情報処理教育システム A-TRAIN の開発*
(第1報　新教育用言語 NELSON の開発)

堀 内 征 治**・堀 内 泰 輔***

1. はじめに

本校の本格的な情報処理教育は，昭和49年度に FACOM 230-25 中型電子計算機システムが設置されたときに，端を発している．このシステムは，バッチ処理方式をとっているが，多人数教育を効率的に行うために，入力メディアとしてのマークカードの採用，およびそれをベースとしたワンマーク方式の FORTRAN 言語処理システム COLT の利用を行ってきた．

しかし，カリキュラム変更に伴う情報関連の科目の増加や従来科目での積極的な計算機利用などにより，いくつかの問題点が指摘されている．それは第1に，FORTRAN 教育におけるアルゴリズム教育部門の不徹底さであり，第2に，マーク方式の非能率性に起因する，学生の負担の増加，およびジョブ件数の増大によるスループットの低下である．

これら諸問題を，現状のハードウェア資源の範囲内で解決しようとするのが，現在開発中の情報処理教育システム A-TRAIN (Algorithmic TRAINing system) である．図1に概要を示すが，これはソフトウェアとハードウェアの2本の柱からなる．その1つは新教育用言語 NELSON (New Educational Language System Of N. T. C) であり，構造化プログラミングに重点を置いたアルゴリズム教育に適している．もう一方は，ポケットコンピュータ

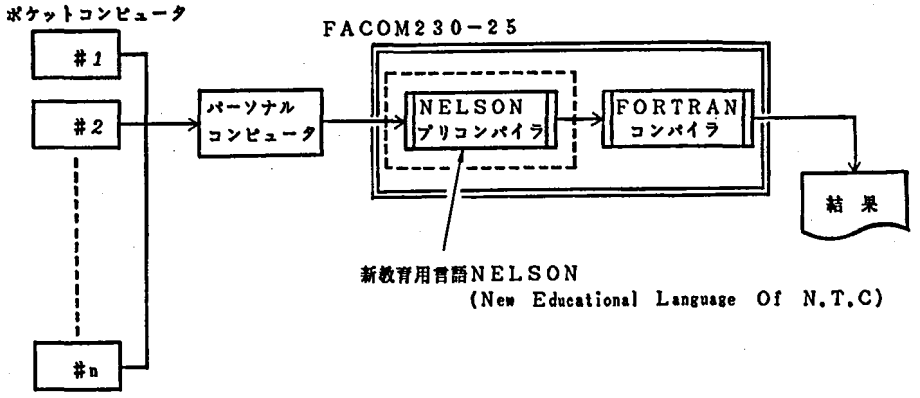

図1　情報処理教育システム A-TRAIN

---

*　昭和57年8月　全国高専情報処理研究協議会情報処理研究発表会において発表
**　機械工学科講師
***　機械工学科助手
原稿受付　昭和57年9月30日

を入力メディアとして利用を可能にする，一連のハードウェア群である．

本論では，このシステムについての第一報として，前者のソフトウェアシステムについて新言語の概要を述べ，これを初心者教育に適用した際の評価について論ずる．

2. 言語処理系の概要

情報処理教育用言語としては，FORTRAN 言語が広く使用されてきているが，この言語は必ずしも初心者教育用として妥当であるとは言い難く，各方面でその改良が行われている．

本稿で述べる，新教育用言語 NELSON も，そのような立場で開発したものである．FORTRAN 言語の特長を十分生かしつつ，その最大の欠点ともいえる非構造的部分を補い，従来軽視されがちであった，アルゴリズム教育を強化することを最大の目的としている．またこれに加え，言語教育面での容易性や，初心者のみならず一般利用者にとっての使いやすさ，などの点についても考慮した．

一般に，言語開発を行う手段として，目的とする言語を直接的に作成する方法と，実在する言語を生成する言語を作成する，いわゆる，プリコンパイル方式の両者があるが，本開発では，開発期間が短縮でき，汎用性に富む後者のプリコンパイル方式を採用した．処理の高速化のため，記述言語はアセンブリ言語を用いた．大きさは4000ステップ程度のものである．

