# 言語処理系によるアルゴリズム教育

## ——構造化プログラミングと構造化流れ図の有効性——

能登 宏

言語処理系環境下でのアルゴリズム理解の諸相をⅠ．表記（法）とその意味，Ⅱ．手順の構成，Ⅲ．「手順の抽出」と「手順進行の契機」，Ⅳ．「処理手順の進行」と「論理関係の展開」，Ⅴ．「構造化プログラミング」の理解，Ⅵ．「構造化プログラミング」の有効性，Ⅶ．アルゴリズム教育上の留意点の側面から分析する．学生のアルゴリズム理解度は，「論理的思考の深さと適応性」，「言語処理系」及び「プログラミング」に関する基礎知識や実習経験に依存する．授業・実習に於いては，学生の情報処理に関する知識や経験の相違に留意した指導が望まれる：アルゴリズム理解のどの相に不明な点があるか，或は，それぞれの相に於いては何処で行詰まっているかを具体的局面で認識させていくことが必要である．本論では3つの処理手順（「給与支給額の計算」，「『最大公約数』を求めるEuclidの『互除法』と『互減法』」，「画面遷移」）を取り上げ，「流れ図」の具体的な事例を引用しながらアルゴリズム教育に於ける要点を考察する．その際，「構造化流れ図」或は「構造化プログラミング」の手法が，「構造化された制御構造」という制約を課すことによって処理手順の整合性を保証し，処理手順の論理展開に有効であることを指摘する．

## 1．はじめに

北星学園大学経済学部経営情報学科で展開されている「情報処理関連教育科目」のうち，「応用プログラミング論（B）」（2年次学生8名）と「演習II」（4年次学生18名）を著者が担当している．いずれの教科目に於いても，「COBOL言語処理系」を採用しており，前者では「プログラミング」教育，後者では「システム設計」の指導を行っている．

本稿では，前回の「『情報処理教育』研究会」(1992年12月)[1]に引き続き，言語処理系(COBOL)環境下に於ける学生のアルゴリズム理解の諸相を，次の7つの側面から分析する：Ⅰ．表記（法）とその意味，Ⅱ．手順の構成，Ⅲ．「手順の抽出」と「手順進行の契機」，Ⅳ．「処理手順の進行」と「論理関係の展開」，Ⅴ．「構造化プログラミング」の理解，Ⅵ．「構造化プログラミング」の有効性，Ⅶ．アルゴリズム教育上の留意点の側面から分析する．

学生のアルゴリズムの理解度には，「論理的思考の深さ」，「プログラミング言語の知識とプログラミングの経験」，「言語処理系及び処理系操作の習得度」等に応じた相違がある．上で述べた各相でのアルゴリズム理解の特質を明確にし，学生の理解度の水準に配慮したアルゴリズム教育の要点を指摘したい．その際，「処理手順に関する構造化手法」がアルゴリズムの理解と処理手順の構築に有効であることを強調する．

Hirosi Noto 北星学園大学経済学部経営情報学科

本稿では，次の4つの処理を事例として，考察を進める．

1）「誤り手順」の事例
2）「給与支給額の計算」(「応用プログラミング論（B）」)
3）「最大公約数」を求めるEuclidの「互除法」と「互減法」
　－「Dowhile型」と「Dountil型」の2通りに描く（「応用プログラミング論（B）」)．
4）「画面遷移」(「演習II」)

以上の処理についての「流れ図」の例図は，図1に事例番号を付して引用する事にする．各事例とも(正)（誤)の区別をして掲載する．

## 2．分析と考察

### I．表記（法）とその意味

「流れ図」を描くに当たっては，「手順」の最小構成要素となっている「単一の処理」，所謂〝ステップ〟という概念とその実体（具体例)，及び，「表記法の基本」を確実に教え込む事が必要である．その後に，次のことが出来なければならない：「流れ図」は，「手順」を視覚的に表現する一つの表記法であるから，自然言語或いは擬似コードで記述されたある「手順」と，「流れ図」で表現された当該「手順」との間で，双方向の対応がとれていること．

1）「手順」の中の「単一の処理」（1ステップ）の仕訳
　手順というのは，「単一の処理」，の集積であるから，「流れ図」を描くためには，「手順」を「単一の処理」に仕訳け出来なければならない．そのためには，「単一の処理」（1ステップ)とは具体的に何を指すのかを実例を通じて理解させる．
2）表記法の基本の習得
　表記上の約束とその約束事が持っている意味合いをよく把握させる．
　●流れ図に於ける「流れ線」
　　－「上から下へ」，「左から右へ」が原則
　　－「流れ線」は，「出口」や「無条件分岐」(GO TO文）を除いて行き止まりになる（袋小路に入る）ことはない
　　－「流れ線」は，交叉することはない
　●「モジュール」(定義済処理)
　　－独立性の強い一まとまりの処理・機能
　　－繰り返し使用される処理をモジュールにしておく
　●「モジュール」や「制御構造」に関して，「流れ線」の「入口は1つ」，「出口は1つ」を原則とする
3）「流れ図」と自然言語或いは擬似コードとの対応
　●「流れ図」として表現されている内容を「手順」として読取れるか，或いは，自然言語で記述出来るか？
　●自然言語或いは擬似コードで記述されている「手順」を「流れ図」に表現できるか？

以下の事例の記載では，事例番号は，図1の【事例［番号］】を参照している．1），2）のような数字は，上記説明文中の分類番号に対応している．これ以降の記載に於いても同様である．

【事例M4】(O.S.君)
1）読込，改行，印刷の中身が「単一処理」又は「モジュール」として仕訳けされていない．

【事例1】【事例2】
2）「反復構造」（ループ）の表記法と「分岐構造」の表記法の約束．

【事例M3】(N.T.君)
2）「流れ線」の行き止まり．「モジュールの出口」が2つになる可能性有り．

【事例M2】(O.K.君)
2）無原則的な矢印(「流れ線」の交叉，「無条件分岐」(GO TO文))が流れ図に記載

される.

【事例3】

2)「処理」と「定義済」の区別が成されていない.

【事例】

2) 同じ処理を繰り返し描かないようにする(掲載「流れ図」なし).

【事例4】

2) iとjの入替えの手順

## II. 手順の構成

手順の構成に当たっては先ず,手順に関する大局的構造を把握しておかなければならない.大局的構造と言っても,言語処理系のもとでの処理手順である事を常に意識しておく必要がある.「大局的構造」としては,次のような内容をおさえておくと良い.

1)「トップダウン」及び/又は「ボトムアップ」による全体的手順の流れ(主プログラム)

2) 制御構造(「順次構造」,「分岐構造」,「反復構造」)

●特に「繰り返し構造」(=「反復構造」)

3)「手続き」(モジュール[定義済記号])の使用

●「手続き」の仕訳或いは抽出と「手続き」に付与されている実体

【事例M4】(O.S. 君)

1) 言語処理系環境下に於けるプログラム全体の流れが理解出来ていない.表現し切れていない.

【事例M2】(O.K. 君)【事例M3】(N.T. 君)

1, 2) 主プログラムに於ける「反復構造」が理解されていない(或は,不注意,或は,頭に入っていない)ために,繰り返される手続き「主処理」の中に,ループ構造を設定してしまう.

【事例3】

3) 主プログラムで,「ループ毎にデータを読み込む」ことになっている場合にそれが手順として表現されていない.

【事例】

3) 同じ処理を繰り返し描かざるを得ない場合には,当該処理単位を「モジュール化」する(掲載「流れ図」なし).

## III.「手順の抽出」と「手順進行の契機」

手順の大局的構造が明らかになった後に,詳細な手順の検討に入る.詳細な手順は,手順分割の最小単位,即ち「単一処理(1ステップ)を抽出」する事によって構成されて行く.同時に,この検討作業の中で,「処理手順が進行する契機」となる「単一処理」や「処理条件」を設定して行く.「処理手順進行の契機」という概念と実例はアルゴリズムの論理展開を理解する上で極めて重要となる.

1)「手順の抽出」

●単一の処理(1ステップ)への仕訳

〈例〉-データ入力
-データ及び/又は処理結果の出力
-代入
-1つの計算式

●特定の言語処理系環境から要請される手順

〈例〉-ファイルの開閉
-紙送り制御(改頁[『表題,見出し』を含む],改行)

2)「処理手順進行の契機」

●「初期化」

●判定(「分岐構造」)

●繰り返し処理(「反復構造」[=「繰り返し構造」])

●「終了条件」或は,「フラグ(監視)」

-データが無くなる或は,「スタック」なり「待ち行列(キュー)」に何も繋がれていない.
-ある条件が満たされる.「=」
-ある条件が破られる.「not=」

●「制御キー」の更新

-「制御キー」の更新が必要になる時と

更新の内容.

●「モジュール」への「入」と「出」のきっかけ

—「親モジュール」の何処で「子モジュール」を引用し,「子モジュール」の何処で「親モジュール」の呼び元に戻るのか

【事例M4】(O.S. 君)

1）読込，改行，印刷の中身が「単一処理」又は「モジュール」として仕訳けされていない.

【事例】

1，2）「改頁」のし忘れ（掲載「流れ図」なし）

【事例M2】(O.K. 君)

2）ループの終了条件がない.

—従って無限ループに入ってしまう可能性がある.

## Ⅳ.「処理手順の進行」と「論理関係の展開」

処理手順の進行は，その進行と共に「アルゴリズムが有する論理関係が展開」されて行く事を意味している.

