

40 - 3
電子計算
技術管理

土 木 研 究 所 資 料

第 102 号

ALGOLによるプログラミング

昭和40年10月

建 設 省 土 木 研 究 所

アルゴルによるプログラミング

目 次

はじめに	1
1° 電子計算機の言葉	3
2° 電子計算機の概要	5
2.1 電子計算機の構成	5
2.2 入力装置	6
2.3 制御装置	6
2.4 演算装置	7
2.5 記憶装置	7
2.6 出力装置	8
2.7 電子計算機の歴史	9
2.8 各社の電子計算機	10
3° ALGOL	11
3.1 概 要	11
3.2 ALGOLの基本原則	13
3.3 アルゴルによる簡単なプログラム	15
4° 基本定義	27
4.1 記 号	27
4.2 名 前	29
4.3 数	30
4.4 変 数	30
5° 宣 言	31
5.1 型宣言	31
5.2 配列宣言	31
6° 式	33
6.1 算術演算子	33

6.2	関係演算子	34
6.3	論理演算子	35
6.4	算術式	35
6.5	関 係	36
6.6	ブール式	36
6.7	指示子	37
7°	命 令	39
7.1	代入命令	39
7.2	FOR命令	40
7.3	GOTO命令	41
7.4	条件付命令	42
7.5	空命令	44
8°	ALGOLによるプログラム例(1)	45
8.1	作 表	45
8.2	分 類	46
8.3	マトリクスの取扱い	48
8.4	スイッチの取扱い	56
8.5	一元配置の分散分析	58
9°	手 続	61
9.1	手続命令	61
9.2	関数指示子	66
10°	そ の 他	68
10.1	コメント	68
10.2	FOR S の出口	69
10.3	own	69
10.4	条件付FOR命令	69
10.5	push down list	70
11°	I/O 命令	71

11.1	OKITAC 5090H 'H2'	71
11.2	OKITAC 5090H 'H1'	74
11.3	JUSE ALGOL 'J'	75
11.4	ISSP ALGOL 'I'	76
11.5	FACOM 230 '230'	77
11.6	OKIPAL 'PAL'	78
11.7	ALCORグループ 'A'	79
12°	アルゴルによるプログラム例(2)	80
12.1	危険度の評価	80
12.2	一元配置の分散分析	82
12.3	平均及び不偏分散の平方根の手続	85
12.4	積和平方和マトリクス及び相関マトリクスを求める手続	87
13°	アルゴルによる有用な諸手続	96
13.1	フーリエ係数	96
13.2	最大, 最小, 絶対値の最大を求める手続	97
13.3	arcsine の手続	98
13.4	DEGREE より RADIAN への変換	99
13.5	RADIAN より DEGREE への変換	100
13.6	二次方程式の根を求める手続	100
14°	他の特殊な取扱い	102
14.1	アルファモードの取扱い	102
14.2	倍精度演算	104
15°	プログラムエラーのチェック	105
16°	アルゴルテープの作り方	108
	おわりに	111
	参考文献	112

アルゴルによるプログラミング

中 村 慶 一 *

は じ め に

電子計算組織の普及が急速に行なわれ、日本の設置台数もアメリカに次いで世界第2位となつた模様で、あちこちの会社や組織に導入された話が出ると、きまつて「当方では何を電子計算機にかけたらよいか検討中です」とか遂には「計算機に何をやらせたら良いのですか」といつた質問にさえ出合う始末である。

この様なときは逆に「あなたは何も問題をお持ちでないのですか」と聞き返したくなるが、事実我々の論理的な仕事は全部計算機に実行させることが可能で、不可能なのは非論理的な、あるいはそれまでの結果と過去の経験とから人に教えられないような方法で判断を下して行かなければならない仕事だけである。

もつとも人によつては2000個までの集計はソロバンの方が早いなどという事を信仰のように唱えて電子計算組織導入の時期尚早を主張される。これはおそらく、十数年前に駐留軍の兵隊と日本青年の間で行なわれた電動計算機対ソロバンの計算コンクールの結果を思い出してのことであろうと想像される。

たしかに一回だけある個数以下の値の和をとり、平均を出すといった作業に対しては、ソロバンの方が早いことは事実で、我々電子計算に従事している人間も、ソロバンを良く利用させて貰っている。しかし「ある一群のデータが手に入つたとき、その平均値だけを出して、それで終りといった取り扱いですむ場合がどれだけあるか」という事が問題である。

「自分の所のデータの取り扱いは、平均値を出すことがほとんどだ」と云われる所では、データのせつかくもたらしてくれた情報の大部分を、みすみす捨てて居られることになり、データを取るに要した費用と考え合わせると、大変勿体ないことになる。

