

# 画像解釈言語 PILS の文法

森 英 雄  
赤 羽 秀 友  
大 島 昌 彦  
神 田 雄 一

(昭和56年 8 月31日受理)

## Picture Interpretation Language for Silhouette: PILS

by Hideo MORI, Hidetomo AKABANE, Akihiko OOSHIMA,  
and Yuich KANDA

### Abstract

This report is intended to serve as a concise, ultimate reference for both programmers and implementers.

PILS is a problem oriented language designed for image representation and understanding of such silhouette images as stomach barium-filled X-ray image, micro-scopic image, chinese chracter etc. 1) PILS represents an image by two data structures; macro-contour and micro-contour. 2) Macro-contour is composed of five hierachical picture elements, which are represented by twelve attributes: area, height, width, etc. Micro-contour is represented by an arc sequence. An arc is represented by seventeen attributes: length, tangent, shape, etc. 3) The attribute and relation of picture element are accessed by picture operators. 4) Searching statement and associative memory are arranged to describe pattern recognition processes. 5) PILS user can define his own picture operators.

### 1. ま え が き

画像解釈言語 PILS (Picture Interpretation Language for Silhouette) は、面画像あるいは立体画像のうちで、何らかの変換処理によってシルエット、すなわち白黒の面画像に還元できるような画像を対象として、その形態の認識や分類のアルゴリズムをわかり易く簡潔に記述するために考案した言語である。その特徴は次のとおりである。

- 1) 初心者でも使える覚え易い使い易い問題向き言語である。
- 2) 画像データは、画像のおおまかな形態を認識するのに必要なマクロ輪郭と、画像の辺縁の細かな凹凸を識別するのに必要なミクロ輪郭からなる。
- 3) マクロ輪郭は5種類の画像要素からなる階層的データ構造で表わし、各画像要素は面積、高さ、幅、

重心座標等の属性で抽象化する。

- 4) マクロ輪郭をガイドにして辺縁を密に走査してミクロ輪郭を抽出する。ミクロ輪郭は、辺縁の濃度勾配の場所による変動に対処するために、3本の局所的等濃度曲線で表わす。3本の輪郭線は、尖点や屈曲点で向きのついた弧に分割する。弧は始端と終端の座標、接線ベクトル方向、弧の形状(凸状、凹状、直線状、ジグザグ状、短線分、ブランク)等の属性で抽象化する。
- 5) 対象の状態やカテゴリ名等の概念データを表わすために、2種類の文字列タイプがある。文字の綴りの誤りは、データの入出力時にチェックする。
- 6) 画像要素の属性をアクセスしたり、その画像要素とデータ構造上親子兄弟関係にある画像要素をアクセスするために、53種類の基本画像演算子がある。

- 7) プログラムはある対象にのみ適用できるようなセマンテックな情報を含んだ専用の画像演算子を定義できる。その定義自身も PILS で記述できる。
- 8) マクロ輪郭の中から目的とする画像要素を探索するための文として、PARASCAN 文（並列同時走査）と SEQSCAN 文（直列継時走査）がある。ミクロ輪郭の探索文として SKIP 文がある。
- 9) 画像要素とその評価値を対にして記憶しておき、対の一方をキーにして他の一方を取出す連想記憶の機能がある。さらに、評価値の大きい順または小さい順に画像要素を取出す最大・最小選択器の機能がある。
- 10) ASSIGN 文で定数の名前とその値を宣言しておくと、プログラムの実行 (RUN) の最初でそれらの名前と値のリストがコンソール 端末に表示される。その時点でコンソール 端末から値を変更できる。
- 11) 計測した結果を数値または概念データの形で保存するファイル機能がある。
- 12) PILS システムは PASCAL で書いてあるので移植が容易である。
- 13) PILS の原始プログラム中に in line で PASCAL プログラムを挿入することができる。したがって PILS プログラムと PASCAL プログラムの結合が容易にできる。
- 14) 画像入出力装置のハンドラーは、PILS システム サブルーチンライブラリに含まれ、PILS コンパイラとは独立しているので、FSS や TV 等のさまざまな仕様の入出力装置がシステム サブルーチンの変更によって使える。
- 15) PILS システムはコンパクトにできているので 32kW 程度のミニコン上で走る。

応用分野としては次のようなものがある。

- 1) 胃 X 線バリウム充満像の認識と診断
- 2) 手書き漢字認識
- 3) 顕微鏡写真の粒子の形状による分類と計測
- 4) 植物の葉や花の形態による分類
- 5) 認知科学における視覚認知モデル
- 6) ロボットアイ

PILS の設計思想と解説については文献<sup>1)2)</sup>を参照されたい。

## 2. 記法と用語

PILS プログラムで使用する文字は、英字と数字および特殊記号である。プログラムの最初は MEASU

RE で始まり、MEND で終わる。MEASURE 以前に書いたプログラムは、in line PASCAL コードを除いて意味を持たない。改行やスペースコードは意味を持たないのでどこに入れても良いが、変数や予約語等の単語の中に入れてはならない。幾つかの予約語からなる複雑な文は、予約語のところで改行すると読みやすい。

PILS は PASCAL 上に implement しているので、文法の一部、特に式のかき方は PASCAL のそれに似ている。以下の文法は拡張 Backus-Naur 形式で記す。

<英字> ::= A|B|C|D|E|F|G|H|I|J|K|L|M|  
N|O|P|Q|R|S|T|U|V|W|X|Y|Z

<数字> ::= 0|1|2|3|4|5|6|7|8|9

<特殊記号> ::= #|+|-|\*|/|( )|=|,| [ ]|. ' |

<予約語> ::= AND|ANTICL|ASSIGN|AXIS|  
BY|CLEAR|CLOSE|COMPONENT|  
CRT|CURSOR|CUT|DRIVE|EMPTY|  
ERASE|EXIT|EXTERNAL|FALSE|  
FEED|FILE|FILTER|FROM|  
GETARC|GLOBAL|GO|IF|INTO|  
KEY|MACRO|MEASURE|MEND|  
MOD|NOT|OPDEF|OPEND|OR|  
ORGANIZE|PARAEND|PARASCAN|  
PENUP|PLOT|PUT|READ|READS|  
EGM|REARC|REORG|RESET|RE|  
WRITE|SCOPE|SEQEND|SEQSCAN|  
SET|SKIP|STEP|TAKE|TAKEMAX|  
TAKEMIN|TAKEPIC|TAKEVAL|  
TARGET|TERM|TO|TRACE|TRUE|  
UNTIL|WHILE|WRITE|WRITESE|  
GM|ZONE|

予約語は文の見出しに使うので変数名として使ってはならない。

[\* と \*] で囲まれた文は注釈で翻訳の対象とならない。行の最初の文字が; (セミコロン) で始まる文は in line PASCAL コードと見なされ目的プログラムにそのまま吐出される。MEASURE 文の前にも in line PASCAL コードが書ける。

## 3. 画像データの表現

PILS では画像を2種類のデータ構造で表現する。一つはマイクロ輪郭で、あと一つはミクロ輪郭である。

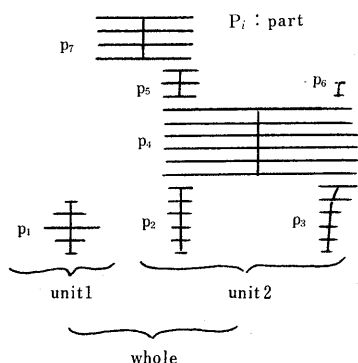


図-1 マクロ輪郭

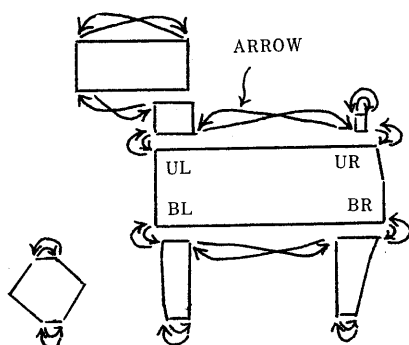


図-2 マクロ輪郭の多角形表現

### 3.1 マクロ輪郭

マクロ輪郭は画面を粗く走査して得られるおおまかな輪郭で、そのデータ構造は6種類の画像要素の階層構造で表わされる。下位のものから順に edge, segment, part, pcorner, unit, whole と呼ぶ。

**edge**: 画面を等間隔の帯状領域 (zone) に分割する。zone を粗く走査して得た加算投影 (signature) をもとに抽出した対象のエッジを edge と名づける。edge はその属性として、 $x, y$  座標、edge の近傍の対象内点の濃淡値、edge 近傍の加算投影曲線のモード等を有する。

