情報処理教育に関する一考察
—初級プログラミング教育の内容と方法について—

海老沢　信　一

A Study of Information Management Education
— Education Content and Method of Courses for Teaching Programming to Beginners —

Shin'ichi Ebisawa

Japan and other industrialized countries are now in the process of becoming highly advanced information societies. Many guidelines and proposals on how to teach information processing have been made by various organizations and committees. There are also a number of books, published in recent years, on information processing. These guidelines and books, however, are offered for various purposes and do not necessarily meet each school's educational policies or curricula.

At present, syllabus and teaching methods are in the hands of individual teachers and lecturers. Each of these people develop their own course plans.

In this situation, the author has himself developed curricular and methods while teaching elementary programming to beginners.

This report, on elementary programming education to construct information processing systems (business systems in particular), is based on my experience teaching information processing and programming both to technical college students and college students majoring in humanities and social sciences.

1．はじめに

日本を含む先進諸国は，高度情報化社会への途上にあると言われている。そのため情報処理教育をどのように行うべきかについては，いくつかの指針や提言が各団体や委員会から出されている。また，情報処理に関する書籍も最近は数多く市販されている。しかし，これらの指針や書籍もいろいろな目的で記述されているので，必ずしもその学校の教育方針や授業に適したものが存在するわけではない。

具体的な教育内容や教育方法については，担当する教員や講師個人々々の工夫に任されているのが現状であり，また教員や講師は各人が言わば手探りの状況で教育を行っているのが実情ではないだろうか。

筆者もそのような状況の中で，情報処理教育を中心としたカリキュラム編成と科目内容

を考え，同時に初心者の情報処理教育やプログラミング教育を行ってきた。

そこで今までの文科系大学でのプログラミング教育と専門学校等における情報処理教育の経験から，筆者は情報処理システム（特にビジネスシステム）を構築するための初級プログラミング教育の内容と方法をまとめたので報告する。

（本稿では，プログラミングとはフローチャート作成とプログラム作成の両方の作業を含むこととする。）

2．情報処理教育

2-1　情報処理教育の背景

教育界では時代の進展に対応できる能力を養うことを一つの柱として教育課程の改訂を続けており，情報化への対応が検討されている。即ち昭和62年の臨時教育審議会の答申は，次のことを掲げている。

・情報モラルの確立
・情報化社会システムの構築
・情報手段と活用
・情報環境の整備

そして教育自体を情報化社会に対応できるように改革する必要があると述べている。

平成4年度から，小，中，高校でコンピュータを使った授業がスタートする。算数，数学，理科などの授業で積極的にパソコンを活用するほか，中学校の技術・家庭科では「情報基礎」が，選択コースのひとつとして用意される。

一方，このような教育界の動きに対して，通産省は早くから情報処理技術者教育に力を入れているのは広く知られている。昭和46年から発足した「情報処理技術者試験」などにその努力がみられる。

また，「日本情報処理開発協会」は「初級情報処理技術者育成指針」（昭和61年改訂）や「高度情報処理技術者育成に関する調査報告書」（昭和62年）などを発表して，情報処理技術者育成のための指針を示している。

小中学校を含めた日本における本格的な情報処理教育はまだその端緒についたばかりであるが，情報処理教育の内容は常に時代の進展と共に改訂されなければならない運命を背負っている。

2-2　情報処理教育の内容

今日必要とされる情報処理教育の内容は何か，またどのように教育すべきかにはいろいろな意見が存在する。

筆者は大きくとらえて，情報処理教育には次の事柄が必要と考えている。

(1)コンピュータリテラシー教育

コンピュータリテラシー教育とは，コンピュータを道具として使いこなし，日常生活や仕事に役立たせる能力であり，次の知識が必要である。

・キーボードやコンピュータの扱い方に関する知識
・各種アプリケーション・ソフトウェアの利用に関する知識

(2)プログラミング教育

プログラミング教育とは，情報処理システムを設計，開発，運用することに関する種々の能力を養うことであり，次の知識が必要である。

・情報処理システムの設計，開発，運用に関する知識
・プログラム言語，文法，作成に関する知識
・ファイルとその扱い方に関する知識

(3)情報処理関連知識教育

情報処理に関連するいろいろな分野の一般知識であり，次の知識が必要である。

・ハードウェアやソフトウェアに関する知識（オペレーティングシステムを含む）
・データ通信やデータベースに関する知識
・情報化社会，組織，プライバシーなど社会的な側面に関する知識
・数学，簿記などコンピュータに関連する

基礎知識

2-3　プログラミング教育の必要性

既に述べたようにコンピュータリテラシー教育は情報処理教育の上で必要なものの１つである。例えば，市販の「LOTUS123」とか「Multiplan」などのソフトウェアを購入し，問題解決に応用しようとする方法即ちアプリケーション・ソフトウェア（ビジネスソフトとも呼ばれる）の活用は，社会生活や職場の仕事で今日不可欠な情報処理手段になっている。

各種アプリケーション・ソフトウェアを利用しつつ，コンピュータリテラシー教育を行う方法は，中学高校での情報処理教育で行われている。また，大学，短大，専門学校等の高等教育機関でも盛んに行われている。