もしこの一群のデータから、位置の情報(その代表値が算術平均値

* 土木研究所技術管理室長工博

$\bar{x} = \sum x/n$)のほかにバラツキの情報(不偏分散の平方根

$\sqrt{V} = \sqrt{\sum (x - \bar{x})^2 / (n - 1)}$ など)とか、他の因子との関連性の情報

(相関係数など)を計算することが望ましいとすれば、これらの計算は既にプログラムも出来ていることであるし、電子計算の方が問題なく速いわけである。

しかし電子計算機の本領は、何も四則演算を高速度で実行するというだけにあるのではなく、我々が部下に仕事の手順を教えるのと同じ要領で、もしこの値がこれ以上だつたらこうするといった判断の基準も含めて教えておくことによつて、ある筋道(それはいくつかに分岐していてもかまわない)に従えば良いような問題は、それがどのように複雑なものであつてもすべて実行させ得る点にある。

従つて我々が手順を人に教えることが出来るような仕事はすべて電子計算機に教え、しかもくりかえし出てくる手続はあらかじめ機械に教えて置いて、名前で呼び出して利用できるのであるから、是非電子計算機を導入すべきであると考えて電子計算機関係の本を読んでみると、そのほとんどは電気回路の説明のようなものから始まつて、わけのわからない機械の言葉の説明に終始しているため、電子計算機は難かしいということになつてしまひ勝ちである。

昔はそうであつたかも知れないが、現在では、我々の日常の言葉に近い A L G O L で書き下して、機械の言葉への翻訳から先の仕事はすべて機械自身にやらせる事ができ、土木研究所においても既に所内 300 名、地建本省等 150 名の人達が、この方式で自分の仕事の助手として電子計算機を利用しつつあり、この A L G O L の習得は極めて容易なので、以下 A L G O L によるプログラミングを中心として説明を試みたい。

1° 電子計算機の言葉

電子計算機は人間なら2年もかかる様な計算を数時間で仕上げてしまうとか、いろいろな症状や検査結果によつて病名を診断するとか、とにかく大変な能力を持っていることはたしかであるが、実のところ電子計算機自身は、仕事の手順や判断の基準を教えて貰わない限りは、何も自から仕事を進める能力を持っていないのであつて、ここに問題の持ち主である人間と、忠実ではあるが命令に従つて仕事を実行するだけの電子計算機との間の会話を、いかにうまく進めるかということが重要なポイントになつてくる。

機械には設計のときに与えられた自分自身の言葉 機械語 machine language があり、たとえば加算せよという命令が10進で表現して02であつたとすれば、2365番地に入っている情報をとり出してきて、ソロバンに入っている数字に加えよという命令は022365などという形を取り、機種ごとに異つた、人間にとつては覚えにくい言葉を使つて、しかも横丁のポストに手紙を入れてこいと云いたいのに、右足を出せ、重心が右足を通過したら左足を出せといった面倒なやり方で教えてやる必要がある。

そのため計算手順の指示プログラミング programming の段階で大変面倒な手続きをふまねばならず、電子計算機を使うのは難かしいとか、プログラミングは高度の専門的知識を必要とする特別の人達の仕事であるといった誤解がある。たしかに5年前まではその通りであり、我々問題を持つ側の人々が、プログラマという専門知識を持った人に問題の内容を話してプログラムに作つて貰うとか、SIP, SAPなどという中間語を用いて、前の02の代りに、A (add 加えよという言葉と多少の関連を持つている)、2365番地という絶対番地の代りに B+5番地という相対番地を使つて貰へておきても、面倒な作業である点はあまり変りはなかつた。

1956年IBM社の704機に対し FORTRAN (formula translatorの略) という言語が開発され、これによつて人間の言葉と云うよりは、機械の側に大分近いものであつた。1958年このFORTRANの長

所を伸ばし、文法的にもすつきりした国際的なプログラム言語ALGOL-58が生まれ、更に1960年及び1962年に手を加えて改訂ALGOL-60が世界的に行なわれるようになった。

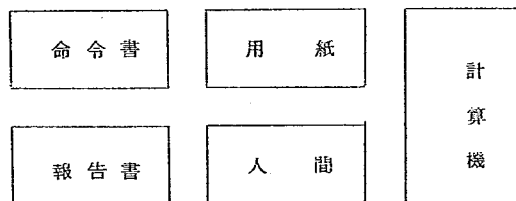
従つて現在では電子計算機を利用したい人は誰でも、この簡単なALGOL-60（以下簡単のためにALGOLと呼ぶ）の文法を習得することによつて、ほとんど電子計算に関する予備知識なしに、自分の問題を書きおろし、計算機に翻訳ならびに演算を実行させることが容易になった。

最初編集部からの依頼は、電子計算機について入門的な記事を書くようにということであつたが、以上のような事情で、本文ではALGOLによるプログラムの書き方について重点を置き、電子計算機はALGOLの理解に必要な程度触れるに止めたい。

