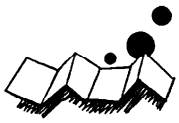
解　説

プログラム設計法 PAD/PAM†



二　村　良　彦††

1. はじめに

プログラムの設計法とは「与えられた仕様書を満足するプログラムを求めよ」という問題（これをプログラム作成問題とよぶ）を解くための問題解決手順である．従来提案されていたプログラム設計法としてはDijkstra 等のトップダウン作成法[1]と Warnier[16], Jackson[9]等のデータ分析法が有名であった．しかし，これらは，設計手順といえるほど手順化されておらず，「トップダウンに作成せよ」といったガイドラインを与えるもの（トップダウン作成法）あるいはプログラム作成問題を「データ構造の設計」といった別の問題に置き換えるもの（データ分析法）であった．すなわち，一般のプログラマがこれらの設計法を容易に実践できるほど詳細化された手順にはなっていなかった（**表-1**）．これら2つの設計法の特徴を活かし，かつその手順をより詳細化する目的で開発したものが PAM (Problem Analysis Method) である．PAM の開発思想は以下の通りである．

われわれは，Dijkstra 等と同様，プログラム作成問題を数学の問題と対応させて考えた．例えば，プログラムの仕様書が方程式に対応し，作成すべきプログラムが方程式の解に対応する．あるいはまた，プログラム作成問題が幾何の作図問題に対応し，作成すべきプログラムが作図すべき図に対応する．このように，プログラム設計法は数学における方程式の解法や作図法に相当するものであると考える．

一方，数学の問題を解く際に役立つ考え方の手順を Polya が文献13)において与えている．それは Descartes[2]等が研究した，下記の点を特徴とする，問題解決法に基づいている．

（1） 解決法が既知であるような類似の問題を見つけ，その解き方を利用する（類似性の利用）．

（2） 問題を小問題に分割し各小問題を独立に解く．そして小問題の解を統合してもとの問題の解を求める（分割と統合）．

Polya が示した手順は「これさえ知っていればどんな問題でも機械的に解ける」というほどには詳細化されたものではない．しかし「その手順を踏んで問題解決を繰り返しているうちには，その実践者の問題解決能力の向上が期待できる」ものである．

Polya の手順をプログラム作成問題に直接適用できるように書き直したものが PAM である．結果として，PAM は構造化プログラム図式 PAD (Problem Analysis Diagram) に基づいた，次の2つの特徴を有するプログラム設計手順となった．

（1） PAD によりデータ構造が記述できる点に着目し，プログラム構造の設計に際して，データ構造とプログラム構造の類似性を利用する．

（2） PAD が図形として分割と統合がしやすい点に着目し，プログラム構造の設計に際して，PAD の分割と統合を積極的に行う．

このように，PAM は PAD と一体となって使われ

† PAD/PAM: A Program Design Method by Yoshihiko FUTAMURA (Central Research Laboratory, Hitachi, Ltd.).

†† (株)日立製作所中央研究所第8部

表-1 在来のプログラム設計法の問題点[1]

方法名	与えられた指針または手順	主作業	問題点
トップダウン作成法	トップダウンにプログラムの構造を設計せよ	制御構造の設計	トップダウン設計の方法自体は与えられていない
データ分析法	(1)データ構造を記述する (2)データ構造に基づき制御構造を決定する (3)プログラムにとって必要な操作を列挙する (4)列挙した操作を(2)で決めた制御構造の中に割り付ける	入出力データ構造の設計	データ構造の設計法自体は与えられていない

ているので，われわれはこれを PAD/PAM とよんでいる．PAD/PAM についてはすでに文献3),6),7)などで詳細を発表した．そしてそれは国の内外において広く利用され，その利用者は数万名以上であると推定している．(株)日立製作所内においては過去3年間に少なくとも500万ステップ以上のプログラムが PAD/PAM に基づいて開発された[4]．プログラム開発技術の有効性を定量的に調べることは容易ではない．しかし PAD/PAM が短期間で多くのプログラマに受け入れられたという事実は，その技術の質のよさを示すものと考えられる．

本論文では PAD と PAM について概説し，最後に「プログラム設計技法の実用化と発展」シンポジウムにおける共通問題「在庫管理システム」[17]の PAD/PAM による解法について述べる．

2. 問題分析図 PAD (Problem Analysis Diagram)

PAD は流れ図に代る，新しいプログラム図式である．プログラム図式とは，流れ図や擬似コードのように計算過程を記述するための図式である．そしてその実際的目的はプログラムの設計，製造，検査，保守等の各段階において常に作業のよりどころとなることである．したがって，プログラム図式の評価基準のなかでは次の5点が重要である．