### A.「処理」に関する4つの「動き」の対応関係

ここで，処理手順の進行に伴う「動き」には,次の4つの相がある事に注意を促したい.

1）「学生のイメージ」の中の「論理的な処理」の動き

2）「流れ図」上の処理の動き

—「構造化流れ図」上の処理の動き

3）「プログラム」上の「処理」の動き

4）パソコン上の言語処理系下で実際に「データの移動」を伴った「処理の動き」

—統合環境の「監視の窓」から把握できるパソコン内部（レジスタ）の動き

「処理手順の進行」と言ったときに，一般的にはこの4つの相が混在しており，分離されていない．この段階では，1）の「学生のイメージ」の中の「論理的な処理の動き」は，2）の「流れ図」上の「処理の動き」として実体化されていなければならない．しかし，そこまで到達していない場合でも，手順に対する学生のイメージを膨らませ，「流れ図」との対応を出来るだけとるようにする．又，言語処理系環境下では，2）の「流れ図」上の「処理の動き」の基礎に，3）と4）の言語処理系から要請される「動き」(「初期化」,「ファイルの開閉」「ファイルの読み書き」など）が想定されていなければならないことにも留意する.

### B. 言語処理系下での「手順」の追跡と検証

言語処理系環境下では，構成された「手順の流れ」を「プログラムの実行」という形で追跡し検証出来る利点を有する.「手順の追跡」にあたっては，上記1），2），3），4）のそれぞれが整理され，それらの間の対応がとれていなければならず,「処理手順の進行」に伴う「論理関係の展開」の把握は，極めて「累相的な動き」に対する理解を必要としている事が分かる.

「手順」の追跡と検証で留意すべきもう一つの点は，我々が処理手順を「流れ図」上で追跡する場合,「いつでも言語処理系と同じ仕方で行っている」訳ではないということである.学生には次のような「柔軟性」と「網羅性」を持ち得るように指導すべきである：

●個々の処理の詳細を隠蔽して「処理の流れ」を追跡出来ること

●必要に応じて「詳細さ」の水準を加減出来ること

●起こり得る全ての場合を想定して「手順」の追跡を行う事が出来ること

### C.「『手順』に不定性がある場合」,「『手順』が進行しない場合」,「矛盾がある場合」を実例として，処理手順の論理を理解させる

「処理手順の進行」の過程で，手順が進行しない，或は，不定性があって一義的に「流れ」が決定されない場合には，アルゴリズムの論理関係に矛盾或は，不定性が生じている事を

意味する．アルゴリズム教育で最も動的で且つ力を入れなければならない局面はこのような論理的不整合が発生した場合である．論理的不整合の原因とその対策を考える良い機会である．この局面での実践的練習の程度がアルゴリズム的な論理思考力の成長の度合いを左右する．

論理矛盾を引起こすアルゴリズムの具体例は次のような場合である．

1）「流れ線」が交叉している（「入れ子」構造になっていない）
－「分岐構造」と「反復構造」が交叉することは無い．

2）「流れ線」に，「出口」や「無条件分岐」（GO TO 文）以外に行き止まり（袋小路に入る）がある．

3）「分岐構造」に於いては，「場合分け」の漏れがある（例外処理（0除算）を含めて全ての場合が尽くされていない）．

4）「反復構造」に於いては，「初期条件」と「終了条件」が明示されていない（無限ループに入る可能性がある）．

【事例M 2】（O.K. 君）
B．ループの度毎に「表題」が出力される．

【事例E 5】（K.Y. 君）
B．《ユークリッドの互除法》に於いて，Dowhile 型から Dountil 型に「反復構造」記述の方式を変更したときに，「論理的な手順の変更」に対応出来ていない．

【事例M 3】（N.T. 君）
B，C 1）2）●「分岐構造」と「繰り返し構造」が交叉している．
●「流れ線」に「行き止まり」がある．

【事例M 2】（O.K. 君）
B，C 1）2）4）無原則的な矢印（「流れ線」の交叉，「無条件分岐」（GO TO 文））が流れ図に記載される
●無限ループに入る可能性がある．
●「構造化プログラミング」の書式が「流れ」を検討させ，必要な条件を書き込ませる．

【事例E 5】（K.Y. 君）【事例E 8】（M.M. 子）
B，C 3）起こり得る全ての場合を想定していない．
●「0による除算」が出てこないような流れ図にしなければならない．《ユークリッドの互除法》

【事例E 7】（M.H. 子）【事例E 9】（M.M. 子）
【事例E 10】（O.K. 君）
B，C 3）4）「分岐構造」の場合に，考えられるすべての場合が尽くされていない．
●入力データが i ＝ j のとき無限ループに入る．《ユークリッドの互減法》

## V．「構造化プログラミング」の理解

「構造化プログラミング」，或いは「構造化流れ図」は，「言語処理系によるコーディング」，或いは「流れ図の表記」に対して「構造化」という制約を課すことによって，処理手順に論理的不整合が発生する可能性を排除しようとするものである．「構造化流れ図」は，その制約を更にパターン（型）化することによって，整合的アルゴリズムの視覚化を指向している．「構造化流れ図」の主要な内容は次の点である．

1）「順次構造」「分岐構造」「反復構造」とその組み合わせ
●「反復構造」で繰り返される処理内容の実体を理解する．
－「反復構造」と「分岐構造」の論理上の動きを構造化されない低次の記号を使って一度図解することが必要な場合もある．

2）極力「『無条件分岐』（GO TO 文）」を使用しない．

3）流れ図に於ける「流れ線」は，「上から下へ」，「左から右へ」が原則である．

4）「モジュール」（定義済処理）は，「1つ

の入口」と「1つの出口」をもつのが基本である.

5)「流れ線」は交叉することはない(「入れ子」構造).

●「分岐構造」と「繰り返し構造」が交叉してはいけない.

【事例M3】(N.T. 君)

1) 主プログラムに於ける「反復構造」が理解されていない(或は, 不注意, 或は, 頭に入っていない) ために, 繰り返される手続き「主処理」の中に, ループ構造を設定してしまう.

【事例E8】(M.M. 子)【事例E9】(M.M. 子)

1)「反復構造」がよく理解されていない.
《ユークリッドの互減法, 互除法》

−「反復構造」の表現上の約束がよく把握されていない.

−ループのなかにデータ入力が入ってくる.

−「反復構造」の中に,「単一処理による繰り返しの記述」がなされている.

【事例M2】(O.K. 君)

2) 5) 無原則的な矢印(「流れ線」の交叉,「無条件分岐」(GO TO 文)) が流れ図に記載される.

【事例M3】(N.T. 君)

4) モジュールに出口が2つになる危険性がある.

【事例M3】(N.T. 君)

5)「分岐構造」と「繰り返し構造」が交叉している.

## Ⅵ.「構造化プログラミング」の有効性

「構造化流れ図」は処理手順をパターン(型)として視覚化することによってアルゴリズムの整合性を実現しようとするものである. 従って,「構造化流れ図」によって,「手順の構成」が明確になると同時に「手順の不備」の検証が容易になる:

1) 図形として整然とした流れ図が可能となる

●手順の把握が正確になる.

2)「分岐構造」の網羅性を意識する事が出来る

●特に「多重選択構造」に於いて

3)「反復構造」(ループ構造) が明示される.

●ループ脱出条件が明示されなければならない.

4)「入れ子構造」が明確になる

●多重ループの概念が自然に形成される.

−入れ子構造が自由に交叉することは許されない.

−入口と出口の条件のチェックが強制的に設定される.

5) 論理的な不整合の検証が容易である

●「分岐構造」と「繰り返し構造」の交叉など

6)「モジュール化」が容易である

●モジュールは,「入口1つ」,「出口1つ」となっているで, ステップの流れに「飛び」がない

−モジュール間の依存関係が強制的に断ち切られる (小さく出来る).

●同じ処理を繰り返し描かない.

【事例M2】(O.K. 君)

1) 3)・「構造化流れ図」が徹底していなかったために, 無原則的な「流れ線」や「矢印」の記載が抑制されなかった.

●その結果, ループの表記法に従わないループが発生し,「終了条件」の記載が強制されなかった.

−従って無限ループに入ってしまう.

−構造化プログラミングに従って描くと, 終了条件を書かざるを得ない.

