FORTRAN型フローチャート言語FFLの設計と試作*

廣瀬　健**・宇都宮公訓***
坂倉正純***・本間典一***

Abstract

The Flowchart Language FL which we developed is a powerful language to describe algorithms. It is completely defined when its ability is determined, which consists of functions and predicates. This paper concerns with the design and the implementation of FORTRAN-Oriented Flowchart Language FFL whose ability almost comes from FORTRAN language.

First, the specification and the use of FFL are described. Next, the FFL compiler and the Autocharter are discussed, which generate FORTRAN coding and FFL-flowchart respectively. Finally, a sample program, which is processed by the FFL system, is exhibited.

1. はじめに

計算機言語は，なるべくは，その言語特有の規則や計算機の特性にとらわれず，素直にアルゴリズムを表現できるものであることが望ましい．

アルゴリズムとは，そのアルゴリズムが適用される対象の分野によって定まる関数および述語の順序づけられた組と考えられる．これらの順序づけを表現するには，フローチャートの方法によるものが見やすいであろう．アルゴリズムをこのように表現する言語を考える．この言語は，使用する関数や述語の与えられ方によって異なるが，われわれは，定められた関数や述語の組を ability と呼び，与えられる ability によって1つの言語が定まるものと考える．

このような言語 FL (Flowchart Language) を考慮した理由は，最初に述べたことの他に，簡単なプログラムの等価性や同等性のチェックに有効であると思われること，さらに，他の言語の implementation などにも有用と考えられたことによる．事実，後述するオートチャータなどを用いることにより，最初に1度フローチャートを書きさえすれば，最終的なフローチャートやドキュメンテーションが自動的に得られること，また，フローチャートを他の言語，たとえばアセンブラ語でコーディングする労力を省けることなどは，かなりの省力化につながる．

FL を計算機言語として使用するためには，次のような方法をとる：

(i) FL の記述方法をある程度制限する．

(ii) (i)で定める方法で記述された FL-program (pFL) を，それに対応する記号列 (sFL) に変換する方法を与える．

(iii) sFL を適当な言語に変換するトランスレータを作成する．

このとき，一番問題になると思われるのは(ii)であるが，われわれがこれまでに行なった実験によれば，キーパンチャを高々数時間教育することにより，pFL を見ながら sFL をパンチすることが可能である．

本稿では，われわれが実験的に作成した FL について述べる．一般的な FL についての詳細は，1),2) などを参照されたい．

2. FFL による記述

ability を FOPTRAN 語に求めた FL，すなわち FORTRAN-oriented Flowchart Language FFL について述べる．この節では，まず，FFL を用いてどのようにコーディングするかを述べよう．

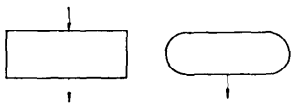
2.1 FFL プログラム

FFL によるコーディングの基本単位は句である．

* Design of FORTRAN-Oriented Flowchart Language FFL and its Implementation by Ken Hirose (Department of Mathematics, School of Science and Engineering, Waseda University), Kiminori Utsunomiya, Masayoshi Sakakura & Fumikazu Homma (Department of Electric Engineering, School of Science and Engineering, Waseda University).

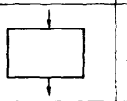
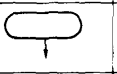
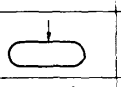
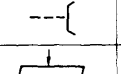
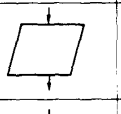
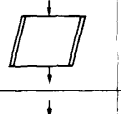
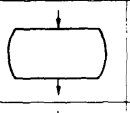
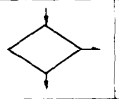
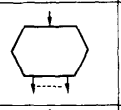
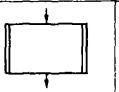
** 早稲田大学理工学部数学科
*** 早稲田大学理工学部電気工学科

句は FL 記号と単語からなる．FL 記号は句の機能を表わし，単語は機能を詳細に記述している．FL 記号の一覧表を表1に示す．図1は代入句と始端子句の例である．この図において，



は FL 記号であり，

表 1　FL 記号一覧表

FL 記号 pFL	FL 記号 sFL	名称	機能
	#	代入	代入が行なわれることを示す．
	%	始端子	文の入口を示す．
	;	終端子	文の出口を示す．
	@	注釈	注釈であることを示す．
	¥	入力	外部装置からデータを読むことを示す．
	Ç	出力	外部装置にデータを書き出すことを示す．
	!	宣言	変数の属性や値を宣言することを示す．
	?	判断	判断が行なわれることを示す．
	??	スイッチ	2方向以上への分岐が行なわれることを示す．
	*	コネクタ	無条件分岐，および，分岐の入口点を示す．
	"	コール	サブルーチンを呼ぶcallすることを示す．
な　し	.	エンド	文の終りを示す．
な　し	−	継続	直前のカードからの継続であることを示す．

A=A+3.2　　START

は単語である．

いくつかの句によって1つの文が構成される．文にはただ1つの始端子句と1つ以上の終端子句がなければならない．FFL プログラムは1つ以上の文からなる．FFL プログラムが2つ以上の文からなるときは，1つは主文とよばれ，残りはすべて副文と呼ばれる．FORTRAN の用語でいえば，文はプログラムセグメントにあたり，主文は主プログラム，副文は副プログラムに相当する．

