解　説

構造化プログラミング ワーニエ・メソド†

鈴　木　君　子††

0. はじめに

ワーニエ・メソドは，フランスの J-D. ワーニエ (Jean-Dominique Warnier) が1967年～69年に開発し，1971年に発表した構造化プログラミングの一手法である．

開発の母体となった研究活動のテーマは，「データ処理の論理的基盤となるルールや法則を解明すること」だったといわれる．そのためか当初このメソドは，「メソドロジイ」として発表された．しかし，開発者の長い実務経験と，発表前2年間もの慎重な実地テストに裏付けられたこのメソドは，意図した方法論としてよりも，実用手法として高く評価され，数年の内にはフランスでは，プログラマ必修の技法として，その習得が，新聞の求人・求職広告の条件欄に記載されるまでになった．

日本での発表（邦訳出版）は1973年である[1),2)]．以来10余年にわたってワーニエ・メソドを使って来たことになるが，未だに古さを感じないのは思えば不思議である．これまで，学会の動向に疎かった筆者は，4月のシンポジウム（「プログラム設計技法の実用化と発展」（昭和59年4月10日～12日））での発表で，ワーニエ・メソドが思ったより多くの方々に支持されていることを知って，驚きもし，嬉しくもあった．しかし，反面，もっとも使ってほしいようなところでは，案外使われていないのが実状のようである．それがこのようなメソドの宿命なのかもしれない．

この稿は，そのシンポジウムでの発表を主体とするもので，筆者自身が現在，実際のプログラム作成に使用しているワーニエ手法の概要である．

1. 概　要

1.1 特　徴

ワーニエ・メソドの特徴の第一は，プログラム構造設計の土台に，一定の細分法則に従って作成する階層型のデータ構造を使用することであろう．特に処理対象となる入力データの役割は重要で，その構造はそのまま，プログラム構造に反映される．

特徴の第二は，構造図作成の手段として，集合論の考え方をデータと業務の分析に取り入れたことであろう．

集合では，その要素に共通の性質（属性）があることが必要だが，ワーニエ・メソドではこの属性を，「どこで，いつ，何回」という問いかけへの答えに求める．たとえば，1枚の売上伝票内に何行か書かれている売上製品データは，伝票データ内（どこ）で，見出し項目群の後に（いつ），売上げられた製品の数だけ（何回）繰返し現われるデータという属性によって，1伝票集合を形成する要素とみなされる．

1.2 作成手順

このような二大特徴を持つワーニエ・メソドのプログラムは，下記の3ステップから成る手順に従って作成される．

第1ステップ：論理設計のステップで，階層構造による処理のモジュール化，流れの把握を行う．

a）出力データ論理構造図の作成

b）入力データ論理構造図の作成

c）（必要あれば）真理値表，ベッチ図表による処理条件の整理と簡単化

d）プログラム構造図，流れ図の作成

第2ステップ：詳細設計のステップで，その業務に必要な処理命令と，それらの命令が使用される場所（モジュール）との関連づけを行う．

a）処理命令の種類別リスト・アップ

b）リストした処理命令の，プログラム構造内該当モジュールへの割当て

c）割当てられた処理命令のモジュール別リスト（プログラミング・テーブルの作成）

第3ステップ：プログラム言語によるコーディング

† Structured Programming Warnier Method by Kimiko SUZIKI (Uchida Oil Hydraulics Mfg. Co., Ltd.).

†† 内田油圧機器工業(株)

のステップで，作成されたプログラミング・テーブルに従い，その内容を使用言語でコーディングする．

上記手順に従った解法の例には，前述のシンポジウムで使用した共通問題（ワーニエ，ジャクソン，複合設計用共通問題）をそのまま使用するが，その例題解説に入る前に，ワーニエ・メソドの基本的な部分であるデータ構造図とプログラム構造図についてここで簡単に触れておこう．

1.3 データ構造図の作成

上述のようにこのメソドでは，データ構造の把握を非常に重要視し，プログラムの作成も，まずデータ構造図を作ることから開始する．構造図は，特徴の項で述べた「どこで，いつ，何回」の基準による階層構造となるが，その作成は，『「現われる回数≠1回」のデータ部分集合を含む集合を，最高水準（最大のもの）から順に細分する』という細分法則を適用して行う．

たとえば，出力帳表として図-1 のような営業所別・商品別の売上高リストがあるとする．

このデータを，売上商品の単位にまとめると，図-1(a)のような複数個のデータ集団ができあがる．これをさらに営業所単位にまとめたのが，図-1(b)である．この図-1(b)に，先述の細分法則を適用すると，一覧表全体の中で，「≠1回」現われる最大のデータ集合は，営業所データであり，その出現回数は，営業所の数と同数である．回数を営業所の頭文字を採って「E回」とすると，この第1レベル部分は，下記のように表現することができる．

