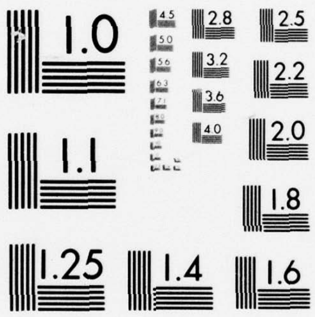
AD-A055 743 RENSSELAER POLYTECHNIC INST TROY N Y COMPUTER RESEAR--ETC F/G 9/2
CHAP USER'S MANUAL.(U)
MAY 78 K P LOEPERE AFOSR-76-2937
UNCLASSIFIED CRL-56 AFOSR-TR-78-1038 NL

1 OF 1
AD
A055743

END
DATE
FILMED
8 -78
DDC


1.0
4.5
5.0
5.6
6.3
7.1
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

2

RENSSELAER POLYTECHNIC INSTITUTE
KNOWLEDGE AND THOROUGHNESS
1824

Rensselaer Polytechnic Institute

Troy, New York 12181

18 AFOSR 19 TR- 78 - 1038

2

AD A055743

Technical Report CRL-56

14 CRL-56

6 # CHAP USER'S MANUAL.

By

10 Keith P./Loepere

11 May 1978

12 59p.

AD No.
DDC FILE COPY

16 2304

17 A2

DDC
RECEIVED
JUN 21 1978
E

9 Interim rept.,

Prepared for

Directorate of Mathematical and Information Sciences
Air Force Office of Scientific Research
Air Force Systems Command, USAF
Grant 15 ✓AFOSR-76-2937

78 06 19 088

Rensselaer Polytechnic Institute

TROY, NEW YORK 12181

Approved for public release;
distribution unlimited.

409 952 [illegible]

ABSTRACT

A computer representation for line drawings that has been found particularly convenient and has become widely accepted is the chain code. This representation is compact and allows most common processing functions to be performed efficiently.

CHAP is a collection of FORTRAN routines designed to process chain-encoded line drawings. Routines exist in CHAP to manipulate, synthesize, analyze, and do input and output upon chains. This report is intended to serve as a user's manual for CHAP. It describes the use of the CHAP routines along with other information needed to write a program utilizing the routines. A programming example is included.

ACKNOWLEDGMENT

The work described here was supported in part by the Directorate of Mathematical and Information Sciences, Air Force Office of Scientific Research, under Grant AFOSR 76-2937, Professor Herbert Freeman Principal Investigator.

CONTENTS

PART 1

INTRODUCTION

One of the ways devised to represent line drawings in a form computers can manipulate is the chain code. CHAP is a set or routines designed to facilitate the manipulation of chains. These routines are written mostly in FORTRAN and are FORTRAN callable.

It is assumed that the reader understands all of the properties of chains. One set of properties that is assumed by CHAP that may not be standard is that a chain starts off to be visible and to have a grey level, elevation, and color of zero. These properties occur as if signal codes to their effect actually occurred. It is convenient to view it this way even though the signal codes do not actually exist at the beginning of the chain. Signal codes may be added to the beginning of a chain, if desired, to change these default attributes.

The origin of a chain is assumed to be (0, 0) if no specification is made to the contrary. When x or y coordinate specifiers are present, the field value in them is 16384 greater than the actual value. That is, the five digit field in the code has a value of 16384 (40000_8) for a value of zero. It is possible to have x or y coordinate values in the range -16384 to 16383.

Sections included herein describe the use of the CHAP routines. CHAP routines exist to take apart, synthesize, analyze, and do input and output upon chains. A few common blocks must be defined at the beginning of the mainline user program. The section discussing this should be noted.

PART 2

CHAINS IN CHAP

Chains are stored in arrays. A chain array has the following format:

word 1 : the maximum number of links this chain array can hold

words 2, 3, and 4 : initial value of zero

words 5 and beyond : the chain links

The part of the array used to store chain links must be large enough to store the links packed as indicated by word one. The number of words it will take to store a given number of links depends on the implementation; see the appropriate section. For a machine that can hold 10 links per word, a 10-word array can hold (10 - 4) * 10 = 60 links.

Chains are passed to a routine by specifying just its name. Remember in writing user routines that this is an integer array being passed.

A single dimension array of chains would consist of a two-dimensional array where the number of columns is the number of chains and the number of rows is the word length of each chain. (This is true for FORTRAN user's programs.) Each chain in the array must have words 1, 2, 3, and 4 set as above. Addressing chain 3, for example, for a routine would be done by specifying chain (1, 3) as the argument (start of column 3). Some implementations may not correctly pass the address of the column to be used as a scalar chain; see the appropriate section. These notions can be extended to higher dimensioned arrays of chains.

PART 3

MANIPULATING THE CHAIN DATA STRUCTURE

A. Chain Synthesis

Chain synthesis routines are used to build up a chain element by element. These routines can generate links or signal codes.

A common first operation is to use subroutine CLEAR (chain) to set chain to be a null chain. A null chain has no links or signal codes in it. It does, however, have the default attributes described before. A chain may not be cleared if it is open (discussed later).

Three routines exist to add links to the end of a chain. Subroutine LINK (chain, link, number) adds the link numbered link a number number of times to the end of the specified chain. This call is ignored if the number specified is less than or equal to zero. The way in which the link is added depends on number; signal codes are generated whenever that would be the most efficient means of storing the links.

A set of links can be added by calling LINKSQ (chain, set, dimension of set, number). Set is a one-dimensional array of links to be added to the end of the specified chain. Number is the number of times this link sequence is to be added. A group repeat code is generated by this routine. The call is ignored if number of the dimension of set is less than one.

The third link generating routine uses the Bresenham algorithm (Bresenham, J. E., "Algorithm for computer control of a digital plotter", IBM Systems J., (4), 1965, pp. 25-30) to generate a series of links that approximates a straight line over an interval of (delta x, delta y). CHLINE (chain, delta x, delta y) adds these links to the end of the specified chain.

Routines also exist to add signal codes to the end of a chain. These are:

INVIS (chain) - invisible chain follows

VISIBL (chain) - visible chain follows

POINT (chain, marker) - a given chain marker number

COLOR (chain, color) - a given color indicator

ELEVAT (chain, value) - a given elevation value

GREY (chain, grey) - a given grey level value

XCOORD (chain, x) - a given x coordinate specifier

YCOORD (chain, y) - a given y coordinate specifier

ROTIND (chain, angle) - a rotation specifier

NODE (chain, node, intersections) - a node specifier where intersections is the number of intersecting chains (zero if undefined)

SCLIND (chain, mode, scale, position) - a scaled chain specifier with position the position of the octal point in the scale factor of type mode

One final routine exists to generate chain elements. This routine is a counterpart to the GET routine (discussed later). A collection of things can be generated by this routine one at a time. PUT (chain, first, second) adds to the end of the specified chain an element as follows:

first	second	element
0-7	0	the link specified by first
10	0	ignored
11	0	invisible chain follows
12	0	visible chain follows
105, 111, 112, 113, 118, 119, 120	x	generates the signal code first - 100 (ie., 05, 13, 14, 15, 22, 23, 24) where second is the string of octal digits to follow the signal code as an octal number (ex. 118, 1 generates a color specifier of 1)
122	x	an x coordinate specifier
123	y	a y coordinate specifier

Only 122 and 123 can have negative values of second. All other values form the octal digits that follow the signal code directly from the value of x. The value of x or y must be in range. The above are the only valid arguments.

