

44 - 4

電 子 計 算

技 術 管 理

土 木 研 究 所 資 料

第 478 号

電 子 計 算 用 語 ALGOL

(4 4 年 度 版)

昭 和 4 4 年 7 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

電子計算用語 ALGOL (44 年度版)

正 誤 表

頁	欄	誤	正
6	左-5	5680,6,10,56,8,8,20,954	;
"	右-17	6,10,56,8,8,20,95,4,そして...,0,0,0,	6,10,56,8,8,20,954, ...,0,0,0,
"	右-5	PRINTI(4,N)	PRINTI(4,N);
7	左-18	exp (指数関係)	exp (指数関数)
10	左-16	 ために, +-	 ために: +-
17	左-8	PRINTS('□□□AOV□TABLE');	PRINTS('###AOV#TABLE');
20	右-22	valve	value
24	右-17	integer M,N,R,AE;	integer M,N,R,AE,I,J;
26	左-5	0D1,0.0005,0.002,do	0.01,0.0005,0.002do
30	-5	条件付算術式の次に追加	ベクトル算術式 { vec 算術式, 算術式, ... tor vec ベクトル算術式の並び tor
31	9	条件は論理式	条件付論理式
	11	条件付論理式の次に追加	ベクトル論理式 { vec 論理式, 論理式..... tor vec ベクトル論理式の並び tor
31	5	条件付文字式の次に追加	ベクトル文字式 { vec 文字式, 文字式..... tor vec ベクトル文字式の並び tor
31	右-14	配列代入文	ベクトル式および* による添字制御
31	右-13	vec A(1,*):=A1,	A(1,*):=vec A1,
33	左4.Ⅲ	配列代入文	ベクトル式および* による添字制御
33	"	vec A(2,*):=A1,	A(2,*):=vec A2,
33	左4.XⅢ	BOOLEAN (0.1を論理値に対応させる)	論理式が算術式の中で使われたときは true 1 false 0 に対応する。
		その下に追加	XIV 初期値設定 integer A,B initial 0; array P[1:3] initial vec 3,1,4,8,2,6 tor;
48	左-6	に限り	に換算し

目 次

はじめに	1	17.2 式 <i>expression</i>	30
1. 建設技術と電子計算	2	17.3 文 <i>statement</i>	31
1.1 電子計算組織の導入を必要とする理由	2	17.4 手続き <i>procedure</i>	31
1.2 建設事業における電子計算向き業務	3	17.5 宣言 <i>declaration</i>	32
2. ALGOLとは	4	18. 各機種間の文法制限と入出力文	32
2.1 はじめに	4	18.1 TOSBAC 3400-40用 T-ALGOL	33
2.2 ALGOLの概要	5	18.2 OKITAC 5090-H用 NEW ALGOL-H	33
2.3 平均と不偏分散の平方根の計算プログラム	7	18.3 FACOM 230用 230ALGOL	34
2.4 普通の言語と ALGOL	10	18.4 FACOM 230-50 ALGOL	35
3. ALGOLの基本原則	11	18.5 TOSBAC用 KT-ALGOL	35
4. 宣言 (<i>declaration</i>)	12	18.6 CDC-3600用 ALGOL	35
5. 式 (<i>expression</i>)	12	19. アルゴリズムプログラムの交換	36
6. 文 (<i>statement</i>)	13	19.1 立方根の問題	36
6.1 基本文	13	19.2 NEW ALGOL-H への変換	36
7. ALGOLによるプログラム例	13	19.3 FACOM 230 ALGOL への変換	36
7.1 不偏分散の平方根の計算(I)	15	20. ALGOLとFORTRANの比較	37
7.2 不偏分散の平方根の計算(II)	16	21. 練習問題(I)	38
7.3 一元配置の分散分析	16	22. 練習問題解	38
8. アルゴリズムによるプログラミングの簡単な例題(I)	17	23. 電子計算機	39
8.1 相 関 係 数	17	23.1 電子計算機の構成	39
8.2 数 値 表	17	23.2 入力装置 (<i>input device</i>)	39
8.3 電気料金計算	17	23.3 制御装置 (<i>control unit</i>) および 演算装置 (<i>arithmetic unit</i>)	40
8.4 2次方程式の根	17	23.4 記憶装置 (<i>memory</i>)	40
9. 例 題 解 [I]	18	23.5 出力装置 (<i>output device</i>)	40
9.1 相 関 係 数	18	23.6 電子計算機の歴史	40
9.2 数 値 表	18	23.7 電子計算機一覧	40
9.3 電気料金計算	18	24. 重力式コンクリート橋壁自動設計プログラム	42
9.4 2次方程式の根	19	25. 重力橋壁自動設計のプログラム	43
10. 手続き (<i>procedure</i>)	19	26. 自動図化	45
10.1 概 要	19	27. これまでのまとめ	45
10.2 手続きの実例	20	28. ALGOL マニュアル	45
10.3 <i>value</i> 指定と規制	20	28.1 基本記号, 名前, 数値, 記号列	47
11. 関数呼び出し (<i>function designator</i>)	22	28.2 式 <i>expression</i>	47
11.1 概 要	22	28.3 節 <i>clause</i>	49
11.2 関数呼び出しの実例	22	28.4 文 <i>statement</i>	49
12. スイッチ (<i>switch</i>)	24	28.5 入出力文	49
13. アルゴリズムによる例題(II)	25	28.6 宣言 <i>declaration</i>	50
13.1 計画量の危険度の評価	25	28.7 データ	51
13.2 一元配置の分散分析	25	29. 改訂アルゴリズム60の構文法	51
14. 例 題 解 [II]	26	29.1 一般的説明	51
14.1 計画量の危険度の評価	26	29.2 基本記号	51
14.2 一元配置の分散分析	26	29.3 式 <i>expression</i>	52
15. 特殊な注意事項	27	29.4 文 <i>statement</i>	53
15.1 <i>array</i> の要素の取り扱い	27	29.5 宣 言	54
15.2 繰り返しの文の終期値の取り扱い	28	30. ALGOL, PL/I, FORTRAN, COBOL 比較	55
15.3 <i>and</i> の後の ; の取り扱い	28	31. 練習問題	57
15.4 繰り返しの文の制御変数	29	32. 練習問題解答	63
15.5 アルファモードの取り扱い	29	お わ り に	64
16. プログラムのエラーチェック	29	参 考 文 献	64
17. KDK ALGOL の構想	30	索 引	65
17.1 型	30		

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と始めると、「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な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があっても疎かなにかに従って判断の行なわれているような仕事か、さなけ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づ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も考えれば、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、あらかじめ辞書を覚えさせておいて、後でより大きなプログラムの中で、あの方法を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づける。

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単なる順序とか記号のみ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ぬ、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っさい不要であつて、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ぬわけであるが、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によって直接会話を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困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う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来た対象が、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、一方大地と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さねば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わないまでも、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、建設技術という場に合わせて一部変形した方が良いものも多い。

この諸技法を仮りに土木計画法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、岩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プログラムの養成も終り、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、土研に於ける創り込み計算や、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つ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)計算機を入れてみ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、全く同じ疑問がくり返され、(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、業績を伸べている所が多い。

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 (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上にあげたと全く同じ反対があつた時に、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、所内の計算機はいわば成功するかどうかわからない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-H (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1000多の稼動とする、(3)夜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ることができた。

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てみよう。

(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、或いはその外注のための積算、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可否を考えてみれば、自から手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があるの所であらう。

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返さ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もはるかに容易である。

もし仕事の内容がコンサルタントしか分か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らう。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る。

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業務提供策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れ妨害にはなから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、「他人に手預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回しかやらぬ事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ひつき或いは過去の経験、勘などをとくに、人に教えることの出来ない(技能的な)仕事である」と繰り返して述べて来た。

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、去年行っ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、またこの様にして作られたプログラムは手続きの形にまとめて格納し、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っ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ひつきも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」という考え方にとっ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。

その他積算・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

A 秘密保持のため

1. 積算関係

- a 実施設計書 変更設計書
- b 比較設計
- c 機械損料調査 歩掛調査

2. 人事関係

- a 給与・昇給等

3. 業者選定等

4. 計画関係

- a 路線選定 道路網計画
- b 工事優先順位

B 即時性のため

1. 工事関係

- a 予算管理
- b 物品管理
- c 工程管理

2. 洪水関係

- a 洪水予報
- b ダム管理

3. 情報検索

- a 人事管理
- b 文献検索
- c 資料検索

C 公物管理のため

1. 道路台帳

2. 河川台帳

- (2) プログラムはこちらで保有し内容の修正も自由に行なう必要のあるもの

A 河川関係

- a 流況調査 気象調査 河床変動調査
- b 災害調査 水需給調査
- c 波に関する計算 河道設計
- d 洪水解析 調節計算

B 道路関係

- a 交通量 OD 等
- b 道路の採算計算
- c 需要予測

C 水理計算

- a 不定流 不等流計算 背水計算

D 構造計算

- a 一般構造物標準設計
- b 橋梁 ダム

- (3) 部内で試的に計算してみる手段の存在によって進歩が促進されるもの

A 計画関係

- a 道路 網
- b 経済 効果
- c 流量 配分

d OR の適用

B 一般技術計算

- a 水理計算
- b 構造物の設計 (基礎)
- c 情報処理

- (4) ルーチンワークで外注しても良いが、計算時間の余裕があれば部内で行なうもの、内容修正の必要が考えられぬ位完成したもの、部内のチェックが完了したもの等。

従って本来部内の計算機に依らなければならない積算人事関係の計算や、技術者のより進んだ手法開発の手段として導入される計算機は又文献などの情報検索や、自動製図などにも有効に利用され得る。

以下 $ALGOL$ の単語と説明用の英字綴りを区別するため $ALGOL$ の単語はすべてイタリック (斜体) で $ALGOL$ 以外の単語はすべて普通文字 (立体) で示す。

2. $ALGOL$ とは

2.1 はじめに

戦時中、非英語系の国民の間に英語を普及するのに大いに役立ったベーシックイングリッシュですら 850 の単語が必要である。これに対して、なるほど恋愛小説は書けないが、推理小説のプロット位は書き下せる国際共通語 $ALGOL$ は、単語わずかに 32、カンマなどの記号が 30 と、アルファベットおよび 10 個の数字で成り立っているに過ぎない。

32 の単語および 30 の記号とは、区切り記号といわれるもので、次のようなものである。

順序作用素 (6) $go\ to\ do\ if\ then\ else\ for$

区 切 り (11) $;\ :\ ::\ :=\ step\ until$
 $while\ comment\ \downarrow$

宣 言 詞 (7) $real\ integer\ Boolean\ array\ switch$
 $procedure\ own$

特別規制詞 (3) $string\ label\ value$

論 理 値 (2) $true\ false$

標準関数 (8) $sin\ cos\ arctan\ sign\ abs\ exp\ ln$
 sqr

換算関数 (1) $entier$

括弧類 (8) $begin\ end\ () [] \cdot \cdot \cdot$

算術作用素 (6) $+\ -\ \times\ /\ \div\ \uparrow$

比較作用素 (6) $>\ \geq\ =\ \leq\ <\ \neq$

論理作用素 (5) $\wedge\ \vee\ \neg\ \supset\ \equiv$

これらは我々が日常使うのとはほとんど同じ意味に用いられるが、特別なものとして、 $go\ to$ は別々には用いられないこと、 10 はスモールテンと呼び 3×10^{-2} のことを $ALGOL$ では $-30-2$ と書くこと、 $:=$ はコロンイコールと読み、その右辺の式の結果を左辺の名前に代入させる術

きのあること, $\square$ は空白の意であること, $sqr(4.6)$ とは $\sqrt{4.6}$ の意であること, $entier(4.3)$ は 4 となること, $\times$ は掛算で $*$ と書かれる場合が多いこと, $\div$ は整数割算で $5 \div 2$ は 2 となること, $\uparrow$ は乗べきで A^* は $A \uparrow B$ 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 $K_{i,j}$ 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 4 5 分もあれば十分である。以下この 4 5 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 6.8 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 3.5 円, そうでなくてももし A が 15 なら 29 0.0 円, そうでなければ (当然 A は 20) 38 4.4 円である。
- ③ 電力量料金 (KF) は使用電力量当り 10 8.9 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 4 2 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 プログラムで書くとき次のようになる。

```
begin
  integer N;
  real A, KWH, F, BF, KF;
  N := 6; A := 10; KWH := 56.8;
  BF := if A = 10 then 19.35 else if A = 15
        then 29.00 else 38.44;
  KF := 108.9 * KWH;
  F := BF + KF;
  if F ≤ 429 then F := F / 1.07;
  PRINTI(4, N); PRINTI(5, F); CRLF(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る。
- 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ない。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 *PRINTI* 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るべきかは *CRLF* 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

$N=6, A=10, KWH=56.8$ 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 + 108.9 * KWH; if F ≤ 429 then F :=
        F / 1.07;
  PRINTI(4, N);
  PRINTI(5, F); CRLF(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ブル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ブルをそう入する。
  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ブルをそう入する
  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

何人のデータか分らぬ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ブルをそう入する
  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;
```

5, 10, 56.8, 8, 20, 95.4, 8, 20, 95.4, 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ブル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、このデータテーブルに組数 (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, 56.8, 8, 59.4,
..... 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 $A^T B$
 $\frac{A+B}{C+D}$ 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適用する。
 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 *sqrt* (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。
 $\sin(A \cdot B) + \cos(B)$
 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につくしていただく為に、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切歩の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 によつて予測を行なうときその一群の位置の尺度としては算術平均 $\bar{x}$ を、散布度の尺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 $\tilde{x}$ や範囲 R_x をその代りに用いることもあるが、得たデータのたまたす情報の 100% を利用しているという点では $\sqrt{V_x}$ が最良である。ここに