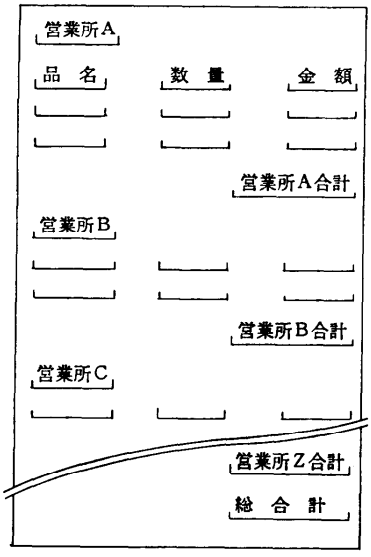


図-1

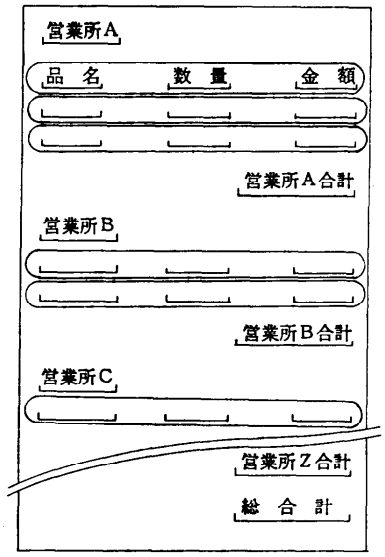


図-1(a)

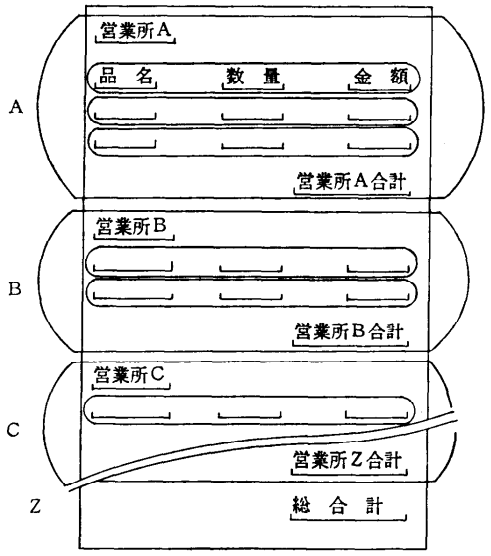


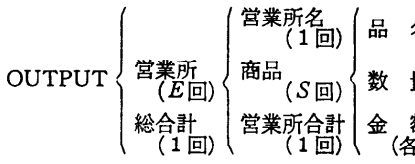
図-1(b)

$$\text{OUTPUT}\ (\text{売上高リスト})\left\{\begin{array}{l}\text{営業所}\\(E\text{回})\end{array}\right.$$

つぎに大きなデータ集団は，売上商品データである．これは，それぞれの営業所内で，売上げられた商品の数（S回）ずつ現われる．これを，代表の1営業所内で考え，上表に連結すると，第2レベルまでのデータ構造は，下記のとおりとなる．

$$\text{OUTPUT}\left\{\begin{array}{l}\text{営業所}\\(E\text{回})\end{array}\right.\left\{\begin{array}{l}\text{商品}\\(S\text{回})\end{array}\right.$$

ところで，出力データの細分では，後にプログラムの検証に使用する関係上，「回数≠1回」の囲みからはずれた「＝1回」のデータ項目まで，すべてを構造図上に，水準別に表示することになっている．したがって，それらを上表に付け加えて，出力データ構造図(Logical Output File—略して「LOF」という）を下記の形に完成させる．

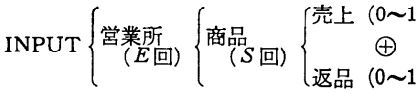


なお，「営業所名」と「営業所合計」を「商品」の上下に分けたのは，先述の「いつ」の基準からである．

このようにデータ構造の分析は，抽象度の高いものから順に，トップ・ダウン型に進む．ここでは出力データのみを例に挙げたが，入力データについても同様である．ただし，入力データでは，「回数＝1回」の要素は，後に述べるプログラム構造との関連で，構造に影響を与えないため，無視してもよいことになっている．

