Return to R. Goldberg

Return to B. HEISING

The Description of Finite Sequential Processes

by

Kenneth E. Iverson

A paper presented at the

4th London Conference on Information Theory

August, 1960

The economical execution of fully-specified algorithms provided by the automatic computer has greatly increased the use of complex algorithms in information theory and other fields. This increased use has in turn generated a need for consistent and powerful programming languages for the description and analysis of complex algorithms. A programming language is commonly character-

The economical execution of fully-specified algorithms provided by the automatic computer has greatly increased the use of complex algorithms in information theory and other fields. This increased use has in turn generated a need for consistent and powerful programming languages for the description and analysis of complex algorithms. A programming language is commonly character-

ized as problem-oriented or machine-oriented, according as it is intended mainly for the description and analysis of algorithms or for their execution. The language outlined* in the present paper was developed primarily for description and analysis, but also lends itself well to execution. The present emphasis is on description and analysis.

what about procedure oriented

A programming language should (1) allow a clear and simple representation of the sequence in which steps of an algorithm are ? performed, (2) provide a concise and consistent notation for the operations occurring in a wide range of processes, (3) permit the description of a process to be independent of the choice of a particular representation for the data, (4) allow economy in operation symbols, and (5) provide convenient subordination of detail without loss of detail.

what about case of using

The sequence of execution of statements will be specified by their order of listing and by arrows connecting a statement to its successor. Branch points, at which alternative successors are chosen according to the outcome of a comparison between a pair of quantities, will be represented by a colon placed between the compared quantities, and by a label attached to each arrow showing the relation under which it is followed. Any well-defined relation may be employed, e.g., equality, inequality, or set membership. The conditions at each branch point must be exhaustive and

pared quantities, and by a label attached to each arrow showing the relation under which it is followed. Any well-defined relation may be employed, e.g., equality, inequality, or set membership. The conditions at each branch point must be exhaustive, and the listed successor is associated with all conditions not included in the labeled arrows.

Commonly occurring operations to be defined include the floor $\lfloor x \rfloor$ (largest integer not exceeding x), the ceiling $\lceil x \rceil$, and the residue of x modulo m, to be denoted by $|x,m|$. The common logical operations and, or, and not will be denoted by $\wedge$, $\vee$, and $\bar{\ }$, and will be augmented by the <u>relational statement</u> $(x \mathcal{R} y)$ defined as follows. If x and y are any quantities and $\mathcal{R}$ is any binary relation defined upon them, then $(x \mathcal{R} y)$ is a logical variable whose value is 0 or 1 according as x does or does not stand in the relation $\mathcal{R}$ to y. For example, the absolute value of x may be defined as follows:

$$|x| = x(1-2(x < 0)).$$

To illustrate the use of the floor and ceiling operations, consider a rectangular array of dimension a $\times$ b whose cells listed in order by rows $(0,1,\ldots,b-1,b,b+1,\ldots,ab-1)$ are denoted by x and in order by column are denoted by y. Then

$$x = b \times |y,a| + \lfloor y \div a \rfloor ,$$

and

$$y = a \times |y,b| + \lfloor x \div b \rfloor .$$

by rows $(0,1,\ldots,b-1,b,b+1,\ldots,ab-1)$ are denoted by x and in order by column are denoted by y. Then

$$x = b \times |y,a| + \lfloor y \div a \rfloor ,$$

and

$$y = a \times |y,b| + \lfloor x \div b \rfloor .$$

These are the transformations used in determining accessibility in a serial-parallel memory of a bands with b slots per band. They may be derived from the identity

$$y = a \times \lfloor y \div a \rfloor + |y,a| .$$

The description of a process can be made independent of its representation by defining certain fundamental operations upon finite ordered sets. The element of a finite simply-ordered set B of <u>dimension</u> (number of elements) $\nu(B)$ can be indexed by the integers $1,2,\ldots,\nu(B)$ such that B_i is the i^{th} element of the set. The <u>k^{th} successor</u> of an element x of B will then be denoted by $x \underset{B}{\uparrow} k$ and defined as the element B_j, where $j \equiv (i+k)(\text{mod } \nu(B))$, and $B_i = x$. The <u>k^{th} predecessor</u> is defined analogously and denoted by $x \underset{B}{\downarrow} k$. If B is the set of integers, the symbol B may be elided, and if $k = 1$ it may be elided - hence $i \uparrow k$ denotes the k^{th} successor of the integer i, and $i\uparrow$ denotes the integer $i + 1$.