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

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

$S(x)$ の計算に当つては $\bar{x}$ を差引く段階で、 $\bar{x}$ の丸め誤差が大きくなっていくので

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

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
  real SX, MX, SXX, RV, X1, X2, X3;
  integer N;
  N := 3; X1 := 25.4; X2 := 28.2; X3 := 21.6;
  SX := X1 + X2 + X3;
  SXX := X1↑2 + X2↑2 + X3↑2;
  MX := SX / N;
  RV := sqrt((SXX - SX↑2 / N) /
    (N - 1));
  PRINTR(4, MX);
  PRINTR(4, RV);
  CRLF(1);
end;
```

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
  real SX, MX, SXX, RV, X1, X2, X3, X;
  integer N;
  N := 3; X1 := 25.4; X2 := 28.2; X3 := 21.6;
  SX := SXX := 0;
  SX := SX + X1; SXX := SXX + X1↑2;
  SX := SX + X2; SXX := SXX + X2↑2;
  SX := SX + X3; SXX := SXX + X3↑2;
  MX := SX / N;
  RV := sqrt((SXX - SX↑2 / N) /
    (N - 1));
  PRINTR(4, MX); PRINTR(4, RV);
  CRLF(1);
end;
```

ここで⑥⑦⑧の右辺に SX , SXX が出てくるが、いづ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。

さらに何度も出てくる所を手続きにまとめたり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
    MX := SX / N; ④
    RV := sqrt((SXX - SX2 / N) / ⑤
              (N - 1)); ⑥
    PRINTR(4, MX); PRINTR(4, RV); ⑦
    CRLF(1); ⑧
  end; ⑨
  N := 3; ⑩
  X1 := 2.54; X2 := 2.82; X3 := 2.16; ⑪
  I := 1; SX := SXX := 0; ⑫
  START: ⑬
  X := if I = 1 then X1 else if I = 2 ⑭
        then X2 else X3; ⑮
  SX := SX + X; ⑯
  SXX := SXX + X2; ⑰
  I := I + 1; ⑱
  if I > N then go to OUT; ⑲
  go to START; ⑳
  OUT: ㉑
  PR; ㉒
end; ㉓

```

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⑮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⑲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。

㉑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㉑でループの始めに戻ることを命じている。

⑬～⑲の部分は如何に今後の一般化の為とはいながらも、いさかかわづらわしい。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⑪⑬と⑱⑲の代入文の代りに、READ 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

今度はデータテープを使用し、データテープ上では個数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, ⑯データ

```

⑥によってデータテープの先頭にある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2.5.4という情報が待機していることになる。⑧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⑨～⑬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の状態です。⑨～⑬を実行した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⑱⑲などの代行になっている。また⑮のREAD(X);によってI = 1, 2, に対応してデータテープ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⑪⑱の代行を行なっているとも考えられる。⑬のP R;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⑤～⑨のテープがつなぎ込んであればそれら)が呼び出されてP R;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XAという名の配列にデータを格納して、このXA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10⁹⁹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N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] ≥ 1070 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='); PRINT I(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] などといっ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 ⑤~⑨を書く end; ⑥
    N:=1;                     ⑦
    START: READ(AE);          ⑧
    if AE ≥ 1070 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='); PRINT I(4, N-1); ⑱
    CRLF(1);                  ⑲
    PR;                        ⑳
end;                          ㉑

```

今度は②でAEという変数を宣言して置いて⑧⑨⑩および⑱⑲で使用している。

序論の部分で用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おく。データはNを先頭にN個のXi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)/ ⑰
            (N-1));            ⑱
    end;                       ⑲
    READ(N);                  ㉑
    for I:=1 step 1 until N do ㉒
    READ(XA[I]);               ㉓
    MXRV;                      ㉔
    PRINTR(4, MX); PRINTR(4, RV); ㉕
    CRLF(1);                   ㉖
end;                          ㉗

```

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・かたかな等より成る。

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、 $+-\times\div/\geq\leq<*\wedge\vee\neg$
 $\equiv()' , []$ などの記号を使用する。

文をくくる括弧

別でない。

表現方法

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

2) $A_{i,j}$ (マトリクス A の i 行 j 列要素)

3) Q^2

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 という名札の所に飛越せ。

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 の値を有効なけたで印刷せよ。

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属する。

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、 $+-\times\div/\geq\leq<*\wedge\vee\neg$
 $\equiv()' , []$: 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) $\text{if } A > 4.0 \times 10^{-2} \text{ then go to } ST;$

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)

```
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)

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ぬ。
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先頭の *begin* の直後の宣言部で宣言されていなければならない。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。

もちろん一段内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. 文は飛び越え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に式 (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、代りに次の表現を用いる。

A_3	$PQ_{k,l}$	は	$A[3]$	$PQ(K,L)$
A^*	C^{*+2}	は	$A↑B$	$C↑(B+2)$
$\frac{A+B}{C+D}$	は	$(A+B) / (C+D)$		

ALGOL 60 では大文字小文字ともに名前として使用

してよいが、機械によって全面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 $3.6 - 2 \times 10^{-3} \sim 2.8 \times 10^2$

など小数点か 10 のべきを含むが列の中の後の 2 つは $-2_{10} \sim 3_{28,10^2}$ と小さい 10 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 P43 L39 B MATRIX*

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 *3 B* 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。

$\sim$ *AE* 算術式 (arithmetic expression)

結果として 1 つの算術値に帰着する表現。

$\sim$ *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。

$\sim$ *BE* 論理式 (Boolean expression)

結果として 1 つの論理値に帰着する表現。

$\sim$ *D* 宣言 (declaration)

名前の型・内容等をその使用に先き立つて示す部分。

$\sim$ *DE* 行先式 (designational expression)

結果として 1 つの名札に帰着する表現。

$\sim$ *FP* 仮パラメタ (formal parameter)

手続きの宣言に用いるパラメタ。

$\overline{FS}$ 繰り返し文 (for statement)

くり返しを指示する文。

$\overline{I}$ 名前 (identifier)

$\overline{S}$ 文 (statement)

計算機に1つの操作を実行させる単位。

$\overline{US}$ 無条件文 (unconditional statement)

文の中で条件文、 $\overline{FS}$ 以外のもの。

$\overline{V}$ 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最初の *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ぐ、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ぐ、さらに同じ添字の境界のものは名前だけをカンマでつなぐでセミコロンで切る。

(例) 実数のベクトル A, B の境界が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];

ただし添字の境界は下限 $\leq$ 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。その後に必要に応じて $\overline{FP}$ 部を括弧につくセミコロンで切る。 $\overline{FP}$ 部があるときはその後に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価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 $:=$ の後にスイッチリストをカンマでつなぐ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) に当るのが式である。

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

1) $\overline{AE}$ (算術式)

$\overline{AE}$ 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

- 算術定数 $-3.5 \quad 4.5_{10} \quad -2 \quad 10^{-2}$ (*real*)
 $4 \quad -8601$ (*integer*)
- 算術変数 $AB \quad R4[1, S]$
 $S41 \text{ [if } P=Q \text{ then } 3 \text{ else } 4 \text{]}$
- 算術関数呼び出し $TAN(A) \quad MAX(P, Q)$
標準関数 $\sin \cos \arctan \ln \exp \operatorname{sign}$
 $\operatorname{abs} \operatorname{sqr}$
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、*sqr* は平方根、*entier* はアンティエと読み、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んだもの。

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

6. 1~5 を $+ \cdot / \div$ という四則演算子でつないだもの。

*は $\times$ の代りに用いられることが多い (大文字のエクスと混同しないため)。

$\div$ は整数割算に用い $5 \div 2$ は 2 である。

$\uparrow$ 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 $\overline{SA}$ と書けば条件付算術式は

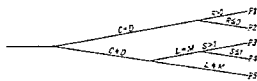
$\text{if } \overline{BE} \text{ then } \overline{SA} \text{ else } \overline{AE}$

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

(例)

$\text{if } P > B \text{ then (if } P \neq Q \text{ then } 3 \text{ else } 3+S$
 $\cdot \text{ (then の後に条件の if が直接くることを禁止しているので括弧でつづんである) 。}$

$\text{if } R \geq S \wedge C = D \text{ then } A \text{ [if } P_1 = P_2 \text{ then } 2 \text{ else } 3,$
 $N] + 4.6 \text{ else } W + \text{(if } L = M \text{ then } 3.6 \text{ else } 4.8) + 2$



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 $PQ A_2 \{R, 4\}$
 $S \{if A=B \text{ then } R \text{ else } 1, 2\}$
3. 論理関数呼び出し $BLANCH(P, Q)$
4. 比較式 $\tilde{A}\tilde{E}_1 \rho \tilde{A}\tilde{E}_2$
ただし ρ とは $\geq, \leq, >, <, =, \neq$ の6種である。
5. 条件付論理式を丸括弧でつつんだもの
6. 1~5のものの前に $\neg$ (not) をつけたもの
7. 1~6 を $\wedge$ (and), $\vee$ (or), $\rightarrow$ (imp), $\equiv$ (equivalent) でつないだり括弧でつつんだりしたもの。

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

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

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 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 $LR \neq LS$

最後の例は — の部分は — に対して if と then にはさ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 ではさ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 $S \{3\}$
 $S \{if P=Q \text{ then } 1 \text{ else } 2\}$
3. 条件付行先式を括弧につづんだもの
4. 1~3 を括弧につづんだもの

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、

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

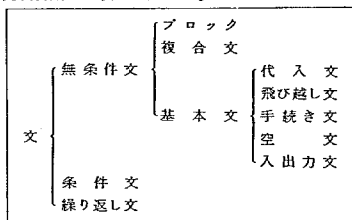
で、これも ALGOL の特長の1つである。

(例)

if $P=Q$ then (if $R \neq S$ 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へて、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 の前にあると考へ、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倍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 同文であ

るときは

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

と何重に重ねてもよい。

(例)

$A := B + C;$

$RS(I) := 4 + X;$

$L1 := P := AV(1, 4) := 0;$

$KN := \text{true};$

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。

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

(例)

$\text{go to } L1;$

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

$\text{go to if } A>0 \text{ then } S(I) \text{ else}$

$\text{if } B=0 \text{ then } PQ1 \text{ else } P2;$

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) 建設省共通入出力文

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省共通入出力文とする。

$\text{READ}(A, B, \dots);$

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。

$\text{PRINTR}(N, A);$

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。

$\underbrace{\underbrace{\pm \times \times \dots \times}_{10} \times \underbrace{\times \times \times \times}_{2}}_N$

10 の後は指数部であり、 $\text{PRINTR}(5, S);$ で S が -196515 なら $-1.9652_{10} + 04$ となる。

$\text{PRINTI}(N, A);$

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。

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

$\text{PRINTF}(n_1, n_2, A);$

式 A の値を real 型で小数点以上 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。

$\underbrace{\pm \times \dots \times}_{n_1} \cdot \underbrace{\times \dots \times}_{n_2}$

たとえば $\text{PRINTF}(23, P);$ で P が 3639 なら

± 36390

という印刷準備が行なわれる。 $\pm$ はスペースの意である。

$\text{CRLF}(N);$

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。

$\text{SPACE}(N);$

N 字分の空白を印刷準備せよ。

$\text{PRINTS}('PQR');$

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、空白を打ちたいれば SPACE を用いてもよいが、' ' の中で $\pm$ をならべて書いてもよい。

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

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

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

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

$\text{READARRAY}(A, 1, M, 1, N);$

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 の型のときは $\bar{B}\bar{E}$ 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 でつつめば $\bar{U}\bar{S}$ となる。

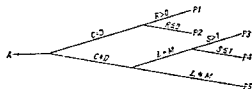
たとえば

$\text{if } A=B \text{ then begin if } C \approx D \text{ then } S; S+1$

$\text{else } S; S-1 \text{ end else } R; P \cdot Q;$

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 () であることがはっきり分る。

Ⅱ) 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 V:=<for list> do 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↑2);
CRLF (1);
end;

```

となろう。

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

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↑2);
CRLF (1);
end;

```

とすればよい。A の終止条件を 1.12 として 1.1 としなかった理由は 2 進の計算機で次々と 0.1 を加えてゆく過程で 10 進のきつちりした 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てきざみの 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、たいいていの問題を取り扱うことができる。そこで、二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 10²⁰ というような非常に大きな数が入っているものとする。

また印刷様式は N = ..., M X = ..., 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↑2 ;
I := I + 1 ;
go to ST ;
OUT : MX := SX / I ;
RV := sqrt ((SXX - SX↑2 / 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 *real* 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確保することを命じている。ここで宣言部を終わり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。

⑧⑨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 *BE* 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。

⑫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が、*SX* には総和が入っているはずである)代入される。

⑬では *RV* に不偏分散の平方根が代入される。

⑭の文によって *N* = という字が、⑮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、⑯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、⑰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 *X*₁ 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                                     ①
  read MX, RV, SX, SXX, MAX;           ②
  integer N, I;                         ③
  array X [1:500];                     ④
  READ (N);                             ⑤
  READARRAY (X, 1, N);                 ⑥
  MAX := -10.50; SX := SXX := 0;        ⑦
  for I := 1 step 1 until N do          ⑧
    begin                                ⑨
      SX := SX + X [I];                 ⑩
```

```
SXX := SXX + X [I] * X [I];             ⑪
MAX := if MAX > X [I] then              ⑫
  MAX else X [I];                       ⑬
end;                                     ⑭
PRINTS ('MAX = '); PRINTR (5, MAX);     ⑮
PRINTS ('MX = '); PRINTR (5, SX / N);   ⑯
PRINTS ('RV = ');                       ⑰
PRINTR (5, sqrt ((SXX - SX * SX / N) / (N - 1))); ⑱
CRLF (1);                                ⑲
end;                                     ⑳
```

