

43—4

電 子 計 算

技 術 管 理

土 木 研 究 所 資 料

第 394 号

電 子 計 算 用 語 ALGOL

昭 和 43 年 4 月

建 設 省 土 木 研 究 所

電子計算用語 ALGOL

建設省土木研究所

技術管理室長

工学博士 中村 慶一

目 次

はじめに	1	17. KDK ALGOL の構想	30
1. 建設技術と電子計算	2	17.1 型	30
1.1 電子計算組織の導入を必要とする理由	2	17.2 式 <i>expression</i>	30
1.2 建設事業における電子計算向き業務	3	17.3 文 <i>statement</i>	31
2. ALGOL とは	4	17.4 手続き <i>procedure</i>	31
2.1 はじめに	4	17.5 宣言 <i>declaration</i>	32
2.2 ALGOL の概要	5	18. 各機種間の文法制限と入出力文	32
2.3 平均と不偏分散の平方根の計算プログラム	7	18.1 TOSBAC 3400-40 用 NEW T ALGOL	33
2.4 普通の言語と ALGOL	10	18.2 OKITAC 5090-H 用 NEW ALGOL-H	33
3. ALGOL の基本原則	11	18.3 FACOM 230 用 230 ALGOL	34
4. 宣言 (<i>declaration</i>)	12	18.4 FACOM 230-50 ALGOL	35
5. 式 (<i>expression</i>)	12	18.5 TOSBAC 用 KT-ALGOL	35
6. 文 (<i>statement</i>)	13	18.6 CDC-3600 用 ALGOL	35
6.1 基本文	13	19. アルゴルプログラムの変換	36
7. ALGOL によるプログラム例	13	19.1 立方根の問題	36
7.1 不偏分散の平方根の計算(1)	15	19.2 NEW ALGOL-H への変換	36
7.2 不偏分散の平方根の計算(2)	16	19.3 FACOM 230 ALGOL への変換	36
7.3 一元配置の分散分析	16	20. ALGOL と FORTRAN の比較	37
8. アルゴルによるプログラミングの簡単な例題(I)	17	21. 練習問題(Ⅲ)	38
8.1 相 関 係 数	17	22. 練習問題解	38
8.2 数 表	17	23. 電子計算機	39
8.3 電気料金計算	17	23.1 電子計算機の構成	39
8.4 2 次方程式の根	17	23.2 入力装置 (<i>input device</i>)	39
9. 例 題 解 (I)	18	23.3 制御装置 (<i>control unit</i>) および 演算装置 (<i>arithmetic unit</i>)	40
9.1 相 関 係 数	18	23.4 記憶装置 (<i>memory</i>)	40
9.2 数 表	18	23.5 出力装置 (<i>output device</i>)	40
9.3 電気料金計算	18	23.6 電子計算機の歴史	40
9.4 2 次方程式の根	19	23.7 電子計算機一覧	40
10. 手続き (<i>procedure</i>)	19	24. 重力式コンクリート擁壁自動設計プログラム	42
10.1 概 要	19	25. 重力擁壁自動設計のプログラム	43
10.2 手続きの実例	20	26. 自動図化	45
10.3 <i>value</i> 指定と規制	20	27. これまでのまとめ	45
11. 関数呼び出し (<i>function designator</i>)	22	28. ALGOL マニュアル	45
11.1 概 要	22	28.1 基本記号, 名前, 数値, 記号列	47
11.2 関数呼び出しの実例	22	28.2 式 <i>expression</i>	47
12. スイッチ (<i>switch</i>)	24	28.3 節 <i>clause</i>	49
13. アルゴルによる例題(Ⅲ)	25	28.4 文 <i>statement</i>	49
13.1 計画量の危険度の評価	25	28.5 入出力文	49
13.2 一元配置の分散分析	25	28.6 宣言 <i>declaration</i>	50
14. 例 題 解 (Ⅱ)	26	28.7 デ ー タ	51
14.1 計画量の危険度の評価	26	29. 改訂アルゴル 60 の構文法	51
14.2 一元配置の分散分析	26	29.1 一般的説明	51
15. 特殊な注意事項	27	29.2 基本記号	51
15.1 <i>array</i> の要素の取り扱い	27	29.3 式 <i>expression</i>	52
15.2 繰り返し文の終期値の取り扱い	28	29.4 文 <i>statement</i>	53
15.3 <i>end</i> の後の ; の取り扱い	28	29.5 宣 言	54
15.4 繰り返し文の制御変数	29	お わ り に	54
15.5 アルファモードの取り扱い	29	参 考 文 献	55
16. プログラムのエラーチェック	29		

は じ め に

あちらでもこちらでも電子計算機が入ったとか、電子計算を行なっているという話が広がり始めると、「私達の所では何を電子計算機にかけたら良いか検討中です」とか、「計算機に何をやらせたら良いか教えて下さい」とか、また逆に「私の所には電子計算機にやらせるような仕事はありません」といった話をよく聞かされる。

いったい自分達の仕事で電子計算機にやらせることのできない仕事とはどのような仕事ののだろうかと戸惑うのであるが、おそらく、その時の出来心でまったく基準なしに判断を下しつつ行なわれる仕事か、自分なりの判断の基準はあっても陳情かなにかに従って判断の行なわれているような仕事か、さもないれば他人にはうかがい知ることのできない、その人だけに与えられた神秘的な判断力（それは経験によるものもあろうし、あるいは先天的なものかも知れない）によって遂行される仕事でもあろう。

しかし、一般の何等かの判断の基準に基いた手順のわかっている仕事はすべて電子計算させることが可能であるはずで、問題は電子計算をさせて得になるような仕事であるかどうかという点であろう。たしかに100個位のデータの和を求めたり平均を出したりする仕事は、データテープを作っている間に算盤を入れ終わることができるのであるから電子計算をさせると損だということになろう。しかし1群のデータが得られたときに、そのデータが全体としてどのくらいの大きさであるかという位置の情報だけを抽出して、データのちりばり具合が示すばらつきを無視することは、せつかくそのデータから将来の予測を行ない得る機会を自から放棄してしまっていることになる。

さらにそのデータに関係している他の因子との関連性の情報のことと考えれば、これらの情報の抽出に要する手数は電子計算の方が問題なく少ないし、また所要時間も短い。

しかも最近の電子計算機には、あらかじめ辞書覚えておいて、後でより大きなプログラムの中で、あの方法はこの情報を用いて実施し、その結果をこのように取り扱えといった教え方が可能になり、事実われわれは、ある意味での電子計算機の教育を行ない得るようになってきている。

ただ、この電子計算機との対話のために用いられる言葉が、以前は人間の方が機械の言葉を覚えてこれを用いなければならなかったのに、だんだん人間の言葉に近いシンボリックな言葉を用いて機械自身に翻訳も行なわせるようになり、1956年にはIBM-704機に対してFORTRAN (formula translator) が生まれ、より人間の言葉に近いもので書いてやればすむようになり、さらに1958年電子計算用語ALGOL-58さらにALGOL-60および1962年の改訂ALGOL-60が国際共通の電子計算用語として生まれ全世界で使用されつつある。

ALGOLはFORTRANに比して文法体系がすっきりしているだけでなく、われわれの言葉に非常に近くなる順序とか記号のみにによる約束が少ないため、われわれと電子計算機との対話に用いられるだけでなく、言葉の異なる人達や、更にわれわれ日本人同志の間でも手順をわかりやすく、かつ正確に伝達する言葉として最適であると考えられる。

電子計算機との対話には別段計算機というものの構造に関する知識はもちろん、昔プログラマなるものの唯一の武器であった機械語に関する知識、あるいはSIP, SAPなどの中間語に関する知識もいささか必要であつて、われわれの言葉に近いALGOLでプログラムを書いてやれば機械が文法チェックを行ない、エラーがなければ中間語を経由して機械語への翻訳ならびにそれによる演算を実行する。

したがって単に電子計算機との対話をスムーズに行なう目的からはALGOL以外の知識は別段いらないわけであるが、昔からプログラマたちが自分達の相対的価値を引上げる目的で、これらの知識が入門として必須であるかのように主張し、その効あつて初めての人達が電子計算機というものに親しもうとして読み始める文章の最初にはまず電気回路か、機械語の話が始まるため、多くの人に電子計算機というものはむずかしいもの、したがってそれと会話を行なっているプログラマは偉い人達という印象を与えがちであつた。

ALGOLの生まれた今日では最早プログラマという特別の職業を持った人が通訳を行なうというよりは、問題を持つ一人一人の人がALGOLによって直接会話することが普通となつてきている。

そこでまづ電子計算組織の導入を阻んでいる考え方について考察を行ない、次にALGOLが一通り理解できるための基礎知識を御紹介し、最後に電子計算との対話には別段必要と言ふほどのことでもないが、電子計算組織を導入しようと考えられた人がメーカーのカタログを読む上に必要な最小限の知識などについて触れてみたい。

1. 建設技術と電子計算

建設技術において、構造物の設計・計算に属する部分については相当な進歩が見られ、電子計算組織の普及につれて、ある仮定のもとでは、ほぼ満足すべき設計上の成果が得られているにもかかわらず、施工技術の上においては多少の、調査・技術の面では相当の立ち遅れが見受けられる現状である。

しかし数学が主として取り扱ってきた対象が、光・熱・振動・音・力などの物理現象であった時代から、経済問題・社会問題や心理学などにまで拡がり、計量経済学・計量心理学などの体系化が進むにつれて、実質科学の側からの要請による数学的方法論の発展と、数学的方法の開発に伴う実質科学の進歩との両方が互いに影響し合っており、今日のオペレーションズリサーチ (OR) やそれを含めたインダストリアルエンジニアリング (IE)、システムズエンジニアリング (SE) の体系化が行なわれて来た。

我々建設技術関係者の側からみても、建設技術も近代工業技術の一環であるという立場からすれば、このような他の分野の要請によって組み立てられた IE 或いは SE の諸技法を、そのままの形で適用し得るものも多いが、一方大に結びついた構造物を不特定多数の人の利益に供するために造り出していくという特殊性から見て、その調査・計画・設計・施工の各段階において、他の分野と全く異なると言えないまでも、他の分野ではほとんど要請もないような特殊な形の問題を多く持つっており、そのままの形で適用可能な技法の中にも、建設技術という場合に合わせて一部変形した方がよいものも多い。

この諸技法を仮りに土木計画画の手法と名づけると、これらは計画技法・計算技法・情報処理技法・管理技法・計量技法の5種に大別できる。何を価値判断の基準として、それをどのように処理して情報抽出を行ない、どのように計画し、かつその計画の線に沿って管理するかということに関連したこれら諸技法を進めるに当たっては、事実より法則性を帰納する為の統計解析と、これらの情報を与えられた手順に従って処理する為の電子計算が、基本的に必要な道具であるということになる。

建設関係に於ては付属三機関 (地理院・建研・土研) に引続き、若手工事事務所などに電子計算組織が導入され各地建を含めて 1500 名の ALGOL プログラマの養成も終り、各技術者が自分自身の問題をもって直接計算機と対話することにより、今迄予想されていた様な作業は勿論、今まであきらめていた様な或いは思いつくこともなかった様な問題と取り組み始めて居り、土研に於ける割り込み計算や、超大型機の衛星計算機としての使用、或いは北陸・中部・九州地建に於ける計算センタとの時間単価契約による自由使用などによって既に相当な成果が得られている。

従ってこれらの成果についてその大要を報告すると共に、

特に興味のある主題についてそのプログラムテクニック上、或いは電子計算組織の利用が可能となった為に生じた新しい観点などに触れつつ人力との対比などをあげて説明する。

また、現在建設技術関係の各組織に於て、電子計算組織の導入が計画されつつあるが、これに関連して(1)その必要性を十分納得させるだけの理由がない。(2)今でもコンサルタントを利用して間に合っているものを自から手を下してやる必要はない。(3)計算機を入れても十分な仕事量がない。(4)単なるあそびで導入するのは反対である。などいろいろな反対論も多い。

各会社などに於ても、導入の当初に於て、全く同じ議論がくり返され、(1)自分の業務遂行上の考え方の壁をつき破る、(2)人間を雑務から開放する、(3)その方がもうかる、(4)他の会社でうまく行っているのだからこちらでもやってみよう、などいろいろな理由で、大半の大組織は自から導入、中小企業は共同使用という形で導入し、業績を伸ばしている所が多い。

その中でも特に FORTRAN ALGOL 等のコンパイラを活用し、多くの社員に開放して成功している組織として東洋工業 (F)、川崎重工業 (A)、トヨタ自動車 (A) などの例があり、土木研究所に於ても当初にあげたと全く同じ反対があった時に、この道具を用いて研究上の壁を破る。従って今迄外注されていたプログラムで手を加える必要のないものや、所内でテスト終了したもので本番計算に時間を要するものは、今後も積極的に外注し、所内の計算機はわざわざ成功するかどうか不明であるプログラムのチェックを主目的とした外部の超大型機の衛星計算機であると割り切つて OKITAC 5090-II (メモリー 16K 長語) を導入し、(1)内部は勿論外部の計算機との接続にも使用用語は ALGOL 一本とする、(2)15 分以内の計算は割り込み使用を許し、見掛け上 100% の稼働とする、(3)使所で作られたプログラムは電送回路により送られ、エラーは同回路によって直ちに送り返す、(4)開発された手続きはできるだけ辞書の形で格納し、その名前を呼び出すだけで自動的に使用可能にする、といった特徴的な使用法に依り、各研究室に於ても相当な実績をあげた事ができた。

以下最初の反対論に対する反論、建設事業に於ける利用面などについて触れてみたい。

1.1 電子計算組織の導入を必要とする理由

標記の理由について考える前に反対論の(2)(3)に対する反論を試みよう。

(1) 「コンサルタントを利用して現に間に合っているものを、手を下して自から行なう必要はない」との意見について。

各組織内に技術者が存在するのはいろいろな技術的業務について重要なものは自から手を下して行なうことは

勿論、外注した場合もその進行状況を監視し、成果を評価し、状況の変化に応じ自分の意図に沿う様に修正を行ない、或いはその外注の為の積算、その全体の計画、或いは出来上ったものの技術的管理を行ない、その間に得られた技術的成果を標準化して次の技術的水準の向上にそなえることであつて、これらの仕事は予算獲得係と予算差引係だけでは遂行不可能である。

従つて重要な作業は勿論のこと、くり返し発生する計算について、そのたびにコンサルタントにプログラム料込みで仕払いを行なう事の可否とか、手順を改良したいと思つても自から手の下しようがないような発注の仕方の是非を考えてみれば、自から後で手を加えやすい形でプログラムを作るとか、そのような形でプログラムさせ、後は自分の組織内の計算機であらうと、そのプログラムを使用する分借りの計算機であらうとどちらの形にもせよ後の修正の容易な言葉たとえばALGOLでプログラムする必要は異論のない所であらう。

一般にある内容を日本語で書き下す手数とALGOLで書き下す手数とはほとんど変わらないが、その状況下でその問題に素人である「いわゆるプログラマ」にその問題の本質をのみ込ませることは容易でない。このような説明をしている間に問題を持つ人がALGOLで書き下した方が後のためにほるかに容易である。

もし仕事の内容がコンサルタントしか分らない場合には今後ともプログラム作成も含めて外注ということになる。しかし後の改良を考えるとこの場合も修正容易なALGOLで書かれ、しかもプログラムを買取することを明確にした方がよい。しかしこのような総てを依頼する形の仕事ばかりで業務は予算獲得係と予算差引係だけの手で進め得るものならば、そのような職場に技術者を必要とするかどうかについても反省して見る必要があらう。

技術的なアイデア、技術の進歩はこのような計算手段が自由に利用できる環境に於て育つ可能性が多く、計算手段は単なる労力の置き換え以上の意味を持つものであると考えられる。

コンサルタントの育成とは現在のような単純労務提供業としてのコンサルタントを、しかも殆んど同じ製品をプログラム料込みであちらこちらに売るといふ今のような形のままで育成する意味ではないと考えられる。製図1枚をいくらで買うのではなく、その着想を買うことによつてこそ育成を計るべきで、コンサルタント側企業者側共に電子計算手段を持つことはコンサルタント業の育成になりこそすべし妨害にはなろう筈がないと考えられる。

- (2) 「電子計算させるべき仕事がない」という意見について。

土木研究所のALGOL研修に於て、「他人に手順を教えることの出来る仕事は何でも計算機にやらせる事が出来、それが2回以上繰り返される時は一般的に計算機の

方が有利であり、1回しかやらぬ仕事でも計算機なしには考えられぬ仕事も多い。そもそも計算機にやらせる事の出来ない仕事はその時の思いつき或いは過去の経験、勘などをもとに、人に教えることの出来ない(技能的な)仕事である」と繰り返して述べた。

結果を見て行なう判断の基準が分っている時はその手順をプログラムすれば良いし、その基準が教えられない時は自分のdecision makingの資料を作るように計算機に命ずれば良いのであつて、この意味で各技術者(事務関係者も)が自分の昨日、先週、先月、去年行つた仕事を振り返つて見ると、電子計算に移行可能であり、又その方が有利である仕事に相当な時間をさいていた事に気付く筈である。

プログラムを作る作業に対しては、各人が直接計算機と対話する為の習得容易な国際共通語を習得し、自分が得意とする分野を手分けしてプログラミングすれば良いわけで、その作業の配分の為には各組織にプログラム委員会が必要であらうし、またこの様に作られたプログラムは手続きの形にまとめ格納し、後で使用する人はその手続き全体につけられた名前を呼び出してプログラム中で使用すれば良い。

- (3) 部内に置かれた計算組織の効用

計算機の効用はこのような今迄やつて来た仕事の置換えに止まらない。むしろこのような仕事はほんの一部であつて、重要な部分はこのような電子計算組織の導入に依つて、今迄あきらめていたか或いは思いつきもしなかった様な新しい構想に基づく方法論に依るプログラムである。

この際計算機を組織内に置くことは「失敗するかもしれないがとにかくやってみよう」という考え方が「失敗するかもしれないから止めて置こう」という考え方に取つて代るという利点をもたらす。この考え方の差は重要で、やがて時間の経過と共にますます大きな開きとなる事が予想される。

この他積算・路線選定など秘密を要する仕事、人事其他公物の管理など本来他に出し得ない種類の仕事や、即時性を要する作業の存在を考えると、この新しい発想を生み出す効用もあわせ、部内に計算組織を持つ必要性は大きいと言わざるを得ない。

1.2 建設事業における電子計算向き業務

前にも触れた様に、我々の行なっている業務はその時々思い付きや気分次第で判断されたり、他人には説明できないような過去の経験や感による業務以外はすべて電子計算させる事が出来るし、又電子計算させた方が有利である。しかしある程度人を納得させる為には例をあげた方がよいと思われるので4つの分類のもとに例示してみよう。

- (1) 本来部門の計算組織に依らなければならぬもの

きのあること、 $\square$ は空白の意であること、 $\text{sqrt}(46)$ とは $\sqrt{46}$ の意であること、 $\text{enter}(43)$ は4となること、 $\times$ は掛算で、 $\div$ は割られる場合が多いこと、 $+$ は整数割算で5 $\div$ 2は2となること、 $\uparrow$ は乗べきで A^4 は $A \uparrow 4$ と書かれること、 $\wedge$ はand(共に成立)、 $\vee$ はor(どちらかが成立)、 $\neg$ はnot(否定)などの意であること、および K_{ij} などの下つきサフィックス表現が右肩の乗べき表現と共に ALGOLには無いので、 $K[I, J]$ などと四角括弧の中に入れて添字を表わすものであることなどを述べておけば、他人の書いたプログラムを読む上には事欠かない筈である。

2.2 ALGOL の概要

ALGOLは習得容易な国際共通語で、その習得には実習を含めて2日間もあれば十分であり、読むだけならば45分もあれば十分である。以下この45分コースの概要を御紹介して置こう。

(1) 問題

従量電灯乙で、契約電力10, 15, 20アンペアの3種がある場合、その人の番号(N)、契約電力(A)、使用電力量(KWH)を知って電気料金(F)を計算する手順は次のようなものである。

- ① 6番の人は10アンペア契約で5.68キロワット時使用したとする。
- ② 基本料金(BF)はもし A が10なら19.35円、そうでなくてももし A が15なら29.00円、そうでなければ(当然 A は20)38.44円である。
- ③ 電力量料金(KF)は使用電力量当り10.89円である。
- ④ 電気料金(税込み)は BF と KF の和であるが、その結果が42.9円以下の時は7%の税金分は差引いたものを料金とする。
- ⑤ 料金は円単位に丸めて番号と料金を1行に印刷する。

これをALGOLプログラムで書くことによる。

```
begin
  integer N;
  real A, KWH, F, BF, KF;
  N := 6; A := 10; KWH := 568;
  BF := if A = 10 then 19.35 else if A = 15
        then 29.00 else 38.44;
  KF := 10.89 * KWH;
  F := BF + KF;
  if F ≤ 42.9 then F := F / 1.07;
  PRINT(4, N); PRINT(5, F); CRFL(1);
end;
```

このプログラムは、まさに上の日本語をそのまま書き

下したものに他ならないがこれを読み下す為の最小限の基礎知識は次のようなものである。

- 1 ALGOLプログラムは文の集積でありこれらの文は;で区切られる。
- 2 文はbeginとendでくくって1つの複合文とすることができ。
- 3 プログラムの中で変数などには勝手に名前を付けて良いが、この名前はbeginに続く宣言の部分で、整数ならinteger、小数点や10のべきを含む実数ならrealなどとして宣言されなければならず異なる意味に同じ名前を使用してはならない。
- 4 整数タイプの印刷準備にはPRINTとして有効桁数と印刷すべき内容を示し、その印刷を実行し何行おくべきかはCRFLによって指示する。実数タイプはPRINTRとする。(これは今の場合だけの約束とする)
- 5 値を代入することは:= (コロンイコール)によって指示し新しい値が代入された時は古い値は失われる。
- 6 掛算には*(エックスとの混同を避ける為)、一般の割算には/を用いる。

(2) 特定の値でない計算用としての一般化

$N = 6$, $A = 10$, $KWH = 568$ の場合でなく一般的にプログラムが使用できる為には、プログラムテープ上の N , A , KWH に別のデータテープ上から値を読み込むことにしてデータが変わった時はデータテープだけを取り変えれば良い。また KF を一旦計算しなくても下の計算で同時に行っても良いので次のプログラムに改める。

```
begin
  integer N;
  real A, KWH, F, BF;
  READ(N, A, KWH);
  BF := if A = 10 then 19.35 else if A = 15
        then 29.00 else 38.44;
  F := BF + 10.89 * KWH; if F ≤ 42.9 then F :=
        F / 1.07;
  PRINT(4, N);
  PRINT(5, F); CRFL(1);
end;
```

6, 10, 56.8,

新らしく追加された基礎知識は次のようなものである。

- 7 データテープは最初のbeginに対応するendの後に与えるがその順序はREADという命令に対応した順序にカンマでつないで置けば良い。(これは今の場合だけの約束とする。)

尚5行目から9行目までは今後くり返し現れるので、

この部分を A, KWH を知って F を計算する手続きをまとめてみると、この部分は

```
procedure RYOKIN (A, KWH, F);
  real A, KWH, F;
begin real BF;
  BF := if A = 10 then 193.5 else if A = 15
        then 290 else 384.4;
  F := BF + 10.89 * KWH;
  if F ≤ 429 then F := F / 1.07;
end;
```

そこでこの手続きのテープが出来ているものとしてこれを利用したプログラムに改めると

```
begin
  integer N;
  real AP, KW, FF;
  procedure RYOKIN (A, KWH, F);
  以下ここに前のテープをそう入する。
  READ (N, AP, KW);
  RYOKIN (AP, KW, FF);
  PRINTI (4, N); PRINTI (5, FF);
  CRLF(1);
end;
```

6, 10, 56.8,

となる。RYOKIN という手続きの仮パラメタは外のプログラムで使った名前とは無関係であることを示す為、ことさらに外では AP, KW, FF という名前を使用した。もちろん同じ名前でも差し支えない。

(3) 何人かの計算用としての一般化

M 人のデータが N, A, KWH の順に並んでいる場合、先頭の M に関する情報を読んで M 回くり返すプログラムは次のようになる。(たとえば、5680人分とする)

```
begin
  integer I, M, N; real AP, KW, FF;
  procedure RYOKIN (A, KWH, F);
  以下ここに前のテープをそう入する
  READ (M);
  for I := 1 step 1 until M do
    begin
      READ (N, AP, KW);
      RYOKIN (AP, KW, FF);
      PRINTI (4, N); PRINTI (5, FF); CRLF(1);
    end;
end;
```

5680, 6, 10, 56.8, 8, 20, 95.4, ...