The successor operation defined upon an element of B can be extended to any subset C of B as follows: $C \underset{B}{\uparrow} k$ denotes the set D such that $D_i = C_i \underset{B}{\uparrow} k$. If C and B are identical, the operation is called <u>left rotation</u> and is denoted by $C \uparrow k$. Rotation is extended analogously to vectors.

Two sets A and B are <u>equal</u> ($A = B$) if they contain the same elements, but are identical ($A \equiv B$) only if they also have the same

D such that $D_i = C_i \underset{B}{\uparrow} k$. If C and B are identical, the operation is called <u>left rotation</u> and is denoted by $C \uparrow k$. Rotation is extended analogously to vectors.

Two sets A and B are <u>equal</u> ($A = B$) if they contain the same elements, but are identical ($A \equiv B$) only if they also have the same

order. Simple modifications in the standard definitions of intersection and union provide a closed system for ordered sets. To achieve economy of operation symbols, intersection and union are denoted by $\wedge$ and $\vee$, already used for the analogous logical operations <u>and</u>, and <u>or</u>. Potential ambiguity is avoided by using distinctive symbols for each class of operand; italics for single variables, lower case boldface italics for vectors, upper case boldface italics for matrices, and ordinary Roman characters for literals, i.e., for

implied dimension limit of $\leq$

A second set of indices, called <u>contracurrent indices</u> will be assigned to the elements of any set A. These indices run from $-\nu(A)$ to -1. The k^{th} element may therefore be denoted alternatively by A_k or A_{-j}, where $j+k = \nu(A)$. In particular, the terminal elements may be denoted by A_1 and A_{-1}. Contracurrent indices will also be employed for vectors and matrices.*

A set obtained from a set B by deleting the first i and the last j elements is called a <u>solid subset</u> or an <u>infix</u> of B. An infix C of B is also called a <u>prefix</u> of B if $C_1 = B_1$, or a <u>suffix</u> if $C_{-1} = B_{-1}$. The statement

$$C \xleftarrow[B]{} \{(x,y)\}$$

specifies C as the infix of B having terminal elements x and y. The symbol B may be elided if B is the set of integers.

Program 1 illustrates the conventions introduced thus far. If H^1_j is the suit and H^2_j the denomination of the j^{th} card in a

$$C \xleftarrow[B]{} \{(x,y)\}$$

specifies C as the infix of B having terminal elements x and y. The symbol B may be elided if B is the set of integers.

Program 1 illustrates the conventions introduced thus far. If H^1_j is the suit and H^2_j the denomination of the j^{th} card in a
hand of thirteen playing cards, and if

$$D \equiv \{\text{deuce, trey, ..., king, ace}\}$$

is the set of denominations, then the quantity q determined by the program is the length of the longest run in any one suit. A left-

---

*In certain work, notably in switching theory and in the use of positional representations (e.g., column sorting) there is some ad-