④は配列宣言、⑥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		<i>M</i>
データ	x_{11}	x_{21}	x_{31}		x_{m1}
	x_{12}	x_{22}	x_{32}		x_{m2}
	$\vdots$	$\vdots$	$\vdots$		$\vdots$
	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*₁ 各水準のデータサイズ *N*₁, *N*₂, ..., *N*_{*m*} データ x_{11} , x_{12} , ..., x_{1n_1} , x_{21} , x_{22} , ..., x_{2n_2} , ..., 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 $SSO = \sum_{i,j} X_{ij}^2 - (\sum_j X_{ij})^2 / \sum_i N_i$
 $= \sum_{i,j} X_{ij}^2 - CF$
 $SSA = \sum_i \{ (\sum_j X_{ij})^2 / N_i \} - CF$
 $SSE = SSO - SSA$
 $NA = M - 1$
 $NO = \sum_i N_i - 1$
 $NE = NO - NA$
 $FA = SSA / NA$
 $FE = SSE / NE$
 $FO = FA / FE$

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];
  read SSA, SSE, SSO, CF, FA, FO, FE, SO;
```

```

array XA{1:20,1:10},SX,MX,SSX,
RV{1:10};
READ(N);
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 1 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( 11 AOV 1 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

8.1 相関係数

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

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

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

を計算し、 $R = \dots$ という形で印刷せよ。

8.2 数表

$N \ N^2 \ N^3 \ 1/N \ N/$ の表を1から1きざみで10,000まで作れ、ただし表頭に $NN \ NNN \ 1/N \ FACTN$ 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÷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?多である。テーブルより番号 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

データテーブルより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 $X1(+)$ の項 $X2(-)$ 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ておく、正しい動作を行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されるべきであろう。

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, SYY, SXY, R;
  array X, Y(1:500);
  READ(N);
  READARRAY(X, 1, N);
  READARRAY(Y, 1, N);
  SX := SY := SXX := SXY := SYY := 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;
  SXX := SXX - SX2/N;
  SYY := SYY - SY2/N;
  SXY := SXY - SX * SY/N;
  PRINTS("R=");
  R := SXY/sqrt(SXX * SYY);
  PRINTF(1, R);
  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)

$N^2, N^3, 1/N, N!$ の表を 1(1) 10000 作り、表頭に $N, NN, NNN, 1/N, FACTN$ 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("1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037 1038 1039 1040 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 1062 1063 1064 1065 1066 1067 1068 1069 1070 1071 1072 1073 1074 1075 1076 1077 1078 1079 1080 1081 1082 1083 1084 1085 1086 1087 1088 1089 1090 1091 1092 1093 1094 1095 1096 1097 1098 1099 1100 1101 1102 1103 1104 1105 1106 1107 1108 1109 1110 1111 1112 1113 1114 1115 1116 1117 1118 1119 1120 1121 1122 1123 1124 1125 1126 1127 1128 1129 1130 1131 1132 1133 1134 1135 1136 1137 1138 1139 1140 1141 1142 1143 1144 1145 1146 1147 1148 1149 1150 1151 1152 1153 1154 1155 1156 1157 1158 1159 1160 1161 1162 1163 1164 1165 1166 1167 1168 1169 1170 1171 1172 1173 1174 1175 1176 1177 1178 1179 1180 1181 1182 1183 1184 1185 1186 1187 1188 1189 1190 1191 1192 1193 1194 1195 1196 1197 1198 1199 1200 1201 1202 1203 1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245 1246 1247 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260 1261 1262 1263 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289 1290 1291 1292 1293 1294 1295 1296 1297 1298 1299 1300 1301 1302 1303 1304 1305 1306 1307 1308 1309 1310 1311 1312 1313 1314 1315 1316 1317 1318 1319 1320 1321 1322 1323 1324 1325 1326 1327 1328 1329 1330 1331 1332 1333 1334 1335 1336 1337 1338 1339 1340 1341 1342 1343 1344 1345 1346 1347 1348 1349 1350 1351 1352 1353 1354 1355 1356 1357 1358 1359 1360 1361 1362 1363 1364 1365 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376 1377 1378 1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1389 1390 1391 1392 1393 1394 1395 1396 1397 1398 1399 1400 1401 1402 1403 1404 1405 1406 1407 1408 1409 1410 1411 1412 1413 1414 1415 1416 1417 1418 1419 1420 1421 1422 1423 1424 1425 1426 1427 1428 1429 1430 1431 1432 1433 1434 1435 1436 1437 1438 1439 1440 1441 1442 1443 1444 1445 1446 1447 1448 1449 1450 1451 1452 1453 1454 1455 1456 1457 1458 1459 1460 1461 1462 1463 1464 1465 1466 1467 1468 1469 1470 1471 1472 1473 1474 1475 1476 1477 1478 1479 1480 1481 1482 1483 1484 1485 1486 1487 1488 1489 1490 1491 1492 1493 1494 1495 1496 1497 1498 1499 1500 1501 1502 1503 1504 1505 1506 1507 1508 1509 1510 1511 1512 1513 1514 1515 1516 1517 1518 1519 1520 1521 1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532 1533 1534 1535 1536 1537 1538 1539 1540 1541 1542 1543 1544 1545 1546 1547 1548 1549 1550 1551 1552 1553 1554 1555 1556 1557 1558 1559 1560 1561 1562 1563 1564 1565 1566 1567 1568 1569 1570 1571 1572 1573 1574 1575 1576 1577 1578 1579 1580 1581 1582 1583 1584 1585 1586 1587 1588 1589 1590 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600 1601 1602 1603 1604 1605 1606 1607 1608 1609 1610 1611 1612 1613 1614 1615 1616 1617 1618 1619 1620 1621 1622 1623 1624 1625 1626 1627 1628 1629 1630 1631 1632 1633 1634 1635 1636 1637 1638 1639 1640 1641 1642 1643 1644 1645 1646 1647 1648 1649 1650 1651 1652 1653 1654 1655 1656 1657 1658 1659 1660 1661 1662 1663 1664 1665 1666 1667 1668 1669 1670 1671 1672 1673 1674 1675 1676 1677 1678 1679 1680 1681 1682 1683 1684 1685 1686 1687 1688 1689 1690 1691 1692 1693 1694 1695 1696 1697 1698 1699 1700 1701 1702 1703 1704 1705 1706 1707 1708 1709 1710 1711 1712 1713 1714 1715 1716 1717 1718 1719 1720 1721 1722 1723 1724 1725 1726 1727 1728 1729 1730 1731 1732 1733 1734 1735 1736 1737 1738 1739 1740 1741 1742 1743 1744 1745 1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756 1757 1758 1759 1760 1761 1762 1763 1764 1765 1766 1767 1768 1769 1770 1771 1772 1773 1774 1775 1776 1777 1778 1779 1780 1781 1782 1783 1784 1785 1786 1787 1788 1789 1790 1791 1792 1793 1794 1795 1796 1797 1798 1799 1800 1801 1802 1803 1804 1805 1806 1807 1808 1809 1810 1811 1812 1813 1814 1815 1816 1817 1818 1819 1820 1821 1822 1823 1824 1825 1826 1827 1828 1829 1830 1831 1832 1833 1834 1835 1836 1837 1838 1839 1840 1841 1842 1843 1844 1845 1846 1847 1848 1849 1850 1851 1852 1853 1854 1855 1856 1857 1858 1859 1860 1861 1862 1863 1864 1865 1866 1867 1868 1869 1870 1871 1872 1873 1874 1875 1876 1877 1878 1879 1880 1881 1882 1883 1884 1885 1886 1887 1888 1889 1890 1891 1892 1893 1894 1895 1896 1897 1898 1899 1900 1901 1902 1903 1904 1905 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915 1916 1917 1918 1919 1920 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931 1932 1933 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944 1945 1946 1947 1948 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025 2026 2027 2028 2029 2030 2031 2032 2033 2034 2035 2036 2037 2038 2039 2040 2041 2042 2043 2044 2045 2046 2047 2048 2049 2050 2051 2052 2053 2054 2055 2056 2057 2058 2059 2060 2061 2062 2063 2064 2065 2066 2067 2068 2069 2070 2071 2072 2073 2074 2075 2076 2077 2078 2079 2080 2081 2082 2083 2084 2085 2086 2087 2088 2089 2090 2091 2092 2093 2094 2095 2096 2097 2098 2099 2100 2101 2102 2103 2104 2105 2106 2107 2108 2109 2110 2111 2112 2113 2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128 2129 2130 2131 2132 2133 2134 2135 2136 2137 2138 2139 2140 2141 2142 2143 2144 2145 2146 2147 2148 2149 2150 2151 2152 2153 2154 2155 2156 2157 2158 2159 2160 2161 2162 2163 2164 2165 2166 2167 2168 2169 2170 2171 2172 2173 2174 2175 2176 2177 2178 2179 2180 2181 2182 2183 2184 2185 2186 2187 2188 2189 2190 2191 2192 2193 2194 2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208 2209 2210 2211 2212 2213 2214 2215 2216 2217 2218 2219 2220 2221 2222 2223 2224 2225 2226 2227 2228 2229 2230 2231 2232 2233 2234 2235 2236 2237 2238 2239 2240 2241 2242 2243 2244 2245 2246 2247 2248 2249 2250 2251 2252 2253 2254 2255 2256 2257 2258 2259 2260 2261 2262 2263 2264 2265 2266 2267 2268 2269 2270 2271 2272 2273 2274 2275 2276 2277 2278 2279 2280 2281 2282 2283 2284 2285 2286 2287 2288 2289 2290 2291 2292 2293 2294 2295 2296 2297 2298 2299 2300 2301 2302 2303 2304 2305 2306 2307 2308 2309 2310 2311 2312 2313 2314 2315 2316 2317 2318 2319 2320 2321 2322 2323 2324 2325 2326 2327 2328 2329 2330 2331 2332 2333 2334 2335 2336 2337 2338 2339 2340 2341 2342 2343 2344 2345 2346 2347 2348 2349 2350 2351 2352 2353 2354 2355 2356 2357 2358 2359 2360 2361 2362 2363 2364 2365 2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2379 2380 2381 2382 2383 2384 2385 2386 2387 2388 2389 2390 2391 2392 2393 2394 2395 2396 2397 2398 2399 2400 2401 2402 2403 2404 2405 2406 2407 2408 2409 2410 2411 2412 2413 2414 2415 2416 2417 2418 2419 2420 2421 2422 2423 2424 2425 2426 2427 2428 2429 2430 2431 2432 2433 2434 2435 2436 2437 2438 2439 2440 2441 2442 2443 2444 2445 2446 2447 2448 2449 2450 2451 2452 2453 
```

```

B:=if A=10 then 175 else
    if A=15 then 263 else
    if A=20 then 350 else
        525;
FEE:=(B+KWH*10.8)*1.07;
PRINTI(4,I);
PRINTI(2,A);
PRINTF(2,1,KWH);
PRINTF(4,2,FEE);
CRLF(1); go to A1;
END;

```

基本料金Bの計算にはAによる条件式を用いた。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

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

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

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
    real 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CR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 ~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、しかも今後他の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なきは、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っ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および

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, SYY, 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;
    R := (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);
  SOKAN(M, P, Q, K);
  PRINTF(1.4, K);
  CRLF(1);

```

```

  go to ST;
  STOP;
end;

```

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ータとする手続きであって、仮
りに N, X, Y, R という名がつけられている。
⑥ N は後で使用するときその実行時の対応するパラメータの
持っている値でおきかえられ、value 指定していないパラ
メータは、実行時の対応するパラメータの名前でおきかえられ
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 ~ ⑩ 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。
⑪ 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。

⑫ で終止条件がきたら⑬にジャンプさせるつもりである
STOP はもちろん単なる名でストップという字の内容と
はまったく無関係である。
⑭ で SOKAN という手続きが呼び出され、実パラメータの
M, P, Q, K がそれぞれ⑤の 仮 パラメータ N, X, Y, R と対応
している。

10.3 value 指定と規制

宣言中仮パラメータ部は ($\tilde{F}P$, $\tilde{F}P$) 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 とはその実パラメータとなっている変数の値を変更
しないという意思表示である。(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ー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 指定されなかったパラメータのうち
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ー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) パラメータとしての procedure

procedure のパラメータとして 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。

① x_1^2 とか $\sum x_i^2$ 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思う。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(L, ARRAY, SUM);
    value L;
    integer L; 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。

⑭～⑲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。

⑳～㉑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。

㉒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。

㉓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。

㉔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。

㉕からプログラムの本文が始まる。

㉖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。

㉗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つて複雑な手続きを *TANP* の中でいちいち書かなくと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8ケタ数値の左側に勝手な字を書きたい場合はそのたとごと

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 ' $'$ につづまれた文字数字の並びがくることを示している。

11. 関数呼び出し (function designator)