ここで，**データ構造の形態**について簡単に述べよう．このメソドの特徴をなす「回数」に，「**1回**」のものと「***n* 回**」のものがあることは，上例に見るとおりだが，分析の基準となる「≠1回」にはこの他に「**0〜1 回**」という考え方がある．これは，ある集合内で1回現われたり，現われなかったりするデータの表現に使用され，「n回」の**反復型**に対して，**選択型**と呼ばれる．

選択型構造を含めた入力データ構造図 (Logical Input File-**LIF**) の例を下記に挙げる．



第3レベルの選択構造内にある「⊕」の記号は，その上下のデータが排他的に現われることを意味する注)．したがって，この構造図からは下記の意味が読み取れる．

注) これに対し，排他的ではない状態で「0〜1 回」現われる複数個のデータは「＋」記号で表示する．

「入力データ全体の中には，反復型の営業所データがあり，それを代表する1営業所データ内には，やはり反復型の商品データがある．その代表の1商品データは，売上か返品のいずれかを含む選択型データである.」

反復型と選択型は，ワーニエ・メソドの基本をなす構造形態で，すべてのデータ，および後述するプログラムの構造は，基本的にはみなこの2種から構成されている．

1.4 プログラム構造図の作成

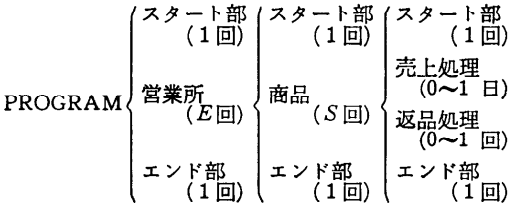
ブレーク・ダウンと呼ぶプログラム構造図の作成は，「処理構造は入力データ構造に基づく」の思想から，LIF をベースに，下記法則に従って行われる．

- 入力データが反復構造であれば，そのデータを処理するプログラムの部分も反復構造である．
- 入力データが選択構造であれば，そのデータ処理のプログラムの部分も選択構造である．

ただし，データ構造とプログラム構造の形態には，いくらかの相違がある．その1はプログラム構造では，1水準の処理（1括弧）内の最初と最後に，「処理回数＝1回」の「スタート部」と「エンド部」をかならず設けることである．第2の相違は，多種の中の1種のデータのみを処理対象とするような選択構造の場合でも，プログラムではかならず2つの「0〜1 回」の処理——たがいに排他的に実行される正補一対の処理——を作ることである．なお，この正補一対の処理構造は，結果的に一方が無処理の集合に終わることになるかもしれないが，プログラムの形態を整え，単純化する上で役立つ．

それでは上例の LIF に，この法則を適用してプログラム構造を作成してみよう．

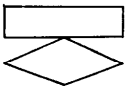
第1レベルでは LIF が「営業所」の反復構造であるため，プログラムも「営業所」の反復構造である．第2レベルも同様に「商品」の反復構造になり，第3レベルでは，「売上」と「返品」がそのまま，排他的な正補一対の処理の選択構造となって，それにスタート部とエンド部をつけると全体のプログラム構造図は下記の形にできあがる．



なお，プログラムの選択構造に排他的記号「⊕」を付さないのは，処理の選択構造がその性質上，かならず排他的になるためである．

このブレーク・ダウンには，「営業所」や「商品」

のように，次のレベルでさらに細分されるものと，スタート部やエンド部，および「売上」や「返品」のように，細分されないものとがある．細分されていない部分は，初めに述べた「どこで，いつ，何回実行されるか」の答えの同じ処理命令が集まるところで，「ロジカル・シーケンス」と呼ばれ，プログラム構造の最小単位(モジュールと考えてよい)を構成する．ロジカル・シーケンスは1個の入口，1個の出口を持つ処理命令の集まりで，フローチャートの1処理シンボルに相当する．ただし，ロジカル・シーケンスには，出口に条件を与えてプログラムの連絡先を決定する分岐命令まで，その中に含めるという特徴がある．したがってそのようなシーケンスでは，出口は1個だが，分岐先が当然複数となり，その表示方法として，処理と判断を一体化させた下図のようなシンボルを用いる．



ブレーク・ダウンはまた，プログラムの流れを示す図表でもある．しかし，見慣れないために，ここから流れを読みとるのが難しいこともあるので，必要であれば，この図からフローチャートを別に作成することになっている．

ブレーク・ダウンとフローチャートの関係は，**図-2** および **図-3** のとおりで，このようにレベル別に作成してもよく，また，数レベルをまとめたフローとして，**図-4** のように作成してもよい．