しかし西暦2000年には約34兆円とも試算されるソフトウェア需要（産業構造審議会情報化人材対策小委員会まとめ）や同じ年に必要とされる230～300万人にものぼる情報技術者需要（文部省教育改革実施本部）など社会的な要請を考えると，これらのアプリケーション・ソフトウェアの教育だけでは不足である。そのためシステム設計を含めたプログラミング教育は依然大切なものであると筆者は確信する。

3．初級プログラミング教育の方法

初心者にビジネスシステム構築のプログラミングを教育するには，アルゴリズムの組立て即ちフローチャートの作成方法を教育しながら，同時にプログラムを作成してすぐにそのアルゴリズムを演習する方法が有効である。

言い換えれば，アルゴリズムの組立てとその実践（プログラムの実行）をいかに有機的に結合し，わかりやすく教育していくかを考え，この観点からカリキュラム編成と科目内容が検討されるべきであると言っても過言ではない。

3-1　フローチャート作成について

企業やソフトウェアメーカ等が初級プログラマに期待する知識と，初心者の理解度や実情の両者を考慮して，筆者はプログラミング教育のために，次のような指導項目とその順序を考えた。

(1)フローチャート基本記号の説明
(2)フローチャート基本三構造の説明
(3)直線型フローチャートとその演習
　①逐次処理の概念把握
　②領域の概念把握
　③内部・外部の概念把握
(4)分岐型フローチャートとその演習
　①分岐の概念把握
(5)繰返し型フローチャートとその演習
　①繰返しの概念把握
　②カウンタの概念把握
　③合計と平均の概念把握
(6)基本プログラミング技術とその演習
　①ファイルを読む演習
　②１つのファイルを取扱う演習
　③２つのファイルを取扱う演習
　④配列を取扱う演習

次にフローチャートの演習問題の作成と提示方法について考察してみる。

フローチャートの演習問題は，最初に複雑な問題を示しても初心者は理解できないことは当然である。問題を作成する場合，１つの問題には多くの機能を含めず，まず簡単な機能を含んだ問題から提示すべきである。その後徐々にいくつかの機能を結びつけ，複雑なフローチャートに自然に導くような演習問題の作成と提示方法を研究する必要がある。

ところが，往々にして市販のテキストなどは，この辺の事情を余り考慮しないテキストが見受けられる。最初から多くの機能を含んでいたり，途中から突然難しくなるような展開の仕方などである。

更に，初心者には認知心理学的な手法は有効な手段であろう。説明する際には，日常生

活に適した図や絵を示し，対象物を具体的なイメージと結びつけて理解しやすいようにすることも大切である。

3-2　プログラム言語の選択について

プログラム言語には多くの種類があるので，どの言語を採用するかについては目的，環境，費用などいろいろな要素を考慮しなければならない。また，プログラミング教育でどの言語を扱うかについても，それぞれの言語に一長一短があり，適切な言語を選択するのはなかなか難しい。

しかし，「初心者にフローチャートを教育する補助手段」という観点から言語を選ぶならば，筆者はBASIC言語が適切であると考える。

プログラミング教育にBASICを採用することは，特別に目新しい試みではない。しかし筆者の知る限り，フローチャートの作成を理解させるための道具としてBASICを有効に活用している教材は少ない。筆者はBASICを言語として教育するよりも，フローチャートを理解するための1つの道具として位置づけ，新しい切り口でこの言語を活用した。

BASICを使用する利点は，次のような点である。

(1)フローチャートの1つ1つの処理とBASICの各命令を，ほぼ1対1で対応することができ，初心者でもフローチャートとプログラムの結びつきが理解しやすい。

(2)BASICはインタープリタ型なので，修正や実行が簡単であり，環境整備に手間取らない。

(3)BASICはどのようなパソコンでも個人で簡単に練習することが可能である。

ところで，BASICは普及している割りには，実際のビジネス用言語として採用されていることは少ない。むしろCOBOLの採用の方が多いと思われる。

しかし，フローチャートが完成したらプログラミング作業の半分以上は完了したと言われるように，フローチャートを理解することは重要であり，そのための道具としてBASICを活用する方法は大変有効である。実際にフローチャートを充分理解させてからCOBOL言語を教育してみると，その理解がはやいことを確認できる。

なお，本稿はBASIC言語として，現在比較的容易に入手できるN_{88} DISK BASICを想定している。

4．フローチャート作成方法の教育

4-1　フローチャート基本記号の説明

フローチャートで使用する記号については，まず最初の段階では基本的かつ最小限の記号を提示し説明するだけに止めるべきである。

この段階で沢山の記号を教えると，記号の意味を混同したり，フローチャートの作成を面倒なものと考え，結果的にプログラミングが嫌いになってしまう恐れがある。筆者が考えたこの段階で教えるべき基本記号とは次の記号に限定した。(図1)

初心者にはフローチャートの導入として日常生活を題材としたフローチャートを作成させてみると，これらの基本記号の意味を良く理解する。フローチャートという抽象的な概念を日常行動という具体的な概念と結びつけられることが，導入としては適切なのであろう。