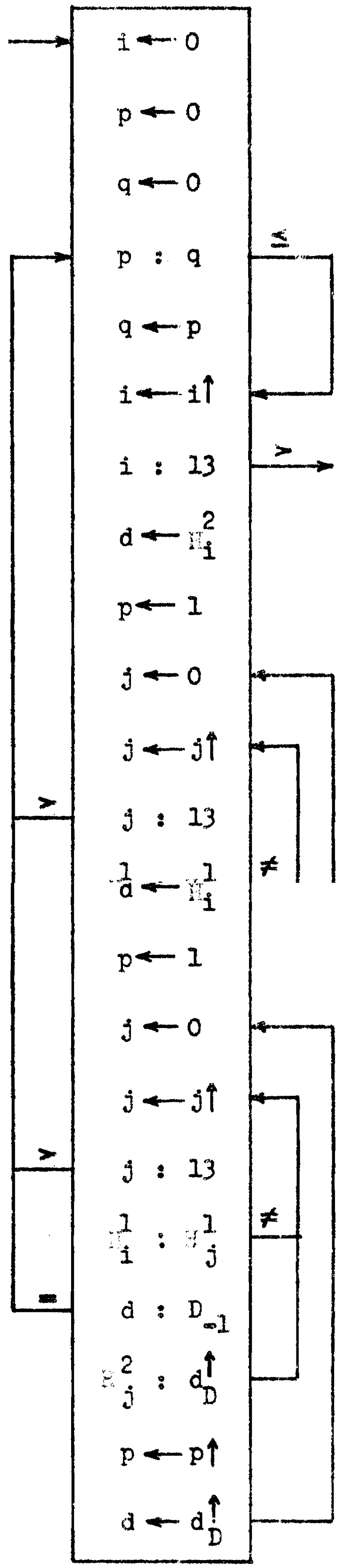

i ← 0
p ← 0
q ← 0
p : q
q ← p
i ← i↑
i : 13
d ← M$_i^2$
p ← 1
j ← 0
j ← j↑
j : 13
d ← M$_i^1$
p ← 1
j ← 0
j ← j↑
j : 13
M$_i^1$: M$_j^1$
d : D$_{-1}$
M$_j^2$: d$_D$↑
p ← p↑
d ← d$_D$↑
≤
>
≠
=

Diagram 1

pointing arrow associates the <u>specifying</u> quantity on the right of each statement with the <u>specified</u> quantity on the left. The arrow is used instead of the sign of equality because it eliminates ambiguity and reserves the sign of equality as a <u>relation</u> to be used in relational statements only.

Significant subordination of detail can be achieved by generalizing each operation defined upon simple variables to structured arrays such as vectors and matrices. For example, if p and q are logical vectors (i.e., each component is a logical variable), then

$$\left.\begin{array}{ll} r \leftarrow \overline{p} \Longleftrightarrow r_i = \overline{p}_i , \\ r \leftarrow p \wedge q \Longleftrightarrow r_i = p_i \wedge q_i , \\ \text{and} & \\ r \leftarrow p \vee q \Longleftrightarrow r_i = p_i \vee q_i , \end{array}\right\} \quad i \in \{(1, \nu(r))\} .$$

$$\left.\begin{array}{ll} r \leftarrow \overline{p} \Longleftrightarrow r_i = \overline{p}_i , \\ r \leftarrow p \wedge q \Longleftrightarrow r_i = p_i \wedge q_i , \\ \text{and} & \\ r \leftarrow p \vee q \Longleftrightarrow r_i = p_i \vee q_i , \end{array}\right\} \quad i \in \{(1, \nu(r))\} .$$

Moreover, if x and y are numerical vectors, then

$$z \leftarrow x + y \Longleftrightarrow z_i = x_i + y_i$$

$$z \leftarrow x \times y \Longleftrightarrow z_i = x_i \times y_i$$

$$z \leftarrow x \div y \Longleftrightarrow z_i = x_i \div y_i .$$

Operations such as floor, ceiling, and residue are extended

The application of any associative binary operation $\odot$ to all components of a vector x is denoted by $\odot/x$. Thus $\times/x$ is the product and $+/x$ is the sum of all components of x. The latter will also be denoted by $\sigma(x)$ and be called the <u>weight</u> of x. The usual notations for matrix algebra are retained, i.e., xy for a scalar product, and XY for a product of matrices. It is clear that $xy = \sigma(x \times y)$ and that $\sigma(x \uparrow k) = \sigma(x)$ for all k.

Greek symbols (in the appropriate type face) will be used for specially defined quantities. Thus, the <u>unit vectors</u> ϵ^i are logical vectors such that $(\epsilon^i)_j = 1$ if and only if $j = i$. The <u>full vector</u> ϵ is the negation of the zero vector. The <u>prefix vector</u> α^j is a logical vector of weight j whose first j components are unity. The <u>suffix vector</u> ω^j is defined analogously. The <u>identity permutation vector</u> ι is defined by the relation $\iota_j = j$. The dimension of a unit, suffix, prefix, or identity permutation vector is normally defined implicity by the compatability requirements

logical vectors such that $(\epsilon^i)_j = 1$ if and only if $j = i$. The <u>full vector</u> ϵ is the negation of the zero vector. The <u>prefix vector</u> α^j is a logical vector of weight j whose first j components are unity. The <u>suffix vector</u> ω^j is defined analogously. The <u>identity permutation vector</u> ι is defined by the relation $\iota_j = j$. The dimension of a unit, suffix, prefix, or identity permutation vector is normally defined implicity by the compatability requirements

of associated operators and operands. The scalar zero, vector zero, and matrix zero will all be denoted by 0.