(4) 人数不明の時の計算用としての一般化

何人のデータから分る時はたとえば最後のデータとして 0, 0, 0, を入れて置いてこれが来たらデータの終りと解釈させ、ついでに人数もプリントするプログラムは

次の通りである。なお最後の人数は $M =$ という字の右に打たせてみる。

```
begin
  integer I, N; real AP, KW, FF;
  procedure RYOKIN (A, KWH, F);
  以下ここに前のテープをそう入する
  I := 0;
  ST: READ (N, AP, KW); if N = 0 then go to EN;
  RYOKIN (AP, KW, FF);
  PRINTI (4, N);
  PRINTI (5, FF);
  CRLF(1); I := I + 1;
  go to ST;
  EN: PRINTS ('M=');
  PRINTI (4, I); CRLF(1);
end;
```

6, 10, 56.8, 8, 20, 95.4, として....., 0, 0, 0,

8字そのものを書きたい時は PRINTS の括弧の中に書きたい字を ' ' でつんで書いて置けばよい。

(5) 配列を利用した一般化

この例のような場合は特に変更のない限り、何番の人が何アンペアで契約しているということは分っており、この値をデータテープとして毎月パンチする必要はない。たとえば一番さんから N 番さんまでの契約電力を N を先頭に N 個パンチしておき、これをベクトルという配列に読み込み、 N 番目と指定すれば配列 N 番目より N 番さんの契約電力が取り出せる。(7000人と仮定する) 毎月電気料金を計算するにあたり、このデータテープに組数 (M) につづき N 番さんという情報と使用電力料を M 組パンチすればよい。

```
begin
  procedure RYOKIN (A, KWH, F);
  以下前のプログラム挿入
  integer M, N, I; real KW, FF;
  array A[1: 7000];
  READ (N);
  for I := 1 step 1 until N do READ (A[I]);
  READ (M);
  for I := 1 step 1 until M do
    begin real AP;
      READ (N, KW);
      AP := A[N];
      RYOKIN (AP, KW, FF);
      PRINTI (4, N);
      PRINTI (5, FF);
      CRLF(1);
    end;
end;
```

7000, 10, 15, 10, ..., 20, 5680, 6, 568, 8, 594,
..... 7000個

追加された基礎知識は次のようなものである。

- 9 A_i, j は $A[I, J]$ A^* は $A \uparrow B$
 $\frac{A+B}{C+D}$ は $(A+B)/(C+D)$
 と書かれる。
- 10 配列は宣言部においてその名前、添字の動く範囲を宣言しなければならない。
- 11 *begin* の後で宣言された名前は、その *begin* に対応する *end* までの間でしか意味を持たない。このような宣言が行なわれるところの部分をブロックと称することになり、外側のブロックで宣言された名前は、内側で同じ名前が宣言されていない限り通用する。
- 12 名札を：を介して文に付けることができるがこの名札はそれを含む最も内側のブロックの宣言部で宣言されたものとみなされる。
- 13 標準関数として *sqr*t (平方根), *abs* (絶対値), *sin*, *cos*, *arctan*, *ln* (自然対数), *exp* (指数関数), *sign* (符号), *entier* (整数に直す) がある。
 $\sin(A*B) + \cos(B)$
 などと使われる。

2.3 平均と不偏分散の平方根の計算プログラム

プログラムの成り立ちを十分なつくとしていただくために、もう一つ、平均と不偏分散の平方根を計算するプログラムを、やはりばかばかしい初歩的プログラムから順次程度の良いものにしていく過程を見てみることにしよう。プログラムの組み立て方が分ったあかつきには、直接進んだ段階のプログラムを組んだ方がよいことはもちろんである。

我々は、一群のデータ x_i によって予測を行なうときその一群の位置の測度としては算術平均 $\bar{x}$ を、散布度の測度としては不偏分散の平方根 $\sqrt{V_x}$ を用いることが多い。もちろん場合によっては中位数 $x_{(n)/2}$ や範囲 R_x をその代りに用いることもあるが、得たデータのもたらす情報の 100% を利用しているという点では $\bar{x}$ と $\sqrt{V_x}$ が最良である。ここに

$$\text{算術平均 } \bar{x} = \frac{n}{\sum_{i=1}^n} x_i / n$$

$$\text{不偏分散の平方根 } \sqrt{V_x} = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1}} = \sqrt{\frac{S(xx)}{\text{自由度}}}$$

$S(xx)$ の計算に当っては $\bar{x}$ を差引く段階で、 $\bar{x}$ の丸め誤差が大きくなってくるので

$$S(xx) = \sum_{i=1}^n x_i^2 - \left(\sum_{i=1}^n x_i \right)^2 / n$$

とした方がよいことは良く知られている。

(I) 特定のデータの場合作のプログラム

いま $n=3$ で $x_1=25.4$ $x_2=28.2$ $x_3=21.6$ として $\sqrt{V_x}$ を計算し有効4けたで印刷させてみよう。

(1.1)

```
begin
  read SX, MX, SXX, RV, X1, X2, X3; ①
  integer N; ②
  N := 3; X1 := 25.4; X2 := 28.2; X3 := 21.6; ③
  SX := X1 + X2 + X3; ④
  SXX := X1↑2 + X2↑2 + X3↑2; ⑤
  MX := SX / N; ⑥
  RV := sqrt((SXX - SX↑2 / N) / ⑦
    (N - 1)); ⑧
  PRINTR(4, MX); ⑨
  PRINTR(4, RV); ⑩
  CRLF(1); ⑪
end; ⑫
```

もちろん、④を省略して⑤、⑥、⑦で

```
SX := 25.4 + 28.2 + 21.6;
SXX := 25.4↑2 + 28.2↑2 + 21.6↑2;
MX := SX / 3;
```

としても良いが、ここでは後のプログラムとの関連を考えて変数 X 1等に受けてある。

このプログラムはいづれにせよ今プログラムで与えたデータに対する計算しか行なわないが、3回でなく N 回に拡大する為に次の準備をしておく。

(1.2)

```
begin
  read SX, MX, SXX, RV, X1, X2, X3, X; ①
  integer N; ②
  N := 3; X1 := 25.4; X2 := 28.2; X3 := 21.6; ③
  SX := SXX; ④
  SX := SX + X1; SXX := SXX + X1↑2; ⑤
  SX := SX + X2; SXX := SXX + X2↑2; ⑥
  SX := SX + X3; SXX := SXX + X3↑2; ⑦
  MX := SX / N; ⑧
  RV := sqrt((SXX - SX↑2 / N) / ⑨
    (N - 1)); ⑩
  PRINTR(4, MX); PRINTR(4, RV); ⑪
  CRLF(1); ⑫
end;
```

ここで⑥⑦⑧の右辺に SX , SXX が出てくるが、いずれも前の文で値を代入され、その結果を用いて式の値を計算し左辺に代入するものである。

さらに何回も出てくる所を手続きにまとめたとして次のプログラムに直しておく。

(1.3)

```

begin
  real SX, MX, SXX, RV, X1, X2, X3, X; ①
  integer N, I; ②
  procedure PR; ③
  begin ④
    MX:= SX/N; ⑤
    RV:= sqrt((SXX-SX2/N)/ ⑥
      (N-1)); ⑦
    PRINTR(4, MX); PRINTR(4, RV); ⑧
    CRLF(1); ⑨
  end; ⑩
  N:= 3; ⑪
  X1:= 2.54; X2:= 2.82; X3:= 2.16; ⑫
  I:= 1; SX:= SXX:= 0; ⑬
  START: ⑭
  X:= if I=1 then X1 else if I=2 ⑮
    then X2 else X3; ⑯
  SX:= SX+X; ⑰
  SXX:= SXX+X2; ⑱
  I:= I+1; ⑲
  if I>N then go to OUT; ⑳
  go to START; ㉑
  OUT: ㉒
  PR; ㉓
end; ㉔

```

今度は④～⑩にP Rという名の手続きの内容を宣言し⑬で使用している。また⑬でXに対してIの情報に従ってX₁, X₂, X₃のどれかを代入させるようにしておいて⑮⑱でこのXを使用している。またSX, SXXは左辺に出てくるが、それまでのSX, SXXの値を用いて右辺の式を計算し、その結果を左辺に代入するものである。

㉔で回数IがNつまり今の場合3より大きくなったらこのループを脱出さなければ㉔でループの始めに戻ることを命じている。

③～⑩の部分は如何に今後一般化の為とは言いがた、いさかかわつらわしい。このようなことは機械が自動的にやってくれないかと考えるのは極く自然であって、その為に次の繰り返し文を用いる。また⑮⑱と⑲⑳の代入の代りに、READとという読み込み文を用いて次のプログラムを得る。

(2) N個のデータを読み込み、平均と不偏分散の平方根を計算するプログラム

今度はデータテーブルを使用し、データテーブル上では個数Nに関する情報を先頭に、その後N個のX_iの値がカンマでつながって入っているものとする。

```

begin ①
  real X, SX, SXX, MX, RV; ②

```

```

integer I, N; ③
procedure PR; ④
begin ⑤
  READ(N); ⑥
  SX:= SXX:= 0; ⑦
  for I:= 1 step 1 until N do ⑧
    begin ⑨
      READ(X); ⑩
      SX:= SX+X; ⑪
      SXX:= SXX+X2; ⑫
    end; ⑬
  PR; ⑭
end; ⑮

```

3, 2.54, 2.82, 2.16, ⑮データ

⑥によってデータテーブルの先頭にある3という情報がNという場所に格納される。なおデータの読み込み口では、次のREADという指令に対応してすぐ読み込みができるように、次の2.54という情報が待機していることになる。⑩は繰り返し文でコントロール変数に初期値1を代入し、次の1つの文(今の場合は⑨～⑫のbegin endでつづかれた1つの複合文)を実行しステップ1であるから今までのIつまり1に1を加えIが2の状態で⑨～⑫を実行しまたIに1を加える。いつまでくり返すかと言うとuntil NつまりIの値がNを越すまでは続けるわけで、この部分が、前のプログラム(1.3)における⑬⑭などの代行になっている。また⑫のREAD(X);によってI=1, 2, に対応してデータテーブルから読み込まれた情報がXという場所に格納されるので、繰り返し文中に使用されたREADはちょうどプログラム(1.3)の⑮⑯の代行を行なっているとも考えられる。⑭のPR;によってこの場所に④⑤の宣言の所で定義された文(いまの場合プログラム1.3の⑥～⑨のテーブルがつき込んであればそれら)が呼び出されてPR;という部分に置き換えられ実行される。

(3) 配列を用いた場合の平均と不偏分散の平方根のプログラム

今までは繰り返し文の中で読み込み、Xという場所を共用し、前のデータは消してしまっていたが、別にX₁という名の配列にデータを格納して、このX₁の何番目という指定によって使用できると、データが別の目的で後で使用するのに便利である。またデータ個数Nが不明で、データの最後に、データ中には出て来ないような大きな数、たとえば10²⁰などという数字を入れておき、データ数をカウントすると共にこれが来たらデータの終りと解釈させることにしよう。データの個数は今度は制限があり、たとえば2500個以下とする。

```

begin ①
  real SX, SXX, MX, RV; ②

```

```

array XA[1:2500];           ③
integer I, N;                ④
procedure RR;                 ⑤
begin 1.3 ⑤~⑦を書く end;    ⑥
N:=1;                         ⑦
START: READ(XA(N));          ⑧
if XA(N) ≥ 0.70 then go to NEXT; ⑨
N:=N+1; go to START;         ⑩
NEXT: SX:=SXX:=0;            ⑪
for I:=1 step 1 until N-1 do ⑫
begin                          ⑬
    SX:=SX+XA[I];             ⑭
    SXX:=SXX+XA[I]↑2;         ⑮
end;                           ⑯
PRINTS('N='); PRINTI(4, N-1); ⑰
CRLF(1);                      ⑱
PR;                            ⑲
end;                           ⑳

```

⑥のように READして格納する場所は配列の何番目という指定でも良い。②~⑥は宣言の部分であるが、宣言の順序は何であつてもよい。プログラムの本体は⑦から始まっている。⑦では N に初期値 1 を入れ N を 2, 3 と送りながら XA の N 番目への格納を計画しているわけである。

このプログラムで別に間違いないが、一般に計算機は XA[I] などと書いてその場所をつきとめさせられると余分の時間を要するので、次のように直した方がより良いプログラムになる。

```

begin                          ①
real SX, SXX, MX, RV, AE;      ②
array XA[1:2500];              ③
integer I, N;                  ④
procedure PR;                   ⑤
begin 1.3 ⑤~⑥を書く end;      ⑥
N:=1;                          ⑦
START: READ(AE);               ⑧
if AE ≥ 0.70 then go to NEXT;  ⑨
XA[N]:=AE; N:=N+1; go to START; ⑩
NEXT: SX:=SXX:=0;              ⑪
for I:=1 step 1 until N-1 do    ⑫
begin                          ⑬
    AE:=XA[I];                 ⑭
    SX:=SX+AE;                  ⑮
    SXX:=SXX+AE↑2;              ⑯
end;                           ⑰
PRINTS('N='); PRINTI(4, N-1);  ⑱
CRLF(1);                       ⑲
PR;                             ⑲
end;                           ㉑

```

今度は②で AE という変数を宣言して置いて⑧⑩⑯および⑭⑮⑯で使用している。

序論の部分で用いている例としてはこれで十分であるが、もしサイズ N の情報とデータ格納場所 XA および平均、不偏分散の格納場所 MX, RV を指定した手続きに直しておくと、他の人がこの手続きを用いてその計算の一部として平均や不偏分散の平方根を利用したより大きなプログラムを作る助けになる。そのために後の手続きの項への予備プログラムとして次のものを用意して置く。データは N を先頭に N 個の X_i が並んでいるものとする。(これは始めての時は飛ばして先へ進んでもよい。)

```

begin                          ①
real MX, RV;                   ②
array XA[1:2500];              ③
integer I, N;                  ④
procedure MXRV;                 ⑤
begin                          ⑥
    real SX, SXX; integer I;    ⑦
    SX:=SXX:=0;                 ⑧
    for I:=1 step 1 until N do  ⑨
begin                          ⑩
    real AE;                     ⑪
    AE:=XA[I];                  ⑫
    SX:=SX+AE;                   ⑬
    SXX:=SXX+AE↑2;               ⑭
end;                             ⑮
    MX:=SX/N;                   ⑯
    RV:=sqrt((SXX-MX*SX)/(N-1)); ⑰
end;                             ⑱
READ(N);                        ㉒
for I:=1 step 1 until N do      ㉓
    READ(XA[I]);                 ㉔
MXRV;                           ㉕
PRINTR(4, MX); PRINTR(4, RV);  ㉖
CRLF(1);                        ㉗
end;                             ㉘

```

2.4 普通の言語と ALGOL

普通の言語

文章 writings

一連のまとまった事柄を伝達する為に、文字・数字・記号等で書きあらわされたものを文章という。

文 sentence

文章の構成単位であり、とにかく何かの事柄を表現している。一般に主部および述部より成る。

定義 definition

新しい用語を使用するときは、誤解を避ける為に、その用語の使用に先立って、その内容を規定する定義が与えられることが多い。

句 phrase

文は句から構成されており、名詞句・形容詞句・副詞句に分れる。

節 clause

主部および述部を含むが、独立した文ではないものに節があり、名詞節・形容詞節・副詞節に分れる。条件をのべるものなどは副詞節に属する。

文字 letter

英語の場合は、大文字および小文字のアルファベット、日本語の場合は漢字・ひらがな・カタカナ等より成る。

数字 digit

0より9までの10進数字

ピリオド period

文の終りにはピリオドを打つ。

記号

表現を簡単にする為に、 $+-\times\div/>\geq=\leq<\wedge\vee\neg$
 $\cap\equiv()' , []$: などの記号を使用する。

文をくくる括弧

別になし。

表現方法

$$1) \frac{A+B}{C+D}$$

$$2) A_{i,j} \text{ (マトリクス } A \text{ の } i \text{ 行 } j \text{ 列要素)}$$

$$3) Q^k$$

4) もしも $P=Q$ ならば A 、そうでなければ B という結果と $C+D$ の積の演算結果を S という名前の場所に代入せよ。

5) もし 4×10^{-2} よりも A が大きいならば ST という名れの所に飛越せ。

6) A について 0.5 より 0.25 きざみで 2.0 まで、 B について -4.0 より 5 きざみで 3.5 まで次の式についてくり返し C の値を有効5けたで印刷せよ。

ALGOL

プログラム program

一連のまとまった事柄を伝達する為に、文字・数字・記号等で書きあらわされたものをプログラムと言う。

文 statement

プログラムの構成単位であり、とにかく何かの事柄を表現している。主部を欠く命令形に限る。

宣言 declaration

すべての名前は混同を避ける為に、文に先立つ宣言の部分でその型・内容等を規定する宣言が必要である。

式 expression

文は式から構成されており、算術式・論理式・行先式に分れる。

節 clause

もしも節と繰り返し節がある。

文字 letter

大文字および小文字のアルファベットより成る。場合により小文字を認めないこともある。

数字 digit

0より9までの10進数字

セミコロン semi-colon

文と文の区切りにはセミコロンを入れる。

記号

表現を正確に行なうために、 $+-\times\div/>\geq=\leq<\wedge\vee\neg$
 $\cap\equiv()' , []$; のほか $:=$ という特殊記号を用いる。

文をくくる括弧

begin と *end* があり 1つの文にまとめることができる。プログラム自身も *begin* と *end* でくくられた1つの文でなければならない。

表現方法

$$1) (A+B)/(C+D)$$

$$2) A[I, J]$$

$$3) Q \uparrow K$$

$$4) S := (\text{if } P=Q \text{ then } A \text{ else } B) * (C+D);$$

$$5) \text{if } A > 4.10^{-2} \text{ then go to } ST;$$

$$6) \text{for } A := 0.5 \text{ step } 0.25 \text{ until } 2.0 \text{ do} \\ \text{for } B := -4.0 \text{ step } 5 \text{ until } 3.5 \text{ do}$$

$$C = A^2 + \sqrt{B}$$

(ただしBは絶対値をとるものとする)

3. ALGOL の基本原則

ALGOLの文法解説に入る前に基本原則をあげてみると次の12項となる。

1. ALGOLのプログラムは基本的には文の集積であり、これらの文は *else* の直前を除いて、すべてセミコロン (;) で区切られる。
2. 文は *begin* と *end* (文をくくる括弧) によってくつて、一つの複合文 (compound statement) にまとめることができる。
3. ALGOLプログラム自身も *begin* と *end* でくつてなければならない。
4. 文の中で使われるすべての、変数・スイッチ・手続き・関数呼び出しの名前はすべて、その名前が使われる範囲の一番最初の *begin* の直後の宣言部で宣言されているなければならない。このようにして宣言された名前は、その *begin* に対応する *end* までの間についてのみ意味を持ち、このことを名前はその宣言されたブロックに関し局所的であるという。

もちろん一般に側のブロックに関しては、その外側で宣言された名前は、内側で同じ名前が宣言されない限り通用し、同じ名前が内側で宣言されたときは、内側のブロック内では内側で宣言された意味のほうが優先する。

またすべての名前 (名札の名前を含む) は、同じ宣言部で2つ以上の意味に対して用いてはならない。

5. 文は飛び越し文によって行先を変更された場合を除き、原則として書かれた順に実行される。
6. 普通の文の句に当るもの式 (expression) がある。すべての変数は、値を代入または読み込みしてからでなければ式の中で使用することはできない。
7. データテープまたはカード上一つのデータを読み込み終われば次のデータが読み込み位置に待機する。
8. 下つき添字、上つき乗べき、水平分割線による割算表示は、タイプライタによる表現が困難なので禁止し、代わりに次の表現を用いる。

$$\begin{array}{l} A_3 P Q k, l \text{ は } A[3] \quad P Q(K, L) \\ A^3 C^{2+2} \text{ は } A \uparrow B \quad C \uparrow (B+2) \\ \frac{A+B}{C+D} \text{ は } \langle A+B \rangle / \langle C+D \rangle \end{array}$$

ALGOL 60 では大文字小文字ともに名前として使用

```
begin
  C := A ↑ 2 + sqrt (abs (B));
  PRINTR (5, C);
  CRLF (1);
end;
```

(ただし PRINT 関係は計算機によって異なる)

してよいが、機械によって全能的にあるいはアルゴルの単語以外には小文字の使用を禁止しているものがあるので、本稿では名前には大文字を使用しておく。

9. 値には算術値と論理値があり、算術値は *real* (実数) と *integer* (整数) に分れ、論理値は *Boolean* といわれるもので、その値は *true* と *false* の2つしかない。
 $real$ は $3.6 - 2 \times 10^{-3} \quad 2.8 \times 10^2$
 など小数点か10のべきを含むが例の中の後の2つは $-2_{10} - 3.28_{10} 2$ と小さい 1_0 を用いて表現する。
10. 変数に対する値の代入はコロンイコール (:=) という記号によって行なうが、新しい値が代入されると古い値は失われる。
11. *begin* の直後または; の直後に *comment* と書けば、次の; の間までは何を書いても機械は無視する。また *end* の後は次の; *end else* までの間何を書いても機械は無視するので注記に利用できる。また改行空白等は一切無視する。
12. 名前としてはアルファベットを先頭としその後にはアルファベットまたは数字を必要によってつけたものとする (機械によっては小文字を禁止するものもありまた一般に先頭の5~8文字のみを区別する)。

したがって *A P A 3 L 3 9 B M A T R I X*

などはすべて名前として使用してよいが、数字が先頭にきた3Bは使用できない。名札としては符号なし整数を認める (機械によってはこれを禁止するものもある)。

以下の説明の都合上次の略号を使用する。

- $\overline{AE}$ 算術式 (arithmetic expression)
 結果として1つの算術値に帰着する表現。
- $\overline{AP}$ 実パラメタ (actual parameter)
 手続き文に用いるパラメタで手続きの宣言における *FP* に対応する。
- $\overline{BE}$ 論理式 (Boolean expression)
 結果として1つの論理値に帰着する表現。
- $\overline{D}$ 宣言 (declaration)
 名前の型・内容等をその使用に先き立てて示す部分。
- $\overline{DE}$ 行先式 (designational expression)
 結果として1つの名札に帰着する表現。
- $\overline{FP}$ 仮パラメタ (formal parameter)
 手続きの宣言に用いるパラメタ。

$\tilde{FS}$ 繰り返し文 (for statement)

くり返しを指示する文。

$\tilde{I}$ 名前 (identifier)

$\tilde{S}$ 文 (statement)

計算機に1つの操作を実行させる単位。

$\tilde{US}$ 無条件文 (unconditional statment)

文の中で条件文。 $\tilde{FS}$ 以外のもの。

$\tilde{V}$ 変数 (variable)

単純変数と添字付き変数に分れる。

4. 宣言 (declaration)

すべての変数・手続き・スイッチ等につけられた名前は
その使われる範囲の一番先頭の *begin* につづく宣言部にお
いて宣言する必要がある (ただし、名札宣言はその名札が
おかれた文の直前に:を介しておかれることによって宣言
されたものとみなされる。名札宣言は同じ文に対して何重
におかれてもよい。たとえば $L13:START;SA;S:=P+1;$
など)。

単純変数宣言: *real, integer, Boolean* の宣言詞を先頭
にして名前をカンマでつなく、同種類のものが終わればセ
ミコロンで切る。

(例) *real A, PQ14, MAX;*

integer IJ, R3;

Boolean SL;

添字付き変数宣言: *real array* (単に *array* でもよい)、
integer array, Boolean array という宣言詞の後に添字の動
く範囲を [] の中に書いて行なう。同種のものはカンマ
でつなぎ、さらに同じ添字の限界のものは名前だけをカン
マでつないでセミコロンで切る。

(例) 実数のベクトル A, B の限界が1から20まで、
整数マトリクス RB の行が1から10まで、列が-1から
3まで動くなら、

array A, B [1:20];

integer array RB [1:10, -1:3];

ただし添字の限界は下限≤上限でなければならない。

手続きの宣言: *procedure* という宣言詞の後に手続き名
を書く。その後必要に応じて $\tilde{FP}$ 部を括弧につづきセ
ミコロンで切る。 $\tilde{FP}$ 部があるときはその後規制部などを
置いてから手続きの本体を書く。詳細は後に例で示す。

関数宣言: *real procedure, integer procedure, Boolean
procedure* という宣言詞の後は手続きの宣言と同じで違
うのは関数呼び出しのほうは一つ関数でしかも答を自分自
身の名前に代入する点である。詳細は後に例で示す。

スイッチ宣言: *switch* という宣言詞の後に名前を書き
: $\sim$ の後にスイッチリストをカンマでつないで書きセミ
コロンで切る。詳細は後に例で示す。

5. 式 (expression)

文を構成している部分であって、普通の文の句 (phrase)
に当るの式である。

式には $\tilde{AE}, \tilde{BE}, \tilde{DE}$ の3種がある。

1) $\tilde{AE}$ (算術式)

$\tilde{AE}$ には単純算術式と条件付算術式とがある。

結果として1つの算術定数に帰着する表現である。

1) 単純算術式

1. 算術定数 $-3.5 \ 4.5_{10} -2 \ 10^{-2} \ (real)$
 $4 \ -8601 \ (integer)$
2. 算術変数 $AB \ R4 [1, S]$
 $S41 [if \ P=Q \ then \ 3 \ else \ 4]$

3. 算術関数呼び出し $TAN(A) \ MAX(P, Q)$

標準関数 $\sin \cos \arctan \ln \exp \sign$

$abs \ sqrt$

換算関数 $entier$

を含む、*sign*、*entier*は答が *integer*、*abs*はその後の
括弧の中の種類に従い、他はすべて *real* となる。*ln*
は自然対数、*exp* は指数関数、*sign* (シグナムと読
む) は符号で *sign(A)* は $A > 0$ なら +1、 $A = 0$ なら
0、 $A < 0$ なら -1 となる。*abs* は絶対値、*sqrt*
は平方根。*entier* はアンティエを読み、括弧の中の
ものを越さぬ最大の整数をとる。たとえば *entier*
(8.6) は8であるが *entier* (-8.6) は-9である。

4. 条件付算術式を丸括弧でつづつたもの。

5. 1~4のものの前に+や-をつけたもの「すでに
+や-のついているものは括弧につづむ」。

6. 1~5を+・-・/・÷という四則演算子でつない
だもの。

・は×の代りに用いられることが多い (大文字のエッ
スと混同しないため)。

÷は整数割算に用い $5 \div 2$ は2である。

↑は乗べきで 3^2 のことを $3 \uparrow 2$ と書く。

2) 条件付算術式

単純算術式を $\tilde{SA}$ と書けば条件付算術式は

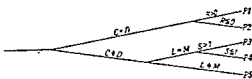
$if \ \tilde{BE} \ then \ \tilde{SA} \ else \ \tilde{AE}$

である。これは *ALGOL* の特長の1つをなすものであつて、
BE の結果が *true* なら $\tilde{SA}$ を、*false* なら $\tilde{AE}$ をとる。

(例)

$if \ P > B \ then \ (if \ P \neq Q \ then \ 3 \ else \ 3+S) \ else \ 3+S$
(*then* の後に条件の *if* が括弧でつづつて禁止している
ので括弧でつづんである)。

$if \ R \geq S \wedge C = D \ then \ A [if \ P1 = P2 \ then \ 2 \ else \ 3,$
 $N] + 4.5 \ else \ W + (if \ L = M \ then \ 3.6 \ else \ 4.8) * 2$



上図のような分岐条件の値をとりたいときは次のような条件付算術式となる。

if $C=D$ then (if $R>0$ then P_1 else P_2) else if $L=M$ then (if $S>1$ then P_3 else P_4) else P_5
ii) $\tilde{B}\tilde{E}$ (論理式)

