
"REAL" FROM,TO,OTHER;
"BEGIN" "INTEGER" P;
  "REAL" LOW, HIGH, HEIGHT, LEVEL, K, LN10;

  "PROCEDURE" MARK(S); "VALUE" S; "REAL" S;
  "BEGIN"

    "IF" S > HIGH "THEN" "GOTO" ENDAXIS "ELSE"
    "IF" S >= LOW "THEN"
    "BEGIN" "REAL" INVS; "INTEGER" ZEROS, S10000;
      INVS:= DISTINV(S);
      "IF" INVS > LEVEL "THEN"
      "BEGIN"
          SYMBOL(XCE(OTHER - 1.3 / XSCALE * YFACTOR),
                 YCE(INVS), CHARHEIGHT, 46, 0, -1);
          S10000:= S * 10000;
          ZEROS:= 3 - ENT(LN(S10000) / LN10 + .000005);
          "FOR" ZEROS:= ZEROS "STEP" -1 "UNTIL" 1 "DO"
          SYMBOL(999, 999, CHARHEIGHT, 48, 0, -1);
          NUMBER(999, 999, CHARHEIGHT, S10000, 0, -1);
          LEVEL:= INVS + HEIGHT + .05 / YSCALE "END";
          PLUP(OTHER,INVS);
          PLOT(OTHER + .2/XSCALE*YFACTOR,INVS)
      "END"
  "END" MARK;

  HEIGHT:= CHARHEIGHT / YSCALE; LEVEL:= -YSHIFT / YSCALE;
  "IF" FROM < TO
  "THEN" "BEGIN" LOW:= FROM; HIGH:= TO "END"
  "ELSE" "BEGIN" LOW:= TO; HIGH:= FROM "END";

  LOW:= DIST(LOW); HIGH:= DIST(HIGH);
  PLUP(OTHER,TO); PLOT(OTHER,FROM);
  LN10:= LN(10);
  "FOR" K:= .0001, .001, .01 "DO"
```

```
    "FOR" P:= 1 "STEP" 1 "UNTIL" 9 "DO"
    "BEGIN" "REAL" S;
      S:= P * K; MARK(S)
    "END";
    "FOR" P:= 10 "STEP" 1 "UNTIL" 90 "DO"
    "IF" P / 10 = ENT(P / 10) "THEN" MARK(P / 100) "ELSE"
    "IF" P/100 > HIGH "THEN" "GOTO" ENDAXIS "ELSE"
    "IF" P/100 >= LOW "THEN"
    "BEGIN" "REAL" INVP; INVP:= DISTINV(P / 100);
      PLUP(OTHER,INVP);
      PLOT(OTHER + ("IF" P / 5 = ENT(P / 5) "THEN" .15
        "ELSE" .1) / XSCALE * YFACTOR,INVP)
    "END";

    "FOR" K:= .01, .001, .0001 "DO"
    "FOR" P:= 9 "STEP" -1 "UNTIL" 1 "DO"

    "BEGIN" "REAL" S;
      S:= 1 - P * K; MARK(S)
    "END";
ENDAXIS:
  "END" *** VERTAXIS ***;


  "PROCEDURE" PLOTLINE(X1,Y1,X2,Y2,XMIN,YMIN,XMAX,YMAX);
  "VALUE" X1,Y1,X2,Y2,XMIN,YMIN,XMAX,YMAX ;
  "REAL" X1,Y1,X2,Y2,XMIN,YMIN,XMAX,YMAX ;
  "BEGIN" "IF" Y1 = Y2 "THEN"
    "BEGIN" "IF" YMIN <= Y1 "AND" Y1 <= YMAX "THEN"
      "BEGIN" PLUP(XMIN,Y1); PLOT(XMAX,Y1) "END"
    "END" "ELSE"
    "IF" X1 = X2 "THEN"
    "BEGIN" "IF" XMIN <= X1 "AND" X1 <= XMAX "THEN"
      "BEGIN" PLUP(X1,YMIN); PLOT(X1,YMAX) "END"
    "END" "ELSE"
    "BEGIN" "REAL" R,A,B,C,D;


      "PROCEDURE" SORT4(A,B,C,D); "REAL" A,B,C,D;
      "BEGIN" "PROCEDURE" CHANGE(U,V); "REAL" U,V;
        "BEGIN" "REAL" W; W:= U; U:= V; V:= W "END";
        "IF" A > B "THEN" CHANGE(A,B);
        "IF" C > D "THEN" CHANGE(C,D);
        "IF" A > C "THEN" "BEGIN" CHANGE(A,C);
                                  CHANGE(B,D) "END";
        "IF" B > C "THEN" CHANGE(B,C);
        "IF" B > D "THEN" CHANGE(B,D);
      "END" SORT4;

      R:=(Y2 - Y1) / (X2 - X1); A:= X1 + (YMIN - Y1) / R;
      D:= X1 + (YMAX - Y1) / R;
      "IF" "NOT"((A > XMAX "AND" D > XMAX) "OR"
                 (A < XMIN "AND" D < XMIN)) "THEN"
      "BEGIN" B:= XMIN; C:= XMAX; SORT4(A,B,C,D);
      PLUP(B,Y1 + R * (B - X1));
      PLOT(C,Y1 + R * (C - X1))
    "END"
  "END"

"END" *** PLOTLINE ***;


"PROCEDURE" MOMENTS(I,VI,LV,UV,MEAN,STDDEV);
"VALUE" LV,UV; "INTEGER" I,LV,UV; "REAL"VI,MEAN,STDDEV;
"BEGIN" "INTEGER" NV;
  NV:= UV - LV + 1;
  MEAN:= STDDEV:= 0;
  "IF" NV >= 1 "THEN"
  "BEGIN" "FOR" I:= LV "STEP" 1 "UNTIL" UV "DO"
    "BEGIN" "REAL" VIL; VIL:= VI; MEAN:= MEAN + VIL;
      STDDEV:= STDDEV + VIL * VIL
    "END";
    MEAN:= MEAN / NV;
    STDDEV:= "IF" NV < 2 "THEN" -1 "ELSE"
      SQRT((STDDEV - NV * MEAN * MEAN) / (NV - 1))
  "END"
"END" *** MOMENTS ***;


"PROCEDURE" TRANSFORM DATA(V,LV,UV,PAR);
"VALUE" LV,UV; "INTEGER" LV,UV; "REAL" PAR; "ARRAY" V;
"BEGIN" "INTEGER" I;

  "IF" WEIBULL2 "AND" PAR < 0 "THEN"
  "BEGIN"
    OUTPUT(61,"("//,
 "("PARAMETER OF THE WEIBULL-DISTR.FUNCTION < 0 ")"")");
    "GOTO" EOP
  "END" "ELSE"
  "IF" WEIBULL3 "AND" PAR = 0 "THEN"
    PAR:= WEIBULL3 PARAMETER(V,LV,UV);
  "IF" WEIBULL2 "AND" PAR = 0 "OR" WEIBULL3 "THEN"
  "BEGIN" "REAL" A;
    "FOR" I:= LV "STEP" 1 "UNTIL" UV "DO"
    "BEGIN"
      V[I]:= -LN(V[I] - PAR);
      "IF" (I<=((UV+LV)/2)) "THEN"
      "BEGIN" A := IND[I];
         IND[I] := IND[UV+LV-I];
         IND[UV+LV-I] := A

      "END";
    "END";
    MOMENTS(I,V[I],LV,UV,MEAN,STDDEV);
    GUMBEL:= "TRUE"
  "END"
"END" *** TRANSFORM DATA ***;
```

```
"REAL" "PROCEDURE" WEIBULL3 PARAMETER(V,LV,UV);
"VALUE" LV,UV; "INTEGER" LV,UV; "ARRAY" V;
"BEGIN"
  "REAL" THETA, THETA1, MIN, Q, Q1, QUANTILE, MEDIAN;
  "INTEGER" N, NG2, NO, I;
  "BOOLEAN" NULNEG;
  "REAL""PROCEDURE" F(THETA);
  "VALUE" THETA; "REAL" THETA;
  "BEGIN" "REAL" L; L := LOGGAMMA(1+THETA);
    "IF" L>700 "OR" THETA>700/LN(N) "THEN"
    OUTPUT(61,"("/,
    "("THE ESTIMATED VALUE OF THETA IS TOO LARGE")" ,/,
    "("LOGGAMMA IS: ")",+4ZD.2D,/,"("THETA IS:")",
    +4ZD.2D")", L,THETA)
    "ELSE" F := (MIN - QUANTILE + (QUANTILE - MEAN) /
    N ** THETA) * EXP(L) + (MEAN - MIN) * Q ** THETA;
  "END" PROCEDURE F;
  "PROCEDURE" ESTIMATE (XL,TOLX,XL1,NULNEG);
  "REAL" XL, TOLX, XL1;
  "BOOLEAN" NULNEG;
  "BEGIN" "REAL" XR, LBOUND, UBOUND; "INTEGER" NSTEP;
    XL:= XL1:= LBOUND:= UBOUND:= .001; NSTEP := 0;
    NULNEG:= F(XL) < 0;
    "IF" NULNEG "THEN"
    "BEGIN"
  L1: NSTEP:= NSTEP + 1;
      "IF" NSTEP > 18 "THEN"
      "BEGIN" OUTPUT(61,"("/,
   "("THETA ESTIMATE NOT FOUND IN [.001,262.144]")"")");
        "GOTO" EOP
      "END" "ELSE"
      "BEGIN" XL:= XL*2;
        "IF" F(XL) < 0
        "THEN" "BEGIN" XL1:= XL; "GOTO" L1 "END";
        LBOUND:= XL1; UBOUND:= XL;
        "IF" "NOT" ZEROIN(XL1, XL, F(XL1), TOLX) "THEN"
        "BEGIN"
          NULNEG:= "FALSE"; OUTPUT( 61, "("/,
          "("ZEROIN DID NOT FIND THETA ESTIMATE IN [ ")",
          3ZD.3D, "(", ")", 3ZD.3D, "("]")"")", LBOUND,
          UBOUND)
        "END"
      "END"
    "END";
    XL:= XR:= UBOUND;
 L: NSTEP:= NSTEP + 1;
    "IF" NSTEP > 18 "THEN"
    "BEGIN" OUTPUT( 61, "("/,
      "("THETA ESTIMATE NOT FOUND IN [ ")", 3ZD.3D,
      "(", 262.144]")" ")", UBOUND);
      "IF" NULNEG
      "THEN" "BEGIN" XL:= XL1; NULNEG:= "FALSE" "END"
      "ELSE" "GOTO" EOP
    "END" "ELSE"
    "BEGIN" XR:= XR*2;
      "IF" F(XR) > 0
      "THEN" "BEGIN" XL:= XR; "GOTO" L "END";
      LBOUND:= XL; UBOUND:= XR;
      "IF" "NOT" ZEROIN(XL, XR, F(XL), TOLX) "THEN"
      "BEGIN" OUTPUT(61, "("/,
        "("ZEROIN DID NOT FIND THETA ESTIMATE IN [ ")",
        3ZD.3D, "(", ")", 3ZD.3D, "("]")"")",
        LBOUND, UBOUND );
        "IF" NULNEG
        "THEN" "BEGIN" XL:= XL1; NULNEG:= "FALSE" "END"
        "ELSE" "GOTO" EOP
      "END"
    "END"
  "END" ESTIMATE;

  MIN := V[IND[LV]]; N := UV-LV+1; NO := ENT(.4*N);
  QUANTILE := V[IND[NO]];
  "IF" ABS(QUANTILE-MEAN)/(MEAN-MIN) < .1 "THEN"
  "BEGIN" NO := ENT(.9*N); QUANTILE := V[IND[NO]] "END";
  OUTPUT(61,"("/,"("INFORMATION WEIBULL3 PARAMETER ...")"
  ,4/, "("I IS           :")",3ZD,/,
  "("RESULTING NO IS:")",3ZD,/")",I,NO);
  Q := NO/(N+1); Q := -LN(1-Q);
  "IF" NO=1 "THEN"
    "BEGIN" OUTPUT(61,"(""("NO = 1")",/")");
           "GOTO" EOP "END";
  ESTIMATE (THETA, THETA*"-5, THETA1, NULNEG);
  Q := MIN - (MEAN-MIN)/(N**THETA-1);
  "IF" NULNEG "THEN"
  "BEGIN" Q1:= MIN - (MEAN-MIN)/(N**THETA1-1);
    NG2:= N//2;
    MEDIAN:= "IF" NG2*2 = N
             "THEN" (V[IND[NG2]]+V[IND[NG2+1]])/2
             "ELSE"V[IND[NG2+1]];
    "IF" ABS( Q1 + (MEAN-Q1)*(LN(2))**THETA1/
         EXP(LOGGAMMA(1+THETA1)) - MEDIAN) <
         ABS( Q + (MEAN-Q)*(LN(2))**THETA/
         EXP(LOGGAMMA(1+THETA)) - MEDIAN) "THEN"
    "BEGIN" Q:= Q1; THETA:= THETA1 "END"
  "END";
  WEIBULL3 PARAMETER := "IF" Q < MIN
                        "THEN" Q "ELSE" MIN - 1/N;
  "IF" Q >= MIN "THEN"
  OUTPUT(61,"("4/,
    "("SINCE THE ESTIMATE OF THE LOCATION PARAMETER")"
    ,/,"("IS LARGER THAN THE MINIMUM OF THE SAMPLE IT")"
    ,/,"("CANNOT BE USED. THE LOCATION PARAMETER USED ")"
    ,/,"("INSTEAD IS THE SAMPLE MINIMUM MINUS 1/N.")"")");
"END" ***  WEIBULL3 PARAMETER *** ;

"PROCEDURE" COORDINATES(X,F,HL,HU,NUV);