（1） プログラムの構造が見やすいこと．

（2） プログラム設計法と相性が良いこと．

（3） 図式からの製造（コーディング）が容易なこと．

（4） 図式の正しさの検査がしやすいこと．

（5） 図式が書きやすくまた修正もしやすいこと．

プログラム図式第1号は流れ図であると思われる．流れ図は1947年 Goldstein と Neumann の論文[8]で使われた．しかし1960年代の終りに Dijkstra[1]等により構造化プログラム技法が提唱されて以来，流れ図におけるプログラム構造の見にくさなどの欠点が明確になってきた．そして流れ図の欠点を解消する目的で多数の構造化プログラム用図式（本論文ではこれを構造化プログラム図式とよぶ）が提案された[4]．しかしそれらの図式はプログラム構造を見やすくするために，図の保守性を犠牲にしたり，あるいは逆に保守性のために見やすさを犠牲にしたりするものが多く，プログラマに流れ図を捨てさせるだけの魅力がなかった．

PAD は，プログラムおよびプログラムが処理するデータを3つの基本形（連接，反復，選択）に基づいて表現するための樹木状の図形である．樹木状にプログラムとデータを表現したことにより，PAD は次の3つの特徴を有する．

（1） プログラムの構造を見えるようにする．

（2） プログラムの製造と検査を系統的にできるようにする．

（3） プログラムで処理するデータの構造を，プログラムの構造と同じ表記法で記述できるようにする．

その結果 PAD は上記5基準をバランスよく満たす図式となり，他の図式よりも非常に速く世の中に受け入れられた．

PAD と他の図式との類似性，相違点等は文献4)で詳しく述べられている．PAD の詳細については文献6),7)等を参照されたい．

3. 問題分析法 PAM (Problem Analysis Method)

大規模ソフトウェアの設計過程を，システム設計（モジュールの定義）とプログラム設計（モジュールの設計）とに分けてみると，PAM は後者のモジュール設計法に関するものである．本章では，まずシステム設計法とプログラム設計法の相違点についてのべ，次に PAM の概要について説明する．

3.1 システム設計法とプログラム設計法の相違点

与えられたシステムの要求仕様書に基づいてプログラムを開発する際には，まずシステムをいくつかのモジュール（すなわちサブプログラム，サブルーチン，コルーチン，オブジェクト等）に分割し，モジュールの間の関連と仕様を明確にする．このとき，システムを分割するために用いる方法がシステム設計法であり，構造化システム設計法[11]等がよく知られている．一般にシステム設計法では分割の際の基準として，変更の容易さ (modifiability)，分かりやすさ (understandability)，モジュールの流用可能性等を最優先とする．すなわち，モジュール間で変数やコードを共有することを避けよと主張する．そして，モジュールとしての機能のまとまりや能率が多少低下しても目をつぶることにしている．これは大規模ソフトウェアを長期間保守していくために必要な，当然の基準である．このような基準に従ってシステムをモジュールへ，モジュールをさらに小さなモジュールへと分割していく．そして，モジュールが適当に小さく分割され，モジュールの独立性 (coupling) と機能としてのまとまり

表-2 システム設計法とプログラム設計法[9]

	システム設計法	プログラム設計法
分割の対象とその単位	システムをモジュールへ	モジュールをスレッドへ
対象の特徴	システムは大規模で作り直しが困難なため (1)長期間使われる (2)多数の人間の共同作業で作られる	モジュールは小さく(数十〜数百ステップ)，ハードウェアの変化の影響を受けやすいため (1)部分的に変更するよりは，作り直すほうが一般的には容易 (2)1人のプログラマにより作られる
分割の主要基準	(1)システムの保守性 (2)モジュール流用可能性 (3)モジュールの機能のわかりやすさ	(1)モジュールの作り直しやすさ (2)正しさの確認のしやすさ (3)実行速度，プログラムサイズ等の効率
最終的な産物	(1)モジュールの間の関係を表す木構造 (2)各モジュールの機能仕様書 (たとえば，HIPO)	スレッドの絡み合ったプログラム

(cohesion) のバランスがとれたところでシステム設計は終了する．したがってシステム設計法はモジュールの機能仕様を決めるだけで，「いかにしてモジュールをプログラム化するか」には言及しない[11]．システム設計の後を受けて，モジュールをプログラム化するための指針を与えるものが，本文でいうプログラム設計法である．

システム設計法とプログラム設計法との違いの詳細については，PAMの説明を読まれた後で表-2を参照されたい．

3.2 PAM の説明

PAM ではモジュールに対してスレッド (thread) という概念を使用する．スレッドは，いわば，モジュールを織り成す糸である．与えられたモジュールの仕様の中で，一体となって絡まり合っているスレッドを一本ずつほぐし，整然としたモジュールに織りあげる方法が PAM である．

PAM は，Polya の問題解決手順と同様に次の4種類の方法の規定からなる．

(1) 問題の理解 (入出力データの条件および対応の明確化)

(2) 分割 (問題の小問題への分割．小問題をスレッドとよぶ)

(3) 統合 (小問題の解答の組み立てなおし)