11.1 概要

10.10.2 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* の場合とほとんど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 $\tilde{P}$ 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かりやすいので前に *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, ..., XS, Y1, Y2, ..., YS, ..., 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, SYY, 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+P↑2;
SYY:=SYY+Q↑2;
SXY:=SXY+P*Q;
end;
SOKANKEISU:=(SX-
SX*S Y/N)/sqrt((SXX
-SX↑2/N)*(SYY-
SY↑2/N));
end;

```

```

ST:=READ(M);
if M=0 then go to STOP;
READARRAY(P, 1, M);
READARRAY(Q, 1, M);
PRINTF(1.4, 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) の答を受ける $\tilde{P}$ 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 を用いて大きさ 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) iii) 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。データタイプは、

```

N, M, X1, ..., XS, Y1, ..., YS}

```

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;
    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;
      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;
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)

大きな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の方向にはかった距離の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、

$$S(xy) = \sum xy - (\sum x)(\sum y)/n$$

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 $\hat{a}, \hat{b}, r$ を有効数字ケタで印刷するプログラムは
 $n \leq 200$ とすれば

```

begin
  real SX,SY,SXX,SYX,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*SY/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;

```

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(4, A);
    PRINTR(4, B);
    PRINTR(4, R);
    CRLF(1);
  end;

```

このプログラムの特色は⑤～⑭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きとしたことで、これは⑬⑭ではそれぞれXおよびYの和となっているが⑬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⑬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。

⑯～㉑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 $1/N$ にでもすためである。

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日を起日とする曜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

$$go\ to\ S(K - K \div 7 * 7 + 1);$$

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(' NIT ');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び越し文の部分であろう。

13. ALGOL によるプログラム例題〔II〕

13.1 計量の危険度の評価

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れば、

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

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

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

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

$$H(P) =$$

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

$K(P)$ 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

$$K(P) = \eta - \left\{ \frac{2.30753 + 0.27061\eta}{1 + 0.99229\eta + 0.04481\eta^2} \right\}$$

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

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 $x_{21}, \dots, x_{2n_2}, \dots$ 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 計画量の危険度の評価

データテーブル

$N, x_1, \dots, x_N$

印刷様式

```

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

```

begin

```

real 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(0.05);

```

```

for P:=0.5, 0.2, 0.1, 0.05, 0.02,
0.01, 0.005, 0.002, do

```

```

begin

```

```

KP:=KPF(P);

```

```

H:=(KP+sqrt(KP↑2-(1-
K5↑2/2/(N-1)))+(KP↑2

```

```

-K5↑2/N))))/(1-K5↑2/(2*
(N-1)));
PRINT!(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
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、A $\bar{O}W$ 表を印刷する procedure A $\bar{O}W$ 1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。

```

begin
integer I, J;
real MX, RV, F $\bar{O}$ ;
integer array IA[1:10];
array XA[1:20, 1:10], MX, RV
[1:10];
procedure A $\bar{O}W$ 1(M, IA, XA, MX,
RV, F $\bar{O}$ );
value M;
integer M; integer array IA;
array XA, MX, RV; real F $\bar{O}$ ;
begin
integer I, J, N $\bar{O}$ , NA, NE;
real SSA, SSE, SS $\bar{O}$ , CF, FA;
array SX, SSX[1:10];
SPACE(5); PRINTS('MEAN');
SPACE(8); PRINTS('RV');
SPACE(8); PRINTS('N');
CRLF(1);
SS $\bar{O}$ :=SS $\bar{O}$ :=SSA:=0; N $\bar{O}$ :=NA:=0;
for I:=1 step 1 until M do
begin
integer K1, K2, K3, K4, K5;
real S, SS;
K1:=IA(I);
S:=SS:=0;
for J:=1 step 1 until K1 do
begin
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;
NO:=NO+K1;
SO:=SO+K4;
end;
NO:=NO-1;
NA:=M-1;
NE:=NO-NA;
CF:=SO↑2/(NO+1);
SSO:=SSO-CF;
SSA:=SSA-CF;
SSE:=SSO-SSA;
FA:=SSA/NA;
FE:=SSE/NE;
FO:=FA/FE;
SPACE(7);
PRINTS('AOP↑TABLE');
CRLF(1);
PRINTS('↑SV');
SPACE(7); PRINTS('SS');
SPACE(7); PRINTS('DF');
SPACE(7); PRINTS('MS');
SPACE(7); PRINTS('F');
CRLF(1);
PRINTS('↑A');
PRINTF(44,SSA);
PRINTI(3,NA);
PRINTF(33,FA);
PRINTF(33,FO);
CRLF(1);
PRINTS('↑E');
PRINTF(44,SSE);
PRINTI(3,NE);
PRINTF(33,FE);
CRLF(1);

```

```

PRINTS('-----');
PRINTS('-----');
CRLF(1);
PRINTS('↑O');
PRINTF(44,SSO);
PRINTI(3,NO);
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));
AOP↑(M,IA,XA,MX,RV,FO);
end;

```

AOP↑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時間を1回限りにしておく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↑は空白に抜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にはい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、計算機の特長などによってちょっとした注意で計算能力が良くなったり、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 AOP↑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 の中味を調べに出發す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、繰り返し文中で array の要素を式中で使用する時にはたとえ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 である場合は終値についても実行させたいと思つ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(12, A);
  PRINTF(14, 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、0.050000001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となつてしまい、これは until の条件を超過し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つてプログラムにとっては内部で2進の演算をしようがそれは知つたことではないわけであるから ALGOL 語訳誤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(訳語集プログラム)製作者にその要望を何回か行なつたが数学出身のコンパイラ製作者はまことに潔癖であつて、このような「きたない」仕事には賛意を表してもらえなかつた。そこで止むを得ずプログラム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

セミコロンは同様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。したがつて文と end との区切りには別段 ; 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。ところがこのような end end の重なりを繞つ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 ; がなくてもよいとち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) の最後のところで

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

とした場合文法的には間違いは別段ない。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 扱いとなつ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つて読みとまれ、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,

となつてるとき、先頭の5もXというarrayの第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へはないし、最後の *end* の後は、まったく読まないからこれが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つ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 $1 \sim N-1$ 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に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プ上 $\rightarrow TOK$ *Y0*, $\rightarrow YOKOHAM$, $\rightarrow A$, などと前に $\rightarrow$ をつけて 7 文字までを 1 組に行ない、代入は、

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

などという形で行って適宜転送処理し、印字は、

$PRINT('TA', K);$

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。しかしなんといっても人間の作っ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前に、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っ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、あまりたびたびかけ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でもある。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・名札・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または読み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) 区切り記号の扱いに誤りはないか?
- 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 $(A+B)C$ とか $4B$ 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み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 *sqrt* 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?
 $A \uparrow -5$ や $3 \downarrow -48$ に注意

iv) 文関係

- 1) 文は *else* の直前を除き、すべて ; で終わっている

か？

特に *begin* と *end* でつづ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* でつづんであるか？

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 ; まで) であるかそれよりよいか？

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使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 ; 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 *1 array		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	'PQ5'等	(alpha)	string	(alpha) array		(alpha) procedure		(文字式)
名 札	L 5 等	(label)	label	(label)*2 array		(label) procedure		行 先 式

* 1 単に array とすれば real array

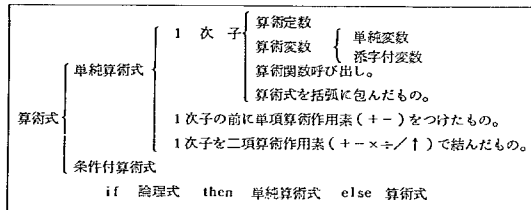
ただし T-ALGOL では記号列に関しては全面的

* 2 ALGOL 60 では switch で宣言と代入を兼ねている。

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

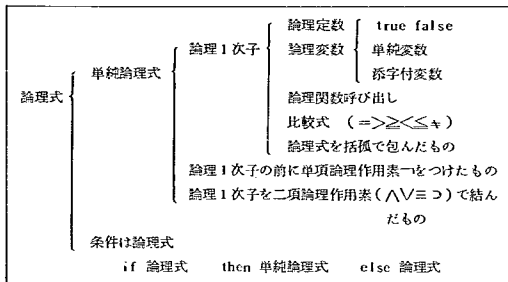


1. 標準関数として sinh, cosh, tanh, log, random (一様乱数), nrandom (正規乱数), fix (切り捨て) を追加する。

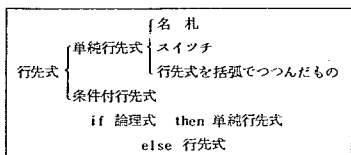
ただし T-ALGOL では大文字とする (標準関数

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。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。
T-ALGOL では KEY ,
BOOLEAN (0,1 を論理値に換算)
を認める。

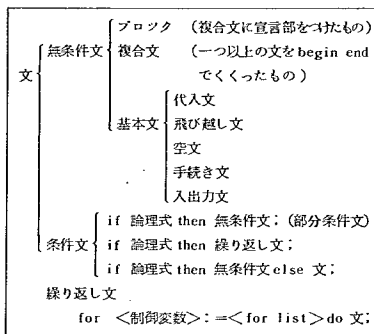


ALGOL の構想を受け入れる。(一語6文字)

式に関する注意

複素数宣言 complex を認める。T-ALGOL も同じ。

17.3 文 statement

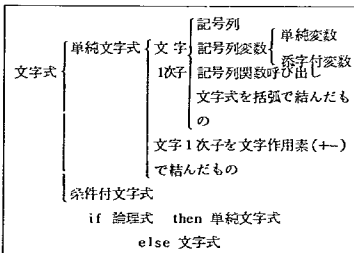


1. スイツチの添字式の値が宣言の範囲外となった時はその飛び越し文はエラーとする (ALGOL 60 では空白扱い) T-ALGOL でもエラーとする。

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が, T-ALGOL はこれらを認めず 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 $\text{vec } A[5, *] := A1, A2, \dots, \text{tor};$ 等の一般配列代入文で満たすようにしたいが, 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) T-ALGOL も同様



- 文字作用素
'ABCDE' + '1234D' は 'ABCDE'
'ABC' - '1234' は 'ABC'
- T-ALGOL では全的にこの KDK

- 配列代入文
 $\text{vec } A[1, *] := A1, A2, \dots, \text{tor};$ と宣言された範囲に逐次代入することを認める。
T-ALGOL も同じ。
- 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。
T-ALGOL もこれに従う。
- 入出力文として JIS 水準 70 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
PRINT(N, E);
PRINTR(N, E);

```

PRINTF( $n_1, n_2, E$ ):  $n_2$  は 9 までに限る
PRINTA( $N, E$ ): 文字式  $E$  の値の先頭  $N$  文字分
PRINTS('x...x'):
SPACE( $N$ ):  $N = 0$  も認める
CRLF( $N$ ):  $N = 0$  も認める
READARRAY( $K, L1, U1, L2, U2, \dots$ ):
PUTD( $N, A$ ): レベル  $N$  より配列  $A$  を 1 時格納
GETD( $N, A$ ):  $\quad\quad\quad$   $\quad\quad\quad$  に読み出し
PUTMT( $N, A$ ):  $\quad\quad\quad$   $\quad\quad\quad$  を  $MT$  に格納
GETMT( $N, A$ ):  $\quad\quad\quad$   $\quad\quad\quad$  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 階段要素 算術式 step 算術式 until 算術式
III 条件要素 算術式 while 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 手続き名<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。

T-ALGOL も同じ

2. 手続き文