2° 電子計算機の概要

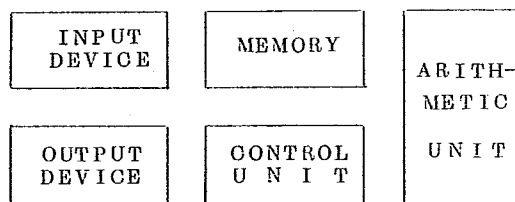
2.1 電子計算機の構成

我々が問題を与えられると、これを用紙に書きとめ、手廻し或いは電動計算機を使つて計算し、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。この手続きを図化すると



ということになる。

電子計算機を使つて計算することも、実はこれと同じことで、模式図を使つて示すと次のようになる。



たゞ人間が行なう場合には、命令書にすべての計算手続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、お前の知つてゐる方法でやれとか、場合によつては、もつと省略されたやり方で指示され、宜しくやれという云い方もあり得るが、電子計算機の場合は、すべての手順を入力装置を通じてやらなければ腹と腹で宜しくやるという工合には行かないという点が異つてゐる。

しかし前もって教えておいた方法呼び出して必要なパラメタを代入し、得た結果を利用させることは可能で、この意味では段々と機械に学習させることができ、判断の基準があいまいでない限りは、ある意味で宜しくやれといった指令を出すこともできるわけである。

このような能力を持った電子計算機を使いこなすにはプログラムの作り方さえ心得て居れば良いわけであるが、良いプログラムを作るための機械の構成に関する最小限の知識をのべておく。

2.2 入力装置 input device

計算機の目に当るが、人間の書いた文字を判読させることは実用段階ではまだ困難で、紙テープやカードにパンチされた情報を光電トランジスタなどで読み取る方式のものが多い。

紙テープの場合は5～8単位の孔で1字を構成し、毎秒200～1000字の読取りを行ない、修正は修正紙によつて、ちょうどフィルムの編集の様に容易に行なうことができる。

カードの場合は1枚80～90字のものを毎秒4枚(ブラシ式)から10枚(光電式)位で読み取る。

2.3 制御装置 control unit

各装置とそれらの間を連絡している情報路の間のスイッチングを制御する部分で、機械の本体をなすものであるが、これと次の演算装置、記憶装置については利用者が直接関連を持たない部分であつて、別段詳しく知つていゝる必要はない。

たゞこれらの装置の間を、情報は正負のパルスという形で伝送され、我々が10進 decimal の形で入力したものを2進 binary の形で処理し、又10進の形で出力してくるものが多いといった程度のことを心得ていゝれば十分であらう。

0か1かという2つのうちの1つを選ぶ情報の最小単位を binary

digit 略してビット bit と云い、この計算機では1語43ビットなどと表現する。2進で40ビット程度のものは10進で12ケタ位の情報に当り、1語40ビット程度のものを長語 long word、20ビット前後のものを短語 short word といい、同じ4000語(これを4K語という)といつても長語と短語では容量が倍違うので注意を要する。

2.4 演算装置 arithmetic unit

いわばソロバンに当る部分であつて、前にのべたように、別段詳細に知る必要はないが、我々のソロバンと違つて01(でなくてもよいがとにかく2つの状態)について and (共に1のとき結果1), or (少くも一方が1のとき結果1), not(1のとき0, 0のとき1)という回路を用いて演算しており、このためには1854年発表以来100年近くうちすてられていたイギリスの数学者 Boole の創始になる、 $+$ $-$ $\times$ $\div$ の代りに $\wedge$ (and), $\vee$ (or), $\neg$ (not) を用いたいわゆるブール代数 Boolean algebra がその基礎となつてゐることをあげておこう。

2.5 記憶装置 memory

前にのべた0と1とで構成される2進情報を貯えるために、磁化の方向、磁化の有無などが利用される。現在多く用いられているものに磁気コア、磁気ドラム、磁気テープなど、将来のものに磁性薄膜、磁気ディスクなどがある。使用場所により内部記憶装置(機構が直接必要な番地からやりとりするメモリーで、一般に呼び出し時間 access time が極めて短かい)と外部記憶装置(一旦ひとまとめにして内部記憶装置に移してから使用されるもので、磁気テープ、磁気ドラム、磁気ディスクなどが用いられることが多い)とに分れる。

磁気コアは径1ミリ以下のフェライトコアをマトリクス状に張り、磁

束の方向を反転させるに必要な電圧の $1/2$ づつを $XY2$ 方向にかけて、必要な交点のコアの磁束の方向だけを変化させるもので、65K長語(65,536語)位まで使用される。呼び出し時間は短かくマイクロ秒(μs 100万分の1秒)単位である。