The conventional vector product of two space vectors (to be denoted by $x \times y$) will illustrate the use of the foregoing notation. It can be defined as

$$x \times y = x\uparrow \times y\downarrow - x\downarrow \times y\uparrow .$$

A trivial formal manipulation shows that $y \times x = -(x \times y)$. The orthogonality theorem $x(x \times y) = 0$ can be established as follows:

$$x(x \times y) = x(x\uparrow \times y\downarrow - x\downarrow \times y\uparrow) = \sigma(x \times x\uparrow \times y\downarrow - x \times x\downarrow \times y\uparrow)$$

$$= \sigma((x \times x\uparrow \times y\downarrow)\downarrow - x \times x\downarrow \times y\uparrow)$$

$$= \sigma(x\downarrow \times x \times y\downarrow 2 - x \times x\downarrow \times y\uparrow) \quad .$$

Since the $\times$ operator is commutative, and since $y\downarrow 2 = y\uparrow$ for a vector of dimension three, the final expression is equal to zero and the theorem is established. Further theorems concerning the magnitude of $x \times y$, the four-vector product, and the box product follow by similarly simple formal manipulation.

Individual components of structured operands can be selected by subscripts and superscripts — x_i for the i^{th} component of the vector x, X^i for the i^{th} row vector of a matrix X, X_j for

and the theorem is established. Further theorems concerning the magnitude of $x \times y$, the four-vector product, and the box product follow by similarly simple formal manipulation.

Individual components of structured operands can be selected by subscripts and superscripts — x_i for the i^{th} component of the vector x, X^i for the i^{th} row vector of a matrix X, X_j for

the j^{th} column vector, and X^i_j for the ij^{th} element. More generally, it is necessary to specify selected subsets of the components. Since the selection is, for each component, a binary operation, it can be specified by an associated logical vector of the same dimension. Thus, for an arbitrary vector x and compatible logical vector u (that is, $\nu(u) = \nu(x)$), the statement implies that the z is obtained by suppressing from x those components x_i for which $u_i = 0$. The operation u/x is called <u>compression of x by u</u>. For example, if $u = (1,0,0,0,1,1)$, and $x = (\text{m}, \text{o}, \text{n}, \text{d}, \text{a}, \text{y})$, then $u/x = (\text{m}, \text{a}, \text{y})$. Clearly, $\nu(u/x) = \sigma(u)$. Set compression is defined analogously.

Two types of compression must be defined for matrices; <u>row compression</u>, defined by

$$Z \longleftarrow u/X \Longleftrightarrow Z^i = u/X^i \quad , \qquad i \;\varepsilon\; \{(1, \mu(X))\} \; ,$$

and <u>column compression</u>, defined by

$$Z \longleftarrow u/\!/X \Longleftrightarrow Z_j = u/X_j \quad , \qquad j \;\varepsilon\; \{(1, \nu(X))\} \; .$$

For example, if the matrix M represents a ledger of $\mu(M)$ bank accounts, with the column vectors M_1, M_2, M_3, and M_4 denoting name, account number, address, and balance, respectively, then the
and <u>column compression</u>, defined by

$$Z \longleftarrow u/\!/X \Longleftrightarrow Z_j = u/X_j \quad , \qquad j \;\varepsilon\; \{(1, \nu(X))\} \; .$$

For example, if the matrix M represents a ledger of $\mu(M)$ bank accounts, with the column vectors M_1, M_2, M_3, and M_4 denoting name, account number, address, and balance, respectively, then the operation of preparing a list P of the name, account number, and balance, for all accounts whose balance exceeds 1000 can be completely prescribed as follows.

$$P \longleftarrow (M_4 > 1000\,\varepsilon) /\!/ (\bar{\varepsilon}^3/M) \quad .$$

The expansion of a vector x by a logical vector u is denoted by $u\backslash x$ and is defined as follows.

$$z \leftarrow u\backslash x \iff u/z = x \text{ and } \bar{u}/z = 0.$$

It is necessary that $\sigma(u) = \nu(x)$. Clearly $\nu(z) = \nu(u)$. Row expansion (denoted by $u\backslash X$) and column expansion ($u\backslash\!\backslash X$) are defined analogously.