2.2 代入句

代入句は，代入 FL 記号と FORTRAN 語で許された代入ステートメントからなる．1つの代入 FL 記号中に2つ以上の代入ステートメントをコンマで区切って書くことができる．図1の左の句は代入句の例である．

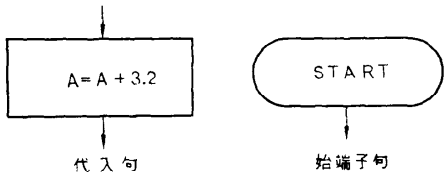


図 1　句　の　例

2.3 入力句，出力句

入力句，出力句は，それぞれ入力 FL 記号，出力 FL 記号と，コンマで区切られた書式付き入出力変数からなる．書式付き入出力変数とは，入出力変数と変換指定を組にしたものである．H変換，X変換，欄区切り記号は，それだけで書式付き入出力変数とみなす．配列の入出力の際の暗示的 DO は，括弧を用いて，たとえば

(B(I), I=1,00) (100 F 8.2)

のように書く．入出力ユニットの指定は，特に断らな

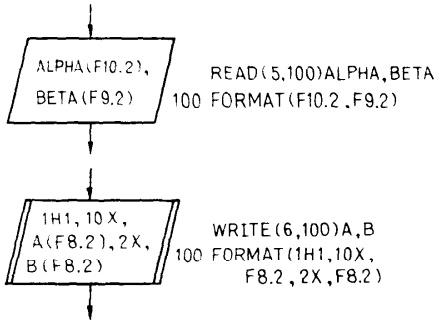


図 2　入出句，出力句の例

いかぎり，入力は5番，出力は6番とみなす．それ以外のユニットを指定したいときは，ユニット番号を括弧でくくって，書式付き入出力変数の最初につける．入力句，出力句の例と，それらのFORTRAN語による説明を図2に示す．

2.4 コネクタ句

コネクタ句は，コネクタFL記号とラベルからなり，無条件分岐，もしくは分岐先の入口点を表わす．ラベルは14個以下の英数字のストリングである．

他の句が続いている句をラベルコネクタ句と呼び，続かないコネクタ句をジャンプコネクタ句と呼ぶ．この2種類のコネクタ句を図3に示す．2個以上のコネクタ句を数珠繋ぎにして用いてはならない．

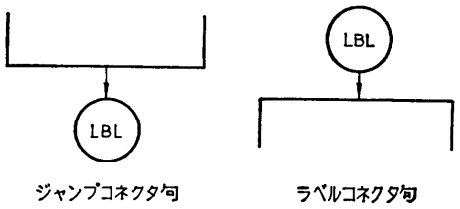


図 3 2種類のコネクタ句

2.5 スイッチ句

スイッチ句は，スイッチFL記号と，1つ以上の論理式からなる．スイッチ句の後には，論理式と同数かもしくはそれ以下の個数のジャンプコネクタ句が続く．スイッチ句中の論理式が順番にチェックされ，その値が真である最初の論理式と同じ番目のジャンプコネクタ句で分岐する．論理式の値がすべて偽であれば，コントロールはその次の句に移される．スイッチ句の例を図4に示す．この図の左側の例では，2番目の論理式が成立しているのでLBL2への分岐が生じる．ジャンプコネクタ句の数が論理式の数より少ない場合は，最後のジャンプコネクタ句がコピーされ使用される．

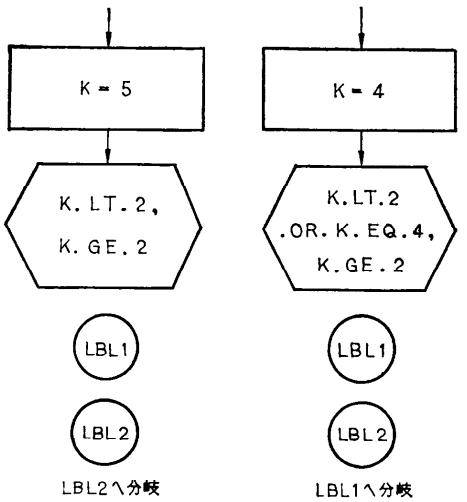


図 4 スイッチ句の例

スイッチ句の後に，変数型ラベルのジャンプコネクタ句を続けることにより，多方向への分岐を簡単に表記することができる．変数型ラベルは，

ラベル名（整数型変数）

の形をしており，スイッチ句中の第n番目の論理式がはじめて真であれば，

ラベル名（n）

のラベルへの無条件分岐が生じる．この種のジャンプコネクタ句を受けるラベルコネクタ句は，同じラベル名で，

ラベル名（自然数）

の形をしていなければならない．このようなラベルを定数型ラベルという．変数型ラベルは，スイッチ句の後のジャンプコネクタ句でしか用いることができないが，定数型ラベルは通常のラベルと同じように用いることができる．図5では，LBL(3)への分岐が用じる．