磁気ドラムは磁性膜をぬつた円筒を毎分3000~18000回転させ、これに20~50バンドに対応する読み書きヘッドをつけ、各バンドを円周に沿って100~200番地に分けて各番地に1語、全体で数千語、数万語の記憶を行なうもので、呼び出し時間は回転数などに関係するが機械的制約からミリ秒(ms 1000分の1秒)単位で、外部記憶装置として用いられることが多い。

磁気テープは録音テープを想像して貰えばよい。1本のテープに20万~100万語入れられるが、任意の情報の転送に要する時間が長く、秒単位である。

磁気ディスクはレコードを想像して貰えばよいわけで、アームは先づレコードを選び、次にトラックを選ぶので呼び出し時間は $1/10$ 秒の単位である。

磁気薄膜は将来の記憶素子として注目されている。まだ大容量のものは実用化されていないが、呼び出し時間は $1/10 \mu s$ 単位である。

2.6 出力装置 output device

1字づつ印字する頁式プリンタは毎秒10字程度のもので、計算速度に対して問題にならない。そこで初期のラインプリンタとして、タイプバーを並べて必要な字のところまでこれを持ち上げて一行に印字する型式のものが用いられ、1行100字のものを毎分100行印字する程度のものであつた。現在ではタイプチェーン式のラインプリンタで1行120字のものを毎分500~1000行打つ位のものが普通である。

このほか電気化学的 xerographic な印字方法で、電気的又は磁氣的に文字のパターンを作り、プラスチック粉を付着させて紙に転写させ

る方法によって、毎秒5000字の印刷が可能であるといわれている。

2.7 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 $10^{-2} \sim 10^{-3}$ 程度なので主として特殊目的に使用されている。

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

これ以降手動、電動の計算機が発明されたが、記憶、制御、入出力装置を持つ自動計算機の構想は1800年代のバベッジの Analytical Engine に始まった。遠大な構想を電氣的にでなく機械的に実現しようとしたため完成することはできなかったが、今日の電子計算機の機能のほとんどを備えようとしたものであった。

以下年表式に電子計算機の歴史について触れてみる。

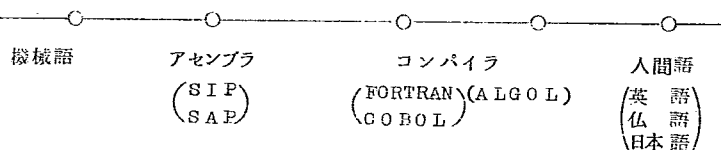
- 1880 Hollerith パンチカード会計機開発 今日のIBM
- 1907 Powers パンチカード会計機開発 今日のRR
- 1944 ハーバード大学の Aiken 等が ASCC (Automatic Sequence Control Calculator) 完成 機械式のもので
加算0.3秒 乗算6秒 除算11.4秒
- 1946 ペンシルバニア大学の Eckert および Mauchly が電子部品を用いた ENIAC (Electronic Numerical Integrator and Computer) 完成 加算0.2ms 乗算
2.8ms 除算0.6ms
- 1951 RRI社 Univac 1 事務用機
- 1953 IBM 701
- 1956 電気試験所 ETL-Mark III トランジスタ計算機

名 称	FACOM 230-30	FACOM 230-50	TOSBAC 3400	OKITAC 5090H	NEAC 2800	HITAC 5020	HITAC 5020E	IBM 7094II	IBM 360-50	CDC 3600	Burroughs B5500
製 造	富士 通信機	富士 通信機	東芝	沖	日電	日立	日立	IBM	IBM	コントロール データ	パロース
構成素子	トランジスタ ダイオード バロメロン 他	16600 45000 — —	8000 23000 — —	5000 20000 — —	25000 50000 — —	— — — —	10000 20000 — —	— — — —	— — — —	— — — —	— — — —
一語 bit 記憶装置 記憶容量 K 長語 (55K 約)	可変長 コア	36 コア	47 コア (薄膜)	42 コア	48 コア	32 コア	32 コア	36 コア	32 コア	48 コア	48 コア
演 算 時 間 μs	加 算 浮 点 乗 算 浮 点 除 算 浮 点	150 58 1400 850 2200 1600	154 4.4 308 252 396 352	118 1.6 1430 430 1760 570	30 30 230 260 290 360	42 24 82 198 384 444	190 8.0 370 240 760 420	28~168 1.0 28~5.6 2.0 28~8.4 6.2	6.9 2.8 21.5 28.8 23.0 33.2	4.2 2.1 6.4 21~6.7 13.0 21~14.9	2 19 20
ソフト ウェア	ALGOL	ALGOL		ALGOL		FORTRAN	FORTRAN	FORTRAN	FORTRAN	ALGOL	ALGOL
月レンタル料 価格 万円	380		210	350	800						900
価格 億円	1.7		1.0	1.4	3.6						3.6