この処理系のプロセスの流れを図2に示す．まず，NELSON 言語仕様のプログラムを入力すると，NELSON-3 プリコンパイラにより，FORTRAN プログラムにジェネレートされ，同時に，NELSON イメージのソースリストが出力される．この際，もしソース中に文法エラーがあれば，引き続きエラーリストを出力し，処理を終了する．この場合，FORTRAN コンパイラに後の処理を委託しないのは，2重のエラーリスト出力を防止するためである．一方，ソースにエラーがなければ，FORTRAN コンパイラは後の処理を委託され，従来通りの FORTRAN 言語としての一連の処理がなされる．

プリコンパイラ NELSON-3 は，ATRAIN 3 および FASPAL の2つのプログラムより構成される．前者は NELSON 言語を FORTRAN 言語にジェネレートするものであり，後

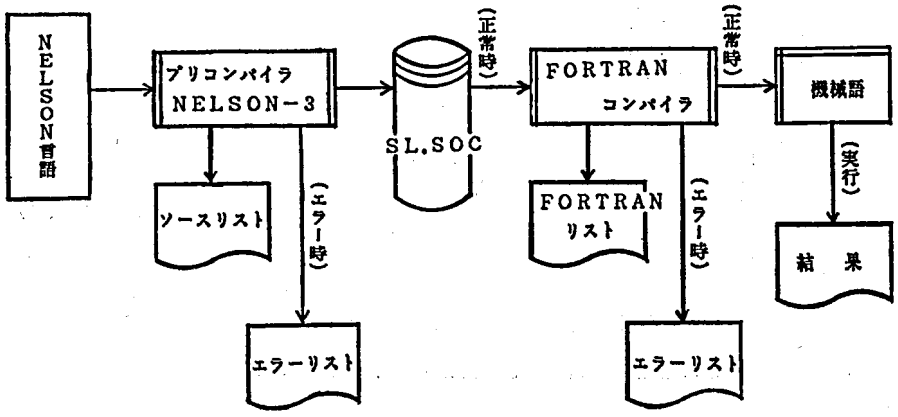

図2　言語処理系プロセスフローチャート

者はそれを外部記憶上のソースライブラリ（SL.SOC）に書き込むプログラムである．FORTRANコンパイラは，このソースライブラリよりソースを入力し，コンパイルを行い，最終的な実行がなされる．

3. 言語仕様

上述の目的を達成するために，NELSON言語仕様の決定に当たっては，プログラムの読みやすさ，書きやすさ，デバッグのしやすさ，プログラムメインテナンスの容易さ，および，アルゴリズムの教育・理解の容易さなどの諸点について検討を重ね，次のような要求仕様を設定した．

(i) プログラム構造をブロック化することにより，プログラムを読みやすくする．

(ii) ブロック化を行うために，PASCAL的な命令を導入する．

(iii) FORTRAN言語の命令を再検討し，拡張，改良，および省略を行う．

(iv) ソースリストを見やすくするために，自動的なレベルごとの字下げを行う．

(v) 入力仕様として，フリーフォーマット方式を採用する．但し，リストの見やすさ，修正の容易さの点から，マルチステートメントは禁止する．

以下，本言語仕様の中で特筆すべき点について述べる．

3—1 フリーフォーマット方式の採用

入力はフリーフォーマットであり，各命令は任意のカラムより書きはじめることができる．注釈行と続続行は，それぞれ，Cまたは＊，＋または＆の第一カラム記入により，命令行あるいは命令部と区別される．

3—2 演算子表記の拡張

演算子のうち，関係演算子については，大小関係を直接的に理解しやすくするための，＜，＞，＝などの表現が可能である．また論理演算子では，PASCAL系言語でよく用いられる，|，&，¬の記号が使用できる．

3—3 ステートメントラベルの新設

従来のFORTRAN言語では，文番号を用いて飛び先や参照個所を指定していたが，単純な数字列のみならず，英字による文字列も可能で，8文字までの英数字をラベルとして使用する．

3—4 欄記述子の改良

FORTRAN文中の欄記述子のうち，H変換はプログラムの書きやすさ，読みやすさの点で支障があり，教育上の弊害でもあったため，″または′のいずれかの記号で文字列の前後をくくるだけの，クォーテーションタイプが使用できるよう配慮した．