【事例E7】(M.H. 子)【事例E9】(M.M. 子)
【事例E10】(O.K. 君)

2) 3)「分岐構造」の場合に, 取り得るすべての場合が尽くされているかが検証さ

れ，「反復構造」の「終了条件」によってｉ＝ｊのときに発生する無限ループが検出されていた筈である．

【事例Ｍ３】（N.T. 君）

1）5）「構造化流れ図」を描くことによって「手順のあいまいさ」と「論理的な不整合」（「分岐構造」と「繰り返し構造」の交叉）が明示されている．

【事例Ｍ３】（N.T. 君）

1）6）モジュールに発生している２つの「出口」（或は，「１つの出口」と「流れ線の行き止まり」）は「構造化書法」としては図形的に明示され，チェックされていた筈のものである．

## Ⅶ．アルゴリズム教育上の留意点

Ⅵまでに述べて来た「処理手順」を構成する各段階は，アルゴリズムを理解するための重要な相を形成している．そのような各段階に特徴的な「手法」や「考え方」や「論理性」の提示と指導は，学生の理解度或いは到達度に応じてなされなければならない．ここで「アルゴリズム教育」上の留意点をあらためて纏めてみよう．

1）理解度の水準を的確に把握し，それに応じた指導を行う．

ｉ）論理的手順を考える契機となるような局面に注意を払う，或は，そのような局面を設定する．

ii）言語によるプログラミング文法の理解度とアルゴリズムの理解度の対応に留意する．

iii）パソコン上の言語処理系の「動き」に関する認識の度合いを把握しておく．

iv）意味の分かる間違いとそうでないものとを識別する．

●内容を理解していないと出来ない間違い

●理解度の低い流れ図

2）「『流れ図』による手順の構成」と「コーディング」の双方向の練習をする．

ｉ）表現された手順をコーディングする

●コーディングの約束事に従がうこと

●図形化された手順をコード化する．擬似言語風に書く

ii）コーディングされたものから「手順」を描く

●「構造化流れ図」として描く

●擬似コード風，或は，簡易言語風に書く

3）自然言語或いは擬似コード，簡易言語で記述された「手順」と「流れ図による表現」との対応を双方向で行う．

4）或る言語処理系に固有の内容と言語処理系に依らない一般的な内容とに分けて考えるべきである．

－言語処理系の選択

5）出来上がった流れ図を一度パソコンになったつもりで追跡してみる．

－間違いを探すコツ

－「パソコンになったつもりの追跡作業」には，Ⅳで述べたような「累相的な動き」が付随している事と，手順を進行させるための「柔軟性」と「網羅性」とが要求される事に留意する．

6）「流れ図」をコーディングしたプログラムを実際に言語処理系で動かしてみる．

●言語処理系が統合環境を保持しているのならば極力利用する．

（※）授業では「COBOL/2」を使用している．

【事例Ｅ１】【事例Ｅ２】【事例Ｅ３】【事例Ｅ４】

1）ｉ）「最大公約数」を求める手順を，《ユークリッドの互減法》と《ユークリッドの互除法》の２法で構成し，それらを「反復構造」の「Dowhile 型」と「Dountil 型」の２つの型を用いて表現する．

－これによって条件が変化したときの論理手順の変更を理解することを目的と

する.

【事例】

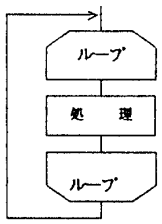
1） iv） ●比較的よく理解している.

【事例E 6】（N.H. 君）

●アルゴリズムについてある程度内容を理解していないと出来な

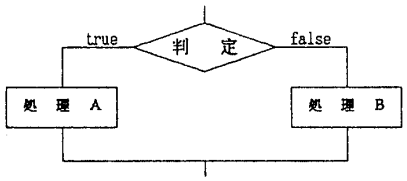
何人かの人にとっては、ループ構造ははじめは分かりずらいようである。

【事例1】



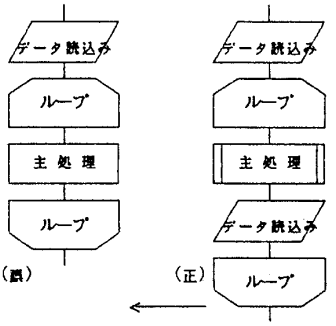

（誤） ループ構造の外側にループを回してしまう。

【事例2】



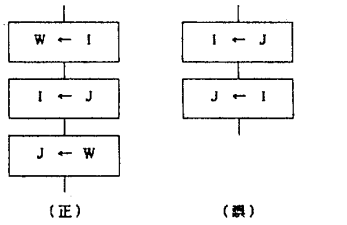

（基本型）分岐構造はこのような図形を基本型とする

【事例3】



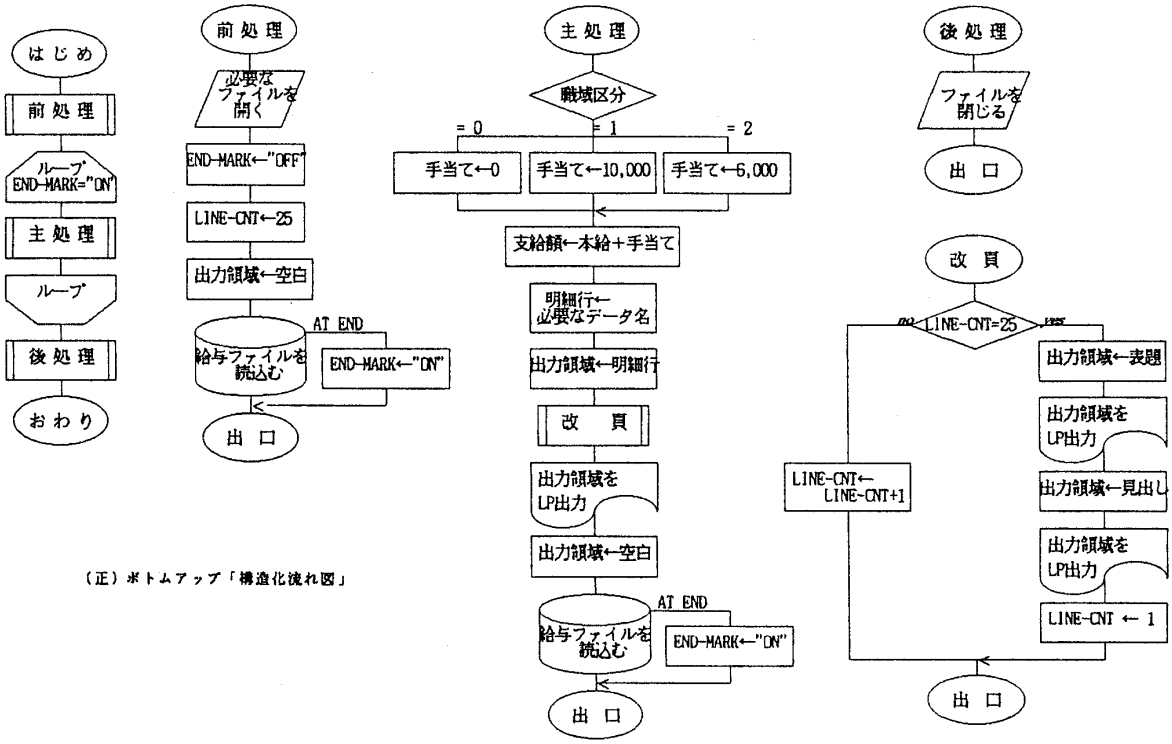

ループの中のデータ読込みが欠落する場合がある。

【事例4】




i と j の入換えのアルゴリズムが違っている。

【事例M 1】給与支給額計算




（正）ボトムアップ「構造化流れ図」

図1−1 「流れ図」事例 【事例1】～【事例4】, 【事例M 1】

い間違い.

【事例M２】(O.K. 君)【事例Ｅ５】(K.Y. 君)【事例Ｅ７】(M.H. 子)【事例Ｅ 10】(O.K. 君)

●理解度の低い「流れ図」

【事例M３】(N.T. 君)【事例M４】(O.S. 君)【事例Ｅ８】(M.M. 子)【事例Ｅ９】(M.M. 子)

【事例M１】

3)「給与支給額の計算」という一纏まりの処理手順を，与えられた処理環境のもとで，与えられた処理条件を満たすように構成する練習が必要である.

【事例M３】(N.T. 君)【事例M４】(O.S. 君)

1) ii) iii), 4) コボルに固有の事であるが,「転記」が記されていない.

【事例M２】(O.K. 君)

5) ●ループの度毎に「表題」が出力される.