```

  手続き名<実パラメタ部>;

```

3. 一個関数の時は *procedure* の前にタイプ宣言詞をつけた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
算術単変数	<i>real, integer</i>	算術式
論理単変数	<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
手続き名	<i>procedure</i>	手続き名

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、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、T-ALGOL では *double complex alpha* のみを追加する。

2. 配列宣言

添字の限界は算術式で良い。

KDK ALGOL では *alpha array* と *label array* を 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

前にものべたように 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じ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 5090H 用の NEW ALGOL-H 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。他の部文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、土本研究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かったもの 2 0 通りを CDC 3600 に実行させることもできる。CDC 3600 は従述するように現在世界最大の機械であって OKITAC 5090H の約 1/40 の時間で処理することができるとのである。

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 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 規制のない仮パラメー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 $\langle \rangle$ <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 配列代入文

$$vec\ A[2, *]:=A_1, A_2, \dots, tor;$$
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 TAP との接続を認める
- 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: に来るまでに: = が来た場合エラーとする。
- x JIS 水準 70 の入出力のはか 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 $(A+B) + E$ 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、= の後は 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おりの array に対して array の identifier でもよい。
- b $READARRAY(\overline{I}, L_1, U_1, L_2, U_2, \dots);$
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

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

また宣言したことおりの大きさの 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; としておけば建設省共通文法に対して PRINTS の (' ') を (" ") になおすだけで、他はすべて適用する。なお詳細な指定は次による。

PRINT(' <format>' , $\tilde{V}_1$, $\tilde{V}_2$ );
 <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 : $\square \pm . \underbrace{\times \times \cdots \times}_{9 \text{ ケタ}} \times 10^{\pm \times \times \times \square}$
 I : $\square \pm \times \times \cdots \times \square$
 13 ケタ
 L : $\square$ TRUE $\square$ または $\square$ FALSE $\square$
 X : レプリケーターで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(ゼロサプレスつまり始めのほうの
 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 ... x' : ' ' の中の文字そのものを印字
 このうち . 10 - + を除いた他のも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
 変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
 を印字しなければ
 REPX(N,M);
 PRINT(' XB X(' * ') / '); でよい。
 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, arr-
 ay 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 は機器, 異なったプログラム間の磁気テープによる
 受渡し。
 h RDBIND(<unit> , <unsigned integer>); 磁
 気テープ巻きもどし。
 i TSEARCH(<unit> , <unsigned integer>);
 指定テープマターサーチ。

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の後で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($\tilde{I}$); 宣言されたとおりの大きさのデー
 タに限る。テープよりarrayを列なりIに読込む。
 c Printinteger($\tilde{AE}$);
 整数を有効4ケタで印刷
 Print($\tilde{AE}$);
 実数を有効10ケタで浮動小数点型で印刷
 Prefix(AE, n_1 , n_2);
 実数を小数点上 n_1 以下 n_2 ケタで印刷
 d Printreal($\tilde{AE}$, n);
 実数を有効nケタで印刷
 Printx($\tilde{AE}$, n);
 整数を有効nケタで印刷
 e Printstring(' x 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 $\tilde{AE}$ 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 $\tilde{FP}$ と $\tilde{AP}$ のタイプは一致しなければならない
- g ↑ 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 $READ(\tilde{V}_1, \tilde{V}_2, \dots, \tilde{V}_n)$;

変数の型に合わせてデータテープより読込まれる。

- b $PRINT(\tilde{AE})$;

$\tilde{AE}$ がそれぞれの標準型で印刷準備される。

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

' ' の中の文字が印刷準備される。

- d $SPACE(\tilde{AE})$;

- e $CRLF$: 1 回復帰改行 n 回のときは

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

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

200 は番号, list は $\tilde{AE}$ をカンマでつないだもの。

format 中

Z ゼロサプレスするケタ

D ゼロサプレスしないケタ

. 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 ベーリヤン (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

' ' 文字そう入

↑ 紙の先頭より印刷開始

/ 復帰改行

□ スペース

たとえば

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

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

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

$Q, R, S)$;

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 $\tilde{FP}$ 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 $READ(\tilde{V}_1, \tilde{V}_2, \dots)$;

- b $READARRAY(I)$;

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

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 ' ' でつまなければならぬ
- 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 $\tilde{FP}$ 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 $\tilde{FP}$ と $\tilde{AP}$ 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 $READ(\tilde{V}_1, \tilde{V}_2, \dots, \tilde{V}_n)$;

変数の型に合わせてデータテープより読み込まれる。

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

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

何個書いてもよいが 6 個ごとに改行される。印刷は $\tilde{AE}$ 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は機番で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

(との両者の場合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

W 文字

WX スペース

/ 復帰改行

で次の例による。

```
OUTPUT(61, '(2B4=, E8.2, 3X,
2B4=, F2.6/)', A, B);
OUTPUT(61, '(10F6.2/)', 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
 他機種用のアルゴル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図表示も入らないの
 でNewtonの近似を用い、

x_i を $i-1$ 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

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

に対して

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

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

の条件のもとに立方根の関数呼び出しを作り、A, Bを説
 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;
  READ(A, B);
  PRINTS('A='); PRINTR(A, A);
  PRINTS('B='); PRINTR(A, B);
  PRINTR(6, CUB(A)+CUB(B));
  CRLF(1);
end;
```

19.2 NEW ALGOL-H への交換

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

```
begin
  real A, B;
  real procedure CUB(C);
    (19)のとおりにただし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 への交換

```
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社の機械向けに作られたという動機と、IBM機のプログラマといえは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日間くらい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。

表 - 1

問 題 点	FORTRAN	ALGOL	PL/I
1. ブロック構造	ない そのかわり無言で変数を使用できるが先頭の文字が1~Nのときはintegerとなる。	ある そのかわりすべての名前が宣言を必要とする。	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。	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	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	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。	可
6. 条件式	ない	真偽式、論理式、指示式に対してある。	ない
7. 配列の添字式	0・V±0の形に限る(0:定数, V:変数型変数)	条件式を含め任意	条件式を除き任意
8. 構造体	ない	ない	ある

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としなければならない。

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/I に変えてみるとPL/I がALGOLに近いものであることがわかる。

FORTRAN	ALGOL	PL/I
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;

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: 1FR $\geq$ 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問題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、筋の通った文法によって、しかも step P until Q 等の視覚にうったえる要素の多い ALGOL が適している」と結論することができよう。

21. 練習問題 (II)

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 FACOM 230 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。

```

17 K:=SQRT(LN(1/P↑2));
18 LX:=LN(X(I));
19 RVLX:=SQRT((SLXX-MLX*SLX)
20 H:=(KP+SQRT(KP↑2)-(1-
21 PRINT('I',1/P);
22 PRINT('B-5Z.2DB/',EXP(
23 MLX+H+RVLX));

```

削除

ii) 一元配置の分散分析

A NEW ALGOL-H への変換
16 PRINT('5B'4MEAN');

```

17 PRINT('8B'RV');
18 PRINT('8B'N'/'');
19 削除
20 K5:=RV(I):=SQRT((SS
21 PRINT('2(B-5Z.5DB)'/,K2,K5);
22 PRINT('7B'AOV$TABLE'/'');
23 削除
24 削除
25 PRINT('5SV'7B'SS'7B DF 7B MS
26 7B'F'/'');
27 削除
28 ♪
29 ♪
30 ♪
31 PRINT('5$A',B-4Z.4DB,B2ZDB,
32 2(B-3Z.3DB)'/,SSA,NA,FA,FO);
33 削除
34 ♪
35 ♪
36 ♪
37 PRINT('5$E'B-4Z.4DB,B2ZDB,
38 B-3Z.3DB'/',SSE,NE,FE);
39 削除
40 ♪
41 ♪
42 PRINT('15('-'/'');
43 削除
44 PRINT('5$O'B-4Z.4DB,B2ZDB'/',SSO,SO);
45 削除
46 ♪
47 ♪
48 ♪

```

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;

```

一番面倒な 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 ケタ印字しその前後に空白をつけるものである。

B. FACOM 230 ALGOL への変換

表-1 変換表

旧 [(13)II] のプログラム	新
SPACE ④⑦④②③③③③	Space
PRINTS ④⑦④③④①③③③② ③③③②⑦③③①	Printstring する
PRINTI ③③③③③	Printx としカッコ内は 3.A とあれば A, 3 に逆転
PRINTF ④④③③④②⑦④⑦③②	Printfix としカッコ内は 3.2.A とあれば A.3.2 と逆転

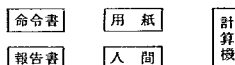
23. 電子計算機

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触れておく、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つて、模式図によって示すと次のようになる。



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、文献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う具合にはゆかない。

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・パラメ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0・1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 S)100万分の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地点の磁化の有無により0・1の状態を判別するもので呼び出し時間はミリ秒(ms)1000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似的な(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。

ディ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年のパスカルに加算機を経て、1694年ライプ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年代のバベッジの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 | 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	20.7	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組織の導入とともに普通に見ら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、橋りょう上部工、橋脚について作業を進めている。

これら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れるものである。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プログラムをご紹介します。

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% の荷重を土の厚さに換算し 0.75 m 土の重量 1800 kg/m^2 、コンクリートと土の摩擦係数 0.5 を用い、

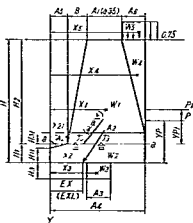


図 - 3

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 以上の引張応力を生じたとき
 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 $1 \sim 5$ 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

$$A7=A1+A2 \quad (\text{頂端と底幅の和})$$

$$W1=A7 \cdot H/2 + 2350/2$$

$$W2=H/1 \cdot A4 + 2350$$

$$W21=H/31 \cdot A5 + 2350/2$$

$$W3=A3 \cdot H/3 + 2350$$

$$W4=A6 \cdot H/2 + 2350/2$$

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

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

b 擁壁前面点Cより各重心までの距離

Y 軸関係

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

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

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

X 軸関係

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

$$X2=A4/2$$

$$X21=A5/3$$

$$X3=A4/2$$

$$X4=A4-A6/3$$

$$X5=A4-A6/2$$

c 土圧

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

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

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

$$X11=X1-A5$$

$$X41=X4-A5$$

$$X51=X5-A5$$

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

$$WT=W1+W4+W5$$

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

$$E1=A2/2-EX1$$

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

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

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

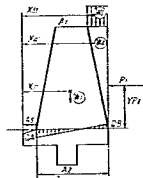


図 - 4

g 地盤反力

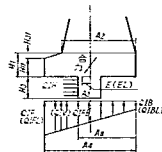


図 - 5

WL-W-W3

E=A4/2-EX

EL=A4/2-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 キーに加わる応力度

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 (SLDE $\geq$ 2)

SLIDEL=WL*0.5/P

TENTO=MOX/(MOP+QIH*H3/2)

(TENTO $\geq$ 1.5)

TENTOL=MOX/MOP

ii) 数量計算

a コンクリート

V1=A7*H2/2+H1*A4-A5*H31/2

V2=V1+H3*A3

b 型わく

KATA1=H1+H11+ $\sqrt{(B^2+H2^2)}$ + $\sqrt{(A5^2+H2^2)}$

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H, 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,
QIFB,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("=");

REPX(N);

PRINT('-.XD10-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:=1/3:=A4/2; X4:=-A4-A6/3;
X5:=-A4-A6/2; X21:=-A5/3;
MOX:=-W1*X1+W2*X2+W3*X3
+H4*X4+H5*X5-W*21*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*H3+1/2;
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("KATAVAKU1",5,KATA1);
PR("KATAVAKU2",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('1/');
PR("GORYOKUX",3,EX);
PR("GORYOKUY",3,EXL);
PR("WEIGHT",5,W);
PR("TEIKOL",5,TEIKOL);
PRINT('12/');
S;
PRINT(".....QWARI.....");
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、図面も原点のみを指示されて、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できた。一番最初にものべたように ALGOL は32の単語30の記号しか用いてないうえ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ってゆくモニタ(監視プログラム)を付加することにより、

土木研究所のOKITAG 5090 H用のNEW ALGOL-Hを用いて第3次優先のPTR(光電リーダー)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う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、プログラマー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だ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一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 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いの気易さで ALGOL によって書き下すことが可能であり、また人の習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がおすすしめたい。

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、この内容で行なった講習会の席で、入門として読み下すには適当だが、実際のプログラミ

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. 算術作用素 + - × ÷ ! (×の代りに・とするものも多い)
	2. 比較作用素 < ≤ ≥ > *
	3. 論理作用素 ¬ ∧ ∨ ⊃ ≡
	4. 順序作用素 (go to if then else for do)
	2. 区切り (. . 10 ; : ; := step until while comment)
	3. 括弧類 (() [] { }) (begin end ' ')
	4. 宣言詞 Boolean integer real own array switch procedure
	5. 特別規制詞 (string label value)

2) 略号

$\tilde{AE}$ (算術式)	$\tilde{AP}$ (実パラメタ)
$\tilde{BE}$ (論理式)	$\tilde{D}$ (宣言)
$\tilde{DE}$ (行先式)	$\tilde{EP}$ (仮パラメタ)
$\tilde{FS}$ (繰り返し文)	$\tilde{I}$ (名前)
$\tilde{L}$ (名札)	$\tilde{S}$ (文)
$\tilde{SA}$ (単純算術式)	$\tilde{SB}$ (単純論理式)
$\tilde{SD}$ (単純行先式)	$\tilde{SV}$ (単純変数)
$\tilde{SE}$ (添字式)	$\tilde{US}$ (無条件文)
$\tilde{UI}$ (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 F O O D S S V V C C Z 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、スイッチなどにつけられる (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

実数 (real) …… 少なくとも .10 の1つを含んだ数値
整数 (integer) …… .10 のいずれをも含まぬ数値

.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 .4 4.6 10-2 310-2 -4.610 5

(不可) 3. 410 3 410 9 510 25
(5*10125 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。

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

1) 変数

単純変数 (例) R PRA L3
添字付変数 (例) D[I] RV[S,T+K]
MI (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。

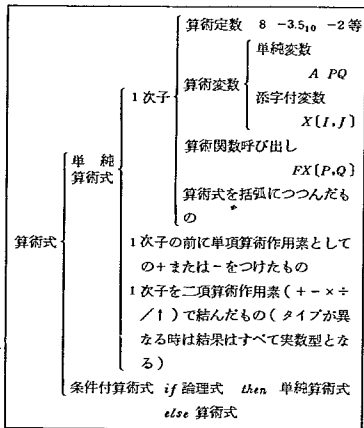
2) 標準関数, 換算関数

```
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 (AE)



- i 演算の優先順位は次のようになる。

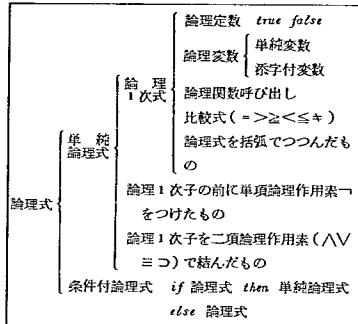
1.	()
2.	↑
3.	* / ÷
4.	+ -

- ii 算術作用素の両側のタイプの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 (BE)



- i 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. ≡

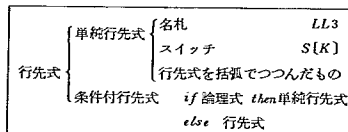
- ii ¬∧∨≡は次の結果をもたらす

¬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 (DE)



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白扱いとなる (エラーとするものも多い)。

28.3 節 (clause)

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

いずれもそのままなく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

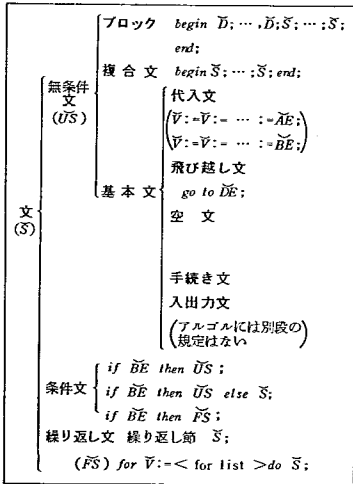
1. $\bar{A}\bar{E}$
2. $\bar{A}\bar{E}$ step $\bar{A}\bar{E}$ until $\bar{A}\bar{E}$
3. $\bar{A}\bar{E}$ while $\bar{B}\bar{E}$
4. 1~3をカンマでつないだもの

がありたとえば