The compress and expand operations provide a powerful extension of ordinary matrix algebra. For example, any numerical vector can be decomposed according to the identity

$$x = \bar{u}\backslash\bar{u}/x + u\backslash u/x \quad .$$

Matrices can be decomposed similarly. Moreover, the conventional operations on paritioned matrices can be generalized in a systematic manner. A few of the more important identities are, for example:

Matrices can be decomposed similarly. Moreover, the conventional operations on paritioned matrices can be generalized in a systematic manner. A few of the more important identities are, for example:

$$(u/X)Y = X(u\backslash\!\backslash Y) \quad ,$$

$$(u\backslash X)Y = X(u/\!/Y) \quad ,$$

$$XY = (u/X)(u/\!/Y) + (\bar{u}/X)(\bar{u}/\!/Y) \quad ,$$

$$u/(XY) = X(u/Y) \quad ,$$

$$u/\!/(XY) = (u/\!/X)Y \quad ,$$

$$(u/v)/(u/X) = (u \wedge v)/X \quad .$$

Maximization over those components of x for which $u_j = 1$ will be denoted by $u\lceil x$. More precisely,

$$v \leftarrow u\lceil x$$

specifies a logical vector v such that $v/x = m\epsilon$ and that $(\bar{v}/x)_j < m$ for $j \epsilon \left\{(1, \nu(\bar{v}))\right\}$. Graphically, v is obtained by lowering a horizontal line over a plot of x until it touches the largest component, and then marking with a 1 all components of x touched by the line. Thus if $x = (6,3,-8,6,6)$, then $v = \epsilon\lceil x = (1,0,0,1,1)$, $v/x = (6,6,6)$, and $(v/x)_1 = 6$. Minimization is denoted analogously by $u\lfloor x$. The minimum over all positive values of x may be denoted, for example, by $(x > 0)\lfloor x$, and for the present example $(x > 0)\lfloor x = (0,1,0,0,0)$.

Program 2 illustrates the use of this notation in a complete description of the Simplex algorithm for linear programming.

$v/x = (6,6,6)$, and $(v/x)_1 = 6$. Minimization is denoted analogously by $u\lfloor x$. The minimum over all positive values of x may be denoted, for example, by $(x > 0)\lfloor x$, and for the present example $(x > 0)\lfloor x = (0,1,0,0,0)$.

Program 2 illustrates the use of this notation in a complete description of the Simplex algorithm for linear programming.
The vector y determined is the optimal solution of the following system; maximize cy subject to the constraints $(Ay \leq b) = \epsilon$, and $(y \geq 0) = \epsilon$. The logical vector u is assumed to be given initially and specifies the current feasible basis; x is the corresponding vector of non-zero variables. A power of a matrix is denoted by a superscript enclosed in square brackets.

[-1]

D ← (u/A)

x ← Db

y ← u\x

P ← DA

g ← c-(u/c)P

v ← s⌈g

j ← (v/ι)$_1$

g_j : 0 → = → y is optimal solution

P ← x ÷ P_j

(> 0) : 0 → = → solution unbounded

← (> 0)|

j ← (v/ι)$_1$

g_j : 0 → = → y is optimal solution

P ← x ÷ P_j

(> 0) : 0 → = → solution unbounded

← (> 0)|

( ) : 1 → ≠ → degenerate

← \ j

The operations occurring in Program 2 can be used in a formal analysis of its behavior. For example,

$$\mathbf{u}/\mathbf{P} = \mathbf{u}/((\mathbf{u}/\mathbf{A})^{[-1]}\mathbf{A}) = (\mathbf{u}/\mathbf{A})^{[-1]}(\mathbf{u}/\mathbf{A}) = \mathbf{I}$$

and $\mathbf{P}$ therefore contains an identity matrix $\mathbf{I}$ in the columns corresponding to the feasible basis $\mathbf{u}$. Moreover, since

$$\mathbf{g} = \mathbf{c} - (\mathbf{u}/\mathbf{c})\mathbf{P} \quad ,$$

then

$$\mathbf{u}/\mathbf{g} = \mathbf{u}/\mathbf{c} - \mathbf{u}/\ (\mathbf{u}/\mathbf{c})\mathbf{P} = \mathbf{u}/\mathbf{c} - (\mathbf{u}/\mathbf{c})(\mathbf{u}/\mathbf{P}) = \mathbf{u}/\mathbf{c} - (\mathbf{u}/\mathbf{c})\ \mathbf{I} = 0.$$

Hence the components of the modified cost function $\mathbf{g}$ are zero for all included variables, as desired.