4-2　フローチャート基本三構造の説明

今日，構造化プログラミングは広く認められているプログラミングの基礎技術である。またプログラムのすべてのアルゴリズムは基本的な3つの構造で書くことができることは広く認められている。フローチャートの作成には，まず最初にこの基本三構造を教育することが適当である。

フローチャートを分析的にとらえた基本三構造は，教育しやすいだけではなく，現在多くの企業やソフトウェアメーカが何らかの形

(1) 端子記号 端子記号はフローチャートの始めと終りを表す記号である。	
(2) 処理記号 処理記号は処理（代入、演算など）を表す記号である。	
(3) 判断記号 判断記号は条件を判断し出口を選ぶ記号である。	
(4) ループ端記号 ループ端記号にはループ始端とループ終端の2つの形があり、それぞれループ（繰返し）の開始と終了を表す記号である。	
(5) データ記号 入力と出力を表す記号である。	
(6) 線 データや制御の流れを表す記号である。	

図1　フローチャート基本記号

で構造化プログラミング技術をとりいれていることを考えれば教育には不可欠な知識である。(フローチャートに代わって，NSチャートやSPDなど構造化を考慮したチャートが各メーカで工夫され使用されている。しかしどれか1つを教育すれば，他の手法は理解できる。)

フローチャート作成にはこれら3つの型式に関する問題を提示し，初心者に充分練習させることがプログラムの上達には早道である。基本三構造とは，次のようなものである。

(1)直線型のフローチャート（図2(a)）

直線型は処理が上から下へ流れる形をしているもので，最も基本的な形である。

(2)分岐型のフローチャート（図2(b)）

分岐型にはIF THEN ELSE型とIF THEN型の2つの形がある。

IF THEN ELSE型は条件を判断し，条件が成り立てばある処理を行い，条件が成り立たなければ別の処理を行う形をしている。

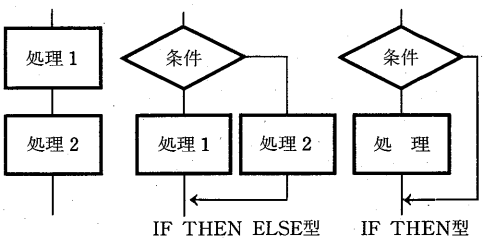
IN THEN型は，条件が成り立てばある処理を行い，条件が成り立たなければ何もしない形をしている。IN THEN型はIF THEN ELSE型の特別の場合といえよう。

(3)繰返し型のフローチャート（図2(c)）

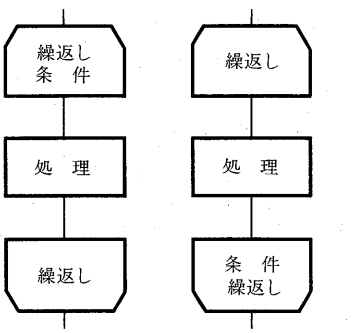
繰返し型にはDO WHILE型とREPEAT UNTIL型の2つの形がある。

DO WHILE型は，条件が成立するまである処理を繰り返す（ループ）形をしている。

REPEAT UNTIL型は条件がループの最後にある形をしている。



(a) 直線型　　(b) 分岐型



(c) 繰返し型

図2　フローチャートの基本三構造

4-3　直線型フローチャートとその演習

初心者にとってフローチャートとプログラムの関係を理解することはなかなか容易なことではない。そこで筆者は両者の関係を分析して，次の概念に着目した。

・逐次処理の概念
・領域の概念
・内部と外部の概念

これらの概念を説明しながら問題を演習させ，フローチャートとプログラムとの関連を教育する。更に必要な場合は簡単なコンピュータの構造を提示し理解させる。これらの概念を繰り返し教えると，その後のプログラミングの上達は早く，またフローチャートやプログラムに関する無用な勘違いが少ないようである。

(1)逐次処理の概念把握

現代のコンピュータは，プログラムを1命令ずつ CPU（Central Processing Unit）で解析し，実行する構造になっている。

そのためフローチャートの流れもこれに沿った形で，処理は上から下へ流れる逐次処理の構造になっている。フローチャートの説明で，1つずつ（逐次に）処理が行われるという概念をはっきり示すことで，初心者の中に「手続き型プログラムの概念」を形成させることが大切である。（図3）

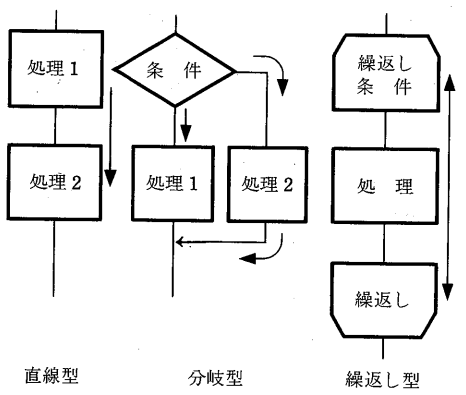


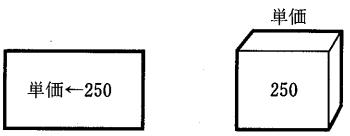
図3　逐次処理の概念

この考え方は上級者にとっては当り前の概念であるが，初心者には明確に示す必要がある。逐次処理の概念の説明には，たとえば「双六」などをイメージさせることは良いことであろう。双六は1つずつマスを進むので，マスを処理に例えれば逐次処理の概念を想定しやすい。