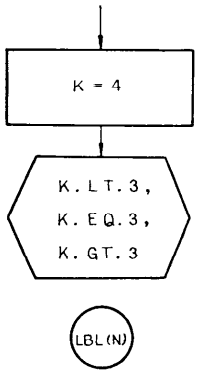


図 5 変数型ラベルの例

2.6 判断句

判断句は，判断FL記号と1つの論理式からなり，論理式の値の真偽によって2方向の分岐を生じさせる．

図6に示すような4種類の分岐が許され，右もしくは左に伸びた枝の先は必ずコネクタ句で受けなければならない．枝の傍のT, Fは分岐条件を示す論理式の値であり，それぞれ真（True），偽（False）を表わす．

FFLでは，FORTRAN語で使える論理作用素の他

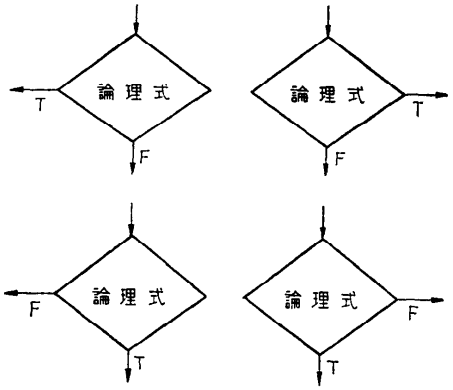


図 6 判断句の 4 つの型

に，implication .IM. と，logical coincidence .CO. を使うことができる．これらは次のように解釈される．

A .CO. B: A .AND. B .OR. .NOT. A .AND. .NOT. B

A .IM. B: .NOT. A .OR. B

2.7 宣言句

宣言句は，宣言 FL 記号と，いくつかの型宣言ステートメント，DIMENSION ステートメント，EQUIVALENCE ステートメント，DATA ステートメントからなり，各ステートメントで表わされた内容を宣言する．ただし，各ステートメントの先頭の予約語のすぐ後にセミコロンをつける．また，暗示的型宣言は FORTRAN に従う．図 7 に宣言句の例を示す．

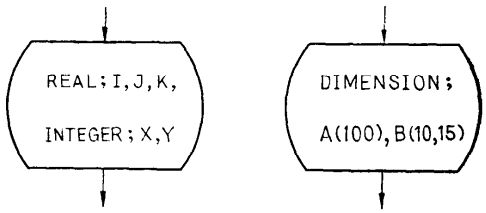


図 7 宣言句の例

FFL では，FORTRAN 語で許された型の変数の他に，ブール変数とスタック変数を使用することができる．これらは，BOOLEAN, STACK を用いて宣言する．

BOOLEAN は FORTAN の LOGICAL と同じである．

STACK 宣言された変数に対してはプッシュダウンスタックが用意され，代入記号の左辺で引用されると自動的にプッシュダウンされ，右辺で引用されると自動的にポップアップされる．スタック変数は，スタック中に格納されるデータの型に応じて，整数型か実数型かを型宣言しなければならない．スタック変数名の後に括弧でくくった整数型変数，もしくは正の整数定数をつけることによって，スタックの任意の位置のデータを引用することができる．この形式で引用した場合は，プッシュダウン，ポップアップは生じない．スタックの大きさは最大 300 である．また，同時に 4 本以上のスタックを活性化してはならない．

2.8 コール句

コール句は，コール FL 記号，サブルーチン名および引き数リストからなり，サブルーチンをコールするために用いられる．

2.9 始端子句

始端子句では，始端子 FL 記号と，次の 3 つのうちのいずれかを書き，文の先頭を表わす．

(i) 主文中では，任意の名前，

(ii) サブルーチン副文では，S，サブルーチン名 (引き数リスト)

(iii) 関数副文では，F，関数名 (引き数リスト)．

1 つの文には，ただ 1 つの始端子句が存在する．図 1 の右の句は始端子句の例である．

2.10 終端子句

終端子句では，終端子 FL 記号と，STOP, RETURN もしくは任意の名前を書く．以下のように解釈される．

(i) 主文では，終端子句中の単語は注釈とみなされ，実行を終了して，制御をシステムに返す．

(ii) 副文においては，STOP では制御をシステムに返すが，それ以外では呼び出したプログラムに返す．

1 つの文に 2 つ以上の終端子句があっても構わない．

2.11 注釈句

注釈句は，注釈 FL 記号に任意の文章を続けることによって構成する．とくに混乱を生じないときは，注釈 FL 記号を省略することができる．

2.12 コーディング

pFL 記号と単語からなる句をフローラインで結んでつくったフローチャート形式のプログラムを pFFL プログラムという．このような FFL のコーディングは，後に述べる記号列としての FFL，すなわち sFFL と区別するために pFFL によるコーディングと呼ばれる．コーディングシートは pFFL のためにだけ用意されており，sFFL のためのコーディングシートはとく

にない．その理由は次節にゆずる．

図10に示すようにコーディングシートは1ページ4レーンである．左から順に第1レーン，第2レーン，第3レーン，第4レーンと呼ばれ，あるページの第4レーンの後には，次のページの第1レーンが続くものと考える．コーディングは第1ページの第1レーンの上から下へ書きはじめ，レーン内に書き込めなくなったときは，コネクタを用いてあいているレーンに移る．判断句において，横方向の分岐先は必ず隣接するレーンでなければならない．かつ，コネクタ句で受けなければならない．FFL においては，あるレーンから別のレーンへ制御を移すときは，原則としてコネクタ句で受ける．