The <u>base $\mathbf{b}$ value of the vector $\mathbf{x}$</u> is denoted by $\mathbf{b}\perp\mathbf{x}$ and defined as the value of $\mathbf{x}$ in the mixed base number system defined

$$\mathbf{u}/\mathbf{g} = \mathbf{u}/\mathbf{c} - \mathbf{u}/\ (\mathbf{u}/\mathbf{c})\mathbf{P} = \mathbf{u}/\mathbf{c} - (\mathbf{u}/\mathbf{c})(\mathbf{u}/\mathbf{P}) = \mathbf{u}/\mathbf{c} - (\mathbf{u}/\mathbf{c})\ \mathbf{I} = 0.$$

Hence the components of the modified cost function $\mathbf{g}$ are zero for all included variables, as desired.

The <u>base $\mathbf{b}$ value of the vector $\mathbf{x}$</u> is denoted by $\mathbf{b}\perp\mathbf{x}$ and defined as the value of $\mathbf{x}$ in the mixed base number system defined
by the radices $b_1, b_2, \ldots, b_{-1}$. More precisely, $\mathbf{b}\perp\mathbf{x} = \mathbf{w}\mathbf{x}$, where $w_{-1} = 1$ and $w_{-i} = \times/(\omega^{i-1}/\mathbf{b})$, for $i \in \{(2, \nu(\mathbf{b}))\}$. If, for example, $\mathbf{b} = (7,24,60,60)$, and $\mathbf{x}$ denotes elapsed time in days, hours, minutes, and seconds, then $\mathbf{b}\perp\mathbf{x}$ denotes the elapsed time in seconds. In particular, $10\epsilon\perp\mathbf{x}$ denotes the value of $\mathbf{x}$ in the decimal system, and $y\epsilon\perp\mathbf{x}$ denotes the polynomial in y whose coefficients are the components of $\mathbf{x}$.

such that $\nu(U) = n$, $\mu(U) = \lceil \log_2(n+1) \rceil$, and $2 \perp U_j = j$. The i^{th} parity check group then includes the components of the vector U^i/x and Program 3 describes the determination of the corrected value y of the code x. An even-parity code is assumed, i.e., legitimate code points satisfy even-parity for all check groups.

For non-numeric vectors, the expand and compress operations do not suffice. The <u>mesh of x and y on u</u> is defined as follows:

$$z \leftarrow \backslash x, u, y \backslash \Leftrightarrow \bar{u}/z = x \text{ and } u/z = y \quad .$$

Clearly, $\nu(z) = \nu(u)$, $\nu(x) = \sigma(\bar{u})$, and $\nu(y) = \sigma(u)$. If, for example, $x = (m, a, y)$, $y = (o, n, d)$, and $u = (0,1,1,1,0,0)$, then

$$\backslash x, u, y \backslash = (m, o, n, d, a, y).$$

The <u>mask of x and y on u</u> is defined as follows:

Clearly, $\nu(z) = \nu(u)$, $\nu(x) = \sigma(\bar{u})$, and $\nu(y) = \sigma(u)$. If, for example, $x = (m, a, y)$, $y = (o, n, d)$, and $u = (0,1,1,1,0,0)$, then

$$\backslash x, u, y \backslash = (m, o, n, d, a, y).$$

The <u>mask of x and y on u</u> is defined as follows:

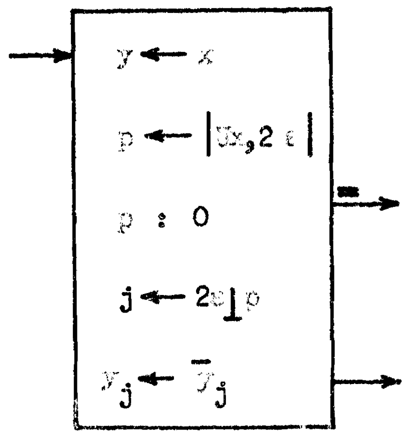
$$z \leftarrow /x, u, y/ \Leftrightarrow \bar{u}/z = \bar{u}/x \text{ and } u/z = u/y \quad .$$