B. Input and Output

Chains can be output in a format that can be subsequently read in. Also, chains can be printed in a meaningful form.

INPUT (unit, chain) reads a chain from FORTRAN logical unit unit. Reading takes place until an end-of-chain code is encountered. The chain is checked for validity. The format of the input (assumed to be from cards) is as follows:

columns 1-6	ignored
columns 7-72	chain data
columns 73-80	ignored

Any character that is not an octal digit is ignored. When called, this routine reads in a new card and continues to read cards until it finds the end-of-chain code. The next chain read in (if any) must begin on a new card. The chain read in overrides the previous value of chain; ie., chain is cleared first. This means that the specfied chain cannot be open (discussed later).

OUTPUT (unit, chain) is the opposite of the above. Onto FORTRAN logical unit unit, chain is punched. Punching begins on a new card and continues as necessary until the end-of-chain code is punched. The output format is the same as the above input format. Sequencing numbers starting at 1 and going up by 1 are also generated in columns 73-80.

A more meaningful output can be generated by LIST (unit, chain). This routine lists a given chain on a given FORTRAN logical unit. The output consists of groups of five links. Blanks separate the groups from one another and from signal codes. Fields within signal codes are separated by blanks. No line printed will have more than 120 characters on it. Single spacing is used. The end-of-chain code is printed.

C. Extracting Parts of Chains

These routines look at the logical structure of chains. They provide a means of extracting information about the chain parts and assembling them differently. The first routine listed here does not extract a part but it does look at the physical arrangement of chain segments. INITXY (chain, x, y) scans down a chain looking for x and y coordinate specifiers. The value of the last x and last y coordinate specifier found before the first actual link is returned. If no x (y) coordinate specifier is found, zero is returned for x (y).

One way to identify a portion of a chain is through marker codes in the chain. A section so identified can be copied from one chain (chains) to the end of another (chaind) by MARKER (chaind, chains, first, second). This routine moves the segment between marker first and marker second. If first is out of range, movement starts at the beginning of chain chains. Given that first is in range for markers, chain chains is scanned until the first occurrence of marker first. If this marker is not encountered, nothing is moved. Assuming it is found, the signal code and the chain following it up until and including the first occurrence of marker second is copied. If marker second does not occur after marker first, there is no marker second, or second is out of range for markers, the copy runs until the end of chain chains. Thus, MARKER (chaind, chains, -1, -1) is a copy of chains to the end of chaind.

Three routines in this set copy portions based on the value of signal codes. They look for all segments with an attribute in a given range. COLORD, ELATED, and GREYD all of (chaind, chains, first, second) copy to the end of chaind all portions of chains where the value of the color, elevation, or grey level (respectively) are in range. Copying a segment also copies the signal code that causes the chain to enter or stay in the desired range. If zero is in the desired range, the initial part of *chains* (which has a default initial attribute of zero) is moved along with a signal code with value of zero. The way the range is chosen is as follows: if *second* is greater than or equal to *first*, all values between and including *first* and *second* are copied. If *second* is less than *first*, all values including *first* and *second* except those between *second* and *first* are used.

In cases where there is no other criterion to use as to what segments to move, it is possible to move segments by number. Whenever a signal code occurs, that signal code and what follows is in the next higher numbered segment. Routines that move on the basis of segment numbers are: VISSEG (visible portions only), INVSEG (invisible portions only), NCOLOR (colored), NELEV (elevated), and NGREY (grey level) all of (*chaind*, *chains*, *first*, *second*). *First* and *second* form a range of segment numbers. If *first* is less than one, it is the same as specifying one. A value of *second* less than one is the same as infinity (effectively). Given these meanings, if *second* is greater than *first*, all segments with numbers between and including *first* and *second* are copied. If *second* is less than *first*, all segments including *first* and *second* except those between *second* and *first* are copied. For *first* equal to *second*, only this one value is copied. Note: for all of the above except INVSEG, the default initial attribute means that segment one starts at the beginning of chain *chains* and so if one is in range, a signal code is generated for this signal code (with a value field of zero) and the chain copy starts from the beginning of the chain.

D. Taking a Chain Apart into Primitives

The standard way of working on a chain is to take it link and signal code by link and signal code. A facility is provided to read a chain sequentially in this manner. A chain that is being read must be open. Only a certain number of chains (4) can be open at a time. To open a new one, OPEN (chain) is used. Chain chain cannot already be open. Once opened , INPUT and CLEAR cannot be applied to the chain. GET (chain, first, second) can now be used to get the next piece of information. In providing this information, all multiple link codes are expanded fully and returned link by link. If the next entity is a link, first is set to the link value and second is set to zero. Signal codes are returned as follows:

signal code	first	second
0400	10	0
0401	11	0
0402	12	0
0405xyz	105	xyz_8
0413abcde	111	$abcde_8$
0414abcdef	112	$abcdef_8$
0415abcdefg	113	$abcdefg_8$
0422a	118	a
0423abcd	119	$abcd_8$
0424abc	120	abc_8
0426abcde	122	$abcde_8 -16384$

signal code	<u>first</u>	<u>second</u>
0427abcde	123	$abcde_8$-16384

Default attributes do not get returned as signal codes. All other codes (except 0404, 0417, 0420, and 0421 which are returned link by link in their expansion) are skipped. If the chain is not open, <u>first</u> is set to 10 and <u>second</u> to 0.

The chain can be un-opened by CLOSE (chain). The chain can now be worked on by INPUT and CLEAR. If the chain is re-opened, sequential reading will begin at the beginning of the chain again.

E. <u>Link-by-Link Movements</u>

This routine differs from the previous copying routines in its method of operation. It uses GET and PUT to copy the chain link by link.

SUBCH (<u>chaind</u>, <u>chains</u>, <u>vertex 1</u>, <u>vertex 2</u>) copies onto the end of <u>chaind</u> that part of <u>chains</u> that is between vertices <u>vertex 1</u> and <u>vertex 2</u>. The chain <u>chains</u> is scanned link by link until a link is found ending at a vertex greater than or equal to <u>vertex 1</u>. Copying then takes place until a link is found that ends at a vertex greater than or equal to <u>vertex 2</u>. Specifying <u>vertex 2</u> as less than one is the same as specifying infinity (effectively). The movement includes all signal codes that GET returns. Link repeat codes are constructed as appropriate while moving; however, group repeat codes are not constructed.

PART 4

SUPPORTING ROUTINES

A. Binary Plane

An alternate way of looking at a chain is through a facility in CHAP called the binary plane. The binary plane is an array of bits of dimensions 0 to nx - 1 and 0 to ny - 1 where nx and ny are implementation-dependent constants. Each bit corresponds to a point in the (x, y) plane. A value of 1 for a bit implies that a visible vertex is at that point.