3. FFL システム

この節では，pFFL でコーディングされたプログラムがどのように処理されるかを述べる．

3.1 FFL コンパイラとオートチャータ

pFFL プログラムがどのように処理されるかを図8に示した．pFFL コーディングは計算機への入力のために，一定の規則にしたがってカードに穿孔される．このようにしてできたコーディングを sFFL という．pFFL コーディングと sFFL コーディングは1対1に対応しており，いずれの方向への変換も機械的に正しく行なうことができる．

FFL コンパイラは，シンタックスアナライザとジェネレータからなっており，sFFL コーディングを FORTRAN コーディングに変換すると同時に，いくつかの付帯リストを生成する．

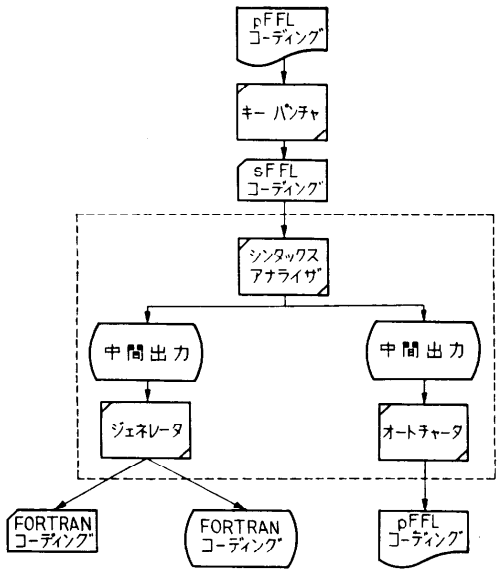


図 8 FFL システムによる処理

シンタックスアナライザは，sFFL コーディングのシンタックスをチェック，解析して，いくつかのテーブルを作る．ジェネレータは，これらのテーブルをもとにして FORTRAN コーディングを生成する．

一方，オートチャータは，シンタックスアナライザによって出力されたテーブルから pFFL コーディング，すなわちフローチャートを生成する．したがって，ユーザにとって，オートチャータの出力は pFFL コーディングのソースリストと考えるべきものであり，3) や 4) によるものとは本質的に異なる．

3.2 pFFL から sFFL への変換

pFFL から sFFL への変換は，カード穿孔時に，キーパンチャによって行なわれる．pFFL コーディングをカードに穿孔する手順を，図9に示したが，これはそのまま sFFL への変換手順になっている．