(4) 検証 (解答の正しさの確認)

PAM におけるプログラム作成手順は次の通りである．

[手順1] 未知のもの (求めるデータ) を明確にする．

(1) 求めるデータ (出力データ) の構造を PAD で書く．

(2) 出力インタフェースを書く．

[手順2] 与えられているもの (入力データ) を明確にする．

(1) 入力データの構造を PAD で書く．

(2) 入力インタフェース (入力パラメータなど) を書く．

[手順3] 求めるプログラムの機能を部分機能に分割する (partition)．この分割された部分機能をスレッド (thread) とよぶ．スレッドは，原則として，それを実現する PAD がプログラマの頭に浮かぶ (すなわち，直感的に分かる) か，プログラマがすでにその存在を知っているプログラム (または PAD) があるような機能である．ただし，一つのスレッドをさらに小さなスレッドに分割するつもりなら，その時点ではそのスレッドの実現法を知っている必要はない．また，もとの機能自身も，その細部を取り除いて見たときには，部分機能と考えることができる．分割の際にはスレッドの機能だけに着目し，他のスレッドとの関係 (変数やコードの共有等) については考える必要はない．分割の方法は一つには限らぬが，たとえばなにが必要な機能または量かを考え，所要機能または量をすべて挙げるまで続ける．

[手順4] スレッド間の依存関係を示すスレッド関連図[3]を書き，スレッド間の関係を整理する．必要ならば関連図が書きやすくなるようにスレッドを書き直す．すべてのスレッドは，一つの関連図上に現れていなければならない．しかし，一つのスレッドが関連図上で2カ所以上に現れても構わない．

[手順5] 一つのスレッドを選び PAD 化する．これは，プログラムの中心的構造 (例えば，一番外側のループ) を与えるものであり，メインスレッドと呼ぶ．

[手順6] 残りのスレッドを一つずつ，すでに作ら

れている PAD に統合 (merge) する．スレッドを統合する前に PAD を最適化しておいたほうが都合のよいことがある．そのときには，PAD を最適化する．この操作を，残りのスレッドがなくなるまで繰り返す．統合の各ステップの正しさは，直感的に明らかか，そうでなければ，証明によって示さなければならない．

［手順7］ PAD が完成したら，テストケース表を作成し，かつレビュー（統合の段階の検証）をする (PAD が完成すれば，プログラムは機械的に作成できる[7])．

PAM の詳細については文献3)を参照されたい．

4. PAD/PAM によるプログラム作成例：在庫管理システム

ここで例として作成する在庫管理システムは文献17)にあるものである．システムにたいしては，オペレータが積荷票や出庫依頼を入力する．そうするとシステムが出庫指示や在庫不足連絡を出力するものである．オペレータとシステムとの対話例を**図-1**に示したので，これによりシステムの大方の機能を把握していただきたい．

図-2にシステムのトップレベルの概略PADを示した．ただし，図-2の中に書かれた各種のリストや変数名の意味は**図-3**の通りである．図-3にはプログラム中で使われるデータの構造が示してある（この程度に簡単な構造はPADを用いて書くまでもない）．これら2つの図が書けて始めてスレッドへの分割の見通しが立つ．したがって，ここまでが問題の理解に相当するフェーズである．

次には分割のフェーズに入る（ただし，諸リスト等のロードやセーブの部分はここでは考えないことにする）．図-2における2つのサブルーチンをまず次の(1)，(2)のようにスレッドに分ける．次にスレッド(1.4)および(2.1)に含まれる機能を各々(3)および(4)のように更にスレッドに分ける（スレッド関連図

```
こんてなﾊﾞんごｳ, はんにゅうねんげﾂ, にちﾞ? 1,84/6,5/14:00      ┐
つみにひんめい,つみにすうりょう? BALLANTINE'S,12                  │
つみにひんめい,つみにすうりょう? DRY SACK,24                      │ 積荷票1
つみにひんめい,つみにすうりょう? Z,0 ←……積荷票の終りを示す       ┘

ちゅうもんﾊﾞんごｳ    1
いらいひんめい,いらいすうりょう,おくりさきめい? BALLANTINE'S,3,FUTAMURA  } 出庫依頼1

しゅっこしﾞ          # 1              FUTAMURA
 1                  BALLANTINE'S     3          } 出庫指示1

ちゅうもんﾊﾞんごｳ    2
いらいひんめい,いらいすうりょう,おくりさきめい? DRY SACK,30,HAMADA  } 出庫依頼2

さﾞいこﾞそくれんらく  DRY SACK          6      } 在庫不足連絡1

ちゅうもんﾊﾞんごｳ    3
いらいひんめい,いらいすうりょう,おくりさきめい? BALLANTINE'S,30,FUJITA  } 出庫依頼3

さﾞいこﾞそくれんらく  BALLANTINE'S     21     } 在庫不足連絡2

こんてなﾊﾞんごｳ, はんにゅうねんげﾂ, にちﾞ? 2,84/8,30/10:30      ┐
つみにひんめい,つみにすうりょう? BALLANTINE'S,21                  │ 積荷票2
つみにひんめい,つみにすうりょう? DRY SACK,6                       │
つみにひんめい,つみにすうりょう? Z,0                              ┘

しゅっこしﾞ          # 3              FUJITA     ┐
 1                  BALLANTINE'S     9          │ 出庫指示2
 2                  BALLANTINE'S     21         ┘

しゅっこしﾞ          # 2              HAMADA
 1                  DRY SACK         24         こんてな--- 1 ---はんしゅつ  ┐
                                                                            │ 出庫指示3
 2                  DRY SACK         6          こんてな--- 2 ---はんしゅつ  ┘
```