"INTEGER" NUV; "ARRAY" X,F,HL,HU;
```

```
"BEGIN" "INTEGER" I,PREVI,M;
  "REAL" K,P;

  "REAL" "PROCEDURE" KRIT(N);
  "VALUE" N; "INTEGER" N;
  "IF" BETA = 0.90 "THEN" KRIT:= 4.5 / SQRT(N) "ELSE"
  "IF" BETA = 0.95 "THEN" KRIT:= 6.2 / SQRT(N) "ELSE"
  "IF" BETA = 0.99 "THEN" KRIT:=  14 / SQRT(N) "ELSE"
  "BEGIN" KRIT:= 0;
    OUTPUT(61,"("//,"("BETA IS NOT EQUAL TO 0.90, 0.95
         OR 0.99 ")"")");
    CONFBAND:= "FALSE"
  "END" KRIT;

  "REAL" "PROCEDURE" H(S,U,K);
  "VALUE" S,U,K; "INTEGER" S; "REAL" U,K;
  H:= (U + K * K / 2 + S * K * SQRT(U * (1 - U) + K *
       K / 4)) / (1 + K * K);

  "IF" V[LV] <= V[UV]
  "THEN""BEGIN" "FOR" I:=LV "STEP" 1
                "UNTIL" UV "DO" IND[I]:=I "END"
  "ELSE" "FOR" I:=LV "STEP" 1
         "UNTIL" UV "DO" IND[I]:=LV+UV-I;
  "IF" "NOT" SORTED "THEN" VECINDQSORT(V,IND,LV,UV);
  "IF" WEIBULL2 "OR" WEIBULL3
  "THEN" TRANSFORM DATA(V,LV,UV,PAR);
  X[LV] := V[IND[LV]]; F[LV-1] := 0;
  M := LV; PREVI := LV;
  "FOR" I:=LV "STEP" 1 "UNTIL" UV "DO"
  "BEGIN" P := V[IND[I]];
    "IF" X[M] < P "THEN"
    "BEGIN"
       F[M] := F[M-1] + I - PREVI;
       X[M+1] := P;
       M := M+1; PREVI := I ;
    "END"
  "END";
  NUV := M; M :=  UV - LV + 1;
  F[NUV] := M; K := KRIT(M);
  "FOR" I:=LV "STEP" 1 "UNTIL" NUV "DO"
  "BEGIN"
     P := (F[I]-0.3)/(M+0.4); F[I] := DISTINV(P);
    "IF" CONFBAND
    "THEN" "BEGIN" HL[I]:= H(-1,P,K);
           HU[I]:= H(+1,P,K) "END"
    "ELSE" "BEGIN" HL[I]:= 0 ; HU[I]:= 1 "END"
  "END"
"END" *** COORDINATES ***;


"PROCEDURE" BOUNDS(NUV,ILV,INUV,F);
"VALUE" NUV; "INTEGER" NUV,ILV,INUV; "ARRAY" F;
"BEGIN" "INTEGER" I;
```

```
    ILV:= "7; INUV:= -"7;
    "IF" LB = 0  "THEN" ILV:= LV "ELSE"
    "BEGIN" LB:= DISTINV(LB);
      "FOR" I:= LV "STEP" 1 "UNTIL" NUV "DO"
      "IF" F[I] >= LB
      "THEN" "BEGIN" ILV:= I; "GOTO" L1 "END";
    "END";
L1: "IF" UB = 1 "THEN" INUV:= NUV "ELSE"
    "BEGIN" UB:= DISTINV(UB);
      "FOR" I:= NUV "STEP" -1 "UNTIL" LV "DO"

      "IF" F[I] <= UB
      "THEN" "BEGIN" INUV:= I; "GOTO" L2 "END"
    "END";
L2: "IF" INUV < ILV "THEN"
    "BEGIN"
        OUTPUT(61, "("/,"("NO OBSERVATIONS LEFT")",/")");
        "GOTO" EOP
    "END";
  "END" *** BOUNDS ***;


  "PROCEDURE" YMINMAX(F,HL,HU,L,U,YMIN,YMAX);
  "VALUE" L,U;
  "INTEGER" L,U;
  "REAL" YMIN,YMAX;
  "ARRAY" F,HL,HU;
  "BEGIN" "REAL" FF,HH;
    FF:= F[L]; HH:= HL[L];
    "IF" HH <= 0 "THEN" YMIN:= FF "ELSE"
    "BEGIN" HH:= DISTINV(HH);
       YMIN:= "IF" FF < HH "THEN" FF "ELSE" HH
    "END";
    FF:= F[U]; HH:= HU[U];
    "IF" HH >= 1 "THEN" YMAX:= FF "ELSE"
    "BEGIN" HH:= DISTINV(HH);
            YMAX:= "IF" FF > HH "THEN" FF "ELSE" HH
    "END"

  "END" *** YMINMAX ***;


  "PROCEDURE" INITPLOT(X,L,U,XMIN,XMAX,YMIN,YMAX,MIN,MAX,
                                              FACTOR,SUB);
  "VALUE" L,U,YMIN,YMAX; "INTEGER" L,U,SUB; "ARRAY" X;
  "REAL" XMIN,XMAX,YMIN,YMAX,MIN,MAX,FACTOR;
  "BEGIN" "INTEGER" I,E,NINT,CHL;
    "REAL" DUM1,DUM2,HULP; "ARRAY" DUMRIJ[1:4];

    XMIN:= X[L]; XMAX:= X[U];
    HULP := -ENT(LN(XMAX-XMIN)/LN(10)) + 2;
    FACTOR := 10.0 ** HULP; E := 10 ** ABS(HULP);
    SUB := -ENT( ABS(XMIN)*FACTOR/1000 ) * 1000 *
                                          SIGN(XMIN);
```

```
SIZEX := "IF" SIZE=0 "THEN" 21.6 "ELSE" "IF" SIZE<=5
         "THEN" 59.4 / 1.414213562 ** (SIZE-2)
         "ELSE" ENT(SIZE/1000) / 10;
SIZEY := "IF" SIZE=0 "THEN" 15.8 "ELSE" "IF" SIZE<=5
         "THEN"   42 / 1.414213562 ** (SIZE-2)
         "ELSE" SIZE / 10 - SIZEX * 1000;

"IF" SIZEY>72.5
"THEN"
    "BEGIN"
       OUTPUT(61,"("/,
        "("THE PLOTSIZE SUGGESTED BY THE PARAMETER")",
        "(" SIZE IS TOO LARGE FOR THE PLOTTER.")",/")");
       "GOTO" EOP
     "END"
"ELSE"
"BEGIN" I:= EQUIV(GRFILE);
    "IF" I = 0 "THEN" I:= EQUIV("("GRFILE")");
    PLOTS(0, 0, I * 4096);
"END";
YFACTOR:= "IF" SIZEY <= 21 "THEN" 1 "ELSE" SIZEY/21;
DUMRIJ[1] := XMIN*FACTOR+SUB;
DUMRIJ[2] := XMAX*FACTOR+SUB;
SCALE(DUMRIJ,SIZEX - 3*YFACTOR,2,1);
MIN := DUMRIJ[3]; DELTA := DUMRIJ[4];
"IF" DELTA >= 100
   "THEN" "BEGIN" FACTOR:=FACTOR/10;
                  SUB:=SUB/10;
                  "IF" HULP > 0
                  "THEN" E:=E/10
                  "ELSE" E:=E*10
          "END"
   "ELSE" "IF" DELTA <= .01
          "THEN" "BEGIN" FACTOR:=FACTOR*10;
                         SUB:=SUB*10;
                         "IF" HULP >= 0
                         "THEN" E:=E*10
                         "ELSE" E:=E/10
                 "END";
"IF" E ^= 1 "OR" SUB ^= 0 "THEN"
"FOR" I:= L "STEP" 1 "UNTIL" U
"DO" X[I]:= X[I] * FACTOR + SUB;
XMIN:= X[L]; XMAX:= X[U];
"IF" DELTA >= 100 "OR" DELTA <= .01 "THEN"
     "BEGIN" DUMRIJ[1]:=XMIN;
             DUMRIJ[2]:=XMAX;
             SCALE(DUMRIJ,SIZEX-3*YFACTOR,2,1);
             MIN:=DUMRIJ[3]; DELTA:=DUMRIJ[4]
     "END";
PLOT CALCOMP(2*YFACTOR, 1.5*YFACTOR, -3);
NINT:= -1; MAX:= MIN;
"FOR" NINT:= NINT + 1 "WHILE" MAX < XMAX
"DO" MAX:= MAX + DELTA;
DUNIT1:= (MAX - MIN)/(SIZEX - 3*YFACTOR);
```

```
DUNIT2:= (YMAX - YMIN)/(SIZEY - 2*YFACTOR);
YFDUNIT1:= YFACTOR*DUNIT1; YFDUNIT2:= YFACTOR*DUNIT2;
PLFRAME(MIN - 2*YFDUNIT1,YMIN - 1.5*YFDUNIT2,
  MAX + YFDUNIT1, YMAX + .5*YFDUNIT2, SIZEX, SIZEY);
AXIS(0, 0, 0, -1, SIZEX - 3*YFACTOR, 0,
     MIN, DELTA);
PLOT CALCOMP(-2*YFACTOR, -1.5*YFACTOR, -3);
CHARHEIGHT:= 0.2 * YFACTOR;
YPOS:= YMIN;
CHL:= CHLENGTH(IDENT);
"IF" CHL > 0 "THEN"
"BEGIN" "IF" CHL < SIZEX*5/YFACTOR "THEN"
    PLTEXT((MIN + MAX)/2 - (.1*CHL + .35)*YFDUNIT1,
            YMIN - 1.4*YFDUNIT2, IDENT)
    "ELSE"
    OUTPUT(61, "("/,
      "("PLOT IDENTIFICATION IS TOO LONG, LENGTH =")",
          3ZD.DB, "("CM")"")", CHL*YFACTOR/5 )
"END";
DUM1:= YPOS - YFDUNIT2;
"IF" WEIBULL2 "AND" PAR = 0 "THEN"
  PLTEXT(MIN, DUM1, "("-LN(OBSERVATIONS) ")") "ELSE"
"IF" WEIBULL3 "THEN"
"BEGIN" PLTEXT(MIN, DUM1, "("-LN(OBSERVATIONS ")");
  SYMBOL(999, 999, CHARHEIGHT,
      "IF" PAR >= 0 "THEN" 45 "ELSE" 43, 0, -1);
  NUMBER(999, 999, CHARHEIGHT, ABS(PAR), 0, 3);
  "FOR" I:= 32, 41
  "DO" SYMBOL(999, 999, CHARHEIGHT, I, 0, -1);
"END" "ELSE"
  PLTEXT(MIN, DUM1, "("OBSERVATIONS")");
"IF" E^=1 "THEN"
   "BEGIN" "FOR" I:= 32,
                   "IF" FACTOR > 1 "THEN" 42 "ELSE" 47,
                   "IF" E >= 0 "THEN" 32 "ELSE" 45
           "DO" SYMBOL(999, 999, CHARHEIGHT, I, 0, -1);
           NUMBER(999, 999, CHARHEIGHT, ABS(E), 0, -1);
   "END";
"IF" SUB ^= 0 "THEN"
      "BEGIN" "FOR" I:=32,32,
                       "IF" SUB >0 "THEN" 43 "ELSE" 45
              "DO" SYMBOL(999,999,CHARHEIGHT,I,0,-1);
           NUMBER(999, 999, CHARHEIGHT,ABS(SUB), 0, -1);
      "END";
PLUP(MIN, YPOS); PLOT(MIN - YFDUNIT1/2, YPOS);
VERTAXIS(YMIN, YMAX, MIN - YFDUNIT1/2);
SWIDTH:= .3*YFDUNIT1; PLUP(MIN + SWIDTH, YMAX);
PLOT(MIN, YMAX);
PLOT(MIN + SWIDTH, YMAX - .225*YFDUNIT2);
PLOT(MIN, YMAX - .45*YFDUNIT2);
PLOT(MIN + SWIDTH, YMAX - .45*YFDUNIT2);
PLTEXT(MIN, YMAX - .75*YFDUNIT2, "("MC")");
DUM1:= YMAX - 1.2*YFDUNIT2;
"IF" EXP1 "OR" EXP2
```

```
  "THEN" PLTEXT(MIN, DUM1, "("EXPONENTIAL")")
  "ELSE"
  "IF" WEIBULL2 "OR" WEIBULL3
  "THEN" PLTEXT(MIN, DUM1, "("WEIBULL")")
  "ELSE" PLTEXT(MIN, DUM1, TYPE);
  PLTEXT(XCONT, YCONT, "(" PROBABILITY PLOT")");
  PLTEXT(MIN,YMAX - 1.5*YFDUNIT2, "("OF ")");
  NUMBER(999, 999, CHARHEIGHT, UV-LV+1, 0, -1);
  PLTEXT(XCONT, YCONT, "(" OBSERVATIONS")");
  "IF" CONFBAND "THEN"
  "BEGIN" PLTEXT(MIN,YMAX - 1.8*YFDUNIT2, "("WITH ")");
     NUMBER(999, 999, CHARHEIGHT, BETA * 100, 0, -1);
     SYMBOL(999, 999, CHARHEIGHT, 37, 0, -1);
     PLTEXT(XCONT, YCONT, "(" CONFIDENCE BAND")")
  "END";
  CHARHEIGHT:= SIZEY / 70;
"END" *** INITPLOT ***;


"PROCEDURE" PLOTPICTURE(X,F,HL,HU,L,U,XMIN,XMAX);
"VALUE" L,U,XMIN,XMAX; "INTEGER" L,U; "REAL" XMIN,XMAX;
"ARRAY" X,F,HL,HU;
"BEGIN" "INTEGER" I; "REAL" H;