pFFL コーディングにはないが，sFFL コーディングでは文の終りを明示するためにエンド句を用いる．これはエンド FL 記号だけからなる句である．1つの

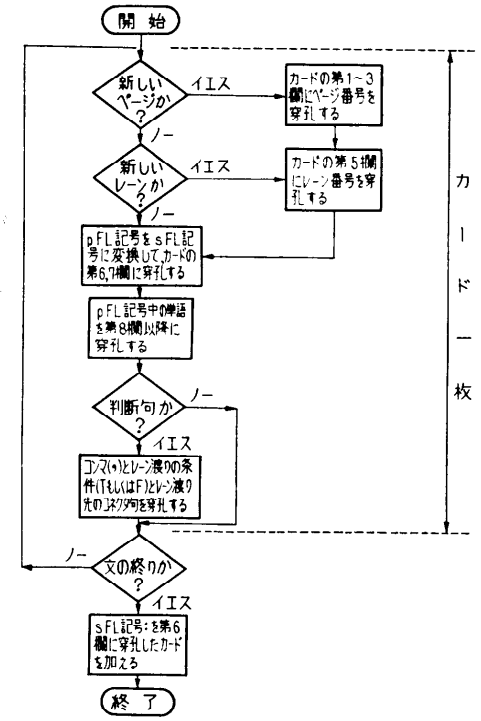


図 9 pFFL コーディングの穿孔手順

句が1枚のカードに穿孔しきれないときは，継続 FL 記号＝を用いることによって，10枚まで継続することができる．

図10の pFFL コーディングを sFFL コーディングに変換した結果が，図13のソースリストに見られる．

4. FFL システムの成分

この節では，シンタックスアナライザ，ジェネレータおよびオートチャータで，どのような処理が行なわれるかを述べる．

4.1 シンタックスアナライザ

シンタックスアナライザによる処理の結果は，単語

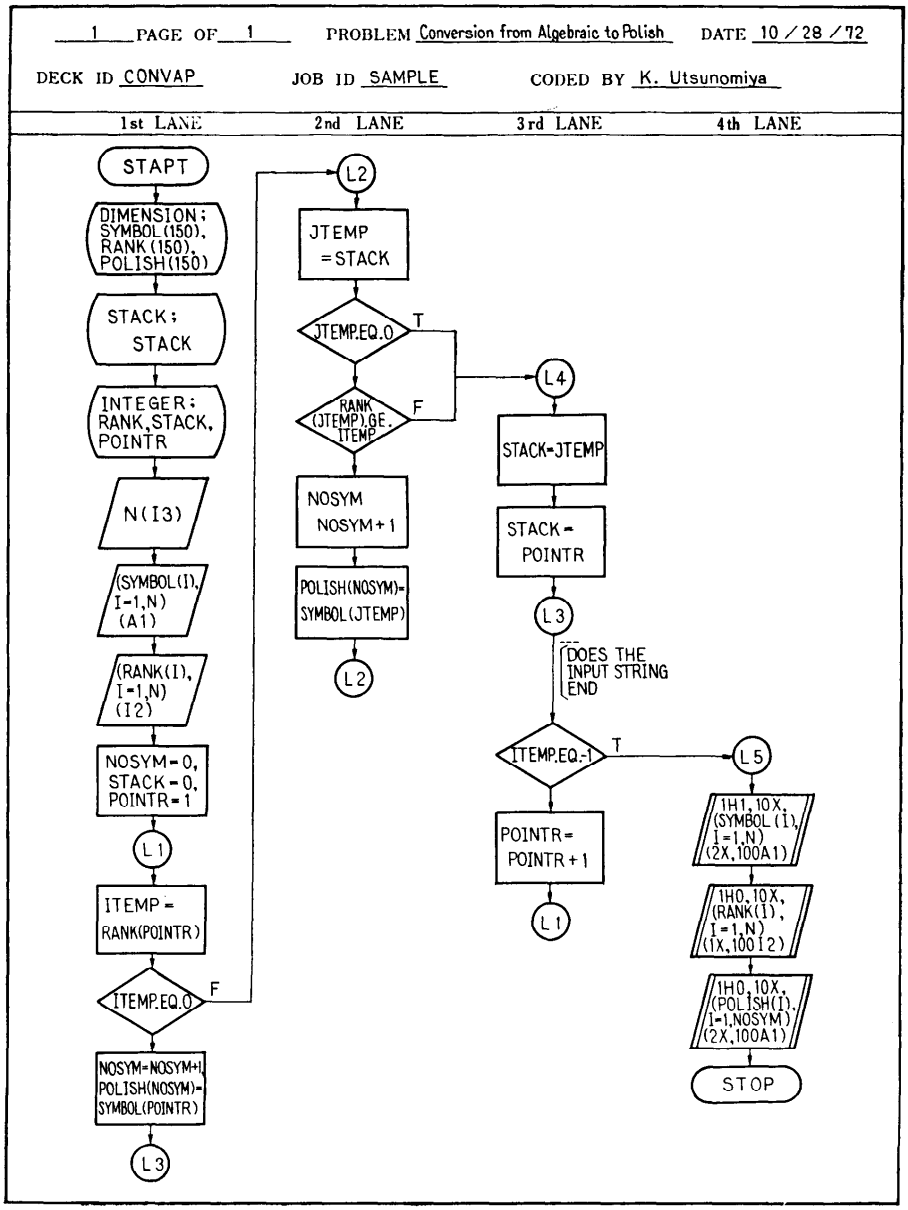


図 10 pFF コーディングの例

CRDNBR	カード通し番号
SNBR	句通し番号
ELEVEL	エラー・レベル
ERNBR	エラー識別番号
FLCODE	FL記号コード
CHRCTR	単語の文字数
BPNBR	ページ番号
BLNBR	レーン番号
FSTCOL	第8欄以降の最初に空白でない欄
DVALUE	判断句でレーン渡りを生じる条件
PTR1	同一レーン内で同一ラベルを結ぶポインタ
PTR2	レーンをまたいで同一ラベルを結ぶポインタ
DECIS	レーン渡りの方向や論理式の数
UM	ラベルの無定義,定義,多重定義
EXTRA	編集したラベル

図 11 属性テーブルエントリの形式

テーブルと，図11に示すような形式の属性テーブルに記入される．また，ラベルの定義と引用に関する情報は CR テーブルに記入され，編集されて Cross Reference Table として出力される．

シンタックスアナライザの重要な作業の1つは，コネクタ句を機能に応じて細分類することである．このため，IDEX, ASW, ACH という3つのフラッグを使用する．スイッチ句の影響が及ぶ範囲，すなわちスイッチ句のスコープ内にあるか否かの2つの場合に分けて考える．

スイッチ句のスコープ内にない場合，

（i） コネクタ句を読んだとき．

IDEX が0であるとき，ASW が0であれば，ASW を1，ACH を0にセットする．ASW が1であれば，直前のコネクタ句のラベルとこのコネクタ句のラベルを比較する．一致していれば，このコネクタ句が不要であることを示すために FLCODE の値を16にセットする．一致していなければ，ACH の値によってさらに2つの場合に分かれる．ACH が0であれば，直前のコネクタ句はジャンプコネクタ句であると識別し，その FLCODE の値を13にセットし，同時に ACH を1にセットする．ACH が1であれば，直前のコネクタ句は孤立しているジャンプコネクタ句であると識別し，FLCODE の値を14にセットする．

IDEX が1であれば，ASW, ACH をともに1にセットし，IDEX を0にリセットする．

（ii） 始端子句，エンド句を読んだとき，

ASW が1であれば，直前のコネクタ句はジャンプコネクタ句であると識別し，FLCODE の値を13にセットし，ASW を0にリセットする．

（iii） 注釈句を読んだとき．

特に何もしない．

（iv） （ii）～（iii）以外の句を読んだとき．

ASW が1であれば，直前のコネクタ句はラベルコネクタ句であると識別し，FLCODE の値を12にセットし，ASW を0にリセットする．ただし，終端子句の場合は，IEEX を1にセットする．