積荷票と出庫依頼がシステムへの入力である．出庫指示と在庫不足連絡がシステムからの出力である．

図-1 在庫管理システムとオペレータとの対話例

は省略した).

(1) 積荷処理のスレッド

(1.1) 積荷票の終りまで読み込む.

(1.2) 積荷品名と積荷数量を積荷リストに登録する.

(1.3) 1つのコンテナ内の積荷総計を数える.

(1.4) 積荷票の各積荷に対してその在庫総数を書き替え，かつ在庫不足品処理を行う.

(2) 出庫依頼処理のスレッド

(2.1) 出庫依頼品に対して在庫が十分あれば出庫処理を行う.

(2.2) 出庫依頼品に対して在庫が十分なければ不足総数を書き替える．さらに，在庫不足連絡をする.

(3) 在庫総数書替えと在庫不足品処理のスレッド

(3.1) 積荷品名が内蔵品リストになければ，在庫総数0の内蔵品として内蔵品リストに登録する.

(3.2) 在庫総数に積荷数量をたし込む.

(3.3) 不足総数を書き替える.

(3.4) 積荷品の入荷を待っていた在庫不足品があり，かつそれが出庫可能である間は出庫処理を繰り返す.

(4) 出庫処理のスレッド

在庫不足品の出庫が完了しない間は

(4.1) 出庫数を計算する.

(4.2) 積荷数量および不足数より出庫数を引き去る.

(4.3) 積荷品が入っているコンテナが含む総品数(コンテナ内総計)から出庫数を引き去る.

(4.4) 在庫総数より出庫数を引き去る.

(4.5) 出庫指示書を出力する.

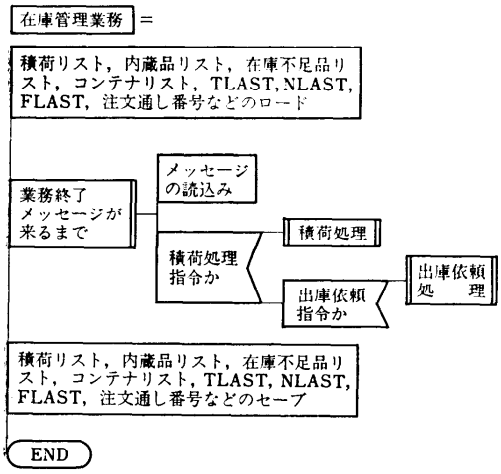
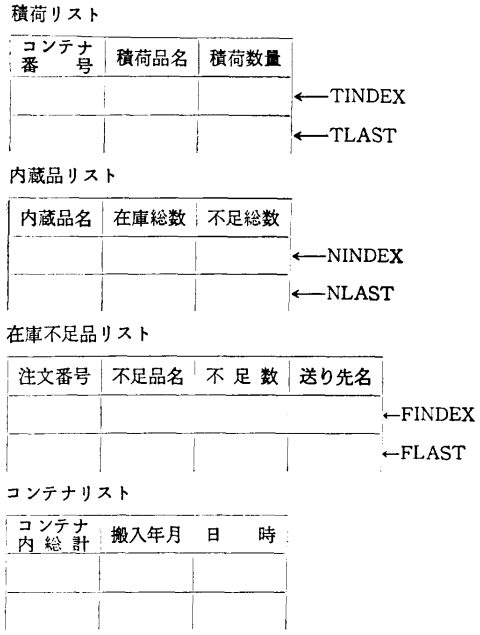
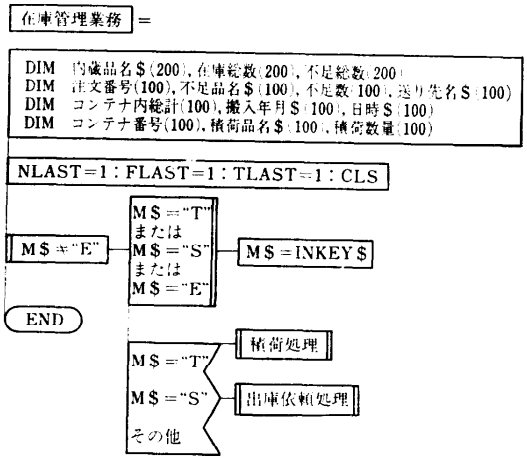


図-2 在庫管理システムのトップレベルの構造



LAST の付く変数はリストにおける最初の空き番地を指すポインタである．INDEX の付く変数はリストを探索する際に用いるポインタである．問題を理解するフェーズでは，これらのポインタまで考える必要はない.