A chain may be added to the binary plane by APLANE (chain). This will set to 1 all bits corresponding to visible vertices in chain. Visible vertices are either the initial vertex or a vertex that is arrived at by a visible link. All other bits in the array are unaffected.

BPLANE (chain) clears the binary plane before adding the chain to the plane.

BPRINT (window) prints the binary plane. Window is a four-element array giving the initial and final x and the initial and final y coordinates of a window in the binary plane. For all 1 bits in the window (including the border), an 'X' is printed. Blanks are printed otherwise.

PRINT (chain, window) clears the binary plane, adds the specified chain to it, and then prints the specified window.

B. Arrays of Links or Vertices

These routines all perform a similar function. They return arrays containing the links or vertices of a chain.

ARRAY (chain, list, dimension of list, no. of elements returned, overflow flag) produces an array of links. The array is filled with all of the links of the chain. The overflow flag is set if list is not large enough to hold all of the links. All signal codes (except link codes) are ignored.

The links of the inverse chain can be found by calling INVERT (chain, list, dimension of list, no. of elements returned, overflow flag). This routine functions in an analogous fashion to ARRAY.

VERTEX (chain, x coordinates, y coordinates, dimension of x coordinates and y coordinates, number of elements returned, overflow flag) is also similar to ARRAY but it returns the x and y coordinates of the vertices of the chain. The values returned are the starting coordinates of all of the links and the final coordinates of the chain. Signal codes are ignored.

CHPAX (chain, list, dimension of list, no. of elements returned) returns the links of the chain in the code of the PAX language. The element after the last link (no. of elements returned + 1) has a value of zero.

PART 5

COMPUTATIONAL ROUTINES

A. Angles, Distances, and Extents

The angle between the x axis and a chord of the chain can be found by ANGLE (chain, vertex 1, vertex 2, angle). The angle is the angle between a directed ray in the positive x direction from the vertex vertex 1 and a directed ray from the vertex numbered vertex 1 to the vertex numbered vertex 2. The resultant angle is in degrees.

LENGTH (chain, length, type) finds the length of a chain. Only invisible links are counted if type is equal to 1 . A type value of 2 will count only visible links. All links are counted for any other value of type.

PDIST (chain, vertex 1, vertex 2, distance) finds the distance between the vertex numbered vertex 1 and the vertex numbered vertex 2. For this routine, vertex i is considered to be the end of link i - 1 except for vertex one which is the start of link one.

The distance from a given point to a chain is scanned by PNTCND (chain, x, y, maximum distance, minimum distance, vertex at maximum distance, vertex at minimum distance). (x, y) is a point from which distances to the vertices of the chain are checked. The maximum and minimum distances to the endpoints of all links and the initial coordinates of the chain (vertex one) are returned.

LNCHD (chain, x1, y1, x2, y2, maximum distance, minimum distance, vertex at maximum distance, vertex at minimum distance) is similar to PNTCND. Here, the maximum and minimum perpendicular distances from the vertices of a chain to a line defined by points (x1, y1) and (x2, y2) are found.

MAXMIN (chain, xy) finds the maximum and minimum points on a chain. Xy is a four-element array which receives, in order, the maximum x, the minimum x, the maximum y, and the minimum y values of all vertices of the chain. "Vertices" here again means the initial coordinates of the chain and the endpoints of all of the links.

WHEX (chain, type, extent) finds the extent (maximum - minimum value) of a chain. Type can have a value of 1, 2, 3, or 4 specifying that the extent is to be computed along the 0, 45, 90, or 135 degree (all respectively) axis relative to the x axis. X and Y coordinate specifiers are ignored in making this computation.

The final routine in this group relates to the difference between the initial and final vertices of a chain. RESID (chain, link type 1, number of link type 1, link type 2, number of link type 2) finds the residue of a chain. The residue of the chain contains the two types of links indicated in the amounts indicated. Using the initial coordinates and this information is the most efficient way to determine the final coordinates of a chain.

B. Areas and Moments

ECAREA (chain, area) returns the integral (area) of all portions of the chain relative to the x axis. A chain portion that is directed in the negative x direction adds a corresponding negative increment of area. Thus, the area of a closed figure is returned by this routine. If the closed figure was encircled counterclockwise, the area will be negative.

The first moment of the chain is computed by MOM1 (chain, angle, first moment). The chain is assumed to be closed and connected although no error will occur if this is not the case. The moment is computed relative to an axis rotated by angle degrees from the x axis. The angle can only be 0, 45, 90, or 135. A clockwise encircled figure above the axis has a positive moment. Reversing either the direction of encirclement or the relation to the axis (but not both) will make the moment negative.

MOM1A (chain, angle, x, y, first moment) computes the first moment of the chain as above but the chain is first translated to have initial coordinates of x and y instead of those it does have.

MOM2 and MOM2A have the same arguments as MOM1 and MOM1A above and perform similar functions except that they return the second moment about the specified axis.

CENTRD (chain, x, y) returns the x and y coordinates of the centroid (center of mass) of the specified figure. The chain is assumed to be closed and connected. The direction of encirclement does not matter.

Another routine that finds a first moment about an axis is LMOM1 (chain, x1, y1, x2, y2, first moment). In this case, the axis is a directed axis from point (x1, y1) to (x2, y2). The chain is assumed to be closed and connected. A clockwise encircled chain above the axis (considering the axis the be the positive x direction, this would correspond to being in the positive y direction) has a positive moment. Reversing either the direction of encirclement or the relation to the axis (but not both) changes the sign of the moment.

LMOM2 has the same arguments as LMOM1 and operates the same as LMOM1 except for returning the second moment of the chain.

C. Correlations

These routines use the temporary array provided in CHAP to compute chain correlations. All signal codes (except link codes) are ignored.

CROSS (chain 1, chain 2, correlation value, shift value) computes a crosscorrelation of chain 1 and chain 2. Chain 2 is shifted by the desired number of links past chain 1. The correlation value is computed with wrap-around on the chains.

AUTO (chain, dimension of correlation array, correlation array, resultant dimension) returns in the correlation array a set of the correlation values in the autocorrelation of the chain. The first element is the autocorrelation function of zero, the second value is the autocorrelation function of one, etc. The resultant dimension of the array is one half of the number of links in the chain. This value is zero if there are not at least two links in the chain.

PART 6

SPECIAL FUNCTIONS

A. Rotation and Scaling

A fundamental chain operation is to rotate and scale. Rotation and scaling requires a requantization of the links. CHAP's rotate and scale algorithm does a requantization but does not do a post edit. Thus, straight lines may not end up perfectly straight. Chains coarsely quantized may end up considerably distorted. Normally, chains retain their shape. ROSCAL (chaind, chains, angle, x scale, y scale) scales chains by the given scale factors and rotates by the specified angle (in degrees) all relative to the start of the chain. The result is added to the end of chaind. When an x or y coordinate specifier is encountered, the new x and y coordinates are computed (given the specified rotation and scaling) and x and y coordinate specifiers are generated. All other non-link codes (those passed by GET) are maintained unchanged but are otherwise ignored.