**segment:** 対象の左右の edge の対を segment と  
言う。図-1 で segment は水平線分で表わされてい  
る。

**part**: 構造化の規則に従って連結した **segment** の集合を **part** と言う。構造化の規則とは、「上下方向に1対1に重なり合った **segment** の中で、中心の位置ずれと長さの変動が小さいものは連結する」と言う規則である。図-1で、1本の中心線で結ばれた水平線分の集合が一つの **part** を表わす。**part** は図-2のような上辺と底辺が平行な多角形でも表わされる。

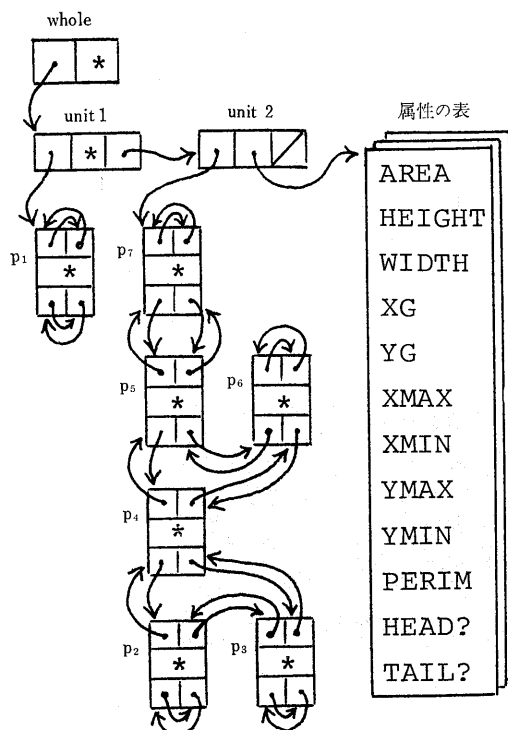


図-3 マクロ輪郭のデータ構造

part の属性は面積 (AREA), 高さ (HEIGHT), 幅 (WIDTH), 重心座標 (XG, YG),  $x$  座標および  $y$  座標の最大値と最小値 (XMAX, XMIN, YMAX, YMIN), および頭か尾かのフラグ (HEAD?, TAIL?) からなる。図-1 で  $P_7$  と  $P_6, P_1$  が頭で  $P_2$  と  $P_3, P_4$  が tail である。

**pcorner**: part の上辺と底辺のかどをそれぞれ BL (Bottom Left), BR (Bottom Right), UL (Upper Left), UR と名づける (図-2 参照)。part とかどの対を pcorner と言う。図-2 に示すように、各 pcorner から **arrow** と呼ぶポインタが出ている。arrow

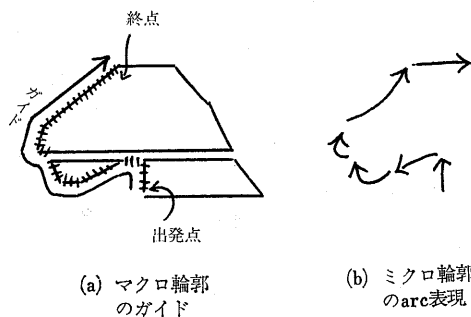


图-4

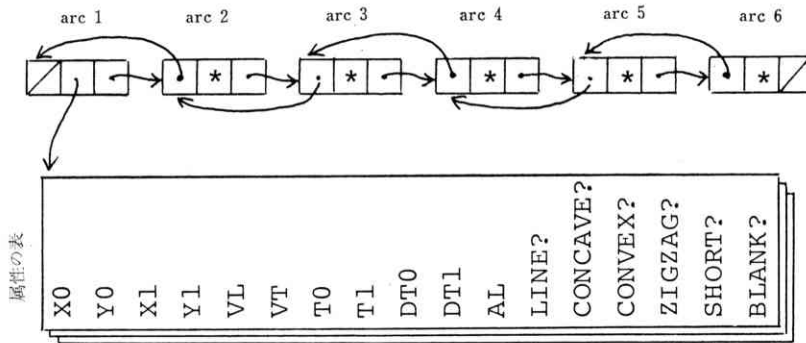
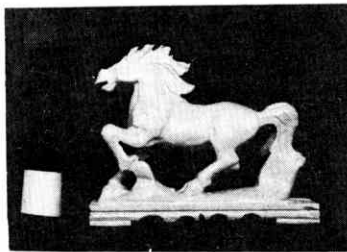
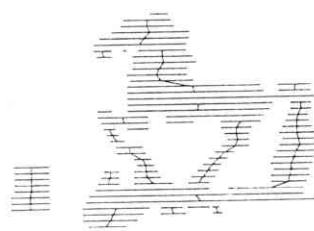


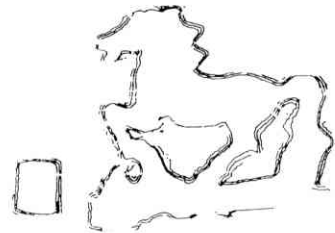
図-5 ミクロ輪郭のデータ構造



a) 原画像



b) マクロ輪郭



c) ミクロ輪郭

図-6 具 体 例

は pcorner 同志を連結する。図-3 に図-1 のマクロ輪郭の arrow の連結の様子を示す。pcorner の属性はかどの座標である。

**unit:** arrow で連結した part の集合を unit と言う。unit は対象の弧立した部分を作る単位である。unit の属性は、part の属性から頭と尾のフラグを除いて周辺長 (PERIM) をつけ加えたものである。

**whole:** 一つの画面の中に含まれる unit の集合を whole と言う。whole の属性は unit と同じである。

図-3 は図-1 のマクロ輪郭のデータ構造を示す。

### 3.2 ミクロ輪郭

マクロ輪郭の edge\* をガイドにして、その近傍を密に走査して得た 3 本の局所的等濃度曲線をミクロ輪郭と言う (図-4 参照)。これらは対象の内側に位置するものから順に内側輪郭線、真中輪郭線、外側輪郭線と言う。一般に内側輪郭線ほど滑らかで、外側に行くほど凹凸が激しくなる。3 本のミクロ輪郭線の間隔の開き具合で、辺縁の濃度勾配がわかる (図-6(c) 参照)。各ミクロ輪郭線は尖点や変曲点で分割して、向きのついた弧で表わす。図-5 にそのデータ構造を表わす。

**arc:** 向きのついた弧を arc と言う。arc の属性は

弧の始端の座標 ( $X_0, Y_0$ ), 終端の座標 ( $X_1, Y_1$ ), 始端から終端へ引いたベクトルの  $x, y$  成分 ( $DX, DY$ ), そのベクトルの長さ ( $VL$ ), と方向 ( $VT$ ), 始端および終端における接線ベクトルの方向 ( $T_0, T_1$ ), 隣接する arc の接線ベクトとの差分 ( $DT_0, DT_1$ ), 弧の長さ ( $AL$ ), 弧の形状 ( $LINE?$   $CONCAVE?$ ,  $CONVEX?$ ,  $ZIGZAG?$ ,  $SHORT?$ ,  $BLANK?$  のいずれか) 等である。

マクロ輪郭とミクロ輪郭の例を図-6 に示す。マクロ輪郭抽出には MACRO 文をミクロ輪郭抽出には GETARC 文を用いる。

## 4. 概念データと算術データの表現

対象の名称や状態を表わす文字列データ (標準では 10 文字以内) を概念データと言う。概念データは主にデータベースを作るときに用いられ、オペレータが与えたり、プログラムで作り出す。オペレータやプログラマの概念データの綴りの誤りを防ぐため、term 型と set 型の宣言を用意している。

**term 型:** TERM 文で宣言した文字列の集合の中の一つの要素を値とする型である\*。

**set 型:** SET 文で宣言した文字列の集合の部分集

\* 座標と対象内点濃淡値モード

\* PASCAL の scalar を文字列に拡張したもの。

表-1 名前のつけ方

画像型	画像要素名	算術型	変数名
whole	$\alpha\alpha$ WW	整数	$\alpha\alpha$ II
unit	$\alpha\alpha$ UU	文字列	$\alpha\alpha$ SS
part	$\alpha\alpha$ PP	論理	$\alpha\alpha$ ?
pcorner	$\alpha\alpha$ CC	実数	$\alpha\alpha\beta\beta$
edge	$\alpha\alpha$ EE	term 型	$\alpha\alpha\beta\beta$
arc	$\alpha\alpha$ AA	set 型	$\alpha\alpha\beta\beta$

注)  $\alpha\alpha$ : 先頭が英字の 8 字以内の英数字

$\alpha\alpha\beta\beta$ : 末尾が WW,..., AA, II,..., ? 以外の文字で終わる 10 字以内の英数字

合 (全集合と空集合を含む) を値とする型である\*。

プログラムの先頭でこれらの型の変数の宣言をしておくと、プログラムのコンパイル時または入出力命令で変数に値をセットする時点で、値の妥当性をチェックする。

算術データは整数型、実数型\*\*, 文字列型 (10 文字以内)、論理型 (TRUE か FALSE を値とする) のいずれかで表わす。

## 5. 名前のつけ方

画像要素を指す名前を画像要素名と言う。画像要素名も変数名もファイル名も全て 10 字以内の先頭が英字で始まる英数字列で、末尾の 2 文字でその型を示すように名づけなければならない。末尾の文字の規則を表-1 に示す。ただし、term 型 set 型の変数の型宣言は TERM 文と SET 文です。