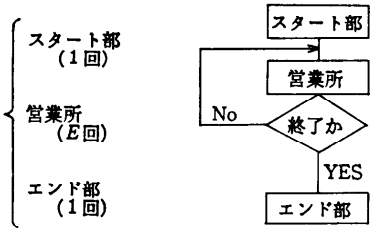


図-2 反復構造

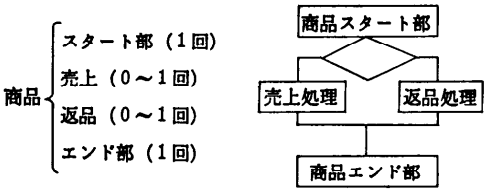
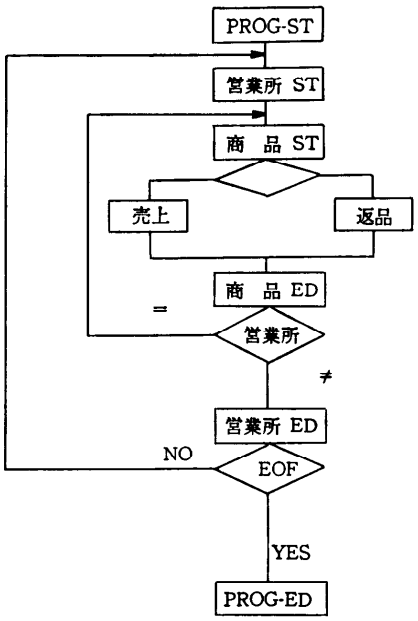


図-3 選択構造



注)「スタート部」には「ST」の，「エンド部」には「ED」の略語を使用した．

図-4 全体のフロー

2. 共通問題による解法例

2.1 共通問題

共通問題は別項に掲載されている，ある酒類販売会社の在庫管理問題である．

この例題には主題が2つある．積荷票による入庫処理と，出庫依頼による出庫処理である．しかし，プログラム作成が要求されているのは，その中の「受付係の仕事」と称する「在庫なし連絡，出庫指示書作成および在庫不足リスト作成」であるので，在庫データの入力処理になる「積荷票」からの入力プログラムは省略し，その処理によってファイル内に，あらかじめ在庫データが作られているとの想定の下に，プログラムを作成した．

また，在庫データをどのような形でファイル内に持つかについては何の指定もなかったので，扱い製品が「食品」であることを考慮し，長期間在庫が発生しないような在庫処理方法——先入れ先出し法（売れないで残るものは別だが）——向きのファイル形態を選んだ．このため結果として，問題中の「倉庫内のコンテナ数をできるかぎり最小にする」ことを優先させなかったことをお断りしておく．

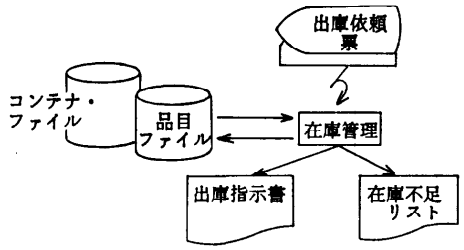


図-5

2.2 処理概要

プログラム作成の対象にした業務図を，図-5 に示す．この業務は次のように処理される．

a）出庫依頼票に基づき，「品名」「数量」「送り先名」を，キーボードから入力する．

b）その「品名」により「品目ファイル」にアクセスして，出庫するべき数量を得る．

c）さらに，「品目ファイル」のレコードから得られる「コンテナ No.」をキーとして，「コンテナ・ファイル」にアクセスし，積載数合計を更新する．

d）在庫数量が出庫依頼数に満たない場合は，在庫数量を出庫し，不足分を不足リストに出力する．

出力の2帳票――「出庫指示書」と「在庫不足リスト」――は図-6 に示した．

在庫データを格納するファイルには，コンテナ別の製品情報を入れる「品目ファイル」と，空コンテナを検出するための「コンテナ・ファイル」の2本を作成した．図-7 がそのファイルのレコードである．

入力データにはこの他に，キー入力の「品名」「数量」「送り先名」がある．

キー入力の処理は，実務のプログラムでは専用画面を設計し，オンライン処理とすべきところだが，手法解説用の例題であることから，コーディングの簡単な「ACCEPT」命令で代用した．