3° ALGOL

3.1 概要

機械語と人間語の間には、機械に近いか人間に近いかによつてアセンブラ(SIP, SAP等), コンパイラ(ALGOL, FORTRAN, COBOL等)があるが、その占める位置は次のようなものと考えられる。



COBOLとは common business oriented language の略でアメリカ国防省が中心となつてまとめた事務用と称せられる言語であるが、やはり FORTRAN (formula translator) 同様に機械的な約束が多い。在庫量管理などの場合を除けば一般の官庁会社での事務系統の人達の仕事もやはり四則演算を中心として、あるデータから情報を抽出する形のもが多く、これらは何れも算法言語 ALGOL (algorithmic language) によつて十分にカバーできる範囲と考えられ、技術計算用としても勿論 ALGOL が一番人間の言葉に近く、自由で、文法的にもスツキリして居ると考えられるので、以下改訂 ALGOL-60 の文法を中心として、各機種による多少の制約を註記しながら説明したい。

註記の都合上、各社のアルゴルを次の略号で表記する。

- 'H1' OKITAC 5090H用 1964年開発のもの
- 'H2' OKITAC 5090H用 1965年開発のもの
- 'PAL' OKITAC 5090 G, D用 OKIPAL
- '230' FACOM 230-30, 230-50, 231用 アルゴル
- 'I' FACOM 202用 ISSP ALGOL

- ・J・ JUSE ALGOL
- ・B・ Burroughs B5000用 アルゴル
- ・A・ ALCOR グループ共通アルゴル(註1)

また記述を簡略化するために本文では次の略号を使用する。

- AE 算術式 (arithmetic expression) の略 数値を計算する規則を与える。
- AP 実パラメタ (actual parameter) の略 手続命令に用いる。
- BE ブール式 (Boolean expression) の略 論理値を計算する規則を与える。
- D 宣言 (declaration) の略
- DE 指示式 (designational expression) の略
- EP 形式的パラメタ (formal parameter) の略 手続宣言に用いる。
- FP FOR命令 (for statement) の略
- I 名前 (identifier) の略 変数, 手続, 名札, スイッチに用いる。
- IAE 整数値の結果となる算術式
- L 名札 (label) の略
- S 命令 (statement) の略
- SA 単純算術式 (simple arithmetic expression) の略
- SB 単純ブール式 (simple Boolean expression) の略
- SD 単純指示式 (simple designational expression) の略
- SV 単純変数 (simple variable) の略
- US 無条件命令 (unconditional statement) の略

註1 Baumann et al: Introduction to ALGOL
Prentice-Hall (1964)

尚本文ではALGOL-60の文法に従ったプログラムを示し、各機種による制約は脚註に示すが、ALGOL-60ではI/O関係が全く規定されていないので便宜上OKITAC 5090H用(1965年開発)アルゴル 'H2' によつておく。

3.2 ALGOLの基本原則

ALGOLによる簡単なプログラムを紹介する前にプログラムの原則をのべておく。

1. アルゴルプログラムは一つ以上の命令から成り立ちこれらはセミコロン(;)で区切られる。(但し else の直前には;を入れてはならない)
2. 命令は begin と end (式におけるカッコに当る)でくくつて1つの命令にまとめることができる。
3. アルゴルプログラム自身は begin と end でくくられる。
4. すべての変数, 手続, スイッチにつける名前はその名前が使われる範囲の一番先頭の begin に続く宣言部で宣言されてなければならない。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, その外側では意味を持たない。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。

勿論一段内側のブロックについては, 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ていない限り通用する(同じ名前が宣言されているときは内側のブロック内ではその内側のブロックで宣言されたものの方が優先する)。

5. 命令はGO TO 命令で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. すべての変数は必ず値を代入又は読み込みしてからでなければ式の中

で使用してはならない。

7. 一つのデータの読み込みが終れば、データテープ上で次のデータが読み位置に待機する。

8. 下つき添字，上つき乗べき，水平分割線による割算は禁止される。従つて次の形をとる。

A_8 $BA_{4,5}$ は $A[3]$ $BA[4,5]$

A^B A^{B+2} は $A \uparrow B$ $A \uparrow (B+2)$

$\frac{A+B}{C+D}$ は $(A+B)/(C+D)$

9. 取扱う数値は -3.6 ， 4×10^{-3} 等の実数 (real) と整数 (integer) に分れるが 4×10^{-3} は $4_{10}-3$ と書かれる。

10. 変数に対する値の代入は $\leftarrow$ (コロンイコール) という記号によるが新しい値が代入されると古い値は失なわれる。

尙一般的注意としてパンチヤの誤読を避けるために 1 と / と I と 7， $\overline{0}$ と 0 と D，5 と S，C と (，V と $\vee$ と U をはつきり区別して書くことが望ましい。