## 6. 定数

整数、実数、文字列、論理型の定数の表わし方は PASCAL に準じる。画像タイプの定数は EMPTY だけである。

例. 1 1, -100, 0.1, 5E-3, 87.35 E+8, 'A', 'PILS', TRUE, FALSE, EMPTY

<定数> ::= <整数> | <実数> | <'文字列'> | TRUE | FALSE | EMPTY

<整数> ::= <無符号整数> | <符号> <無符号整数> |

<無符号整数> ::= <数字> {<数字>}

<符号> ::= + | -

\* PASCAL の set を文字列に拡張したもの。

\*\* 整数と実数のとりうる値の範囲は implement する PASCAL のそれと一致する。

<実数> ::= <無符号実数> | <符号> <無符号実数>

<無符号実数> ::= <無符号整数> · <無符号整数> | <無符号整数> · <無符号整数> E <整数> | <無符号整数> E <整数>

<文字列> ::= <文字> {<文字>}

<文字> ::= <英字> | <数字> | <特殊記号>

## 7. 式

式は代入文の右辺に書いたり、IF 文等の条件部に書いたり、パラメータとして書いたりする。式はオペランドと演算子からなる。演算子は演算の優先順位によって五つのグループに分ける。

優先順位 演算子のグループ

- 1 位 画像演算子 (#<演算子名>)
- 2 位 否定演算子 (NOT)
- 3 位 乗除算演算子 (\*, /, AND, MOD)
- 4 位 加減算演算子 (+, -, OR)
- 5 位 関係演算子 (=, <, >, <=, >=, =, <=, >=)

<式> ::= <単純式> <関係演算子> <単純式> | <単純式>

<関係演算子> ::= <= < > | = | > | < | > = | = > | <= | = <

<単純式> ::= <項> | <符号> <項> |

<単純式> <加減演算子> <項>

<項> ::= <因子> | <項> <乗除算演算子> <因子>

<因子> ::= <変数> | <画像要素名> <画像演算子部> | <関数名> (<パラメータリスト>) (<式>) |

NOT <因子> | <定数> | <名前> |

[ ] { } <名前> { , <名前> }

<加減演算子名> ::= + | - | OR

<乗除演算子名> ::= \* | / | AND | MOD

<画像演算子部> ::= # <演算子名> { # <演算子名> } | <空白>

<演算子名> ::= <名前> | <名前> ?

<名前> ::= <英字> {<英数字>}

### 7.1 画像演算子

画像演算子はオペランドが指す画像要素に作用して、その属性の一つをアクセスしたり、データ構造上

\* PILS の演算子は PASCAL のそれから DIV と IN を除いて画像演算子を加えたものである。

表-2 属性を求めるマクロ画像演算子

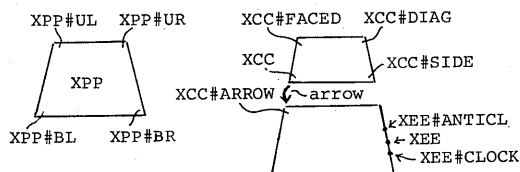
オペランドの型	結果の型	画像演算子
whole, unit part	実数	{#AREA, #HEIGHT, #WIDTH, #XG, #YG, #XMAX, #XMIN, #YMAX, #YMIN
whole, unit	実数	#PERIM
pcorner, edge part	実数	#X, #Y
pcorner	論理	#HEAD?, #TAIL?
全ての型	論理	#BL?, #BR, #UL?, #UR? #EMPTY?, #EXIST?

表-3 親子兄弟関係にあ要素を求めるマクロ画像演算子

オペランドの型	結果の型	画像演算子
unit	part	#UNIT
part	pcorner	#BL, #BR, #UL, #UR
pcorner	part	#PART
pcorner	pcorner	#FACED, #SIDE, #DIAG, #ARROW
pcorner	edge	#EDGE
edge	edge	#CLOCK, #ANTICL

表-4 ミクロ輪郭用画像演算子

オペランドの型	結果の型	画像演算子
arc	実数	#X0, #Y0, #X1, #Y1, #DX, #DY, #VL, #AL
arc	整数	#T0, #T1, #DT0, #DT1, #VT
arc	論理	{#LINE?, #CONCAVE?, #CONVEX? #ZIGAZAG?, #BLANK?, #EXIST?, #EMPTY?
arc	arc	#NEXT, #LAST, #OUTER

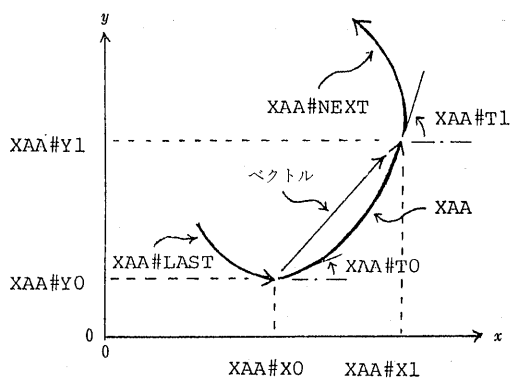


XPP, XCC, XEE を与えると, XPP#UL  
XCC#FACED, XEE#CLOCK 等は図のようになる。

図-7 マクロ輪郭用関係画像演算子

その画像要素と親子兄弟関係にある画像要素をアクセスしたりする。オペランドは画像演算子の左側に書く。表-2に属性を求めるマクロ画像演算子を、表-3に親子兄弟関係にある要素を求める画像演算子を示す。その働きを図-7に示す。表-4にミクロ画像演算子を、図-8にその働きを示す。

例. 2 XPP#BL#ARROW#PART#AREA  
XCC#FACED#ARROW#PART#HEAD?  
XUU#XG-XEE#X  
XAA#DT0>5



XAA#DX=XAA#X1-XAA#X0  
XAA#DY=XAA#Y1-XAA#Y0  
XAA#DT0=XAA#T0-XAA#LAST#T1  
XAA#DT1=XAA#NEXT#T0-XAA#T1  
が常に成り立つ

図-8 ミクロ輪郭用画像演算子

## 7.2 算術演算子

算術演算子の一覧表を表-5に示す。

表-5 算 術 演 算 子

分 類	演 算 子	作 用	オペランドの型	結 果 の 型
否 定	NOT	オペランドの否定	論 理	論 理
乗 除 算	*	{乗算 積集合 除算 モジュール演算 論理積	実数または整数*	実数または整数
	/		set	set
	MOD		実数または整数*	実数
	AND		整数	整数
		論理	論理	
加 減 算	+	{加算 和集合 減算 差の集合 論理和	実数または整数*	実数または整数
	-		set	set
			実数または整数*	実数または整数
	OR		set	set
		論理	論理	
比 較	=	等号, 一致	{すべての型 (含 set, term)  実数または 整数	{論理   実数または 整数
	< >	不等号, 不一致		
	>, <	}比較		
	>=, <=			
	=>, =<			

注) \* 二つのオペランドは同じ型でなければならない

表-6 PILS 標 準 関 数

関 数	結果の型	意 味
注 1 ABS(x), ABS(xii) ARCTAN(x) COS(x) EXP(x) EXP10(x) LN(x) LOG(x) SQRT(x) SIN(x) SQR(x), SQR(xii) ROUND(x) TRUNC(x)	注 2 実数 整数	x または xii の絶対値 $\tan^{-1}x$ $\cos x$ $e^x$ $10^x$ $\log_e x$ $\log x$ $\sqrt{x}$ $\sin x$ $x^2$ または $xii^2$ x の小数部四捨五入 x の小数部切捨て

注 1) x は実数, xii は整数型のパラメータ

2) パラメータと同じ型の結果

### 7.3 関 数

PILS の関数は PASCAL の標準関数から ORD と CHR, PRED, SUCC を除いて LOG を加えたものである。

## 8. 文

PILS プログラムは, MEASURE で始まり, 宣言部を書き次に実行部を書いて MEND で締めくくる。

<PILS プログラム>:: =

MEASURE <宣言部> <実行部> [<ラベル>:]  
MEND

<宣言部>:: = <宣言文> {<宣言文>}

<実行部>:: =

<ラベルつき実行文> {<ラベルつき実行文>}

<ラベルつき実行文>:: = [<ラベル>:]<実行文>

### 8.1 宣言文

<宣言文>:: =

<TERM 文> | <SET 文> | <FILE 文> | <OP  
DEF 文> | <ASSIGN 文>

**TERM 文** と **SET 文** は term 型と set 型の変数とそのとりうる値を宣言する。

<TERM 文>:: =

TERM<変数名>=(<名前>{, <名前>})