図-3 在庫管理システムにおけるリスト類の構造



諸リストなどのロード，セーブの処理は，説明を簡単にするために，省略した.

Q は WHILE Q と同じ意味である.

図-4 在庫管理業務の PAD

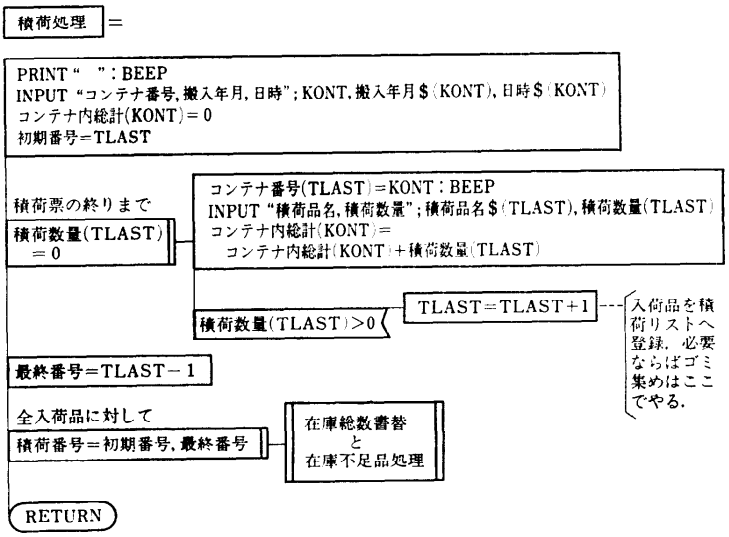


図-5 積荷処理の PAD

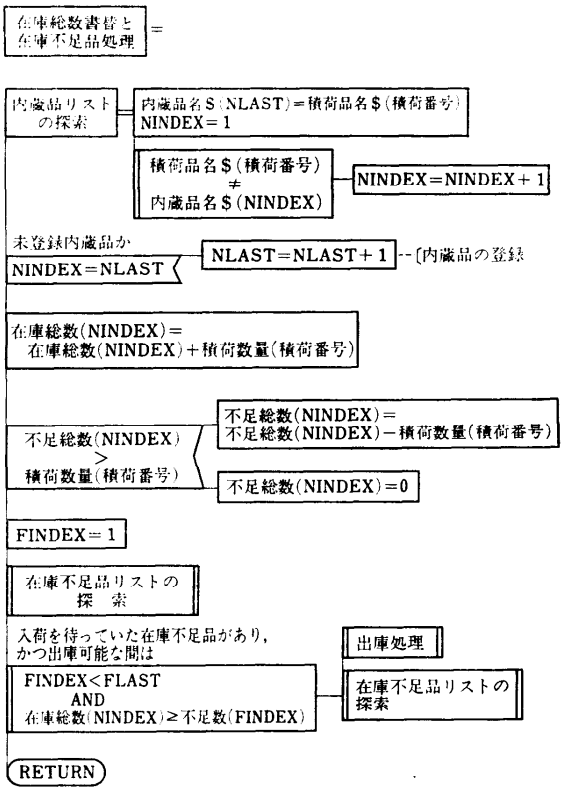
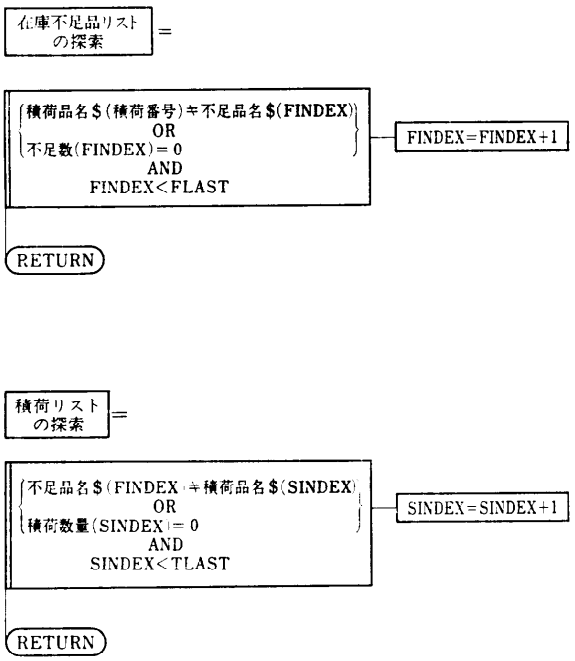