  PLOT CALCOMP(2*YFACTOR,0,-3);
  XSCALE:= 1/DELTA;XSHIFT:= -MIN/DELTA;
  "IF" MODE = -1 "THEN"
  "BEGIN" "IF" PART = 0 "THEN"
    "BEGIN" PLUP(X[U],F[U]) ;
      "FOR" I:= U-1 "STEP" -1 "UNTIL" L
      "DO" PLOT(X[I],F[I])
    "END" "ELSE"
    "IF" PART > 0 "THEN"
    "BEGIN" "REAL" NEWX,STEPX;
      NEWX:= XMAX; STEPX:= (XMAX - XMIN) / PART;
      PLUP(NEWX,F[U]) ;
      "FOR" I:= U "STEP" -1 "UNTIL" L "DO"
      "BEGIN" "IF" X[I] <= NEWX "THEN"
        "BEGIN" PLOT(NEWX,F[I]); NEWX:= NEWX - STEPX;
          I:= I + 1
        "END"
      "END"
    "END"
  "END" "ELSE"
  "IF" PART = 0 "THEN"
  "BEGIN" "FOR" I:= U "STEP" -1 "UNTIL" L "DO"
    SYMBOL(XCE(X[I]), YCE(F[I]), CHARHEIGHT, MODE, 0, -1);
  "END" "ELSE"
  "IF" PART > 0 "THEN"
  "BEGIN" "REAL" NEWX,STEPX;
    NEWX:= XMAX; STEPX:= (XMAX - XMIN) / PART;
    "FOR" I:= U "STEP" -1 "UNTIL" L "DO"
    "BEGIN" "IF" X[I] <= NEWX "THEN"
      "BEGIN" SYMBOL(XCE(NEWX), YCE(F[I]),
                         CHARHEIGHT, MODE, 0, -1);
        NEWX:= NEWX - STEPX; I:= I + 1
      "END"
    "END"
  "END";
  "IF" CONFBAND "THEN"
  "BEGIN" PLUP(X[L],DISTINV(HL[L]));
    "FOR" I:= L + 1 "STEP" 1 "UNTIL" U "DO"
    PLOT(X[I],DISTINV(HL[I]));
    PLUP(X[U],DISTINV(HU[U]));
    "FOR" I:= U - 1 "STEP" -1 "UNTIL" L "DO"
    PLOT(X[I],DISTINV(HU[I]))
  "END"
"END" *** PLOTPICTURE ***;



"PROCEDURE" LEAST SQUARES LINE (N);
"VALUE" N;
"INTEGER" N;
"BEGIN" "REAL""ARRAY" XX,YY[1:9]; "INTEGER" I,J;
  "REAL" XMEAN, YMEAN, A, B, C;

  XMEAN := YMEAN := A := B := 0;
  "FOR" I:=1 "STEP" 1 "UNTIL" 9 "DO"
  "BEGIN"
    J := LV + ENT(I*N/10) + 2;
    "IF" J>NUV+1 "THEN" J := NUV + 1;
    "FOR" J:=J-1 "WHILE" F[J]>=DISTINV(I/10) "DO";
    J:= J+1; YY[I] := F[J];
    "IF" ((LV<J "AND" J<ILV) "OR" (NUV >J "AND" J>INUV))
    "THEN" XX[I]:= X[J]*FACTOR + SUB "ELSE" XX[I]:= X[J];
    XMEAN := XMEAN + XX[I];
    YMEAN := YMEAN + YY[I]
  "END";
  XMEAN := XMEAN/9; YMEAN := YMEAN/9;
  "FOR" I:=1 "STEP" 1 "UNTIL" 9 "DO"

  "BEGIN"
    C := XX[I] - XMEAN;
    A := A + YY[I] * C;
    B := B + C * C
  "END";
  B := A/B; A := YMEAN - B * XMEAN;
  "COMMENT" THE LEAST SQUARES LINE IS GIVEN BY  A + BX ;
  PLOTLINE (0,A,1,A+B,MIN,YMIN,MAX,YMAX)
"END" *** LEAST SQUARES LINE ***;



"BEGIN" "INTEGER" I; I := EQUIV(TYPE);
   UNIFORM  := I = EQUIV("("UNIFORM")");
   NORMAL   := I = EQUIV("("NORMAL")");
   EXP1     := I = EQUIV("("EXP1")");
   EXP2     := I = EQUIV("("EXP2")");
   GUMBEL   := I = EQUIV("("GUMBEL")");
   LAPLACE  := I = EQUIV("("LAPLACE")");
   CAUCHY   := I = EQUIV("("CAUCHY")");

   WEIBULL2 := I = EQUIV("("WEIBULL2")");
   WEIBULL3 := I = EQUIV("("WEIBULL3")");
   SPECIAL  := I = EQUIV("("SPECIAL")");
   "IF" "NOT" (UNIFORM "OR" NORMAL "OR" EXP1 "OR" EXP2
               "OR" GUMBEL "OR" LAPLACE "OR" CAUCHY "OR"
               WEIBULL2 "OR" WEIBULL3 "OR" SPECIAL)
   "THEN"
     "BEGIN"
        OUTPUT(61,"("/,"("DISTRIBUTIONTYPE "")",N,
                     "("" NOT ALLOWED !")",/")", TYPE);
        "GOTO" EOP
     "END";
"END";

CONFBAND:= OPTION // 10 = 1;
ESTLINE:= OPTION - OPTION // 10 * 10 = 1;
MOMENTS(I,V[I],LV,UV,MEAN,STDDEV);
COORDINATES(X,F,HL,HU,NUV);
BOUNDS(NUV,ILV,INUV,F);
YMINMAX(F,HL,HU,ILV,INUV,YMIN,YMAX);
INITPLOT(X,ILV,INUV,XMIN,XMAX,YMIN,YMAX,MIN,MAX,
                                          FACTOR,SUB);
PLOTPICTURE(X,F,HL,HU,ILV,INUV,XMIN,XMAX);


"IF" ESTLINE "THEN"
"BEGIN" MEAN:= MEAN * FACTOR + SUB;
  STDDEV:= STDDEV * FACTOR;
  XMIN := X[LV]; XMAX := X[NUV];
  "IF" ILV>LV "THEN" XMIN := XMIN*FACTOR + SUB;
  "IF" INUV<NUV "THEN" XMAX := XMAX*FACTOR + SUB;
  MIN := X[ILV]; MAX :=X[INUV];
  "IF" UNIFORM "THEN"
  "BEGIN" "INTEGER" N; "REAL" A,H;
    N:= UV - LV + 1;
    A:= (XMIN + XMAX) / 2;
    H:= (XMAX - XMIN) / 2 * (N + 1) / (N - 1);
    PLOTLINE(A - H,0,A + H,1,MIN,YMIN,MAX,YMAX)
  "END" "ELSE"
  "IF" NORMAL  "THEN"
    PLOTLINE(MEAN,0,MEAN + STDDEV,1,MIN,YMIN,MAX,YMAX)
  "ELSE"
  "IF" EXP1 "THEN"

    PLOTLINE(0,0,MEAN,1,MIN,YMIN,MAX,YMAX) "ELSE"
  "IF" EXP2 "THEN"
  "BEGIN" "INTEGER" N;
    N:= UV - LV + 1;
    PLOTLINE((N * XMIN - MEAN) / (N - 1),0,MEAN,1,MIN,
                                             YMIN,MAX,YMAX)
    "END" "ELSE"
    "IF" GUMBEL "THEN"
    "BEGIN" "REAL" PI,MEANG,STDDEVG;
      PI:= 3.141592653589;
      STDDEVG:= STDDEV * SQRT(6) / PI;
      MEANG:= MEAN - 0.5772156649 * STDDEVG;
      PLOTLINE(MEANG,0,MEANG + STDDEVG,1,MIN,YMIN,MAX,YMAX)
    "END" "ELSE"
    "IF" LAPLACE "THEN"
    PLOTLINE(MEAN,0,MEAN + STDDEV / SQRT(2),1,MIN,
                                           YMIN,MAX,YMAX)
    "ELSE"
    "IF" WEIBULL2 "AND" PAR > 0 "OR" CAUCHY "OR" SPECIAL
    "THEN" LEAST SQUARES LINE (NUV)
  "END";

EOP:

"END" ****** PLDIST ******;
         "EOP"
```

7. AUXILARY PROCEDURES

This section contains several procedures which are frequently used in STATAL-procedures. Although these procedures are not intended to be used alone, nevertheless there are some examples of use included.

TITLE: **Loggamma**

AUTHOR: C. van Putten

INSTITUTE: Mathematical Centre

RECEIVED: 760501

BRIEF DESCRIPTION

The procedure computes the logarithm of the gamma function.

KEYWORDS
Logarithm of the gamma function

CALLING SEQUENCE

Heading

```
"REAL" "PROCEDURE" LOGGAMMA (X);
"VALUE" X;
"REAL" X;
"CODE" 40400;
```

Formal parameters

X: <arithmetic expression>, argument of the function.

DATA AND RESULTS

The value of the function is assigned to the procedure identifier LOGGAMMA.
The following error message may appear:
Errornumber 1 (if X $\leqslant$ 0)

PROCEDURES USED

STATAL3 ERROR	STATAL 40100

LANGUAGE

Compass

METHOD AND PERFORMANCE

The algorithm is basically that of CACM ALG. 309.
For arguments greater than 7 a slight modification of CACM ALG. 309 is used. For smaller arguments the recurrent relation LOGGAMMA (T+1)=LOGGAMMA (T)+LN(T) is applied until an argument greater than 7 is reached.
The precision is 10^{-10}.

REFERENCE

[1] Collected algorithms from CACM,
Association for computing machinery,
New York, 1975

Example of use

Program:

```
"BEGIN"
  OUTPUT(61, "("3(3Z.6D,/)")",
      LOGGAMMA( 4),
      LOGGAMMA(10),
      LOGGAMMA(47))
"END"
```

Output:

```
  1.791759
 12.801827
132.952575
```

Source text

The procedure is written in COMPASS; an equivalent ALGOL 60 text is given.

```
"CODE" 40400;
"REAL" "PROCEDURE" LOGGAMMA(T); "VALUE" T; "REAL" T;
"BEGIN" "COMMENT" ADOPTED FROM CACM ALGORITHM 309;
    "REAL" "PROCEDURE" LGM(W); "VALUE" W; "REAL" W;
    "BEGIN" "ARRAY" C[1:20];
        "REAL" W2, PRESUM, CONST, DEN, SUM;
        "INTEGER" I;

        C[ 1]:= +8.3333333333333"-002;
        C[ 2]:= -2.7777777777778"-003;
        C[ 3]:= +7.9365079365080"-004;
        C[ 4]:= -5.9523809523810"-004;
        C[ 5]:= +8.4175084175084"-004;
        C[ 6]:= -1.9175269175269"-003;
        C[ 7]:= +6.4102564102564"-003;
        C[ 8]:= -2.9550653594771"-002;
        C[ 9]:= +1.7964437236883"-001;
        C[10]:= -1.3924322169059"+000;
        C[11]:= +1.3402864044168"+001;
        C[12]:= -1.5684828462600"+002;
        C[13]:= +2.1931033333333"+003;
        C[14]:= -3.6108771253725"+004;
        C[15]:= +6.9147226885131"+005;
        C[16]:= -1.5238221539407"+007;
        C[17]:= +3.8290075139142"+008;
        C[18]:= -1.0882266035784"+010;
        C[19]:= +3.4732028376500"+011;
        C[20]:= -1.2369602142271"+013;
        CONST:= +9.1893853320467"-001;
        "COMMENT" CONST = LN(SQRT(2 * PI));
        DEN:= W; W2:= W * W;
        PRESUM:=(W - .5) * LN(W) - W + CONST;
        "FOR" I:= 1 "STEP" 1 "UNTIL" 20 "DO"
        "BEGIN" SUM:= PRESUM + C[I] / DEN;
            "IF" SUM = PRESUM "THEN" "GOTO" UIT;
            DEN:= DEN * W2; PRESUM:= SUM
        "END";
   UIT: LGM:= SUM
    "END" OF PROCEDURE LGM;

    "IF" T <= 0 "THEN" STATAL3 ERROR("("LOGGAMMA")", 1, T);
    "IF" T > 7 "THEN" LOGGAMMA:= LGM(T) "ELSE"
    "BEGIN" "REAL" F; F:= T;
        "FOR" T:= T + 1 "WHILE" T < 7 "DO" F:= F * T;
        LOGGAMMA:=LGM(T)-LN(F)
    "END";
"END" LOGGAMMA;
          "EOP"
```

TITLE: **Incomplete Beta**

AUTHOR: R. Kaas

INSTITUTE: Mathematical Centre

RECEIVED: 750201

BRIEF DESCRIPTION

The procedure computes the incomplete beta function ratio with parameters ALPHA1 and ALPHA2.

KEYWORDS

Incomplete beta function ratio

CALLING SEQUENCE

Heading

```
"REAL" "PROCEDURE" INCOMPLETE BETA (X, ALPHA1, ALPHA2, EPS);
"VALUE" X, ALPHA1, ALPHA2, EPS;
"REAL" X, ALPHA1, ALPHA2, EPS;
"CODE" 40401;
```

Formal parameters

X: <arithmetic expression>, argument of the function;

ALPHA1: <arithmetic expression>, first shape parameter;

ALPHA2: <arithmetic expression>, second shape parameter;

EPS: <arithmetic expression>, precision.

DATA AND RESULTS

The value of the function is assigned to the procedure identifier INCOMPLETE BETA.

The following error messages may appear:

Errornumber 2	(if ALPHA1 $\leq$0)
Errornumber 3	(if ALPHA2$\leq$0)
Errornumber 4	(if EPS $\leq$0)

PROCEDURES USED

STATAL3 ERROR	STATAL 40100

LANGUAGE

Algol 60

METHOD AND PERFORMANCE

The function is computed as follows:

If ALPHA1+ALPHA2>500, the values of these parameters are decreased step by step until ALPHA1+ALPHA2$\leq$500, using a recurrence relation from Abramowitz and Stegun (1970). The remaining part is computed as the sum of two series of

summations using CACM ALG. 179.
The precision of the result is determined by the parameter EPS.

REFERENCES

[1] M. Abramowitz & I.A. Stegun
Handbook of mathematical functions
Dover Publications, New York, 1970

[2] Collected algorithms from CACM,
Association for computing machinery,
New York, 1975

EXAMPLE OF USE

Program:

```
"BEGIN"
  OUTPUT(61, "("3(Z.6D,/)")",
      INCOMPLETE BETA(.5, 5.0, 5.0, "-10),
      INCOMPLETE BETA(.1, 6.2, 8.3, "-10),
      INCOMPLETE BETA(.6, 8.1, 3.2, "-10))
"END"
```

Output:

```
 .500000
 .000778
 .185051
```

SOURCE TEXT

```
"CODE" 40401;
"REAL" "PROCEDURE" INCOMPLETE BETA(X, P, Q, EPSILON);
"VALUE" X, P, Q, EPSILON; "REAL" X, P, Q, EPSILON;
"BEGIN" "REAL" INC;

    "REAL" "PROCEDURE" MINIB(X, P, Q);
    "VALUE" X, P, Q; "REAL" X, P, Q;
    "BEGIN"
        "REAL" FINSUM, INFSUM, TEMP, TERM, QRECUR, INDEX;
        "BOOLEAN" ALTER;

        "IF" X <= 0.5 "THEN" ALTER:= "FALSE" "ELSE"
        "BEGIN" ALTER:= "TRUE"; TEMP:= P; P:= Q;
            Q:= TEMP; X:= 1 - X
        "END";
        FINSUM:= 0; TERM:= 1; TEMP:= 1 - X;
        QRECUR:= INDEX:= Q;
        "FOR" INDEX:= INDEX - 1 "WHILE" INDEX > 0 "DO"
        "BEGIN" QRECUR:= INDEX;
            TERM:= TERM * (QRECUR + 1) /
                   (TEMP * (P + QRECUR));
            FINSUM:= FINSUM + TERM
        "END";
        INFSUM:= TERM:= 1; INDEX:= 0;
        "FOR" INDEX:= INDEX + 1
        "WHILE" (TERM / INFSUM) > EPSILON "DO"
        "BEGIN" TERM:= TERM * X * (INDEX - QRECUR) *
            (P + INDEX - 1) / (INDEX * (P + INDEX));
            INFSUM:= INFSUM + TERM
        "END";
        TEMP:= X ** P * (INFSUM * EXP(LOGGAMMA(QRECUR + P) -
        LOGGAMMA(QRECUR) - LOGGAMMA(P + 1)) + FINSUM *
        (1 - X) ** Q * EXP(LOGGAMMA(P + Q) - LOGGAMMA(P) -
        LOGGAMMA(Q + 1)));
        MINIB:= "IF" ALTER "THEN" 1 - TEMP "ELSE" TEMP
    "END";

    "REAL" "PROCEDURE" MAXIB(X, P, Q);
    "VALUE" X; "REAL" X, P, Q;
    "BEGIN" "REAL" L1, L2, PCUM;
        "IF" Q < P
        "THEN" MAXIB:= 1 - MAXIB(1 - X, Q, P) "ELSE"
        "BEGIN" PCUM:= 0; L1:= LN(X); L2:= LN(1 - X);
            "FOR" P:= P - 1 "WHILE" P >= 250 "DO"
            PCUM:= PCUM + EXP(LOGGAMMA(P + Q) -
            LOGGAMMA(P + 1) - LOGGAMMA(Q) + P*L1 + Q*L2);
            P:= P + 1;
            "FOR" Q:= Q - 1 "WHILE" P + Q >= 500 "DO"
            PCUM:= PCUM - EXP(LOGGAMMA(P + Q) - LOGGAMMA(P)
            - LOGGAMMA (Q + 1) + P * L1 + Q * L2);
            Q:= Q + 1;
            MAXIB:= MINIB(X, P, Q) - PCUM
        "END"
    "END";

    "IF" P <= 0 "THEN"
    STATAL3 ERROR("("INCOMPLETE BETA")", 2, P) "ELSE"
    "IF" Q <= 0 "THEN"
    STATAL3 ERROR("("INCOMPLETE BETA")", 3, Q) "ELSE"
    "IF" EPSILON <= 0 "THEN"
    STATAL3 ERROR("("INCOMPLETE BETA")", 4, EPSILON);
    INC:= "IF" X <= 0 "THEN" 0 "ELSE"
          "IF" X >= 1 "THEN" 1 "ELSE"
    "IF" P + Q > 500 "THEN" MAXIB(X, P, Q)
    "ELSE" MINIB(X, P, Q);
    INCOMPLETE BETA:= "IF" INC < 0 "THEN" 0 "ELSE"
        "IF" INC > 1 "THEN" 1 "ELSE" INC
"END" INCOMPLETE BETA;
         "EOP"
```

TITLE: **Wilcoxons W**

AUTHOR: R. Kaas

INSTITUTE: Mathematical Centre

RECEIVED: 760901

BRIEF DESCRIPTION

The procedure is a technical one, which is used in the computation of Wilcoxon's one- or two-sample test statistic. Given two segments of arrays of numbers the procedure computes twice the number of times a number from the first segment is greater than a number from the second segment plus the number of times a number from the first segment is equal to a number from the second segment.

KEYWORDS
Wilcoxon's test statistic

CALLING SEQUENCE

Heading

```
"INTEGER" "PROCEDURE" WILCOXONS W (X, XL, UX, Y, LY, UY, SORTED, D);
"VALUE" LX, UX, LY, UY, SORTED;
"INTEGER" LX, UX, LY, UY;
"BOOLEAN" SORTED;
"ARRAY" X, Y;
"REAL" D;
"CODE" 40000;
```

Formal parameters

X: <array identifier>, one-dimensional array containing the first segment;

LX: <integer arithmetic expression> smallest index of the first segment;

UX: <integer arithmetic expression>, largest index of the first segment;

Y: <array identifier>, one-dimensional array containing the second segment;

LY: <integer arithmetic expression>, smallest index of the second segment;

UY: <integer arithmetic expression>, largest index of the second segment;

SORTED: <boolean expression>, to indicate whether the segments are sorted in non-decreasing order or not;

D: <real variable>, output parameter, sum of the cubes of the sizes of the ties.

DATA AND RESULTS

The value of the test statistic is assigned to the procedure identifier WILCOXONS W.
The following error messages may appear:

Errornumber 1	(if LX>UX)
Errornumber 4	(if LY>UY)

PROCEDURES USED

VEC QSORT	STATAL 11020
STATAL3 ERROR	STATAL 40100

LANGUAGE

Algol 60

METHOD AND PERFORMANCE

The computation of WILCOXONS W is straightforward, using midranks in case of ties.

REFERENCE

[1] D. Wabeke & C. van Eeden
Handleiding voor de toets van Wilcoxon
Mathematical Centre report SW 4/70
Mathematical Centre, Amsterdam, 1970

EXAMPLE OF USE

Program:

```
"BEGIN" "ARRAY" A[1:8], B[1:6], C[1:10]; "REAL" D;

  INARRAY(60, A); INARRAY(60, B); INARRAY(60, C);
  OUTPUT(61, "("3(2ZD,/)")",
      WILCOXONS W(A, 1, 8, B, 1, 6, "TRUE", D),
      WILCOXONS W(A, 1, 5, C, 1,10, "TRUE", D),
      WILCOXONS W(B, 1, 4, C, 1, 9, "TRUE", D))
"END"
```

Input:

```
 8  11  14  16  18  19  22  25
 9  10  12  16  17  22
 6   9  10  11  13  14  15  19  21  23
```

Output:

```
60
48
30
```

Source text

```
"CODE" 40000;
"INTEGER" "PROCEDURE" WILCOXONS W(X, XLOW, XUPP, Y,
    YLOW, YUPP, SORTED, D);
"VALUE" XLOW, XUPP, YLOW, YUPP, SORTED;
"INTEGER" XLOW, XUPP, YLOW, YUPP; "BOOLEAN" SORTED;
"REAL" D; "REAL" "ARRAY" X, Y;
"BEGIN" "INTEGER" W, I, J, MNU, NNU, TNU, MTOT;
    "REAL" A, B, MIN;


    "IF" XLOW > XUPP "THEN"
    STATAL3 ERROR("("WILCOXONS W")", 1, XLOW - XUPP);
    "IF" YLOW > YUPP "THEN"
    STATAL3 ERROR("("WILCOXONS W")", 4, YLOW - YUPP);
    "IF" "NOT" SORTED "THEN"
    "BEGIN" VEC QSORT(X, XLOW, XUPP);
        VEC QSORT(Y, YLOW, YUPP)
    "END";
    W:= 0; D:= 0; MTOT:= XUPP - XLOW + 1;
    I:= XLOW; J:= YLOW; A:= X[I]; B:= Y[J];
    "FOR" MIN:= "IF" A < B "THEN" A "ELSE" B
                "WHILE" I <= XUPP & J <= YUPP "DO"
    "BEGIN" MNU:= I; NNU:= J;
        "FOR" A:= X[I] "WHILE" A = MIN & I < XUPP,
            A "WHILE" A = MIN & I = XUPP "DO" I:= I + 1;
        "FOR" B:= Y[J] "WHILE" B = MIN & J < YUPP,
            B "WHILE" B = MIN & J = YUPP "DO" J:= J + 1;
        MNU:= I - MNU; NNU:= J - NNU;
        TNU:= MNU + NNU; D:= D + TNU * TNU * TNU;
        W:= W + NNU * (2 * MTOT - MNU); MTOT:= MTOT - MNU
    "END";
    "IF" I <= XUPP "THEN"
    "BEGIN" MIN:= A; MNU:= 1;
        "FOR" I:= I + 1 "STEP" 1 "UNTIL" XUPP "DO"
        "BEGIN" A:= X[I];
            "IF" A = MIN "THEN" MNU:= MNU + 1 "ELSE"
            "BEGIN" D:= D + MNU * MNU * MNU;
                MNU:= 1; MIN:= A
            "END"
        "END"; D:= D + MNU * MNU * MNU
    "END";
    "IF" J <= YUPP "THEN"
    "BEGIN" MIN:= B; NNU:= 1;
        "FOR" J:= J + 1 "STEP" 1 "UNTIL" YUPP "DO"
        "BEGIN" B:= Y[J];
            "IF" B = MIN "THEN" NNU:= NNU + 1 "ELSE"
            "BEGIN" D:= D + NNU * NNU * NNU;
                NNU:= 1; MIN:= B
            "END"
        "END"; D:= D + NNU * NNU * NNU
    "END";
    WILCOXONS W:= W
"END" WILCOXONS W;
        "EOP"
```

TITLE: **Limit**

AUTHOR: E. Opperdoes

INSTITUTE: Mathematical Centre

RECEIVED: 760901

BRIEF DESCRIPTION
The procedure computes asymptotic distribution function of the Cramer-von Mises' test statistic

KEYWORDS
Asymptotic distribution Cramer-von Mises' test statistic

CALLING SEQUENCE
Heading

```
"REAL" "PROCEDURE" LIMIT (X);
"VALUE" X;
"REAL" X;
"ALGOL" LIMIT;
```

Formal parameters
X: <arithmetic expression>, argument of the distribution function.

DATA AND RESULTS
The value of the distribution function is assigned to the procedure identifier LIMIT.

LANGUAGE
Algol 60

PROCEDURES USED

BESS KA01	NUMAL 35191

METHOD AND PERFORMANCE
The distribution function is computed using a formula in Anderson and Darling (1952), and has a precision of 10^{-6}.

REFERENCE

[1] T.W. Anderson & D.A. Darling
Asymptotic theory of certain goodness of fit criteria based on stochastic processes
Ann. of Math. Stat., 23 (1952), pp. 193-212.

Example of use

Program:

```
"BEGIN"
  OUTPUT(61, "("3(2ZD.5D,/)")",
      LIMIT(0.1),
      LIMIT(0.2),
      LIMIT(0.5))
"END"
```

Output:

```
  0.41513
  0.73253
  0.96017
```

Source text

```
"ALGOL" LIMIT;
"REAL" "PROCEDURE" LIMIT(X); "VALUE" X; "REAL" X;
"BEGIN" "REAL" SUM, INV16X, A, TERM, TOL, FACTINV;
    "INTEGER" J, J4P1;

    "REAL" "PROCEDURE" BESS(X); "VALUE" X; "REAL" X;
    "BEGIN" "REAL" KA, KA1;
        BESS KA01(0.25, X, KA, KA1);
        BESS:= KA * EXP(-X)
    "END" BESS;

    INV16X:= 1 / 16 / X; J4P1:= 5; A:= 0.5;
    TOL:= 0.5"-6; J:= 1;
    FACTINV:= 4 * ARCTAN(1) * SQRT (X);
    SUM:= BESS(INV16X);
    TERM:= 0.5 * SQRT(5) * BESS(25 * INV16X);
    "FOR" J:= J + 1 "WHILE" TERM >= TOL "DO"
    "BEGIN" SUM:= SUM + TERM;
        "IF" SUM > FACTINV "THEN"
        "BEGIN" LIMIT:= 1; "GOTO" EXIT "END";
        A:= A * (1 - 0.5 / J); J4P1:= J4P1 + 4;
        TERM:= A * SQRT (J4P1) * BESS(J4P1 * J4P1 * INV16X);
    "END";
    SUM:= SUM / FACTINV;
    LIMIT:= "IF" SUM <= 0 "THEN" 0 "ELSE"
             "IF" SUM >= 1 "THEN" 1 "ELSE" SUM;
  EXIT:
"END" LIMIT;
"EOP"
```

TITLE: **Bound**

AUTHOR: A. Nonymous

INSTITUTE: Mathematical Centre

RECEIVED: unknown

BRIEF DESCRIPTION

The procedure computes the upper bound of the confidence interval for the probability P, given the number of successes in a sequence of N independent experiments with probability P of succes.

KEYWORDS
Upper confidence bound for a binomial probability

CALLING SEQUENCE

Heading