また入出力に関しては ALGOL-60 に規定がないので，便宜 'H 2' によるが簡単に次のものだけを採用する。

1. 読み込み

READ (A, B, C, ...);

テープより順に書かれた名前の場所に読み込まれる。

2. real タイプの印字

たとえば A については

PRINT ('R', A);

'R' と指定すると

$\underbrace{\pm . \times \times \cdots \times}_{9 \text{ ケタ}} \times 10^{\pm \times \times}_{2 \text{ ケタ}}$

の形で印字する。

3. integer タイプの印字

```
PRINT ( 'I' , A );
```

'I' の指定により

$\underbrace{\quad \pm x x \cdots x \quad}_{13 \text{ ケタ}}$

で不必要なゼロを取り除き符号を移動して印字

4. Boolean タイプの印字

```
PRINT ( 'L' , A );
```

もし true なら $\underbrace{\quad \text{TRUE} \quad}$

false なら $\underbrace{\quad \text{FALSE} \quad}$

と印字する。

5. これらは何個かをまとめてもよい。

たとえば integer I と real A をスペース 2 個をはさんで
印字し、それで 1 行おくりたければ

```
PRINT ( 'I, 2 BR /' , I , A );
```

とすればよい。

'/' はそれまでの情報を印字して 1 行おくことを命じたもので
2 行おくりたければ 2 / とすればよい。2 B はブランク 2 個の
意である。

また字そのものを打ちたければ ' ' の中で更に ' ' でつつんで書く。

1)

3.3 アルゴルによる簡単なプログラム

例 1 A, B, C に夫々 4.5, 3.8, 2.6 を代入し

$$X = A^B + (A + B)C$$

を計算し印字したいときは

となる。勿論xの右辺の式に数値を直接書き込んでも良いし、プリント命令の中に式を書き込んでもよい。従つて

でもよい。

$$X = A^B + (A + B)C$$

例3 データテープの先頭に組数Nとその後にA, B, CがN組入っている。これを読み込んで

$$X = A^B + \frac{C}{A^B}$$

```
begin real A, B, C ;
    integer N, I ;
    for I := 1 step 1 until N do
```

```

begin  READ(A, B, C);
      PRINT('N=' I 2 B, 'X=' R /',
            I, A↑B+C/(A*B));
end;
end;

```

例4 データテープの先頭に個数N, その後にX, からX_Nに至るN個の
データが入っているものを読み込み,

算術平均値 $MX = \sum X / N$

不偏分散の平方根

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

を計算し

MX = ... RV = ...

の形で印刷せよ。

```

begin  real  X, SX, SXX, MX, RV;
      integer N, I;
      READ(N);
      SX:=SXX:=0;
      for I:=1 step 1 until N do
      begin
        READ(X);
        SX:=SX+X;
        SXX:=SXX + X↑2;
      end;
      MX:=SX/N; (註1)
      RV:=sqrt((SXX-SX*MX)/(N-1));
      PRINT('MX=' R 2 B, 'RV=' R /', MX, RV);
end;

```

(註1) 一般にrealとintegerの混合演算を認めるが'J', 'H1',
'PAL'では認めない。このためNの代りにfloat(N)などにおき
換える必要がある。

これを計算機に代つて読み下してみると次のようなものになるう。

- これからアルゴリズムで命令する（一番始めの begin）
 - X, SX, SXX, MX, RV という名前は実数（real）として使用するからその記憶場所を留保せよ。
 - N, I という名前は整数（integer）として使用するからその記憶場所を留保せよ。
 - データテープから1組の情報を読み込み、その値を前にNという名前で教えておいた場所に格納せよ。
 - SX と SXX に 0 を代入せよ。
 - I に 1 を代入し次の begin と end でくくられた内容を実行し、終わったら I に 1 を加え、又実行することをくり返し、I が N になるまで行なえ。
- くり返し実行する内容は
- 「データテープより1組の情報を読み込んでXに格納 SX + X の値を計算し、その結果を新しく SX に代入、SXX + X² の値を計算し、その結果を新しく SXX に代入」
- ということがらである。
- SX（この段階ではXの総和が入っている）をNで割つてその結果をMXに代入せよ。
 - $(SXX - \frac{SX^2}{N}) / (N - 1)$ の平方根（sqrt）の値をRVに代入せよ。
 - MX= という字の次にMXの内容、2字ブランクをとつてRV= という字の次にRVの内容をプリントする準備をし、以上を印刷して1行おくれ（/）,
 - 以上でアルゴリズム命令を終る（最初の begin に対応する end）

例5 データテープに何個か分らないが $X_1, \dots, X_N, -10^{30}, Y_1, \dots, Y_N, -10^{30}$ が入っている。ただし -10^{30} はデータの終りであることを示す非常に小さい数であるとする。これを読み込んで
相関係数