```
for I := 1 step 1 until 20.22 step 2
until 50.70.100.1 * 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、
名れ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 名れ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 の前には空文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 捨 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ぬ。

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 (AP) 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 でくくっ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もので、きざみの 1/4 ~ 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	関数 { 手続き integer procedure Boolean procedure
	スイッチ宣言	switch

型	定数	単		添		関		式
		宣言	規制	宣言	規制	宣言	規制	
算術	実数 -4E0-5 等	real	real	(real)array	(real)array	real procedure	real procedure	算 術 式
整数	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)
行先	L5 等		label	switch	switch			行 先 式

1) 型宣言

宣言詞の後に単純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 ; = 行先式のなりび ;

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

1) 手続き宣言は

procedure 手続き名 < P P 部 > ;
< 値 の 部 >
< 規 制 部 >
< 手続き本体 > ;

の形をとる。

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。

宣言中仮パラメタ部は (P P , P P ……) の形をとらない場合はこの部分を省略する。値の部は P P 中値を外から値で買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味表示である。(array の value 指定を禁止するものもある)。

規制部は P 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へ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だに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線を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づ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)のよりなものである
	または
ゴシック	ALGOL の単語
$< \dots >$	$\dots$ 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へをつけた。

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

$<区切り記号> ::= <作用素> | <区切り> |$

$<括弧類> | <宣言詞> |$

$<特別規制詞>$

$<作用素> ::= <算術作用素> | <比較作用素> |$

$<論理作用素> | <順序作用素>$

$<算術作用素> ::= +|-|*|/|\div|$

($\times$ の代りに $\cdot$ を用いるものも多い)

$<比較作用素> ::= <| \leq | \geq | > | \neq$

$<論理作用素> ::= \equiv | \vee | \wedge | \neg$

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

$<区切り> ::= . | , | : | ; | := | step | until | while | comment | \neg$

($\neg$ はスペースの意で $\#$ を用いるものもある)

$<括弧類> ::= (| [| { | ' | ' | begin | end$

$<宣言詞> ::= Boolean | integer | real | array | switch | procedure |oun$

(*oun* 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

$<名前: \tilde{T} \text{ と略} > ::= <文字> | <\tilde{T}> | <文字> | <\tilde{T}>$

<単純論理式> ::= <内包> | <内包> ≡ <内包>
 <論理式: $\bar{B}\bar{E}$ > ::= <単純論理式> | <条件節>
 <単純論理式> $\bar{E}$ else < $\bar{B}\bar{E}$ >

5) 行先式 (< $\bar{D}\bar{E}$ >)

<名札> ::= < $\bar{I}$ > | <符号のない整数>
 <スイッチ名> ::= < $\bar{I}$ >
 <スイッチ> ::= <スイッチ名> [<派字式>]
 <単純行先式> ::= <名札> | <スイッチ> |
 (< $\bar{D}\bar{E}$ >)
 <行先式: $\bar{D}\bar{E}$ > ::= <単純行先式> |
 <条件節> <単純行先式> else < $\bar{D}\bar{E}$ >
 (名札として符号のない整数を禁止するものも多い)

29.4 文 statement

1) 複合文 compound statement

<名札のない基本文> ::= <代入文> |
 <飛び越し文> | <空文> |
 <手続き文> | <入出力文>
 <基本文> ::= <名札のない基本文> |
 <名札> : <基本文>
 <無条件文> ::= <基本文> | <複合文> |
 <ブロック>
 <文> ::= <無条件文> | <条件文> |
 <繰り返し文>
 <複合後部> ::= <文> end | <文> ;
 <複合後部>
 <ブロックの頭> ::= begin <宣言> |
 <ブロックの頭>; <宣言>
 <名札のない複合文> ::= begin <複合後部>
 <複合文> ::= <名札のない複合文> |
 <名札> : <複合文>
 <名札のないブロック> ::= <ブロックの頭>;
 <複合後部>
 <ブロック> ::= <名札のないブロック> |
 <名札> : <ブロック>
 <プログラム> ::= <ブロック> |
 <複合文>

(入出力文は改訂アルゴリズム 6.0 には規定されていないがもしあるとすれば名札なし基本文に入ると考えられる)

2) 代入文 assignment statement

<左辺> ::= <変数> := | <手続き名> :=
 <左辺のなりび> ::= <左辺> | <左辺のなりび>
 <左辺>
 <代入文> ::= <左辺のなりび> < $\bar{A}\bar{E}$ > |
 <左辺のなりび> < $\bar{B}\bar{E}$ >

(左辺が整数型変数の場合代入は自動的に 4 捨 5 入される)

3) 飛び越し文

<飛び越し文> ::= go to < $\bar{D}\bar{E}$ >

通常文は書かれた順序に実行されるが飛び越し文はその

行先を変更し行先式で定められた名札の次にある文を実行することを命じている。スイッチ呼び出しを用いた場合それが宣言された範囲外の時は空文扱いとなる (空文扱いでなく定義されないとするものもある。そのほうが危険が少ない)。

4) 空文 dummy statement

<空文> ::= <空>

5) 条件文

<条件節> ::= if < $\bar{B}\bar{E}$ > then
 <無条件文> ::= <基本文> | <複合文> |
 <ブロック>
 <部分条件文> ::= <条件節> <無条件文>
 <条件文> ::= <部分条件文> |
 <部分条件文> else <文> |
 <条件節> <繰り返し文> |
 <名札> : <条件文>

6) 繰り返し文

<繰り返し要素> ::= < $\bar{A}\bar{E}$ > | < $\bar{A}\bar{E}$ > step
 < $\bar{A}\bar{E}$ > until < $\bar{A}\bar{E}$ > |
 < $\bar{A}\bar{E}$ > while < $\bar{B}\bar{E}$ >
 <繰り返し要素のなりび> ::= <繰り返し要素> |
 <繰り返し要素のなりび> ,
 <繰り返し要素>
 <繰り返し節> ::= for <変数> := <繰り返し要素
 のなりび> do
 <繰り返し文> ::= <繰り返し節> <文> |
 <名札> : <繰り返し文>

繰り返し文の外から繰り返し文の中の文にある名札を指す飛び越し文の効果は定義されない。また繰り返し文の中から外に出るときはコントロール変数はその飛び出し直前の値を持っているが繰り返し要素の並びを全部終了した後ではコントロール変数の値は定義されない。

7) 手続き文 procedure statement

<実パラメタ: $\bar{A}\bar{P}$ > ::= <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> : (< $\bar{A}\bar{P}$ > のなりび) ::= < $\bar{A}\bar{P}$ > | < $\bar{A}\bar{P}$ > のなりび
 <パラメタの区切り> < $\bar{A}\bar{P}$ >
 < $\bar{A}\bar{P}$ 部> ::= <空> | (< $\bar{A}\bar{P}$ > のなりび)
 <手続き文> ::= <手続き名> < $\bar{A}\bar{P}$ 部>

手続き文の実パラメタの数と順序は対応する手続き宣言の仮パラメタの数と順序に合っていなければならない (パラメタの区切りをカンマ (,) のみで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などがある)。

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

<型のなりび> ::= <単純変数> | <単純変数> ,
<型のなりび>
<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> <型のなりび>
(*own* を認めないものも多い)

2) 配列の宣言

<下 限> ::= < $\bar{n}$ >
<上 限> ::= < $\bar{n}$ >
<上下限> ::= <下限> : <上限>
<上下限のなりび> ::= <上下限> | <上下限のなりび> , <上下限>
<配列片> ::= <配列名>
[<上下限のなりび>] |
<配列名> , <配列片>
<配列のなりび> ::= <配列片> | <配列のなりび> ,
<配列片>
<配列の宣言> ::= *array* <配列のなりび> |
<局所または占有型> *array*
<配列のなりび>

(*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) でなければならぬ。

3) スwitchの宣言

<行先式のなりび> ::= < $\bar{d}$ > |
<行先式のなりび> , < $\bar{d}$ >

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

4) 手続きの宣言

<仮パラメタ: $\bar{F}P$ > ::= < $\bar{I}$ >
< $\bar{F}P$ のなりび> ::= < $\bar{F}P$ > | < $\bar{F}P$ のなりび>
<パラメタの区切り> < $\bar{F}P$ >
< $\bar{F}P$ 部> ::= <空> | (< $\bar{F}P$ のなりび>)
< $\bar{I}$ のなりび> ::= < $\bar{I}$ > | < $\bar{I}$ のなりび> , < $\bar{I}$ >
<値の部> ::= *value* < $\bar{I}$ のなりび> ; | <空>
<規制詞> ::= *string* | <型> | *array* |
<型> *array* | *label* | *switch* | *procedure* |
<型> *procedure*
<規制部> ::= <空> | <規制詞>
< $\bar{I}$ のなりび> ; | <規制部>
<規制詞> < $\bar{I}$ のなりび> ;
<手続きの頭> ::= <手続き名> < $\bar{F}P$ 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 $\bar{F}P$ 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)。

30. ALGOL, PL/1, FORTRAN, COBOL の比較

はじめに

よくプログラム用言語の比較が問題にされる。科学技術用には ALGOL, FORTRAN 事務用は COBOL それらを総合したのが PL/1 などとよく云われる。ひどい場合には科学用 ALGOL 技術用 FORTRAN などと云われることもある。科学用, 技術用, 事務用の定義がよく分らないが, 大学の人が好んで使うから科学用, 工場の人がよく使うから技術用, 事務屋が使うから事務用というのでは話がおかしくなる。事務屋さんがよく統計解析用に使うが, これは純然たる科学技術用であろうし, FORTRAN は約束ごとが多いから自由を好む大学人にきられるというわけのものでもなかろう。

全く同じ問題をそれぞれプログラムしてその難易を比較することができればある程度はつきりするので, 適当な問題をさがしていたが, たまたま安田寿明著プログラム入門が給与計算のプログラムをはじめ FORTRAN 後で COBOL で書いているのを見て, これを後の修正の便を考えて税, 扶養などの手続きにまとめたものを4つの言語で書いてみた。

実用プログラムとしては毎月その人の本俸, 家族数などを読み込むのはばかけて居るので台帳に何級何号, 家族構成などを記入しておいて, 毎月読み込むのは残業時間だけにすべきであろうが, プログラムを長くしないために, 毎月読みこむ形をとった。

なおプログラムの難易を比較してみても一度作れば良い問題については問題にならないという声もあるが, 情勢の変化に応じてプログラム修正が可能である。必要がないならば全部外注して差し支えない筈で, 我々は来るべき情報化社会において人間機械系のパートナーとして計算機を評価するという立場に立つべきであり, そのためにはこの系における対話用語が人間に近い機械に近いかは大いに問題になる筈であると考えられる。

問 題

部課コード(6けた) LBUKA

社員番号(5けた) NO

本 係(6けた) HONPO

職 務 手 当(5けた) SHOKMU

妻帯コード(1けた) SAI (あり1, なし0)

家族コード(2けた) PKAZU (妻を除く扶養家族数)

残 業 時 間(3けた) ZTIME

の順にデータを読みこみ

1. 残業手当 ZANGYO 1時間当り本俸の0.5%
2. 扶養手当 FUYO 妻2000円, 家族一人当り1,000円
3. 所 得 SHOTOKU 本俸, 職手当, 家族手当, 残

業手当の和

4. 税 金 ZEIKIN

所得	12万 以上	10万 以上	8万 以上	5万 以上	3万 以下	3万 未満
税率%	20	18	15	10	5	0

5. 手 取 り TEDORI 所得=税金

残業時間は時間単位, 残業手当, 税金は円以下切りすてとし, 部課別にそれぞれの合計も印刷する。報告書の形式は次の通り

BUKA	NO	HONPO	SHOKMU	FUYO
xxxxxx	xxxxx	xxxxxxx	xxxxxx	xxxx
.....				
.....				
TOTAL	xxx (人数)	xxxxxx	xxxxxx	xxxx
ZANGYO	SHOTOKU	ZEIKIN	TEDORI	
xxxxxx	xxxxxxx	xxxxxxx	xxxxxxxx	
.....				
.....				
xxxxxxx	xxxxxxx	xxxxxxx	xxxxxxxx	

方針としては

1. 合計欄クリア KSW に初期値1代入
2. データ読みこみ
3. 終りのデータなら6へ KSW 1なら2として部課コードLをMに変更
4. 部課コードが変わったらトータルを印字し, クリアして2へ
5. 手当計算し, 個人別印字と部課別合計
6. おわり

ALGOL

1. 法律改正時の変更を見越して次のものは手続きにまとめ先頭におく。

a ZEI 所得による税金計算

```

1 案 条件式を用いる。 ZEI=( if S ≥ 120000
then 0.2 else if ..... ) * S - 0.5;
2 案 ベクトル式を用いる。(KDK-ALGOLの場合)
A[*]:=vec 120000, 100000, 80000,
50000, 30000 for;
B[*]:=vec 0.2, 0.18, 0.15, 0.1, 0.05, for;
ZEI:=0;
for I:=1 step 1 until 5 do
if S ≤ A[I] then
begin ZEI:=S*B[I]; go to END;
end;
END;
```