(2)領域の概念把握

現代のコンピュータは，メモリ内に確保された領域（エリア：Area）にプログラムの命令やデータを格納し，実行する構造になっている。そのためフローチャートの中でもこの領域を定義する必要がある。フローチャートの領域がプログラムの変数と対応することを，ここで教えるべきである。

フローチャートで領域を定義するには，領域に名前をつけて定義する。領域の名前は英数字でも日本語の単語でも良い。次のように日本語を領域名にしても，領域が定義され，領域の中にデータが保存されるということを理解させる必要がある。



領域の概念の説明には，日常生活での「箱」とか「カセットテープ」をイメージさせる手法が良い。カセットテープにデータ（音楽などを想定させる）を記録するというイメージであり，新しいデータを記録すると，

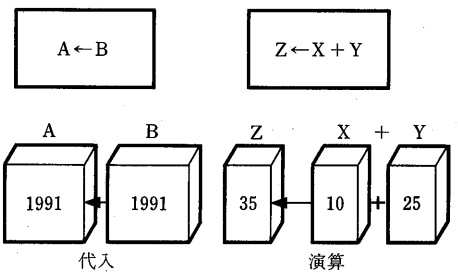


図4　領域の概念

前のデータは消滅してしまうことを理解させなければならない。(図4)。

ところで領域をいくつか並べ同一の名前を付けたものが配列（テーブル：Table）である。初心者にとって領域に関する何の知識もなく，配列の概念を理解することは難しい。ここで領域の概念をしっかり把握させることは，プログラムにおける配列の概念を理解させる上で大変役立つ。配列は領域を複合化したものと考えることができる。しかし，この段階で配列を説明する必要はない。混乱を増すだけで何の効果もないと思われる。

(3)内部と外部の概念把握

現代のコンピュータは，ほとんどの機械が通常ノイマン型コンピュータである。そこでプログラムはメモリーに搭載され，処理されるべきデータはコンピュータの外部から内部（メモリ）に入力されたり，コンピュータの内部から外部に出力されたりする構造になっている。(図5)

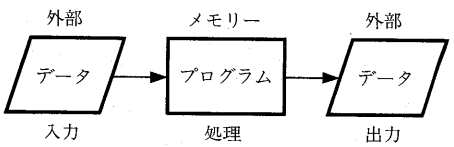


図5　内部と外部の概念

内部と外部の概念の説明には，日常生活での「電卓」をイメージさせる。キーボードからデータを入力し，計算して，結果を出力（表示）するイメージは，内部と外部の概念把握に役立とう。

データとプログラムとは別であり，データを集めたものがファイルである。これを外部から入力し，処理した結果を再びデータとして外部のファイルに集めることを説明しておくと，「ファイルの概念」が初心者の中に形成される。

4-4　分岐型フローチャートとその演習

分岐型フローチャートのポイントは，判断記号を理解するかどうかにかかっている。

(1)分岐の概念把握

分岐型には基本三構造のところで示したように，IF THEN ELSE 型と IF THEN 型の2つの形がある。どちらにしろ判断記号の意味を充分理解させることが必要である。判断記号自体は領域（エリア）同士を比較する関係式であり，さして難しくはない。分岐型フローチャート作成の指導上の難しさは，「条件を何にし，条件をどのように設定するか」を考えさせることにある。

4-5　繰返し型フローチャートとその演習

(1)繰返しの概念把握

繰返し型には，DO WHILE 型と REPEAT UNTIL 型の2つの形がある。

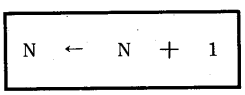
繰返し（ループ）については，繰返し型のフローチャートを示し，その動き方を説明すれば比較的容易に理解しよう。ループ端記号は比較的新しい記号なので，誤解のないように説明すべきである。

しかし，ここでも難しいのは「ループの終了条件を何にし，終了条件をどのように設定するか」を考えさせることである。

フローチャートの作成は数多くの問題を演習させ，身を持って理解させる事が大切であり，講義だけでは不足であろう。

(2)カウンタの概念把握

領域Nを用意し，Nをある一定数ずつ増加させるときNをカウンタ（Counter）と呼ぶ。カウンタの概念はプログラムを作成する場合，必要不可欠な概念である。例えば，Nを1つずつ増やすフローチャートの処理は次のように表される。



そこで，初期値としてNに0を代入しておき，ループの中にカウント処理を含めれば，Nは1つずつ増加する要領である。(図6)