<SET 文>:: =

SET<変数名>=[<名前>{, <名前>}]

計測データの **FILE 文** はファイルの名前と, そのファイルの1レコードを構成する項目のならびを宣言する。項目は算術型(整数と実数, 文字列, 論理)と term 型, set 型の変数名で, その名前と型を表わす。ファイル名は実行 (RUN) 時に変更することができる。ファイルは順編成ファイルで, データは全て文字コードに変換して記録する。

<FILE 文>:: =

FILE <ファイル名>=<変数名>{, <変数名>}

**ASSIGN 文** は定数の名前とその値を定義する。定数の名前は式の中やパラメータの中に書くことができる。

<ASSIGN 文>:: =

ASSIGN<定数宣言部>{, <定数宣言部>}

<定数宣言部>:: =<定数の名前>=<定数>

PILS プログラムを実行 (RUN) すると、すぐに ASSIGN 文で宣言した定数の名前とその値がコンソールに表示される。値を修正したいときはその場で修正する。

例. 3 ASSIGN E=2.718, PI=3.15

上記の ASSIGN 文を含むプログラムを実行すると、そのリストがコンソールに表示される。このとき PI の値を 3.14 に修正したいときは、アンダラインで示したようにデータをコンソールから入力する。

E=2.71828)

PI= 3.14)

OPDEF 文 は新しい画像演算子を定義する

<OPDEF 文>:: =

OPDEF <結果を表わす引数>=<オペランドを  
わす引数>  
#<新画像  
演算子名>

[GLOBAL <グローバル変数>{, <グローバル  
変数>}<本体部>

<結果を表わす引数>:: =

<変数名>|<画像要素名>

<オペランドを表わす引数>:: =<画像要素名>

<グローバル変数>:: =

<変数名>|<画像要素名>|<定数名>

<本体部>:: =

<通常文>{<通常文>} OPEND | EXTERNAL  
本体部で新しい画像演算の処理本体を定義する。この画像演算の本体が別コンパイルしてあり、PILS ユーザライブラリに登録してあるときは、単に EXTERNAL と書けば良い。本体部で使う変数名や画像要素名は GLOBAL で宣言したものを除いて、本体部の範囲内でのみ有効で、OPDEF 文の外では無効である。ラベルの有効範囲についても同様である。本体部を書けるのは通常文だけで、画像入力文やファイル文は書けない。

例. 4 unit の面積が定数 PI=3.14 より大であるとき TRUE, 小であるとき FALSE を返す画像演算子 #BIG? を定義する。

ASSIGN PI=3.14

OPDEF X?=XUU#BIG?

GLOBAL PI

X?:=FALSE

IF XUU#AREA PI: X?:=TRUE:

OPEND

代入文は PASCAL と同様に 変数と式 の間に := を書く。IF 文は、IF <論理型式>: [<論理型式が TRUE のとき実行する文>]: [<論理型式が TRUE のとき分岐するラベル>] と書く (詳細は 8.2.3.4 参照)。

## 8.2 実行文

<実行文>:: =

<画像入力文>|<ファイル操作文>|<通常文>

### 8.2.1 画像入力文

画像入力文は画像を読取ってそのデータ構造を生成したり、その構造を変更する文からなる\*。

<画像入力文>:: =

<MACRO 文>|<REORG 文>|<GETARC  
文>|<REARC 文>|<DRIVE 文>

MACRO 文 FSS (flying spot scanner) または TV カメラを通してマクロ輪郭の edge を抽出し、それを構造化して最上位の画像要素、すなわち whole を引渡す文である。

<MACRO>:: =

MACRO <whole 型要素名>

[AXIS <パラメータリスト>]

[ZONE <パラメータリスト>]

[FILTER <パラメータリスト>]

[ORGNIIZE <パラメータリスト>]

<パラメータリスト>:: =

<算術型の式>{, <算術型の式>}

パラメータリストに書くパラメータの意味とその既定値は、入力装置によって若干異なる。詳細は付録 1 に示す。パラメータリストは途中省略して良い。省略すると既定値が用いられる。

REORG 文 は、CUT 文が定義した線分でマクロ輪郭の segment を分割し、マクロ輪郭を再構成する文である。REORG-REVERSAL と書くと、マクロ輪郭を図地反転、すなわち今までの segment を消して替りに segment と segment の狭間 (背景の部分) を新しい segment として、新しいマクロ輪郭を再構成する。

<REORG 文>:: =

REORG [-REVERSAL] <whole 型要素名>

[BY <パラメータリスト>]

\* 画像入力処理の本体は PILS システムサブルーチンライブラリにある。パラメータの意味やその既定値はライブラリを変えることによって変更できる。



パラメータリストの意味は MACRO 文の ORGANIZE のそれと同じである。

whole 型画像要素名で分割前と分割後のマクロ輪郭を指す。REORG 文を実行した後では、分割前のマクロ輪郭は消えてしまい、それにつけた画像要素名は意味を持たなくなることに注意しなければならない。

**GETARC 文** は、図-4 (a) のようにマクロ輪郭の edge をガイドにして、時計回りに (TO の場合) または反時計回りに (ANTICL の場合) 辺縁を走査して 3 本以内のマイクロ輪郭線を抽出する。各マイクロ輪郭線を尖点や変曲点で分割して arc 列を作る。

```
<GETARC 文>:: =
GETARC [-CRT | -PLOT | -FILE]
<arc 型要素名>{, <arc 型要素名>}
[ZONE <パラメータリスト>]
[BY <パラメータリスト>]
TRACE <unit 型式>|
<pcorner 型式> <方向指示詞> <pcorner 型式>
```

<方向指示詞>:: = TO | ANTICL

arc 型要素名は各 arc 列の先頭要素につける名前で、内側、真中、外側輪郭線の arc 型要素名の順に書く。真中と外側のそれは省略して良い。TRACE の後の式はガイドとなる unit か、図-4 のようにガイドの出発点となる pcorner を示し、方向指示詞の後の式はガイドの終点となる pcorner を示す。ZONE のパラメータリストはマイクロ輪郭抽出のパラメータを示し、BY のそれは arc 列変換のパラメータを示す。詳細は付録 1 に示す。

**REARC 文** はマイクロ輪郭の arc 列の分割をやり直す。BY のパラメータの意味は GETARC 文のそれと同じである。arc 型要素名は arc 列の先頭要素を指さなければならない。

```
<REARC 文>:: =
REARC <arc 型要素名> BY <パラメータリスト>
```

**DRIVE 文** は、画像入力装置上の対象を移動する。画像入力装置の種類によってパラメータの意味が異なる。詳細は付録 1 参照

```
<DRIVE 文>:: =
DRIVE [<パラメータリスト>]
```

### 8.2.2 ファイル操作文

ファイル操作文は、宣言文 FILE で定義した計測データのファイルまたはマクロ輪郭の segment をディスクファイルに書いたり、読込んだりする。

```
<ファイル操作文>:: =
<REWRITE 文>|<WRITE 文>|<WRITESEG-
EGM 文>|<RESET 文>|<READ 文>|
<READSEGM 文>|<CLOSE 文>|
```

**REWRITE 文** は新しいファイルを書込む準備をする。

```
<REWRITE 文>:: =
```

```
REWRITE <ファイル名>
```

ファイル名は、計測データの場合は FILE 文で宣言したファイル名を書く。マクロ輪郭のファイルの場合は、新しいファイルの名前をここで宣言する。このファイル名は、REWRITE 文の実行時にコンソールに表示され、その場で変更することができる。

例. 5 実行時修正の例

```
REWRITE FILE NAME=AUG25=AUG26
```

```
REWRITE FILE NAME=NEWFILE
```

最初の文は AUG25 を AUG26 に変更し、次の文は無変更。

**WRITE 文** は計測データをファイルに書込む。

```
<WRITE 文>:: =WRITE <ファイル名>
```

ファイル名は FILE 文と REWRITE 文で書いたものでなければならない。

**WRITESEGMENT 文** は、whole 型要素名が指すマクロ輪郭を segment レベルでファイルに書込む。

```
<WRITESEGMENT 文>:: =
```

```
WRITESEGMENT <whole 型要素名> TO <ファイル名>
```

ファイル名は REWRITE 文で書いたものである。

**RESET 文** はディスクにあるファイルを読み取る準備をする。ファイル名については REWRITE 文と同様である。

```
<RESET 文>:: =RESET <ファイル名>
```

**READ 文** は計測データをファイルから読む

```
<READ 文>:: =READ <ファイル名>
```

**READSEGMENT 文** は、ディスク上のファイルから segment レベルのマクロ輪郭を読み取り、それに whole 型要素名をつける。この文の後に必ず REORG 文を書いて、マクロ輪郭の構造化を行わなければならない。

```
<READSEGMENT 文>:: =READSEGMENT
```

```
<whole 型要素名> FROM <ファイル名>
```