スイッチ句のスコープ内にある場合．

変数型ラベルのコネクタ句のときは，スイッチ句に続くそのコネクタ句だけがジャンプコネクタ句と識別される．変数型ラベルのコネクタ句でないときは，スイッチ句内の論理式と同数のコネクタ句をジャンプコネクタ句と識別する．コネクタ句の数が不足しているときは，最後のコネクタ句をコピーして用いるが，その際は必ずレーンが改められる．ASW は0にリセッ

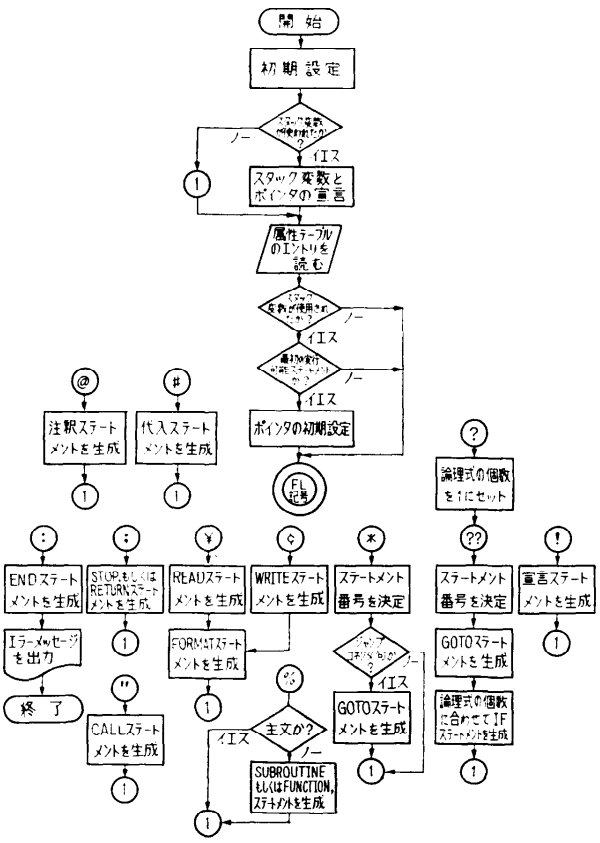


図 12 ジェネレータの流れ図

トされる.

4.2 ジェネレータ

ジェネレータの動作を概略を図12の流れ図で示す. スタック変数に関する部分を除けば, 特に変ったところはない.

スタック変数が認識されると, 長さ300の一次元配列を宣言するための DINENSION ステートメントが生成される. スタック変数が初めて実行可能な句に現われたとき, そのスタックのためのポインタが0に初期セットされる. スタックは同時に3本まで使えるようになっており, それらに対するポインタ名は, それぞれ O0O0O1, O0O0O2, O0O0O3 である.

FFL でコーディングする際のスタックのイメージと, 実際に計算機中につくられるスタックは異なる. このようすを図13に示す. したがって, スタック変数の引用は,

スタック変数名 (ポインタ名)

の形式のコーディングとして生成される. また, スタックAの第I番目のデータの引用は, そのスタックのためのポインタが O0O0O1 であれば,

A(O0O0O1−I+1)

として行なわれる.

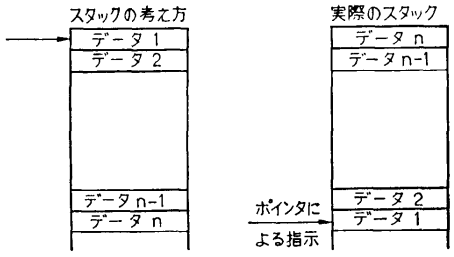


図 13 スタックの考え方と実際

4.3 オートチャータ

オートチャータは3つのフェイズからなっている.

フェイズ1では, 単語を書き入れる箱 (すなわち, pFL 記号) と, その箱をレーン中のどの位置にかくかを決定する. 句の種類に対して, 1つもしくは2つの異なる大きさの箱が用意されている. 単語を構成する文字数に応じてどちらの大きさの箱を採用するかを決定する. 大きい箱の中にも納まりきらないときは, その旨を示す記号 $ を用いて, 納まりきらない部分を箱の外に書く.

フェイズ2では, 隣接するレーン間のレーン渡り, ページ渡りを検出し, フローラインの接続を完全に決定する.

同一レーン内でのフローラインの決定は簡単であるから, レーン渡りの場合についてのみ述べる.

隣接するレーンにあるコネクタ句で同一ラベルをもつものと, そのうちの1つへ分岐する判断句は, シンタックスアナライザによって論理的なチェインを構成するようになっている. 図11の PTR 1, PTR 2 はそのために使用する. このチェインから, レーン渡りのためのフローラインを以下の規則に従って決定する.

規則 1: チェイン中にラベルコネクタ句があれば, 判断句によるレーン渡りと, ジャンプコネクタ句はすべてそのラベルコネクタ句に向うようにフローラインを生成し, ジャンプコネクタ句は生成されるフローチャート上には現われないようにする. ただし, ジャンプコネクタ句が判断句の受け口になっているときは, その判断句の受け口とし, ラベルコネクタ句とは結ばない.

規則 2: 判断句をふくむチェイン中にラベルコネクタがない場合は, ジャンプコネクタ句のうち, その判断句との距離が最小のものへフローラインをひく.

規則 3: ジャンプコネクタ同士では, フローラインはひかない.