- 3 案 初期値設定を用いる (KDK-ALGOLの場合)

```

array A[1:5] initial
  vec 120000,100000,80000,50000,
    30000 for ;
  B[1:5] initial
  vec 0.2,0.18,0.15,0.1,0.05 for ;

```

として後をつづける。

いまは条件式を採用する。

- b FU 妻などの家族数による扶養手当
- c ZAN 残業時間と本俸による残業手当
- 2. 表題 印刷と合計欄のクリアをCLEAR とする。
- 3. 節課コードの変更による true となるブーリアン関数 B をつくる。
- 4. 合計と各行の印刷は最初の 2 個を除き同じ形式なので上の B を用いて配列 E (各行), T (合計) の使い分けをする手続き TOTAL を作る。

PL/1

- 1. 手続きと関数は ALGOL 同様区別がなく、ただ関数の時は自分の名前に入力するかわりに RETURN の後の括弧内に式を書く。

- 2. 条件式がないため

```

ZEI:PROCEDURE(S);
  DECLARE(A,B)(5) INITIAL(120000,
    100000,80000,50000,30000,0.2,
    0.18,0.15,0.1,0.05);
  DO I=1 TO 5;
    IF S >= A(I) THEN RETURN(S*B
      (I));
  END;
  RETURN(0);
END ZEI;

```

という形をとる。

また TOTAL も合計専用とせざるを得ない。

- 3. Boolean procedure に当るものがないのでそのたびに条件により分岐させざるを得ない。
- 4. ALGOL 同様構文にはげられない。
- 5. ALGOL 同様文の区切り; を入れる。
- 6. ALGOL 同様 IF THEN ELSE を (文に対してのみ) 用いるが ELSE の前には; が必要。
- 7. ALGOL 同様ブロック構造が存在し、すべての名前には宣言が必要。BEGIN; は DO; と同様 END; と共に文をくくるのに用いるが、宣言は BEGIN; の後に限る。
- 8. END; が重なるとき END LP; などとすると LP というレベルまでの対応 END; は入るものとみなされる。なお、レベルは ALGOL 同様名前が良い。
- 9. 混合演算、再帰の呼出、添字の下限は負も可、添字数無制限、添字部の書き方が自由、複合代入文 DO に WHILE の使用や正整数以外も自由、などすべて ALGOL と同様である。

- 10. KDK-ALGOL 同様初期値設定、文字型変数、A(I,*) などの使用等が可能

- 11. T = T + E;

などという配列全体を名前だけで構造物として扱い得る。(PL/1 の特長)

FORTRAN

- 1. 関数 ZEI と FU, ZAN を同様つづけるが、後 2 者は文関数とし、サブルーチンの後に移す。

- a ZEI

条件式とベクトル式がないので A と B にデータ文により定数を代入して DO を用いる。

- b FU

文のはじめに FU = S * 2000.0 + P * 1000.0 と混合算をさせて .0 をつける。

- 2. CLEAR

- a 配列 T にひとつずつ代入する位ならひとつづつ

THONPO などとしてこれに和をとった方が早い、ブロック構造がないので、長々とパラメタ部を書くのがいやなら COMMON とせざるを得ない。

- b THONPO などに初期値 0 を入れるのに ALGOL では

THONPO := TSHOKM := = 0;

という複合代入文があるが FORTRAN では 1 つ 1 つ書かざるを得ない。

KDK-ALGOL では

T(*) := vec 0 for ;

または

array T(1:7) initial 0;

が許される。

- 3. TOTAL

各行と合計欄の区別によって印刷様式を変更することは条件式がないので二度書く手数と変わなくなる。したがって合計欄専用のものとする。

- 4. 先頭から 6 字分は特別な意味をもつほか、データがフォーマットに従ったものになっている必要がある。

- 5. 論理式は算術式同様混合算を許さない。

COBOL

- 1. 関数がないが ALGOL の procedure に似た PERFORM があるのでこれにまとめて最後に 1 つ変更に応じ得るようにする。

ただしパラメタ部が設けられないので、条件によって GO TO を行なうてやるより仕方がない。

SUBROUTINE とちがつて共通の名前はそのまま使用できるので COMMON の必要はない。

- 2. CLEAR

PERFORM TOTAL-CLEAR VARYING I

FROM BY 1 UNTIL 1 GREATER 8.
TOTAL-CLEAR.

MOVE ZERO TO TOTAL-WORK(1).

などと単に配列TOTAL-WORKの8要素に0を入れるだけでも大変で、しかもこのような1時作業部分も

WORKING-STORAGE SECTIONで

17 TOTAL-WORK PICTURE 9(10)

OCCURS 8 TIMES.

などとその書式指定まで必要とされる(今は10けた)

3. 条件式がないので

IF WORK-AREA(6) IS NOT LESS
THAN 30000 MOVE 0.05 TO WORK-3.

IF WORK-AREA(6) IS NOT LESS THAN
50000 MOVE 0.10 TO WORK-3.

などとひとつひとつ書かなければならない。

4. 文は11個まで、それ以外は7個まで特別な意味を持つほか、=+* />< は前後は1文字以上あけなければならぬとか、小数点以外の、や、;)の前には空白を置いてはならず後には空白をおかなければならないとか、(の前には1字以上の空白をおき、後には空白を置いてはならないといった機械的な制約が多い。

5. IS NO DATA等約320個の保留語があるため、いままでの社員番号に使っていたNOが使えなくなってNAとせざるを得なかった。

6. データの記述で読み込み用、計算、印字用のFILE, RECORDが別々に必要

7. カナを扱うことはコボルに限らずALGOL, PL/1, FORTRANでもそのように作れば可能である。

ALGOL

```

begin
  comment PROGRAM TEST OF KYUGO KEISAN;
  boolean procedure B;
    RETURN(LBUKA);
  array F(1:7);
  integer I, J, IZU, KSW, LBUKA, HBUKA, ND;
  real AE, HONPO, SHOKU, SAI, PKAZU, FUYO, ZANGYO, ZTIME, SHOTOK, ZEIKIN;
  integer procedure ZEI(S);
    real S;
    ZEI:=if S<120000 then 0.2 else if S<210000 then 0.18 else
      if S<240000 then 0.15 else if S<250000 then 0.1 else
      if S<30000 then 0.05 else 0;S-0.5;
  integer procedure FUI(S,P);
    integer S,P;
    FUI:=0.000+P*1.000;
  integer procedure ZANIT,H;
    integer H;
    ZANIT:=0.005+H-0.5;
  procedure CLEAR;
    begin
      procedure A(N,S); integer N; string S;
        begin
          SPACE(N); PRINT(S); end;
          for I:=1 step 1 until 7 do T(I):=0;
          NINZU:=0; NEWPAGE;
          A(4,"BUKA"); A(7,"NO"); A(5,"HONPO"); A(3,"SHOKU");
          A(5,"FUYO"); A(5,"ZANGYO"); A(5,"SHOTOK"); A(6,"ZEIKIN");
          A(7,"TEDORI"); CRLF(3);
        end;
      procedure TOTAL(A);
        real A;
        begin
          if B then begin SPACE(4); PRINT("TOTAL"); end
          else PRINT(6,LBUKA);
          PRINT(6,if B then NINZU else NO);
          for I:=1 step 1 until 7 do
            PRINT(12,if I<4 then 0 else if I=5 then 6 else
              if I=6 then 8 else 10,A(I));
          CRLF(2);
        end;
      CLEAR; KSW:=1;
      READIN;
      READ(LBUKA,ND,HONPO,SHOKU,SAI,PKAZU,ZTIME);
      if LBUKA=0 then begin TOTAL(7); go to END; end;
      if KSW=1 then begin KSW:=2; MBUKA=LBUKA; end;
      if B then begin TOTAL(1); CLEAR; HBUKA=LBUKA; end;
      ZANGYO=ZANIT*(ZTIME-HONPO);
      FUYO=FUI(SAI,PKAZU);
      SHOTOK=HONPO+SHOKU+FUYO+ZANGYO;
      ZEIKIN=ZEI(SHOTOK);
      for AE=HONPO,SHOKU,ZANGYO,FUYO,SHOTOK,ZEIKIN,SHOTOK-ZEIKIN do
        begin
          E(I):=AE; I:=I+1; end;
          for I:=1 step 1 until 7 do
            T(I):=E(I);
          NINZU=NINZU+1;
          TOTAL(E);
          go to READIN;
        end;
      END;
    end;

```

PL/1

```

KYUJO: PROCEDURE OPTIOUS(MAIN);
/* PROGRAM TEST OF KYUJO KEISAN */
DECLARE (T1),F,HONPO,SHOKU,SAI,PKAZU,FUYO,ZANGYO,
      ZTIME,SHOTOK,ZEIKIN,TEDORI) FLOAT,
      I,NINZU,KSU INITIAL(1),LBUKA,KBUKA,NO) FIXED;
FU: PROCEDURE (S,P) FIXED;
DECLARE (S,P) FLOAT;
RETURN(S*2007+P*1000);
END FU;
ZEI: PROCEDURE (S) FIXED;
DECLARE S FLOAT;
/* PHS) INITIAL
   (120000,100000,80000,50000,30000,0,2,0.18,0.15,0.1,0.05)
DO I=1 TO 5;
  IF S=S(X(I)) THEN RETURN(S(I));
END;
RETURN(0);
END ZEI;
ZAN: PROCEDURE (T,H) FIXED;
DECLARE (T,H) FLOAT;
RETURN(H*0.005*T);
END ZAN;
CLEAR: PROCEDURE;
T=0;
PUT EDIT (PAGE,X(4),*EUKA*,X(7),*NO*,X(5),*HONPO*,
      X(3),*SHOKU*,X(5),*FUYO*,X(5),*ZANGYO*,X(5),
      *SHOTOKU*,X(8),*ZEIKIN*,X(6),*TEDORI*,SKIP(2));
END CLEAR;
TOTAL: PROCEDURE;
PUT EDIT ('TOTAL',NINZU,T(1) D3 I=1 TO 7)
      (X(3),X(5),F(9),(2)F(10),F(9),F(11),
      (3)F(13),SKIP(2));
END TOTAL;
CALL CLEAR;
READIN:
GET LIST (LBUKA,NO,HONPO,SHOKU,SAI,PKAZU,ZTIME);
IF LBUKA=0 THEN
  DO CALL TOTAL; GO TO LEND; END;
IF KSU=1 THEN DO KSU=2; KBUKA=LBUKA; END;
IF LBUKA=KBUKA THEN
  DO CALL TOTAL; CALL CLEAR; END;
ZANGYO=ZAN(ZI,RE,HONPO);
FUYO=FU(SAI,PKAZU);
SHOTOK=HONPO-SHOKU+FUYO-ZANGYO;
ZEIKIN=ZEI(SHOTOK);
TEDORI=SHOTOK-ZEIKIN;
BEGIN;
DECLARE E(7) FLOAT;
I=1;
DO RE=HONPO,SHOKU,ZANGYO,FUYO,SHOTOK,ZEIKIN,TEDORI;
  E(I)=E;
  I=I+1;
END;
T=E;
NINZU=NINZU+1;
PUT EDIT (LBUKA,NO) (SKIP(2),F(6),F(9));
PUT EDIT (E(1) D3 I=1 TO 7) ((2)F(10),F(9),F(13),
      (3)F(13));
GO TO READIN;
LEND;
END KYUJO;

```

FORTRAN

```

/JOE      FIC
/Exc      PROGRAM TEST OF KYUJO KEISAN
C          FUNCTION TZEI(I)
            DIMENSION A(5),B(5)
            DATA A/123000,100000,80000,50000,30000/,
            1      0,0,15,0,15,0,10,0,05/
            TZEI=0
            DO 7 I=1,5
            IF (S-A(I)) 7,8,8
            8 TZEI=S+B(I)
            GO TO 9
            7 CONTINUE
            9 RETURN
            END

/Exc      FIC
SUBROUTINE CLEAR
COMMON THONPO,TSHOKM,TZAN,TFUYO,
1      TSHOT,TZEI,TTEDO,ITININ
THONPO=0
TSHOKM=0
TZAN=0
TFUYO=0
TSHOT=0
TZEI=0
TTEDO=0
ITININ=0
WRITE(6,1)
1      FORMAT(1H,4X,4HBUKA,7X,2HND,5X,
2      1 SHOHPO,2X,7HSHOKUMU,5X,4HFUYO,
3      5X,6HZANGVO,5X,5HSHOTOKU,2X,
4      6HZEIKIN,6X,6HTEDORI//)
RETURN
END

/Exc      FIC
SUBROUTINE TOTAL
COMMON THONPO,TSHOKM,TZAN,TFUYO,
1      TSHOT,TZEI,TTEDO,ITININ,MINZU
WRITE(6,12)ITININ,THONPO,TSHOKM,
2      TFUYO,TZAN,TSHOT,TZEI,TTEDO
12     FORMAT(1H,3X,3HTOTAL,19,2F10,0,F9,0,F11,0,
3      1 3F13,0//)
RETURN
END

/Exc      FIC
COMMON THONPO,TSHOKM,TZAN,TFUYO,TSHOT,
1      TZEI,TTEDO,ITININ,MINZU
IF(IIS,PJ=5*2000,0+P=1000,0
IZANIT,7)=HND,005+I
CALL CLEAR
KSW=5
11 READ(5,2)LRUKA,NO,HONPO,SHOKM,SAI,PKAZU,ZTIME
2      FORMAT(16,15,F6,0,4F5,0)
IF(LRUKA,NO,0)GO TO 9
GO TO (3,4),KSW
3      KSW=2
HBUKA=LRUKA
ZANGVO=IZANITZIME,HONPO)
FUTO=IFUISAI,PKAZU)
SHIUK=HONPO+SHOKM+FUYO+ZANGVO
ZEIKIN=TZEI(SHOTOK)
TEDORI=SHOTOK-ZEIKIN
THONPO=THONPO+HONPO
TSHOKM=TSHOKM+SHOKMU
TZAN=TZAN+ZANGVO
TFUYO=TFUYO+FUYO
TZEI=TZEI+ZEIKIN
TTEDO=TTEDO+TEDORI
MINZU=MINZU+1
WRITE(6,10)LRUKA,NO,HONPO,SHOKMU,FUYO,
1      ZANGVO,SHOTOK,ZEIKIN,TEDORI
10     FORMAT(1H,18,19,2F10,0,F9,0,F11,0,3F13,0)
GO TO 11
5      CALL TOTAL
CALL CLEAR
GO TO (6,15),KEND
9      KEND=2
GO TO 5
15     STOP
END