**CLOSE 文** はファイルを閉じる。特にファイルの書込みの場合はこれを省くと新しいファイルが完成しない。

```
<CLOSE 文>:: =CLOSE <ファイル名>
```

例. 6

```
MEASURE
```

```
FILE AUG25=FILMII, WAREA
```

```
REWRITE AUG25
```

```
REWRITE AUG25M
```

```
FILMII: =1
```

```
100: MACRO XWW
```

```
WAREA: =XWW#AREA
```

```

WRITE AUG25
WRITESEG XWW TO AUG25M
DRIVE 1
IF FILMII=10: FILMII:=FILMII+1: 100
CLOSE AUG25
CLOSE AUG25M
RESET AUG25
RESET AUG25M
ERASE [*erase CRT*]
200: READ AUG25
CRT FILMII, WAREA
  [*write FILMII and WAREA on CRT*]
READSEG XWW FROM AUG25M
REORG XWW
CRT XWW
  [*draw macro-contour of XWW on CRT*]
IF FILMII=10: FILMII:=FILMII+1: 200
CLOSE AUG25
CLOSE AUG25M
MEND

```

### 8.2.3 通常文

OPDEF 文の中や探索文の中に書ける文を通常文と言う。

<通常文>::=  
 <探索文>|<記憶操作文>|<グラフィックス文>|<代入文等>

#### 8.2.3.1 探索文

探索文は、マクロ輪郭を探索する PARASCAN 文および SEQSCAN 文と、ミクロ輪郭を探索する SKIP 文からなる。

<探索文>::=  
 <PARASCAN 文>|<SEQSCAN 文>|<SKIP 文>

**PARASCAN 文** は、探索範囲の中の画像要素を全て調べあげて、目的とする画像要素を捜出す文である。

<PARASCAN 文>::=  
 PARASCAN  
 SCOPE <探索範囲を表わす画像要素名>  
 COMPONENT <探索単位の画像要素名>  
 <PARASCAN 処理部>  
 PARAEND

<探索の範囲を表わす画像要素名>::=  
 <whole 型要素名>|<unit 型要素名>

<探索の単位となる画像要素名>::=  
 <unit 型要素名>|<part 型要素名>

<PARASCAN 処理部>::=  
 <通常文>{<通常文>}|  
 <TARGET 部><処理部>

表-7 探索範囲と探索単位の画像要素型

探索範囲の画像要素	探索単位の画像要素
whole 型要素名	unit 型要素名
unit 型要素名	part 型要素名

<TARGET 部>::=  
 <TARGET 文>{<TARGET 文>}  
 <TARGET 文>::=  
 TARGET-<ラベル>{,<ラベル>}<論理型の式>  
 <ラベル>::=1|2|3|4|5|6|7|8|9  
 <処理部>::=<処理文>{<処理文>}  
 <処理文>::=  
 <ラベル>{,<ラベル>}:<通常文>{<通常文>}|  
 ELSE:<通常文>{<通常文>}

SCOPE と COMPONENT に書ける画像要素名のタイプは、表-7 に示したものに限る。

SCOPE の中の COMPONENT を全て処理の対象としたいときは、PARASCAN 処理部にその処理の手順を記述する通常文を幾つか書けば良い。

例. 7 XUU の中に含まれる part の数を数える。

```

PARTNOII:=0
PARASCAN
  SCOPE XUU
  PARTNOII:=PARTNOII+1
PARAEND

```

捜出したい目的画像要素が幾つかあるときは、それらにラベルをつける。各目的画像要素毎に TARGET 文と処理文を書いて、それが満足すべき論理型の式と、それが見つかったときする処理の手順を示す。TARGET 文と処理文には目的画像要素のラベルを書いておのおのを区別する。

例. 8 XWW の中に含まれる unit の中で、周辺長/面積が  $6\pi(18.85)$  より大なる unit の数とそれ以下の unit の数を求める。

```

LESSII:=0
GREATII:=0
PARASCAN
  SCOPE XWW
  COMPONENT XUU
  TARGET-1 XUU#PERIM>=XUU#AREA*

```

18.85

```

1: GREATII:=GREATII+1
ELSE: LESSII:=LESSII+1
PARAEND

```

目的画像要素 i と j がともに満たすべき条件を有するとき、TARGET 部と処理は次のように書けるであろう。

TARGET-i <論理型式 1> AND <論理型式 2>  
 TARGET-j <論理型式 3> AND <論理型式 2>  
 i: <処理文 1>  
 j: <処理文 2>

上の文は次のように書いても同じ効果を有する。

TARGET-i <論理型式 1>  
 TARGET-j <論理型式 3>  
 TARGET-i,j <論理型式 2>  
 i: <処理文 1>  
 j: <処理文 2>

逆に、目的画像要素 i と j の処理部に共通な処理文 3 があるとき

TARGET-i <論理型式 1>  
 TARGET-j <論理型式 2>  
 i: <処理文 1>  
 j: <処理文 2>  
 i,j: <処理文 3>

と書いて良い。

**SEQSCAN 文**は、ある画像要素を出発点にしてある順序に従って画像要素をたどり、探索継続条件を満たす間、または、探索中止条件が成立するまで、目的画像要素の探索を続ける文である。

<SEQSCAN 文>:: =

SEQSCAN  
 FROM <出発点部>  
 [WHILE <探索継続条件>]  
 <TARGET 部> <処理部>  
 STEP <探索順序部>  
 [UNTIL <探索中止条件>]  
 SEQEND

<出発点部>:: = <通常文> { <通常文> }

<探索順序部>:: = <通常文> { <通常文> }

<探索継続条件>:: = <論理型の式>

<探索中止条件>:: = <論理型の式>

TARGET 部と処理部の書き方は PARASCAN 文に全く同じである。

SEQSCAN-TRM と書くと、目的画像要素が一つでも見つかり探索を中途打ち切り (terminate) する。探索継続条件と探索中止条件は両方書いても良いが、両方書いてはならない。

例. 9 part BASEPP の左上かどを出発点にしてマクロ 輪郭を時計回りにたどり、最初に出合った head part に HEADPP という名をつける。

```
SEQSCAN-TRM
  FROM XPP: =BASEPP
    HEADPP: =EMPTY
  TARGET-1 XPP#HEAD?
  1: HEADPP: =XPP
  STEP XPP: =XPP#UL#ARROW#PART
```

UNTIL XPP=BASEPP  
 SEQEND

一般に、出発点部と探索順序部には被探索画像要素への代入文を書く。

**SKIP 文**は、ある arc を出発点にして順方向に arc 列をたどり、目的画像要素を見つける文である。

<SKIP 文>:: = SKIP

<arc 型要素名>: = <出発点となる arc の式>

TARGET <目的画像要素の条件> [EXIT <ラベル>]

<目的画像要素の条件>:: = <論理型の式>

目的画像要素が見つかり探索は中止し、arc 要素名はそれを指すようになる。arc 列の中に目的画像要素が一つもないときは、探索は終了する。EXIT <ラベル> を書くと、列の終わりで探索終了したあとに、このラベルの文に分岐する。

例 BGNA A を出発点にして arc 列を探索し、ジグザグ状 arc を見つけ、なければ 300 へ分岐する。

```
SKIP XAA: =BGNA A
  TARGET XAA#ZIGZAG?
  EXIT 300
```

#### 8.2.3.2 連想記憶操作文

画像要素とそれに与えた評価値の対の系列を記憶するところを **連想記憶** と言う。評価値は整数 または 実数、10 字以内の文字列である。一つの連想記憶に記憶できる系列の長さはハードウェアのメモリサイズによって規制される。連想記憶は全部で 10 組あり、MM-0, MM-1, ..., MM-9 という名がついている。

<連想記憶名>:: = MM- <整数型式>

**CLEAR 文** は連想記憶を空にする。

<CLEAR 文>:: =

CLEAR <連想記憶名> {, <連想記憶名> }

**PUT 文** は画像要素と評価値の対を連想記憶にしまう。

<PUT 文>:: =

PUT <画像要素の式>, <評価値の式>

INTO <連想記憶名>

連想記憶を CLEAR しないまま、同一画像要素と同じ型の評価値 (ただし整数型か実数型) を PUT すると、最初に PUT した画像要素の評価値に、つぎつぎと新たな評価値が加算される。