なお，出力帳表内の「注文 No.」は，注文データ（出庫依頼）をインプットするたびに，計算機内部で自動的に作成されるようプログラムを組んだ．

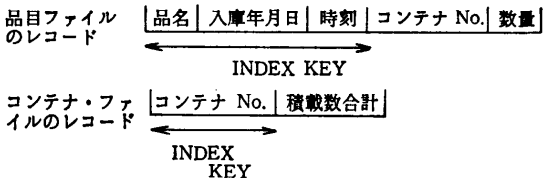
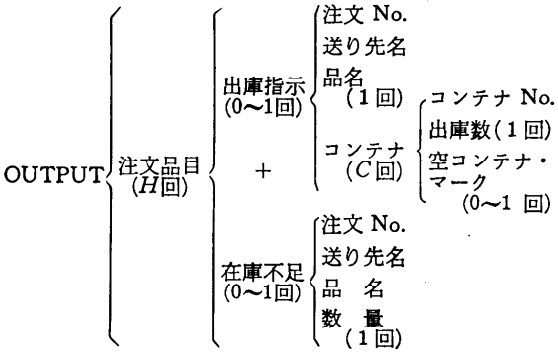


図-7

2.3 出力データ構造図（LOF）

この例題の出力データは，フィジカルには上記2種のリストである．しかし論理構造である LOF は，この物理的形態にはとらわれずに，前述したとおり「現われる回数≠1回」のデータ集合を，業務全体から順に捜す形で作成する．

まず，全体の中の最大集合は，両出力に共通の「注文品目」集合である．それをさらに分析すると，それが「出庫指示」データとなることもあり，「在庫不足」データとなることもあり，さらに両者にわたることもあることがわかる．この両集合の回数は「0～1 回」である（下図の第2レベル）．それらをさらに分析し，「＝1回」のデータも加えてでき上がったのが下記の構造である．



上図中の第2水準は，2個のデータが互いに排他的

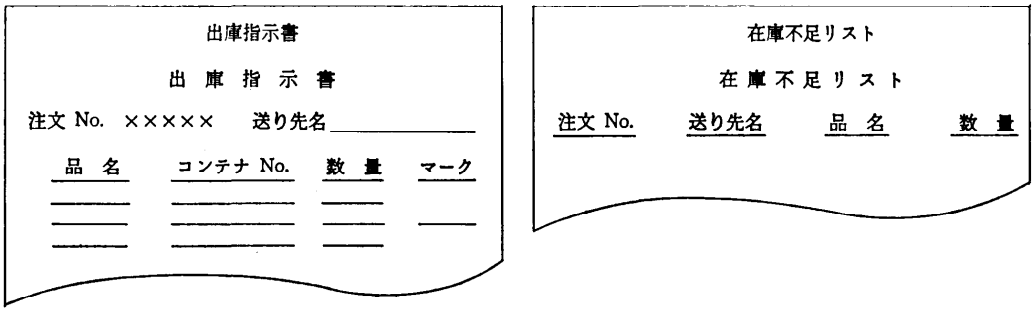


図-6

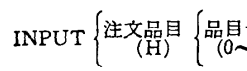
でない状態で 0〜1 回現われることを示す．すなわち，「出庫指示」と「在庫不足」のデータは，一方のみが現われる場合もあれば，両方が同時に現われる場合もある．なお，この形のデータ構造を，複合選択構造と呼ぶ．

2.4 入力データ構造図 (LIF)

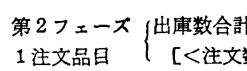
LIF の作成手順も LOF と同様で，対象の2ファイルと，キー入力のデータ全体の中に，抽象度のもっとも高い「≠1回」のデータを捜すことから開始する．

まず現われるのは「注文品目」データである．しかしその中には，キー入力はされたが，当の品目がファイル内にない場合もあるので，第2水準は「0〜1 回」の「品目データ」(レコード群) となる．さらに，該当の品目が存在した場合は，複数のコンテナに分かれて格納されているその品目を，注文数に達するまで各コンテナから拾い集めねばならない．これが第3水準の「コンテナ (C回)」である．これは同時に「積載数合計」を持つコンテナ・ファイルの「コンテナ」レコードでもある．

このようにして，下記の LIF ができあがる．



さて，この例題には，「ある品目の在庫数をすべて集めてみたが注文数に足りなかった」というような，入力データには含まれないが，プログラム内で計算され，プログラム論理に影響を及ぼす特別のデータがある．ワーニエ・メソドではこのようなデータを「第2フェーズの入力データ」と呼び，たとえば上記の場合なら，「出庫数合計が注文数に満たない場合が0〜1回ある」として，下記のように表現することにしている．



注) 大括弧 [ ] は，検知の対象とそれの現われる状態が，データ項目の内容 (値) に関するものであることを示す．

第2フェーズのデータにはこの他に，注文数に達するまで在庫数を集計してゆく際に一時的に起こる「合計数>注文数」の場合と，コンテナが空の状態になる場合があり，下記のように表現する．