●無原則的な矢印(「流れ線」の交叉,

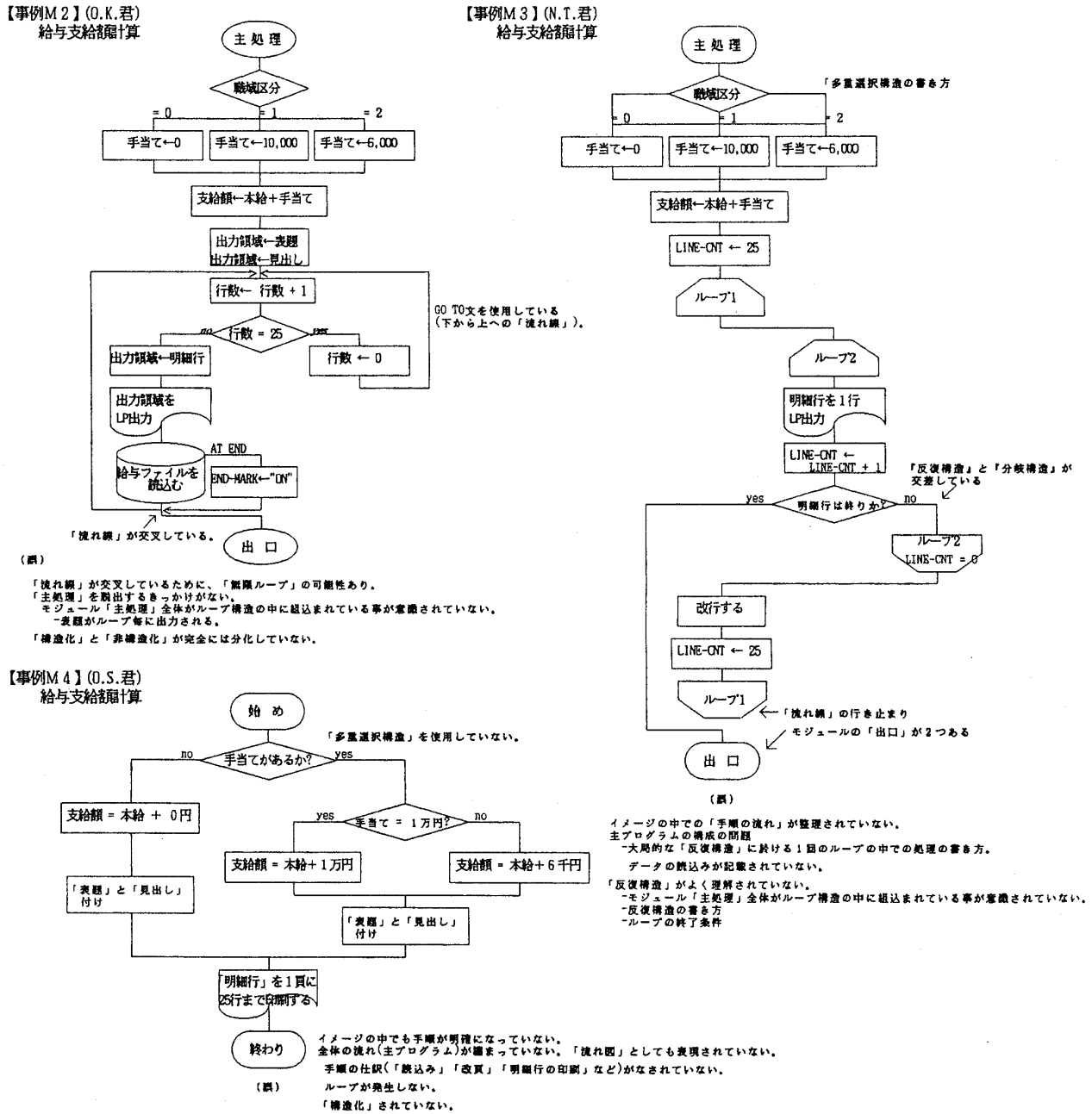
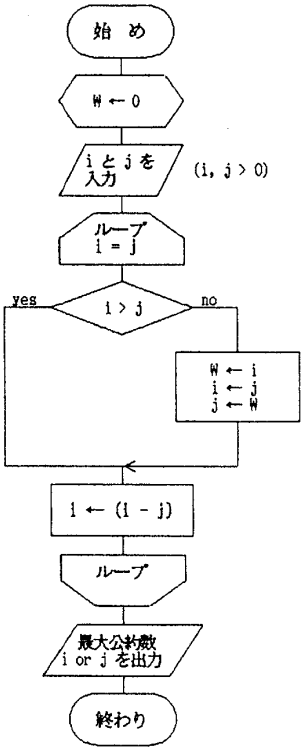
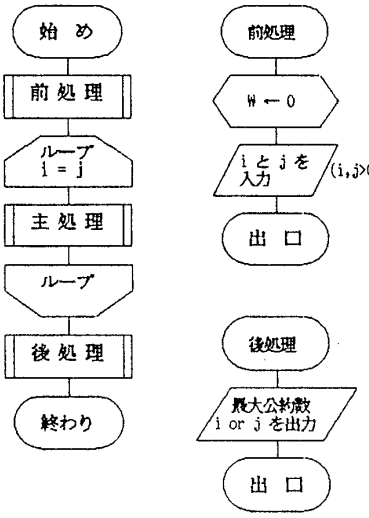



図1−2　「流れ図」事例　【事例M２】〜【事例M４】

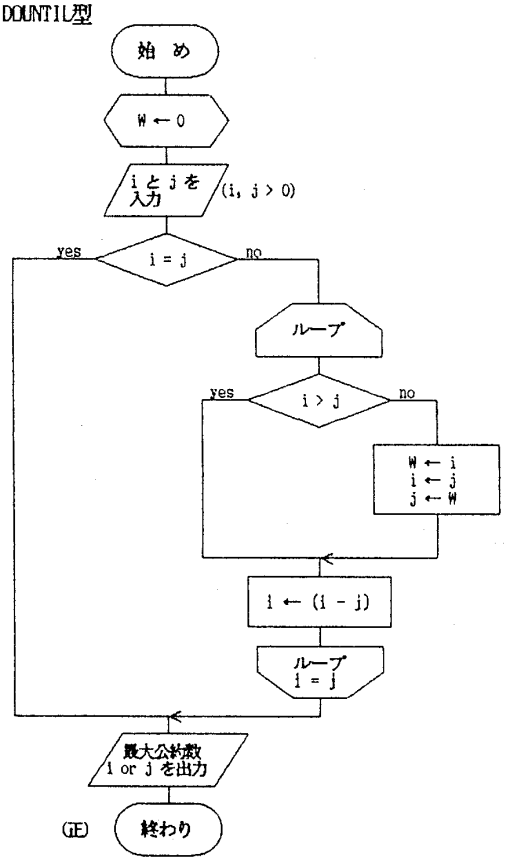
【事例Ｅ１】
Euclidの互減法
DOWHILE型



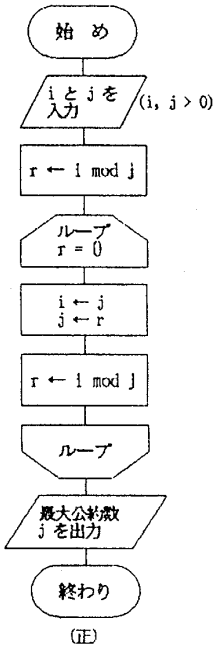




(正) ボトムアップ的構造化プログラミング

【事例Ｅ２】
Euclidの互減法
DOUNTIL型




【事例Ｅ３】
Euclidの互除法
DOWHILE型




【事例Ｅ４】
Euclidの互除法
DOUNTIL型

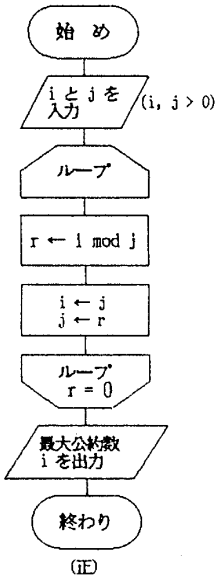
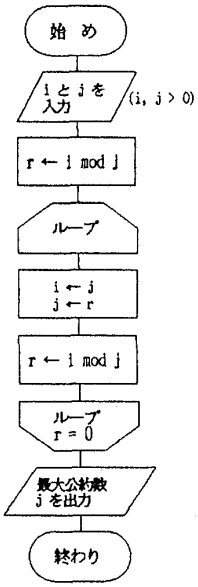



図１−３ 「流れ図」事例 【事例Ｅ１】〜【事例Ｅ４】

【事例Ｅ５】K.Y.君

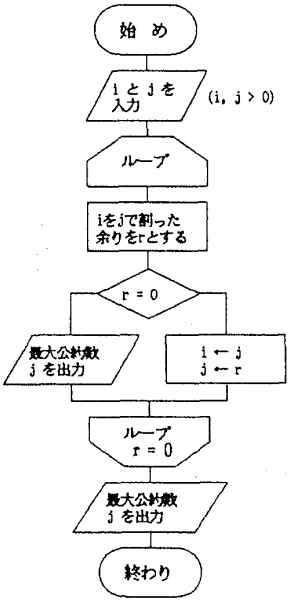
Euclidの互除法
DOUNTIL型




(誤)　入力データｉがｊで割り切れるとき、0除算エラー。

【事例Ｅ６】N.H.君

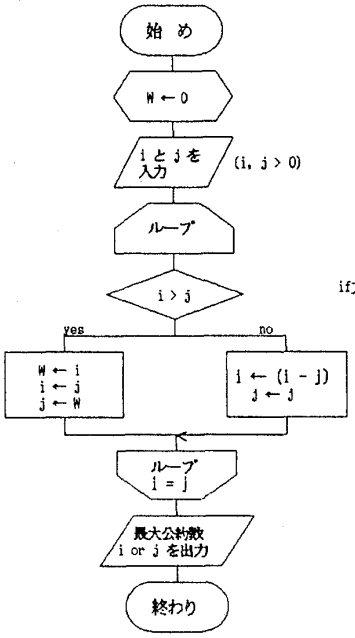
Euclidの互除法
DOUNTIL型




(少し無駄がある)