図-6 在庫総数書替と在庫不足品処理の PAD



SINLEX は，出庫処理サブルーチンが積荷リストを探索する際に用いるポインタ．

図-7 リスト探索用サブルーチンの PAD

(4.6) コンテナ内総計が0ならば，コンテナ搬出マークを出力する．

これからいよいよ各種のスレッドを実際のプログラム言語を用いて作成し，それらを統合するフェーズに入る．以下ではプログラム言語としては，BASIC を用いることにする．更に，作成するプログラムについて，次に述べるような条件を付ける．

(i) 積荷票を入力する前には文字“T”を入力する．そして積荷票の終りは数量0の内蔵品（名前は何でもよい）により示すこととする（図-1の積荷票1参照）．

(ii) 出庫依頼を入力する前には文字“S”を入力する．

(iii) 文字“E”により業務終了を指示する．

(iv) プログラムの作成を簡単にするためにコンテナ番号は，1から100までの整数とする（原問題通りコンテナ番号を5桁の数とするためにはコンテナリストに“コンテナ番号”の欄を追加すればよい）．

(v) リスト（表）の各欄を別々の配列とする．

(vi) リストの探索に際しては順次探索(sequential search) を行う．

(vii) 積荷があったときに，不足品の出庫をFIFO (first-in-first-out) で行う．

(viii) 在庫不足連絡に記入する“数量”（不足数量）は，出庫依頼品を出庫するために入荷が必要な最小数とする（図-1の在庫不足連絡1参照）．

(ix) 積荷リストと在庫不足品リストは満杯になったら，各々積荷数量と不足数が0の行を取り除き，リストの詰替えを行う（ゴミ集め）．

(x) 一度も積荷のなかった品目に対する出庫依頼があった場合には，メッセージ“非取扱い商品”を出力する．

統合の過程は紙面の都合でお見せできない．しかし結果として得られたPADを**図-4～図-9**に示した．問題があまり難しくないので，これらのPADを読者が読まれることを期待する．ただし初めは，PAD中のCLS（ディスプレイ画面の消去），PRINT“ ”（改行），COLOR（メッセージの着色），BEEP（オペレータの注意を喚起するための音の発信）などは無視して，PADを読んでいただきたい．

これらのPADをBASICに翻訳するためには文献7)等で述べられている翻訳規則に従えばよい．図-4と図-5のPADに対応するプログラムを**図-10**に示した（システムプログラム全体では約100行になる）．ここでは漢字で書かれた変数名等はローマ字に直してある．検証のフェーズは省略した（紙面の都合でプログラム作成過程の説明が不十分になってしまったことをお詫びしたい．PAD/PAMの詳細については文献3),7)などを参照願いたい）．

5. おわりに

PAD/PAMの概要およびPAD/PAMによるプログラム作成例を示した．PAMの特徴は分割と統合のフェーズを明確に分けたところにある．

5.1 問題の分割法とプログラム言語

筆者は**4**章で述べた在庫管理プログラムを初め，Concurrent Prolog (CP)[15]風な擬似言語で記述した[5]．その際の考え方は，積荷リスト等，各種リストの要素をオブジェクトと考える，いわゆるオブジェクト指向的な考え方であった[18]．コンテナ，積荷票，出庫依頼，内蔵品等の物理的なものがオブジェクトに対応し，問題の分割が極めて自然に行えたという印象を持った．ちなみに，オブジェクト指向的な考え方ではオ