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

を計算し $R = \dots$

の形で印刷せよ。

但し N は高々200までとする。

```
begin real SX, SY, SXX, SYI, SXY, R ;
integer N, I ;
array X, Y (1:200) ;
I := 1 ;
START1:
  READ(X (I)) ;
  if X (I) < -10 20 then go to NEXT1 ;
  I := I + 1 ;
  go to START1 ;
NEXT1:
  I := 1 ;
START2:
  READ(Y (I)) ;
  if Y (I) < -10 20 then go to END ;
  I := I + 1 ;
  go to START2 ;
END:
N := I - 1 ;
SX := SY := SXX := SYI := SXY := 0 ;
```

```

for I:=1 step 1 until N do
begin
  SX:=SY+X[I];
  SY:=SY+Y[I];
  SXX:=SXX+X[I]2;
  SYY:=SYY+Y[I]2;
  SXY:=SXY+X[I]*Y[I];
end;
R:=(SXY-SX*SY/N)/sqrt((SXX-SX2/N)*(SYY-SY2/N)); (註1)
PRINT('R=' R/, R);
end;

```

これを電子計算機に代つて読んでみると次のようなものになる。

- これからアルゴリズムで命令する(最初のbegin)
- SX, ..., Rは実数型の情報を格納する予定であるからその記憶場所を留保せよ。
- I, Nはinteger型で同様留保せよ。
- X, Yはそれぞれ1番から200番に至る配列(実数型)であるから、その記憶場所を留保せよ。(array)
- 以上で宣言部を終りまづIに1を代入せよ。
- 次に命令にSTART1という名札をつけよ。
- データテープより1組の情報を読み込みXという配列のI番目(オ1回はIに1が入っている)に格納せよ。
- 若しX[I]が-10²⁰より小さければNEXT1という名札のついた命令を実行せよ(さもなければ当然次の命令を実行する)
- I+1の結果を新たにIに代入せよ。
- START1という名札のついた命令より又順に実行せよ。

(註1) 'H2'ではsqrtはSQRTと大文字になる。

- NEXT 1 へ行く条件が成立したときは I に 1 を代入せよ。
 - 次の命令に START 2 という名札をつけよ。
 - データテープより 1 組の情報を読み Y [I] に格納。
 - 若し Y [I] が -10^{20} より小さければ END という名札のついた命令に行け。(さもなければ当然次の命令)
 - $I + 1$ を新たに I に代入。
 - START 2 という名札のついた命令に行け。
 - END に行く条件が成立したら N に $I - 1$ を代入。
 - SX.....等に 0 を代入せよ。
 - I に初期値 1 を代入し $+1$ きざみで I が N になるまで次の begin から end までを実行せよ。
 - くり返す内容は SX に X の和を SXX に X^2 の和を, SY に Y の和を SYY に Y^2 の和を SXY に XY の和を作つて入れることである。
 - $R = (\sum XY - (\sum X)(\sum Y)/N) / \sqrt{(\sum X^2 - (\sum X)^2/N)(\sum Y^2 - (\sum Y)^2/N)}$ を計算せよ。
 - R = の右に R の内容を入れ印刷し 1 行改行せよ。
 - 以上でアルゴリズムによる命令を終る(最初の begin に対応する end)
- 例 6 $\sqrt[3]{N}$ をニュートンの近似を使つて 10^{-6} の正確度で求め印刷せよ。

たゞしニュートンの近似は i 回目の近似値を X_i とすれば X_{i+1}
 $= X_i - \frac{f(X_i)}{f'(X_i)}$ であるから $f(X) = X^3 - N = 0$ に対し

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

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

```

begin comment CUBIC ROOT,
    X:=CUBIC ROOT(N).;
    real X, DX;
    integer N;
    READ(N);
    X:=N/3;
    START:
    DX:=N/X2-X;
    X:=X+DX/3;
    if abs(DX/3) ≤ 10-6
    then PRINT( R/, X )
    else go to START;
end;

```

comment の後はセミコロンまで何を書いても計算機は無視する。
 従つてここにプログラマが読者の為の注意書を記入することができる。
 初期値は何からスタートしてもよいが、ここでは仮りに $X/3$ から
 スタートした。名札はたまたま START を用いたが先頭が文字であ
 りさえすれば他の変数の名前と同様何を用いても良い（たゞし一般に
 6文字までしか判別せず、またタイプライタの上段コードに限るもの
 も多い）（註1）

abs () は絶対値をとることを意味し、else 以下は if の後
 の条件が成り立たないときに実行される。

また縦線は begin と end の間に対応関係を分りやすくするた
 めに便宜つけたもので、パンチヤーはこれを見無視する。

例7 相関係数の計算において例4にあげたN個のデータの読み込みを1
 つの手続きにまとめこれを用いて例5のプログラムを書き直してみる。
 たゞし今回は先頭にN次に $X_1, \dots, X_N, Y_1, \dots, Y_N$ のデータが入つ