【事例Ｅ７】M.H.子

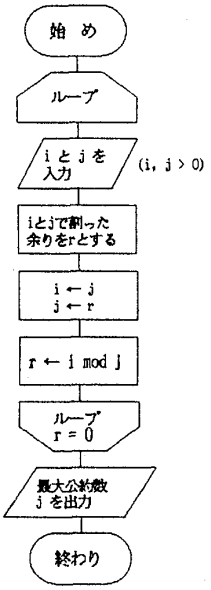
Euclidの互減法
DOUNTIL型




(誤)　入力データがi = jの時、無限ループに入る。

【事例Ｅ８】M.M.子

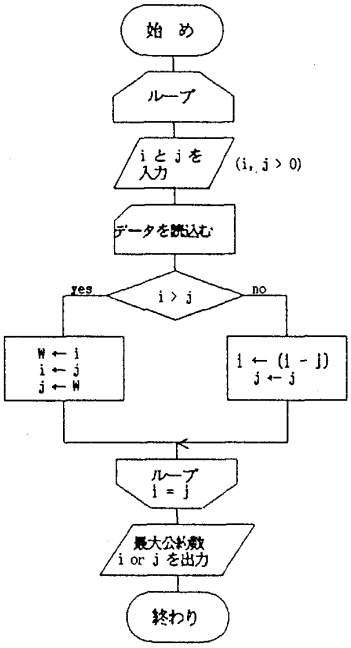
Euclidの互除法
DOUNTIL型




(誤)　反復構造がよく理解されていない。
ループの中にデータ入力が入っている。
入力データｉがｊで割り切れるとき、0除算エラー。

【事例Ｅ９】M.M.子

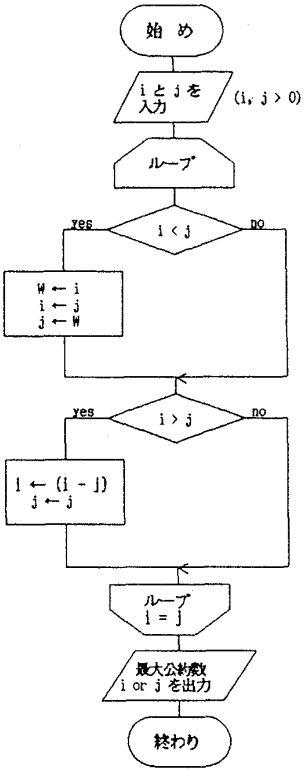
Euclidの互減法
DOUNTIL型




(誤)　反復構造がよく理解されていない。
ループの中にデータ入力が入っている。
入力データがｉ = ｊの時、無限ループに入る。

【事例Ｅ１０】O.K.君

Euclidの互減法
DOUNTIL型




(誤)　入力データがｉ = ｊの時、無限ループに入る。
2番目の if 判定は無駄である。

図1－4　「流れ図」事例　【事例Ｅ５】～【事例Ｅ 10】

「無条件分岐」(GO TO 文)) によって無限ループに入る可能性がある.

## 3. 議 論

この節では,「応用プログラミング論(B)」を履修している学生を例にとり, それぞれの学生の「情報処理 (プログラミング言語, 言語処理系の操作, 計算機 (パソコン) との接し方, 等) に関する知識の度合い」や,「学習態度」と「アルゴリズムの理解の程度」がどのように関わっているかを,『履修者の寸描』として個別に見てみよう.

〈O.S. 君〉(履修を途中放棄する)

- プログラム全体の流れが理解出来ていない. 表現しきれていない.
- 読込, 改行, 印刷などの処理の中身が手順に仕訳けされていない.

〈M.H. 子〉(プログラムの完成, アルゴリズムの理解の進展に自信と喜びを感じ取っている.やるべきこと(課題など)をしている.積極的に質問する. 同僚の遭遇している問題を一緒に考えている.「流れ図」をワープロを利用して線画で描くなどにも積極的反応を示す)

- 手順の論理展開の把握に手応えを感じている.
- 構造化プログラムについて例題を手掛けると良い.

〈K.S. 君〉(継続する力薄弱 [体力? 健康上の理由?]. 途中脱落 3 度目)

- 基礎力の不足.
- 大局的手順を描く練習が殆ど出来ていない.

〈K.Y. 君〉(再履修だが, 努力している. 独りで考える. 質問はする)

- 言語処理系の知識が不足している.
- 手順についてもう少し考察を深くすべきである.

〈O.K. 君〉(のめり込むタイプ. 物分かりが早い. あまり積極的には質問はしない. 同僚の相談に応じながら自分も考える)

- 言語処理系固有の思想と文法事項について整理しておくべきである.
- 手順の論理展開について潜在的な理解力はある.「構造化表記」の例題を手掛けると良い.

〈N.H. 君〉(バランスのとれた思考をする. 同僚と一緒に考える)

- 手順の論理展開に無理なく対応出来ている.

〈N.T. 君〉(プログラミングに関しては覇気がない. 頑張りと集中力がない. 質問をしない. 始め同僚を頼りにしていたが, その後授業時の態度としては若干内向的になる)

- 言語処理系の文法知識不足.
- 手順の論理展開に興味を持ち得ていない.
- 処理手順のイメージを整理し, それを「制御構造」によって図示する練習をすべきである.
- パソコン上の言語処理系に向かう時間を増やす必要がある.

〈M.M. 子〉(意欲が少し乏しい. 最初から「手順の論理的思考はそこそこに」と楽な立場をとっている処がある. 質問を自分からはしない)

- 「文法」や「構造化書法」の約束ごとを明確にしておかなければならない.
- 「反復構造」などの制御構造についての「自分なりの納得」が出来るようにしておくことが必要がある.
- パソコン上の言語処理系に向かう時間を増やすこと.

## 4. アルゴリズムの応用問題

「演習 II」では,「販売管理システム売上エントリ処理」を行うシステム設計に取組んでいる. システム設計はいろいろの工程から成立っているが, この節では, グラフィカル・ユーザ・インタフェース(GUI) として「演習 II」で手掛けている「画面遷移」を取上げ, アルゴリ

1)

初期(スタンバイ)画面

ここで本システムの概要と諸注意 が述べられる.

画面遷移の制御方法を会得して下さい。
「関数キー」によるものと「数字入力」によるものとの両用が使用されています。

尚、この画面では、「色付け」、「高輝度」、「反転表示」、
「点滅(ブリンク)」、「罫線による枠付け」がなされています。
参考にして下さい。

復改キー: 次頁(メニュー画面)表示
PF9 : 終了する

関数キー(PFキー)を押して下さい

2)

主選択(メニュー)画面

PF1 第1班メニュー表示
PF2 第2班メニュー表示
PF3 前頁表示
PF9 終了

関数キー(PFキー)を押して下さい。

3)

この画面は、1班の選択画面です。

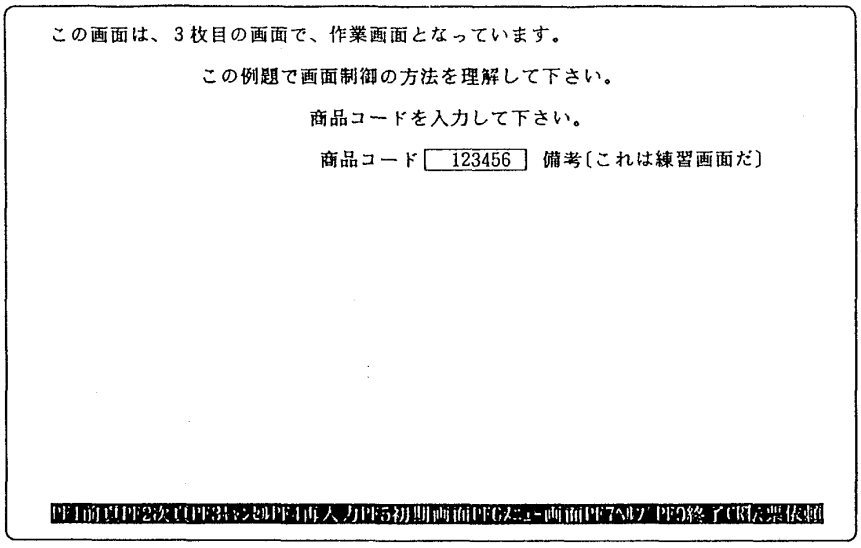
サブ選択画面

1 受注エントリ画面表示
2 前頁表示
3 キャンセル
4 初期画面表示
5 メニュー画面表示
6 ヘルプ表示
9 終了

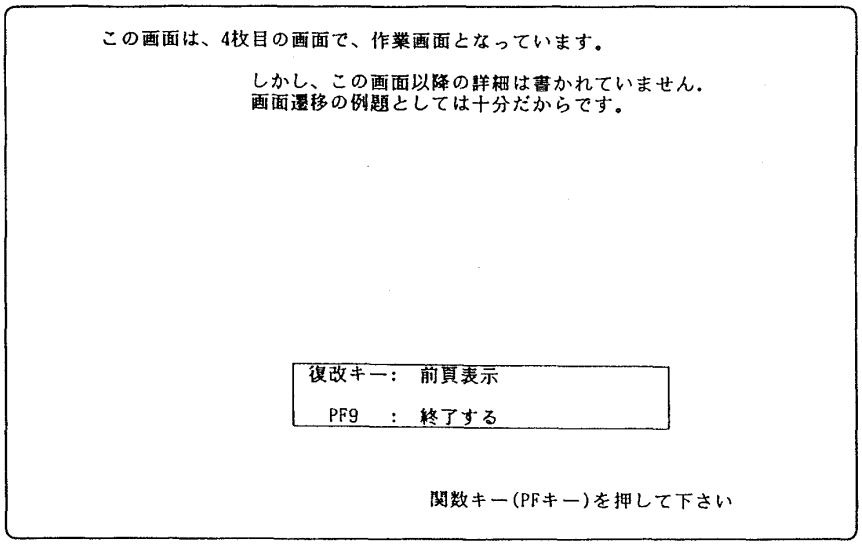
処理する番号を入力して下さい。
====> [1]

図2−1 「販売管理システム売上エントリ処理」における「画面遷移」のハードコピー 1)〜3)

4)




5)




6)