B. Intersections

All intersections of one chain with another can be found with INTERS (chain 1, chain 2, dimension of all arrays, type of intersections array, chain 1 location array, chain 2 location array, number of intersections found). The result is three arrays whose elements form corresponding three element sets of information. There are two possibilities for each set of three values. If the intersection type is one, the intersection occurs at vertices of the two chains and the chain locations returned are the vertex numbers in the two chains where the intersection occurs. A non-nodal intersection is flagged by an intersection type of two. In this case, the chain locations are the link numbers of the two intersecting links. All signal codes (in particular, x and y coordinate specifiers) except link type codes are ignored. This routine uses temporary workspace.

C. Polygonal Approximation

The algorithm of U. Ramer (in "An Iterative Procedure for the Polygonal Approximation of Plane Curves", Computer Graphics and Image Processing, 1, 1972, pp. 244-256) is implemented by this routine. POLYGN (chain, xcoordinates array, y coordinates array, polygonal approximation vertex numbers, desired tolerance, number of vertices in approximation, dimension of all arrays, number of vertices returned) first sets the first two arrays to the vertices of the chain. All non-link signal codes are ignored in this process. The polygonal approximation vertex numbers are indices of points defined by the x and y coordinates arrays in the polygonal approximation. These points define a polygonal curve such that the maximum distance from a chain vertex to the polygonal section approximating the curve section is less than the desired tolerance. This routine uses temporary workspace.

D. Profiles

These routines provide for profile computation and matching.

XPROFL (chain, flag, profile, column dimension of profile, number of profile pairs returned) computes a maximum edge profile if flag is true and a minimum edge profile if flag is false. Profile is a two row array which will be given the profile pairs. The profile pairs are given in order following the curve counterclockwise. To determine the direction of encirclement, the endpoints are connected by a straight line, the enclosed area is computed, and the sign checked. The chain is inverted, if necessary, to provide the correct direction. This routine uses temporary workspace.

A y edge profile can be computed by YPROFL which functions similarly to XPROFL and has the same arguments.

Two profiles may be matched by MATCH (inward shift, lateral shift, profile 1, column dimension of profile 1, profile 2, column dimension of profile 2). The inward shift and lateral shift given provide for maximum area overlap of encasing rectangles of the two chains.

E. Centroidal Profiles

Another type of profile is the centroidal profile. The centroidal profile contains the distances (measured along the residues) from the centroid to the vertices of the closed chain. The first value in the profile is the greatest value and all values are normalized by the magnitude of the greatest distance. Positive values correspond to vertices that are on portions of the curve that are encircling the centroid in a clockwise sense; negative values correspond to counterclockwise.

CENPRO (chain, array of profile values, dimension of the profile array, number of values returned) computes such a profile.

F. Shape Features

BAYPEN (chain, maximum bay depth, maximum peninsular depth, bay area, peninsular area, chord length) provides some shape features of an open chain. A chord between the initial and final endpoints of the chain are used to determine the values. A bay is an area to the left of this chord; a peninsula is to the right. The maximum distance to the chord in all bays and all peninsulas are computed along with the areas of all bays and the area of all peninsulas.

PART 7

COMMON BLOCKS

The operation of certain routines requires that the user define a certain common block in his program. This is the temporary workspace array. Temporary workspace is defined as: /WORK/ _size_, _array_ where _array_ is an integer array of dimension _size_. _Size_ is also an integer. No initial values need be given to the array.

Temporary workspace is needed by only six routines. Estimates of the space needed by them follows.

AUTO - space to hold all of the links of the given chain

CROSS - space to hold all of the links of both chains

INTERS - space to hold 3 * (2 + sum of the number of links in both chains)

POLYGN - an upper limit is the number of vertices in the chain

XPROFL, YPROFL - space to hold all of the links of the given chain

A system defined common block which may be of use is /XYCOMP/ _ax_, _ay_. The x and y components of the eight link types in order, 0 to 7, are in _ax_ and _ay_, respectively.

The system defined common blocks are defined in a block data subroutine that must be linked together with the program.

PART 8

FORMAL CALL SEQUENCES

ANGLE (CHAIN, NP1, NP2, A)

CHAIN - integer chain array

NP1 - number of vertex 1 (integer)

NP2 - number of vertex 2 (integer)

A - computed angle in degrees (real)

APLANE (CHAIN)

CHAIN - integer chain array

ARRAY (CHAIN, LIST, N, L, OVER)

CHAIN - integer chain array

LIST - array to hold links (integer)

N - dimension of LIST (integer)

L - number of links returned (integer)

OVER - overflow flag (logical)

AUTO (CHAIN, NA, ACORR, M)

CHAIN - integer chain array

NA - dimension of ACORR (integer)

ACORR - resultant autocorrelation values (real)

M - number of values returned (integer)

BAYPEN (CHAIN, MAXBAY, MAXPEN, BAYAR, PENAR, LENGTH)

CHAIN - integer chain array

MAXBAY - maximum bay distance (real)

MAXPEN - maximum peninsular distance (real)

BAYAR - bay area (real)

PENAR - peninsular area (real)

LENGTH - chord length (real)

BPLANE (CHAIN)

CHAIN - integer chain array

BPRINT (IW)

IW - four element window specification

CENPRO (CHAIN, PROFL, DIM, N)

CHAIN - integer chain array

PROFL - profile values (real)

DIM - dimension of PROFL (integer)

N - number of values returned (integer)

CENTRD (CHAIN, CX, CY)

CHAIN - integer chain array

CX - x coordinate of centroid (real)

CY - y coordinate of centroid (real)

CHLINE (CHAIN, DELTX, DELTY)

CHAIN - integer chain array

DELTX - desired x change (integer)

DELTY - desired y change (integer)

CHPAX (CHAIN, LIST, N, L)

CHAIN - integer chain array

LIST - resultant link list (integer)

N - dimension of LIST (integer)

L - number of links (not counting 0) returned (integer)

CLEAR (CHAIN)

CHAIN - integer chain array

CLOSE (CHAIN)

CHAIN - integer chain array

COLOR (CHAIN, VALUE)

CHAIN - integer chain array

VALUE - desired color value (integer)

COLORD (CHAIND, CHAINS, LIMIT1, LIMIT2)

CHAIND - destination chain array (integer)

CHAINS - source chain array (integer)

LIMIT1 - low end limit (integer)

LIMIT2 - high end limit (integer)

CROSS (CHAIN1, CHAIN2, CORR, J)

CHAIN1 - first chain array (integer)

CHAIN2 - array of chain to be shifted (integer)

CORR - resultant correlation value (real)

J - shift value (integer)

ECAREA (CHAIN, S)

CHAIN - integer chain array

S - resultant x axis integral (real)

ELATED (CHAIND, CHAINS, LIMIT1, LIMIT2)

CHAIND - destination chain array (integer)

CHAINS - source chain array (integer)

LIMIT1 - low end limit (integer)