論理式には単純論理式と条件付論理式がある。

結果として1つの論理定数に帰着する表現である。

1) 単純論理式

1. 論理定数 true および false (つまり真か偽かという値)
2. 論理変数 $PQ A_2 [R, 4]$
 S (if $A=B$ then R else 1, 2)
3. 論理関数呼び出し $BLANCH(P, Q)$
4. 比較式 $\tilde{A}\tilde{E}_1 P \tilde{A}\tilde{E}_2$
ただし P とは $\geq, \leq, <, >$ の6種である。
5. 条件付論理式を括弧でつづんだもの
6. $1 \sim 5$ のものの前に $\neg$ (not) をつけたもの
7. $1 \sim 5$ を $\wedge$ (and), $\vee$ (or), $\supset$ (imply), $\equiv$ (equivalent) でつないだり括弧でつづんだりしたもの。

2) 条件付論理式

単純論理式を $\tilde{S}\tilde{B}$ と書けば条件付論理式は

if $\tilde{B}\tilde{E}_1$ then $\tilde{S}\tilde{B}$ else $\tilde{B}\tilde{E}_2$

でこれも ALGOL の一つの特長をなすものである。

(例)

if $A=B \wedge C=D$ then (if $\neg P=Q$ then $R=S$ else $K=P$) else $L>1$

if if $A=B$ then $A_1=B_1$ else $A_2=B_2$ then $R=S$
else $R_1=S_1$ then $R_2=S_2$ else $LR \wedge LS$

最後の例は --- の部分は --- に対して if と then には含まれた論理式となり --- の部分はまた残の全体に対し if と then で含まれた論理式となっている。

iii) $\tilde{D}\tilde{E}$ (行先式)

行先式は単純行先式と条件付行先式に分れる。
結果として1つの名札に帰着する表現である。

1) 単純行先式

1. 名札 $L_1 START$
2. スイッチ $S\{3\}$
 S (if $P=Q$ then 1 else 2)
3. 条件付行先式を括弧でつづんだもの
4. $1 \sim 3$ を括弧でつづんだもの

2) 条件付行先式

単純行先式を $\tilde{S}\tilde{D}$ と書けば条件付行先式は、

if $\tilde{B}\tilde{E}$ then $\tilde{S}\tilde{D}$ else $\tilde{D}\tilde{E}$

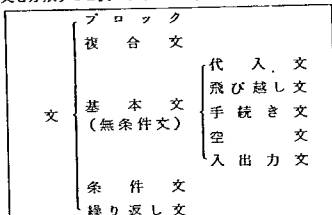
で、これも ALGOL の特長の1つである。

(例)

if $P=Q$ then (if $R=S$ then L_2 else L_3) else $S(I)$

6. 文 (statement)

文を分類すると次のようになる。



1) 基本文

i) ブロック

begin の後に宣言部のあるもので、次の形をとる。

begin $\tilde{D}; \dots; \tilde{D}; \tilde{S}; \dots; \tilde{S}; \text{end};$

end の直前では、はなくてもよいが、が落ちることによる他の誤りを防ぐため end の前には常に空文があるものと考えて、を入れることにしたほうがよい。end の後も常に、を入れるようにしておくほうが誤りが少ない。

2) 複合文

文を begin と end でかこんだもので、次の形をとる。

begin $\tilde{S}; \dots; \tilde{S}; \text{end};$

ブロックとの違いは $\tilde{D}$ の有無だけである。; は $\tilde{D}$ や $\tilde{S}$ の区切りに用い、end の直前では不用であるが end end と続いたときに end の後で不用であるかのように誤解する人が多いので、常に空文が end の前にあると考え、最後の $\tilde{S}$ と空文の区切り、を end の前にも入れておくように習慣づけたほうがよい。

3) 代入文

右辺の式の結果を左辺の変数に代入する文で次の形をとる。左辺の変数のアドレスは左から順に決定し(添字つきの場合はその添字の値はその文が実行される直前の値をとる)その後右辺の式の値を決定し代入される。

$\tilde{V} := \tilde{A}\tilde{E};$
 $\tilde{V} := \tilde{B}\tilde{E};$

$\tilde{A}\tilde{E}$ の結果を代入する予定の左辺の変数は real または integer として宣言されたものに限り、左辺の型に従って自動的に変換される。特に右辺が real で左辺が integer のときは4捨5入される。

$\tilde{B}\tilde{E}$ の結果を代入する予定の左辺の変数は Boolean として宣言されたものに限る。

いずれの場合も、左辺が real, integer, Boolean 同志であ

るときは

$\bar{P}_1 := \bar{P}_2 := \dots :=$

と何重に重ねてもよい。

(例)

$A := B + C;$

$RS[I] := 4 + X;$

$L1 := P := AN[1, 4] := 0;$

$KN := true;$

4) 飛び越し文

一般に文は書かれた順序に従って実行されるが、行先を変更したいときにはこの飛び越し文によって次に実行すべき文に対してつけられた名れを行先式によって指示する。

$go\ to\ \bar{D}\bar{E};$

(例)

$go\ to\ L1;$

$go\ to\ if\ P=Q\ then\ R1\ else\ R2;$

$go\ to\ if\ A>0\ then\ S[I]\ else$

$if\ B=0\ then\ PQ1\ else\ P2;$

5) 手続き文

すでに宣言されている手続き宣言で示された名前を持ち出して、これが $\bar{P}\bar{P}$ つきのものであればこの $\bar{P}\bar{P}$ に対応する $\bar{A}\bar{P}$ を指定して手続きの宣言で指示された内容を実行することを命ずるもので、後に実例によって詳細に説明する。

6) 空白文

空白の文であって何もしないで次の文に移る。これは終わりの end に行きたいときに end の直前に名れをつけるのに役立つ。使用の実例は(1)にあげた。

7) 建設者共通入出力文

ALGOL-60 には入出力に関するなんらの規定もないので次のものを建設者共通入出力文とする。

$READ(A, B, \dots),$

データテープ (あるいはカード) より、順に情報を読み込み、それぞれ $A, B, \dots$ に格納せよ。

$PRINTR(N, A);$

式 A の値を $real$ 型の有効 N けたの情報として印刷準備せよ。一般に次の形と約束する。

$$\underbrace{\square \pm x \times \dots \times x}_{N} \pm \underbrace{x \times \dots \times x}_{2}$$

$_{10}$ の後は指数部であり、 $PRINTR(5, S);$ で S が -1965.15 なら $\square \dots \square 1965_{10} + 04 \square$ となる。

$PRINTI(N, A);$

式 A の値を $integer$ 型の有効 N けたの情報として印刷準備せよ。一般に次の形と約束する。

$$\underbrace{\square \pm x \times \dots \times x}_N$$

$PRINTF(n_1, n_2, A);$

式 A の値を $real$ 型で小数点以下 n_1 けた、以下 n_2 けたの形で印刷準備せよ。一般に次の形と約束する。

$\square \pm x \times \dots \times x \cdot \underbrace{x \times \dots \times x}_{n_2} \square$

n_1

n_2

たとえば $PRINTF(23, P);$ で P が 3639 なら

$\square \square 36390 \square$

という印刷準備が行なわれる。 $\square$ はスペースの意である。

$CRLF(N);$

今までの情報を印刷し N 行おくれ、 $CRLF$ は carriage return line feed の略でこの指令がくるまでは情報は次々と同じ行の右側につめ込まれ印刷準備される。

$SPACE(N);$

N 字分の空白を印刷準備せよ。

$PRINTS('PQR=');$

$PQR=$ という字を印刷準備せよ。

' ' の中に書きたいものを書けばそのとおり印刷する準備が行なわれる。もし、空白を打ちたければ $SPACE$ を用いてもよいが ' ' の中で $\bar{w}$ をならべて書いてもよい。

$READARRAY(A, 1, M, 1, N, \dots);$

データテープ (あるいはカード) 上より、行数 1 より M まで、列数 1 より N まで…… (何次元でもよい) 変えて配列 A を順次行ないに読み込み格納するにはたとえばマトリクス A が

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

という形のものでデータテープ上

$a_{11} a_{12} \dots a_{1n} a_{21} a_{22} \dots a_{2n} \dots a_{m1} \dots a_{mn}$

の順に入っていれば

$READARRAY(A, 1, M, 1, N);$

という命令によって読み込まれる。

ii) 条件文

条件によって 2 つの文のうちの 1 つを選択させるための文で次の 3 種がある。

$$\begin{array}{l} a. \text{ if } \bar{B}\bar{E} \text{ then } \bar{U}\bar{S}; \bar{S}; \\ b. \text{ if } \bar{B}\bar{E} \text{ then } \bar{U}\bar{S} \text{ else } \bar{S}_1; \bar{S}_2; \\ c. \text{ if } \bar{B}\bar{E} \text{ then } \bar{F}\bar{S}; \bar{S}; \end{array}$$

a の型とは $\bar{B}\bar{E}$ が $true$ なら $\bar{U}\bar{S}$ を、 $false$ ならその次の $\bar{S}$ を実行する。

b の型では $false$ なら $\bar{S}_1$ を実行する。もちろんその次には $\bar{S}_2$ を実行することになる。

$\bar{U}\bar{S}$ は条件文・ $\bar{F}\bar{S}$ 以外の文であるが、条件文であっても $begin$ と end でつめれば $\bar{U}\bar{S}$ となる。

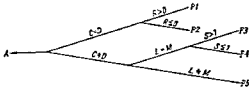
たとえば

$if\ A=B\ then\ begin\ if\ C \neq D\ then\ S := S + 1$

$else\ S := S - 1\ end\ else\ R := P * Q;$

算術式のところで例としてあげた分岐の問題の結果を A

に代入するならば



if C=D then begin if R>0 then A:=P1 else A:=
P2 end else if L=M then begin if S>1 then A
:=P3 else A:=P4 end else A:=P5 ;

これをもし A に代入する代入文に統一し右辺を条件付算
術式とするならば

A:=if C=D then (if R>0 then P1 else P2) else
if L=M then (if S>1 then P3 else P4) else
P5 ;

となり文をくくる括弧は begin end 式をくくる括弧は ()
であることがつき分る。

Ⅲ) 繰り返し文

コントロール変数にいろいろな値を与えて、同じ文を実
行させるためのくり返し文で、次の形をとる。

for $\bar{V}$:= $\bar{a}$ for list > do $\bar{S}$;
ただし < for list > としては
a. $\bar{A}\bar{E}$
b. $\bar{A}\bar{E}_1$ step $\bar{A}\bar{E}_2$ until $\bar{A}\bar{E}_3$;
c. $\bar{A}\bar{E}$ while $\bar{B}\bar{E}$
d. a ~ c をカンマでつないだもの等がある。

したがって I を 1 から 1 きざみで 10 まで、2 きざみで
50 まで、30 きざみで 3960 になるまで I と I² を印刷
させたいならば、

for I := 1 step 1 until 10, 12 step 2 until 50,
I + 30 while I ≤ 3960 do

begin
PRINTI (4, I) ;
PRINTI (9, I + 12) ;
CRLF (1) ;
end ;

となろう。

もちろん最後の部分は 80 step 30 until 3960 do とし
ても同じことである。

A を 0.1 から 0.2 きざみで 1.1 まで、B を 3.5 から 0.5
きざみで 5.5 まで変化させながら AB² を印刷させたいとき
は

for A := 0.1 step 0.2 until 1.12 do
for B := 3.5 step 0.5 until 5.6 do
begin
PRINTF (1.1, A) ;
PRINTF (1.1, B) ;

PRINTR (5, A * B + 12) ;

CRLF (1) ;

end ;

とすればよい。A の終止条件を 1.12 として 1.1 としなかつた理由は 2 進の計算機で次々と 0.1 を加えてゆく過程で 10 進のきっちりした 0.1000... に当るものを加えられないため最後のきざみを加えたとき 1.1 をわずかに越して 1.1 に対する計算を実行しないことをおそれきざみの 1/4 ~ 1/5 程度を終止条件に加え (もしきざみが負のときは引く) ておいたものである。

また A, B の値を印刷したのは、これがないと結果だけがズラリと印刷されてしまつてどの A, B に対応する結果かわからなくなることをおそれたためである。

7. ALGOL によるプログラム例

以上の知識があれば、手続きや関数呼び出し、スイッチなどを使用しない限り、たいいてい問題を取り扱うことができる。そこで二、三の例題を解いてみよう。

7.1 不偏分散の平方根の計算 (I)

一群のデータの位置の情報 $\bar{X}$ とバラツキの情報 $\sqrt{V}$ を計算したい。ただし、

$$\bar{X} = \sum X / N$$

$$\sqrt{V} = \sqrt{\frac{\sum (X - \bar{X})^2}{N - 1}} = \sqrt{\frac{\sum X^2 - (\sum X)^2 / N}{N - 1}}$$

である。

データテープは N 個の X が並んでいるが個数は不明でデータの終わりの印に最後には 10⁵⁰ というような非常に大きな数が入れてあるものとする。

また印刷様式は N = ..., MX = ..., RV = ... と一行に印刷するものとする。

```
begin
  real X, MX, RV, SX, SXX ;
  integer I ;
  I := 0 ; SX := SXX := 0 ;
  ST : READ (X) ;
  if X > 10 40 then go to OUT ;
  SX := SX + X ;
  SXX := SXX + X + X ;
  I := I + 1 ;
  go to ST ;
  OUT : MX := SX / I ;
  RV := sqrt ((SXX - SX + SX / I) / (I - 1)) ;
  PRINTS ('N = ' ;
  PRINTI (4, I) ;
  PRINTS ('MX = ' ;
  PRINTR (5, MX) ;
  PRINTS ('RV = ' ;
  PRINTR (5, RV) ;
  CRLF (1) ;
end ;
```

①の *begin* は ALGOL プログラムによって指令を行なうことを知らせている。

②の *read* は以下のプログラムにおいて、これらの文字を実数タイプの単変数として

③の *integer I*; は *I* を整数タイプの単変数として使用することを宣言し、このように命名された記憶場所を留保することを命じている。ここで宣言部を終わる④では *I* と *SX* および *SXX* に次々と和を格納する予定なので、0 を代入している。*I* だけは整数タイプなので別に代入を行なう必要があるわけである。

⑤で *S T* : とあるのはこの後の文に対して *S T* という名札をつけたことを意味し、名札は読解の便宜上いくら立てても別段害はないし、また同じ文に違う名札を重ねて立ててもよい。*READ(X)* はデータテープ上より情報を読み込んで *X* という場所に格納することを命じている。

⑥では終止条件がきたかどうかの判定で、まだなら当然次の文つまり⑦に行く。

⑦⑧⑨は和をとる操作で、⑩によって④の *S T* という名札のつけられた文に行き、次のデータが読み込まれる。このときとの *X* の値は当然新しい今読み込んだ値に書き換えられる。何回かくり返した後に 10 50 の値が読み込まれるとこれは 10 40 よりまだ大きいので⑥の *BF* が *true* となり *OUT* という名札のつけられた文つまり⑪に行くことになる。

⑪では *MX* に平均値が(この段階までで *I* には個数に関する情報で、*SX* には総和が入っているはずである)代入される。

⑫では *RV* に不偏分散の平方根が代入される。

⑬の文によって *N* → という字が、⑭の文によってその右に有効ケタの整数タイプで変数 *I* の内容が、以下同様に予定した印刷様式の指定を終わり、⑮の *CRLF(1)* によって以上の情報を 1 行に印刷して 1 行おくり、⑯の一番最初の *begin* に対応する *end* によってこのプログラムは完全に終了することを示している。

7.2 不偏分散の平方根の計算 (2)

同じ問題を、今度は個数 *N* の情報を先頭に *X₁* から *X_N* までデータテープに入れ、この中の最大値と *MX*, *RV* の値をプリントするプログラムは次のようなものになる。

```
begin
  read MX, RV, SX, SXX, MAX;
  integer N, I;
  array X[1:500];
  READ(N);
  READARRAY(X, 1, N);
  MAX := -1050; SX := SXX := 0;
  for I := 1 step 1 until N do
    begin
      SX := SX + X[I];
```

①
②
③
④
⑤
⑥
⑦
⑧
⑨
⑩

```
SXX := SXX + X[I] ↑ 2;
MAX := if MAX ≥ X[I] then
  MAX else X[I];
end;
PRINTS('MAX = '); PRINTR(5, MAX);
PRINTS('MX = '); PRINTR(5, SX/N);
PRINTS('RV = ');
PRINTR(5, sqrt((SXX - SX ↑ 2/N)/(N-1)));
CRLF(1);
end;
```

⑪
⑫
⑬
⑭
⑮
⑯
⑰
⑱
⑲
⑳

⑭は配列宣言、⑮は配列の読み込み文である。⑦で *MAX* にきわめて小さい値を入れて置き、*X[I]* がこれより大きかったら順次その値を *MAX* に入れるようにして *I* を 1 から *N* まで変えれば最後には *MAX* に最大値が残ることになる。

⑱⑲のように印刷文の中に直接式を書くことも認められている。

7.3 一元配置の分散分析

水 準	1	2	3		M
デ ー タ	x_{11}	x_{21}	x_{31}		x_{m1}
	x_{12}	x_{22}	x_{32}		x_{m2}
	$\vdots$	$\vdots$	$\vdots$		
	x_{1n_1}	x_{2n_2}	x_{3n_3}		x_{mn_m}
データサイズ	n_1	n_2	n_3		n_m

以上のデータがあるときデータテープ上より、水準数 *M*, 各水準のデータサイズ *N₁*, *N₂*, ..., *N_m* データ x_{11} , x_{12} , ..., x_{1n_1} , x_{21} , x_{22} , ..., x_{mn_m} が入っているとき、これを読み込んで各水準ごとの平均、不偏分散の平方根を印刷し、次に分散分析表

AOV TABLE				
SV	SS	DF	MS	F
A	SSA	NA	FA	FO
E	SSE	NE	FE	
O	SSO	NO		

を印刷するプログラムを作れ。

ただし $SSO = \sum_{i,j} X_{ij}^2 - \left(\sum_{i,j} X_{ij} \right)^2 / \sum N_i$
 $= \sum X_{ij}^2 - CF$
 $SSA = \sum_i \left(\sum_j X_{ij} \right)^2 / N_i - CF$
 $SSE = SSO - SSA$
 $NA = M - 1$
 $NO = \sum N_i - 1$
 $NE = NO - NA$
 $FA = SSA/NA$
 $FE = SSE/NE$
 $FO = FA/FE$

とする。また $M \leq 10$ $N_i \leq 20$ とする。

```
begin
  integer M, I, J, NO, NA, NE;
  integer array IA[1:10];
  real SSA, SSE, SSO, CF, FA, FO, FE, SO;
```

```

array XA[1:20,1:10], SX, MX, SSX,
RV[1:10];
READ(M);
READARRAY(IA,1,M);
for I:=1 step 1 until M do
  for J:=1 step 1 until IA[I] do
    READ(XA[I,J]);
  SPACE(5); PRINTS('MEAN'); SPACE(15);
  PRINTS('RV'); SPACE(15); PRINTS('N');
  CRLF(1);
for I:=1 step 1 until M do
begin
  real SS,S;
  SS:=S:=0;
  for J:=1 step 1 until IA[I] do
    begin real P; P:=XA[I,J];
      S:=S+P;
      SS:=SS+P↑2;
    end;
  MX[I]:=S/IA[I];
  SSX[I]:=SS;
  RV[I]:=sqrt((SS-S↑2/IA[I])/(IA[I]-1));
  SX[I]:=S;
  PRINTR(9,MX[I]); PRINTR(9,RV[I]);
  PRINTI(2,I);
  CRLF(1);
end;
S0:=SS0:=SSA:=0; NA:=N0:=0;
for I:=1 step 1 until M do
begin
  SS0:=SS0+SSX[I];
  SSA:=SSA+SX[I]↑2/IA[I];
  N0:=N0+IA[I];
  S0:=S0+SX[I];
end;
N0:=N0-1; NA:=M-1;
NE:=N0-NA; CF:=S0↑2/(N0+1);
SS0:=SS0-CF; SSA:=SSA-CF;
SSE:=SSE-SSA; FA:=SSA/NA;
FE:=SSE/NE; F0:=FA/FE;
PRINTS(┌┌┌ A0V ─┐ TABLE );
CRLF(1);
SPACE(5); PRINTS('SV'); SPACE(14);
PRINTS('DF'); SPACE(14); PRINTS('MS');
SPACE(15); PRINTS('F');
CRLF(1);
PRINTS('A'); PRINTR(9,SSA);
PRINTI(13,NA); PRINTR(9,FA);

```

```

PRINTR(9,F0);
CRLF(1);
PRINTS('E'); PRINTR(9,SSE);
PRINTI(13,NE); PRINTR(9,FE);
CRLF(1);
PRINTS('O'); PRINTR(9,SS0);
PRINTI(13,N0);
CRLF(1);
end;

```

8. ALGOL によるプログラミングの簡単な

例題 (1)

以下に手続きなどを使わないで今までの知識だけで解く簡単な例題をあげる。解答は次号にのせるので試みておいていただきたい。

8.1 相関係数

データテープとして先頭に組数 N の後に X_1 から X_N まで続いて Y_1 から Y_N までのデータが入っている。これを読み込んで相関係数

$$R = \frac{S(XY)}{\sqrt{S(SX) \cdot S(SY)}} = \frac{\sum (X - \bar{X})(Y - \bar{Y})}{\sqrt{(\sum X - N\bar{X})^2 \cdot \sum (Y - \bar{Y})^2}}$$

$$= \frac{\sum XY - (\sum X)(\sum Y)/N}{\sqrt{(\sum X^2 - (\sum X)^2/N) \cdot (\sum Y^2 - (\sum Y)^2/N)}}$$

を計算し、 $R = \dots$ という形で印刷せよ。

8.2 数表

$N \ N^2 \ N^3 \ 1/N \ N \ 1/N$ の表を 1 から 1 きざみで 10000 まで作れ、ただし表頭に $N \ NN \ NNN \ 1/N \ FACTN$ を打ち以下 50 行打つと 10 行とばして打つようにせよ (ヒント I が 50 で割り切れるかどうかには $\div$ が利用できる)。

8.3 電気料金計算

電気料金は基本料金 10 アンペア 175 円, 15 アンペア 263 円, 20 アンペア 350 円, 30 アンペア 525 円で従量料金 108 円/kWh, 電気ガス税 7 円である。テープより番号 I と契約電力 A , 使用電力量 kWh を読んでこの順に印刷し、その右に料金を印刷せよ。ただしデータがつきた記号として 0, 0, 0 を入れておくものとする。

8.4 2 次方程式の根

データテープより 2 次方程式 $AX^2 + BX + C = 0$ の係数 A, B, C が何組か入っている。これを読込んで組番号および A, B, C を印刷しその右に実根のときは X_1 (+ の項) X_2 (- の項), 虚根のときは実部, 虚部の順に印刷せよ。

ただしデータの最後の印しとして 0, 0, 0 が入っているものとする。

9. 例題解 [1]

前号に練習問題をあげておいたが今後の素材に利用するため答をかかげおく、正しい動作を行なわせるための方法はいくつも考えられ、分りやすいこと、計算時間が短かくてすむことなどの観点や、後で一般化して使用する容易さの点で検討するべきであろう。

9.1 相関係数

データテーブル

$N, X_1, X_2, \dots, X_N, Y_1, \dots, Y_N$

ただし $N \leq 500$

$$\text{相関係数 } R = \frac{S(XY)}{\sqrt{S(XX) \cdot S(YY)}}$$

$$S(XY) = \sum XY - (\sum X)(\sum Y)/N$$

印字は $R =$ と打った右に答を小数点以上1けた、以下4けたの形で印刷する。

begin

```
integer N, I;
real SX, SY, SXX, SYX, SXY, R;
array X, Y(1:500);
READ(N);
READARRAY(X, 1, N);
READARRAY(Y, 1, N);
SX := SY := SXX := SXY := SYX := 0;
```

for I:=1 step 1 until N do

begin

```
real P, Q;
P := X(I);
Q := Y(I);
SX := SX + P;
SY := SY + Q;
SXX := SXX + P*P;
SYX := SYX + P*Q;
SXY := SXY + Q*Q;
```

end;

SXX := SXX - SX*2/N;

SYX := SYX - SY*2/N;

SXY := SXY - SX*SY/N;

PRINTS(R*2);

R := SXY/sqrt(SXX*SXY);

PRINTF(1.4R);

CRLF(1);

end;

繰り返し文の中で local に P, Q を宣言し、これに X(I), Y(I) を受けたのは array の要素をさがすのに割合手間をくうので、何回も用いるときはこのようにいったんある変

数に受けてから使用するほうが演算時間が少なくてすむからである。

9.2 数表 (解)

$N^2 \cdot N^3 \cdot 1/N \cdot N!$ の表を 1(1) 10000 作り、表頭に $N \cdot N \cdot N \cdot N \cdot 1/N \cdot FACTN$ を打ち 50 行ごとに 10 行とばして打つ問題である。

begin

integer I;

real FACT;

FACT := 1;

PRINTS('#####NN#####

#####1/N#####

#####FACTN');

CRLF(2);

for I:=1 step 1 until 10000 do

begin

if (I-1)÷50×50=I-1 then CRLF(10);

PRINTI(5, I);

PRINTI(9, I*2);

PRINTR(9, I*3);

PRINTR(9, I/1);

FACT := FACT * I;

PRINTR(9, FACT);

CRLF(1);

end;

end;

階乗の初期値は 1 にしておかないと累乗できない。また 51 行になったら、印刷前に 10 行おくるためには $I-1$ が 50 で割り切れるかどうか、つまり $(I-1) \div 50$ は 1 が 51 なら 1 となり 50 倍したものは $I-1$ に等しいのでここで 10 行おくることになる。

9.3 電気料金計算

データテーブル

1, 10, 851, 2, 10, 885, 5,

20, 1432, ..., 0, 0, 0,

番号 (10000 以内), 契約アンペア数, 使用電力量の順に並べてあり、最後に 0, 0, 0 が入っている。

印刷は 1 行に番号, アンペア数, 電力量, 料金を印刷するものとする。

begin

integer I;

real B, A, kWh, FEE;

A1: READ(I, A, kWh);

if I=0 then go to END;

```

B:=if A=10 then 175 else
    if A=15 then 263 else
        if A=20 then 350 else
            525;

```

```

FEE:=(B+NVH*108)*1.07;

```

```

PRINTI(4,I);