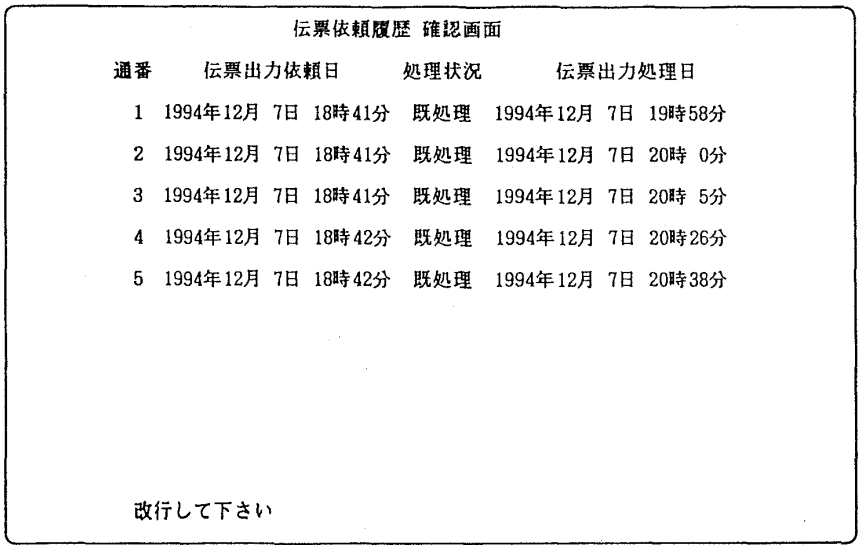



図2−2 「販売管理システム売上エントリ処理」における「画面遷移」のハードコピー 4)〜6)

7）

伝票依頼履歴 確認画面

| 通番 | 伝票出力依頼日 | 処理状況 | 伝票出力処理日 |
|---|---|---|---|
| 36 | 1994年12月13日 14時 3分 | 未処理 | 1900年 0月 0日 0時 0分 |

データがなくなりました。

伝票出力依頼を行いますか? (Y/N) Y

伝票依頼の通番を入れて下さい。037

伝票出力依頼を追加しました (1995年 1月26日 19時47分)
14ﾒｯｾｰｼﾞを送りました 宛先 HOKUSEI/Z00102 (ｽﾃｰｼｮﾝ 1)

送信元 Z00102[1]: 伝票出力依頼 95/01/26 19:47 第A-1班 (クリア CTRL-実行)

8）

この画面は、2班の選択画面です。

サ ブ 選 択 画 面

1 マスタファイル作成画面
2 伝票出力画面表示
3 前頁表示
4 キャンセル
5 初期画面表示
6 メニュー画面表示
7 ヘルプ表示
9 終了

処理する番号を入力して下さい。
====> [2]

9）

伝票依頼履歴 確認画面

| 通番 | 伝票出力依頼日 | 処理状況 | 伝票出力処理日 |
|---|---|---|---|
| 1 | 1994年12月 7日 18時41分 | 既処理 | 1994年12月 7日 19時58分 |
| 2 | 1994年12月 7日 18時41分 | 既処理 | 1994年12月 7日 20時 0分 |
| 3 | 1994年12月 7日 18時41分 | 既処理 | 1994年12月 7日 20時 5分 |
| 4 | 1994年12月 7日 18時42分 | 既処理 | 1994年12月 7日 20時26分 |
| 5 | 1994年12月 7日 18時42分 | 既処理 | 1994年12月 7日 20時38分 |

改行して下さい

図2−3 「販売管理システム売上エントリ処理」における「画面遷移」のハードコピー 7)〜9)

10)

伝票依頼履歴 確認画面

| 通番 | 伝票出力依頼日 | 処理状況 | 伝票出力処理日 |
|---|---|---|---|
| 36 | 1994年12月13日 14時 3分 | 未処理 | 1900年 0月 0日 0時 0分 |
| 37 | 1995年 1月26日 19時47分 | 未処理 | 1900年 0月 0日 0時 0分 |

データがなくなりました。

これ以上レコードはありません
改行して下さい

11)

伝票依頼履歴 確認画面

| 通番 | 伝票出力依頼日 | 処理状況 | 伝票出力処理日 |
|---|---|---|---|
| 31 | 1994年12月 9日 15時21分 | 既処理 | 1994年12月13日 14時 5分 |
| 32 | 1994年12月10日 17時43分 | 既処理 | 1994年12月15日 20時 1分 |
| 33 | 1994年12月10日 19時30分 | 既処理 | 1994年12月26日 22時 8分 |
| 34 | 1994年12月12日 19時23分 | 既処理 | 1994年12月26日 22時 9分 |
| 35 | 1994年12月12日 19時47分 | 既処理 | 1995年 1月26日 17時16分 |

伝票出力を実行しますか? (Y/N) Y

伝票出力の処理を致しました (1995年 1月26日 19時58分)

処理結果を更新しました。

**図2−4 「販売管理システム売上エントリ処理」における「画面遷移」のハードコピー 10)〜11)**

ズム教育の応用例としたい.「画面遷移の制御」は, アルゴリズム教育の点から好ましいテーマの一つとなるように思われる.「画面遷移の手順」が, 第2節で述べたアルゴリズム理解の諸相とよく対応しているからである. この教材は,「構造化プログラミング」,「構造化流れ図」を検討する例題としても興味深い.

図2は,「演習II」で現在も作成が進行している一連の「画面遷移」のハードコピーを掲載したものである.「販売管理システム売上エントリ処理」の設計は2班に分かれて行われている. 第1班は,「受注エントリ処理」, 第2班は,「マスタファイル作成及び売上伝票出力処理」を受持っている. 図2の画面進行に則して画面種を説明する. 各画面種の説明の後,[ ]内に当該画面で選択されたキー或は, 数字を示す.

1) 初期画面(スタンバイ画面) [復改キー押下]
2) 第1班, 第2班共通のメニュー画面(主選択画面) [PF1 キー押下]
3) 第1班のメニュー画面(サブ選択画面) [〝1〞 選択]
4) 第1班:「作業画面」及び「次処理選択用状態表示行」[PF2 キー押下]
5) 第1班:「次の作業画面」と「画面遷移

メニュー」

6）第1班：「伝票依頼履歴確認画面」[4）画面で復改（CR）キー押下]

7）第1班：「第2班に対する伝票依頼実行画面」[当該画面で〝Y〟選択]

8）第2班のメニュー画面(サブ選択画面)[2)画面でPF2押下][当該画面で〝2〟選択]

9）10）第2班：「伝票依頼履歴確認画面」

11）第2班：「伝票出力実行及び処理確認画面」[当該画面で〝Y〟選択]

《画面遷移の手順》

1）「処理手順」が，一連の「画面遷移」の「動き」として把握し易い．

2）処理のステップが「画面内の表示」と「画面遷移」とに単位化される．
－従ってモジュール化も容易である（1画面が1モジュール）．

3）一連のステップを手順として論理化して行く為の「契機の抽出」が基本的にはモジュール(処理単位)への「入」とモジュールからの「出」で明確に規定されている．

4）「手順の誤り」は，「画面遷移の失敗」として視覚的に明示される．誤りの箇所も特定され易い．

「画面遷移」に関する手順の表記は，「従来の『流れ図』」と「構造化『流れ図』」とを対比させて検討するのには恰好の教材である．「画面遷移」は，処理手順として「無条件分岐(GO TO文)」の多用が自然のように見える