```

COBOL

```

IDENTIFICATION DIVISION.
PROGRAM-ID. KYUJO-KEISAN.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. TOSBAC-3400-B.
OBJECT-COMPUTER. TOSBAC-3400-B.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT CARD-FILE ASSIGN TO IN.
    SELECT PRINT-FILE ASSIGN TO OUT.

DATA DIVISION.
FILE SECTION.
FD CARD-FILE
    DATA RECORD IS CARD-REC.
01 CARD-REC.
    02 LBUKA PICTURE 9(6).
    02 NA PICTURE 9(5).
    02 HONPD PICTURE 9(6).
    02 SHOKMU PICTURE 9(5).
    02 SAI PICTURE 9.
    02 PKAZU PICTURE 9(12).
    02 ZTIME PICTURE 9(3).
    02 FILLER PICTURE X(52).
FD PRINT-FILE
    DATA RECORD IS PRINT-REC.
01 PRINT-REC PICTURE X(119).
WORKING-STORAGE SECTION.
77 MBUKA PICTURE 9(10) VALUE ZERO.
77 I PICTURE 9(10).
77 J PICTURE 9(10).
01 MY-1.
02 TOTAL-WORK PICTURE 9(10) OCCURS 8 TIMES.
01 MY-2.
02 WORK-AREA PICTURE 9(10) OCCURS 8 TIMES.
01 HEADER.
    02 FILLER PICTURE X(91) VALUE #1 BUKA NO HON
       #PD SHOKMU FUD ZANGYO SHOTOKU ZEIKIN
       # TUDURA.
01 PRINT-LINE.
    02 FILLER PICTURE X VALUE #0.
    02 LP PICTURE Z(10) OCCURS 9 TIMES.
01 TOTAL.
    02 FILLER PICTURE X(11) VALUE #0 TOTAL.
    02 T PICTURE Z(10) OCCURS 8 TIMES.

PROCEDURE DIVISION.
OPEN INPUT CARD-FILE.
OUTPUT PRINT-FILE.
PERFORM CLEAR.
YONIKOME.
    READ CARD-FILE AT END GO TO TOTAL-PRINT.
    IF MBUKA IS EQUAL TO ZERO MOVE LBUKA TO MBUKA.
    IF MBUKA IS EQUAL TO LBUKA GO TO KEISAN.
    MOVE LBUKA TO MBUKA.
    PERFORM TOTAL-PRINT THRU CLEAR.
KEISAN.
    MOVE LBUKA TO LP (1).
    MOVE NA TO LP (2).
    MOVE HONPD TO LP (3).
    MOVE SHOKMU TO LP (4).
    COMPUTE WORK-AREA (4) = HONPD * 0.005 * ZTIME.
    COMPUTE WORK-AREA (5) = SAI * 2000 + PKAZU * 1000.
    ADD HONPD SHOKMU WORK-AREA (4) WORK-AREA (5)
        GIVING WORK-AREA (6).
    MOVE ZERO TO WORK-3.
    IF WORK-AREA (6) IS NOT LESS THAN 10000 MOVE 0.05 TO WORK-3.
    IF WORK-AREA (6) IS NOT LESS THAN 50000 MOVE 0.10 TO WORK-3.
    IF WORK-AREA (6) IS NOT LESS THAN 80000 MOVE 0.15 TO WORK-3.
    IF WORK-AREA (6) IS NOT LESS THAN 100000 MOVE 0.18 TO WORK-3.
    IF WORK-AREA (6) IS NOT LESS THAN 120000 MOVE 0.20 TO WORK-3.
    MULTIPLY WORK-AREA (6) BY WORK-3 GIVING WORK-AREA (7).
    SUBTRACT WORK-AREA (7) FROM WORK-AREA (6)
        GIVING WORK-AREA (8).
    ADD HONPD TO TOTAL-WORK (2).
    ADD SHOKMU TO TOTAL-WORK (3).
    ADD 1 TO TOTAL-WORK (1).
    PERFORM LP-MOVE VARYING I FROM 4 BY 1 UNTIL
        I GREATER 8.
    WRITE PRINT-REC FROM PRINT-LINE.
    GO TO YONIKOME.
TOTAL-PRINT.
    PERFORM TOTAL-PRINT.
    GO TO QWARI.
TOTAL-PRINT.
    PERFORM TOTAL-MOVE VARYING I FROM 1 BY 1 UNTIL
        I GREATER 8.
    WRITE PRINT-REC FROM TOTAL.
CLEAR.
    PERFORM TOTAL-CLEAR VARYING I FROM 1 BY 1 UNTIL
        I GREATER 9.
    WRITE PRINT-REC FROM HEADER.
TOTAL-CLEAR.
    MOVE ZERO TO TOTAL-WORK (1).
    TOTAL-MOVE.
    MOVE TOTAL-WORK (1) TO T (1).
LP-MOVE.
    ADD 1. I GIVING J.
    MOVE WORK-AREA (1) TO LP (J).
    ADD WORK-AREA (2) TO T (1).
QWARI.
    CLOSE CARD-FILE,PRINT-FILE.
    STOP RUN.

```

SCBEND

31. 練習問題

紙テープに回数 N を先頭に N 個のデータが入っている。
これを読んで最大値, 最小値を求め次の形で印字せよ
(有効 6 桁で印字)。

MAX = ...

MIN = ...

2. 任意形の面積を求めるのに矩形と三角形に区分して計算した後で合計するプログラムをつくれ。

ただし, 各辺の長さのみしか判定できなかったとし, 各々の面積を計算する部分は *procedure* を使用すること。

短 形 $A = L_2 \cdot L_3$

三角形 $S = \frac{1}{2}(L_1 + L_2 + L_3)$

$A = \sqrt{S(S-L_1)(S-L_2)(S-L_3)}$

3. N 個のデータ ($A[I]$) があたえられたとき, このデータを大きさの順 ($B[I]$) にならべかえ, それが何番目のデータ ($C[I]$) かを求める手続きをつくれ。

4. 1 組のデータ $X_1, X_2, \dots, X_N$ のヒストグラムを求める手続きをつくる。ただし区間を $D1, DK$ とし, それを K 等分するものとする。

32. 練習問題解答

```

1. begin
    integer I, N;
    READ(N);
    begin array A(1..N);
        real MAX, MIN;
        READARRAY(A, 1, N);
        MAX := MIN := A[1];
        for I := 2 step 1 until N do
            begin real P;
                P := A[I];
                if P > MAX then MAX := P;
                if P < MIN then MIN := P;
            end;
        PRINTS('MAX='); PRINTR(6, MAX);
        CRLF(1);
        PRINTS('MIN='); PRINTR(6, MIN);
        CRLF(1);
    end;
end;

```

```

2. begin real SA, A, L1, L2, L3;
    procedure AREA(D1, D2, D3, A);
        real D1, D2, D3, A;
        begin real S;
            if D1 ≠ 0 then go to LL;
            A := D2 * D3; go to END;
        LL: S := (D1 + D2 + D3) / 2;
            A := SQRT(S * (S - D1) * (S - D2) * (S - D3));
        END;
    end;
    SA := 0;
    ST: READ(L1, L2, L3);
    if L1 = 0 ∧ L2 = 0 ∧ L3 = 0 then go to OWARI;
    AREA(L1, L2, L3, A);
    SA := SA + A; go to ST;
    OWARI:
    PRINTS('SA='); PRINTR(5, SA);
    CRLF(1);
end;

```

3. procedure ORDER (N, A, B, C);

```

integer N; array A, B;
integer array C;
begin integer I, J;
    for I := 1 step 1 until N do
        begin C[I] := I;
            B[I] := A[I];
        end;
    for I := 1 step 1 until N-1 do
        for J := I+1 step 1 until N do
            if B[J] > B[I] then
                begin real P; integer M;
                    P := B[I]; M := C[I];
                    B[I] := B[J]; C[I] := C[J];
                    B[J] := P; C[J] := M;
                end;
        end;
end;

```

4. procedure HSTGRM(N, D1, DK, K, X, A);

```

integer N, K; real D0, DK; array X;
integer array A;
begin integer I, J;
    real H;
    procedure CUNT(X, Y);
        real X, Y;
        if X ≤ Y then
            begin A[J] := A[J] + 1; go to END; end;
        A[*] := dec 0 for;
        H := (DK - D1) / K;
        for I := 1 step 1 until N do
            begin
                for J := 1 step 1 until K do
                    CUNT(X[I], D1 + J * H);
            end;
        END;
    end;
end;

```

※ KDK - ALGOLによる

```

for I := 1 step 1 until N do A[I] := 0;
と同じ効果である。

```

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に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認めることにより、他の技術分野と同じ様にこの利器をパートナーとして活動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 慶一：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 O 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・宮川地：一般応用統計学 川島書店
4. 岩田 統一：経済分析のための統計的方法 東洋経済
5. 中村 慶一：技術者のための統計解析 山海堂
6. 中村 慶一：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

索

引

あ行

ALGOL マニアル	45
アルファモード	29
演算装置	40
エラーチェック	29

か行

改定アルゴル 60	51
型	30
型 宜 言	32, 50
括 弧 類	4, 47
換 算 関 数	4, 48
関 数 宜 言	12
関数呼び出し	22
仮パラメタ ($\tilde{FP}$)	11, 32
記 憶 装 置	40
記 号 列	47
基 本 文	14, 31, 49
基 本 記 号	47
規 制 制	20
規 制 詞	20
規 制 部	20
区 切 り	4, 47
区 切 り 記 号	4, 47
繰り返し文 ($\tilde{FS}$)	12, 15, 31, 49
繰 り 返 し 節	49
空 文	14
KDK-ALGOL	30
コロニクオル (:-)	5
コ メ ン ト	49, 52

さ行

作 用 素	47
算 術 作 用 素	4, 30, 47, 48
算 術 式 ($\tilde{AE}$)	11, 12, 30, 47, 48
CRLF	5, 14
条 件 節	49
条件付算術式	12, 30, 48
条件付論理式	13, 31, 48
条件付行先式	13, 31, 48
条件付文字式	31
条 件 文	14, 31, 49
式 ($\tilde{AE}$)	10, 12, 30, 47
実パラメタ ($\tilde{AP}$)	11, 32
順序作用素	4, 47
出 力 装 置	40

ス イ ッ チ	24
スイッチ宣言	12, 24, 50
SPACE	14
数	47
数 字	10, 47
制 御 装 置	40
節	49
宜 言	10, 11, 12, 32, 52
宜 言 詞	4, 47
宣言の範囲	7

た行

単純変数宣言	12
単 純 算 術 式	12, 30, 48
単 純 論 理 式	13, 31, 48
単 純 行 先 式	13, 31, 48
単 純 文 字 式	31
単 純 変 数	47
代 入 文	13, 49
手 続 き	6, 14, 19
手 続 き 宜 言	12, 15, 50
手 続 き 文	49
添字付き変数	47
添字付き変数宣言	12
電子計算機の歴史	40
飛び越し文 ($\phi\phi$ 文)	11, 13
特 殊 記 号	10
特 別 規 制 詞	4, 47

な行

名 札	7
名 前	12, 47
入 出 力 文	14
入 力 装 置	39

は行

配 列	6, 7
配 列 宜 言	32, 50
配 列 代 入 文	31
標 準 関 数	4, 7, 30, 48
比 較 作 用 素	4, 47
文	5, 12, 13, 31, 49
複 合 文	5, 11, 13, 31, 49
プ ロ グ ラ ム	10
プ ロ ッ ク	11, 13, 31, 49
変 数 ($\tilde{V}$)	12, 47

<i>for list</i>	15,32
<i>FORTRAN</i>	1,37
<i>PL/1</i>	37
<i>PRINT I</i>	5,14
<i>PRINT R</i>	5,14
<i>PRINT F</i>	14
<i>PRINT S</i>	6,14
<i>procedure</i>	6
<i>VALVE</i> 指定	20
ま行	
文 字	10,47
文 字 式	31

文 字 作 用 素	31
無条件文 (<i>US</i>)	12,13,31,49
や行	
行 先 式 (<i>DE</i>)	11,13,31,47,48
ら行	
論 理 値	4,31,47,48
論 理 式	11,13,47
論 理 作 用 素	4,31,47
<i>READ</i>	5,14
<i>READARRAY</i>	14