```
"REAL" "PROCEDURE" BOUND (T, N, ALPHA, TOL, TEXT);
"VALUE" T, N, ALPHA, TOL;
"INTEGER" T, N;
"REAL" ALPHA, TOL;
"STRING" TEXT;
"ALGOL" BOUND;
```

Formal parameters

T: <integer arithmetic expression>, number of successes;

N: <integer arithmetic expression>, number of experiments;

ALPHA: <arithmetic expression>, one minus confidence coefficient;

TOL: <arithmetic expression>, precision of the upperbound to be computed;

TEXT: <string>, name of procedure to be printed by STATAL3 ERROR when no upperbound is found.

DATA AND RESULTS

The value of the upperbound is assigned to the procedure identifier BOUND. If no upper bound can be found, an error message, refering to errornumber 0 in the procedure with name given in the string TEXT, appears.

PROCEDURES USED

ZEROINDER	NUMAL 34453
LOGGAMMA	STATAL 40400
STATAL3 ERROR	STATAL 40100

Language

Algol 60

Method and performance

The procedure computes a zero P of the function

$$\sum_{K=0}^{T} \left[\frac{N}{K}\right] P^{K}(1-P)^{N-K} - \mathrm{ALPHA}$$

using derivatives.

Source text

```
"ALGOL" BOUND;
"REAL" "PROCEDURE" BOUND(T, N, ALPHA, TOL, TEXT);
"VALUE" T, N, ALPHA, TOL;
"INTEGER" T, N; "REAL" ALPHA, TOL; "STRING" TEXT;
"BEGIN" "INTEGER" NT, N1, K, NT1;
    "REAL" AK, X, Y, LOG CONST;
    "ARRAY" A[0 : T];

    "REAL" "PROCEDURE" LOW ARG(X); "VALUE" X; "REAL" X;
    "BEGIN" "REAL" X1, Q, S; "INTEGER" I;
        S:= 0; X1:= 1 - X; Q:= X / X1;
        "FOR" I:= 0 "STEP" 1 "UNTIL" T "DO"
            S:= S * Q + A[I];
        LOW ARG:= Q ** NT * X1 ** N * S
    "END" LOW ARG;

    "REAL" "PROCEDURE" HIGH ARG(X); "VALUE" X; "REAL" X;
    "BEGIN" "REAL" Q, S; "INTEGER" I;
        S:= 0; Q:= (1 - X) / X;
        "FOR" I:= T "STEP" -1 "UNTIL" 0 "DO"
            S:= S * Q + A[I];
        HIGH ARG:= X ** N * S
    "END" HIGH ARG;

    NT:= N - T; N1:= N + 1; NT1:= NT - 1;
    Y:= AK:= A[0]:= 1;
    "FOR" K:= 1 "STEP" 1 "UNTIL" T "DO"
    "BEGIN" AK:= A[K]:= AK * (N1 - K) / K;
        Y:= Y + AK
    "END";
    LOG CONST:=
        LOGGAMMA(N1) - LOGGAMMA(T + 1) - LOGGAMMA(NT);
    Y:= (0.5) ** N * Y;
    "IF" Y > ALPHA "THEN"
    "BEGIN" X:= 0; Y:= 0.5;
        "IF" ZEROINDER(X, Y, LOW ARG(X) - ALPHA,
             "IF" X = 0 "THEN" 0 "ELSE"
             EXP(LOG CONST + NT1 * LN(X) + T * LN(1 - X)),
             ABS(X) * TOL + TOL)
        "THEN" BOUND:= (X + Y) / 2
        "ELSE" STATAL3 ERROR(TEXT, 0, X)
    "END" "ELSE"
    "BEGIN" X:= 0.5; Y:= 1;
        "IF" ZEROINDER(X, Y, HIGH ARG(X) - ALPHA,
             "IF" X = 1 "THEN" 0 "ELSE"
             EXP(LOG CONST + NT1 * LN(X) + T * LN(1 - X)),
             ABS(X) * TOL + TOL)
        "THEN" BOUND:= (X + Y) / 2
        "ELSE" STATAL3 ERROR(TEXT, 0, X)
    "END";
"END" BOUND;
"EOP"
```

TITLE: **Inverse**

AUTHORS: P. van der Tweel, J.M. Buhrman

INSTITUTE: Mathematical Centre

RECEIVED: 760501

BRIEF DESCRIPTION

The procedure computes the argument X, for which a user provided continuous distribution function DISTR has a given value PROB.

KEYWORDS

Inverse continuous distribution function

CALLING SEQUENCE

Heading

```
"REAL" "PROCEDURE" INVERSE (X, DISTR, PROB, TOLX);
"VALUE" PROB, TOLX;
"REAL" X, DISTR, PROB, TOLX;
"CODE" 40001;
```

Formal parameters

X: <real variable>, estimate of INVERSE (X, DISTR, PROB, TOLX) and Jensen parameter for DISTR and TOLX;

DISTR: <arithmetic expression>, distribution function to be inverted, depending on X as argument;

PROB: <arithmetic expression>, left hand tail probability of the value to be computed;

TOLX: <arithmetic expression>, relative precision, depending on X.

DATA AND RESULTS

Before calling the procedure, X should have an estimated value of the inverse of the distribution function. If little is known about the location of the distribution the value 0 is recommended as an initial estimate for X.

The value of the inverse distribution function is assigned to the procedure identifier INVERSE.

The following error messages may appear:

Errornumber 0	(if no satisfactory X can be found)
Errornumber 3	(if PROB $\leqslant 10^{-14}$ or PROB $\geqslant 1-10^{-14}$)

PROCEDURES USED

STATAL3 ERROR	STATAL 40100

Language

Algol 60

Method and performance

An interval containing a zero of the function DISTR (X)−PROB is computed. On this interval the procedure ZEROIN from Dekker (1970) is used to determine the zero.

Reference

[1] T.J. Dekker
Algol 60 procedures in numerical algebra
Mathematical Centre Tracts 22
Mathematical Centre, Amsterdam, 1970

Example of use

Program:

```
"BEGIN" "REAL" X1, X2, X3;
  X1:= -5; X2:= .7; X3:= 2;

  OUTPUT(61, "("3(+Z.6D,/)")",
      INVERSE(X1, LOGISTIC(X1,  1,  2), .100,
                  ABS(X1) * "-10),
      INVERSE(X2, FISHER  (X2, 10, 18), .338,
                  ABS(X2) * "-10),
      INVERSE(X3, NORMAL  (X3, 10, 12), .250,
                  ABS(X3) * "-10))
"END"
```

Output:

```
-3.394449
 +.762042
+1.906123
```

Source text

```
"CODE" 40001;
"REAL" "PROCEDURE" INVERSE(X, FX, PROB, TOLX);
"VALUE" PROB, TOLX; "REAL" X, FX, PROB, TOLX;
"BEGIN" "REAL" Y, H, F1, F2;

    "BOOLEAN" "PROCEDURE" ZEROIN(X, Y, FX, TOLX);
    "REAL" X, Y, FX, TOLX;
    "BEGIN" "INTEGER" EXT;
        "REAL" C, FC, B, FB, A, FA, D, FD, FDB, FDA, W, MB,
        TOL, M, P, Q, DW;
        DW:= MINREAL; B:= X; FB:= FX; A:= X:= Y; FA:= FX;
    INTERPOLATE: C:= A; FC:= FA; EXT:= 0;
    EXTRAPOLATE: "IF" ABS(FC) < ABS(FB) "THEN"
```

```
        "BEGIN" "IF" C ^= A "THEN"
            "BEGIN" D:= A; FD:= FA "END";
            A:= B; FA:= FB; B:= X:= C;
            FB:= FC; C:= A; FC:= FA
        "END" INTERCHANGE;
        TOL:= TOLX; M:= (C + B) * 0.5; MB:= M - B;
        "IF" ABS(MB) > TOL "THEN"
        "BEGIN" "IF" EXT > 2 "THEN" W:= MB "ELSE"
            "BEGIN" TOL:= TOL * SIGN(MB);
                P:= (B - A) * FB; "IF" EXT <= 1 "THEN"
                Q:= FA - FB "ELSE"
                "BEGIN" FDB:= (FD - FB) / (D - B);
                    FDA:= (FD - FA) / (D - A);
                    P:= FDA * P; Q:= FDB * FA - FDA * FB
                "END"; "IF" P < 0 "THEN"
                "BEGIN" P:= -P; Q:= -Q "END";
                W:= "IF" P < DW "OR" P <= Q * TOL "THEN" TOL
                "ELSE"
                "IF" P < MB * Q "THEN" P / Q "ELSE" MB
            "END"; D:= A; FD:= FA; A:= B; FA:= FB;
            X:= B:= B + W; FB:= FX;
            "IF"
            ("IF" FC >= 0 "THEN" FB >= 0 "ELSE" FB <= 0)
            "THEN" "GOTO" INTERPOLATE "ELSE"
            "BEGIN" EXT:=
                "IF" W = MB "THEN" 0 "ELSE" EXT + 1;
                "GOTO" EXTRAPOLATE
            "END"
        "END"; Y:= C;
        ZEROIN:= "IF" FC >= 0 "THEN" FB <= 0 "ELSE" FB >= 0
    "END" ZEROIN;

    "IF" PROB <= "-14 "OR" PROB >= 1 - "-14 "THEN"
      STATAL3 ERROR("("INVERSE")",3,PROB);
    F1:= FX - PROB;
    "IF" ABS(F1) < TOLX "THEN" INVERSE:= X "ELSE"
    "BEGIN" H:= "IF" F1 > 0 "THEN" -.5 "ELSE" .5;
        Y:= X; X:= X + H; F2:= FX - PROB;
     L: "IF" ABS(F2) < TOLX "THEN" INVERSE:= X "ELSE"
        "IF" F1 * F2 > 0 "THEN"
        "BEGIN" Y:= X; H:= H * 2; X:= X + H;
            F1:= F2; F2:= FX - PROB;
            "GOTO" L
        "END" "ELSE"
        INVERSE:= "IF" ZEROIN(X, Y, FX - PROB, TOLX)
        "THEN" X "ELSE"
        STATAL3 ERROR("("INVERSE")", 0, PROB)
    "END"
"END" INVERSE;
         "EOP"
```

TITLE: **Truncate Left**

AUTHOR: H. Elffers

INSTITUTE: Mathematical Centre

RECEIVED: 750205

BRIEF DESCRIPTION

The procedure computes the probability that a random variable with a specified continuous distribution function DISTR is not larger than a given value X, provided that it is not smaller than a given value LEFT.

KEYWORDS

Left-sided truncated distribution function

CALLING SEQUENCE

Heading

```
"REAL" "PROCEDURE" TRUNCATE LEFT (X, LEFT, Y, DISTR);
"VALUE" X, LEFT;
"REAL" X, LEFT, Y, DISTR;
"CODE" 40402;
```

Formal parameters

X:	<arithmetic expression>, argument of the truncated distribution function;
LEFT:	<arithmetic expression>, truncation point;
Y:	<real variable>, Jensen parameter for DISTR;
DISTR:	<arithmetic expression>, distribution function of the non-truncated distribution, depending on Y as argument.

DATA AND RESULTS

The value of the truncated distribution function is assigned to the procedure identifier TRUNCATE LEFT.

The following error message may appear:

Errornumber 2 (if DISTR (LEFT) $\geq$1)

PROCEDURES USED

STATAL3 ERROR	STATAL 40100

LANGUAGE

Algol 60

Method and performance

The truncated distribution function is computed as follows:

$$\begin{cases}\texttt{(DISTR(X)-DISTR(LEFT))/(1-DISTR(LEFT))} & \text{if } \texttt{X}\geqslant\texttt{LEFT}\\ \texttt{0} & \text{if } \texttt{X<LEFT}\end{cases}$$

Warning:
DISTR must be a continuous distribution function. If this is not the case results are meaningless, and even a run time error of the operating system may occur.

Example of use

Program:

```
"BEGIN" "REAL" Z;

  "COMMENT" THE FOLLOWING VALUES ARE COMPUTED:
  AT 3.59 LEFT SIDED TRUNCATED CHI-SQUARE DISTRIBUTION
  WITH 11 DEGREES OF FREEDOM AND ARGUMENT 11.61,
  AT 0.65 LEFT SIDED TRUNCATED STUDENT DISTRIBUTION
  WITH 18 DEGREES OF FREEDOM AND ARGUMENT 1.74,
  AT 4.562 LEFT SIDED TRUNCATED FISHER DISTRIBUTION
  WITH 7 AND 9 DEGREES OF FREEDOM AND ARGUMENT 16.32;

  OUTPUT(61, "("3(Z.6D,/)")",
      TRUNCATE LEFT(11.61, 3.59 , Z, CHISQ  (Z, 11)),
      TRUNCATE LEFT( 1.74, 0.65 , Z, STUDENT(Z, 18)),
      TRUNCATE LEFT(16.32, 4.562, Z, FISHER (Z, 7, 9)))
"END"
```

Output:

```
 .598476
 .811170
 .990156
```

Source text

```
"CODE" 40402;
"REAL" "PROCEDURE" TRUNCATE LEFT
                   (ARG, LEFT, JENSEN, DISTRIBUTION);
    "VALUE" ARG, LEFT; "REAL" ARG, LEFT, JENSEN, DISTRIBUTION;
"BEGIN" "REAL" DISTR LEFT;
    "IF" ARG < LEFT "THEN" TRUNCATE LEFT:= 0 "ELSE"
    "BEGIN" JENSEN:= LEFT; DISTR LEFT:= DISTRIBUTION;
        "IF" 1 - DISTR LEFT <= 0 "THEN"
        STATAL3 ERROR("("TRUNCATE LEFT")", 2, DISTR LEFT);
        JENSEN:= ARG;
        TRUNCATE LEFT:= (DISTRIBUTION - DISTR LEFT)
            / (1 - DISTR LEFT)
    "END"
"END" TRUNCATE LEFT;
          "EOP"
```

TITLE: **Truncate Right**

AUTHOR: H. Elffers

INSTITUTE: Mathematical Centre

RECEIVED: 750205

BRIEF DESCRIPTION

The procedure computes the probability that a random variable with a specified continuous distribution function DISTR is not larger than a given value X, provided that it is not larger than a given value RIGHT.

KEYWORDS

Right-sided truncated distribution function

CALLING SEQUENCE

Heading

```
"REAL" "PROCEDURE" TRUNCATE RIGHT (X, RIGHT, Y, DISTR);
"VALUE" X, RIGHT;
"REAL" X, RIGHT, Y, DISTR;
"CODE" 40403;
```

Formal parameters

X: <arithmetic expression>, argument of the truncated distribution function;

RIGHT: <arithmetic expression>, truncation point;

Y: <real variable>, Jensen parameter for DISTR;

DISTR: <arithmetic expression>, distribution function of the non-truncated distribution, depending on Y as argument.

DATA AND RESULTS

The value of the truncated distribution function is assigned to the procedure identifier TRUNCATE RIGHT.

The following error message may appear:

Errornumber 2 (if DISTR (RIGHT)$\leqslant$0)

PROCEDURES USED

STATAL3 ERROR STATAL 40100

LANGUAGE

Algol 60

Method and performance

The truncated distribution function is computed as follows:

$$\begin{cases}\texttt{DISTR(X)/DISTR(RIGHT)} & \text{if } \texttt{X}\leqslant\texttt{RIGHT}\\ \texttt{1} & \text{if } \texttt{X>RIGHT}\end{cases}$$

Warning:
DISTR must be a continuous distribution function. If this is not the case results are meaningless, and even a run time error of the operating system may occur.

Example of use

Program:

```
"BEGIN" "REAL" Z;

  "COMMENT" THE FOLLOWING VALUES ARE COMPUTED:
  AT 2.54 RIGHT SIDED TRUNCATED STANDARD NORMAL DISTRIBUTION
  WITH ARGUMENT 1.88,
  AT 0.7 RIGHT SIDED TRUNCATED NORMAL(3, 1.78) DISTRIBUTION
  WITH ARGUMENT -1.4,
  AT 2.5 RIGHT SIDED TRUNCATED LOGNORMAL(2.1, .5)
  DISTRIBUTION WITH ARGUMENT 2.23;

  OUTPUT(61, "("3(Z.6D,/)")",
    TRUNCATE RIGHT(1.88, 2.54 , Z, PHI      (Z)),
    TRUNCATE RIGHT(-1.4, 0.7  , Z, NORMAL   (Z,   3, 1.78)),
    TRUNCATE RIGHT(2.23, 2.5  , Z, LOGNORMAL(Z, 2.1,   .5)))
"END"
```

Output:

```
 .975352
 .068460
 .526538
```

Source text

```
"CODE" 40403;
"REAL" "PROCEDURE" TRUNCATE RIGHT
                   (ARG, RIGHT, JENSEN, DISTRIBUTION);
"VALUE" ARG, RIGHT; "REAL" ARG, RIGHT, JENSEN, DISTRIBUTION;
"BEGIN" "REAL" DISTR RIGHT;
    "IF" ARG > RIGHT "THEN" TRUNCATE RIGHT:= 1 "ELSE"
    "BEGIN" JENSEN:= RIGHT; DISTR RIGHT:= DISTRIBUTION;
        "IF" DISTR RIGHT <= 0 "THEN"
        STATAL3 ERROR("("TRUNCATE RIGHT")", 2, DISTR RIGHT);
        JENSEN:= ARG;
        TRUNCATE RIGHT:= DISTRIBUTION / DISTR RIGHT
    "END"
"END" TRUNCATE RIGHT;
         "EOP"
```

TITLE: **Truncate Twosided**

AUTHOR: H. Elffers

INSTITUTE: Mathematical Centre

RECEIVED: 750205

BRIEF DESCRIPTION

The procedure computes the probability that a random variable with a specified continuous distribution function DISTR is not larger than a given value X, provided that it is not smaller than a given value LEFT and not larger than a given value RIGHT.

KEYWORDS

Two-sided truncated distribution function

CALLING SEQUENCE

Heading

```
"REAL" "PROCEDURE" TRUNCATE TWOSIDED (X, LEFT, RIGHT, Y, DISTR);
"VALUE" X, LEFT, RIGHT;
"REAL" X, LEFT, RIGHT, Y, DISTR;
"CODE" 40404;
```

Formal parameters

X: <arithmetic expression>, argument of the truncated distribution function;

LEFT: <arithmetic expression>, left-hand truncation point;

RIGHT: <arithmetic expression>, right-hand truncation point;

Y: <real variable>, Jensen parameter for DISTR;

DISTR: <arithmetic expression>, distribution function of the non-truncated distribution, depending on Y as argument.

DATA AND RESULTS

The value of the truncated distribution function is assigned to the procedure identifier TRUNCATE TWOSIDED.

The following error messages may appear:

Errornumber 2	(if RIGHT <LEFT)
Errornumber 3	(if DISTR(RIGHT) ≤DISTR(LEFT))

PROCEDURES USED

STATAL3 ERROR	STATAL 40100

Language

Algol 60

Method and performance

The truncated ditribution function is computed as follows:

$$\begin{cases} \texttt{0} & \text{if } \texttt{X<LEFT} \\ \texttt{(DISTR(X)-DISTR(LEFT))} & \texttt{/(DISTR(RIGHT)-DISTR(LEFT))} \\ & \text{if } \texttt{LEFT<X}\leqslant\texttt{RIGHT} \text{ or } \texttt{LEFT}\leqslant\texttt{X<RIGHT} \\ \texttt{1} & \text{if } \texttt{X>RIGHT} \text{ or } \texttt{LEFT=X=RIGHT} \end{cases}$$

Warning:

DISTR must be a continuous distribution function. If this is not the case results are meaningless, and even a run time error of the operating system may occur.

Example of use

Program:

```
"BEGIN" "REAL" Z;

  "COMMENT" THE FOLLOWING VALUES ARE COMPUTED:
  AT 7.32 AND 25.23 TRUNCATED EXPONENTIAL(.56)
  DISTRIBUTION WITH ARGUMENT 11.28,
  AT 4.15 AND 13.36 TRUNCATED LOGISTIC(5.11, 9.65)
  DISTRIBUTION WITH ARGUMENT 12.69,
  AT 8.27 AND 87.44 TRUNCATED CAUCHY(12.27, 9.12)
  DISTRIBUTION WITH ARGUMENT 14.49;

  OUTPUT(61, "("3(Z.6D,/)")",
    TRUNCATE TWOSIDED(11.28, 7.32 , 25.23, Z,
                      EXPON(Z, .56)),
    TRUNCATE TWOSIDED(12.69, 4.15 , 13.36, Z,
                      LOGISTIC(Z, 5.11, 9.65)),
    TRUNCATE TWOSIDED(14.49, 8.27 , 87.44, Z,
                      CAUCHY(Z, 12.27, 9.12)))
"END"
```

Output:

```
 .891169
 .934925
 .349957
```

Source text

```
"CODE" 40404;
"REAL" "PROCEDURE" TRUNCATE TWOSIDED
       (ARG, LEFT, RIGHT, JENSEN, DISTRIBUTION);
"VALUE" ARG, LEFT, RIGHT;
"REAL" ARG, LEFT, RIGHT, JENSEN, DISTRIBUTION;
"BEGIN" "REAL" DISTR LEFT, DISTR RIGHT;
    "IF" RIGHT < LEFT "THEN"
        STATAL3 ERROR("("TRUNCATE TWOSIDED")",
                      2, LEFT - RIGHT) "ELSE"
    "IF" ARG < LEFT "THEN" TRUNCATE TWOSIDED:= 0 "ELSE"
    "IF" ARG > RIGHT ! (ARG = LEFT & ARG = RIGHT) "THEN"
        TRUNCATE TWOSIDED:= 1 "ELSE"
    "BEGIN" JENSEN:= LEFT; DISTR LEFT:= DISTRIBUTION;
        JENSEN:= RIGHT; DISTR RIGHT:= DISTRIBUTION;
        "IF" DISTR RIGHT - DISTR LEFT <= 0 "THEN"
        STATAL3 ERROR("("TRUNCATE TWOSIDED")", 3,
                      DISTR RIGHT - DISTR LEFT);
        JENSEN:= ARG;
        TRUNCATE TWOSIDED:= (DISTRIBUTION - DISTR LEFT) /
                            (DISTR RIGHT - DISTR LEFT)
    "END"
"END" TRUNCATE TWOSIDED;
         "EOP"
```

TITLE: **Statal 3 Error**

AUTHOR: H. Elffers

INSTITUTE: Mathematical Centre

RECEIVED: 750101

BRIEF DESCRIPTION

If a STATAL procedure is called with a wrong value of a parameter, STATAL3 ERROR terminates the execution of the program. The procedure identifier, the errornumber and the wrong value of the parameter are printed. A message is given in the dayfile.

KEYWORDS

Error message

CALLING SEQUENCE

Heading

```
"REAL" "PROCEDURE" STATAL3 ERROR (TEXT, ERNUM, WVAL);
"VALUE" ERNUM, WVAL;
"INTEGER" ERNUM;
"REAL" WVAL;
"STRING" TEXT;
"CODE" 40100;
```

Formal parameters

TEXT: <string>, name of the procedure involved;

ERNUM: <integer arithmetic expression>, errornumber, to be explained in data and results of the description of the procedure involved;

WVAL: <arithmetic expression>, wrong value of the parameter involved.

DATA AND RESULTS

In an output statement the values of the parameters of STATAL3 ERROR are printed, after which the procedure EXIT is called. In general the errornumber is equal to the number of the parameter in the procedure call. The procedure ALGMESS is used to write a message in the dayfile.

PROCEDURES USED

EXIT	STATAL 11010
ALGMESS	STATAL 11017

Language

Algol 60

Example of use

Program:

```
"BEGIN"
  OUTPUT(61,"("Z.6D")", BIN(5, 10, -2.14))
"END"
```

Output:

```
************************************************************
*                                                          *
* STATAL3 ERROR MESSAGE:                                   *
*                                                          *
* NAME OF THE PROCEDURE  : BIN                             *
* ERROR NUMBER           :  3                              *
* VALUE CAUSING THE ERROR:  -2.14000000000000"+000         *
* EXECUTION IS TERMINATED                                  *
*                                                          *
* IN GENERAL THE ERROR NUMBER IS EQUAL TO THE NUMBER OF    *
* THE PARAMETER IN THE PARAMETERLIST.                      *
* FOR FURTHER INFORMATION SEE THE DESCRIPTION OF THE       *
* PROCEDURE IN THE STATAL REFERENCE MANUAL.                *
*                                                          *
************************************************************
```

Source text

```
"CODE" 40100;
"REAL" "PROCEDURE" STATAL3 ERROR(TEXT, ERRORNUMBER, VALUE);
"VALUE" ERRORNUMBER, VALUE;
"STRING" TEXT; "INTEGER" ERRORNUMBER; "REAL" VALUE;
"BEGIN" ALGMESS("("STATAL3 ERROR IN PROCEDURE")");
    ALGMESS(TEXT);
    OUTPUT(61, "("3/,60("("*")"),/,"("*")",58B,"("*")",/,
    "("* STATAL3 ERROR MESSAGE:")",35B,"("*")",/,
    "("*")",58B,"("*")",/,
    "("* NAME OF THE PROCEDURE  : ")",N")", TEXT);
    SYSPARAM(61, 2, 59);

    OUTPUT(61, "(""("*")",/,"("* ERROR NUMBER           :")"
    ,2ZD,30B,"("*")",/,"("* VALUE CAUSING THE ERROR: ")"
    ,N")",ERRORNUMBER, VALUE); SYSPARAM(61, 2, 59);

    OUTPUT(61, "(""("*")",/,"("* EXECUTION IS TERMINATED")"
    ,34B,"("*")",/,"("*")",58B,"("*")",/,
    "("* IN GENERAL THE ERROR NUMBER ")",
    "("IS EQUAL TO THE NUMBER OF    *")",/
    "("* THE PARAMETER IN THE ")",
    "("PARAMETERLIST.                     *")",/
    "("* FOR FURTHER INFORMATION SEE ")",
    "("THE DESCRIPTION OF THE        *")",/
    "("* PROCEDURE IN THE STATAL REFERENCE ")",
    "("MANUAL.               *")",/
    "("*")",58B,"("*")",/,60("("*")")")");
    STATAL3 ERROR:= 0;
    STOP
"END" STATAL3 ERROR;
"EOP"
```

TITLE: **Channel Cards**

AUTHOR: A. Nonymous

INSTITUTE: Mathematical Centre

RECEIVED: unknown

BRIEF DESCRIPTION
The procedure reads channel cards via the channel FROM.

KEYWORDS
Channel cards

CALLING SEQUENCE
Heading