フェイズ2までの結果は, 属性テーブルを一部修正, 追加した形の修正属性テーブルとしてフェイズ3に渡される. フェイズ3では, このテーブルと単語テーブルを用いて, 入力された sFFL コーディングに対するフローチャート, すなわち pFFL コーディングを生成する.

5. 実施例

図10の pFFL プログラムを FFL システムで実際に処理した結果を示す. FFL システムは, 現在 IBM 7040 上で動いているので, 文字セットの都合上 FL 記号は表1の括弧の中で示すように変更されている. 図14は sFFL ソースリスト, 図15は生成された FORTRAN コーディング, 図16はオートチャータ出力である.

6. むすび

従来のコーディングはすべて1次元的に行なわれてきたが, Flowchart Language では2次元的に行なわれ, 人間のパターン認識力に直接訴えるため, コーディング, ディバッグの効率が著しく向上する. また, 常に更新されたフローチャートが得られるので, ドキュメンテーションのための労力もずっと軽減され

```
      SYMBOLIC FLOW-CHART LANGUAGE      SOURCE LIST

CRDNO   SQNBR     P LS     STATEMENT
    1       1     1 1( START
    2       2       / DIMENSION  SYMBOL(150),RANK(150),POLISH(150)
    3       3       / STACK  STACK
    4       4       / INTEGER  RANK,STACK,POINTR
    5       5       . N(13)
    6       6       . (SYMBOL(I),I=1,N)(A1)
    7       7       . (RANK(I),I=1,N)(I2)
    8       8       + NOSYM=0,STACK=0,POINTR=1
    9       9       * L1
   10      10       + ITEMP=RANK(POINTR)
   11      11       - ITEMP.EQ.0,F*L2
   12      12       + NOSYM=NOSYM+1,POLISH(NOSYM)=SYMBOL(POINTR)
   13      13       * L3
   14      14      2* L2
   15      15       + JTEMP=STACK
   16      16       - JTEMP.EQ.0,T*L4
   17      17       - RANK(JTEMP).GE.ITEMP,F*L4
   18      18       + NOSYM=NOSYM+1
   19      19       + POLISH(NOSYM)=SYMBOL(JTEMP)
   20      20       * L2
   21      21      3* L4
   22      22       + STACK=JTEMP
   23      23       + STACK=POINTR
   24      24       * L3
   25      25       % DOES THE INPUT STRING END
   26      26       - ITEMP.EQ.-1,T*L5
   27      27       + POINTR=POINTR+1
   28      28       * L1
   29      29      4* L5
   30      30       , 1H1,10X,(SYMBOL(I),I=1,N)(2X,100A1)
   31      31       , 1H0,10X,(RANK(I),I=1,N)(1X,100I2)
   32      32       , 1H0,10X,(POLISH(I),I=1,NOSYM)(2X,100A1)
   33      33       # STOP
   34      34       )
```

図 14 sFFL ソースリストの例

```
      SYMBOLIC FLOW-CHART LANGUAGE      TRANSLATOR LIST

TNBR
    1         FORTRAN STATEMENT
    2         DIMENSION STACK(300)
    3         DIMENSION SYMBOL(150),RANK(150),POLISH(150)
    4         INTEGER RANK,STACK,POINTR
    5         INTEGER GOOOO1
    6         GOOOO1=0
    7         READ(5,1) N
    8       1 FORMAT(I3)
    9         READ(5,2) (SYMBOL(I),I=1,N)
   10       2 FORMAT(A1)
   11         READ(5,3) (RANK(I),I=1,N)
   12       3 FORMAT(I2)
   13         NOSYM=0
   14         GOOOO1=GOOOO1+1
   15         STACK(GOOOO1)=0
   16         POINTR=1
   17       4 ITEMP=RANK(POINTR)
   18         IF(.NOT.(ITEMP.EQ.0)) GO TO 5
   19         NOSYM=NOSYM+1
   20         POLISH(NOSYM)=SYMBOL(POINTR)
   21         GO TO 6
   22       5 JTEMP=STACK(GOOOO1)
   23         GOOOO1=GOOOO1-1
   24         IF(JTEMP.EQ.0) GO TO 7
   25         IF(.NOT.(RANK(JTEMP).GE.ITEMP)) GO TO 7
   26         NOSYM=NOSYM+1
   27         POLISH(NOSYM)=SYMBOL(JTEMP)
   28         GO TO 5
   29       7 GOOOO1=GOOOO1+1
   30         STACK(GOOOO1)=JTEMP
   31         GOOOO1=GOOOO1+1
   32         STACK(GOOOO1)=POINTR
         C     DOES THE INPUT STRING END
   33       6 IF(ITEMP.EQ.-1) GO TO 8
   34         POINTR=POINTR+1
   35         GO TO 4
   36       8 WRITE(6,9) (SYMBOL(I),I=1,N)
   37       9 FORMAT(1H1,10X,2X,100A1)
   38         WRITE(6,10) (RANK(I),I=1,N)
   39      10 FORMAT(1H0,10X,1X,100I2)
   40         WRITE(6,11) (POLISH(I),I=1,NOSYM)
   41      11 FORMAT(1H0,10X,2X,100A1)
   42         STOP
              END
```