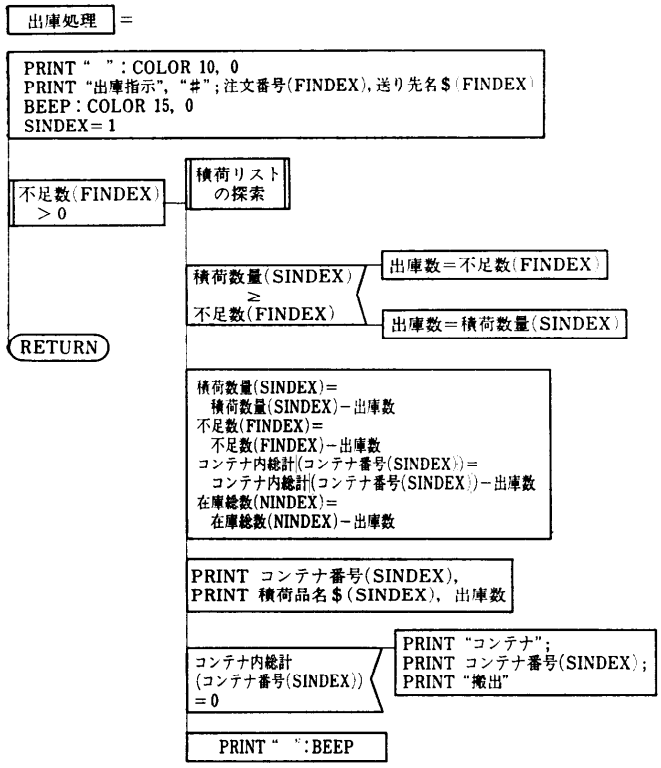


図-8 出庫処理の PAD

ブジェクト＝プログラムという関係が成り立つ．したがって，問題に現れる適当なオブジェクトを認識することが問題の分割に相当する．

しかし，手軽に使える CP の処理系が筆者の手許にないという理由で今回は CP を使わなかった（そのかわり，ICOT の大木優氏等がこのシステムを CP を用いて作成し，デバッグを済ませておられる[10]）．筆者は自分の書斎にあるパソコン（MB 16000）の BASIC でプログラムを作成した（この程度の大きさのプログラムは，使用言語にこだわらず，一番手軽に使える言語処理系で作成するのが能率的であると考えている）．ただし，BASIC ではオブジェクトをそのままプログラムとして書けないので，同種のオブジェクトを集めて図-3 の 4 つのリストにせざるをえなかった．感じとしては，図-3 の各リストの行がオブジェクトに対応し，各欄がオブジェクトの内部状態に対応する．

図-3 を使うような問題の理解の仕方をすると，BASIC, PL/I, COBOL, FORTRAN, PASCAL, LISP, PROLOG など，在来の言語ででもプログラムが書けるようになる．逆にいうと，上記の言語でプログラムを作成する際には，CP 等の並列型オブジェクト指向言語を用いるよりも，もう一段余計に問題を抽象化する必要があるように感じた（例えば，共通のオブジェクトをまとめて表にするとか，抽象データ型を作るとかする必要がある）．

プログラムを設計する際には，問題を分割するところが一番難しい．例えば，図-2 と図-3 およびスレッドの集合（1）-（4）を与えれば多くのプログラマはほとんど苦労せずにプログラムを作成可能である（このことは筆者等が PAD/PAM の教育コースで，別の問題に対してではあるが，実験済みである）．その意味では，問題の分割が一番容易な考え方（パラダイム．もちろん問題によって異なる）を直接的にサポートする言語を利用できれば一番都合が良い．

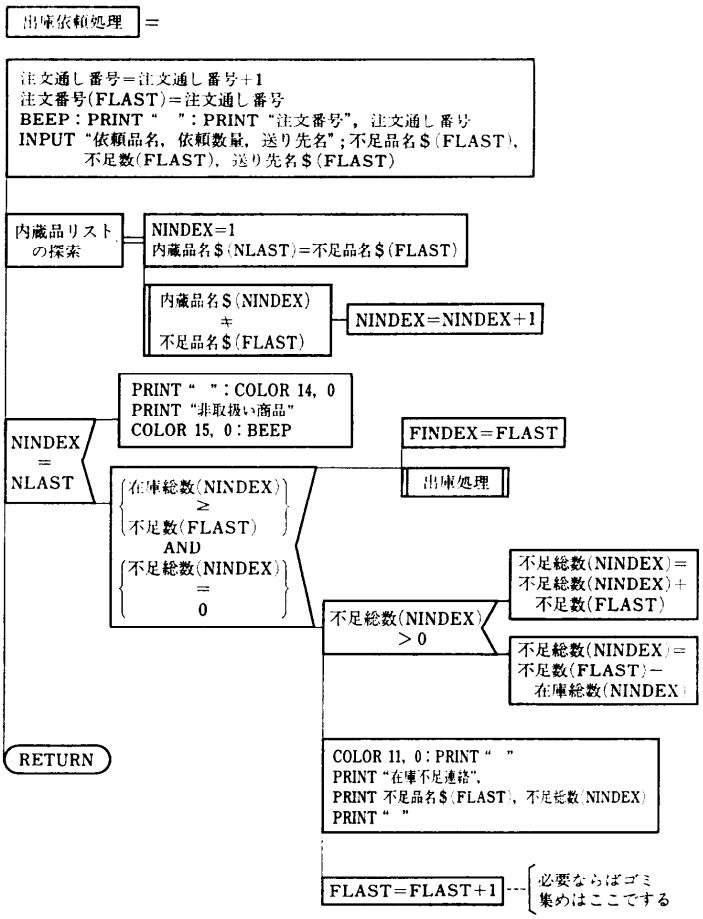
5.2 機能の統合

分割に比較して統合は容易である．これは，図形としての PAD の見やすさ，統合のしやすさに負うところが大きいと思われる．

謝辞 本論文における PAD/PAM の概要のまとめ方について貴重なご意見を下さった北大工学部宮本衛市教授に厚くお礼申し上げます．また本論文の草稿をていねいに査読し，有益なコメントを下さった本学会の査読者に深謝します．最後に，多くのプログラマ，研究者などが興味を持って取り組める本共通問題の出題者，および共通問題の解法に関する本特集号の企画者に対して敬意を表します．

参 考 文 献

1) Dahl, O. -J., Dijkstra, E. W. and Hoare, C. A. R.: Structured Programming, Academic Press, New York (1972).
2) デカルト：知能指導の規則，世界の大思想 21，河出書房新社，東京 (1974).
3) 二村：問題分析法 PAM によるプログラムの設計と審査，情報処理学会論文誌，Vol. 23, No. 4, pp. 451-458 (1982).
4) 二村：構造化プログラム図式，コンピュータソ



出庫依頼品は，処理の都合上，在庫不足品リストに登録する点に注意されたい．

図-9 出庫依頼処理の PAD

フトウェア（日本ソフトウェア科学会誌），Vol. 1, No. 1, pp. 64-77 (1984).