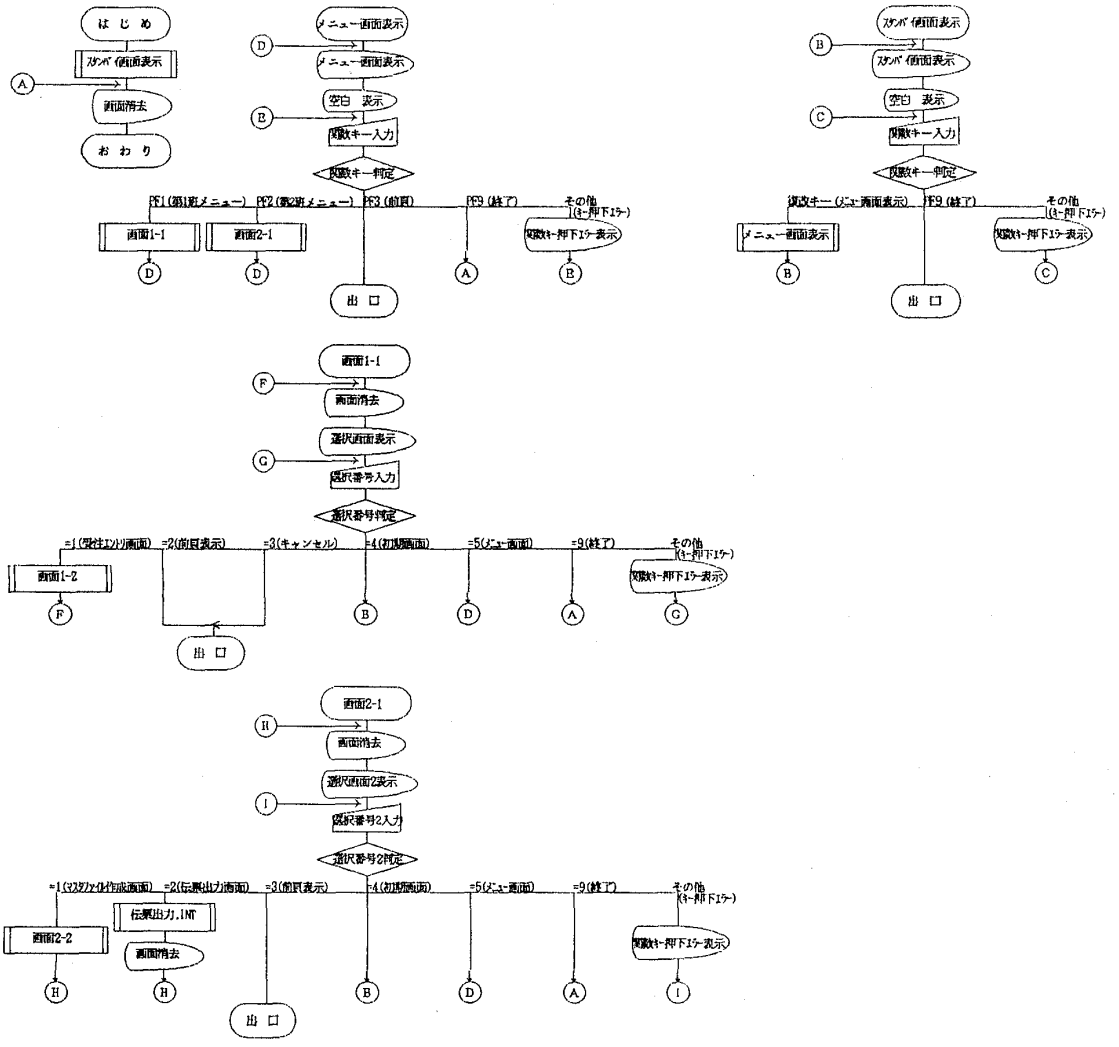



図3－1 「売上エントリ処理」のための「従来の流れ図」

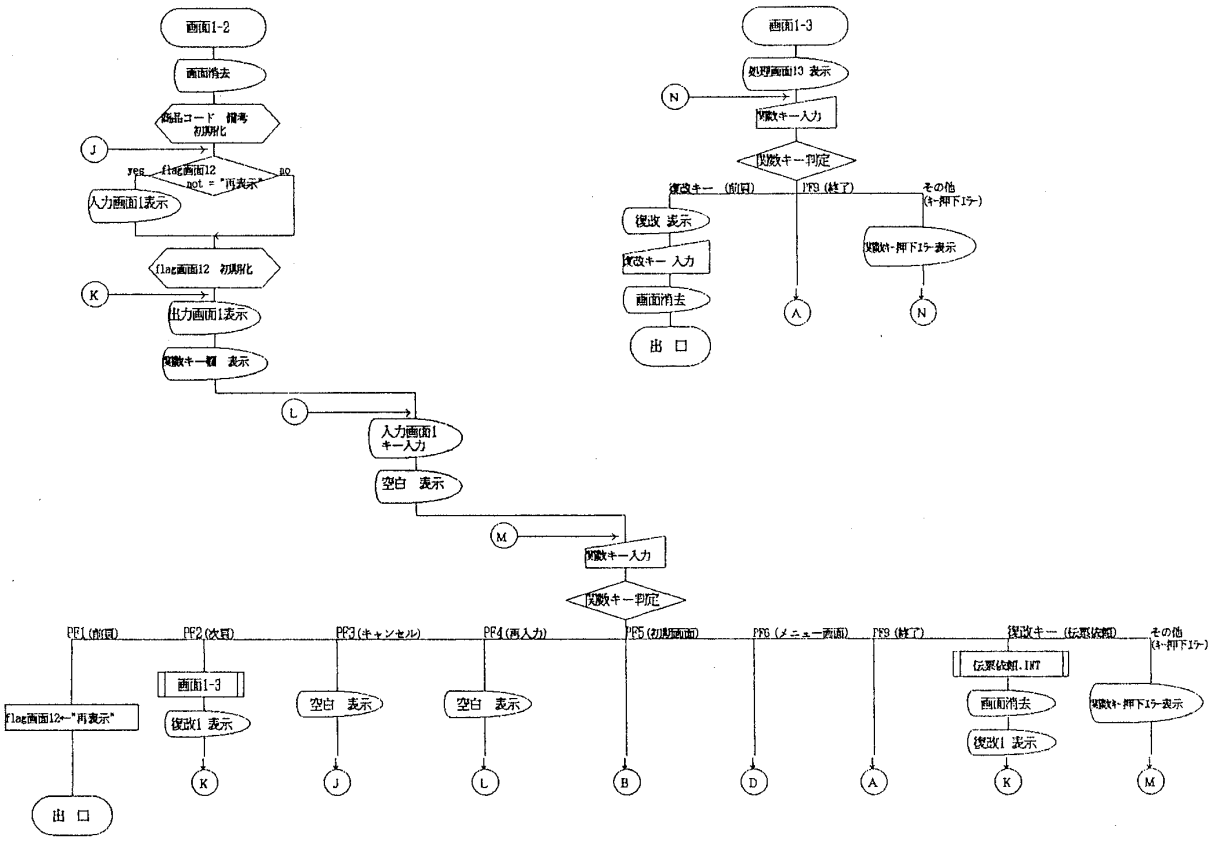



図3−2 「売上エントリ処理」のための「従来の流れ図」

かも知れないが，「構造化手法」に基いて記述する場合との得失を考えてみよう．

図3と図4には，図2に示したような画面遷移を制御する「処理手順」に対する「流れ図」を示してある．図3は「従来の『流れ図』」によって描かれている．図4は「構造化流れ図」によって描かれている．図3と図4の2つの「流れ図」を対比しながら「画面遷移の手順」に対する《表現法の違い》を検討する．

《「画面遷移の手順」の表現法の比較》

◎「構造化流れ図」と「従来の『流れ図』」

〈「従来の『流れ図』」〉

1．自由奔放な「流れ図」となる．

●「GO TO 文」の使用によって，「流れ図」は，容易に，或いは，安易に描けてしまう．

－このような構造を「流れ線」を使用して描くのは不可能である．

●「処理単位」（必ずしも「モジュール」と言う訳ではない．例えば「多重分岐」）に於ける「出口」が複数箇所ある．「処理単位」への「入口」も自由である．

－「処理単位」への「入」と「処理単位」からの「出」の条件の検査がおろそかになる（無原則）．

2．「GO TO 文」の使用によって，「反復構造」の概念が成長しない．

3．「入れ子構造」となっている「多重ループ」の概念がない（従って〝入れ子構造〟が〝自由に交叉〟出来る）．

4．モジュールが相互依存してしまう．

●モジュールの入口と出口が1つずつになっていない．

●無条件分岐（出口と入口）が自由．

●モジュール化がしにくくなる．

〈「構造化流れ図」〉

1．整然とした流れ図が可能となる．

2．手順の把握が正確になる（手順の「進

行条件」が明確に意識されている).

3．モジュールは,「入口１つ」,「出口１つ」となっているで，ステップの流れに「飛び」がない.

4．モジュール間の依存関係が強制的に断ち切られるので，部品化(モジュール化)が容易である.

5．「反復構造」(ループ構造) が明示される.