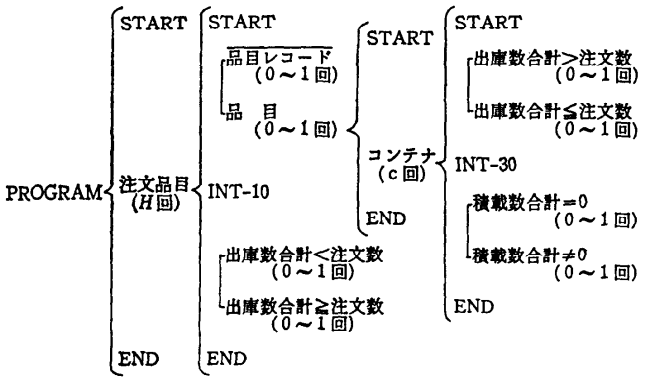
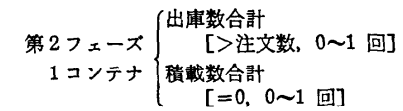


図-8



以上で入力データ構造図の作成は終了する．つぎはこれに基づくプログラム構造の設計である．

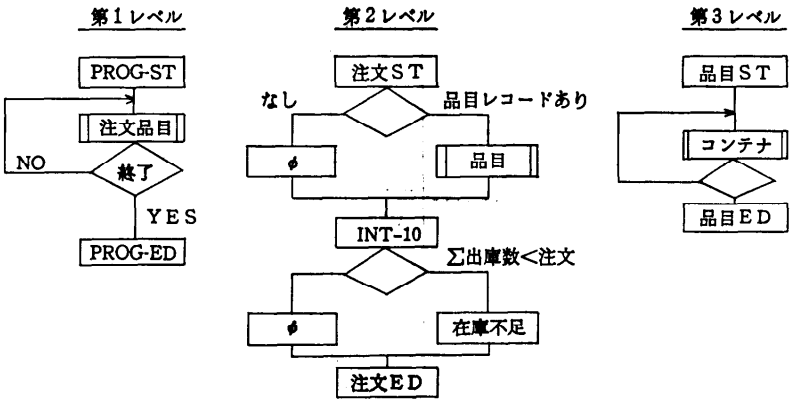
2.5 プログラム構造図

プログラム構造図は，第1，第2の両フェーズの入力データから，既述の法則にしたがって作成するが，ここで新たに発生する問題は，第2フェーズのコンテナ・データのように，1データ内に2個以上のデータ構造が含まれる場合のプログラム構造の作り方である．この対処方法は次のとおりである．

- プログラム階層の1レベル（1括弧）内に，2つ以上の反復または選択の処理が含まれる場合には，そのおのおのの中間に「処理回数=1 回」の中間モジュール「INT」(Intermediate Sequence) を置く．なお，この形態が反復型のみから成るものを「複合反復」，選択型のみから成るものを「複合選択」，両者が混じっているものを「混合」構造という．

上例のプログラム構造はつぎのようになる．

- 第1レベル：「注文品目 (H回)」のみの単純な反復構造である．
- 第2レベル：「1注文品目」内に含まれるデータの2つの選択構造（第1フェーズと第2フェーズ）に対応して，プログラムも2組の選択型の処理となり，中間を「INT (1回)」で結ぶ複合選択構造となる．
- 第3レベル：「品目データ」が存在する場合の処理として，「コンテナ」ごとの処理が繰返される単純反復構造である．



「φ」印は無処理（ブランク・シーケンス）を表わす.

図-9

● 第4レベル：第2フェーズの「コンテナ」内のデータの2状態に対応する複合選択構造である.

こうして，図-8 のプログラム構造ができあがる.

このブレーク・ダウンをレベル別にフローチャートに置きかえたのが，図-9 であり，全体として作成したのが図-10 である．レベル別の図の第4レベルは，第2レベルと同形なので省略した．各レベルは，2重枠部分で下位レベルとつながっている．すなわち，上位レベルの2重枠部分を詳細化したものが，次レベルのフローである.

2.6 処理命令の種類別リストと該当モジュールへの割当て

このステップではまず，対象の業務で使用すべき処理命令を，下記の4種に分けてリストアップする.

a）入力命令とその準備の命令
b）反復・選択の各条件の設定と，その準備の命令
c）演算命令と，その準備の命令
d）出力命令とその準備の命令

ついでこれらの処理命令を，「どこで，いつ，何回」の基準にしたがい，該当するシーケンス(モジュール)に割当てる．たとえば，コンテナごとの出庫数（在庫数）を注文数に足りるまで合計する加算命令は，該当するコンテナの数だけ実行すべきであるから，その回数だけ実行されるコンテナ処理の中のスタート部に置く．また，その合計エリアの消去（演算準備命令）は，注文ごとに1回，計算に先立って行うべきであるから，「注文スタート部」に置く.