3—5 制御文の改良および新設

3—5—1 IF文の改良

FORTRANのIF文は条件が真であるとき，単一の命令しか記述できない．ところが実際のプログラム作成においては，複数の命令を条件の真偽それぞれの場合に記述したいことが多い．もちろん，FORTRAN言語でも分岐を用いれば可能であるが，GOTO文が多くなり，構造化プログラミングの面から望ましいことではない．これらのことから，他の言語では頻繁に見られる，IF～THEN～ELSE構造を採用した．

3—5—2 PASCAL 系命令の導入

構造化プログラミングという観点に立つと，従来の FORTRAN 言語の命令体系では，前述の低レベルな IF 文しか用意されておらず，はなはだ貧弱である．これを克服すべく，PASCAL 系言語の命令体系より，図3に示す，5種の構造化命令を採用した．初心者においては，制御文選択に迷ったり端末文のミスが多発しようが，慣れるに従い，自然に構造化プログラミングが会得できるものと思われる．

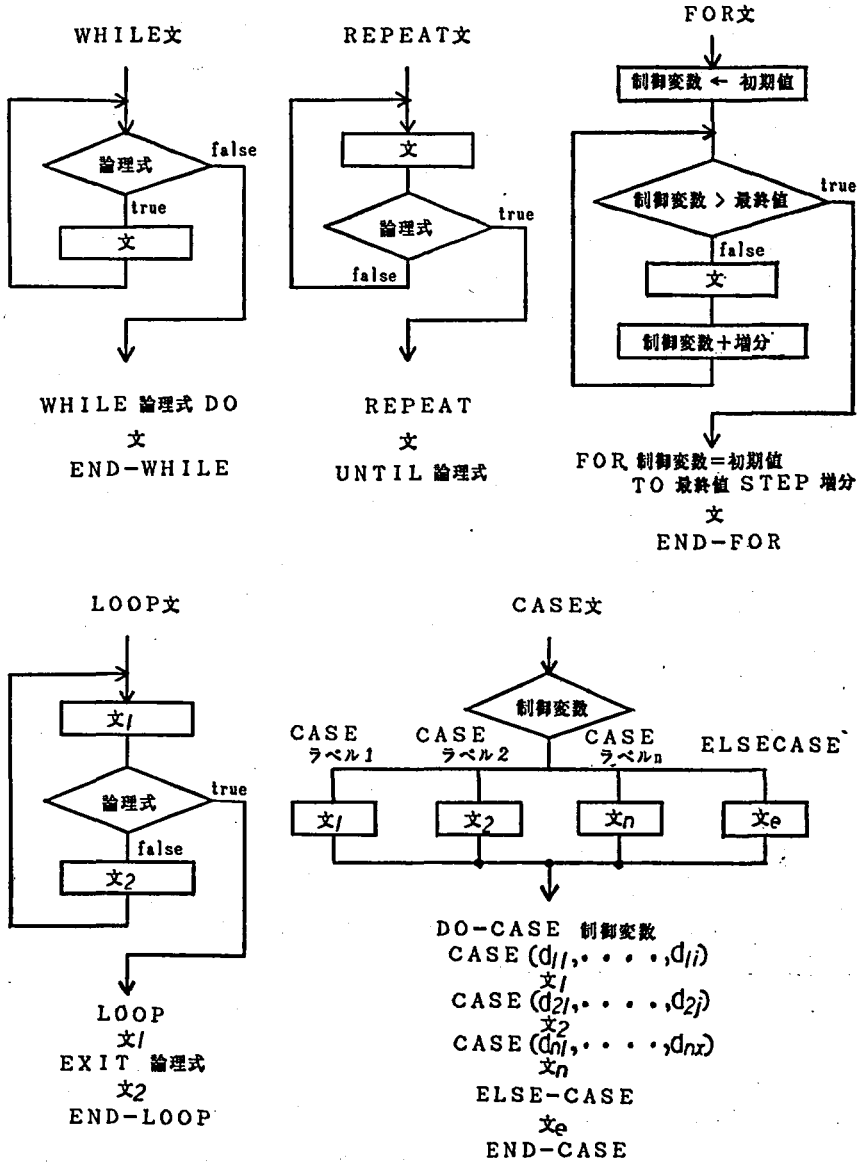

図3 PASCAL 系制御文の制御論理と記述形式

```
NTC   A-TRAIN   NELSON-3      -820318-        PROGRAM: MAIN

***  SOURCE PROGRAM IMAGE  ***

ISN   SSN          STATEMENT

      (001)  *   TEST OF NUMBER
001   (002)            WRITE(HEAD)
002   (003)            I=0
003   (004)            REPEAT
004   (005)              I=I+1
      (006)              DO-CASE I
005   (007)              CASE (1,3,5,7,9)
007   (008)                J=1
008   (009)                WRITE(CAPPO) I,J
011   (010)              CASE (2,4,6,8)
013   (011)                J=2
014   (012)                WRITE(CAPPO) I,J
016   (013)              ELSE-CASE
017   (014)                J=0
018   (015)                WRITE(CAPPO) I,J
019   (016)              END-CASE
020   (017)            UNTIL I=10
021   (018)            STOP
022   (019)       HEAD:FORMAT(5,"TEST OF NUMBER"/2,'    I    J')
023   (020)      CAPPO:FORMAT(2,2I5)
024   (021)            END
```

(a)

```
 FACOM  BOS   FORTRAN  -731103-  V02 L19          82.09.29  PAGE   1

*** SOURCE LIST ***

ISN          STATEMENT

  1          WRITE(6,001)
  2          I=0
  3     002  CONTINUE
  4          I=I+1
  5          IF(.NOT.(I     .EQ.1.OR.I     .EQ.3.OR.I     .EQ.5.OR.I     .EQ.7.