```
"PROCEDURE" CHANNEL CARDS (FROM);
"VALUE" FROM;
"INTEGER" FROM;
"ALGOL" CCARDS;
```

Formal parameters

FROM: <integer arithmetic expression>, channel number from which the channel cards must be read.

DATA AND RESULTS
The channel FROM must be defined in the program. The channel cards must be defined as in version 3 of CDC-ALGOL.
The following error messages may appear in the dayfile; each error terminates execution of the program:
channel cards syntax error
identifier exeeds 7 characters
duplicate option on channel card
(first) channel already defined
second channel not defined
CHANNEL, END **missing**
conflicting options on channel card
illegal options on channel card

PROCEDURES USED

ALGMESS	STATAL 11017

LANGUAGE
Algol 60

Reference

[1] ALGOL 60 reference manual, version 3,
Control Data Corporation, 1972.

Source text

```
"ALGOL" CCARDS;
"PROCEDURE" CHANNEL CARDS(FROM);
"VALUE" FROM; "INTEGER" FROM;
"BEGIN" "INTEGER" CHN1, CHN2, I, CURRENT, FILE NAME, OPTION,
                  P, PP, K, PFROM;
    "INTEGER" "ARRAY" C[1 : 73];

    "PROCEDURE" ERROR(MESS); "STRING" MESS;
    "BEGIN"
        OUTPUT(61, "("/,N,/,"("JOB ABORT")",3/")", MESS);
        ALGMESS(MESS); STOP
    "END" ERROR;

    "PROCEDURE" SYNTAX ERROR;
    ERROR("("CHANNEL CARD SYNTAX ERROR")");

    "PROCEDURE" DUPLICATE OPTION;
    ERROR("("DUPLICATE OPTION ON CHANNEL CARD")");

    "BOOLEAN" "PROCEDURE" ALPHA;
    ALPHA:= CURRENT <= EQUIV("("9")");

    "BOOLEAN" "PROCEDURE" LETTER;
    LETTER:= CURRENT <= EQUIV("("Z")");

    "BOOLEAN" "PROCEDURE" DIGIT;
    DIGIT:= CURRENT <= EQUIV("("9")") "AND"
            CURRENT > EQUIV("("Z")");

    "INTEGER" "PROCEDURE" POSITIVE NUMBER(I); "INTEGER" I;
    "BEGIN" "INTEGER" NUMBER;
        NUMBER:= 0;
        "IF" I > 73 "THEN" "GOTO" END WHILE;
    WHILE: CURRENT:= C[I];
        "IF" DIGIT "THEN"
        "BEGIN"
            NUMBER:= NUMBER * 10 + CURRENT // 2 ** 42 - 27;
            I:= I + 1;
            "IF" I <= 73 "THEN" "GOTO" WHILE
        "END";
    END WHILE: POSITIVE NUMBER:= NUMBER
    "END" POSITIVE NUMBER;

    "INTEGER" "PROCEDURE" IDENTIFIER(I); "INTEGER" I;
    "BEGIN" "INTEGER" NAME, SHIFT, LENGTH;
        NAME:= 0;
        "IF" I > 73 "THEN" "GOTO" END WHILE;
        SHIFT:= 1; LENGTH:= 0;
 WHILE: CURRENT:= C[I];
        "IF" ALPHA "THEN"
        "BEGIN" NAME:= NAME + CURRENT // SHIFT;
            SHIFT:= SHIFT * 64; I:= I + 1;
            LENGTH:= LENGTH + 1;
            "IF" LENGTH > 7 "THEN"
            ERROR("("IDENTIFIER EXCEEDS 7 CHARACTERS")");
            "IF" I <= 73 "THEN" "GOTO" WHILE
        "END";
END WHILE: IDENTIFIER:= NAME
    "END" IDENTIFIER;

    "INTEGER" "PROCEDURE" LETTERSTRING(I); "INTEGER" I;
    "BEGIN" "INTEGER" STRING, SHIFT;
        STRING:= 0;
        "IF" I > 73 "THEN" "GOTO" END WHILE;
        SHIFT:= 1;
 WHILE: CURRENT:= C[I];
        "IF" LETTER "THEN"
        "BEGIN" STRING:= STRING + CURRENT // SHIFT;
            SHIFT:= SHIFT * 64; I:= I + 1;
            "IF" I <= 73 "THEN" "GOTO" WHILE
        "END";
END WHILE: LETTERSTRING:= STRING
    "END" LETTERSTRING;

    "PROCEDURE" CHANNEL EQUATE CARD;
    "BEGIN" "IF" CHEXIST(CHN1) "THEN"
        ERROR("("FIRST CHANNEL ALREADY DEFINED")");
        "IF" "NOT" CHEXIST(CHN2) "THEN"
        ERROR("("SECOND CHANNEL NOT DEFINED")");
        CHANNEL(CHN1, "("E")", CHN2);
        "GOTO" NEW CHANNEL CARD
    "END" CHANNEL EQUATE CARD;

    "PROCEDURE" BINARY SEQUENTIAL;
    "BEGIN" "IF" CHEXIST(CHN1) "THEN"
        ERROR("("CHANNEL ALREADY DEFINED")");
        CHANNEL(CHN1, "("B")", "("ILF")", FILE NAME);
        "GOTO" NEW CHANNEL CARD
    "END" BINARY SEQUENTIAL;

    "PROCEDURE"  IO CHANNEL;
    "BEGIN" "INTEGER" FL;
        "IF" CHEXIST(CHN1) "THEN"
        ERROR("("CHANNEL ALREADY DEFINED")");
        K:= ABS(K); "IF" PP = -1 "THEN" PP:= 0;
        "IF" PP = 0 "THEN"
        FL:= "IF" P = 80 "THEN" 138 "ELSE" 80
        "ELSE"
        FL:= "IF" P = -1 "THEN" 137 "ELSE" P + 1;
        "IF" P = -1 "THEN"
        P:= "IF" PP = 0 "THEN" 80 "ELSE" 136;
        CHANNEL(CHN1, "("C")", "("ILF")", FILE NAME,
            "("P")", P, "("PP")", PP, "("K")", K, "("FL")"
            , FL);
        "GOTO" NEW CHANNEL CARD
    "END" IO CHANNEL;

    EOF(FROM, CHANNEL END);
    SYSPARAM(FROM, 5, PFROM); SYSPARAM(FROM, 6, 80);
NEW CHANNEL CARD:
    INPUT(FROM, "("H,72(A)")", C);
    OUTPUT(61, "("H,72(A),/")", C);

    "IF" C[1] ^= EQUIV("("CHANNEL,")") "THEN"
    ERROR("("CHANNEL,END MISSING")");

    "IF" C[2] = EQUIV("("E")")
        "AND" C[3] = EQUIV("("N")")
        "AND" C[4] = EQUIV("("D")")
    "THEN" "GOTO" CHANNEL END;

    I:= 2; CHN1:= POSITIVE NUMBER(I);
    "IF" CHN1 = 0 "THEN" SYNTAX ERROR;
    "IF" CURRENT ^= EQUIV("("=")") "THEN" SYNTAX ERROR;

    I:= I + 1;
    "IF" I > 73 "THEN" SYNTAX ERROR;
    CURRENT:= C[I];
    "IF" DIGIT "THEN"
    "BEGIN" CHN2:= POSITIVE NUMBER(I);
        "IF" CHN2 = 0 "THEN" SYNTAX ERROR;
        CHANNEL EQUATE CARD
    "END";

    FILE NAME:= IDENTIFIER(I);
    "IF" FILE NAME = 0 "THEN" SYNTAX ERROR;
    "IF" CURRENT = EQUIV("(" ")") "THEN" BINARY SEQUENTIAL
    "ELSE"
    "IF" CURRENT ^= EQUIV("(",")") "THEN" SYNTAX ERROR;

    P:= PP:= -1; K:= -2;
OPTIONS: I:= I + 1;
    "IF" I > 73 "THEN" SYNTAX ERROR;
    OPTION:= LETTERSTRING(I);
    "IF" OPTION = 0 "THEN" SYNTAX ERROR "ELSE"
    "IF" OPTION = EQUIV("("P")") "THEN"
    "BEGIN" "IF" P = -1 "THEN" P:= POSITIVE NUMBER(I)
                           "ELSE" DUPLICATE OPTION
    "END" "ELSE"
    "IF" OPTION = EQUIV("("PP")") "THEN"
    "BEGIN" "IF" PP = -1 "THEN" PP:= POSITIVE NUMBER(I)
                         "ELSE" DUPLICATE OPTION
    "END" "ELSE"
    "IF" OPTION = EQUIV("("K")") "THEN"
    "BEGIN" "IF" K = -2 "THEN" K:= POSITIVE NUMBER(I)
```

```
                        "ELSE" DUPLICATE OPTION
    "END" "ELSE"
    "IF" OPTION = EQUIV("("A")") "THEN"
    "BEGIN" "IF" "NOT" (PP = -1 "AND" P = -1 "AND" K = -2)
        "THEN"
        ERROR("("CONFLICTING OPTIONS ON CHANNEL CARD")");
        BINARY SEQUENTIAL;
    "END" "ELSE"
    "IF" OPTION ^= EQUIV("("R")") "THEN"
    ERROR("("ILLEGAL OPTION ON CHANNEL CARD")");
    "IF" CURRENT = EQUIV("(",")") "THEN" "GOTO" OPTIONS
    "ELSE"
    "IF" CURRENT = EQUIV("(" ")") "THEN" IO CHANNEL
                                  "ELSE" SYNTAX ERROR;

CHANNEL END: SYSPARAM(FROM, 6, PFROM);
"END" CHANNEL CARDS;
"EOP"
```

TITLE: **Open Scratch**

AUTHOR: A. Nonymous

INSTITUTE: Mathematical Centre

RECEIVED: unknown

BRIEF DESCRIPTION

The procedure opens a scratch file.

KEYWORDS

Scratch file

CALLING SEQUENCE

Heading

```
"INTEGER" "PROCEDURE" OPEN SCRATCH (NAME);
"STRING" NAME;
"ALGOL" OSCR;
```

Formal parameters

NAME: <string>, identifier of the scratch file.

DATA AND RESULTS

A channel number, which is not already in use, is assigned to the procedure identifier OPEN SCRATCH.

The following error message may appear:

Errornumber 1 (if NAME is not a file identifier)

PROCEDURES USED

STATAL3 ERROR STATAL 40100

LANGUAGE

Algol 60

SOURCE TEXT

```
"ALGOL" OSCR;
"INTEGER" "PROCEDURE" OPEN SCRATCH(NAME); "STRING" NAME;
"BEGIN" "INTEGER" CHN, L, C;
    "COMMENT" IS NAME AN IDENTIFIER?;
    L:= LENGTH(NAME);
    "IF" L = 0 "OR" L > 7 "THEN"
    STATAL ERROR("("OPEN SCRATCH")", 1, 0);
    STRING ELEMENT(NAME,
        1, "("ABCDEFGHIJKLMNOPQRSTUVWXYZ")", C);
    "IF" C = 0 "THEN"
    STATAL ERROR("("OPEN SCRATCH")", 1, 0);
    "FOR" L:= L "STEP" -1 "UNTIL" 2 "DO"
    "BEGIN" STRING ELEMENT(NAME, L,
           "("ABCDEFGHIJKLMNOPQRSTUVWXYZ")"
           "("0123456789")", C);
        "IF" C = 0 "THEN"
        STATAL ERROR("("OPEN SCRATCH")", 1, 0);
    "END";
    "COMMENT" LOOK FOR A CHANNEL AND DEFINE ONE. ;
    CHN:= 0;
    "FOR" CHN:= CHN + 1 "WHILE" CHEXIST(CHN) "DO";
    CHANNEL(CHN, "("W")", "("LFN")", NAME);
    OPEN(CHN, "("IO")"); OPEN SCRATCH:= CHN
"END" OPEN SCRATCH;
"EOP"
```

TITLE: **Close Scratch**

AUTHOR: A. Nonymous

INSTITUTE: Mathematical Centre

RECEIVED: unknown

BRIEF DESCRIPTION
The procedure closes a scratch file.

KEYWORDS
Scratch file

CALLING SEQUENCE
Heading

```
"PROCEDURE" CLOSE SCRATCH (FILENR);
"VALUE" FILENR;
"INTEGER" FILENR;
"ALGOL" CSCR;
```

Formal parameters

FILENR: <integer arithmetic expression>, channel number of the scratch file.

SOURCE TEXT

```
"ALGOL" CSCR;
"PROCEDURE" CLOSE SCRATCH(FILENR);
"VALUE" FILENR; "INTEGER" FILENR;
RETURN(FILENR);
"EOP"
```

Title: **Algmess**

Brief description

The procedure writes the string `MESS` as a message in the dayfile.

Calling sequence

Heading

```
"PROCEDURE" ALGMESS (MESS);
"STRING" MESS;
"CODE" 11017;
```

Formal parameters

`MESS`: <string>, text to be written in dayfile.

Language

Compass

Source text

The procedure is written in COMPASS; an equivalent ALGOL 60 text is given.