なお，順ファイルの読込み命令は原則として，ファイルごとに2命令（第1 READ と第2以降のデータの READ）とし，第2以降の READ は，前のデータの処理終了時点に置く.

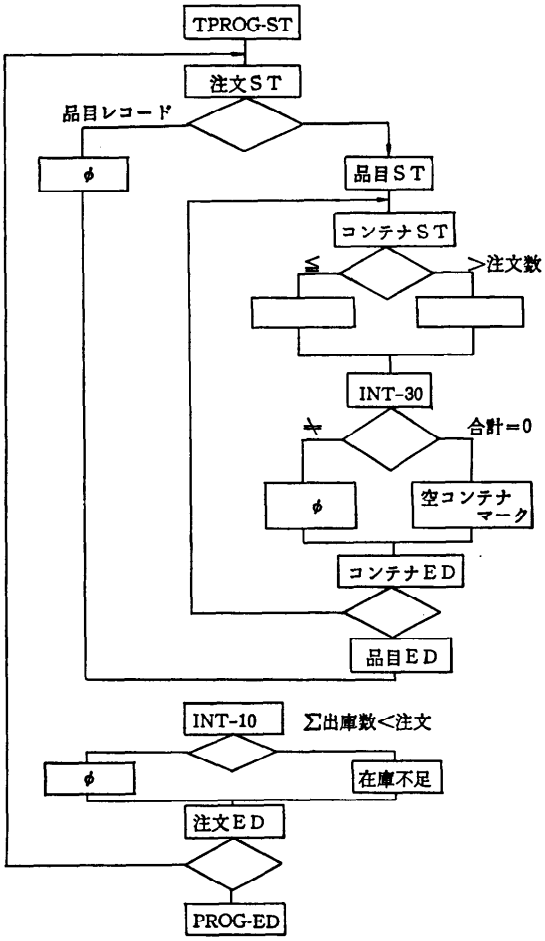


図-10

このようにして作成されたのが下記のリストで

ある.

a) 入力命令とその準備命令

- 第1データの KEY-IN……PROG-ST
- 第2以降のデータの KEY-IN……注文品目 ED
- 品目ファイル第1 READ 用位置づけ ……注文 ST
- 1注文ごとの第1 READ……品目 ST
- 1注文ごとの第2以降の READ ……コンテナ ED
- コンテナ・ファイルの読込み……INT-30
- 注文 No.（自動採番）の取込み……注文 ST

b) 反復条件の設定命令とその準備の命令

- 注文品目処理の終了……終了記号 (KEY-IN の品名＝スペース) まで
- コンテナ処理の終了……「読込みレコードの品名≠KEY-IN の品名」または「出庫数合計≧注文数」まで
- 品名の比較用保有……注文 ST
- 注文数の比較用保有……注文 ST

c) 選択命令とその準備の命令

- 品目レコードなしの判定……注文 ST(位置付け命令の INVALID)
- 在庫不足の判定（出庫数合計＜注文数）……INT-10
- 出庫数合計＞注文数（コンテナ内在庫を全部出庫すると注文数を超える場合)……コンテナ ST
- コンテナの積載量＝0……INT-30
- 注文数の保有（既出）

d) 演算命令とその準備命令

- 在庫数量の出庫数への加算……コンテナ ST
- 積載数－出庫数＝積載数……INT-30
- 出庫数合計－数量＝出庫数合計
- 注文数－出庫数合計＝出庫数
- 数量－出庫数＝数量

（「出庫数合計＞注文数」で実行するモジュール）

- 注文数－出庫数合計＝不足数……「出庫数合計＜注文数」のモジュール
- 出庫数合計エリアの消去……注文 ST

e) 出力命令とその準備の命令

イ) 出庫指示書

- 注文 No. 送り先名のプリント……品目 ST
- それらデータの編集……注文 ST
- コンテナ・ラインのプリント……コンテナ ED
- 品名の編集……品目 ST
- コンテナ No. の編集……コンテナ ST
- 出庫数（コンテナ別）の編集……INT-30
- 空コンテナ・マークの編集……「積載合計＝0」のモジュール

ロ) 在庫不足リスト

- 注文 No. ラインのプリント
- そのデータの編集

（「出庫数合計＜注文数」のモジュール）

ハ) 品目ファイルの更新

- 数量の更新，再書込み……INT-30

ニ) コンテナ・ファイルの更新

- 積載合計数の更新……INT-30
- 再書込み……「積載数≠0」のモジュール
- コンテナ・レコードの削除…「積載数＝0」のモジュール
- 注文 No.（自動累進）の再書込み……注文 ST