  6         +OR.I     .EQ.9)) GO TO 004
  7          J=1
  8          WRITE(6,005) I,J
  9          GO TO 003
 10     004  CONTINUE
 11          IF(.NOT.(I     .EQ.2.OR.I     .EQ.4.OR.I     .EQ.6.OR.I     .EQ.8)
 12         +) GO TO 006
 13          J=2
 14          WRITE(6,005) I,J
 15          GO TO 003
 16     006  CONTINUE
 17          J=0
 18          WRITE(6,005) I,J
 19     003  CONTINUE
 20          IF(.NOT.(I.EQ.10)) GO TO 002
 21          STOP
 22     001  FORMAT(5(/)1H+,014HTEST OF NUMBER/2(/)1H+,010H    I    J)
 23     005  FORMAT(2(/)1H+,2I5)
 24          END
```

(b)

図4 NELSON 言語ソースリストおよび FORTRAN 言語生成リスト

3—6 出力仕様およびジェネレーション技法

出力仕様を示すために，本システムの実行例を図4に掲げる．(a)は NELSON のソースリストであるが，レベルごとの自動的な字下げがなされ，また，ジェネレート後の FORTRAN

命令との対応番号を印字することによって，プログラムを見やすくし，デバッグ時の能率を上げるのに貢献している．

(b)は(a)を FORTRAN 言語にジェネレートしたリストを示す．ジェネレーションは，定型的な埋め込み方式を用いるため，最適化していないが，後述するように，実行時間の大幅なロスには至らない程度のものである．

4. システム評価

4—1 言語処理系の処理速度

多人数教育用という観点から，システムの実行時間は，これら言語の大きなファクタとなる．この点に着目し，同一の実行結果を示すような FORTRAN と NELSON，それぞれのプログラムを用意し，ベンチマークテストを実施した．これは，各ジョブタイムおよび各ジョブのステップごとのラップを測定する方法をとった．

処理時間は，NELSON 方式が，プリコンパイル時間 (Nt_0)，FORTRAN コンパイル時間 (Nt_1)，および実行時間 (Nt_2) の3要素から成るのに対し，一方，FORTRAN ジョブにおいては，プリコンパイル時間がないため，コンパイル時間 (Ft_1) と実行時間 (FT_2) の和で与えられる．

この結果である，最大値・最小値および平均値をグラフ化したものが図5である．前述のようにジェネレーションの最適化を行っていないため，実行時間は，若干，NELSON の方が多くかかっているが，1ステートメント当り，最大0.1秒であり，さほど問題とはならない．

FORTRAN コンパイル時間の比較においては，かなりの差が生じているが，むしろ，コンパイルまでの所要時間を含めた (Nt_0+Nt_1) と Ft_1 との比較が有用である．これは FORTRAN 方式では，マークカードが直接，コンパイル入力になるのに対し，NELSON 処理系においては，プリコンパイル入力としてのマークカード入力と，FORTRAN コンパイル入力としての外部記憶（ディスク）の入力，の2回に渡るためである．この増分と，実質的なプリコンパイル時間との和だけ，余分に時間がかかることは否めない．