例. 10

```
PUT XPP, 1 INTO MM-0
PUT XPP, 3.14 INTO MM-0
PUT XPP, 2 INTO MM-0
```

上の 3 行は、下の 2 行と同じ効果を有する。

```
PUT XPP, 3 INTO MM-0
PUT XPP, 3.14 INTO MM-0
```

TAKE 文は、TARGET 部に書いた論理型式を満

たす画像要素と評価値の対を取出す。該当する対が幾つもあるときは、早く PUT したもののから順に取出す。

<TAKE 文>:: =

<画像要素名>[, <変数名>] [<TARGET 部>] FROM <連想記憶名> [EXIT <ラベル>]  
<TARGET 部>:: = TARGET <論理型式>

変数名は取出した評価値(整数型か実数型か文字列型)をしまつ変数の名前である。該当する対は、結局画像要素のもつ型と変数名のもつ型、および TARGET 部の論理型式の3条件を満たさなければならない。変数名と TARGET 部は省略できる。省略するとその条件は取出すべき対の条件からはずされる。該当する対を取出した後では、その対は連想記憶に存在しなくなる。EXIT <ラベル> を書くと、該当する対が一つも見つからないときラベルの指す文へ分岐する。

例. 11

TAKE XU, XSS

TARGET (XU#AREA>10.0)

AND (XSS=' TRIANGLE')

FROM MM-0 EXIT 200

TAKE XU, XSS FROM MM-0

TAKE XU FROM MM-0

TAKEMAX 文は、連想記憶の中からその評価値が最大の画像要素と評価値の対を取出す。

<TAKEMAX>:: =

TAKEMAX <TAKE 本体部>

<TAKE 本体部>:: = <画像要素名>[, <変数名>]

FROM <連想記憶名> [EXIT<ラベル>]

TAKEMIN 文は、連想記憶の中からその評価値が最小の画像要素と評価値の対を取出す。

<TAKEMIN>:: =

TAKEMIN <TAKE 本体部>

例. 12 XWW の中から、面積が3.14に最も近い unit を見つけ、OPTUU という名をつける。

CLEAR MM-0

PARASCAN

SCOPE XWW

COMPONENT XU

PUT XU, ABS (XU#AREA-3.14)

INTOMM-0

PARAEND

TAKEMIN OPTUU FROM MM-0

TAKEPIC 文は、評価値を与えて、その評価値と対をなす画像要素を取出す。

<TAKEPIC 文>:: =

TAKEPIC <画像要素名>, <取出すべき評価値の式>

FROM <連想記憶名> [EXIT <ラベル>]

TAKEVAL 文は、画像要素を与えてそれと対をなす評価値を取出す。

<TAKEVAL 文>:: =

TAKEVAL <取出すべき画像要素の式>, <変数名>

FROM <連想記憶名> [EXIT <ラベル>]

例. 13 unit XU が指すマクロ輪郭に孔があいているか否かを調べる。まず, XU の左下隅の part BLPP を見つけ, BLPP のかど UL を出発点にしてマクロ輪郭を時計回りにたどり, pcorner を通るたび毎にその所属する part に2点を与える。たどり終わった後で、孔がなければ全ての part は4点で、孔があれば2点の part が存在する。

CLEAR MM-0, MM-1

PARASCAN

SCOPE XU

COMPONENT XPP

PUT XPP, XPP#XG+XPP#YG INTO MM-0

PARAEND

TAKEMIN BLPP FROM MM-0

SEQSCAN

FROM X0CC: =BLPP#BL

XCC: =X0CC

TARGET-1 TRUE

1: PUT XCC#PART, 2 INTO MM-1

STEP XCC: =XCC#FACED#ARROW

UNTIL XCC=X0CC#FACED#ARROW

SEQEND

TAKEPIC XPP, 2 FROM MM-1 EXIT 100

CRT 'HOLE FOUND'

GO 200

100: CRT 'NOT FOUND'

8.2.3.3 グラフィック文

抽出したマクロ輪郭やマイクロ輪郭、算術型のデータを CRT ディスプレまたは XY プロッタに描く文である。XY プロッタに描くときは、同時に CRT ディスプレにも呈示され、プロッタ上の図形を早目に確認することができる。

<グラフィックス文>:: =

<ERASE 文>|<FEED 文>|<CURSOR 文>|

<PENUP 文>|<CRT 文>|<PLOT 文>|

<KEY 文>

ERASE 文は、CRT ディスプレの全画面を消去する。

<ERASE 文>:: =ERASE

FEED 文は、XY プロッタを1画面分紙送りするとともに CRT ディスプレの全画面を消去する。

<FEED 文>:: =FEED

表-8 KEY 文の入力型式

入力データの型	入 力 型 式
整 数	例 1981, -25
実 数	例 3.14, 0.314E+1, -0.8
文 字 列	例 YAMANASHI
論 理	TRUE または FALSE
set	例 [ALPHA, BETA], [ ]
term	例 NORMAL

**CURSOR 文** は, CRT ディスプレ上のカーソルの位置を決める。カーソルは CRT 文で算術データを書く際の先頭文字の位置を与える。

<CURSOR 文>::=CURSOR <位置>

<位置>::=<点座標>|<edge タイプの式>

<点座標>::=<整数型の式>, <整数型の式>

**PENUP 文** は, ペン先きを上げたまま指定した位置までペンを移動する。このとき CRT のカーソルも一緒に移動する。

<PENUP 文>::=PENUP <位置>

CRT 文は, whole, unit part, arc 等の画像要素や, 実数, 整数, 文字列, 論理, term, set 等のタイプの算術データを CRT ディスプレ上に書く。

<CRT 文>::=

CRT-SCALE <実数型式>|

CRT <描画データ>{, <描画データ>}

<描画データ>::=

<whole 型式>|<unit 型式>|

<part 型式>|<arc 型式>|<整数型式>|

<実数型式>|<文字列型式>|<term 型式>|

<set 型式>|/

CRT-SCALE は CRT に描くときの倍率を設定する。この倍率は次の CRT-SCALE 文がくるまで有効である。指定がないときは倍率は 1.0 として処理される。

算術データはカーソルの指す位置から書き始める。  
/は改行を意味する。

例. 14 CRT ディスプレに XUU の指すマクロ輪郭を描き, 重心に文字 X を描く。

CRT XUU

CURSOR (XUU#XG, XUU#YG)

CRT 'X'

**PLOT 文** は, XY プロットと CRT ディスプレに算術データを描く。

<PLOT 文>::=

PLOT-SCALE <実数型式>

PLOT-FILE <ファイル名>

PLOT <描画データ>{, <描画データ>}

PLOT-FILE と書くと以後の描画データは全てファイルに作られる。FEED 文でファイルは閉じる。

**KEY 文** は, コンソール端末から算術データを入力する。入力するときの型式を表-8に示す。

<KEY 文>::=

KEY <入力データ>{, <入力データ>}

<入力データ>::=

<実数型変数>|<整数型変数>|<文字列型変数>|

<論理型変数>|<term 変数>|<set 変数>

term, set タイプのデータは, KEY 文実行時に, その値が TERM 文, SET 文で定義した値の集合の中にあるか否かチェックされる。

8.2.3.4 代入文等

<代入文等>::=

<代入文>|<IF 文>|<GO 文>|<CUT 文>

**代入文** は, 算術変数や term タイプ変数, set タイプ変数, ファイルの項目画像要素名に式の値を代入する。

<代入文>::=<名前>:=<式>

<名前>::=<変数名>|<画像要素名>

式の左辺と右辺の型は次の例外を除いて一致しなければならない。

1) 変数が実数型で式が整数型は許す。このとき式の値は自動的に実数型に変換され代入される。

2) 全てのタイプの画像要素名に定数 EMPTY が代入できる。

例. 15

MEASURE

TERM BLOODTYP=(A, B, AB, O)

SET FAMILY=[WIFE, SON, DAUGHTER]

FILE ADAM=NOII, BOODTYP, FAMILY

NOII:=2

BLOODTYP:=A

FAMILY=[WIFE]

.....