LIMIT2 - high end limit (integer)

ELEVAT (CHAIN, VALUE)

CHAIN - integer chain array

VALUE - desired elevation value (integer)

GET (CHAIN, FIRST, SECOND)

CHAIN - integer chain array

FIRST - link or signal code type (integer)

SECOND - signal code value (integer)

GREY (CHAIN, VALUE)

CHAIN - integer chain array

VALUE - desired grey level (integer)

GREYD (CHAIND, CHAINS, LIMIT1, LIMIT2)

CHAIND - destination chain array (integer)

CHAINS - source chain array (integer)

LIMIT1 - low end limit (integer)

LIMIT2 - high end limit (integer)

INITXY (CHAIN, X, Y)

CHAIN - integer chain array

X - initial x coordinate (integer)

Y - initial y coordinate (integer)

INPUT (UNIT, CHAIN)

UNIT - FORTRAN logical unit (integer)

CHAIN - integer chain array

INTERS (CHAIN1, CHAIN2, J1, ITYPE, INLCHA, INLCHB, NUMINT)

CHAIN1 - first chain array (integer)

CHAIN2 - second chain array (integer)

J1 - dimension of ITYPE, INLCHA, and INLCHB (integer)

ITYPE - type of intersections array (integer)

INLCHA - first chain locations (integer)

INLCHB - second chain locations (integer)

NUMINT - number of intersections found (integer)

INVERT (CHAIN, LIST, N, L, OVER)

CHAIN - integer chain array

LIST - resultant link array (integer)

N - dimension of LIST (integer)

L - number of links returned (integer)

OVER - overflow flag (logical)

INVIS (CHAIN)

CHAIN - integer chain array

INVSEG (CHAIND, CHAINS, LIMIT1, LIMIT2)

CHAIND - destination chain array (integer)

CHAINS - source chain array (integer)

LIMIT1 - low end limit (integer)

LIMIT2 - high end limit (integer)

LENGTH (CHAIN, CHL, LCF)

CHAIN - integer chain array

CHL - computed length (real)

LCF - length type flag (integer)

LINK (CHAIN, LINKS, NTIM)

CHAIN - integer chain array

LINKS - link to be added (integer)

NTIM - number of times to add (integer)

LINKSQ (CHAIN, LIST, DIM, NTIM)

CHAIN - integer chain array

LIST - list of links to add (integer)

DIM - dimension of LIST (integer)

NTIM - number of times to add (integer)

LIST (UNIT, CHAIN)

UNIT - FORTRAN logical unit (integer)

CHAIN - integer chain array

LMOM1 (CHAIN, X1, Y1, X2, Y2, FMNT)

CHAIN - integer chain array

X1 - x coordinate of first point (real)

Y1 - y coordinate of first point (real)

X2 - x coordinate of second point (real)

Y2 - y coordinate of second point (real)

FMNT - computed first moment (real)

LMOM2 (CHAIN, X1, Y1, X2, Y2, SMNT)

CHAIN - integer chain array

X1 - x coordinate of first point (real)

Y1 - y coordinate of first point (real)

X2 - x coordinate of second point (real)

Y2 - y coordinate of second point (real)

SMNT - computed second moment (real)

LNCHD (CHAIN, X1, Y1, X2, Y2, CLDMAX, CLDMIN, JMAX, JMIN)

CHAIN - integer chain array

X1 - x coordinate of first point (real)

Y1 - y coordinate of first point (real)

X2 - x coordinate of second point (real)

Y2 - y coordinate of second point (real)

CLDMAX - maximum distance found (real)

CLDMIN - minimum distance found (real)

JMAX - vertex at maximum distance (integer)

JMIN - vertex at minimum distance (integer)

MARKER (CHAIND, CHAINS, MARK1, MARK2)

CHAIND - destination chain array (integer)

CHAINS - source chain array (integer)

MARK1 - first marker number (integer)

MARK2 - second marker number (integer)

MATCH (XX, YY, PROF1, N1, PROF2, N2)

XX - inward shift computed (integer)

YY - lateral shift computed (integer)

PROF1 - first two row array of profile pairs (integer)

N1 - column dimension of PROF1 (integer)

PROF2 - second two row array of profile pairs (integer)

N2 - column dimension of PROF2 (integer)

MAXMIN (CHAIN, XY)

CHAIN - integer chain array

XY - four element array of maxmin values (integer)

MOM1 (CHAIN, DEGREE, FMNT)

CHAIN - integer chain array

DEGREE - axis angle (integer)

FMNT - computed first moment (real)

MOM1A (CHAIN, DEGREE, NITX, NITY, FMNT)

CHAIN - integer chain array

DEGREE - axis angle (integer)

NITX - initial translated x coordinate (integer)

NITY - initial translated y coordinate (integer)

FMNT - computed first moment (real)

MOM2 (CHAIN, DEGREE, SMNT)

CHAIN - integer chain array

DEGREE - axis angle (integer)

SMNT - computed second moment (real)

MOM2A (CHAIN, DEGREE, NITX, NITY, SMNT)

CHAIN - integer chain array

DEGREE - axis angle (integer)

NITX - initial translated x coordinate (integer)

NITY - initial translated y coordinate (integer)

SMNT - computed second moment (real)

NCOLOR (CHAIND, CHAINS, LIMIT1, LIMIT2)

CHAIND - destination chain array (integer)

CHAINS - source chain array (integer)

LIMIT1 - low end limit (integer)

LIMIT2 - high end limit (integer)

NELEV (CHAIND, CHAINS, LIMIT1, LIMIT2)

CHAIND - destination chain array (integer)

CHAINS - source chain array (integer)

LIMIT1 - low end limit (integer)

LIMIT2 - high end limit (integer)

NGREY (CHAIND, CHAINS, LIMIT1, LIMIT2)

CHAIND - destination chain array (integer)

CHAINS - source chain array (integer)

LIMIT1 - low end limit (integer)

LIMIT2 - high end limit (integer)

NODE (CHAIN, NODES, INTERS)

CHAIN - integer chain array

NODES - node value (integer)

INTERS - number of intersecting chains (integer)

OPEN (CHAIN)

CHAIN - integer chain array

OUTPUT (UNIT, CHAIN)

UNIT - FORTRAN logical unit (integer)

CHAIN - integer chain array

PDIST (CHAIN, NODE1, NODE2, DIST)

CHAIN - integer chain array

NODE1 - first vertex number (integer)

NODE2 - second vertex number (integer)

DIST - computed distance (real)

PNTCND (CHAIN, XP, YP, DMAX, DMIN, LMAX, LMIN)

CHAIN - integer chain array

XP - x coordinate of point (real)

YP - y coordinate of point (real)

DMAX - maximum distance found (real)

DMIN - minimum distance found (real)

LMAX - vertex at maximum distance (integer)

LMIN - vertex at minimum distance (integer)

POINT (CHAIN, VALUE)

CHAIN - integer chain array

VALUE - desired marker number (integer)

POLYGN (CHAIN, XCOORD, YCOORD, ICL, TOL, IC, JJ, K)