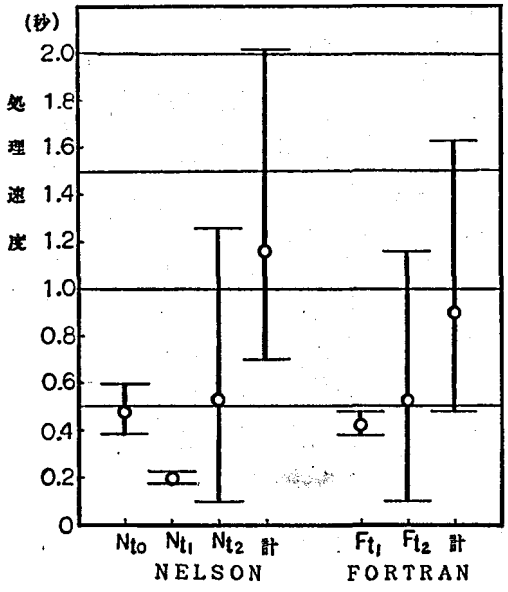

図5 1ステートメント当りの処理速度の比較

しかし，たとえば，40～50秒のジョブタイムを示す，FORTRAN プログラムにくらべ，NELSON プログラムは3秒程度，すなわち1割に満たない遅れですむというのは，プリコンパイラとしては，高速の部類に入るものと思われる．

更に，今回の評価が従来のマークカード入力によったのに対し，A-TRAINシステム全体が完成した時点で，入力メディアがポケットコンピュータになった場合，その高速性と，パーソナルコンピュタでの分散処理化などにより，上述結果が逆転することが予想される．

4—2　初心者教育の適用例

本言語の主目的は，「初心者にわかりやすい言語」である．この目的の達成度を実測するために，本校電算機部1年生17名を2分し，FORTRAN言語学習グループ(以下Fグループとよぶ）とNELSON 言語学習グループ（以下Nグループとよぶ），それぞれに対し，初心者向けのプログラミング言語教育を実験的に実施した．教育内容は基礎的事項から，配列や関数が扱える程度までとし，5回の小テストと3回の演習（実習を含む）をカリキュラムに盛り込んだ．