5) 二村，藤田：プログラム設計法 PAD/PAM とその効果，「プログラム設計技法の実用化と発展」シンポジウム，情報処理学会 (1984).

6) Futamura, Y. and Kawai, K.: Problem Analysis Diagram (PAD), JARECT, Vol. 12, オーム社 (June 1984).

7) 二村：プログラム技法—PAD による構造化プログラミング—，オーム社 (1984).

8) Goldstein, H. H. and von Neumann, J.: Planning and Coding Problems for an Electronic Computing Instrument, Part II, in von Neumann Collected Works, Vol. V, pp. 80-151, McMillan, New York (1947).

9) Jackson, M. A.: Principles of Program Design, Academic Press, 1975. 邦訳（鳥居宏次），「構造的プログラム設計の原理」，日本コンピュータ協会 (1980).

10) 大木他：Concurrent Prolog によるオンライン在庫管理システムの記述，プログラミングシンポジウム，1985年1月，発表予定.

11) Page-Jones, M.: The Practical Guide to Structured Systems Design, Yourdon Press, New York (1980).

12) Peters, L.: Software Design: Method and Techniques, Yourdon Press, New York (1981).

13) ポリア（柿内訳）：いかにして問題をとくか，丸善，東京 (1954).

14) 紫合：ソフトウェア設計法について，コンピュー

```
10 REM ZAIKOKANRI
20 DIM NAIZOUHINMEI$(200),ZAIKOSOUSUU(200),FUSOKUSUURYOU(200
)
30 DIM CHUUMONBANGOU(100),FUSOKUHINMEI$(100),FUSOKUSUU(100),
OKURISAKIMEI$(100)
40 DIM KONTENANAISOUKEI(100),HANNYUUNENGETU$(100),NICHIJI$(1
00)
50 DIM KONTENABANGOU(100),TUMINIHINMEI$(100),TUMINISUURYOU(1
00)
60 NLAST=1:FLAST=1:TLAST=1:CLS
70 WHILE NOT M$="E"
80 REM
90 M$=INKEY$
100 IF NOT (M$="T" OR M$="S" OR M$="E") THEN 80
110 IF M$="T" THEN GOSUB 150:GOTO 130
120 IF M$="S" THEN GOSUB 790
130 WEND
140 END
150 REM TUMINISHORI
160 PRINT " "
170 BEEP
180 INPUT "こんてなばんごう, はんにゅうねんげつ, にもつ";KONT,HANNYUUNENGETU
$(KONT),NICHIJI$(KONT)
190 KONTENANAISOUKEI(KONT)=0
200 SHOKIBANGOU=TLAST
210 REM
220 KONTENABANGOU(TLAST)=KONT
230 BEEP
240 INPUT "つみにひんめい,つみにすうりょう";TUMINIHINMEI$(TLAST),TUMINISUUR
YOU(TLAST)
250 KONTENANAISOUKEI(KONT)=KONTENANAISOUKEI(KONT)+TUMINISUUR
YOU(TLAST)
260 IF TUMINISUURYOU(TLAST)>0 THEN TLAST=TLAST+1:GOTO 210
270 SAISHUUBANGOU=TLAST-1
280 FOR TUMINIBANGOU=SHOKIBANGOU TO SAISHUUBANGOU
290 GOSUB 320
300 NEXT TUMINIBANGOU
310 RETURN
```

図-10 在庫管理システムプログラムの一部
(システム全体では約100行)

タソフトウェア (日本ソフトウェア科学会誌), Vol. 1, No. 2, pp. 55-68 (1984).

15) 竹内：論理型並列プログラミング言語—Concurrent Prolog, コンピュータソフトウェア (日本ソフトウェア科学会誌), Vol. 1, No. 2, pp. 25-37 (1984).

16) ワーニエ (鈴木訳)：ワ・・ニエプログラミング法則集, 日本能率協会 (1975).

17) 山崎：設計方法解説のための共通例題, プログラム設計技法の実用化と発展シンポジウム, 情報処理学会 (1984).

18) 米澤：オブジェクト指向型プログラミングについて, コンピュータソフトウェア (日本ソフトウェア科学会誌), Vol. 1, No. 1, pp. 29-41 (1984).

(昭和59年9月25日受付)