●ループ脱出条件が明示されなければな

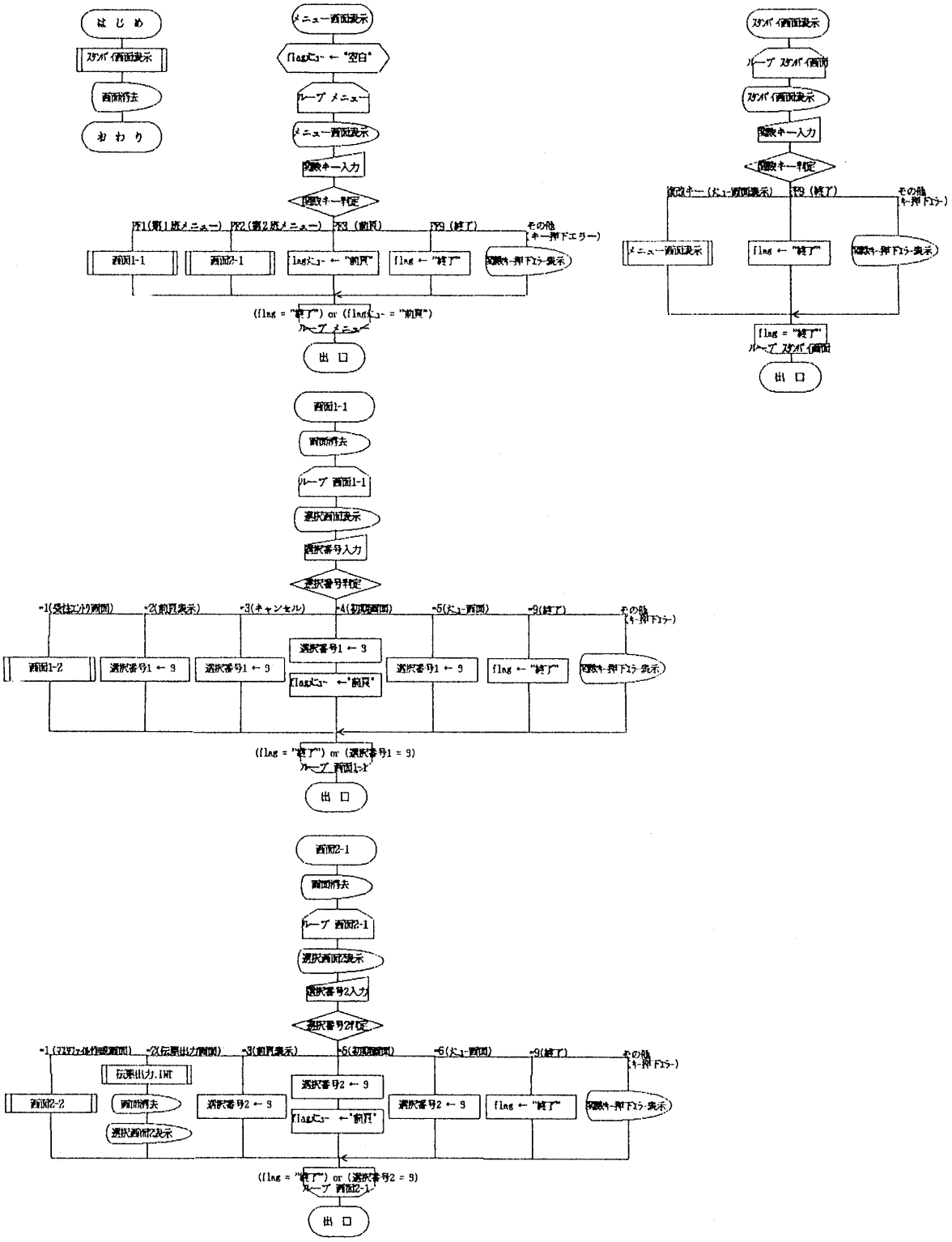



図4−1 「売上エントリ処理」のための「構造化流れ図」

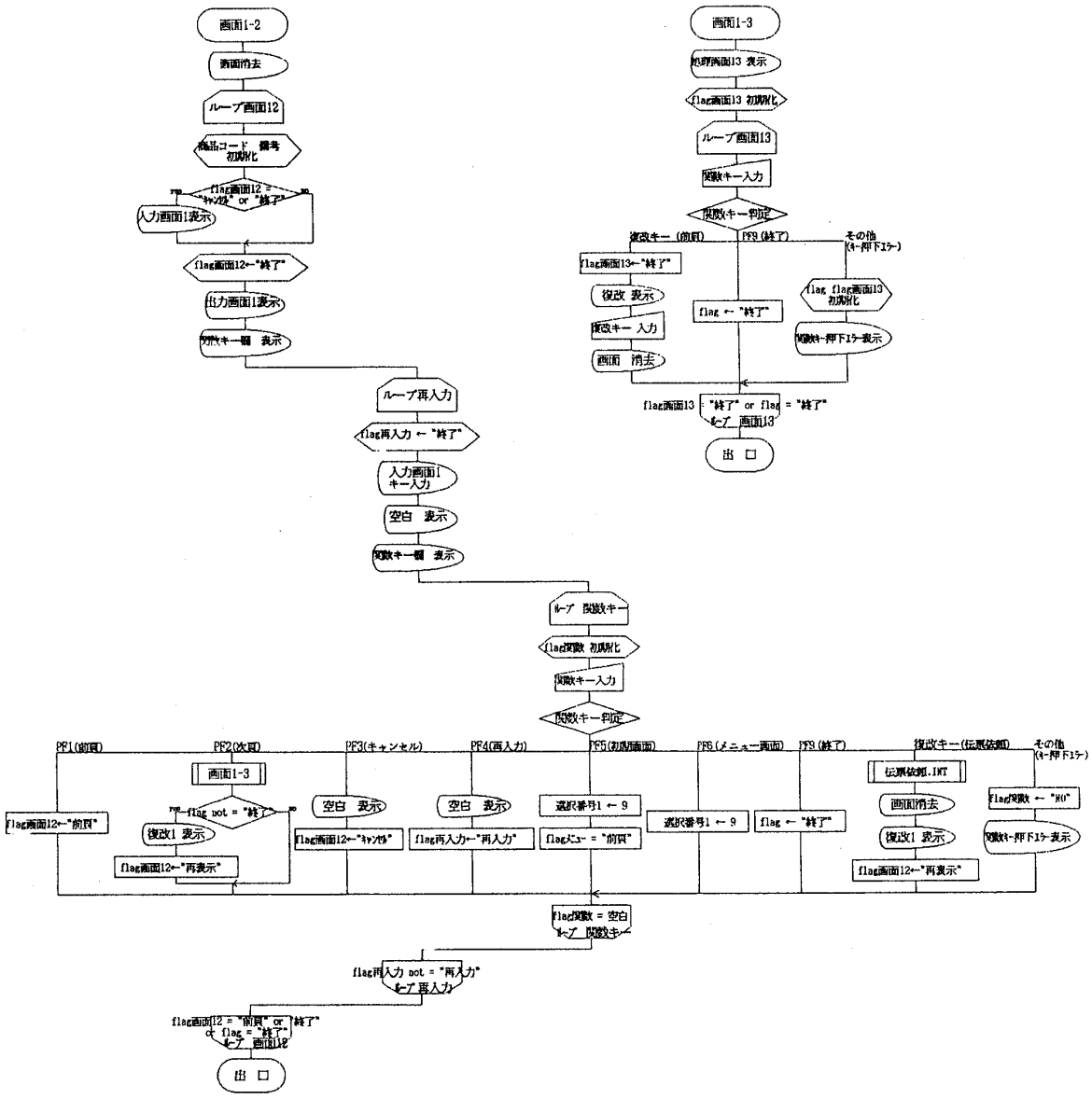



図4－2 「売上エントリ処理」のための「構造化流れ図」

らない.

6．「入れ子構造」となっている「多重ループ」の概念が自然に形成される.

- ●「入れ子構造」が自由に交叉することは許されない
- ●「入れ子構造」の「入口」と「出口」の条件が強制的に設定される.

以上の比較からも分かるように,「構造化流れ図」によって,「画面遷移の手順」が整然と然も「制御構造の累層的な動き」, 例えば「入れ子構造」になっている「反復構造」, 或は,「反復構造」の中での「多重選択構造」の「開始と終了」或は,「入口と出口」の条件を明示しながら進んでいる様子が視覚的に良く理解出来る.

## 5．おわりに

本稿は,「言語処理系」環境下で著者が担当している授業と実習を基に,「アルゴリズム教育」について論じたものである.

「処理手順」或は，「アルゴリズム」の理解には，それぞれ特徴的な「考え方・着眼点」や「手法」や「論理性」を必要とする「段階」或は，「相」があることを指摘した．「誤例」として引用された「流れ図」に表現されている「処理手順」が，アルゴリズム理解のどの段階に錯誤があったのかを具体的に検討した．更に「処理手順」や「アルゴリズム」の構築の手段として，「構造化流れ図」或は，「構造化されたプログラミング」の「手法」を援用する事が有効であることを強調した．

学生に見られるアルゴリズム理解度の相違は，「論理的思考の深さと適応性」や「言語処理系」及び「プログラミング」に関する基礎知識や実習経験の程度に依存している．勿論，学生の「学習態度・学習意欲」もアルゴリズム理解度を左右する．授業に於いては，学生の「情報処理（プログラミング言語，言語処理系の操作，計算機（パソコン）との接し方，等）に関する知識の度合い」や，「学習態度」に即応した指導が望まれる：アルゴリズム理解の諸段階のうち，どの相に不明な点があるのかを学生に認識させること．又，アルゴリズム理解に於けるそれぞれの相では，学生が何処で行詰っているのか，何処に困難を感じているのかを実例に則して突きとめ，その局面で具体的に考えさせて行くことが必要である．

最後に別の側面からのコメントを述べて本稿を終える．

1．アルゴリズム教育に於いては，「手順の表記法」を「使える形」で理解させておく事が必要である．そのためには，「良い例題」を取り上げ，正解と学生の解答とを比較検討させるのが良い．問題にしている処理単位の表記法の説明に止まらず，当該処理単位を含む，言語処理系で即実行可能な完全な解答も用意すべきである．

2．与えられた一纏まりの処理に於いて，「手順の進行の契機」や「手順に矛盾がある場合」をよく理解させる．この場合，言語処理系が備えている「統合環境」の支援の下に，「流れ図」と実際の処理系の「動き」をよく比較検討させる事は大いに効果がある．個人で検討する以外に，グループで行っても良いが，一度は授業全体で考察してみるのが良い．

3．以上の検討作業を，より「良い例題」を使用して行なうことである．「良い例題」とは，1）学生の興味を喚起し，2）視覚に訴える題材で，3）処理手順の数が多過ぎず，4）処理手順の論理が明快なものが良い．与えられた条件の変化に応じて，「手順進行の契機」や「手順の論理展開」の対応が良く見えるものが良い．「画面遷移」はそのような例の1つであろう．

4．「構造化プログラミング」或は，「構造化された手順」の記述は，「処理手順の論理性」を「構造化された制御構造」という制約を課す事によって実現して行こうとするものである．「構造化された手順の記述」は，制御構造を記述する作業過程では，処理ステップを分析的に考察し，制御構造の組合わせと構成の作業過程では，処理単位同士を整合的に結合して処理手順を統合化することを促す枠組みとなっている．

**謝辞** この小論を纏めるに当たって，今回の「アルゴリズム教育研究ワークショップ」のための2回にわたる準備会（森田彦，原田融，新國三千代の各氏と筆者）に於ける作業と議論，及び「ワークショップ」研究会に於ける講演と討論が大変参考になりました．今回の「ワークショップ」の準備にあたられた，森田彦，原田融，新國三千代の各氏，及び「ワークショップ」研究会で講演された坂東昌子氏をはじめとする先生方，そして討論に参加された皆様方に感謝致します．

**文献**

(1) 能登 宏：プログラミング言語処理系とアルゴリズム教育，社会情報，札幌学院大学社会情報学部紀要，Vol.2，No.2，pp.87－96(1993)．