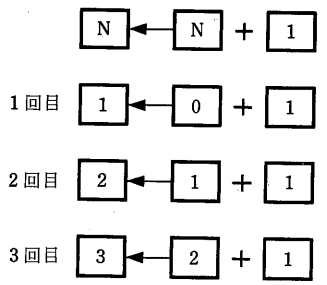


図6　カウンタの概念

(3)合計と平均の概念把握

カウンタの概念を更に進めて，合計と平均の概念を理解させる。

この段階までくると，筆者の経験では直線型フローチャートの箇所で示した「逐次処理の概念」はもとより「領域の概念」，「内部と外部の概念」をかなり理解するようになっている。

合計処理を行うには例えば合計という名前の領域を用意し，得点をその合計に加算する処理をループの中に含めることで行える。

ただし，合計を初期値として0にしておかなければならないことを注意する。

```
合計 ← 合計 + 得点
```

個人差にもよるが，この合計を加算する概念が理解できなかったり，誤解するような初心者がいる。主な原因は合計と得点を加算した値を，再び合計に保存することが理解できないためである。すなわち「合計という領域」にとっては，自分自身を読んだ後，自分自身を壊しながら加算した値を再び保存するという概念が理解できないことに起因することが多い。

さて，合計とカウンタを組み合わせれば，平均が計算できる。即ちループの中に，合計処理と件数のカウンタを含めておき，ループが終了した時点で合計値を件数で割り算すれば，平均が求められる。

4-6　基本プログラミング技術とその演習

以上で基本三構造を提示しながら，フローチャートの作成とプログラムの結びつきを教えたら，次に初心者にとって理解しにくいもう一つのポイント，即ちファイルの取扱いを教えることが大切である。

技術計算やシステムプログラム作成のためのFORTRANやC言語を扱う場合，ファイルの取扱いがCOBOL程には重要視されない傾向がある。しかし，ビジネスシステムの構築にはファイルの概念は不可欠であり，是非教育しておくべきテーマである。

ところでファイルの概念そのものについては「内部と外部の概念」の説明で，ある程度理解していることを前提とするが，基本プログラミング技術を説明する中でより深くファイルの概念を理解させるようにしたい。

(1)ファイルを読む演習

初心者にファイルの概念を教えるためには，「ファイルを読む」ことから演習させた方が良い。即ち，初心者にはファイルは作成させない（書かせない）方針を筆者はとる。それは次の理由による。

- ファイルを作成する作業は，環境整備に手間がかかることが多い。
- データを作成する作業は，時間と労力がかかる。
- 学習者がデータ自体を作成することに興味を示したり，夢中になったりしてデータを作成する作業に埋没することがある。

筆者はファイルを「データを作成する作業」から理解させるよりも，通常では目でみえないブラックボックスとしてのファイルを，「プログラムというツールを使って読むという作業」から理解させたほうが理解がはやいと考える。

そこで教える側がフロッピーディスクなどにあらかじめファイルを支給（コピー）し，

学習者はこのファイルを読むプログラムを作ることから演習に入るのが適当である。

(2)1つのファイルを取扱う演習

基本プログラミング技術の演習としては，まず1つのファイルを取扱う演習を行うのが良い。1つのファイルを取扱う技術として代表的な，次の演習を行う必要がある。

・明細出力処理
・ページコントロール処理
・グループコントロール処理

①明細出力処理

学習者にブラックボックス（データ内容を教えない）としてのファイルを支給する。そしてファイルの内容を何の加工もせず，そのまま明細として表示するようなプログラムを作成させる。このような処理を明細出力処理と呼ぶ。

明細出力の演習はプログラムとデータの扱い方が明確に区別されている現代のコンピュータの特質を理解させるのに有効であり，またファイルの概念を把握させるのに役立とう。

明細出力処理は繰返し型の応用であり，次の要素を含んでいるので，これまでの復習を兼ねて説明すると良い。

・前処理，主処理，後処理というようにフローチャートをカテゴリ分けして考える考え方。
・入力，処理，出力の明確な区別。
・ループ（繰返し）の終了方法。

更に進んで，明細行を編集する演習すなわち編集コマンドの練習をさせてみることも良いであろう。編集とは，例えばPRINT USINGなどを使用して，明細行をわかり易くすることである。

②ページコントロール処理

明細出力処理の演習が終了したら，今度は1ページに一定行数の明細行を出力する制御構造を教育すべきである。この制御をページコントロールと呼ぶ。通常の帳票は，このような制御構造で印刷されて見易い帳票となっている。

ページコントロールの制御にはページカウンタの導入が不可欠である。ページカウンタとは，「出力しているページについて現在何行目を印刷中なのかを記憶して置く1種のカウンタ」である。カウンタの概念については，ここではすでに学習済みなので容易に理解しよう。カウンタの制御構造だけを抜き出して示すことも有効であろう。（図7）