MEND

**IF 文** は, 条件が成立したら実行文群を実行し, その後ラベルの指す文へ分岐する。ラベルは省略することができる。ただしコロン(:)は省略できない。省略すると条件の成立不成立にかかわらず IF 文の次にある文を実行する。

<IF 文>::=

IF <条件>:[<通常文群>]: [<ラベル>]

<条件>::=<論理型の式>

<通常文群>::=<通常文>{<通常文>}

GO 文はラベルの指す文へ無条件に分岐する。

<GO 文>::=GO <ラベル>

CUT 文は, マクロ輪郭を segment レベルで分割するさいの分割線を定義する。CUT 文はただ分割線を定義するだけで, 分割の操作は REORG 文による。分割線は REORG 文を実行した後自動的に消されて

しまう。

<CUT>::=<位置>, <位置>

<位置>の書換え規則は CURSOR 文に同じである。

例. 16 XUU を重心を通る 垂線 で左右に分割する。その XUU の属する whole を XWW とする。

CUT (XUU#XG, XUU#YMAX), (XUU#XG, XUU#YMIN)

REORG XWW

分割線と交差する segment が分割の対象となり、分割線の延長線上の segment は分割の対象とならない。

## 9. む す び

PILS は、最初 PIL-1 という名で、胃 X 線写真の輪郭抽出と診断の処理プロセスをスマートに記述することを目的として設計した。途中で対象がシルエットで表わされるものならば何でも適用できることに気づき、機能の拡張を行った。一方、コンパイラの implement の結果、プログラムサイズが大き過ぎて 32kW のミニコンにはそのままでは入らないことがわかった。そこで、コンパイラシステムをオーバーレイするために文法の一部を変更した。上記の変更を行って PIL-1 を PILS と改名した。

PILS の言語設計は第一の著者が、コンパイラの設計と製作は第二の著者が、システムサブルーチンの開発は第三の著者が担当した。

この開発に際し有意義な助言をいただいた、有沢誠本学助教授、伊藤誠本学助教授、ハードウェアシステム開発を担当した二木弘枝官に感謝する。

## 付 録 1

MACRO 文や GETARC 文 DRIVE 文等のパラメータリストの意味や既定値は、PILS コンパイラと独立しており、PILS システムサブルーチンを交換することによって変更できる。

MACRO 文の標準パラメータリストを表-9に示す。図-8に画像入力領域を示し、AXIS と ZONE のパラメータの意味を示す。

GETARC 文の標準パラメータリストを表-10に示す。図-9に示すように、ミクロ輪郭は対象内点濃淡値  $f$  をいき値とする変動いき値法で抽出する。

DRIVE 文は、FSS の場合 35mm フィルム上の原画像を正方向または逆方向にコマ送りし、その後モニタ TV 上に原画像を一定時間映写する。パラメータの意味を表-11に示す。パラメータリストを省略するとコマ送りはオペレータ委せとなる。オペレータは表-12のキーを押すことによって、コマ送りと映写時間を制御する。

## 付 録 2

PILS V-1 システムは、1) 64KB のメモリを有す

表-9 MACRO 文 の パ ラ メ ー タ

パラメータ	記号とその意味	既定値	
		FSS	TV
AXIS	i1 x0, 図-8 参照	128	0
	i2 y0,       〃	128	0
	i3 $\theta$ ,       〃	0	0
	x1 scale*	1.0	1.0
ZONE	i1 lx, 図-8 参照	768	256
	i2 ly,       〃	768	256
	i3 xintv,   〃	8	1
	i4 yintv,   〃	24	3
	i5 dw,       〃	4	1
	i6 yfirst**	500	0
FILTER	i1 bias segment 抽出のいき値	10	1
	i2 segml, 短 segment 除去いき値	16	5
	i3 dx, 濃淡値差分の間隔	5	/
	i4 df. エッジモード判定のいき値	1	/
ORGANIZE	i1 Dmode デバッキングモード***	0	0
	i2 wlimit 構造化の幅のいき値	80	80
	i3 climit 構造化の中心ずれのいき値	80	80
	i4 w 構造化の係数	1	1

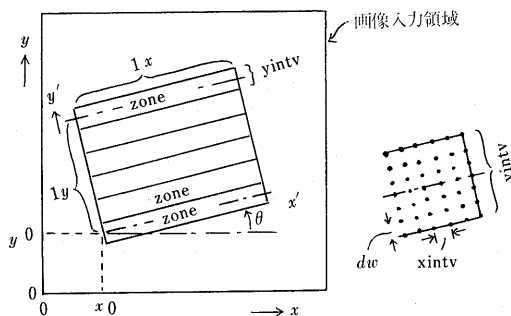
\* 長さ面積等の属性の単位は装置固有の単位のままではなくて scale 倍して取出される。

\*\* 濃淡いき値決定のため最初に走査する ZONE の y 座標

\*\*\* 0: 何も表示しない。1: マクロ輪郭を CRT ディスプレに表示する。2: unit と part の数をコンソールに表示する。

表-10 GET ARC 文のパラメータ

パラメータ	記号とその意味
ZONE	i1 b, バイアスレベル, 図-9 参照
	i2 $\delta$ , レベル差
	i3 win, 内側マイクロ走査幅
	i4 wout, 外側マイクロ走査幅
BY	i1 $\kappa$ , 近似ベクトル長
	i2 $\nu$ , 近似ベクトルステップ間隔
	i3 $\theta$ , 角度分割いき値 (度)
	i4 nz, ジグザクの下限
	i5 lb, BLANK 状 arc の長さの下限
	x1 d, CONCAVE いき値



画像入力領域: FSS では  $0 \leq x, y \leq 1023$

TV では  $0 \leq x, y \leq 255$

$x0, y0$ : 最下位 zone の左端の座標

$1x$ : zone の長さ

$1y$ : 最上位と最下位の zone の中心間の長さ

$xintv, dw$ :  $x', y'$  方向サンプリング間隔

$yintv$ : zone の幅

$\theta$ : 座標回転角

図-9 マクロ輪隔抽出のパラメータ

るミニコンピュータで動作すること, 2) さまざまな種類の画像入出力装置を使えること, 3) 画像入出力装置を替えても PILS システムの変更がわずかで済むこと, 3) 処理速度が速いこと, 等を考慮して幾つかのプロトコルを定めた。

機器構成は, PASCAL コンパイラを有するセントラルプロセッサと, 画像入出力装置を有するフロントエンドプロセッサからなる複合ミニコンピュータシステムとし, 二つのプロセッサ間は GP-IB (IEEE Standard 488-1975) または16ビット 並列通信インタフェースで結ぶものとする。

PILS システムの範囲は, セントラルプロセッサとインタフェースまでであり, フロントエンドプロセッサのシステムは, 画像入出力装置の仕様に合わせて別個に作成するものとする。インタフェース上のメッセージのプロトコルは次のとおりである。

1) セントラルプロセッサからフロントエンドプロ

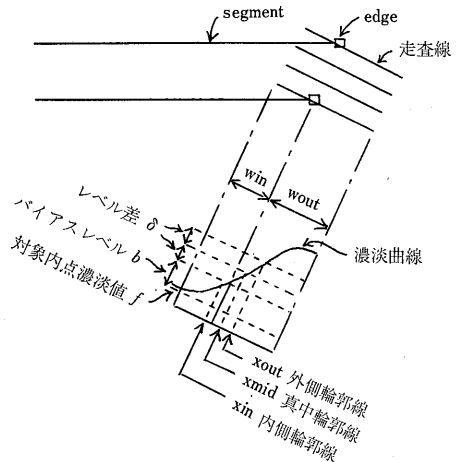


図-10 ミクロ輪郭抽出法

表-11 DRIVE 文のパラメータ

パラメータ	意味
i1	コマ送り数, 負のとき逆方向送り
i2	微動コマ送り数,
i3	映写時間, 単位秒

\* 8 微動コマ送りで 1 コマ送りに対応する。

表-12 キー操作によるコマ送り

キ	ー	コ マ 送 り
(↑), (→) または $F_1$		1 コマ正方向送り
$F_2$		1/8 コマ正方向送り
(↓), (←) または $F_3$		1 コマ逆方向送り
$F_4$		1/8 コマ逆方向送り
O		コマ送りと映写終了

Subr	N	Data
------	---	------

Subr: サブルーチン名, 4 文字 ASCII コード

N: メッセージ全体のワード数

Data: バイナリデータ

GP-IB の場合 N は無意味で終わりに EOI 信号を出す。

図-11 インタフェースのプロトコル

セッサへサブルーチン名とパラメータまたはデータを送る。フロントエンドプロセッサからそのサブルーチンを実行した後, 必らずサブルーチン名と実行結果のデータがあればそのデータを送り返す。

2) メッセージの型式は 図-11 のとおりである。

1981年現在, セントラルプロセッサとして LSI-11/23, OS として RT-11 および RSX11M を, PASC-

AL コンパイラとして OMSI PASCAL V-1を使い、  
フロントエンドプロセッサとして OKITAC-4300 と  
Z80 を使っている。OKITA-4300 には FSS (フィル  
ム駆動装置と CRT モニタつき) と Tektronix 611

ディスプレイ, XY プロッタが接続しており, Z-80 に  
は TV カメラインフェースと CRT モニタおよび自  
由度2の回転卓(この上に対象となる物体をのせる)  
が接続されている。