```
"CODE" 11017;
"PROCEDURE" ALGMESS(TEXT); "STRING" TEXT;
"BEGIN" "INTEGER" I, N, AI;
    "INTEGER" "ARRAY" A[1 : 40];
    "PROCEDURE" AMESS(A);
    "INTEGER" "ARRAY" A; "FORTRAN" AMESS;

    N:= LENGTH(TEXT); I:= 0;
    "FOR" I:= I + 1 "WHILE" I <= N "DO"
    "BEGIN" STRINGELEMENT(TEXT, I,
        "("ABCDEFGHIJKLMNOPQRSTUVWXYZ")"
        "("0123456789+-*/()$= ,.")", AI);
        A[I]:= "IF" AI = 0 "THEN" 45 "ELSE" AI
    "END";
    "FOR" I:= I "STEP" 1 "UNTIL" 40 "DO" A[I]:= 45;
    AMESS(A)
"END" ALGMESS;
"EOP"
```

TITLE: **Exit**

BRIEF DESCRIPTION

A call of EXIT results in a jump to an imaginary label just before the last "END" of the program, and thus terminates the execution of the program.

CALLING SEQUENCE

Heading

```
"PROCEDURE" EXIT;
"CODE" 11010;
```

SOURCE TEXT

```
"CODE" 11010;
"PROCEDURE" EXIT;
STOP;
"EOP"
```

TITLE: **Available**

BRIEF DESCRIPTION

The procedure returns the number of unused CM-words, i.e. the space available for array storage.

CALLING SEQUENCE

Heading

```
"INTEGER" "PROCEDURE" AVAILABLE;
"CODE" 11011;
```

SOURCE TEXT

```
"CODE" 11011;
"INTEGER" "PROCEDURE" AVAILABLE;
AVAILABLE:= MAXIMUMFIELDLENGTH - PROGRAMSIZE - SCALARSTACK
          - HEAPSIZE - 100;
"EOP"
```

CWI SYLLABI

1 Vacantiecursus 1984: *Hewet - plus wiskunde.* 1984.

2 E.M. de Jager, H.G.J. Pijls (eds.). *Proceedings Seminar 1981-1982. Mathematical structures in field theories.* 1984.

3 W.C.M. Kallenberg, et al. *Testing statistical hypotheses: worked solutions.* 1984.

4 J.G. Verwer (ed.). *Colloquium topics in applied numerical analysis, volume 1.* 1984.

5 J.G. Verwer (ed.). *Colloquium topics in applied numerical analysis, volume 2.* 1984.

6 P.J.M. Bongaarts, J.N. Buur, E.A. de Kerf, R. Martini, H.G.J. Pijls, J.W. de Roever. *Proceedings Seminar 1982-1983. Mathematical structures in field theories.* 1985.

7 Vacantiecurus 1985: *Variatierekening.* 1985.

8 G.M. Tuynman. *Proceedings Seminar 1983-1985. Mathematical structures in field theories, Vol.1 Geometric quantization.* 1985.

9 J. van Leeuwen, J.K. Lenstra (eds.). *Parallel computers and computations.* 1985.

10 Vacantiecursus 1986: *Matrices.* 1986.

11 P.W.H. Lemmens. *Discrete wiskunde: tellen, grafen, spelen en codes.* 1986.

12 J. van de Lune. *An introduction to Tauberian theory: from Tauber to Wiener.* 1986.

13 G.M. Tuynman, M.J. Bergvelt, A.P.E. ten Kroode. *Proceedings Seminar 1983-1985. Mathematical structures in field theories, Vol.2.* 1987.

14 Vacantiecursus 1987: *De personal computer en de wiskunde op school.* 1987.

15 Vacantiecursus 1983: *Complexe getallen.* 1987.

16 P.J.M. Bongaarts, E.A. de Kerf, P.H.M. Kersten. *Proceedings Seminar 1984-1986. Mathematical structures in field theories, Vol.1.* 1988.

17 F. den Hollander, H. Maassen (eds.). *Mark Kac seminar on probability and physics. Syllabus 1985-1987.* 1988.

18 Vacantiecursus 1988. *Differentierekening.* 1988.

19 R. de Bruin, C.G. van der Laan, J.R. Luyten, H.F. Vogt. *Publiceren met LATEX.* 1988.

20 R. van der Horst, R.D. Gill (eds.). *STATAL: statistical procedures in Algol 60, part 1.* 1988.

21 R. van der Horst, R.D. Gill (eds.). *STATAL: statistical procedures in Algol 60, part 2.* 1988.

22 R. van der Horst, R.D. Gill (eds.). *STATAL: statistical procedures in Algol 60, part 3.* 1988.

MC SYLLABI

1.1 F. Göbel, J. van de Lune. *Leergang besliskunde, deel 1: wiskundige basiskennis.* 1965.

1.2 J. Hemelrijk, J. Kriens. *Leergang besliskunde, deel 2: kansberekening.* 1965.

1.3 J. Hemelrijk, J. Kriens. *Leergang besliskunde, deel 3: statistiek.* 1966.

1.4 G. de Leve, W. Molenaar. *Leergang besliskunde, deel 4: Markovketens en wachttijden.* 1966.

1.5 J. Kriens, G. de Leve. *Leergang besliskunde, deel 5: inleiding tot de mathematische besliskunde.* 1966.

1.6a B. Dorhout, J. Kriens. *Leergang besliskunde, deel 6a: wiskundige programmering 1.* 1968.

1.6b B. Dorhout, J. Kriens, J.Th. van Lieshout. *Leergang besliskunde, deel 6b: wiskundige programmering 2.* 1977.

1.7a G. de Leve. *Leergang besliskunde, deel 7a: dynamische programmering 1.* 1968.

1.7b G. de Leve, H.C. Tijms. *Leergang besliskunde, deel 7b: dynamische programmering 2.* 1970.

1.7c G. de Leve, H.C. Tijms. *Leergang besliskunde, deel 7c: dynamische programmering 3.* 1971.

1.8 J. Kriens, F. Göbel, W. Molenaar. *Leergang besliskunde, deel 8: minimaxmethode, netwerkplanning, simulatie.* 1968.

2.1 G.J.R. Förch, P.J. van der Houwen, R.P. van de Riet. *Colloquium stabiliteit van differentieschema's, deel 1.* 1967.

2.2 L. Dekker, T.J. Dekker, P.J. van der Houwen, M.N. Spijker. *Colloquium stabiliteit van differentieschema's, deel 2.* 1968.

3.1 H.A. Lauwerier. *Randwaardeproblemen, deel 1.* 1967.

3.2 H.A. Lauwerier. *Randwaardeproblemen, deel 2.* 1968.

3.3 H.A. Lauwerier. *Randwaardeproblemen, deel 3.* 1968.

4 H.A. Lauwerier. *Representaties van groepen.* 1968.

5 J.H. van Lint, J.J. Seidel, P.C. Baayen. *Colloquium discrete wiskunde.* 1968.

6 K.K. Koksma. *Cursus ALGOL 60.* 1969.

7.1 *Colloquium moderne rekenmachines, deel 1.* 1969.

7.2 *Colloquium moderne rekenmachines, deel 2.* 1969.

8 H. Bavinck, J. Grasman. *Relaxatietrillingen.* 1969.

9.1 T.M.T. Coolen, G.J.R. Förch, E.M. de Jager, H.G.J. Pijls. *Colloquium elliptische differentiaalvergelijkingen, deel 1.* 1970.

9.2 W.P. van den Brink, T.M.T. Coolen, B. Dijkhuis, P.P.N. de Groen, P.J. van der Houwen, E.M. de Jager, N.M. Temme, R.J. de Vogelaere. *Colloquium elliptische differentiaalvergelijkingen, deel 2.* 1970.

10 J. Fabius, W.R. van Zwet. *Grondbegrippen van de waarschijnlijkheidsrekening.* 1970.

11 H. Bart, M.A. Kaashoek. H.G.J. Pijls, W.J. de Schipper, J. de Vries. *Colloquium halfalgebra's en positieve operatoren.* 1971.

12 T.J. Dekker. *Numerieke algebra.* 1971.

13 F.E.J. Kruseman Aretz. *Programmeren voor rekenautomaten; de MC ALGOL 60 vertaler voor de EL X8.* 1971.

14 H. Bavinck, W. Gautschi, G.M. Willems. *Colloquium approximatietheorie.* 1971.

15.1 T.J. Dekker, P.W. Hemker, P.J. van der Houwen. *Colloquium stijve differentiaalvergelijkingen, deel 1.* 1972.

15.2 P.A. Beentjes, K. Dekker, H.C. Hemker, S.P.N. van Kampen, G.M. Willems. *Colloquium stijve differentiaalvergelijkingen, deel 2.* 1973.

15.3 P.A. Beentjes, K. Dekker, P.W. Hemker, M. van Veldhuizen. *Colloquium stijve differentiaalvergelijkingen, deel 3.* 1975.

16.1 L. Geurts. *Cursus programmeren, deel 1: de elementen van het programmeren.* 1973.

16.2 L. Geurts. *Cursus programmeren, deel 2: de programmeertaal ALGOL 60.* 1973.

17.1 P.S. Stobbe. *Lineaire algebra, deel 1.* 1973.

17.2 P.S. Stobbe. *Lineaire algebra, deel 2.* 1973.

17.3 N.M. Temme. *Lineaire algebra, deel 3.* 1976.

18 F. van der Blij, H. Freudenthal, J.J. de Iongh, J.J. Seidel, A. van Wijngaarden. *Een kwart eeuw wiskunde 1946-1971, syllabus van de vakantiecursus 1971.* 1973.

19 A. Hordijk, R. Potharst, J.Th. Runnenburg. *Optimaal stoppen van Markovketens.* 1973.

20 T.M.T. Coolen, P.W. Hemker, P.J. van der Houwen, E. Slagt. *ALGOL 60 procedures voor begin- en randwaardeproblemen.* 1976.

21 J.W. de Bakker (red.). *Colloquium programmacorrectheid.* 1975.

22 R. Helmers, J. Oosterhoff, F.H. Ruymgaart, M.C.A. van Zuylen. *Asymptotische methoden in de toetsingstheorie; toepassingen van naburigheid.* 1976.

23.1 J.W. de Roever (red.). *Colloquium onderwerpen uit de biomathematica, deel 1.* 1976.

23.2 J.W. de Roever (red.). *Colloquium onderwerpen uit de biomathematica, deel 2.* 1977.

24.1 P.J. van der Houwen. *Numerieke integratie van differentiaalvergelijkingen, deel 1: eenstapsmethoden.* 1974.

25 *Colloquium structuur van programmeertalen.* 1976.

26.1 N.M. Temme (ed.). *Nonlinear analysis, volume 1.* 1976.

26.2 N.M. Temme (ed.). *Nonlinear analysis, volume 2.* 1976.

27 M. Bakker, P.W. Hemker, P.J. van der Houwen, S.J. Polak, M. van Veldhuizen. *Colloquium discretiseringsmethoden.* 1976.

28 O. Diekmann, N.M. Temme (eds.). *Nonlinear diffusion problems.* 1976.

29.1 J.C.P. Bus (red.). *Colloquium numerieke programmatuur, deel 1A, deel 1B.* 1976.

29.2 H.J.J. te Riele (red.). *Colloquium numerieke programmatuur, deel 2.* 1977.

30 J. Heering, P. Klint (red.). *Colloquium programmeeromgevingen.* 1983.

31 J.H. van Lint (red.). *Inleiding in de coderingstheorie.* 1976.

32 L. Geurts (red.). *Colloquium bedrijfsystemen.* 1976.

33 P.J. van der Houwen. *Berekening van waterstanden in zeeën en rivieren.* 1977.

34 J. Hemelrijk. *Oriënterende cursus mathematische statistiek.* 1977.

35 P.J.W. ten Hagen (red.). *Colloquium computer graphics.* 1978.

36 J.M. Aarts, J. de Vries. *Colloquium topologische dynamische systemen.* 1977.

37 J.C. van Vliet (red.). *Colloquium capita datastructuren.* 1978.

38.1 T.H. Koornwinder (ed.). *Representations of locally compact groups with applications, part I.* 1979.

38.2 T.H. Koornwinder (ed.). *Representations of locally compact groups with applications, part II.* 1979.

39 O.J. Vrieze, G.L. Wanrooy. *Colloquium stochastische spelen.* 1978.

40 J. van Tiel. *Convexe analyse.* 1979.

41 H.J.J. te Riele (ed.). *Colloquium numerical treatment of integral equations.* 1979.

42 J.C. van Vliet (red.). *Colloquium capita implementatie van programmeertalen.* 1980.

43 A.M. Cohen, H.A. Wilbrink. *Eindige groepen (een inleidende cursus).* 1980.

44 J.G. Verwer (ed.). *Colloquium numerical solution of partial differential equations.* 1980.

45 P. Klint (red.). *Colloquium hogere programmeertalen en computerarchitectuur.* 1980.

46.1 P.M.G. Apers (red.). *Colloquium databankorganisatie, deel 1.* 1981.

46.2 P.G.M. Apers (red.). *Colloquium databankorganisatie, deel 2.* 1981.

47.1 P.W. Hemker (ed.). *NUMAL, numerical procedures in ALGOL 60: general information and indices.* 1981.

47.2 P.W. Hemker (ed.). *NUMAL, numerical procedures in ALGOL 60, vol. 1: elementary procedures; vol. 2: algebraic evaluations.* 1981.

47.3 P.W. Hemker (ed.). *NUMAL, numerical procedures in ALGOL 60, vol. 3A: linear algebra, part I.* 1981.

47.4 P.W. Hemker (ed.). *NUMAL, numerical procedures in ALGOL 60, vol. 3B: linear algebra, part II.* 1981.

47.5 P.W. Hemker (ed.). *NUMAL, numerical procedures in ALGOL 60, vol. 4: analytical evaluations; vol. 5A: analytical problems, part I.* 1981.

47.6 P.W. Hemker (ed.). *NUMAL, numerical procedures in ALGOL 60, vol. 5B: analytical problems, part II.* 1981.

47.7 P.W. Hemker (ed.). *NUMAL, numerical procedures in ALGOL 60, vol. 6: special functions and constants; vol. 7: interpolation and approximation.* 1981.

48.1 P.M.B. Vitányi, J. van Leeuwen, P. van Emde Boas (red.). *Colloquium complexiteit en algoritmen, deel 1.* 1982.

48.2 P.M.B. Vitányi, J. van Leeuwen, P. van Emde Boas (red.). *Colloquium complexiteit en algoritmen, deel 2.* 1982.

49 T.H. Koornwinder (ed.). *The structure of real semisimple Lie groups.* 1982.

50 H. Nijmeijer. *Inleiding systeemtheorie.* 1982.

51 P.J. Hoogendoorn (red.). *Cursus cryptografie.* 1983.