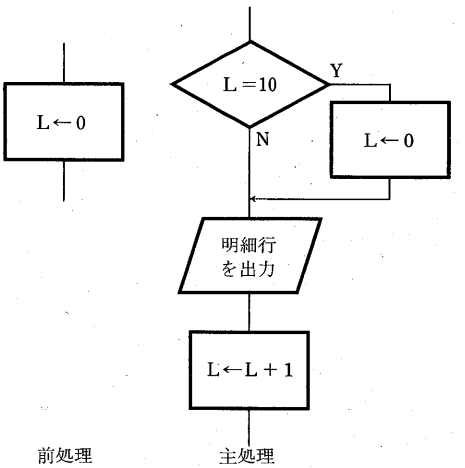
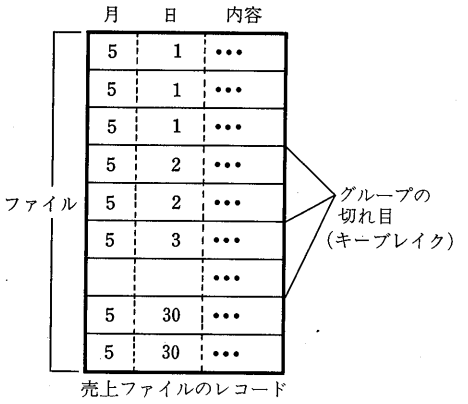


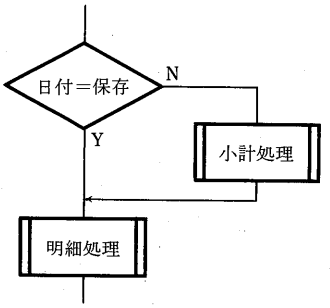
図7　ページコントロール

③グループコントロール処理

次にレコードの中のある項目（コントロールキー）に注目し，この項目が等しい値を持つレコードの集まりを1つのグループとして，集計や印字を行う処理をグループコントロールと呼ぶ。グループコントロールはキーが昇順（あるいは降順）に並んでいることが前提である。この処理も基本的なアルゴリズムなので，充分に理解させるべきである。

この処理はコントロールキーがどこから変わったのか（キーブレイクという）を見つけることがポイントであり，次に示す制御構造を充分理解させるべきである。例えば，日付毎に売上ファイルを集計する場合を考えてみる。（図8）





・保　存…1つ前のグループの日付を保存する領域
・小計処理…日付毎の売上げ集計など
・明細処理…入力したレコードを明細行として出力するなど

図8　グループコントロール

(3)2つのファイルを取扱う演習

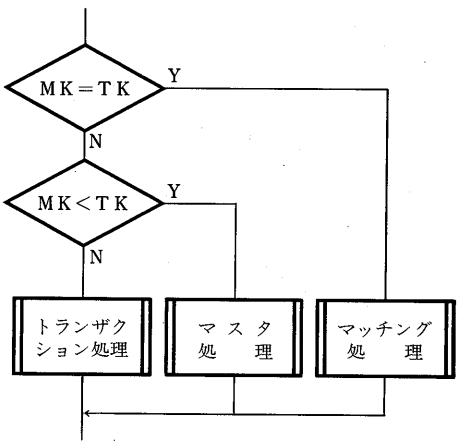
2つのファイルを取扱うアルゴリズムとして次の演習を行う必要がある。

・ファイルの突合せ処理（1：1のマッチング）
・ファイルの突合せ処理（1：Nのマッチング）
・ファイルの併合（マージ）処理

①ファイルの突合せ処理（1：1のマッチング）

手作業では在庫台帳と入出庫伝票を突合せる必要があるように，コンピュータ処理にとっても，ファイルの突合せ処理は不可欠なアルゴリズムである。突合せするためのキーをマッチングキーと呼ぶ。この処理はトランザクションデータでマスタファイルを更新する時や，トランザクションファイルのキーとマスタファイルのキーを1：1で対応（マッチング）させる時に使用する代表的なものである。この処理もキーが昇順（あるいは降順）に並んでいることが前提である。

次に示すように，キーを比較する部分の制御構造だけを抜き出して説明する必要があろう。



・MK　マスタキー
・TK　トランザクションキー
・トランザクション処理
…TKに対応するMKがないことを示し、また次のトランザクションデータを入力する。
・マスタ処理
…次のマスタデータを入力する。
・マッチング処理
…キーが一致した場合の処理を行い、次のマスタデータとトランザクションデータを入力する。

図9　1：1のマッチング

トランザクションファイルに同一キーが複数件存在する場合を，1：Nマッチング処理と呼ぶ。また，2つのファイルを突合わせて，1つファイルにまとめることを併合（マージ）と呼ぶ。

(4)配列を取扱う演習

領域の概念を進めて配列の概念につなげることができる。既に述べたように，領域はメモリの中に確保した「箱」というイメージで考えることができる。この箱をいくつか並べて処理する場合，１つ１つに名前をつけて定義していては非常に不便である。そこで箱全体に１つの名前をつけ，個々の箱を呼び出すには添字をつけて呼び出す方法が考えられた。これは配列（テーブル：Table）と呼ばれている。

配列の説明では，ベクトルやマトリクスが例として使用されることが多い。日常生活を題材とする観点からいえば，筆者は「マンション」の例を挙げて説明することにしている。即ち，マンションは建物全体として何々マンションという名前を持ち，個々の部屋は１階ならば101，102，103，…２階ならば201，202，203，…というようになっている場合が多い。この考え方はそのまま配列の概念に一致するので，初心者はイメージを作り易い。