CHAIN - integer chain array

XCOORD - x coordinates of vertices (integer)

YCOORD - y coordinates of vertices (integer)

ICL - vertex indices of approximation (integer)

TOL - approximation tolerance (real)

IC - number of vertices in approximation (integer)

JJ - dimension of XCOORD, YCOORD, and ICL (integer)

K - number of vertices returned in XCOORD and YCOORD (integer)

PRINT (CHAIN, IW)

CHAIN - integer chain array

IW - four element window specification array (integer)

PUT (CHAIN, LINK, FLAG)

CHAIN - integer chain array

LINK - link or signal code type (integer)

FLAG - signal code flag (integer)

RESID (CHAIN, LRES1, NLRES1, LRES2, NLRES2)

CHAIN - integer chain array

LRES1 - first link type (integer)

NLRES1 - number of links of type LRES1 (integer)

LRES2 - second link type (integer)

NLRES2 - number of links of type LRES2 (integer)

ROSCAL (CHAIND, CHAINS, ANGLE, XSCALE, YSCALE)

CHAIND - destination chain array (integer)

CHAINS - source chain array (integer)

ANGLE - rotation angle (real)

XSCALE - x scale factor (real)

YSCALE - y scale factor (real)

ROTIND (CHAIN, ANGLE)

CHAIN - integer chain array

ANGLE - desired angle specifier (real)

SCLIND (CHAIN, MODE, SCALE, POS)

CHAIN - integer chain array

MODE - mode of scaling (integer)

SCALE - scale factor (integer)

POS - position of octal point (integer)

SUBCH (CHAIND, CHAINS, VERT1, VERT2)

CHAIND - destination chain array (integer)

CHAINS - source chain array (integer)

VERT1 - first vertex number (integer)

VERT2 - second vertex number (integer)

VERTEX (CHAIN, XCOORD, YCOORD, N, L, OVER)

CHAIN - integer chain array

XCOORD - x coordinates of vertices (integer)

YCOORD - y coordinates of vertices (integer)

N - dimension of XCOORD and YCOORD (integer)

L - number of vertices returned (integer)

OVER - overflow flag (logical)

VISIBL (CHAIN)

CHAIN - integer chain array

VISSEG (CHAIND, CHAINS, LIMIT1, LIMIT2)

CHAIND - destination chain array (integer)

CHAINS - source chain array (integer)

LIMIT1 - low end limit (integer)

LIMIT2 - high end limit (integer)

WHEX (CHAIN, ITYPE, W)

CHAIN - integer chain array

ITYPE - extent type desired (integer)

W - computed extent (real)

XCOORD (CHAIN, VALUE)

CHAIN - integer chain array

VALUE - desired x value (integer)

XPROFL (CHAIN, FLAG, PROFL, N, INDEX)

CHAIN - integer chain array

FLAG - maximum edge flag (logical)

PROFL - two row array of profile pairs (integer)

N - column index of PROFL (integer)

INDEX - number of profile pairs returned (integer)

YCOORD (CHAIN, VALUE)

CHAIN - integer chain array

VALUE - desired y value (integer)

YPROFL (CHAIN, FLAG, PROFL, N, INDEX)

CHAIN - integer chain array

FLAG - maximum edge flag (logical)

PROFL - two row array of profile pairs (integer)

N - column index of PROFL (integer)

INDEX - number of profile pairs returned (integer)

PART 9

RESERVED NAMES

The following names are reserved in that the user should not define routines or common blocks with these names.

ADDR	ECAREA	LMOM2	NVIS	ROSCAL
ANGLE	ELATED	LNCHD	OCTAL	ROTIND
APLANE	ELEVAT	LOCTA	OCTIN	SCLIND
ARRAY	GET	MARKER	OPEN	SIGEND
AUTO	GETDIG	MATCH	OUTPUT	SIGLIS
BAYPEN	GREY	MAXMIN	PCLOSE	SIGNAL
BPLANE	GREYD	MOM1	PDIST	SPACET
BPRINT	GTLINK	MOM1A	PLANES	STATUS
CENPRO	ICHDIG	MOM2	PNTCND	STLINK
CENTRD	INITXY	MOM2A	POINT	SUBCH
CHAPMC	INPUT	MSTORE	POLYGN	VERTEX
CHDIG	INTERS	NCOLOR	PPUT	VISIBL
CHLINE	INVERT	NCOLS	PRINT	VISSEG
CHPAX	INVIS	NDUMB	PROFIL	VSEG
CLEAR	INVSEG	NELEV	PUT	WHEX
CLOSE	LENGTH	NELVS	PUTCH	WORK
COLOR	LINIT	NGREY	PUTCHF	XCOORD
COLORD	LINK	NGRYS	PUTDIG	XPROFL
CROSS	LINKSQ	NINV	QCOMM	XYCOMP
DIGCH	LISDIG	NMOVE	QMOVE	YCOORD
DIGLIS	LIST	NODE	RESID	YPROFL
DUMB	LMOM1	NTHSEG		

PART 10

ERROR MESSAGES

ANGLE:	NODE NUMBERS ILLEGAL
	NODE NUMBERS OUT OF RANGE FOR THIS CHAIN
	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
APLANE:	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
ARRAY:	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
AUTO:	ARRAY OVERFLOW
	TEMPORARY ARRAY OVERFLOW
	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
BAYPEN:	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
BPLANE:	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
BPRINT:	none
CENPRO:	ARRAY OVERFLOW
	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
CENTRD:	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
CHLINE:	CHAIN OVERFLOW
CHPAX:	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
	ARRAY OVERFLOW

CLEAR:	CHAIN IS OPEN
CLOSE:	none
COLOR:	NUMBER NOT IN RANGE
	CHAIN OVERFLOW
COLORD:	ILLEGAL LIMITS
	CHAIN OVERFLOW
	SOURCE AND DESTINATION CHAINS ARE THE SAME
CROSS:	TEMPORARY ARRAY OVERFLOW
	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
ECAREA:	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
ELATED:	ILLEGAL LIMITS
	CHAIN OVERFLOW
	SOURCE AND DESTINATION CHAINS ARE THE SAME.
ELEVAT:	NUMBER NOT IN RANGE
	CHAIN OVERFLOW
GET:	none
GREY:	NUMBER NOT IN RANGE
	CHAIN OVERFLOW
GREYD:	ILLEGAL LIMITS
	CHAIN OVERFLOW
	SOURCE AND DESTINATION CHAINS ARE THE SAME
INITXY:	none
INPUT:	CHAIN IS OPEN
	CHAIN OVERFLOW
	INVALID SCALE MODE
	ILLEGAL SIGNAL CODE

INTERS:	ARRAY OVERFLOW
	TEMPORARY ARRAY OVERFLOW
	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
INVERT:	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
INVIS:	CHAIN OVERFLOW
INVSEG:	CHAIN OVERFLOW
	SOURCE AND DESTINATION CHAINS ARE THE SAME
LENGTH:	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
LINK:	NUMBER OF TIMES IS TOO BIG
	ILLEGAL LINK
	CHAIN OVERFLOW
LINKSQ:	ILLEGAL LINK
	TOO MANY LINKS
	NUMBER OF TIMES IS TOO BIG
	CHAIN OVERFLOW
LIST:	none
LMOM1:	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
LMOM2:	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
LNCHD:	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
MARKER:	CHAIN OVERFLOW
	SOURCE AND DESTINATION CHAINS ARE THE SAME
MATCH:	none