```

```

PRINTI(2,A);

```

```

PRINTF(2,I,NVH);

```

```

PRINTF(4,2,FEE);

```

```

CRLF(1); go to A1;

```

```

END:

```

end;

基本料金Bの計算にはAによる条件式を用いてみた。Iが0のときはENDという名札の所にゆくが、ENDという名札は空文に対してつけられており、何もしないで次へつまりendにゆくことになる。

9.4 2次方程式の根 (解)

データテーブル

$$A_1, B_1, C_1, A_2, B_2, C_2, \dots, 0, 0, 0,$$

$$\left(\frac{-B}{2A}\right) \pm \sqrt{\left(\frac{-B}{2A}\right)^2 - \left(\frac{C}{A}\right)}$$

で $A \neq 0$ とする。また虚根のときは (実部) + (虚部)

* I と印刷する。

begin

```

read A,B,C,RE,IM,S;

```

```

integer I;

```

```

Boolean CR;

```

```

I:=0;

```

```

ST: READ(A,B,C);

```

```

if A=0 ^ B=0 ^ C=0 then go to OVERI;

```

```

RE:=-B/(2*A);

```

```

S:=RE^2-C/A;

```

```

CR:=S<0;

```

```

IM:=sqrt(abs(S));

```

```

PRINTR(5,A);

```

```

PRINTR(5,B);

```

```

PRINTR(5,C);

```

```

if ~CR then PRINTR(5,RE+IM)

```

else

begin

```

PRINTS(' ');

```

```

PRINTR(5,RE);

```

```

PRINTS(' ') + '+';

```

end;

```

if ~CR then PRINTR(5,RE-IM)

```

else

begin

```

PRINTS(' ');

```

```

PRINTR(5,IM);

```

```

PRINTS(' ') + 'I';

```

end;

```

CRLF(1);

```

```

go to ST;

```

```

OVERI:

```

end;

Boolean の C R には判別式 $B^2 - 4AC$ が負であれば true, 負でなければ false が代入される。したがって if $\neg CR$ then …… という文で判別式が負でなければ then 以下が、負なら虚根であるから else 以下が実行される。

10. 手続 (procedure)

10.1 概要

(8) i) であげた相関係数の計算を、個数 N を先頭にした X および Y の配列が何組か入っているデータテーブルから読みとっては計算することになり、しかも今後他人が相関係数を計算したいときに、組数 N と配列 X および Y の値を与えられて答を R に格納するような一連の手続きがくり返し使用されそうときは、この相関係数を計算するプログラム全体に適当な名前をつけ、受渡しに使うパラメタとして個数および 2 つの配列の名前ならびに答の相関係数を格納する場所を指定したものを作っておくと大変便利である。

このようにした手続きの進め方を教えたテーブルを作っておけば 2 度目にこれを使用する人はこのテーブルを自分のプログラムの宣言の部分につなぎ込むだけで本文の中でこの手続きの名前と実際に使用するパラメタの値を指定してあらかじめ宣言した相関係数計算のいつさいの手続きを実行させることが可能である。

さらに計算機によってはこの手続きの内容を機械語に翻訳したものを計算機に覚えさせておいて、後の人はその名前だけを external 宣言しておいて本文の中で名前と実パラメタを指定することによって計算を実行させることすら可能である。

手続き宣言のやり方は procedure という宣言詞を用いて次の形をとる。

```

procedure 名前 (FP, FP, …… FP);
    <value part>
    <specification part> } heading 部
    <procedure body>;

```

パラメタのまったくない場合はパラメタ部のカッコをおよ

び heading 部は不用であり、 $\tilde{F}P$ に対し値でもらうてく
ることを指定しなければ value part も不用である。
specification part は規制部といわれ、計算機に $\tilde{F}P$
がどんなものであるかを教える部分で宣言とは異なる。ま
た大部分の計算機では $\tilde{F}P$ のすべては規制しておかなけれ
ばならない。

$\tilde{F}P$ はこの procedure に対して局所的であって、外の
プログラムとはいっさい無関係でありこの procedure の
中かぎりのものである。このようにしておくことによつて
この procedure はだれのプログラムの中でも使用可能
となるわけである。

10.2 手続きの実例

とりあえず今の相関係数の例を用いて procedure の作
り方と使い方を示そう。データは個数および 2 つの配列の
順で並んでおり最後に 0 が入っているものとする。

```

begin
  integer M;
  real K;
  array P, Q(1:500);
  procedure SOKAN(N, X, Y, R);
  value N;
  integer N; array X, Y; real R;
  begin
    integer I;
    real SX, SY, SXX, SiY, SXY;
    SX := SY := SXX := SiY := SXY := 0;
    for I := 1 step 1 until N do
      begin
        real P, Q;
        P := X(I);
        Q := Y(I);
        SX := SX + P;
        SiY := SiY + Q;
        SXX := SXX + P↑2;
        SiY := SiY + Q↑2;
        SXY := SXY + P * Q;
      end;
    R := (SXY - SX * SiY / N) / sqrt((SXX -
      SX↑2 / N) * (SiY↑2 / N));
  end;
ST: READ(M);
  if M = 0 then go to STOP;
  READARRAY(P, 1, M);
  READARRAY(Q, 1, M);
  SOKAN(M, P, Q, K);
  PRINTF(1, K);
  CRLF(1);

```

- go to ST; ①
STOP; ②
end; ③
⑤ SOKAN という名の手続きは個数、2 つの配列および相
関係数の名前を受渡しパラメタとする手続きであつて、仮
りに N, X, Y, R という名がつけられてある。
⑥ N は後で使用するときその実行時の対応するパラメタの
持っている値でおきかえられ、value 指定していないパラ
メタは、実行時の対応するパラメタの名前で置きかえられ
ることを示している。
⑦ array の規制は宣言と違つて名前だけ書けばよい。
⑧ ~⑨ は SOKAN という名前の手続きの本体である。
⑨ で I を local に宣言してある。

SX もこの中限りで使いすてにするものであるから
local にしてある。
⑤ で終止条件がきたら⑥にジャンプさせるつもりである
STOP はもちろん単なる名札でストップという字の内容和
はまったく無関係である。
⑥ で SOKAN という手続きが呼び出され、実パラメタの
M, P, Q, K がそれぞれ⑤の仮パラメタ N, X, Y, R と対応
している。

10.3 value 指定と規制

宣言中仮パラメタ部は ($\tilde{F}P$, $\tilde{F}P$) の形をとるが、
ない場合はこの部分を省略する。値の部は $\tilde{F}P$ 中値を外か
ら値で貰ってくるものについて指定を行なう部分であつて、

value 名前, ... , 名前;

という形をとる。

このようにしたほうが繰り返し現われる変数については
演算時間が早くなる。しかし value 指定しなくても差し
支えはないわけで、このときは値の部は省略される。
value とはその実パラメタとなっている変数の値を変更
しないという意志表示である。(配列の value 指定を禁
止するものもある)。

規制部は $\tilde{F}P$ にあげられたパラメタがどんなものである
かを規制する部分である(名前で呼ばれるものはすべて規
制不用のもの、value 指定されなかつたパラメタのうち
整数型のみ規制を必要とするもの、value 指定されたも
のを含んですべてのパラメタの規制を必要とするものなど
機種によっていろいろある)。

規制詞としては

string, label, real, integer, Boolean, array,
real array, integer array, Boolean array,
switch, procedure, real procedure, integer
procedure, Boolean procedure がある。
1) パラメタとしての procedure
procedure のパラメタとして procedure と規制され

たものを含むとは次のような場合のことである。

$\sum x_i^2$ とか $\sum x_i^3$ とかの tangent をとりたい場合、この関数形を *procedure* としておいてそのどれを指定するかを実行時に決めることができれば tangent をとる *procedure* の利用価値が大いにふえるものとしよう。そのときは次のようなものを作ればよい。ただしデータは N 、 M を先頭に $X_1, \dots, X_M, Y_1, \dots, Y_M$ に入っているものとする。

```
begin
  array X,Y(1:200);
  integer N,M;
  real TANSX2,TANSY;
  procedure SUM1(J,ARRAY,SUM);
    value I;
    integer I;array ARRAY;real SUM;
    begin
      integer I;
      SUM:=0;
      for I:=1 step 1 until L do
        SUM:=SUM+ARRAY(I);
      end;
  procedure SUM2(J,NAME,SU);
    value J;
    integer J;array NAME;real SU;
    begin
      integer I;
      SU:=0;
      for I:=1 step 1 until J do
        SU:=SU+NAME(I)↑2;
      end;
  procedure TANP(FUNC,I,NAME,RESULT);
    value I;
    array NAME;procedure FUNC;
    real RESULT;integer I;
    begin
      real V;
      FUNC(I,NAME,V);
      RESULT:=sin(V)/cos(V);
    end;
  READ(N,M);
  READARRAY(X,1,N);
  READARRAY(Y,1,M);
  TANP(SUM2,N,X,TANSX2);
  TANP(SUM1,M,Y,TANSY);
  PRINTR(5,TANSX2);
  PRINTR(5,TANSY);
  CRLF(1);
end;
```

⑤～⑬は *ARRAY* という名の大きさ L の配列の要素の和を *SUM* に格納する手続き *SUM1* の宣言。

⑭～⑳は *NAME* という名の大きさ J の配列の要素の2乗和を *SU* に格納する手続き *SUM2* の宣言であって、パラメタの名前はこの手続き限りのものであるから勝手な名前を用いている。

㉑～㉒は1次の和か2次の和かの手続きの名前を仮りに *FUNC* とつけ、これと使用する大きさ I の配列 *NAME* およびその *FUNC* の結果の tangent をとった値を格納する場所 *RESULT* をパラメタとする手続き *TANP* の宣言の部分である。

㉓で V を局所的に宣言したのは *FUNC* という手続きの結果を一時受けて使用するためである。

㉔で *FUNC* が発動されているので本文で実行するときに *FUNC* として *SUM1* が指定されるか *SUM2* が指定されるかに従ってその指定によって手続き内容を大きさ I の配列 *NAME* について実行し答を V に格納する。

㉕ではこの結果 V を用いて $\tan(V)$ を計算し *RESULT* に格納している。

㉖からプログラムの本文が始まる。

㉗で *TAN* を発動し大きさ N の配列 X について *SUM2* という手続きを経由して結果を *TANSX2* に格納することを命じている。

㉘で実行を命ぜられた手続き *TAN* では大きさ M の配列 Y について *SUM1* という手続きを経由して結果を *TANSY* に格納する。

このように *procedure TANP* の仮パラメタとして *procedure* と規制された *FUNC* は実行に当たって指定された実パラメタ *SUM1* または *SUM2* におきかえられることによって複雑な手続きを *TANP* の中でいちいち書かなくてもすむことになるわけである。

なおこのような1個関数の場合には次章にのべる関数呼び出しのほうがいっそう簡単に取扱うことができる。

2) *string*
たとえば $A = (A \text{ の内容})$ $B = (B \text{ の内容})$ というふうに K ケタ数値の左側に勝手な字を書きたい場合はそのたびごとに

```
PRINTS('A-');
PRINTR(8,A);
CRLF(1);
```

または

```
PRINTS('B-');
PRINTR(8,A);
CRLF(1);
```

と書いてもいいが字と変数が変わるとにこれだけの文を書くのが面倒な場合宣言部で

```
procedure PR(S,A);
  value A;
```

```

real A; string S;
begin
  PRINTS('S');
  PRINTR(8, A);
  CRLF(1);
end;

```

と宣言しておけば、たとえばX = と書いてその右に変数Pの内容を書き、次の行にY = と書いてその右に変数Qの内容を印字したいときには、

```

PR('X=', P);
PR('Y=', Q);

```

でよい。このようにstring SとはSに対応する実パラメタの所に''につつまれた文字数字の並びがくると示している。

11. 関数呼び出し (function designator)

11.1 概要

(9) ⅱ) 2) でも述べたがprocedureの結果得られる答が1つの場合は関数呼び出しを用いる。関数呼び出しであることの宣言はprocedureの場合と同様であるが宣言詞procedureの前にその結果がreal, integer, Booleanのいずれであるかによってreal procedure等と付加することと $\tilde{F}P$ の中の結果を受けるパラメタを消して、結果は自分自身の名前に代入する点が異なること、使用はprocedureの場合手続き文として使用するのに対して関数呼び出しの場合は式の中で使用することである。

例で示したほうが分りやすいので前にprocedureであげたもののうち1個関数を扱ったものを関数呼び出しに直してみる。

11.2 関数呼び出しの実例

1) 相関係数

(9) ⅱ) で示した相関係数は1個関数なのでこれを関数呼び出しに直してみよう。データは前と同じく

```

N, X1, X2, ..., Xn, Y1, Y2, ..., Yn, ..., 0, ...

```

とする。

```

begin
  integer M;
  array P, Q(1:500);
  real procedure SOKANKEISU(N, X, Y);
  value N;
  integer N; array X, Y;
  begin
    integer I;
    real SX, SY, SXX, SY, SXY;

```

```

SX := SY := SXX := SYY := SXY := 0;
for I := 1 step 1 until N do
  begin
    real P, Q;
    P := X(I); Q := Y(I);
    SX := SX + P;
    SY := SY + Q;
    SXX := SXX + P2;
    SYY := SYY + Q2;
    SXY := SXY + P * Q;
  end;
SOKANKEISU := (SXY -
  SX * SY / N) / sqrt((SXX
  - SX2 / N) * (SYY -
  SY2 / N));
end;

```

```

ST: READ(M);
if M = 0 then go to STOP;
READARRAY(P, 1, M);
READARRAY(Q, 1, M);
PRINTF(14, SOKANKEISU(M, P, Q));
CRLF(1);
go to ST;
STOP;
end;

```

④でreal procedureという宣言詞を用いた、名前はSOKANKEISUという長いものにしたが、ここでは先頭の6文字しか判別しないものとして扱うからSOKANKとして扱われる。このように名前はいくら長くてもよいが計算機は先頭からの定められた文字数しか判別しない。また(9) ⅱ)での答を受ける $\tilde{F}P$ のRはとり去ってある。

⑤までは同じことであるが前はRに代入したものが今度はSOKANKという自分自身の名前に代入されている。

⑥でいきなりプリント文の中の式に関数呼び出しが使われており、実パラメタM, P, Qを用いて大ききMの配列PおよびQの間の相関係数が計算され自分自身の名前SOKANKに代入されたその結果がプリント準備される。

2) 実例 (2)

前に(9) ⅱ) 2)にあげたプログラムはいずれも1個関数の手続きなのでこれを関数呼び出しに直してみよう。データテーブルは、

```

N, M, X1, ..., Xn, Y1, ..., Yn

```

とする。

```

begin
  array X,Y(1:200);
  integer M,N;
  real procedure SUMA1(L,ARRAY);
    value L;
    integer L; array ARRAY;
    begin
      integer I;
      SUMA1:=0;
      for I:=1 step 1 until L do
        SUMA1:=SUMA1+ARRAY(I);
      end;
    end;
  real procedure SUMA2(J,NAME);
    value J;
    integer J; array NAME;
    begin
      integer I;
      SUMA2:=0;
      for I:=1 step 1 until J do
        SUMA2:=SUMA2+NAME(I)↑2;
      end;
    end;
  real procedure TAN(FUNC,I,NAME);
    value I;
    array NAME; real procedure FUNC;
    integer I;
    begin
      TAN:=sin(FUNC(I,NAME))/
        cos(FUNC(I,NAME));
    end;
  READ(N,M);
  READARRAY(X,1,N);
  READARRAY(Y,1,N);
  PRINTR(5,TAN(SUMA2,N,X));
  PRINTR(5,TAN(SUMA1,M,Y));
  CRLF(1);
end;

```

今度は3つとも関数宣言を行なっている。したがって
TAN のパラメタとしての FUNC も real procedure という
規制を行ない、これは式の中で使用してある。

本文の中でも TAN という関数呼び出しは PRINTR 文
の式の中で使われている。

3) 実例 (3)

大きさ N の配列 $X_1, \dots, X_N, Y_1, \dots, Y_N$ を読み込んで直
線の当てはめを行ない

$$\hat{y} = \hat{a} + \hat{b}x$$

という推定式を立てた場合 y の方向にはかつた距離の 2 乗
和を最小にするような $\hat{a}$ および $\hat{b}$ は

$$\hat{b} = S(xy)/S(xx)$$

$$\hat{a} = \bar{y} - \hat{b}\bar{x}$$

である。ただし、

$$S(xy) = \sum_{i=1}^n xy - (\sum_{i=1}^n x)(\sum_{i=1}^n y)/n$$

である。

またついでに相関係数

$$r = S(xy)/\sqrt{S(xx) \cdot S(yy)}$$

も計算し $\hat{a}, \hat{b}, r$ を有効数字 4 けたで印刷するプログラムは
 $n \leq 200$ とすれば

```

begin
  real SX,SY,SXX,SYY,SXY,MX,MY,R,A,B;
  array X,Y(1:200);
  integer N,I;
  READ(N);
  READARRAY(X,1,N);
  READARRAY(Y,1,N);
  SX:=SXX:=SY:=SYY:=SXY:=0;
  for I:=1 step 1 until N do
    begin real P,Q;
      P:=X(I); Q:=Y(I);
      SX:=SX+P;
      SY:=SY+Q;
      SXX:=SXX+P↑2;
      SYY:=SYY+Q↑2;
      SXY:=SXY+P*Q;
    end;
  MX:=SX/N;
  MY:=SY/N;
  SXX:=SXX-SX↑2/N;
  SYY:=SYY-SY↑2/N;
  SXY:=SXY-SX*MY/N;
  B:=-SXY/SXX;
  A:=MY-B*MX;
  R:=-SXY/sqrt(SXX*SYY);
  PRINTR(4,A);
  PRINTR(4,B);
  PRINTR(4,R);
  CRLF(1);
end;

```

end;

このプログラムを SXX, SYY, SXY, MX, MY を計算する
過程を procedure にまとめて同じことを実行するプログ
ラムに直せば次のようなものになろう。

```

begin
  real SXX, SYY, SXY, MX, MY, R, A, B;
  array X, Y(1:200);
  integer N;
  real procedure SS(N, X, Y, P);
    value N, P;
    integer N, P; array X, Y;
    begin
      integer I;
      SS := 0;
      for I := 1 step 1 until N do
        SS := SS + X(I) * (if P = 1
          then Y(I) else 1);
      end;
    READ(N);
    READARRAY(X, 1, N);
    READARRAY(Y, 1, N);
    MX := SS(N, X, X, 1) / N;
    MY := SS(N, Y, Y, 1) / N;
    SXX := SS(N, X, X, 2) - MX * 2 * N;
    SYY := SS(N, Y, Y, 2) - MY * 2 * N;
    SXY := SS(N, X, Y, 2) - MX * MY * N;
    B := SXY / SXX;
    A := MY - B * MX;
    R := SXY / sqrt(SXX * SYY);
    PRINTR(1, A);
    PRINTR(1, B);
    PRINTR(1, R);
    CRLF(1);
  end;

```

このプログラムの特色は⑤～⑬のSSという名の関数宣言の中でPに1を与えたら1次の和かPが1以外なら2次の和を求める手続としたことで、これは⑬⑭ではそれぞれXおよびYの和となっているが⑭の実パラメタ中3番目のXおよび⑭の実パラメタ中3番目のYは実はarrayの名前であれば何でもよいことになる。したがって⑭でSS(N, X, Y, 1)としてもまた⑭でSS(N, Y, X, 1)としてもそれぞれXおよびYの和を求めることには変りはない。

⑭～⑮でNが掛けているのはMXがすでにNで割っており、これを2乗しているので $1/N^2$ を $1/N$ にもどすためである。

12. スイッチ (switch)

real integer または Boolean の一群の値に名前をつけ、その何番目かを指定して使用することはarray という操作によって可能となったがこれと同様に名札を並べ全体に名前をつけてその何番目かを指定して使用できると

非常に便利である。この目的のためにスイッチがある。スイッチは宣言部において次の形による宣言を行なった後その名前の何番目として行先式の中で用いる。

```

switch I: = DE, ..., DE;
go to I (AE);

```

例として1966年1月1日が土曜日であることを知ってM月N日の情報を読んで何曜日をプリントするプログラムを作ってみよう。そこで1966年1月1日を起日とする総日数を出してこれを7で割ったあまりに1を加えたものをKとすればKが1なら金、2なら土……7なら木となるから、

$$K = K \div 7 * 7$$

によって7で割った余りが出ることを利用して

```
go to S(K - K ÷ 7 * 7 + 1);
```

という文で指定された曜日を印刷するようにしよう。

```

begin
  integer M, N, R, AE;
  switch S: = FR, SA, SU, MO, TU, WE, TH;
  procedure G; go to PR;
  integer array A(1:11);
  go to ST;
  FR: PRINTS('KIN'); G;
  SA: PRINTS('DO'); G;
  SU: PRINTS('NITI'); G;
  MO: PRINTS('GETU'); G;
  TU: PRINTS('KA'); G;
  WE: PRINTS('SUI'); G;
  TH: PRINTS('MOKU'); G;
  ST: READ(M, N);

```

```

  I := 1;
  for AE := 31, 28, 31, 30, 31, 30, 31, 31, 30, 31,

```

```
30 do
```

```

  begin
    A(I) := AE;
    I := I + 1;
  end;

```

```
R := 0;
```

```
for I := 1 step 1 until M - 1 do
```

```
R := R + A(I);
```

```
J := R + N;
```

```
go to S(J - J ÷ 7 * 7 + 1);
```

```
PR: CRLF(I);
```

```
end;
```

このプログラムの特長はA(I)にIを月数にして各月の日数を代入する部分と最後の飛越し文の部分であろう。

13. ALGOL によるプログラム例題 [11]

13.1 計画量の危険度の評価

過去 N 年間 ($N \leq 150$) の各年最大量のデータがある。
これを N を先頭に読み込み、今後 2 年、5 年、10 年、
20 年、50 年、100 年、200 年、500 年間にそれぞれ
1 回は上回るかもしれないが 2 回以上上回ることはないと
考えて、その考えが誤りである確率が 95% 以下であるよ
うな限界を求め、これを 2, 5, …… という数字の右側に印
刷するプログラムを作れ。

ただし各年極値は対数正規分布するものとするば、

$$MLX = \sum \ln X_i / N$$

$$PV LX = \sqrt{S(\ln X - MLX)^2 / (N-1)}$$

$$P \approx 1/M \quad (\text{たとえば 5 年に対して } 1/5)$$

$$\text{限界 } Q(P) = MLX + H(P) \cdot PV LX$$

$$H(P) =$$

$$K(P) + \sqrt{\left(1 - \frac{K(0.05)^2}{2(N-1)}\right) \left(K(P)^2 - \frac{K(0.05)^2}{N}\right)} \\ 1 - \frac{K(0.05)^2}{2(N-1)}$$

$K(P)$ は正規偏差であって

$$P = \frac{1}{\sqrt{2\pi}} \int_{K(P)}^{\infty} \exp\left(-\frac{t^2}{2}\right) dt$$

であり、その近似値は Hastings によれば $0 < P \leq 0.5$ に対し

$$K(P) = \eta - \left\{ \frac{230753 + 027061\eta}{1 + 0.99229\eta + 0.04481\eta^2} \right\}$$

$$\eta = \sqrt{\ln\left(\frac{1}{P^2}\right)}$$

である (詳細は中村: 技術者のための統計解析 10 章危険度の評価 参照)。

13.2 一元配置の分散分析

水準数 M 、各因子に属するデータサイズ $N_1, \dots, N_m$ を先
頭に第 1 水準 $x_{11}, \dots, x_{1n_1}$ 、第 2 水準 $x_{21}, \dots, x_{2n_2}$ ……
が入っているデータを読み込んで各水準ごとの平均値 MX 、
不偏分散の平方根 RVX を印刷し、次に分散分析表、

AOL TABLE

SV	SS	DF	MS	F
A	SSA	NA	FA	F $\bar{O}$
E	SSE	NE	FE	
$\bar{O}$	SS $\bar{O}$	N $\bar{O}$		

を印刷するプログラムを作れ。

ただし、

$$RVX = \sqrt{(\sum X^2 - (\sum X)^2 / N) / (N-1)}$$

$$SS\bar{O} = \sum \sum X^2 - (\sum \sum X)^2 / \sum N$$

$$= \sum \sum X^2 - CF$$

$$SSA = \sum_i \{ (\sum X)^2 / N_i \} - CF$$

$$SSE = SS\bar{O} - SSA$$

$$NA = M - 1$$

$$N\bar{O} = \sum N_i - 1$$

$$NE = N\bar{O} - NA$$

$$FA = SSA / NA$$

$$FE = SSE / NE$$

$$F\bar{O} = FA / FE$$

とする。

14. 例題解〔II〕（解）

前号においておいた練習問題の1つの解を示す。

14.1 計量量の危険度の評価

データテープ

$N, x_1, \dots, x_k$

印刷様式

```

2  x x x x x . x x
5  x x x x x . x x
10 .....