ただし，ここで是非注意を喚起したいのは，配列はあくまでコンピュータのメモリー上の概念であるが，初心者は往々にしてファイルの概念と混乱することがある。特に学習者の注意を喚起する必要があろう。

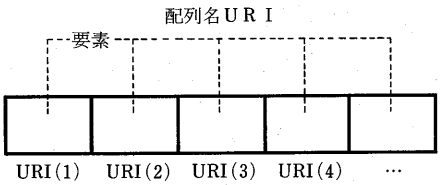


図10　配列の概念

配列には１次元配列や２次元配列などがあるが，１次元配列だけは教育しておきたい。

5．プログラム作成方法の教育

5-1　プログラム言語と文法の説明

(1) BASIC 言語の使い方

筆者はフローチャートの理解を助けるための道具として BASIC 言語を選んだ。そこで，BASIC 言語そのものの教育については，フローチャートを理解するのに必要な範囲の BASIC 文法のみを教えることとし，これ以外の文法は BASIC として典型的な命令でも，敢えて教えないこととした。

言わば，新しい切り口で BASIC という言語をとらえ直すことを試みた。次のような試みである。

① BASIC の「FOR～NEXT 命令」は繰返しを処理する典型的な命令である。プログラムとしては便利な命令であるが，繰返し型のフローチャートと結びつけると１：１で対応せず，かえって理解を難しくすることが考えられる。

そのため最初のうちはこの命令の代りに，敢えて IF 命令と GOTO 命令をフローチャートの流れにあわせて使用するようにした。

②繰返し型のフローチャートを BASIC で演習するには，まず上述のように IF 命令と GOTO 命令を使用して理解させ，その後「WHILE～WEND 命令」を教えるようにした。「WHILE～WEND」は，COBOL 言語の「PERFORM～END-PERFORM」を連想しやすく，構造化プログラミング手法を理解させるための一助となろう。

（繰返しを理解したあとに，FOR～NEXT 命令を教えても何の支障もない。）

③ BASIC の「READ 命令」は，プログラムの内部（DATA 文）で作成したデータをある領域に代入する命令である。この命令はコンピュータにおける内部と外部を説明する場合には，むしろ邪魔になる場合が多いので教えないようにした。

一見難しい命令に見えても，INPUT 命令（キーボードからの入力）や INPUT＃命令（ファイルからの入力）を使用し，説明をした方がアルゴリズムは理解しやすい。

(2) BASIC 言語の命令一覧

筆者がフローチャートの理解のために採用したBASIC言語の命令は次の命令である。

・定数と変数
・LET
・＋－＊／＾
・REM
・PRINT（LPRINT）
・PRINT（LPRINT）～USING
・END
・INPUT
・IF～THEN～ELSE～
・＝＜＞
・GOTO
・OPEN CLOSE ＃
・INPUT ＃
・IF EOF(1) THEN，IF NOT EOF(1) THEN
・WHILE～WEND
・GOSUB～RETURN
・DIM
・WRITE ＃（教員がファイルを作成する時のみ使用）

5－2　教材についての一考察

以上述べたことを基本に，筆者が作成した具体的な教材提出の一例を報告する。

(1)直線型のフローチャート

直線型のフローチャートの練習には，まず「出力」だけを中心とした例題を提示すると良い。

例題1（図11）
Xに50，Yに60を代入しその和を求める。

この例題ではフローチャートの処理は上から下へ流れるという「逐次処理の概念」を示すことが可能である。

次に「領域の概念」もここで説明すると良い。すなわち領域としてXやYが必要で，この領域に数値を代入することや，またZには計算の途中結果を保存することを説明する。

更に「内部と外部の概念」を示す。即ち計算結果（Z）を内部から外部（画面）に出力する考え方である。

この例題を学習する中で，これらの概念を初心者が明確に認識すれば，その後の上達は早い。次に「入力」を中心とした教材を提示すると良い。

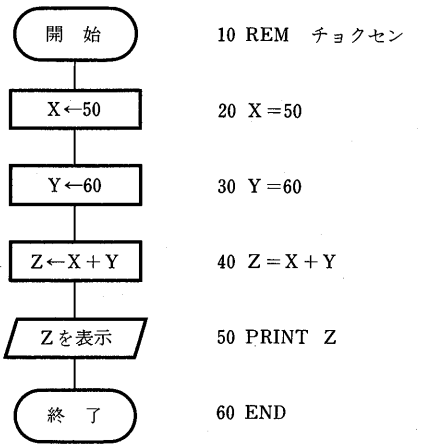


図11　直線型（出力）の例題

例題2（図12）
XとYに数値を入力し，その和を求める。

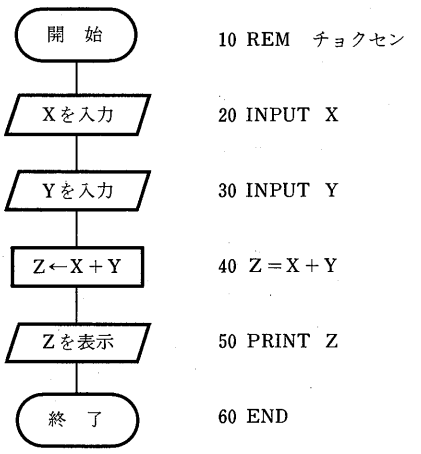


図12　直線型（入力）の例題

ここでも再度「逐次処理の概念」，「領域の概念」を説明し，知識の定着をはかるべきである。「内部と外部の概念」としては，外部からデータを入力する考え方を説明することになる。