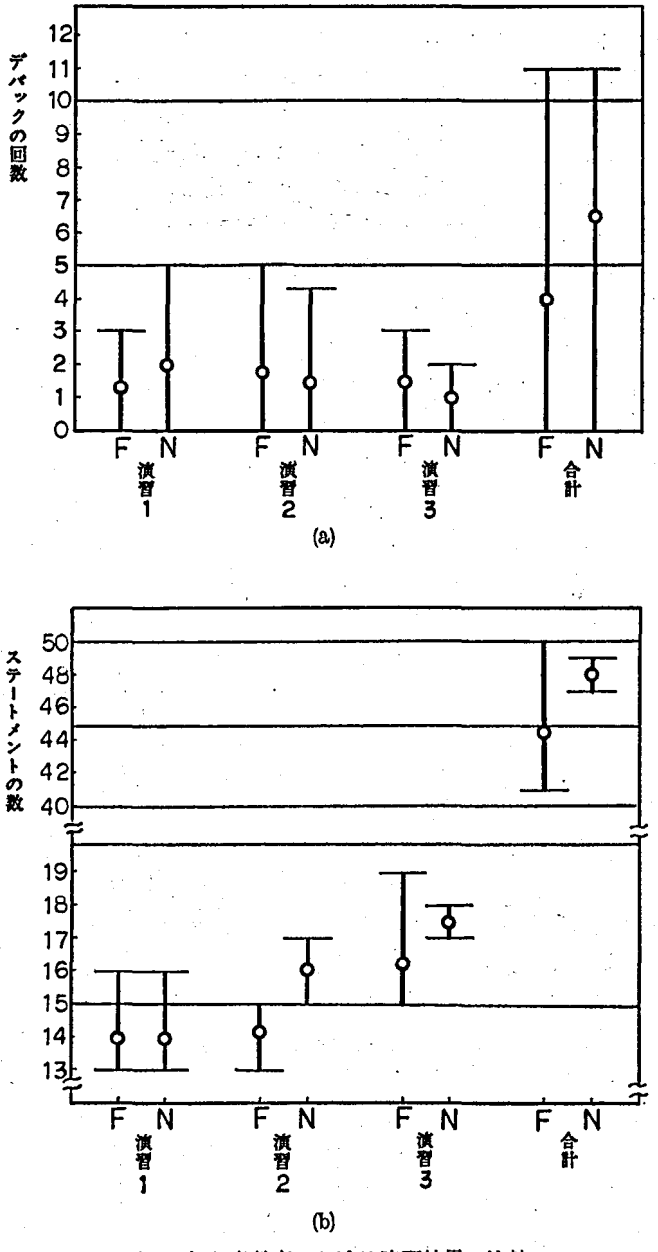

図6　初心者教育における演習結果の比較

結果の第一として，教育に要する時間の比較では，予想通り，FORMAT文や関係・論理演算子に関してはNグループへの教育時間は短縮でき，いわゆる，文法教育の第一段階での混乱の減少が証明された．逆に，制御文に関しての教育時間は，Nグループの方が70分程度多い．これは PASCAL 系の命令の理解に要した時間であるが，構造化プログラミングを，早期の段階から無意識に教育している，という面に立てば，所要

時間の若干の増加は，必ずしも，ディメリットとはならないであろう．

次に言語の理解度であるが，5回の小テストと3つの簡単な演習により，結果を得た．最初に基本事項の小テストを行い，初期段階での能力の均一性を確認した．入出力文関係の小テストでは，Fグループの H変換の混乱に対し，Nグループでのクォーテーションタイプの有用性が実証された．しかし，フローを与え，それからプログラミングを行わせる小テストでは，Nグループに制御構造の末端文の誤りが目立った．また，制御文の選択の誤りも見られたが，その指摘により効率的なデバッグが実現した．更に，Fグループにおいて，関係演算子の記述の誤りが多かったのに対し，Nグループでは，ほぼフローチャート通りプログラミング可能であるため，総合的な所要時間の面で，Fグループに優った．これらのことから，NELSON 言語は，制御構造に慣れさえすれば，わかりやすい言語であり初心者にとって，有用な言語である，という確証を持った．

最後に，3回に渡る演習の結果として，デバック回数とステートメント数について，グラフ化したものを，図6に示す．前者の比較では，Nグループは当初，比較的，回数が多いが，演習が進むにつれ，わずかではあるが，Fグループより少なくなることが見い出された．一方，(b)のステートメント数の比較では，演習内容により差はまちまちであるが，合計に着目すると，Nグループの方がステートメント数は若干多いものの，ちらばりが極めて少ないという結果は注目に値する．これは NELSON 言語教育では，学ぶ側にレベル差を生じさせぬ現われと思われ，ひいては，従来の FORTRAN プログラムに見られる，職人芸的な技法から脱皮して，標準的な構造化プログラミングを習得できるものと考えられる．

5. あ と が き

情報処理教育用言語を開発し，実際に初心者教育を実施することにより，この言語の評価を行った．その結果，初心者教育の面では，問題なく使用でき，従来の FORTRAN 言語に比べ，多くのメリットがあることが実証された．

しかし，このことは，初心者教育に限って言えることであり，多くの非構造的プログラマの再教育に，果たして有用であるかは，まだ評価を行っていないため，疑問が残る．長年の FORTRAN 一辺倒のプログラマに対し，これを道具としてアルゴリズムを教育することは，既に培った非構造的思考体系を覆すことに等しい．このため，言語の記述能力にも増して，それに適合した，教育方法を考えていかねばならないことも，事実である．

今後は，上述の残された評価を行うとともに，A-TRAIN 全体の開発と相まって，より良い，時代にマッチした情報処理教育システムを模索しつつ構築して行きたいと考える．

なお，本研究は，昭和57年度科学研究費補助金の助成を受けたものであることを付記する．

おわりに，本研究の一部は本校機械工学科の卒業研究の1テーマとして与えたものでもあり，ほぼ全体のプログラミングをしてくれた，一本木敏盛君（現エプソン(株)）と，評価を行ってくれた品田聡君の両名に感謝の意を表する．

参 考 文 献

(1) 間野浩太郎，プリコンパイラ・プロセッサ，情報処理研修センター，1980.

(2) 米田信夫他，PASCAL プログラミング，サイエンス社，1979.