```

```

begin
  read LX, MLX, SLX, SLXX, RV, LX,
    P, H, KP, K5;
  integer I, N;
  array X[1:150];
  real procedure KPF(P);
    value P; real P;
  begin
    real K;
    K:=sqrt(ln(1/P↑2));
    KPF:=K-(230753+027061
      *K)/(1+099229*K
      +004481*K↑2);
  end;
  READ(N);
  READARRAY(X, 1, N);
  SLX:=SLXX:=0;
  for I:=1 step 1 until N do
    begin
      LX:=ln(X[I]);
      SLX:=SLX+LX;
      SLXX:=SLXX+LX↑2;
    end;
  MLX:=SLX/N;
  RV/LX:=sqrt((SLXX-MLX*SLX)
    /(N-1));
  K5:=KPF(005);
  for P:=05, 02, 01, 005, 002,
    001, 0005, 0002, do
    begin
      KP:=KPF(P);
      H:=(KP+sqrt(KP↑2-(1-
        K5↑2/2/(N-1)))*(KP↑2

```

```

      -K5↑2/2/(N-1)))/(1-K5↑2/(2*
      (N-1)));
    PRINTI(3, 1/P);
    PRINTF(52, exp(MLX
      +H*RV/LX));
    CRLF(1);
  end;
end;

```

real 型の $1/P$ を integer 型の印刷様式で打つと自動的に四捨五入を行なって整数にして印字する。

14.2 一元配置の分散分析

後の利用の便からこれ自身を1つの procedure にまとめ上げて、水準数 M 、各水準におけるくり返し数の配列 IA 、データマトリクス XA を与えて各水準平均の配列 MX 、各水準不偏分散の平方根の配列 RV および F 値 $F\bar{O}$ を計算し、 $\bar{A}\bar{O}V$ 表を印刷する procedure $\bar{A}\bar{O}V1$ を用いてみる。水準数は10以下、各水準のくり返し数は20以下としてみる。

```

① begin
② integer I, J;
③ real MX, RV, F̄O;
④ integer array IA[1:10];
⑤ array XA[1:20, 1:10], MX, RV
⑥ [1:10];
⑦ procedure AOV1(M, IA, XA, MX,
⑧ RV, F̄O);
⑨ value M;
⑩ integer M; integer array IA;
⑪ array XA, MX, RV; real F̄O;
⑫ begin
⑬ integer I, J, N̄O, NA, NE;
⑭ real SSA, SSE, S̄O, CF, FA;
⑮ array SX, SSX[1:10];
⑯ SPACE(5); PRINTS('MEAN');
⑰ SPACE(8); PRINTS('RV');
⑱ SPACE(8); PRINTS('N');
⑲ CRLF(1);
⑳ S̄O:=S̄O:=SSA:=0; N̄O:=NA:=0;
㉑ for I:=1 step 1 until M do
㉒ begin
㉓ integer K1, K2, K3, K4, K5;
㉔ real S, SS;
㉕ K1:=IA[I];
㉖ S:=SS:=0;
㉗ for J:=1 step 1 until K1 do
㉘ begin
㉙ real P;

```

```

P:=XA(I,J);
S:=S+P;
SS:=SS+P↑2;
end;
K2:=MX(I):=S/K1;
K3:=SSX(I):=SS;
K4:=SX(I):=S;
K5:=RV(I):=sqrt((SS
-S↑2/K1)/(K1-1));
PRINTI(2,I);
PRINTF(55,K2);
PRINTF(55,K5);
CRLF(1);
SSO:=SSO+K3;
SSA:=SSA+K4↑2/K1;
NÖ:=NÖ+K1;
SÖ:=SÖ+K4;
end;
NÖ:=NÖ-1;
NA:=M-1;
NE:=NÖ-NA;
CF:=SÖ↑2/(NÖ+1);
SSO:=SSO-CF;
SSA:=SSA-CF;
SSE:=SSO-SSA;
FA:=SSA/NA;
FE:=SSE/NE;
FÖ:=FA/FE;
SPACE(7);
PRINTS('AÖVTABLE');
CRLF(1);
PRINTS('SV');
SPACE(7); PRINTS('SS');
SPACE(7); PRINTS('DF');
SPACE(7); PRINTS('MS');
SPACE(7); PRINTS('F');
CRLF(1);
PRINTS('SA');
PRINTF(44,SSA);
PRINTI(3,NA);
PRINTF(33,FA);
PRINTF(33,FÖ);
CRLF(1);
PRINTS('SE');
PRINTF(44,SSE);
PRINTI(3,NE);
PRINTF(33,FE);
CRLF(1);

```

```

PRINTS('-----');
PRINTS(' ');
CRLF(1);
PRINTS('§§Ö');
PRINTF(44,SSO);
PRINTI(3,NÖ);
CRLF(1);
end;
READ(M);
READARRAY(IA,1,M);
for I:=1 step 1 until M do
for J:=1 step 1 until IA(I) do
READ(XA(I,J));
AÖV1(M,IA,XA,MX,RV,FÖ);
end;

```

AÖV1のprocedureの中で K1,K2,K3,K4,K5にarrayの要素をいったん受けるのは、何回も出てくる要素のI,Jを頼りにその番地をさがすのは時間がかかるのでさがす手間を1回限りにしておこうとの意図からである。字の印刷でPRINTS('§§SV');などとしたのは' 'の中では空白に換かうという本文内だけの規約に従ったものでSPACE(2); PRINTS('SV');と同じことである。プログラムの本体は最後の6行だけであって、それまではすべて宣言の部分である。

15. 特殊な注意事項

プログラミングを進めてゆくうえにはいまままでにあげた文法を守って書き下しさえすればよいわけであるが、計算機特性などによってちょっとした注意で計算能率が良くなったり、また文法的なエラーにならないが予定した計算を実行しないという誤りを防ぐことができるので以下その注意事項を別記する。

15.1 arrayの要素の取り扱い

(13)ii) にあげた例題の中でprocedure AÖV1の2つ目のbeginの中のJをコントロール変数とした繰り返し文に次のような部分がある。

```

for J:=1 step 1 until K1 do
begin
real P;
P:=XA(I,J);
S:=S+P;
SS:=SS+P↑2;
end;

```

このPという変数はこのブロックの中で局所的に宣言されXAという2次元のarrayのI行J列の要素を受けて以後はXA(I,J)の代りにPを用いている。これはもちろん

次のように書いても文法的には誤りでない。

```
for J:=1 step 1 until K1 do
begin
  S:=S+XA(I,J);
  SS:=SS+XA(I,J)↑2;
end;
```

ところがこのようにすると $XA(I,J)$ が式の中に現われるたびに I, J をたよりに XA という array の中味を調べに出かけるので多少演算時間がのび、これが 1 回ならたいしたことはないが繰り返し文の中で用いると累加されて相当な時間を要することもある。

また後にのべるようにモニタ（監視プログラム）を用いて次々と仕事の実行順序を制御し、ことに優先度の高い仕事を割り込んで実行させるためには array の要素の情報をいつ割り込まれてもよい状態に保つために記憶装置を占有することになり、自由に利用できる記憶容量が減少するので、繰り返し文の中で array の要素を式中使用するときにはたとえ 1 箇所しか現われない場合であってもいったん局所的に変数を宣言し、array の要素をそれに受けて使用することが望ましい。

15.2 繰り返し文の終期値の取り扱い

繰り返し文のコントロール変数が integer である場合は終期値についても実行させたいと思って書いたプログラムはそのとおり実行されるが real の場合は計算機によっては（2 進の計算機の場合）事情が異ってくる。たとえば、

```
for A:=0.1 step 0.05 until 3 do
begin
  PRINTF (1.2, A);
  PRINTF (1.4, A↑2);
  CRLF (1);
end;
```

というプログラムを書いた場合最後に、

3.00 9.0000

という印字を期待しているわけであるが 2 進の計算機では、

2.95 8.7025

で終わって最後の $A=3$ の場合は実行しないことがある。これはわれわれの書いた 10 進の内容を計算機が 2 進に変換するさいにきざみの 0.05 である 2 進数に変換され、途中でそれを 10 進に直してチェックしてみると、0.0500000001 になっているかも知れないためこれを数多く加えてゆくとちょうど $A=2.95$ のつもりが、10 進に直してみれば 2.950000005 くらいになっていて、これに 0.05 を加えると 3.000000005 となってしまい、これは until の条件を超えてしまっていることになり $A=3.0$ の場合を実行しないことがあるためである。

これに対しては until の後はきざみの $1/4 \sim 1/5$ を加え（符号も考えてつまり負のきざみのときは引いて）おくより仕方がないわけである。前の例でいえば、

```
for A:=0.1 step 0.05 until 3.01 do
```

とすればよいことになる。

これはまことにつまらぬ規則であってプログラマにとっては内部で 2 進の演算をしようがそれは知ったことではないわけであるから ALGOL 語翻訳のとき機械が自分自身できざみの $1/4$ を加え（または減じ）ておくべきものと考え、コンパイラ（翻訳編集プログラム）製作者にその要望を何回か行なったが数学出身のコンパイラ製作者はまことに落着いていて、このような「きかない」仕事には賛意を表してもらえなかった。そこで止むを得ずプログラマのほうで注意せざるを得ないわけである。

もちろん integer の場合は 2 進でも 10 進でも互いに変換した後は厳密に一致するのでこのような問題は起こり得ない。

15.3 end の後の ; の取り扱い

セミコロンは同種の宣言と他の宣言または文との区切り、および文と文の区切りに用いられる。したがって文と end との区切りに別段 ; は必要でないわけであるから (13)

1) の例題の最後のところは、

```
      | CRLF(1)
      | end
      | end
```

として別段さしつかえない。ところがこのように end end の重なりを続けて書いているうちに end の前は ; がなくてもよいということを end の後は ; がなくてもよいとどりがえると大変なことになってくる。

たとえば (13) 1) の procedure KPF(P) の最後のとこで

```
      | end
      | READ(N);
      | READARRAY(X, 1, N);
```

とした場合文法的には間違いないが別段ない。しかしわれわれが考えた演算結果は得られないはずである。それは end の後は次の ; がくるまでは comment扱いとなって読みとばしになるという規則があるため、次の READ(N) という文が落書きとなって読みとばされ、最初の ; の次つまり READARRAY 以降を正式の文と解釈するため、たとえばデータが

5, 3, 6, 8, 2, 9, 3, 2, 6, 4, 1,

となっているとき、先頭の 5 も X という array の 0 才 1 要素として読み込もうとするが、さて N という値がまだ数えられていないため計算機がウロウロすることになるわけである。

このような誤りを避けるためにも、また一つのプログラム全体を他のプログラムの一部として利用するときの便宜のためにも、文は end の前であらうと常に ; を入れ、最後の end の後にも ; を入れる原則を押しとおしたほうがよ

い。この場合、機械は *end* の前に空文が一つ置かれて、前の文と空文とが；で区切つてあるものと解釈し別段差しつかえないし、最後の *end* の後は、まったく読まないからこれもかまわないわけである。

15.4 繰り返し文の制御変数

繰り返し文の制御変数には完全に終止条件が終わつて次の文に移った後はどのような値が入っているかは保証の限りでない。

たとえば、

```
for K:=1 step 1 until N-1 do
begin
  read R;
  R:=A[K];
  S:=S+R;
end;
P:=A[K+1];
```

というプログラムは一見 *S* に *A* という array の 1~*N*-1 の要素の和が *N* を入れた後に *K* にはちょうど *N*-1 が入っているはずであるから *K*+1 つまり *N* 番目の要素の値を代入するという内容になっているように思われる。

ところが最初に述べたように繰り返し文終了後は *K* に何が入っているのかわからないので *K*+1 は何になっているか保証の限りでないわけである。この場合は当然、

$P:=A[N]$;

とすべきである。

もちろん繰り返し文の途中で飛び越し文によってその外に飛び出したときはコントロール変数の値はその脱出直前の値を持っているのでそのまま利用して差し支えない。

15.5 アルファモードの取り扱い

ALGOL 60 では文字の情報を変数に代入して転送したりした後で印字したりする取扱いは触れていない。しかし名簿や情報検索に使用するとき文字も情報として取扱いたい場合が多い。

これに対して Burroughs 5500 では *ALPHA* というタイプ型宣言詞があり 6 ケタの文字を 1 つの変数に格納できる OKITAC 5090 H 用の NEW ALGOL-H では *integer* の変数に対し読み込みはデータテーブル上 $\rightarrow TOKYO, \rightarrow YOKOHAMA, \rightarrow A$, などと前に $\rightarrow$ をつづて 7 文字までを 1 組に行ない、代入は、

$K := EQUIV('TOKYO');$

などという形で行なつて適宜転送処理し、印字は、

$PRINT('A', K);$

として *K* に入っている情報をアルファモードで 7 文字印刷させることができる。

TOSBAC 3400-40 用 NEW T-ALGOL では *alpha* 宣言を認める。

16. プログラムのエラーチェック

今までに述べたやり方で、問題を持つ人は自由に自分の計算させたいと考えていることをプログラムに書きおろすことができる。しかしなんと人間的作つたものであるから文法上許されない書き方や考え方のあやまりは避けられない。そこで計算機はプログラムの翻訳にかかると、あらかじめ教えられたやり方で文法チェックを行ない、エラーを検出したときはその場所とエラーの性質とを知らせてくる。

われわれはこのエラーメッセージに従つてプログラムを自分の意図する形に修正し、また計算機にかけてやればよいわけであるが、あまりたびたび直すことも面倒であるし、高価な機械を切等的なエラーの検出に用いることはもったいないこともある。そのためプログラムを書いたならば機械に掛ける前に次の点について見直しておいたほうがよい。

i) 変数関係

- 1) 未宣言の変数を使っていないか?
- 2) 宣言されたブロックの外で、そのブロック内と同じ意味に同じ名前が使用されていないか?
- 3) 同じブロック内で変数・名・手続き・関数呼び出し・スイッチ等の名前に同じものを使っていないか?
- 4) 変数はそれに代入文がまたは読込み文によって値を与えられる前に式の中で使用されていないか?

ii) 区切り記号

- 1) 区切り記号の繰りに誤りはないか?
- 2) *begin* と *end*, (と), [と], 'と' などのカッコが対応しているか?

iii) 式関係

- 1) 条件付き算術式・論理式・行先式は *if then else* の 3 つをともに含んでいるか?
- 2) 条件付きの式で *then* の次に *if* がくるときは *if* 以下の式を () につつんであるか?
- 3) 条件付きの式を作用素で結ぶときは、その条件付きの式を () でつつんであるか?
- 4) 掛算の * がおちていないか? 特に $(i + B) C$ とか $4 B$ などに注意?
- 5) 割算の分母に 0 が入るようになっていないか? これはそのつもりでなくても何も代入や読込みをしていない変数をいきなり式の中で使用すると 0 が入っているおそれがある。

- 6) 負または 0 の *ln* をとっていないか?

- 7) 負の *sqr* をとっていないか?

- 8) 2 つ以上の作用素が並んでいないか?

$A \div 5$ や $3 \div 48$ に注意

iv) 文関係

- 1) 文は *else* の直前を除き、すべて ; で終っている

か？

特に *begin* と *end* でつつんだ複合文またはブロックの場合に *end* の後に ; を落してはいないか？

2) 多重代入文の左辺の変数の型は皆同じものか？

3) 条件文で *then* の後に *if* がくるときは *if* 以下の文を *begin* と *end* でつつんであるか？

4) 繰り返し文の *do* 後のくり返すべき文は最初の ; まで (*begin* がくるときは *end* の後の ; まで) であるがそれでよいか？

5) 繰り返し文のコントロール変数を、指定されたとおり計算終了してから次の命令の中で終期値がまだ入ってい

るかと考えて使っているようなことはないか？

6) 手続き文の実パラメタが未宣言ではないか？

(仮パラメタは手続き宣言中の局所的なものに過ぎない)。

7) 手続き宣言中に非局所変数 (特に繰り返し文のコントロール変数) を使用してあるのに、手続き文の外で同じ変数を違った意味で使用していないか？

V) そ の 他

1) *comment* は : で終わっているか？

2) *end* の後の ; を落したために次の文が *comment* 扱いにされて読みとばされていないか？

17. KDK ALGOL の構想

17.1 型

型	定 数	単 純 変 数		添 字 付 変 数		関 数		式
		宣 言	規 制	宣 言	規 制	宣 言	規 制	
実 数	-4810-5等	real	real	real array	*1	real procedure		算 術 式
整 数	8 等	integer	integer	integer array		integer procedure		"
論 理	true false	Boolean	Boolean	Boolean array	同 左	Boolean procedure	同 左	論 理 式
記 号 列	'PQS'等	(alpha)	string	(alpha) array		(alpha) procedure		(文字式)
名 札	L 5 等	(label)	label	(label) array	*2	(label) procedure		行 先 式

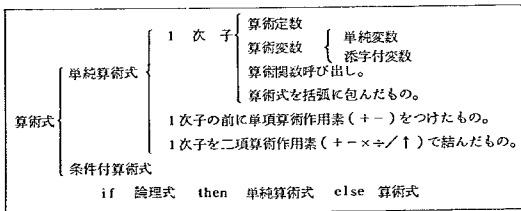
* 1 単に array とすれば real array

* 2 ALGOL 60 では switch で宣言と代入を兼ねている。

ただし NEW T-ALGOL では記号列に関しては全面的に認め、名札に関しては label array を switch として現行通りとする。

() 内は KDK ALGOL で新設

17.2 式 expression

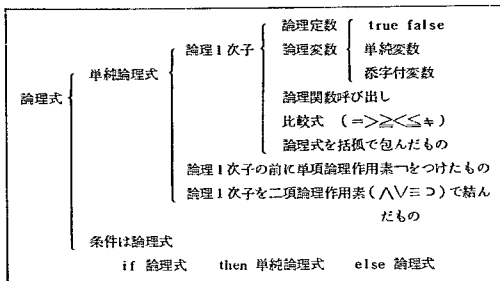


1. 標準関数として sinh, cosh, tanh, log, random (一様乱数), nrandom (正規乱数), fix (切り捨て) を追加する。

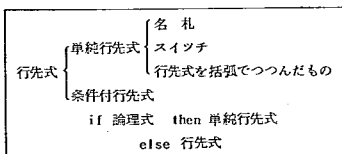
ただし NEW T-ALGOL では大文字とする (標準関数

も大文字) はか、INTEGER (論理値を 0,1 にする)。

2. double (倍精度実数) 宣言を認める。NEW T-ALGOL でも同様

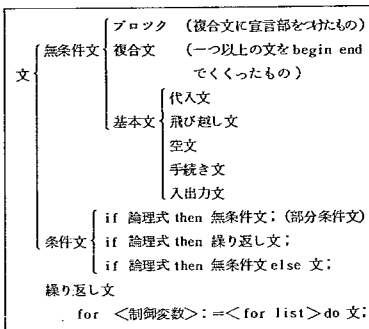


1. 標準関数として key- を認め
コンソールの対応ボタンが on なら
true そうでなければ false とする。
NEW T-ALGOL では KEY,
BOOLEAN(0,1 を論理値に換算)
を認める。



ALGOL の構想を受け入れる。(一語6文字)
式に関する注意
複素数宣言 complex を認める。NEW T-ALGOL
も同じ。

17.3 文 statement

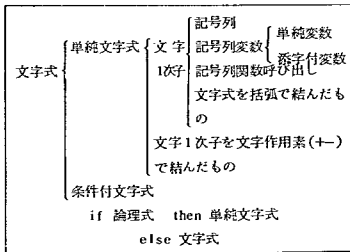


1. スwitch の添字式の値が宣言の範囲外となった時はその
飛び越し文はエラーとする (ALGOL 60 では空白扱い)
NEW T-ALGOL でもエラーとする。

2. KDK アルゴルでは単純行先式を名札, 名札変数
(単純変数, 添字は変数), 名札関数呼び出し, 行先式を
括弧で包んだものとしたいが, NEW T-ALGOL はこ
れを認めず ALGOL 60 通りとする。

3. KDK アルゴルでは名札配列宣言 label array を
認める代りに switch を廃し, その機能のうちの代入を
fill A(5, *) := A1, A2, ...; 等の一般配列代入文
で満たすようにしたいが, NEW T-ALGOL では名札
に関してはこれを禁止する。

4. 名札は名前に限る (符号なし整数を認めない)
NEW T-ALGOL も同様



1. 文字作用素

'ABC□E' + '□□□D' は 'ABCDE'
'ABC' - '□□B□' は 'A□□'

2. NEW T-ALGOL では全面的にこの KDK

1. 配列代入

fill A(1, *) := A1, A2, ...; と宣言された
範囲に逐次代入することを認める。

NEW T-ALGOL も同じ。

2. 空白及び end の後のコメントに, 関するトラブルを
避けるため do ; 及び end の後; が出てくるまでの間に,
:= が来るとエラーとする。

NEW T-ALGOL もこれに従う。

3. 入出力文として JIS 水準 7 0 のほか次のものを認
める。

READ(A, B, C, ...); 宣言した通りの大きさなら配列 名も可
PRINTI(N, E);
PRINTR(N, E);

```

PRINTF( $n_1 \cdot n_2, E$ ):  $n_2$  は9までに限る
PRINTA( $N, E$ ): 文字式  $E$  の値の先頭  $N$  文字分
PRINTS('×...×');
SPACE( $N$ ):  $N = 0$  も認める
CRLF( $N$ ):  $N = 0$  も認める
READARRAY( $K, L_1, U_1, L_2, U_2, \dots$ ):
PUTD( $N, A$ ): レベル  $N$  より配列  $A$  を1時格納
GETD( $N, A$ ): 〃 に読み出し
PUTMT( $N, A$ ): 〃 をMTに格納
GETMT( $N, A$ ): 〃 にMTより読み出し

```

4. 条件文で *then* の後の繰り返し文の後に *else* が続くことはあり得ない。その場合は繰り返し文を *begin end* で包む。

5. *<for list>* は次のようなものである:

I	算術式要素	算術式
II	階度要素	算術式 <i>step</i> 算術式 <i>until</i> 算術式
III	条件要素	算術式 <i>while</i> 論理式
IV	複合要素	I~III及びIVをカンマでつないだもの

17.4 手続き *procedure*

1. 手続き宣言

```

procedure 手続名<仮パラメタ部>;
  <値の部>
  <規制部>
  <手続き本体>;

```

手続き本体は常にブロックとして取り扱われる。

規制のない仮パラメタはすべて実数型とみなす。NEW T-ALGOL も同じ

2. 手続き文

```

手続き名<実パラメタ部>;

```

3. 一価関数の時は *procedure* の前にタイプ宣言詞をつけ本体中に手続き名に対する代入文を設け、その使用は関数呼び出しとして式の中で行なう。

4. パラメタの区切りとして、のほかに
<文字列>: (を認める。

5. 仮パラメタと実パラメタの対応

仮パラメタ	規制詞	実パラメタ
算術単純変数	<i>real, integer</i>	算術式
論理単純変数	<i>Boolean</i>	論理式
配列名	<i>array</i> <タイプ> <i>array</i>	配列名
関数名	<タイプ> <i>procedure</i>	関数名
手続名	<i>procedure</i>	手続名

スイッチ名	<i>switch</i>	スイッチ名
名札変数名	<i>label</i>	行先式
記号列変数名	<i>string</i>	記号列
文字変数名	<i>alpha</i>	文字式

スイッチ名は本来なら *label array* の宣言があれば、配列名の中に入れるべきであるが、NEW T-ALGOL ではこのようにしておく。

17.5 宣言 *declaration*

1. 型宣言

real integer Boolean のほか KDK ALGOL では *double complex string label* を認めるが、NEW T-ALGOL では *double complex alpha* のみを追加する。

2. 配列宣言

添字の境界は算術式で良い。

KDK ALGOL では *alpha array* と *label array* を NEW T-ALGOL では *alpha array* のみを追加する。

18. 各機種間の文法制限と入出力文

前にのべたように ALGOL 60 には入出力に関する何の規定もない。これは各機種の固有の特性を尊重したものであるが、かえって互換性の点で不便を感ずることも多く国際標準化機構 (ISO) でも一つの方式の勧告を行なっている。OKITAC 5090 II 用の NEW ALGOL-II においてもこの勧告を大綱に取り入れてあるが、現在のところ ALGOL を受け入れる機械によってこの入出力関係だけがいろいろな形をとっている。他の部分では標準関数や論理値を大文字にしたり、区切り記号もすべて大文字にしたりする点と多少の文法上の制限事項を除けば完全に共通に使用できるので、次にあげる入出力関係の書き換えだけでほとんどそのまま一つの機械から他の機械に移ることも可能である。また場合によってはその I/O 関係の変換すら計算機に行なわせて、一つの機械に通ったプログラムをそのまま他の機械に掛けることも可能である。

NEW ALGOL-H と CDC 3600 用の ALGOL の間にもこの種の変換プログラムができており、土木研究所の機械で文法チェックもすませ1つ1時間かかることが分ったもの20通りを CDC 3600 に実行させることもできる。CDC 3600 は後述するように現在世界最大の機械であって OKITAC 5090 II の約 1/40 の時間で処理することができものである。

以下各機種変換の便のために文法制限事項と入出力文を列記する。

18.1 TOSBAC 3400-40用 NEW T-ALGOL

1. ALGOL 60よりの制限

i 名前に小文字を禁止し先頭の6文字について判別する。

ii 名前は名前に限る。

iii 規則のない仮パラメタはすべて実数型とする。

2. ALGOL 60の範囲内だがNEW ALGOL-Hよりの拡張

i *own* を認める。

ii 一般に区切り、のほかに<文字列>:(を認める。

iii 条件式中の型は混合することを認める。

iv その段階で宣言していない非局所変数を手続宣言中で使用することを認める。

v 配列宣言の限界に算術式を認める。

vi 括弧の次に直接名前が来た場合エラーとする。

3. ALGOL 60よりの変更

i 標準関数は大文字とし次のものを追加する。*sinh*
cosh tanh log random (一様乱数) *nrandom* (正規乱数) (論理値は小文字)

ii データの区切りは・とし最後は、でも、でも良い。

iii 宣言の範囲を越えたスイッチはエラーとする。

4. ALGOL 60よりの拡張

i 宣言詞に *alpha* を追加する。

ii 宣言詞に *double complex* 追加

iii 配列入文

fill A(2,) := A1, A2, ...;*
を認める。

iv 標準手続格納ならびにその使用のための宣言詞
external を認める。

v プログラムの読み込み停止 *halt*; 及びデータの読み込み停止/を認める。

vi (データに対し 3×5.1 といった表現を認める。)

vii *T.F.A.P.* との互換を認める

viii プログラム分割処理のため *segment* を認める。

ix *do*; 及び *end* の後に; が来るまでに; が来た場合エラーとする。

x JIS水準70の入出力の記号 *PRINTI, PRINTR, PRINTF, PRINTS, PRINTA, SPACE, CRLF, PUTD, GETD, PUTMT, GETMT, READ, READARRAY* を認める。

xi 実数型の階段要素の *until* の後の算術式には *step* の後の算術式の $1/5$ が自動的に符号も含めて加えられる。

xii *KEY*

(コンソールの対応ボタンが on なら *true* そうでなければ *false*)

xiii *BOOLEAN*(0,1)を論理値に対応させる)