## 付 録 3

PILS の文の定型とその簡単な説明を表13に示す。種類の欄の宣は宣言文を, フはファイル操作文を, 無  
印は通常文を表わす。表中の記号の意味を次に示す。

const: 定数	Er: 実数型式	s: 通常文
Ea: 算術型式	Es: set 型式	Xa: 算術型変数
Eaa: arc 型式	Et: term 型式	Xaa: arc 型変数
East: 算術型または set 型, term 型式	Euu: unit 型式	Xap: 算術型変数または画像要素名
Ecc: pcorner 型式	Eww: whole 型式	Xast: 算術型または set 型, term 型変数
Edisp: Eww または Euu, Epp, Eaa, East, /	Ewwuu: Eww または Euu	
Eee: edge 型式	E?: 論理型式	
Eii: 整数型式	fname: ファイル名	Xp: 画像要素名
Eloc: Eee または (Eii, Eii)	ident: 名前	Xs: set 型変数名
Ep: 画像型式	i: 目的画像要素の番号	Xt: term 型変数名
Epp: part 型式	l: ラベルを表わす整数	Xuupp: unit 型または part 型要素名
	opname: 画像要素名, 末尾は ? でも良い	Xww: whole 型要素名

表-13 PILS 文 一 覧 表

文 種類*	定 型	意 味
ASSIGN 宣	ASSIGN Xa=const {,Xa=const}	算術型の定数 Xa の値 const を宣言する。
CLEAR	CLEAR MM-Eii {,MM-Eii}	番号 Eii の連想記憶を帰零する。
CLOSE フ	CLOSE fname	ファイル fname を閉じる。
CRT	CRT Edisp {, Edisp}	描画データ Edisp を CRT ディスプレに描く。
CURT	CURT SCALE Er	以後の CRT の描画データを Er 倍して描く。
CURT	CURT Eloc	CRT ディスプレのカーソルを位置 Eloc に移動する。
CUT	CUT Eloc <sub>1</sub> , Eloc <sub>2</sub>	マクロ輪郭の分割線 Eloc <sub>1</sub> Eloc <sub>2</sub> を定義する。
DRIVE 入	DRIVE Ea {,Ea}	原画像を移動する。付録1参照
ERASE	ERASE	CRT ディスプレの画面を消す。
FEED	FEED	XY プロッタを1画面分紙送りする。
FILE フ	FILE fname=Xast {,Xast}	ファイル fname のレコードを構成する項目 Xast を 宣言する。
GETARC 入	GETARC[-CRT]-PLOT-FILE] Xaa{,Xaa} [ZONE Ea {,Ea}] [BY Ea {,Eii}] TRACE {Ecc, {TO ANTICL Ecc <sub>2</sub> Euu	Ecc <sub>1</sub> を出発点にして Ecc <sub>2</sub> までマクロ輪郭をたどり, 内側, 真中, 外側のミクロ輪郭線を抽出し, それら を arc 列に変換する。arc 列の先頭要素に Xaa と 言う名をつける。付録1参照。
GO	GO l	ラベル l の文へ分岐する。
IF	IF [E?]:[s {,s}]:[l]	E? が TRUE だったら [s{s}] を実行する。その後 l が書いてありかつ E? が TRUE のとき, ラベルの 文へ分岐する。
KEY	KEY Xast {,Xast}	鍵盤より算術型または term 型, set 型のデータを読 取り, Xast にしまう。
MACRO 入	MACRO Xww [AXIS Ea {,Ea}] [ZONE Ea {Ea}] [FILTER Ea {,Ea}] [ORGANIZE Ea {,Ea}]	画像入力装置よりマクロ輪郭を読み取り, 構造化して, その最上位の画像要素である whole に Xww と言 う名をつける。付録1参照
MEASURE 宣	MEASURE	PILS プログラムの始まりを宣言する。
MEND 宣	MEND	PILS プログラムの終わりを宣言する。
OPDEF 宣	OPDEF Xap=Xp#opname [GLOBAL Xap {,Xap}] [s {,s}] [EXTERNAL	Xp 型の画像要素をオペランドとし, Xap 型の結果を 得る新画像演算子 #opname を定義する。GLOBAL でグローバルな変数 Xap を宣言する。次に画像演 算の本体部 s{s} OPEND を定義する。ライブラ リにあるときは EXTERNAL と書く。



PARASCAN	PARASCAN SCOPE Ewwuu COMPONENT Xuupp TARGET-i <sub>1</sub> , i <sub>2</sub> , ... E? [TARGET-i <sub>2</sub> , i <sub>3</sub> , ... E?] i <sub>2</sub> , i <sub>4</sub> , ...: s {s} [i <sub>2</sub> , i <sub>5</sub> , ...: s {s}] [ELSE: s{s}] PARAEND PARASCAN SCOPE Ewwuu COMPONENT Xuupp s {s} PARAEND	探索範囲 Ewwuu 中の全ての構成要素 Xuupp を 走査して、もし、i <sub>2</sub> を添字とする全ての TARGET 条件を満たす目的画像要素 Xuupp が存在したら、 i <sub>2</sub> をラベルとする全ての通常文群 s{s} を実行す る。  探索範囲 Ewwuu 中の全ての構成要素 Xuupp に対 して通常文群 s{s} を実行する。
PENUP	PENUP Eloc	XY プロットのペン先きを上げたまま Eloc に移動す る。
PLOT	PLOT Edisp {, Edisp} PLOT-SCALE Er PLOT-FILE fname	描画データ Edisp を XY プロットに描く。 以後の FLOT の描画データを Er 倍して描く。 以後の PLOT の描画データをファイルにしまう。
PUT	PUT Ep, Ea INTO MM-Eii	画像要素 Ep と評価値 Ea の対を連想記憶 Eii にし まう。
READ	フ READ fname	ファイル fname から1レコード分読取る。
READS- EGM	フ READSEGM Xww FROM fname	ファイル fname より1画面分のマクロ輪郭を Xww に読取る。
REARC	REARC Xaa BY Ea {, Ea}	Xaa の指すミクロ輪郭の arc 分割をやり直す。
REORG	REORG[-REVERSAL] Xww [BY Ea {, Ea}]	Xww の指すマクロ輪郭を分割し構造化をやり直す。 REVERSAL と書くと、マクロ輪郭を図地反転して から再構成する。
RESET	フ RESET fname	ファイル fname から READ できるように、ファイ ルを開く。
REWRITE	フ REWRITE fname	ファイル fname へ WRITE できるように、ファイ ルを開く。
SEQSCAN	SEQSCAN [-TRM] FROM s {s} WHILE E? TARGET-i <sub>1</sub> , i <sub>2</sub> , ... E? [TARGET-i <sub>2</sub> , i <sub>3</sub> , ... E?] i <sub>2</sub> , i <sub>4</sub> , ...: s {s} [i <sub>2</sub> , i <sub>5</sub> , ...: s {s}][ELSE: s{s}] STEP s {s} UNTIL E? SEQEND	FROM で与えた画像要素を出発点にして、STEP で 与えた順序に従って画像要素をたどり、WHILE で 与えた探索継続条件を満たす間、または、UNTIL で与えた探索中止条件が成立するまで、目的画像要 素の探索を続ける。TARGET 部と処理部の書き方 は PARASCAN に同じである。SEQSCAN-TRM と書くと、目的画像要素が一つでも見つければ探索 を中止。
SET	宣 SET Xs {, Xs}=(ident {, ident})	set 型変数 Xs のとりうる値 ident の集合を宣言する。
SKIP	SKIP Xaa:=Eaa TARGET E? [EXIT I]	画像要素 Eaa を出発点にして arc 列をたどり、E? を満たす目的画像要素を見つけ Xaa にしまう。見 つからぬとき I へ。
TAKE	TAKE Xp [, Xa] [TARGET E?] FROM MM-Eii [EXIT I]	TARGET の条件を満たす画像要素と評価値の対を連 想記憶 Eii から取出し Xp と Xa にしまう。見つ からぬとき I へ分岐。
TAKEMAX	TAKEMAX Xp [, Xa] FROM MM-Eii [EXIT I]	評価値が最大の画像要素と評価値の対を取出し Xp と Xa にしまう。
TAKEMIN	TAKEMIN Xp [, Xa] FROM MM-Eii [EXIT I]	評価値が最小の画像要素と評価値の対を取出し Xp と Xa にしまう。
TAKEPIC	TAKEPIC Xp, Ea FROM MM-Eii [EXIT I]	評価値が Ea の画像要素を取出し Xp にしまう。
TAKEVAL	TAKEVAL Ep, Xa FROM MM-Eii [EXIT I]	画像要素 Ep の評価値を取出し Xa にしまう。
TERM	宣 TERM Xt (, Xt)=[ident {, ident}]	term 型変数 Xt のとりうる値 ident の集合を宣言す る。
WRITE	フ WRITE fname	ファイル fname の1レコードをデスクに書込む。
WRITE- SEGM	フ WRITETEGM Xww TO fname	ファイル fname に1画面分のマクロ輪郭 Xww を書 込む。

文 献

- 1) Mori: Introduction to Picture Interpretation

Language for Silhouette PILS, 山梨大学計算  
機科学科テクニカルレポート, CS81-022

- 2) 森, 赤羽: 画像解釈言語 PILS, 情報処理学会論  
文誌 (印刷中)