Clearly,

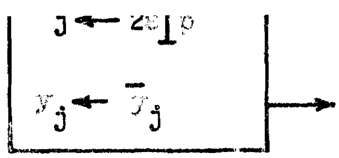
$$\backslash x, u, y \backslash = /\bar{u} \backslash x, u, u \backslash y/ \quad ,$$

$$/x, u, y/ = \backslash \bar{u}/x, u, u/y \backslash \quad ,$$

and



Program 3


Program 3

Analogous column mask, row mask, column mesh, and row mesh operations are defined upon matrices.

If two sets A and B are equal (but not necessarily identical), one is said to be a permutation of the other, and there exists a vector p such that

$$B_i = A_{p_i} .$$

Moreover, the components of p are some permutation of the integers $1, 2, \ldots, \nu(B)$, and p is called a permutation vector. If $B_i = A_{p_i}$ for some permutation vector, then B may be denoted by A_p.

Permutation will be extended analogously to vectors and matrices. For example, X^p_p denotes an elementary similarity transformation on the square matrix X. It is easily shown that

$$p_q = \iota \Longleftrightarrow q_p = \iota$$

some permutation vector, then B may be denoted by A_p.

Permutation will be extended analogously to vectors and matrices. For example, X^p_p denotes an elementary similarity transformation on the square matrix X. It is easily shown that

$$p_q = \iota \Longleftrightarrow q_p = \iota$$

and the permutations p and q are then said to be inverse. Clearly $(x_p)_q = x$ for any pair of inverse vectors p and q.

Any biunique mapping from an element b of an arbitrary set B to a correspondent a of an arbitrary set A can be represented by a permutation vector p such that B_i maps into A_{p_i}. If, for example,

$$A \equiv \{\text{apple, booty, dust, eye, night}\} ,$$

$$B \equiv \{\text{Apfel, Auge, Beute, Nacht, Staub}\} ,$$

Program 4 describes a mapping from the argument $b \varepsilon B$ to the function $a \varepsilon A$ prescribed by the vector p. The process consists of three steps, the ranking of b in B, the permutation of the index j by p, and the selection of the correspondent A_{p_j}. Since any set can be considered as a vector, the process can be expressed more concisely in terms of vector operations as shown in Program 5. The expression $\iota(B)$ denotes the <u>identification vector</u> of the set B, defined as the set considered as a vector, i.e.,

$$b \leftarrow \iota(B) \Longleftrightarrow b_i = B_i \quad .$$

A matrix whose rows and columns are all permutation vectors will be called a <u>permutation matrix</u>.* A permutation matrix can clearly represent the operations in an abstract group. The group is Abelian if and only if the matrix is symmetric.

A vector is frequently represented (stored) in a <u>serial-access file</u> in which the components are made available only in

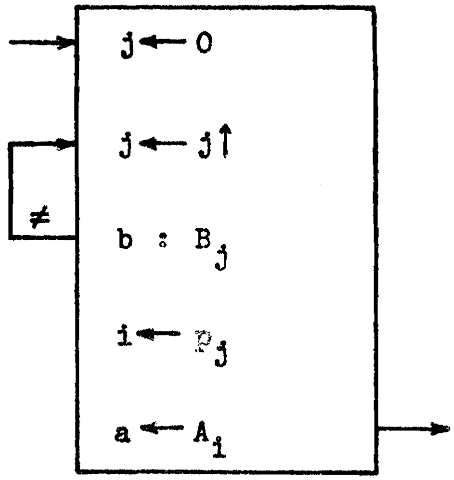
will be called a <u>permutation matrix</u>.* A permutation matrix can clearly represent the operations in an abstract group. The group is Abelian if and only if the matrix is symmetric.