5. その他モニタの支配下で10分間までの優先割り込み

計算を認めるものとする。また一般的に手続きの再帰的呼び出しを認める。又 *comment* は区切り記号 名前の次に任意に置くことができる。

18.2 OKITAC 5090-H用 NEW ALGOL-H

A. 文法上の制限

a *identifier* に小文字を認めない。

b 標準関数・論理定数は大文字とする。

c *own* を認めない。

d 名前は *identifier* に限る。

e 手続きおよび関数呼び出しの $\overline{FP}$ と $\overline{AP}$ の種類は一致しなければならない。

f 手続きおよび関数呼び出しの $\overline{FP}$ はすべて規制されなければならない。

g *procedure* の *heading* 部に *comment* を書けない。*body* で *global* な量を使用するときは既宣言でなければならない。

h *variable array* を認めない。

i 算術式でカッコの直後の文字数字は無視する。たとえば $(A+B) \cdot C+E$ を $(A+B)C+E$ と誤って書くこと $(A+B) \cdot E$ とみなす。

j 算術式の *then* の後と *else* の後の $\overline{AE}$ の型は一致しなければならない。

k *array* の *value* 指定を認めない。

B. 文法上の拡張

a アルファモードの代入・読込・印字を *integer* タイプの変数に対して許す。代入は *EQUIV* ('') からなり、読み込みは1語、印字後は7文字まで。

b *external procedure* を認める。

c on 6~10 の数 *then* によりコンソールよりのコントロールを認める。

d *FIX* による切り捨て、*FLOAT* による *real* 化ができる。

C. I/O 文

a *READ*($\overline{V_1}, \overline{V_2}, \dots$);

変数の型は *real integer* いずれであっても $\overline{V}$ の型に合わせ変換する。なお大きさが宣言されたこととの *array* に対して *array* の *identifier* でもよい。

b *READARRAY*($\overline{I}, L1, U1, L2, U2, \dots$);

テープより $\overline{I}$ という *array* に格納、たとえば N 行 M 列のマトリクス A なら

READARRAY($A, 1, N, 1, M$);

また宣言したことの大きさの *array A* に対しデータを読み込むなら *READ*(A); でもよい。

c 宣言部 *external, PRINTR, PRINTF,*

PRINTI, SPACE, CRLF, PRINTS; としておけば建設的共通文法に対して *PRINTS* の ('') を (") になおすので、他はすべて適用する。なお詳細な指定は次による。

PRINT(<'format'>, $\tilde{V}_1, \tilde{V}_2, \dots$);
 <'format'>の指定に従って $\tilde{V}_1, \tilde{V}_2$ の順に印字
 <'format'>としては

R: $\square \pm \underbrace{\times \times \dots \times}_{9 \text{ ケタ}} \square$
 I: $\square \pm \underbrace{\times \times \dots \times}_{13 \text{ ケタ}} \square$
 L: $\square$ TRUE, $\square$ または $\square$ FALSE
 X: レプリケータで10ケタまでに限る。
 B: ブランク
 †: 新しい頁の始めより印字などMT およびHTP に対してTMを訂正
 /: それまでの情報を印字して1行おくり
 : : 改行なしの印字
 D: 1ケタの情報
 Z: 1ケタの情報(ゼロサブレスつまり始めの0の0は消して005は5とする, ただし最後の1つの0は消さない。)
 . : 小数点印字
 10: 10印字
 - : 符号を負のときは印字
 + : 符号を印字
 F: 小数点の位置を印字なしに指定
 S: 字を印字
 C: カンマの印字
 A: アルファモードの情報を印字(つまりTOKYOと
 いった文字の代入してあるintegerのidentifier
 の内容を印字)
 F: Boolean タイプの情報印字
 P: Boolean タイプの情報印字(TRUE なら1 FALSE なら0)
 T: 打ち切りでDのときは四捨五入される。
 'x...x': ' 'の中の文字そのものを印字
 このうち、10-+を除いた他のはその前に正整数を書いて何回かのくり返しを指示できる。またくり返しを回数で指示したいときたとえばN個のブランクとM個の*印を印字したければ
 REPX(N, M);
 PRINT('XB X('*)/'); だよ。
 3つのreal A, B, C と * * * の符号ならびに2つのinteger I, J を1行に印字したければ
 PRINT('3R, '*****'2I/', A, B, C, I, J);
 d PUT(<label>, <list>);
 GET(<label>, <list>);
 label は符号なし整数, list は式またはarray名・磁気テープにレベル1よりarray Aの内容を移し, array Bに受取るなら
 PUT(1, A);

として後、

GET(1, B);

としてBを使用すればよい。

e PUNCH(<'format'>, <list>);

高速度テープパンチ機により紙テープ作成。

f MWRITE(<unit>, <'format'>, <list>);

g MTREAD(<unit>, <'format'>, <list>);

unit は機番, 異なったプログラム間の磁気テープによる受渡し。

h REWIND(<unit>, <unsigned integer>); 磁気テープ巻きもどし。

i TMSEARCH(<unit>, <unsigned integer>);

指定テープマターチ。

18.3 FACOM 230用 230 ALGOL

A 文法上の制限

a own を認めない。

b 手続きおよび関数呼び出しの $\tilde{F}\tilde{P}$ と $\tilde{A}\tilde{P}$ の種類は一致しなければならない。

c arrayのvalue 指定はできない。

d 名前はidentifierに限る。

e 手続きおよび関数呼び出しの $\tilde{F}\tilde{P}$ は規制されなければならない。

f 空白は意味をもつ。

g 手続きおよび関数呼び出しにrecursive call(再帰的呼出し)を認めない。

B 文法上の拡張

a 先頭のbeginの後にcomment reallength := 15, integerlength := 8; などとケタ数指定ができる(可変長語の計算機に限る)。

C I/O 文

a Readreal($\tilde{I}$);

Readinteger($\tilde{I}$);

b Readarray($\tilde{I}$); 宣言されたおりの大きさのデータに限る。テープよりarrayを列なりに $\tilde{I}$ に読む。

c Printinteger($\tilde{A}\tilde{E}$);

整数を有効Aケタで印刷

Print($\tilde{A}\tilde{E}$);

実数を有効10ケタで浮動小数点型で印刷

Printfix($\tilde{A}\tilde{E}, n_1, n_2$);

実数を小数点上 n_1 以下 n_2 ケタで印刷

d Printreal($\tilde{A}\tilde{E}, n$);

実数を有効nケタで印刷

Printx($\tilde{A}\tilde{E}, n$);

整数を有効nケタで印刷

c Printstring('x...x');

' 'の中のものを印刷

Space(n);

空白を n 個印刷

$CRLF(n)$;

以上の結果を 1 行に印刷し n 行おくる。

ただし $CRLF$; とすれば 1 行おくる。

$LFEEED$;

新しい用紙の先頭までおくる。

18.4 FACOM 230-50 ALGOL

A. 文法上の制限

a own を除く

b 名札は identifier に限る

c identifier の有効長は 12 文字

d 実際に評価してみなければわからぬ $\overline{AE}$ のタイプはすべて $real$ とする

e call by name のパラメータは $integer$ と規制されなければすべて $real$ とする。

f $\overline{FP}$ と $\overline{AP}$ のタイプは一致しなければならない

g $\uparrow$ の演算結果はすべて $real$ とする。

B. 文法上の拡張

a 標準関数に $LOG, TANH, RANDOM$ 等を追加

C. I/O 命令

a $READ(\overline{V}_1, \overline{V}_2, \dots, \overline{V}_n)$;

変数の型に合わせてデータテープより読込まれる。

b $PRINT(\overline{AE})$;

$\overline{AE}$ がそれぞれの標準型で印刷準備される。

c $PRINTSTRING('x \times \dots \times')$;

' ' の中の文字が印刷準備される。

d $SPACE(\overline{AE})$;

e $CRLF$; 1 回復帰改行 n 回ときは

for $I:=1$ step 1 until N do $CRLF$;

f $OUTPUT(200, 'format', list)$;

200 は機番, list は $\overline{AE}$ をカンマでつないだもの。

format 中

Z ゼロサプレスするクダ

D ゼロサプレスしないクダ

. 小数点

V 小数点扱い (印字しない)

+ 符号表示

- 負表示

A アルファタイプ

F ブーリヤン (T または F)

I ノンフォーマット integer

R ノンフォーマット real

L ノンフォーマット Boolean

S 文字

() くり返し

'' 文字より入

$\uparrow$ 紙の先頭より印刷開始

/ 復帰改行

$\sqcup$ スペース

たとえば

$OUTPUT(200, 'A=' \sqcup R, \sqcup, 'B=' \sqcup -4Z, \sqcup 2D,$

$\sqcup, 'C=' \sqcup, \sqcup -5D_{10} \sqcup -ZD, \sqcup, \sqcup /', A, B, C)$;

$OUTPUT(200, '4(-4D_{10} \sqcup -ZD, \sqcup, \sqcup /', P,$

$Q, R, S)$;

18.5 TOSBAC 用 KT-ALGOL

A. 文法上の制限

a identifier は文字のみとする

b 下段の delimiter はすべて ' ' でかこむ

c own array の限界は定数で指示しなければならない

d 名札は identifier に限る

e 手続きおよび関数呼び出しの $\overline{FP}$ は call by name の $integer$ タイプのもののみ規制必要

B. 文法上の拡張

C. I/O 文

a $READ(\overline{V}_1, \overline{V}_2, \dots)$;

b $READARRAY(I)$;

c $WRITE(n, \overline{AE}_1, \overline{AE}_2, \dots, \overline{AE}_n, <format>)$;

18.6 CDC-3600 用 ALGOL

A. 文法上の制限

a identifier は小文字を認めない

b 下段の delimiter は ' ' でつづかなければなら
ない

c 手続きおよび関数呼び出しの recursive call

(再帰的呼出し) を認めない

d array の value 指定を認めない

e 手続きおよび関数呼び出しの $integer$ type の

$\overline{FP}$ は明細指定が必要である

f 手続きおよび関数呼び出しの $\overline{FP}$ と $\overline{AP}$ の種類は一致しなければならない

B. 文法上の拡張

a $PRINT(\text{for } I:=1 \text{ step } 1 \text{ until } 4 \text{ do } A(I))$;

を認める

b $WRITE(\text{if } D=0 \text{ then 'TRUE' else 'FALSE'})$;

を認める

C. I/O 文

a $READ(\overline{V}_1, \overline{V}_2, \dots, \overline{V}_n)$;

変数の型に合わせてデータテープより読込まれる。

b $WRITE('x \times \dots \times')$;

の中の字が印字される (実行し 1 行おくる) 字は上段に限る。

c $PRINT(\overline{AE}_1, \overline{AE}_2, \dots, \overline{AE}_n)$;

何語書いてもよいが 6 個ごとに改行される。印刷は $\overline{AE}$ の型に従って行なわれる (実行して 1 行おくる)

d WRITE(if D=0 then 'TRUE' else 'FALSE');
 プーリヤン変数 D を印刷したい場合でこのように条件式が
 書ける。

e PRINT(for I:=1 step 1 until 10 do A(I));
 配列の印刷でこのような特別な for 式というべきものが書
 ける。

f OUTPUT(61, '(format)', list);
 一般の印字はこの文による。61 は番番で 61 を用いる。
 次は format であるが常に '()' でかこむ。3 番目が
 AE をカンマでつないだものである。

format は w, d をくり返し変数とすれば
 Ew, d 指数部をもつ実数
 Fw, d 指数部なしの のみを含む実数
 (この両者の場合 w は全体のケタ数 d は小数
 点以下のケタ数)
 Iw 整数
 Aw アルファタイプ情報
 ull 文 字
 wX スペース
 / 復帰改行

で次の例による。

```
OUTPUT(61, '(2H A=, E8.2, 3X,
2H B=, F2.6 /)', A, B);
OUTPUT(61, '(10 F6.2 /)', for I:=1
step 1 until 10 do B(I));
```

19. ALGOL プログラムの変換

(16) でのべたように ALGOL 語族に属するものは多
 少の文法上の制限の差異と入出力文の違いだけで簡単に
 他機種用のアルゴリズムプログラムに変換できる。そこで次の
 例をいままて本文でとり上げた文法によって記述し、それ
 を変換してみよう。

19.1 立方根の問題

$A = \sqrt[3]{X}$ を求めるには

$$A := X \uparrow (1/3);$$

としてもよいが、この方法では $\ln(X)$ の $1/3$ 倍を exp
 するという手続きになり多少演算時間が長くなる。また誤
 差を 10^{-6} にとどめたいなどという意志表示も入らないの
 で Newton の近似を用い、

x_i を i 回目の近似値とすれば、

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

であるから

$$f(x) = x^3 - N = 0$$

に対し

$$x_{i+1} = x_i + \frac{1}{3} \left(\frac{N}{x_i^2} - x_i \right)$$

$$\left| \frac{1}{3} \left(\frac{N}{x_i^2} - x_i \right) \right| \leq 10^{-6}$$

の条件のもとに立方根の関数呼び出しを作り、A, B を読
 込んで $\sqrt[3]{A} + \sqrt[3]{B}$ をプリントさせるプログラムを作ってみ
 よう。

```
begin
  real A, B;
  real procedure CUB(C);
    value C;
    real C;
    begin
      real C1, C2, D;
      C1 := C/3;
      ST: D := (N/C1↑2 - C1)/3;
      if abs(D) ≤ 10-6 then
        CUB := C1 + D
      else begin C1 := C1 + D;
              go to ST;
            end;
    end;
  end;
```

19.2 NEW ALGOL-H への変換

(19)1) のプログラムを OKITAC 5090 H 用 NEW
 ALGOL-H に変換すると次のようになる。

```
begin
  real A, B;
  real procedure CUB(C);
    (19)1) のとおりただし abs は ABS に変える
  READ(A, B);
  PRINT('A= ', AD10-ZDB, 'B= ', AD10-ZDB,
    A, B);
  PRINT('B= ', 6D10-ZDB, CUB(A)+CUB(B));
end;
```

19.3 FACOM 230 ALGOL への変換

```
begin
  real A, B;
  real procedure CUB(C);
```

(19)1) のとおり

```
Readreal(A);
Readreal(B);
Printstring('A=');
Printreal(A,4);
Printstring('B=');
Printreal(B,4);
Printreal(CUB(A)+CUB(B),6);
CRLF;
end;
```

20. ALGOL と FORTRAN の比較

前にのべたようにIBM 704機用に1956年 FORTRAN (formula translator) が生まれ、1958年この長所もとり入れて国際共通語ALGOL が生まれ、逆にこの長所を見てFORTRAN IV が作られている。

ALGOL は現在1962年の改訂ALGOL-60 が基となっているがこれらの二つの言語の間にはその成立の経過からもわかるように、多くの相似点がある。しかも現在までのIBM社の世界市場の支配ぶりからみても多くのFORTRAN プログラムが過去において作成され、現在もこれらの遺産に頼って仕事をする向きには重要な言語であることはもちろんであるが、何といってもIBM社の機械向きに作られたという動機と、IBM機のプログラマといえど機械語および中間語の訓練を受けたプログラマということを前提としているためか、あまりにも機械の側に都合良くしようとするあまり、人間側に過度の注意を強制される結果となっている。この点で始めから人間と機械はもちろ、人間と人間の間の会話を念頭においてきっちりと筋の通った文法のもとに組み立てられたALGOL との間には大分差異が見受けられ、筆者自身も数年前にFORTRANによる短期教育を試み、ALGOL のように2日間くらの教育によって習得させることは困難であると感じたので建設省関係はすべてALGOL 一本による方針のもとに現在まで、1,500人に近いプログラマの養成を行なってきた。

しかし、よくALGOL とFORTRANの差異についての質問を受けることが多いので、以下表-1に簡単にその比較を行なってみた。

FORTRAN	ALGOL	PL/1
IF(P.EQ.Q)GO TO 10 IF(R.GE.S)GO TO 20 A=B(4)-3.0*L2 GO TO 50 20 A=B(4)-3.0*L1	if P=Q then goto 10; if R≥S then goto 20; A:=B[4]-3.0*L2; goto 50; 20:A:=B[4]-3.0*L1;	IF P=Q THEN GO TO L10; IF R>=S THEN GO TO L20; A=B(4)-3.0*L2; GO TO L50; L20:A=B(4)-3.0*L1;

表 - 1

問題点	FORTRAN	ALGOL
1. フロック構造	ない そのかわり無宣言で変数を使用できるが先頭の文字がI~Nのときはintegerとなる。	ある そのかわりすべての名前は宣言を必要とする。
2. integerとrealの混合演算	認めない。	認める。
3. 配列次元数	FORTRAN IIでは2次元、IVでは3次元	無制限
4. くり返しのコントロール変数	正のintegerに限る	無制限
5. 倍精度演算	IVではDOUBLE宣言	可変語長の場合を除き特別措置要。算術式、論理式、指示式に対してある。
6. 条件式		条件式を含め任意
7. 配列の添字式	C+V±Cの形に限る(C:定数, V:整数型変数)	

したがって、AにBというベクトルのI番目(IはP=Qなら3、さもなければ4)と3×(もしR≥SならL1さもなければL2)の差を代入したいとき、

ALGOL ならば

```
A:=B[if P=Q then 3 else 4]-3*(if R≥S
then L1 else L2);
```

でよいが、これをFORTRAN で書くとき条件式がないために次のように長いプログラムを必要とすると共にL1、L2がrealのときは3でなくて3.0としなければならぬ。

またこのFORTRAN のプログラムをALGOLに苦もなく対応させることが出き、同様にPL/1に変えてみるとPL/1 がALGOLに近いものであることがわかる。

GO TO 50	goto 50	GO TO L50;
10 IF(R.GE.S)GO TO 30	10: if R $\geq$ S then goto 30;	L10: IF R $\geq$ S THENGO TO L30
A=B(3)-3.0*L2	A:=B(3)-3.0*L2;	A=B(3)-3.0*L2;
GO TO 50	goto 50;	GO TO L50;
30 A=B(3)-3.0*L1	30:A:=B(3)-3.0*L1;	L30: A=B(3)-3.0*L1;
50	50:	L50:

この問題点にてらしてみても過去においてすでに用意されたプログラムに頼るのではなく、これから自分の問題に向って行かなければならないしかも多種多様の専門分科された分野の科学・技術者達は専門外のプログラムに問題を説明してプログラムを書いて貰うというやり方でなく自からプログラムを書く方がはるかに容易で、その目的のためには位置とか順序が単なる約束で書かれる部分の多い FORTRAN よりは、筋の通った文法によって、しかも step P until Q 等の視覚にうたえる要素の多い ALGOL が適していることと結論することができよう。

21. 練習問題 (11)

例によって練習問題をかかげておく、解の一例は次号に示す。

i) 計画量の危険度の評価

(13)i) にあげたプログラムを NEW ALGOL-H に変換せよ。

答は何行をどのように直すということだけでよい。

ii) 一元配置の分散分析

(13)ii) のプログラムを NEW ALGOL-H および FACOM230 アルゴルに変換せよ。

22. 練習問題解

前号の(17)にあげた練習問題の解の一例を示す。

i) 計画量の危険度の評価

NEW ALGOL-H に直すには(13)i) のプログラムの次の行を訂正すればよい。

① K:=SQRT(LN(1/P12));

② LX:=LN(X(I));

③ RVLX:=SQRT((SLXX-MLX*SLX)

④ H:=(KP+SQRT(KP12-(1-

⑤ PRINT('I', 1/P);

⑥ PRINT('B-5Z. 2DB/ ', EXP(

⑦ MLX+H+RVLX));

⑧ 削除

ii) 一元配置の分散分析

A NEW ALGOL-H への変換

⑬ PRINT('5B"MEAN");

⑪ PRINT('8B"RV");

⑫ PRINT('8B"N"/');

⑬ 削除

⑭ K5:=RV(I):=SQRT((SS

⑮ PRINT('2(B-5Z. 5DB)/ ', K2, K5);

⑯ PRINT('7B"AOV"TABLE"/');

⑰ 削除

⑱ 削除

⑲ PRINT('3SI" 7B"SS" 7B DF 7B MS

⑳ 7B"F"/');

㉑ 削除

㉒ *

㉓ *

㉔ *

㉕ PRINT('33I" B-4Z. ADB, B2ZDB,

㉖ 2(B-3Z. 3DB)/ ', SSA, NA, FA, FO);

㉗ 削除

㉘ *

㉙ *

㉚ *

㉛ PRINT('33I" B-4Z. ADB, B2ZDB,

㉜ B-3Z. 3DB)/ ', SSE, NE, FE);

㉝ 削除

㉞ *

㉟ *

㊱ PRINT('15(' - ')/');

㊲ 削除

㊳ PRINT('33I" B-4Z. ADB, B2ZDB/ ', SS0, S0);

㊴ 削除

㊵ *

㊶ *

㊷ *

もし先頭の宣言部に次の SPACE, CRLF, PRINTS, PRINTR, PRINTF, PRINTI について次のように procedure 宣言を追加しておけば(13)ii) のプログラムはそのままでよい。

procedure CRLF(N);

integer N;

begin

REPX(N);

PRINT('X/');

end;

```

procedure SPACE(N);
  integer N;
  begin
    REPX(N);
    PRINT ('XB');
  end;

procedure PRINTS(A);
  string A; PRINT(A);

procedure PRINTI(N,A);
  integer N,A;
  begin
    REPX(N-1);
    PRINT('B-XZDB',A);
  end;

procedure PRINTR(N,A);
  integer N; real A;
  begin
    REPX(N);
    PRINT('B-XD10-ZDB',A);
  end;

procedure PRINTF(S,A);
  real S,A;
  begin
    integer I,J;
    I:=FIX(S);
    J:=(S-I)*10;
    REPX(I-1,J);
    PRINT('B-XZD.XDB',A);
  end;

```

一番例な PRINTF について説明すると S に 4.3 が与えられているとき I は 4 となり J に 3 が入る。そこで小数点以上 3 ケタはゼロサプレスし、1 ケタはゼロサプレス（無効な 0 のときは印字しない）しないものとし、小数点の下は 3 ケタ印字しその後空白をつけるものである。

B. FACOM230 ALGOL への変換

表-1 変換表

旧 [(13)II] のプログラム	新
SPACE ①⑥⑦⑧⑨⑩⑪⑫⑬⑭⑮⑯⑰⑱⑲⑳㉑㉒㉓㉔㉕㉖㉗㉘㉙㉚㉛㉜㉝㉞㉟㊱㊲㊳㊴㊵㊶㊷㊸㊹㊺㊻㊼㊽㊾㊿	Space
PRINTS ①⑥⑦⑧⑨⑩⑪⑫⑬⑭⑮⑯⑰⑱⑲⑳㉑㉒㉓㉔㉕㉖㉗㉘㉙㉚㉛㉜㉝㉞㉟㊱㊲㊳㊴㊵㊶㊷㊸㊹㊺㊻㊼㊽㊾㊿	Printstring する
PRINTI ㉑㉒㉓㉔㉕	Printx としカッコ内は 3.A とあれば A, 3 に逆転
PRINTF ㉑㉒㉓㉔㉕㉖㉗㉘㉙㉚㉛㉜㉝㉞㉟㊱㊲㊳㊴㊵㊶㊷㊸㊹㊺㊻㊼㊽㊾㊿	Printfx としカッコ内は 3.2.A とあれば A.3.2 と逆転

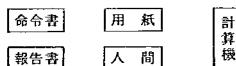
23. 電子計算機

人間の言葉に近いもので書いてやれば機械が自ら翻訳しそのまま実行する以上電子計算機の構造に関する知識は不必要であるといつてよい。

ただ、今後電子計算組織を導入しようとしておられる人々のために性能表の読み方を少し舐れておく、これらの機械的部分は、いわゆるハードウェア (hardware 金物) といわれるもので、ユーザーとしてはプログラム言語特に ALGOL, FORTRAN 等で書かれたプログラムを翻訳実行するコンパイラ、これらの作業を自動的に監視するモニタ等のソフトウェア (software 紙物) の方がより重要であるが計算組織の導入に当って最小限の知識をまとめて置く。

23.1 電子計算機の構成

われわれが計算指令を与えられて手回しあるいは電動計算機を用いて計算し、中間結果を用紙に書きとめなどして最終結果を出し、これを報告書にして提出する過程は次のようなものになる。



電子計算による実行過程もこれと同じことであって、模式図によって示す次のようになる。



ただ人間が行なう場合には命令書にすべての計算手続きが書いてある場合もあるし、文庫名を指定してそれに基づいてやれとか、お前の知っているあの方法でやれとか、場合によってはもっと省略されてよろしくやれといういい方もあり得るが、電子計算機の場合は、すべての手順をプログラムという形で教えてやらなければよろしくやるといふ場合にはゆかない。

ただし計算機にあらかじめやり方を教えておけば 2 度目にはあのやり方でこれとこれを変えたものを使用して、その結果を次の計算の一部に利用せよといったやり方である意味での辞書を計算機に蓄積させることができ、この意味では計算機にだんだん学習させておいて後で判断の基準を与えてよろしくやれといった指令を出すことは可能である。

23.2 入力装置 (input device)

紙テープまたはカードによって特定の位置の孔の有無を光電トランジスタを用いて判読させる。ALGOL は紙テープベースのものが多く、毎秒 200~1000 字の読み取

りを行なう。紙テープはカードに比してきわめて経済的であり、最近スプライサが開発されたので修正も容易である。

23.3 制御装置 (control unit) および演算装置 (arithmetic unit)

回路の素子としてはトランジスタ・ゲルマニウムダイオード・パラトロン等が多く用いられる。情報は1長語40ビット程度、1短語20ビット程度を単位として処理される。(1ビットは0か1かの2つのうち1つを選ぶ情報の最小単位)ICのものもある。

23.4 記憶装置 (memory)

直接使用される内部記憶装置と、一旦ひとまとめに内部記憶装置に移してから使用される外部記憶装置とがある。

磁気コア 径1ミリ以下のフェライトコアの磁束の方向によって01の状態を記憶させるもので呼び出し時間はマイクロ秒(μ S100万分の1秒)単位。

磁気ドラム 円筒の表面に磁性塗料を塗って毎分3000~18000回転させて表面の特定点の磁化の有無により01の状態を判別するもので呼び出し時間はミリ秒(ms1000分の1秒)単位であり、外部記憶装置として用いられる。

磁気テープ 録音テープのようなもので1本のテープに20万~100万くらい入るが任意の情報伝送に要する時間は長く秒単位

磁気ディスク レコードを想像していただければ良い。呼び出し時間は100ms単位であり容量は1億字位のものもある。

このほか磁性薄膜なども考えられている。なお記憶容量としてはK語(1024語)単位で数えられることが多い。

23.5 出力装置 (output device)