MAXMIN:	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
MOM1:	ILLEGAL ANGLE
	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
MOM1A:	ILLEGAL ANGLE
	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
MOM2:	ILLEGAL ANGLE
	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
MOM2A:	ILLEGAL ANGLE
	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
NCOLOR:	CHAIN OVERFLOW
	SOURCE AND DESTINATION CHAINS ARE THE SAME
NELEV:	CHAIN OVERFLOW
	SOURCE AND DESTINATION CHAINS ARE THE SAME
NGREY:	CHAIN OVERFLOW
	SOURCE AND DESTINATION CHAINS ARE THE SAME
NODE:	NUMBER NOT IN RANGE
	CHAIN OVERFLOW
OPEN:	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
OUTPUT:	none
PDIST:	NODE NUMBERS OUT OF RANGE FOR THIS CHAIN
	CHAIN ALREADY OPEN

	TOO MANY CHAINS OPEN
PNTCND:	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
POINT:	NUMBER NOT IN RANGE
	CHAIN OVERFLOW
POLYGN:	ARRAY OVERFLOW
	TEMPORARY ARRAY OVERFLOW
	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
PRINT:	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
PUT:	ARGUMENTS NOT VALID (invalid combination)
	NUMBER NOT IN RANGE
	CHAIN OVERFLOW
RESID:	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
ROSCAL:	CHAIN OVERFLOW
	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
	SOURCE AND DESTINATION ARE THE SAME
	NUMBER NOT IN RANGE (overflow on x or y coordinates)
ROTIND:	CHAIN OVERFLOW
SCLIND:	NUMBER NOT IN RANGE
	CHAIN OVERFLOW
SUBCH:	CHAIN OVERFLOW
	SOURCE AND DESTINATION CHAINS ARE THE SAME

VERTEX:	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
VISIBL:	CHAIN OVERFLOW
VISSEG:	CHAIN OVERFLOW
	SOURCE AND DESTINATION CHAINS ARE THE SAME
WHEX:	ILLEGAL EXTENT TYPE
	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
XCOORD:	NUMBER NOT IN RANGE
	CHAIN OVERFLOW
XPROFL:	DISCONTINUITY ON EDGE
	ARRAY OVERFLOW
	TEMPORARY ARRAY OVERFLOW
	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN
YCOORD:	NUMBER NOT IN RANGE
	CHAIN OVERFLOW
YPROFL:	DISCONTINUITY ON EDGE
	ARRAY OVERFLOW
	TEMPORARY ARRAY OVERFLOW
	CHAIN ALREADY OPEN
	TOO MANY CHAINS OPEN

PART 11

PROGRAMMING EXAMPLE

The following program was run through the IBM version of CHAP.

```
      INTEGER HEART,CHAIN,WINDOW,PART,I
      REAL AREA,LENG,MOMENT
      DIMENSION HEART(90),CHAIN(40),WINDOW(4),PART(40)
      DATA HEART/1,0,1,0,1,0,1,0,1,1,0,1,1,1,1,1,2,1,2,2,1,2,2,3,2,
     1          3,2,3,3,4,3,4,3,4,4,4,4,4,5,4,4,5,6,5,6,2,3,2,3,4,
     2          4,3,4,4,4,4,4,5,4,5,4,5,5,6,5,6,5,6,6,7,6,6,7,6,7,
     3          7,7,7,7,0,7,7,0,7,0,7,0,7,0,7/
      DATA CHAIN/360,39*0/,WINDOW/0,40,0,40/,PART/360,39*0/
         note that CHAIN and PART can hold 10 * (40 - 4) links
C NOTE: WORKSPACE ARRAY NOT NEEDED by any called routine here
C CREATE A FIGURE - INPUT IS WORD CHAP WITH C A COLOR OF 1, H A COLOR
C       OF 2, A A COLOR OF 3, AND P A COLOR OF 4; ADD A HEART OF COLOR 7
      CALL INPUT(5,CHAIN)
      CALL COLOR(CHAIN,7)
      CALL XCOORD(CHAIN,18)
      CALL YCOORD(CHAIN,0)
      CALL LINKSQ(CHAIN,HEART,90,1)
      CALL LIST(6,CHAIN)
      CALL PRINT(CHAIN,WINDOW)
      CALL COLORD(PART,CHAIN,4,1)
      CALL LIST(6,PART)
      CALL PRINT(PART,WINDOW)
C USE ONLY HEART IN COMPUTATIONS (remove all letters)
      CALL CLEAR(CHAIN)
      CALL COLORD(CHAIN,PART,7,7)
      CALL LENGTH(CHAIN,LENG,0)
      WRITE(6,1)LENG
    1 FORMAT(20H LENGTH OF HEART IS:,F9.4)
      CALL ECAREA(CHAIN,AREA)
      WRITE(6,2)AREA
    2 FORMAT(18H AREA OF HEART IS:,F11.4)
      CALL MOM1(CHAIN,0,MOMENT)
      WRITE(6,3)MOMENT
    3 FORMAT(30H FIRST MOMENT ABOUT X AXIS IS:,F11.4)
      CALL MOM2(CHAIN,90,MOMENT)
      WRITE(6,4)MOMENT
    4 FORMAT(42H SECOND MOMENT ABOUT (INVERTED) Y AXIS IS:,F13.4)
      STOP
      END
```

The input to this program was the following.

```
EXAMPLE   04221042640011042740015554433222211OO77  C
          0422204264001304274001304172010042640021O4176010
                04274001744444  H
          04223042640023042740013222121221766767 66
                0426400300427400164444   A
          0422404264003304274001322222222000007665444440400   P
```

In the following output, the chain listings are modified to fit on the page.

```
0422 1  0426 40011  0427 40015  55443 32222 11007 7  0422 2
        0426 40013  0427 40013  0417 2 010  0426 40021  0417 6 010
```

color of 1, x coordinate of 11_8, y coordinate of 15_8, a series of links; color of 2, x of 13_8, y of 13_8, 10_8 occurrences of link 2, x of 21_8, 10_8 occurrences of link 6

```
0427 40017  44444 4  0422 3  0426 40023  0427 40013  22121
        22176 67676 6  0426 40030  0427 40016  4444  0422 4  0426
```

y of 17_8, a series of links; color of 3, x of 23_8, y of 13_8, a series of links, x of 30_8, y of 16_8, a series of links; color of 4, start of x specifier

```
40033  0427 40013  22222 22200 00076 65444 44  0422 7  0426
        40022  0427 40000  0421 132 0001 10101 01011 01111 12122
```

x of 33_8, y of 13_8, a series of links; color of 7, x of 22_8, y of 0_8, group repeat of length 132_8 to be repeated once

```
12232 32334 34344 44454 45656 23234 43444 44545 45565 65667
        66767 77770 77070 70707  0400
```

(14 blank lines)

Notice the division in the chain. The chain has five segments each starting with a color specifier and then an x and y coordinate specifier. The first four groups (C, H, A, and P) were on the data cards. The fifth group (heart) was added piece by piece explicitly.