図 15 生成された FORTRAN コーディングの例

る．

FFL の使用経験から，

（i） コネクタ句が多くなる，

（ii） FORTRAN 語の DO に相当するものがない，

（iii） スイッチ句の記述が複雑になり勝ちで，生成される FORTRAN コーディングの効率もよくない，

（iv） 入出力句が見にくい

などが指摘された．現在，これらの意見をすべて取り入れ，しかも実用上便利と考えられる機能を追加した改良型の FL を設計中である．

FFL の応用も数多く考えられている．たとえば，FORTRAN プログラムを sFFL プログラムに変換することを自動化し，FFL システムと結びつけることにより FORTRAN プログラムの自動フローチャーティングが可能であるし，FFL によるアルゴリズムの等価性のチェックもいくつか行なわれている．

7. 謝辞

FL の仕様検討に参加して下さった大谷真氏，四条忠雄氏，平本厳氏，坂本陽之助氏，計算機にかける手伝いをして下さった佐古哲也氏，藤原寛文氏に心から感謝します．

参 考 文 献

1) Hirose, K., Utsunomiya, K. et al.: A Formalization of Algorithms and Flowchart Language, to appear.
2) Hirose, K., Utsunomiya, K. et al.: A Design of Flowchart Language and its Implementation; to appear.
3) 前川：自動フローチャーティング，情報処理，Vol. 9, No. 3 (1968), pp. 128～136.
4) 山本，山口：自動フローチャーティング，情報処理，Vol. 11, No. 12 (1970), pp. 711～720.

（昭和 48 年 2 月 10 日 受 付）
（昭和 48 年 4 月 10 日 再受付）

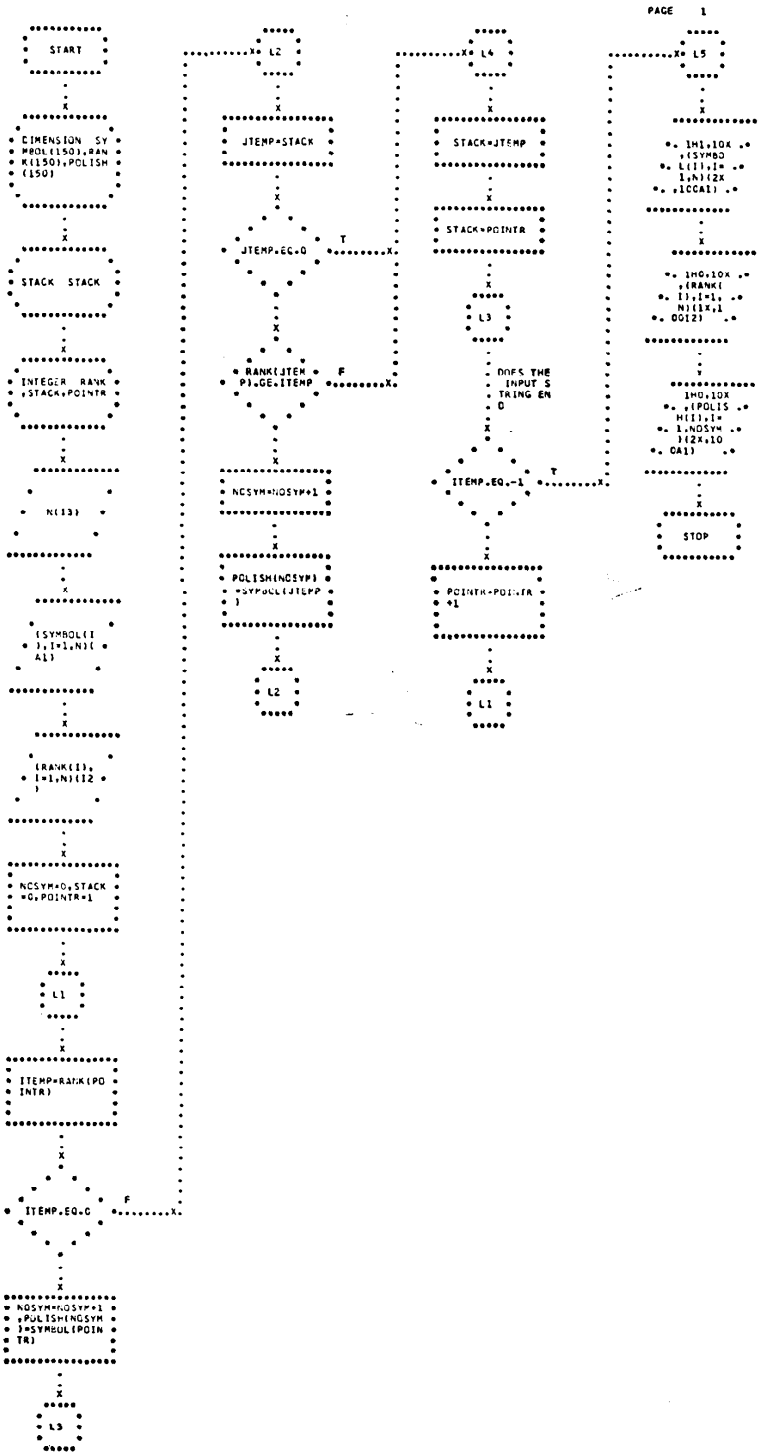


図 16 オートチャータ出力の例