一般に1行120~135字毎分500~1000行印刷するラインプリンタが用いられる。

23.6 電子計算機の歴史

計算方式には計算尺で代表されるような類同的な(analogous)方式と、ソロバンで代表されるような計数的な(digital)方式とがあるが、アナログ方式はその精度が 10^{-2} ~ 10^{-3} 程度なので主として特殊目的に使用されている。

デジタル計算機は石や指で数えるやり方に始まり、1614年のネピアの骨木、1642年のパスカルの加算機を経て、1694年ライブニッツの計算機に至り、加減の反覆による乗除算が可能となった。

記憶、制御、入出力装置を持つ自動計算機の構想は1880年代のパベッジのAnalytical Engineに始まるが、これは完成するに至らなかった。以下年表式に電子計算機の歴史について触れてみる。

1880 Hollerith パンチカード会計機開発

今日のIBM

1907 Powers パンチカード会計機開発

今日のRR

1944 ハーバード大のAiken等 ASCC 完成、機械式で加算0.3秒

1946 ペンシルバニア大EckertおよびMauchlyが電子部品を用いたENIAC完成、加算0.2ms

1951 RR社Univac I事務用機完成

1953 IBM 701

1956 電気試験所 トランジスタ計算機完成

23.7 電子計算機一覧

名 称	FACOM 230-30	FACOM 230-50	FACOM 230-60	FACOM 270-30	HITAC 5020	HITAC 5020E	NEAC 2200-400	NEAC 2200-500
構 造	富士通	富士通	富士通	富士通	日立	日立	日電	日電
主記憶容量 (1000語)	4~32	8~65	32~262	8~65	8~65	16~262	16~262	65~524
1語(ビット)	36	36	36	18	32	32	9	9
加 算 (マイクロ秒)	58.3	4.4	1.15	1.9	8	0.76	22	6
掛 算 (マイクロ秒)	800	25.3	4.4	9.6	24	2	200	9
月レンタル料金 (万円)		300 ~1500		150 ~800			250 ~1500	400 ~2000
ソフトウェア A: ALGOL F: FOR TRAN C: COBOL	A. F. C	A. F. C	A. F	A. F	A. F	A. F	C. F	C. F

名 称	OKITAC 5090-H	OKIMIN ITAC 7000	TOSBAC 3400-30	TOSBAC 3400-40	Burroughs B5500	G D C 3600	I B M 360-75	UNIVAC 1108II
製 造	沖	沖	東 芝	東 芝	Burroughs	Control Data	I B M	R R
主記憶容量 (1000語)	8~16	4~65	4~32	8~262	4~32	32~262	262~1048	65~262
1語(ビット)	42	24	24	24	48	48	8	36
加 算 (マイクロ秒)	30	5.6	14.2	4.5	2	2.07	0.8	0.75
乗 算 (マイクロ秒)	250	44.8	49	13.4	44	6.5	25	39
月レンタル料金 (万円)	230 ~400	200~400	100~700	250 ~1000	600 ~6000	1500 ~4000	1800 ~6億	1800 ~6億
ソフトウェア A: ALGOL F: FOR TRAN C: COBOL	A. F. C	F	A. F. C	A. F	A. F. C	A. F. C	C. F	A. F. C

24. 重力式コンクリート擁壁自動設計のプログラム

土研の技術管理室においては電子計算機の導入とともに普通に現われる土木構造物の設計手順を自動化し、設計書の上では図面番号の称呼のみで数量に至るまで(ただし延長のあるものは単位あたり)拾い出せるような標準設計方式を目論み、すでに[I] 側溝類・暗きょ類・重力式擁壁・突げた式擁壁・石積工など、[II] 扶壁式擁壁・二連ボックスカルバート、[IV] 橋台等について全日本建設技術協会より出版され、機りょう上部工、橋脚について作業を進めている。

これらには与えられた荷重条件・主要寸法に対して応力計算を行ない、不足ならば鉄筋や断面寸法を変更して最適設計に到着くまで計算を実行し終われば材料の寸法数量・型わく等の設計に必要な情報を印刷してくるものである。われわれはこれに基づいて製図を行なったがその段階で多くのミスをおかす危険があるので図化も計算機に行なわせることとした。以下重力擁壁設計に必要な手順を日本語で列記し、次章にこれに対応するALGOLプログラムを紹介しよう。

プログラムをご紹介します。

1) 設計条件

高さ H 、前面コウ配 N (H に対する比率)、土圧係数 C 、土の重量 W をデータテープより読み込み、図-3に与えた変数およびコンクリート重量 2350 kg/m^3 、サーチャージ20%荷重を土の厚さに換算し 0.75 m 土の重量 1800 kg/m^3 、コンクリートと土の摩擦係数 0.5 を用い、

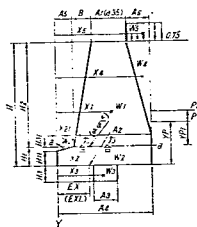


図 - 3

1. 合力の作用点が中央 $1/3$ をこえたとき
 2. 擁壁内部で 1 kg/cm^2 以上の引張応力が生じたとき
 3. 滑動安全率が2以下のとき
 4. 転倒安全率が1.5以下のとき
 5. キイ部のせん断応力が 5 kg/cm^2 をこえたとき
- 自動的に断面を 5 cm ずつ増して1~5の条件をすべて満足するまで計算をくり返し、設計に必要な数値を印字させる。土圧係数は 0.28 、 0.33 、 0.4 の3種とする。

ii) 安定計算

a 重量算定

$$A7 = A1 + A2 \quad (\text{頂幅と底幅の和})$$

$$W1 = A7 \cdot H2 \cdot 2350 / 2$$

$$W2 = H1 \cdot A4 \cdot 2350$$

$$W21 = H31 \cdot A5 \cdot 2350 / 2$$

$$W3 = A3 \cdot H3 \cdot 2350$$

$$W4 = A6 \cdot H2 \cdot 2350 / 2$$

$$W5 = A6 \cdot 0.75 \cdot 1800$$

$$W = W1 + W2 + W3 + W4 + W5 - W21$$

b 擁壁前面点Cより各重心までの距離

Y 輪関係

$$YP = H(H/3 + 0.75) / (3(H/2 + 0.75))$$

$$Y = (2 \cdot A1 + A2) H2 / (3 \cdot A7)$$

$$YP1 = H2(H/2 + 3 \cdot 0.75) / (3(H/2 + 0.75))$$

X 輪関係

$$X1 = (A1^2 + A1 \cdot A2 + A2^2) / (3 \cdot A7) + B \cdot Y / H2 + A5$$

$$X2 = A4 / 2$$

$$X21 = A5 / 3$$

$$X3 = A4 / 2$$

$$X4 = A4 - A6 / 3$$

$$X5 = A4 - A6 / 2$$

c 土 圧

$$P = 1800 \cdot \frac{1}{2} \cdot C \cdot ((H/2 + 0.75)^2 - 0.75^2)$$

$$P1 = 1800 \cdot \frac{1}{2} \cdot C \cdot ((H/2 + 0.75)^2 - 0.75^2)$$

d モーメント

$$MOX = (W1 \cdot X1) + (W2 \cdot X2) + (W3 \cdot X3) + (W4 \cdot X4) +$$

$$(W5 \cdot X5) - (W21 \cdot X21)$$

$$MOXL = MOX - W3 \cdot X3$$

$$MOP = P \cdot YP$$

$$MOP1 = P1 \cdot YP1$$

e 合力の作用点

$$EX = (MOX - MOP) / W \quad (EX > \frac{A4}{3})$$

$$EXL = (MOXL - MOP) / (W - W3) \quad (EXL > \frac{A4}{3})$$

f 応力度算定

$$X11 = X1 - A5$$

$$X41 = X4 - A5$$

$$X51 = X5 - A5$$

$$MOX1 = W1 \cdot X11 + W4 \cdot X41 + W5 \cdot X51$$

$$WT = W1 + W4 + W5$$

$$EX1 = (MOX1 - MOP1) / WT$$

$$E1 = A2 / 2 - EX1$$

$$QA = WT / A2 \cdot (1 + 6 \cdot E1 / A2)$$

$$QB = WT / A2 \cdot (1 - 6 \cdot E1 / A2)$$

$$QB > -10000 \quad (\text{引張 } 10000 \text{ kg/m}^2 \text{ 以下})$$

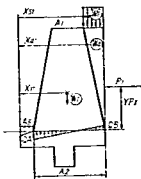


図 - 4

g 地盤反力

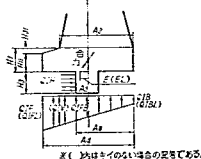


図 - 5

$$WL=W-W3$$

$$E=AA/2-EX$$

$$EL=AA/2-EXL$$

$$QIF=W/A4*(1+6*E/A4)$$

$$QIB=W/A4*(1-6*E/A4)$$

$$QIEL=WL/A4*(1+6*EL/A4)$$

$$QIBL=WL/A4*(1-6*EL/A4)$$

h キイに変わる応力度

$$QIFB=QIB+(QIF-QIB)*A8/A4$$

$$QIV=(QIE+QIFB)/2$$

$$QIH=QIV+C1$$

ただし

$$C1 = \frac{1 + \sin \theta}{1 - \sin \theta}$$

であるがCに対応して次の内部摩擦角 θ を採用した。

C	θ	C1
0.28	34°	3.5372
0.33	30°40'	3.0
0.40	26°	2.5672

$$SS=QIH*H3/A3$$

(SS $\geq$ 50000) セン断応力5kg/cm²以下

i 安全率の算定

$$TEIKO=QIH*H3+(QIFB+QIB)*A8*0.5/2$$

$$SLIDE=TEIKO/P \quad (SLDE \geq 2)$$

$$SLIDE=WL*0.5/P$$

$$TENTO=MOX/(MOX+QIH*H3^2/2)$$

$$(TENTO \geq 1.5)$$

$$TENTOL=MOX/AOP$$

ii) 数量計算

a コンクリート

$$V1=A7*H2/2+H1*A4-A5*H3/2$$

$$V2=V1+H3*A3$$

b 型わく

$$KATA1=H1+H11+\sqrt{(E^2+H2^2)}+\sqrt{(A6^2+H2^2)}$$

$$KATA2=KATA1+2*H3$$

以上の手続をALGOL で書くと次章のようになる。

25. 重力擁壁自動設計のプログラム

入力データは 高さHm, 前面こう配N(1分こう配なら0.1), 土圧係数C(0.28, 0.33, 0.4の3種とする)とし, NEW ALGOL-II によって記述する。

プリントは設計図作成に必要と思われる数値を××=……という形で印字させる。

また初期値は A5=0.1H A4=0.5H A3=0.3 A1=0.35 とする。

begin

```
real H,C,C1,P,P1,A1,A3,A4,A5,H1,H2,A2,
A9,A8,A6,A7,H3,W1,W2,W3,W4,W5,W,WL,
Y,YP,YP1,X1,X2,X3,X4,X5,MOX,MOXL,
MOP,MOP1,EX,EXL,T,X11,X41,SLIDEL,
X51,MOX1,EX1,E,EL,QIF,QIB,QIFL,QIBL,
QIBF,QIV,QIH,TEIKO,TEIKOL,SLIDE,
TENTO,V1,V2,KATA1,KATA2,H3,EXL,H31,
W21,X21,TENTOL,WT,E1,QA,QB,SS,H11;
```

procedure PR(L,N,A);

value N,A;

integer N; real A; string L;

begin

PRINT(L);

PRINT("=");

REFX(N);

PRINT(' -XD01-ZDB',A);

end;

READ(H,N,C);

C1:=if C=0.28 then 3.5372 else

if C=0.33 then 3.0 else 2.5672;

P:=900*C*((H+0.75)↑2-0.75↑2);

A5:=0.1*H; A1:=0.35;

Z:A4:=0.5*H; A3:=0.3;

if H $\leq$ 3 then

begin H1:=0.35; H31:=0.05; end

else

begin H1:=0.6; H31:=0.1; end;

H2:=H-H1;

P1:=900*C*((H2+0.75)↑2-0.75↑2);

```

Z1:A2:=A4-A5;A9:=-A2-A1;
  A8:=(A4+A3)/2;
  T:=N*H2;
  A6:=-A2-A1-T;
  A7:=-A1+A2;
  H3:=0.3; H11:=-H1-H31;
  W1:=A7*H2+2350/2;
  W2:=-H1*A4+2350;
  W21:=-A5*H31+2350/2;
Z11:W3:=-H3*A3+2350;
  W4:=-A6*H2+900; W5:=-A6*2350/2;
  W:=-W1+W2+W3+W4+W5-W21;
  WL:=-W-W3;
  Y:=(2*A1+A2)*H2/(3*A7);
  YP:=-H*(H+225)/(3*(H+1.5));
  YP1:=-H2*(H2+225)/(3*(H+1.5));
  X1:=(A1+2+A1*A2+A2+2)/
    (3*A7)+(T*Y)/H2+A5;
  X2:=X3:=A4/2; X4:=-A4-A6/3;
  X5:=A4-A6/2; X21:=-A5/3;
  MOX:=-W1*X1+W2*X2+W3*X3
    +W4*X4+W5*X5-W21*X21;
  MOXL:=MOX-W3*X3;
  MOP:=-P*YP; MOP1:=-P1*YP1;
  EX:=(MOX-MOP)/W;
  EXL:=(MOXL-MOP)/WL;
  if EX<A4/3 then
    begin A4:=-A4+0.05; go to Z1; end;
  X11:=X1-A5; X41:=-X4-A5;
  X51:=-X5-A5;
  MOX1:=-W1*X11+W4*X41+W5*X51;
  WT:=-W1+W4+W5;
  EX1:=(MOX1-MOP1)/WT;
  E1:=A2/2-EX1;
  QA:=(WT/A2*(1+6*E1/A2))/10000;
  QB:=(WT/A2*(1-6*E1/A2))/10000;
  if QB<1 then
    begin A4:=-A4+0.05; go to Z1; end;
  E:=-A4/2-EX; EL:=-A4/2-EXL;
  QIF:=W/A4*(1+6*E/A4);
  QIFL:=-WL/A4*(1+6*EL/A4);
  QIB:=W/A4*(1-6*E/A4);
  QIBL:=-WL/A4*(1-6*EL/A4);
  QIFB:=QIB+(QIF-QIB)*A8/A4;
  QIV:=(QIF+QIFB)/2;
  QIH:=QIV*C1;
  TEIKO:=-QIH*H3+(QIFB+QIB)*A8*0.25;
  SS:=-QIH*H3/A3/10000;

```

```

TEIKOL:=2*P-W1*0.5;
SLIDE:=-TEIKO/P;
SLIDEL:=WL*0.5/P;
if SLIDE>2 then
  begin H3=H3+0.05;
    go to Z11;
  end;
if SS>5 then
  begin A3:=-A3+0.05; go to Z1; end;
TENTO:=-MOX/(MOP+QIH*H3+2/2);
TENTOL:=-MOXL/MOP;
if TENTO<1.5 then
  begin A4:=-A4+0.05;
    go to Z1;
  end;
V1:=-A7*H2/2+H1*A4-A5*H31/2;
V2:=-V1+H3*A3;
KATA1:=-H1+H11+SQRT(T+2+H2+2);
  +SQRT(A6+2+H2+2);
KATA2:=-KATA1+2*H3;
PRINT("20B");
PRINT("*****DESIGN##OF##
  GRAVITY##TYPE##RETAINING##WALL
  *****/'");
PR("H",2,H);PR("C",3,C);
PRINT('3/');
PR("A1",4,A1); PR("A2",4,A2);
PR("A3",4,A3); PR("A4",4,A4);
PR("A5",4,A5); PR("A6",3,A6);
PR("A8",3,A8); PRINT('');
PR("T",4,T); PR("H1",4,H1);
PR("H11",4,H11); PR("H2",4,H2);
PR("H3",4,H3);
PR("A9",3,(A4-A3)/2);
PRINT('2/');
PR("CONCRETE1",4,V1);
PR("CONCRETE2",4,V2);
PR("KATSVAKU1",5,KATA1);
PR("KATSVAKU2",5,KATA2);
PRINT('2/');
PRINT('40B'); PRINT('*****STRESS
  *****/'');
PR("QA",3,QA); PR("QB",3,QB);
PR("QIF",4,QIF/1000);
PR("QIB",3,QIB/1000);
PR("QIFL",4,QIFL/1000);
PR("QIBL",3,QIBL/1000);
PR("KEY",2,SS);

```

```

PRINT('2/');
PRINT('38B');
PRINT('****SAFETY##FACTOR****/');
PR('TENTO',3,TENTO);
PR('TENTOL',3,TENTOL);
PR('SLIDE#',3,SLIDE);
PR('**SLIDEL',3,SLIDEL);
PRINT('/');
PR('GORYOKUX',3,EX);
PR('GORYOKUY',3,EXL);
PR('WEIGHT',5,W);
PR('TEIKOL',5,TEIKOL);
PRINT('12/');
S:
PRINT('*****QVARI*****');
PRINT('15/');
end;

```

26. 自動図化

土木研究所で作製した自動設計結果を、図化機の描き出したそのままの形で御紹介して置く。これは基本的な設計条件のみを示して最適設計を行ない、図面も原点のみを示されて、縮尺を自から適当に選定して描き出したもので、横断図などという字も横の字の左下の座標、字の方向角および字の大きさを示されて、後はあらかじめ格納してあるODANZUと言う手続きを呼び出して使用したものであるし、図中の数字も自から計算した結果を書いているものである。(図-6)

27. これまでのまとめ

以上4号を通じて ALGOL による電子計算機との対話方法についてお話ししてきた。一番最初にもべたように ALGOL は32の単語30の記号しか用いてないのに文法体系がすっきりと組み立てられているので学ぶのもきわめて簡単で、お話しした場合でも実習を含めて2日もあれば一人前のプログラマーとして自分の問題を書き下すことができる。

それだけでなく最近の本や文献などに、手順をALGOLで説明したもの(文献8)さえ現われてきており、ALGOLの読み書きができることは少なくとも技術者にとってはあたり前のことになりつつある。

処理プログラムは単位にこの言葉で書かれたプログラムを機械語に翻訳するという機能のみに止らず、指定の優先順位に従って優先度の高いプログラムから逐次自動的に処理を行なつてゆくモニタ(監視プログラム)を付加することにより、

土木研究所のOKITAC 5090 H用のNEW ALGOL-Hを用いて第3次優先のPTR(光電テプリーダ)から入って処理中のプログラムに第2次優先のPTRから入ったプログラムが割り込み、これにさらに第1次優先のPTRから入ったプログラムの文法チェックが割り込んで逐次割り込みと同時にそのとき現在の情報を凍結し、割り込んだものの処理が終わるとただちに以前の状態にもどって処理を続けることができる。

このほかattachという特殊な区切り記号によって今後有用と思われる標準手続を翻訳したものを磁気テープに格納し、以後はexternal宣言でその手続の名前だけを呼び出してやれば、自由にその手続が使用できるとい機能などを持っている。

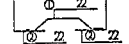
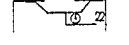
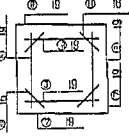
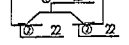
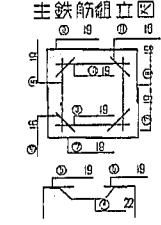
このようにALGOLは問題を持つ者が直接機械に話しかけやすく、かつできるだけその労力を省くように構成されており、自分のところに計算組織が導入されないまでも各地の計算センターを利用するのにプログラム持ち込みによって計算機を格安に時間借り(分借り?)する途をひらいてくれる。事実計算センターにプログラミングを依頼した場合、プログラムに題意を説明するのに骨折るほか機械語1ステップあたり100~150円のプログラム料を必要とし、ALGOLプログラム1行が平均10ステップくらいに当ることもあるので自からプログラムを作るときわめて計算が安上りになることになる。

また入出力関係の文と多少の文法制限事項の差異を除けばALGOLを受け入れるどの計算機にも適用できるためたとえば土木研究所のNEW ALGOL-Hで書いたプログラムテープは、記憶容量や計算速度の関係でエラーチェックや組だけの計算を完了した段階で約40倍くらい計算速度が早いCDC 3600に対しほんの一部の修正のみで掛けることができ、すでにこの変換のプログラムもできて、土研のテープそのままでCDCに掛けることが可能となっている。

このようにわれわれが自分の問題に対する手順を日本語で書くと同じくらしい気易さでALGOLによって書き下すことが可能であり、また人の書いた手順は日本語でかかれたものよりその構造が分りやすくかつ厳密であり、一度作っておけば後からその問題よりひとまわり大きな問題の中で、それを辞書の中から引き出すようなやり方で利用できるなどALGOLを修得することの利便はいろいろ考えられるので、是非本文および後にあげた文献などに親しまれて自から試してみることをおすすめしたい。

28. ALGOL マニュアル

今まで27節にわたってアルゴルによるプログラミングの方法をのべてきたが、この内容で行なつた講習会の席で、入門として読み下すには適当だが、実際のプログラミ



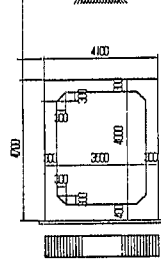
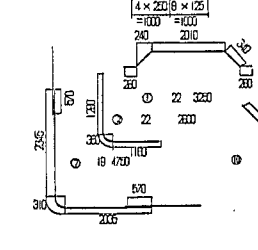
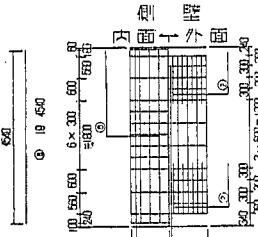
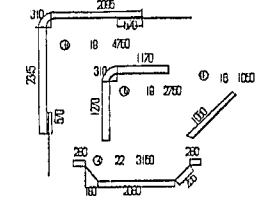
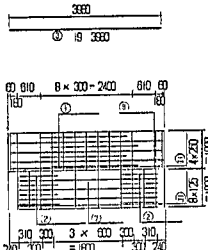
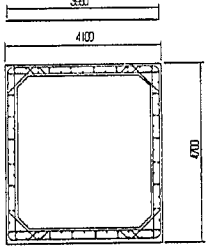
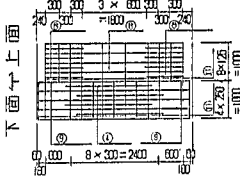
主鉄筋方法

材料	規格	数量	単位	重量	備注
①	13	210	kg	260	
②	13	210	kg	260	
③	13	210	kg	260	

下面上面

断面図 08/1

頂版



- 1 設計条件
 1 冠 初 止
 2 冠 止 点
 3 部 止 点
 4 止 点 後
 2 止 点 後

材 料 表

材料	規格	数量	単位	重量	備注
①	13	210	kg	260	
②	13	210	kg	260	
③	13	210	kg	260	
④	13	210	kg	260	
⑤	13	210	kg	260	
⑥	13	210	kg	260	
⑦	13	210	kg	260	
⑧	13	210	kg	260	
⑨	13	210	kg	260	
⑩	13	210	kg	260	
⑪	13	210	kg	260	
⑫	13	210	kg	260	
⑬	13	210	kg	260	
⑭	13	210	kg	260	
⑮	13	210	kg	260	
⑯	13	210	kg	260	
⑰	13	210	kg	260	
⑱	13	210	kg	260	
⑲	13	210	kg	260	
⑳	13	210	kg	260	
㉑	13	210	kg	260	
㉒	13	210	kg	260	
㉓	13	210	kg	260	
㉔	13	210	kg	260	
㉕	13	210	kg	260	
㉖	13	210	kg	260	
㉗	13	210	kg	260	
㉘	13	210	kg	260	
㉙	13	210	kg	260	
㉚	13	210	kg	260	
㉛	13	210	kg	260	
㉜	13	210	kg	260	
㉝	13	210	kg	260	
㉞	13	210	kg	260	
㉟	13	210	kg	260	
㊱	13	210	kg	260	
㊲	13	210	kg	260	
㊳	13	210	kg	260	
㊴	13	210	kg	260	
㊵	13	210	kg	260	
㊶	13	210	kg	260	
㊷	13	210	kg	260	
㊸	13	210	kg	260	
㊹	13	210	kg	260	
㊺	13	210	kg	260	
㊻	13	210	kg	260	
㊼	13	210	kg	260	
㊽	13	210	kg	260	
㊾	13	210	kg	260	
㊿	13	210	kg	260	

ング中にときどき参照するマニュアルがほしいという希望が多かったので、4回で打切る予定であったものを延長してマニュアルを追加する。

28.1 基本記号, 名前, 数値, 記号列

1) 基本記号

基本 記号	1. 文字	AからZ, aからzまでの52文字(大文字に限るものも多い)
	2. 数字	0から9までの10個
	3. 論理値	true と false の2個(大文字とするものもある)
	4. 区切り記号	1. 作用素 1. 算術作用素 $+ - \times \div$ ($\times$の代りに$*$とするものも多い) 2. 比較作用素 $< \leq = \geq >$ 3. 論理作用素 $\neg \wedge \vee \equiv$ 4. 順序作用素 $\left(\begin{array}{l} \text{go to if} \\ \text{then else} \\ \text{for do} \end{array} \right)$
	2. 区切り	$\left(\begin{array}{l} \cdot \cdot \cdot 10 : ; \cdot \cdot \cdot \\ \text{step until while} \\ \text{comment } _ \end{array} \right)$
	3. 括弧類	$\left(\begin{array}{l} () [] \\ \text{begin end } \cdot \cdot \cdot \end{array} \right)$
	4. 宣言詞	Boolean integer real own array switch procedure
	5. 特別規制詞	(string label value)

2) 略号

$\tilde{A}E$ (算術式)	$\tilde{A}P$ (実パラメタ)
$\tilde{B}E$ (論理式)	$\tilde{D}$ (宣言)
$\tilde{D}E$ (行先式)	$\tilde{F}P$ (仮パラメタ)
$\tilde{F}S$ (繰り返し文)	$\tilde{Y}$ (名前)
$\tilde{L}$ (名札)	$\tilde{Z}$ (文)
$\tilde{S}A$ (単純算術式)	$\tilde{S}B$ (単純論理式)
$\tilde{S}D$ (単純行先式)	$\tilde{S}V$ (単純変数)
$\tilde{S}E$ (添字式)	$\tilde{U}S$ (無条件文)
$\tilde{U}I$ (符号のない整数)	$\tilde{V}$ (変数)