The 14 blank lines are part of the window in the print call.

```
        XXXXXX         XXXXXX
      XX      XXX   XXX      XX
    XX           X X           XX
   X             X X             X
  X               X               X
  X               X               X
 X                                 X
 X   XXX   X          X    XXXXXX  X
X   X   X  X     X   X X   X     X  X
X  X     X X     X   X X   X     X  X
X  X       X     X   X X   X     X  X
 X X       XXXXXXX  X   X  XXXXXX  X
 X X       X     X  XXXXX  X       X
 X X     X X     X X     X X       X
  X X   X  X     X X     X X      X
  X  XXX         X       X        X
   X                             X
    X                           X
     X                         X
      X                       X
       XX                   XX
         X                 X
          XX             XX
            XX         XX
              XX     XX
                XX XX
                  X
```

```
0422 0  0422 1  0426 40011  0427 40015  55443 32222 11007 7
        0422 4  0426 40033  0427 40013  22222 22200 00076 65444 44
```

a color of 0; color of 1, x of 11_8, y of 15_8, a series of links;
color of 4, x of 33_8, y of 13_8, a series of links

```
0422 7  0426 40022  0427 40000  0421 132 0001 10101 01011 01111
        12122 12232 32334 34344 44454 45656 23234 43444 44545
```

color of 7, x of 22_8, y of 0, group repeat of length 132_8 to be
repeated once

```
45565 65667 66767 77770 77070 70707  0400
```

(14 blank lines)

Notice that only the endpoints of links are printed; this explains the missing points in the letters. The extracted chain parts, listed above, show that the portions with colors between 1 and 4 have been removed. Since 0 is in range (default attribute), a color specifier of 0 appears at the beginning of the chain.

```
        XXXXXX         XXXXXX
      XX      XXX   XXX      XX
    XX           X X           XX
   X             X X             X
  X               X                X
  X               X                X
 X                                  X
 X   XXX                   XXXXXX   X
X   X   X                  X      X  X
X  X     X                 X      X  X
X  X                       X      X  X
 X X                       XXXXXX   X
 X X                       X        X
 X X     X                 X        X
  X X   X                  X       X
  X  XXX                           X
   X                              X
    X                            X
     X                          X
      X                        X
       XX                    XX
         X                  X
          XX             XX
            XX         XX
              XX     XX
                XX XX
                  X
LENGTH OF HEART IS: 108.2254
AREA OF HEART IS:  -664.0000
FIRST MOMENT ABOUT X AXIS IS: -9753.6172
SECOND MOMENT ABOUT (INVERTED) Y AXIS IS:  267600.3125
```

The letters H and A no longer appear because their colors were between 1 and 4. All else remains the same.

The computations were performed on the heart. The C and P were removed by taking only those portions with a color of 7. The heart still has an origin of (18, 0). As such, the axis runs through the bottommost and leftmost points of the heart. The heart is drawn out counter-clockwise. This is the reason for the negative area and first moment. Remember that the second moment was computed about an inverted axis.

PART 12

MACHINE DEPENDENCIES

A. IBM 360 / 370

The IBM 360 / 370 implementation stores both chain data and binary plane data packed 30 bits per word in the low order bits. A full-word integer holds 10 links.

The binary plane size is 120 X 120 bits.

Arrays of chains can be used as described in the text.

B. UNIVAC 1108 / 1110

The full 36 bit UNIVAC word is used to store chain data and binary plane data. An integer holds 12 links.

The binary plane size is 144 X 120.

Arrays of chains can be used as described in the text.

PART 13

REFERENCES

1. Freeman, H., "On the Encoding of Arbitrary Geometric Configurations", IRE Trans. on Elect. Computers, EC-10, (2), June 1961, 260-268.

2. __________, "Techniques for the Digital Computer Analysis of Chain-Encoded Arbitrary Plane Curves", Proc. Nat'l. Elect. Conf., 17, 321-421, October 1961.

3. __________, "Boundary Encoding and Processing", in Picture Processing and Psychopictorics, ed. by B. Lipkin and A. Rosenfeld, Academic Press, Inc., New York, 1970.

4. __________, "Computer Processing of Line-Drawing Images", Computing Surveys, 6, (1), March 1974, pp. 57-97.

5. Johnston, E. G., "The PAX User's Manual", Computer Science Center, University of Maryland, College Park, MD. 20742, June 1972, (NTIS AD745 973).

6. Loepere, K., "Documentation Manual for CHAP", Technical Report CRL-57, ESE Department, Rensselaer Polytechnic Institute, Troy, N. Y. 12181.

PART 14

SIGNAL CODE SUMMARY

The following is a list of the signal codes recognized by CHAP. A complete list can be found in the references (4).

0400 - end of chain code

0401 - invisible chain follows

0402 - visible chain follows

0403 - ignored

0404 - 04 link combination

0405xyz - marker xyz

0406 - illegal

0407uv, $_{uv}$, - identification number

0410wxyz, $_{wxyz}$, - comment

0411 . . . 04127777 - comment

0413abcde - abcd - node number, e - number of intersecting chains

0414abcdef - rotation indicator of a.bcdef$_8$ radians

0415abcdefg - a - mode of scaling, bcdef - scale factor, g - position of octal point

0416 - illegal

0417uxyz - link u is to be repeated xyz times

0420un, $_{n+4}$, - repeat link u the number of times specified

0421abcwxyz, $_{abc}$, - repeat the group of abc links wxyz times

0422u - color indicator of u

0423abcd - elevation indicator of abcd

0424abc	grey level of abc
0425u	ignored
0426abcde	x coordinate of abcde - 40000_8
0427abcde	y coordinate of abcde - 40000_8

PART 15

ROUTINE INDEX

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 78-1038	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) CHAP USER'S MANUAL		5. TYPE OF REPORT & PERIOD COVERED Interim
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Keith P. Loepere		8. CONTRACT OR GRANT NUMBER(s) AFOSR 76-2937
9. PERFORMING ORGANIZATION NAME AND ADDRESS Electrical & Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, New York 12181		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research/NM, Bolling AFB, Washington, DC 20332		12. REPORT DATE May 1978
		13. NUMBER OF PAGES 56
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

line drawing processing
image processing
pattern recognition
cartography
computer graphics
graphics languages

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A computer representation for line drawings that has been found particularly convenient and has become widely accepted is the chain code. This representation is compact and allows most common processing functions to be performed efficiently.

CHAP is a collection of FORTRAN routines designed to process chain-encoded line drawings. Routines exist in CHAP to manipulate, → [illegible]

20.

synthesize, analyze, and do input and output upon chains. This report is intended to serve as a user's manual for CHAP. It describes the use of the CHAP routines along with other information needed to write a program utilizing the routines. A programming example is included.