A vector is frequently represented (stored) in a <u>serial-access file</u> in which the components are made available only in
their natural sequence. To describe algorithms upon vectors so represented, it is convenient to introduce speical notation for a file as follows. A <u>file</u> þ <u>of length n</u> is a representation of a vector x of dimension n arranged as follows:

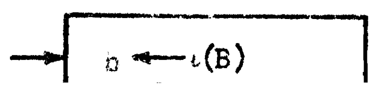
$$\lambda(1), x_1, \lambda(2), x_2, \ldots, \lambda(n), x_n, \lambda(n+1) \quad .$$

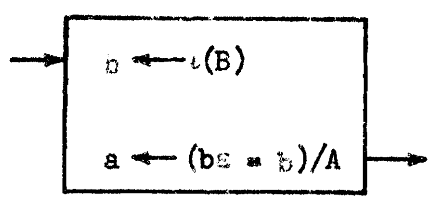
---

*This is a departure from conventional usage in which a permutation matrix is a logical matrix whose application corresponds to the application of a permutation vector.



Program 4





Program 5

The operation of transferring a component from a file to specify a quantity y is called <u>reading</u> the file and is denoted by $y \leftarrow \Phi$. The transfer is terminated by the occurrence of a partition symbol, and if this symbol is $\lambda(j)$ the file is then said to be <u>in position j</u>. A file may either be read <u>forward</u> (denoted by ${}_0\Phi$) or <u>backward</u> (denoted by ${}_1\Phi$). If a file originally in position j is read forward it transfers the component x_j and stops in position $(j+1)$, $j \varepsilon \{(1,n)\}$. A file read backward from position $j+1$ transfers the component x_j and stops in position j, $j \varepsilon \{(1,n)\}$.

The position of a file Φ will be denoted by $\pi(\Phi)$. Thus the statement $y \leftarrow \pi(\Phi)$ specifies y as the position of Φ, whereas $\pi(\Phi) \leftarrow z$ <u>positions</u> the file to z. In particular, $\pi(\Phi) \leftarrow 1$ denotes the <u>rewinding</u> of the file, and either $\pi(\Phi) \leftarrow (n+1)$ or (using contra-current indexing on the $(n+1)$ positions) $\pi(\Phi) \leftarrow -1$ denote positioning to the end of the file. Any file for which the general positioning operation $\pi(\Phi) \leftarrow z$ is to be avoided as impossible or inefficient
$\pi(\Phi) \leftarrow z$ <u>positions</u> the file to z. In particular, $\pi(\Phi) \leftarrow 1$ denotes the <u>rewinding</u> of the file, and either $\pi(\Phi) \leftarrow (n+1)$ or (using contra-current indexing on the $(n+1)$ positions) $\pi(\Phi) \leftarrow -1$ denote positioning to the end of the file. Any file for which the general positioning operation $\pi(\Phi) \leftarrow z$ is to be avoided as impossible or inefficient
is called a <u>serial</u> or <u>serial-access</u> file.

A file may be produced by a sequence of <u>recording</u> statements, either forward:

$$ {}_0\Phi \leftarrow x_i, \qquad i = 1,2,\ldots,n \quad , $$

or backward

$$ {}_1\Phi \leftarrow x_i, \qquad i = n,n-1,\ldots,1 \; . $$

As in reading, each forward (backward) record operation increments (decrements) the position of the file by one. A file which is only recorded during a process is called an <u>output file</u> of the process; a file which is only read is called an <u>input file</u>.

Each partition symbol may assume one of several values, $\lambda_0,\lambda_1,\ldots,\lambda_p$, the partitions with larger indices demarking larger subgroups within the file. Thus if each component x_j were itself a vector y^j(i.e., x is a matrix), then the last component of each y^j might be followed by the partition λ_1, while the remaining components would each be followed by λ_0. The last component of the entire array might be followed by a partition λ_2. In recording an item, the associated partition is indicated by listing it after the item (e.g., $\phi \longleftarrow y,\lambda_2$), except that the partition λ_0 is usually elided. The indicated partition then follows or precedes the associated item in the file according as the recording is forward or backward.

entire array might be followed by a partition λ_2. In recording an item, the associated partition is indicated by listing it after the item (e.g., $\phi \longleftarrow y,\lambda_2$), except that the partition λ_0 is usually elided. The indicated partition then follows or precedes the associated item in the file according as the recording is forward or backward.

The indication provided by the distinct parition symbols is used to control an immediate (p+1)-way branch in the program following each read operation. The branch is determined by the partition symbol which terminates the read.

Different files occurring in a process will be distinguished by righthand subscripts and superscripts, the latter being generally reserved to denote major classes of files (e.g., input and output).

File notation is particularly useful in the description of sorting algorithms and of algorithms employing so-called "push-down stores."