ここの説明で入力のためのINPUT命令に，入力を促すためのプロンプト文を最初から示すと，PRINT命令の出力と混同し，混乱することが多いので，むしろプロンプト文をこの段階では説明しないほうが良い。

直線型のフローチャートについては，次に例えば「数値を入力して，面積を計算し表示する」ような入力と出力の両方を含む多くの問題を作成し演習させると良い。

(2)分岐型のフローチャート

問題3（図13）
XとYに数値を入力し，大きいほうの数値を表示する。

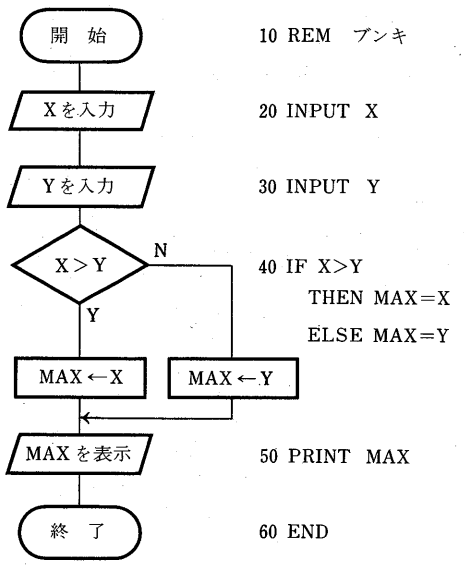


図13　分岐型の例題

この段階の説明は領域MAXの意味を理解させることに重点を置くべきである。また，フローチャートの分岐とプログラムにおけるIF命令との対応を良く説明すべきである。

(3)繰返し型のフローチャート

問題4（図14）
1から10までの整数を表示する。

この段階の説明はループ始端とループ終端

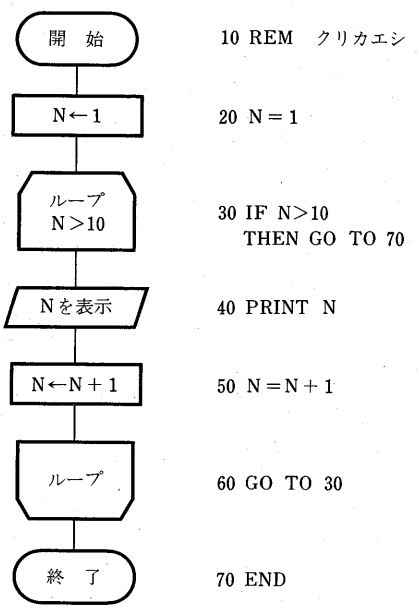


図14　繰返し型の例題

の記号の意味を説明しながら，繰返し（ループ）の動き方を良く理解させることを中心にすべきである。

また，変数Nについて次の項目の相互関係を十分に理解させるべきである。

・変数Nに初期値として何を設定すべきか。
・変数Nの表示とカウントアップの場所をループの中で，どこに置くべきか。
・ループの終了条件は何か。

授業等で繰返し型フローチャートまでの説明になるべく時間を割きながら，多くの問題を演習させると，筆者の経験ではその後のプログラミング技術の習得ははやくかつ確実なものとなる。

6．おわりに

本稿は拙著に掲載した教育内容と方法の展開を論文形式として書き直して報告したものである。本稿の紀要への掲載を快く許可くださった文教大学情報学部紀要委員会に感謝いたします。

[参考文献]

・BASICで学ぶフローチャート技法
啓学出版　海老沢信一・太田信宏著
・初級情報処理技術者育成指針（1986-3）
日本情報処理開発協会情報処理研修センタ
・高度情報処理技術者育成に関するニーズ調査報告書（1987-3）
日本情報処理開発協会情報処理研修センタ
・プログラム流れ図の作成技法
オーム社　江村潤郎・野津　明著
・「情報基礎」の解説（1989-1）
東京書籍　中学校技術・家庭科教授用資料
・中学校技術科におけるコンピュータ教育に関する基礎的研究（1983-3）
東京都立教育研究所科学研究部
・学校におけるパーソナルコンピュータ利用の研究（1988-3）
東京都立教育研究所科学研究部
・情報処理専門学校教育改定標準カリキュラム試案の指導の手引（1990-3）
専修学校教育振興会
・高等教育における情報処理教育の推進について（1990-9）
情報処理教育教員研修会における文部省高等教育局専門教育課長補佐レポート
・プログラミングに関する認知科学的研究(1)
松原康夫　文教大学情報学部情報研究第7号
・学習指導と認知心理学
E・D・ガニエ著　パーソナルメディア社