2.7 プログラミング・テーブルの作成

このステップでは，上記の処理命令を所属するモジュールごとに集めて，下表のように記述する．モジュールの記述順序はプログラム構造図に準じ，モジュール内処理命令の記述順序は，原則として下記に拠る．

i) 反復・選択条件用の準備命令
ii) 演算の準備および演算の命令
iii) 出力の準備および出力命令
iv) 入力命令
v) 分岐命令

プログラミング・テーブル

*** LEVEL 1 全体 ***
PROG-ST.
ファイルの OPEN
KEY-IN データの取込み
注文処理ルーチンの反復実行
(KEY-IN 品名＝スペースまで)
PROG-ED.
ファイルの CLOSE
STOP RUN.
*** LEVEL 2 注文品目ルーチン ***
注文 ST.
品名・注文数のストア
出庫数合計エリアの消去
注文 No. の取込み

```
        注文 No. を更新して再書込み（自動更新）
        注文 No. 品名，送り先名の編集
        インプットの品名を KEY に，品目ファイルの位置付け
          INVALID なら INT-10 へ
        品目処理ルーチンの実行
INT-10.
        出庫数合計＜注文数なら
              在庫不足処理　実行
注文 ED.
        KEY-IN データの取込み
```

```
*** LEVEL 3 品目ルーチン ***
品目 ST.
        注文 No. 送り先名のプリント
        プリント・エリア消去
        品名（出庫指示）の編集
        品目ファイルの順 READ
              END ならエラー処理
        コンテナ処理ルーチンの反復実行
              出庫数合計≧注文数　または
              読込み品名≠保有品名　まで
品目 ED.
        EXIT.
```

```
*** LEVEL 4 コンテナ・ルーチン ***
コンテナ ST.
        数量＋出庫数合計＝出庫数合計
          出庫数合計＞注文数なら
              そのコンテナからの出庫数を算出
          出庫数合計≦注文数なら
              そのコンテナの数量を出庫数とする
        コンテナ No. の編集
INT-30.
        出庫数により品目ファイルの数量を更新して再書込み
        品目レコードのコンテナ No. を KEY に，
        コンテナ・ファイルの読込み
              INVALID ならエラー処理
        積載数合計を更新
        積載数合計＝ゼロなら
              空コンテナ・マークを編集
コンテナ ED.
        コンテナ・ラインのプリント
        プリント・エリア消去
        品目ファイルの順 READ
              END なら　品名に HIGH-VALUE を移送
```

```
*** 在庫不足処理 ***
在庫不足 ST.
        注文数－出庫数合計＝不足数
        不足リスト・ラインの編集とプリント
在庫不足 ED.
        EXIT.
```

3. おわりに

以上がワーニエ・メソドの概要である．「どこで，いつ，何回」という素朴な基準がデータ構造を生み，さらにその入力データ構造図が，相似形のようなプログラム構造となってゆく過程が，一通りおわかり頂けたと思う．

ワーニエ・メソドにはこれらの基本手法の他にも，複雑な処理条件を「真理値表」や「ベッチ図表」を使って整理・簡単化し，プログラム構造につなげてゆく方法や，マッチング処理の簡単な解決方法など，実用に適した便利な小技法がいくつかあるが，これらの説明はまた別の機会に譲る．

数年間で世代が交代するといわれる変遷はげしいコンピュータの世界で，10数年もの間利用され，いまだに古さを感じさせない手法というのは，それほど多くはないであろう．その理由を，ワーニエ・メソドがそれだけ，情報処理というものの本質に迫った理論であるためと見るのは，筆者の単なる欲目だろうか．

参考文献

1) Warnier, J-D. and Flanagan, B. M.: ENTRAINEMENT A LA PROGRAMMATION Tome 1--Construction des Programmes. Les Editions d' Organisation, Paris (1971). (以下原文の発行所はみな同じ.)
鈴木訳，ワーニエ方式によるプログラミング学習（上巻），日本能率協会 (1973).

2) Warnier, J-D.: ENTRAINEMENT A LA PROGRAMMATION Tome 2--Exploitation des Données, 鈴木訳，ワーニエ方式によるプログラミング学習（下巻），日本能率協会 (1973).

3) Warnier, J-D.: LES PROCÉDURES DE TRAITEMENT ET LEURS DONNÉES.
鈴木訳，ワーニエ・プログラミング法則集，日本能率協会 (1975).

(昭和59年7月25日受付)