またパンチャの誤読を避けるために

I I 7 0 0 D 5 S V V C C 2 Z

の区別をはっきりさせる必要がある。

3) 名前

名前は文字を先頭としその後必要によって文字または数字をつけたもので、変数、手続き、関数呼び出し、名札、スイッチなどに付けられる(先頭の5~8文字のみ判別するものが多い)。

(可) A AP SUM B36C MATRIX

(不可) 12A A-B

4) 数

数	実数 (real) …… 少なくとも -10 の1つを含んだ数値
	整数 (integer) …… 10 のいずれをも含みぬ数値

$\cdot$ や 10 で始まるのはよいが終わることは認めない。指数部は integer に限る。

(可) $-5 \cdot 4 \cdot 46 \cdot 10^{-2} \cdot 3 \cdot 10^{-2} \cdot -4 \cdot 6 \cdot 10 \cdot 5$

(不可) 3. $4 \cdot 10 \cdot 3 \cdot 4 \cdot 10 \cdot 9 \cdot 5 \cdot 10 \cdot 2 \cdot 5$

($5 \cdot 10 \cdot 12 \cdot 5$ はある計算を要し1つの数値ではない)

5) 記号列 string

記号列は基本記号の任意のなびを $'$ でつつんだものである。

(例) 'PQR' 'This is a string'

28.2 式 (expression)

式とは結果として1つの算術値、論理値または名札に帰着する表現である。

式	算術式 ($\tilde{A}E$) 結果として1つの算術値に帰着する表現
	論理式 ($\tilde{B}E$) 結果として1つの論理値に帰着する表現
	行先式 ($\tilde{D}E$) 結果として1つの名札に帰着する表現

1) 変数

変数	単純変数 (例) R PRA L3
	添字付変数 (例) D(I) RV(S,T+K)
	M1 [if K=L then P else 2]

添字式が real のときは自動的に4捨5入される。添字変数はその所在をつきとめるのに時間がかかるので、式中で何回も現われるときはいったん他の単純変数に受けて使用するほうがよい。

変数はすべて宣言されていなければならない。

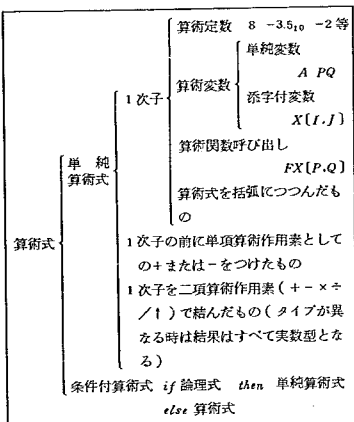
2) 概算関数, 換算関数

abs sign sin cos sqrt
arctan ln exp entier

(大文字となるものもある)

- i absは()の中と同じタイプの絶対値。
- ii sign は integer の結果で()の中が正のとき+1, 0のとき0, 負のとき-1となるが他はすべてrealの結果となる。
- iii entierはそれを越さぬ最大の整数をとる。したがって entier (8.6)は8であるが entier (-8.6)は-9となる。

3) 算術式 (A/E)

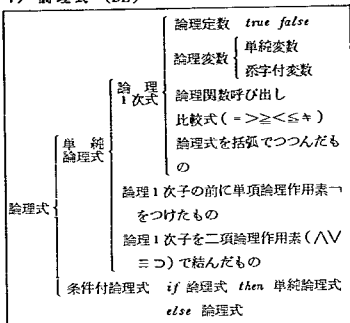


- i 演算の優先順位は次のようになる。

- | |
|----------|
| 1. () |
| 2. ↑ |
| 3. * / ÷ |
| 4. + - |

- ii 算術作用素の両側のタイプが異なるときはその結果は real となる。ただし+の両側は integer に限り5 ÷ 2 は2となる。
- iii 条件式は if then else がそろっている必要があり, 条件式を作用素で結んだり then の後で用いるときは ()で包んで単純式に直さなければならない。
- iv 算術作用素を/等と連続して用いてはならない。

4) 論理式 (B/E)



演算の優先順位は次のようになる

- | |
|----------|
| 1. () |
| 2. 算術作用素 |
| 3. 比較作用素 |
| 4. ¬ |
| 5. ∧ |
| 6. ∨ |
| 7. ≡ |
| 8. ≡ |

- ii ¬∧∨≡は次の結果をもたらす

¬A………Aが true のとき false, Aが false のとき true

A ∧ B………A, Bともに true のときのみ true

A ∨ B………A, Bともに false のときのみ false

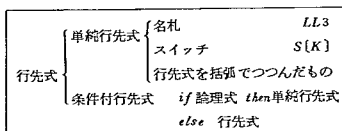
A ≡ B………A, Bともに true またはともに false のときのみ true

A ⊃ B………A true かつ B false のときのみ false

- iii 論理作用素は∧などと連続して用いてはならない。したがって-1 > x > 1 は-1 > x ∧ x > 1 としなければならない。

- iv 条件式を演算記号で結んで用いるときや then の後で用いるときは丸括弧に包んで単純式とする。

5) 行先式 (D/E)



スイッチの添字式がrealのときは4捨5入される。

スイッチとして宣言された範囲をこえるときは空文扱いとなる(エラーとするものも多い)。

28.3 節 (clause)

条件節 if $\bar{B}\bar{E}$ then
繰り返し節 for $\bar{V} := < \text{for list} > \text{do}$

いずれもそのままでも条件節は条件式や条件文の一部、
繰り返し節は繰り返し文の一部として用いられる。

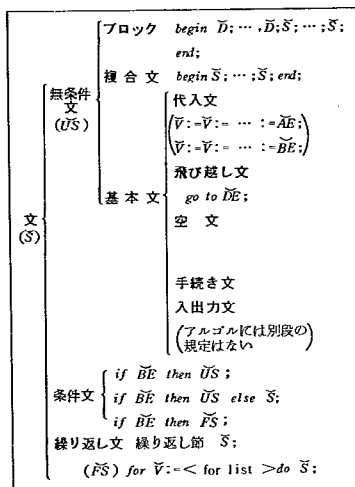
< for list >としては

1. $\bar{A}\bar{E}$
2. $\bar{A}\bar{E}$ step $\bar{A}\bar{E}$ until $\bar{A}\bar{E}$
3. $\bar{A}\bar{E}$ while $\bar{B}\bar{E}$
4. 1~3 をカンマでつないだもの

がありたとえ

```
for I:=1 step 1 until 20,22 step 2
  until 50,70,100,I*2 while I<10000 do
    で、次の文を I の 1,2,3,……,20,22,24,26,……,50,
    70,100,200,400,800,……,6400
    について実行する。
```

28.4 文 (statement)



1) ブロック、複合文

- i ブロックと複合文の差違は宣言の有無だけであるが、
名札はその場において宣言されたものとみなされる。
- ii 宣言はそれを含む最も小さいブロックについてしか意味を持たず、その外側において宣言された名前には内側で
同じ名前が宣言されていない限り有効である。同じ名前
が内側でも宣言されているときは、内側では内側で宣言

された意味のほうが優先する。

- iii 名札についてもブロック外に飛び出すことは可能であ
るが内側には begin から入っていないかなければならない。
- iv セミコロン宣言や文の区切りとして使用され end の
前では不用であるが、つい end の後に不用であるか
のように錯覚することが多いので end の前には空白があ
るものと解釈して常にセミコロンを入れるようにしたほ
うがよい。もちろん end の後は else の直前の場合を
除いてセミコロンが必要である。

2) 代入文

左右辺のタイプが異なるときは自動的に変換され特に左
辺 integer の場合は4倍5入が行なわれる。また左辺が
何重かになった多重代入文が許されるが左辺の中では同じ
タイプでなければならない。

3) 手続き文

宣言された手続きの内容はその名前の後に対応する実バ
ラメタ ($\bar{A}\bar{P}$) を書いた手続き文によって実行される。

4) 条件文

then の後は無条件文または繰り返し文に限る。したが
って条件文が then の後にくるときは begin end でくつ
て無条件文に直す必要がある。

5) 繰り返し文

<繰り返し節> $\bar{S};$

つまり

for $\bar{V} := < \text{for list} > \text{do } \bar{S};$

で do の効力は次の1つの文だけにしか及ばない。したが
っていくつかの文についてくり返したいときは begin
end でくくって1つの文に直す必要がある。

until の後は2進変換の都合で実行しない場合もあるの
で、きざみの $1/4 \sim 1/5$ を加え (負のきざみなら引く)
ておくほうがよい。

繰り返し文の途中で外にとび出すときは、コントロール
変数はその値を保持しているが、< for list > について完
全に終了したときは、コントロール変数がどのような値に
なっているか分らない。

6) コメント

begin および、の直後に comment と書けば次の、まで
何を書いても機械は無視する。また end の後は次の end
else または、までの間は何を書いても機械は無視する。

28.5 入出力文

改訂アルゴリズム 60 には入出力に関する何の規定もないの
で各コンパイラの規定を用いるより仕方がない。一般には
入力

READ(A, B, ……); Read(A, B, ……); 等が多いが
Readreal(A); などと1つに限るものもある。出力は

PRINT ('フォーマット' , リスト) ; PRINT
(n , A) ; Printreal (A , n) ; などまちまちである。

ただしフォーマットの中の記号としては ISO の勧告で
B (ブランク) C (カンマ) D (数字) V (印字な
しの小数点) Z (数字ただしゼロサプレス) / (行お
くり) などを共通に用いるものが多い。

28.6 宣言 declaration

宣言	型宣言	real integer Boolean
	配列宣言	(real) array integer array Boolean array
	手続き の宣言	手続き procedure 関数 (real procedure integer procedure Boolean procedure)
	スイッチ宣言	switch

型	定数	単		複		関		式
		宣言	規制	宣言	規制	宣言	規制	
算術 実数	-4.8 ₁₀ -5等	real	real	(real)array	(real)array	real procedure	real procedure	算術式
	8 等	integer	integer	integer array	integer array	integer procedure	integer procedure	
論理	true false	Boolean	Boolean	Boolean array	Boolean array	Boolean procedure	Boolean procedure	論理式
文字	'PQ5' 等		string					(文字式)
先行	L5 等		label	switch	switch			行先式

1) 型宣言

宣言詞の後に単変数をカンマで区切って記入し一段落でセミコロン。

2) 配列宣言

宣言詞の後に名前を書き [] の中に添字の限界を行いついて下限 : 上限 , 列について下限 : 上限の順で書く。

何次元であってもこのように , でつないで書けばよい。また同じ限界なら名前を , でつないで並べてもよい。添字の限界には式を用いてよいが整数定数に限るものもある。また (下限) ≤ (上限) でなければならない。

単に array と書けば real array とみなされる。

(例) array A , B [1 : 10] , PQ [1 : 5 , -8 : 4] ;

Boolean array CR [0 : 4] ;

3) スイッチ宣言

一種の名札の配列であって ,

switch 名前 : = 行先式の並び ;

スイッチの要素は行先式であるから , また他のスイッチであってもよい。

使用は go to SW [I] ; と飛び越し文などでその名前のスイッチの何番目の要素であるかを指定する形で行なう。

(例) switch Q : = L1 , L2 , if A = B then D
else LD , RS , S [I] ;

この場合は L1 , L2 , D , LD , RS は名札 , S [I] は他のスイッチリストの要素である。

4) 手続きの宣言

i 手続宣言は

procedure 手続名 < F P 部 > ;
< 値 の 部 >
< 規 制 部 >
< 手続本体 > ;

の形をとる。

ii 関数宣言は 1 個関数の場合に , real procedure などとその答のタイプを前につけると同時に手続きの本体中で自分自身の名前に対し答を代入する文が必要である。また手続きは手続き文として使用されるのに対して , 関数呼び出しは式の中で使用される。

iii 宣言中仮パラメタ部は (F P , F P ……) の形をとるがない場合はこの部分を省略する。値の部は F P 中値を外から値で貫ってくるものについて指定を行なう部分であって , [value 名前 , …… , 名前 ;

このようにしたほうがくり返し現れる変数については演算時間が早くなる。しかし value 指定しなくても差し支えはないわけで , このときは値の部は省略される。

value とはその実パラメタとなっている変数の値を変更しないという意志表示である。 (array の value 指定を禁止するものもある) 。

iv 規制部は F P にあげられたパラメタがどんなものであるかを規制する部分である (value 指定されなかったパラメタのうち integer のみ規制を必要とするもの , value 指定されたものを含んですべてのパラメタは規制

を必要とするものなど機種によっていろいろある)。
 規制詞としては *string, label, real, integer, Boolean, array, real array, integer array, Boolean array, switch, procedure, real procedure, integer procedure, Boolean procedure* がある。

仮パラメタと実パラメタの対応

仮パラメタ	規制詞	実パラメタ
算術単純変数	<i>real, integer</i>	算 術 式
論理単純変数	<i>Boolean</i>	論 理 式
配 列 名	<i>array, <タイプ>array</i>	配 列 名
関 数 名	<i><タイプ>procedure</i>	関 数 名
手 続 名	<i>procedure</i>	手 続 名
スイッチ名	<i>switch</i>	スイッチ名
行先変数名	<i>label</i>	行 先 式
文字変数名	<i>string</i>	記 号 列

v 機械語等との接続を可能にするため code という宣言詞もあるがコンパイラによってはこれを禁止している。

vi 手続きの実行は、手続き名の後に仮パラメタのある場合はそれに対応する位置に実パラメタを書きこんだ手続き文によって行なう。

関数呼び出しも同様実パラメタを書き込み、式の中で使用する。

28.7 データ

データは read 文に対応して必要な情報を並べて用意する。コンパイラによりカンマで区切るものなどいろいろある。

29. 改訂 ALGOL 60 の構文法

改訂アルゴル 60 の文法は Backus notation といわれる厳密ではあるがいささかとりつきにくい形で記述されている。そのため ALGOL は難解だなどといわれたが実はいままですら説明したようにきわめてやさしくかつ厳密な文章を書き下すことができ、問題の解き方を日本語で書き下すのとまったく同じ努力でより厳密な内容を盛り込むことができる言葉である。

しかし文法ギリギリいっぱいのもので利用しようとする向きにはどうしても厳密な文法が必要になるので意味の説明は前のマニュアルにゆずって構文法のみを述べておく。

29.1 一般的説明

以下の文章に次の記号を用いる。

記号	意 味
::=	(左辺)とは(右辺)のようなものである
	または
ゴシック	ALGOL の単語
<...>	……というもの (ALGOL 外の言葉)

さっそくこれを用いて根本的な定義を一つ与えておく

<空> ::=

つまり空とは何も書いてないことを意味する。また略号としてはマニュアルにあげたものを使用し上記をつけた。

29.2 基本記号

<基本記号> ::= <文字> | <数字> | <論理値> |

<区切り記号>

1) 文 字

<文字> ::= A|B|C|D|E|F|G|H|I|J|K|L|M|N|
 O|P|Q|R|S|T|U|V|W|X|Y|Z|a|b|
 c|d|e|f|g|h|i|j|k|l|m|n|o|p|
 q|r|s|t|u|v|w|x|y|z

(小文字を禁止するものもある)

(A) 数 字

<数字> ::= 0|1|2|3|4|5|6|7|8|9

(B) 論 理 値

<論理値> ::= true | false

(大文字のものもある)

3) 区切り記号 delimiter

<区切り記号> ::= <作用素> | <区切り> |

<括弧類> | <宣言詞> |

<特別規制詞>

<作用素> ::= <算術作用素> | <比較作用素> |

<論理作用素> | <順序作用素>

<算術作用素> ::= + | - | × | ÷ | ↑

(×の代りに*を用いるものも多い)

<比較作用素> ::= <| ≤ | ≥ | > | =

<論理作用素> ::= & | ∨ | ∧ | ¬

<順序作用素> ::= goto | if | then | else | for | do

<区切り> ::= , | ; | : | ! | = | step | until |

while | comment |

(= はスペースの意で#を用いるものもある)

<括弧類> ::= (|) | [|] | { | } | begin | end

<宣言詞> ::= Boolean | integer | real | array |

switch | procedure | own

(own を認めないものも多い)

<特別規制詞> ::= string | label | value

空白や改行はプログラム上の意味を持たない(空白に意味を持たせたものもある)。

4) 名 前 identifier

<名前: 記号> ::= <文字> | <T> <文字> | <T>

<数字>

(先頭の5~8文字を判別するものが多い)

5) 数 値

<符号のない整数: $\tilde{0}\tilde{1}$ と略> ::= <数字> | < $\tilde{0}\tilde{1}$ >

<数字>

<整数> ::= < $\tilde{0}\tilde{1}$ > | + < $\tilde{0}\tilde{1}$ > | - < $\tilde{0}\tilde{1}$ >

<小数> ::= . < $\tilde{0}\tilde{1}$ >

<指数部> ::= $_{10}$ <整数>

<10進数> ::= < $\tilde{0}\tilde{1}$ > | <小数> | < $\tilde{0}\tilde{1}$ >

<小数>

<符号のない数> ::= <10進数> | <指数部> |

<10進数> <指数部>

<数> ::= <符号のない数> | + <符号のない数> |

- <符号のない数>

6) 記 号 列 string

<記号の基本記号> ::= <'> を除く基本記号の任意の列> | <空>

<表題の記号列> ::= <記号の基本記号> | '<表題の記号列>' | <表題の記号列> <表題の記号列>

<記号列> ::= '<表題の記号列>'

記号列は手続きの実パラメタとして使われる。

7) comment

; comment <; をふくまぬ任意の記号列>;
begin comment <; をふくまぬ任意の記号列>;
end <end も ; も else もふくまぬ任意の記号列>

は

;
begin
end

に等しい。

29.3 式 expression

<式> ::= <算術式: $\tilde{A}\tilde{E}$ > | <論理式: $\tilde{B}\tilde{E}$ > |

<行先式: $\tilde{D}\tilde{E}$ >

1) 変 数

<変数名> ::= <名前: $\tilde{Y}$ >

<単純変数> ::= <変数名>

<添字式> ::= < $\tilde{A}\tilde{E}$ >

<添字のなりひ> ::= <添字式> | <添字のなりひ> ,

<添字式>

<配列名> ::= < $\tilde{Y}$ >

<添字付変数> ::= <配列名> [<添字のなりひ>]

<変 数> ::= <単純変数> | <添字付変数>

(添字式が実数のときは4捨5入される)

(A) 関数呼び出し function designator

<手続き名> ::= < $\tilde{Y}$ >

<実パラメタ: $\tilde{A}\tilde{P}$ > ::= <記号列> | <式> |

<配列名> | <スイッチ名> | <手続き名>

<文字列> ::= <文字> | <文字列> <文字>

<パラメタの区切り> ::= , | <文字列> : (

< $\tilde{A}\tilde{P}$ のなりひ> ::= < $\tilde{A}\tilde{P}$ > | < $\tilde{A}\tilde{P}$ のなりひ>

<パラメタの区切り> < $\tilde{A}\tilde{P}$ >

< $\tilde{A}\tilde{P}$ 部> ::= <空> | (< $\tilde{A}\tilde{P}$ のなりひ>)

<関数呼び出し> ::= <手続き名> < $\tilde{A}\tilde{P}$ 部>

関数呼び出しの実パラメタの数と順序は対応する関数呼び出し宣言の仮パラメタの数と順序に合っていない(パラメタの区切りを、のみに制限するものも多い。またパラメタの integer real のタイプも合致を要求するもの、value 指定していないものの合致を要求するものがある)。

(B) 標準関数

abs sign sin cos sqrt arctan ln exp

(大文字を使うものもある)

(C) 換算関数

entier

(大文字を使うものもある)

それを越さぬ最大の整数値をとる。

3) 算 術 式 ($\tilde{A}\tilde{E}$)

<加減作用素> ::= + | -

<乗除作用素> ::= * | / | $\div$

<一次子> ::= <符号のない数> | <変数> |

(< $\tilde{A}\tilde{E}$ >) | <関数呼び出し>

<因子> ::= <一次子> | <因子> * <一次子>

<項> ::= <因子> | <項> <乗除作用素>

<因子>

<単純算術式> ::= <項> | <加減作用素>

<項> | <単純算術式>

<加減作用素> <項>

<条件節> ::= if <論理式> then

<算術式: $\tilde{A}\tilde{E}$ > ::= <単純算術式> | <条件節>

<単純算術式> else <算術式>

4) 論 理 式 ($\tilde{B}\tilde{E}$)

<比較作用素> ::= <= > | <= > | > | < | <= > | <= >

<比較式> ::= <単純算術式> <比較作用素>

<単純算術式>

<論理一次子> ::= <論理値> | <変数> |

<関数呼び出し> | <比較式> |

(< $\tilde{B}\tilde{E}$ >)

<論理二次子> ::= <論理一次子> |

<論理一次子>

<論理因子> ::= <論理二次子> | <論理二次子>

<論理二次子>

<論理項> ::= <論理因子> | <論理項>

<論理因子>

<内包> ::= <論理項> | <内包> <論理項>

-53-

29.5 宣言

<宣言> ::= <型の宣言> | <配列の宣言> |
<スイッチ宣言> | <手続きの宣言>

1) 型の宣言

<型のならび> ::= <単統変数> | <単統変数> ,
<型のならび>

<型> ::= *real* | *integer* | *Boolean*

<局所または占有型> ::= <型> | *own* <型>

<型の宣言> ::= <局所または占有型> <型のならび>
(*own* を認めないものも多い)

2) 配列の宣言

<下 限> ::= <*NE*>

<上 限> ::= <*NE*>

<上下限> ::= <下限> : <上限>

<上下限のならび> ::= <下限> | <上下限のならび> , <上下限>

<配列片> ::= <配列名>
[<上下限のならび>] |
<配列名> , <配列片>

<配列のならび> ::= <配列片> | <配列のならび> ,
<配列片>

<配列の宣言> ::= *array* <配列のならび> |
<局所または占有型> *array*
<配列のならび>

(*own array* を認めないものも多い)

array と書けば *real array* と解される。

(下限) ≤ (上限) でなければならない。

3) スイッチの宣言

<行先式のならび> ::= <*NE*> |
<行先式のならび> , <*NE*>

<スイッチの宣言> ::= *switch* <スイッチ名> :=
<行先式のならび>

4) 手続きの宣言

<仮パラメタ: *FP*> ::= <*T*>

<*FP* のならび> ::= <*FP*> | <*FP* のならび>
<パラメタの区切り> <*FP*>

<*FP* 部> ::= <空> | (<*FP* のならび>)

<*T* のならび> ::= <*T*> | <*T* のならび> , <*T*>

<値の部> ::= *value* <*T* のならび> ; | <空>

<規制詞> ::= *string* | <型> | *array* |
<型> *array* | *label* | *switch* | *procedure* |

<型> *procedure*

<規制部> ::= <空> | <規制詞>

<*T* のならび> ; | <規制部>

<規制詞> <*T* のならび> ;

<手続きの頭> ::= <手続き名> <*FP* 部> ;
<値の部> <規制部>

<手続きの本体> ::= <文> | <ALGOL 以外の言語
による記述>

<手続きの宣言> ::= *procedure* <手続きの頭>
<手続きの本体> | <型>
procedure <手続きの頭> <手続きの本体>

(パラメタの区切りは、だけとするものも多い。また
array を *value* 指定できないものも多い。

コードは計算機によって適宜機械語等との接続に用いられる。

規制は *FP* 全部に必要なもの、*value* 指定されなかったもののうち *real* タイプのときのみ必要なものなどいろいろある)。

お わ り に

電子計算機は膨大な可能性を秘めている。しかしそれを引き出すのは人間であり、我々は今我々の行なっている仕事の手順をそのまま計算機に指令し、それに依って得た余裕を次のより良い計画に向けてと同時に計算機の出した結果により更によい手順や今迄思いつきもしなかった様な手順を考える方向に向かうべきである。

そのためには我々が日常語で書き下すのと殆んど同じ気易さで書き下せる電子計算用語ALGOL を習得して気軽に計算機と会話する必要がある。

建設事業関係の各機関にも電子計算組織の導入が始まって居り、まだ導入のすんでいない所でも使用時間当りの単価契約で自分の所の計算機と同じ様な気持ちで使用を始めたりしていろいろな仕事の手順を手続き (procedure) として蓄積する作業や互いに情報交換しながらそれらの諸手続きの改良などの作業が進められつつある。

電子計算組織を利用することは決して難かしいことではない。難かしいのは、自由に計算組織を活用して我々の果せなかった夢を実現しようとする動きを阻む考え方を取り除くことである。電子計算機を使うのは難かしいとか我々をきたえるには始めから手を下してやらせなければならぬとか、我々の仕事が無くなってしまおうとか、すべての仕事を外注してやらせてしまえば良いとか、夢の実現などという我々の遊びに金を出す訳には行かないとか云った考えが如何に誤りであるかを一人一人が説くことにより、他の技術分野と同じ様にこの利器を活用する方向に進む必要があることを強調して本文を終ることにしたい。

参 考 文 献

I 電子計算関係

1. 加瀬 滋男：コンピュータの常識 日刊工業新聞 (1963)
2. 小野 勝章：電子計算機 紀伊国屋 (1964)
3. 森口繁一編：ALGOL入門 日本科学技術連盟
4. 岡崎・角山：わかりやすいプログラミング 産報 (1965)
5. 中村 肇一：アルゴル・プログラミング入門 森北出版 (1968)
6. Naur: Revised Report on the Algorithmic Language ALGOL 60. ACM Vol. 6 No. 1 (1963)
7. Baumann et al: Introduction to ALGOL. Prentice - Hall (1964)
8. 山内・一松・森口：電子計算機のための数値計算法 I II 培風館 (1965)
9. 一松信：数値解析 至文堂 (1963)

II 土木計画学・R統計解析関係

1. 三上 操：仕事の数字 現代教養文庫
2. 森口 繁一：初等数理統計学 培風館
3. 小林・宮川他：一般応用統計学 川島書店
4. 岩田 曉一：経済分析のための統計的方法 東洋経済
5. 中村 肇一：技術者のための統計解析 山海堂
6. 中村 肇一：土木計画学通論 理工図書
7. ドレーバ、スミス、中村訳：応用回帰分析 森北出版