（註1）これらの制限は後の名前のところで触れる。

ているものとする。

```
begin real SX, SY, SXX, SYI, SXY, R;  
integer M, I;  
array X, Y[1:200];  
procedure READA1(N, A);  
value N;  
integer N; array A;  
begin integer I;  
for I:=1 step 1 until N do  
    READ(A[I]);  
end;  
SX:=SXX:=SYI:=SXY:=0;  
READ(M);  
READA1(M, X);  
READA1(M, Y);  
for I:=1 step 1 until M do  
begin real P, Q;  
P:=X[I];  
Q:=Y[I];  
SX:=SX+P;  
SY:=SY+Q;  
SXX:=SX+P2;  
SYI:=SY+Q2;  
SXY:=SXY+P*Q;  
end;  
R:=(SXY-SX*SY/N)/sqrt((SXX-SX2/N).  
    (SYI-SY2/N));  
end; PRINT("R=" R/, R);
```

此処で新しく現れたのは procedure とブロックである。

- procedure の部分は後で詳細に説明するが、ここで電子計算機に代つて読んでみると次のようになる。

宣言として procedure という宣言子が来ればその後の名前 READ A D A 1 とはその後にある内容を代表するものという扱いになりカッコ中のパラメタを媒介としてこの procedure の内と外で情報のやりとりをする予定と解釈する。

value N ; で N のところには実際この手続が使用された時には N として使いたいものの値が書きこまれそれは integer 型であり、A は配列の名前であることを了解する。

procedure の本体は A [1] から順次 A [N] まで N 回、データテープよりの読み込みを行つて A という配列に格納を行なわせるものである。

実際に使っているのは READ (M , X) ; と READ (M , Y) ; X [1] から X [M] までと、Y [1] から Y [M] までの読み込みをこの手続命令によつて完了している。

- プログラムは一般に $\bar{S}$ (statement 命令) を命令用のカッコ begin と end でくくつたものでその中で使う名前を定義するための宣言 $\bar{D}$ (declaration) が先頭の begin につづいて置かれている。本プログラムも $SX := \dots := 0 ;$ という代入命令からは $\bar{S}$ であるがそれまでは real $\dots ;$ integer $\dots ;$ array $\dots ;$ 及び procedure $\dots$ (後の 6 行を含み) $\dots end ;$ はすべて宣言である。従つて模式的に書けば

begin $\bar{D}_1 ; \bar{D}_2 ; \bar{D}_3 ; \bar{D}_4 ; \bar{S}_1 ; \bar{S}_2 ; \bar{S}_3 ; \bar{S}_4 ; \bar{S}_5 ; \bar{S}_6 ; \bar{S}_7 ; end ;$

となつておりこのような形をしたものをブロックという。

このうち S_5 は do の後の 7 つの $\bar{S}$ を更に begin と end でくくつてありしかもその先頭には real P, Q ; という $\bar{D}$ がついている。したがつて S_5 そのものもブロックである。

若し $\bar{D}$ を持たず単に数個の $\bar{S}$ を begin と end でくくつたものは複合命令 (compound statement) と呼ばれ、ブロックも複合命令も外から見れば1つの $\bar{S}$ 扱いとなる。ブロック構造はALGOLの一つの大きな特長であつて、すべての宣言はそれを含む最小のブロックの中でのみ有効であり、もしその外のブロックで宣言されたものと同じ名前のものが中のブロックで宣言されたときは、中の方が優先するが、ブロックの外に出たときは中で宣言されたものはその意味を失う。すなわち、次の例のAはブロック2の中と外では異なつた意味に使用され、またこのゆゑに何人かのプログラマが、受渡しに使う名前以外はまったく任意につけたものでも、そのまま継ぎ合せて一つのプログラムとすることができるわけである。

```

begin real A, B ;
   $\bar{S}$ ;
  L1:  $\bar{S}$ ;
  begin real A ;
    L1:  $\bar{S}$ ;
     $\bar{S}$ ; go to L1;
  end ;
   $\bar{S}$ ; go to L1;
   $\bar{S}$ ;
end ;

```

} ブロック 2

名札 (label) についても同様で、その名札を立てられたブロック内のみで有効であり、前の例でいえば3行目の L^1 は、もし内側のブロックの中で同じ名前の名札が立てられたときは、内側ブロック内ではその方が優先する。したがつて8行目の go to L1によつて5行目の L^1 という名札に行くことは不可能である。

以上を要するに宣言および名札はその宣言が行なわれたブロックに対して局所的 (local) であり、その外にある宣言および名札は

そのブロックに対し、非局所的 (non-local, global) であるという
ことになり、これが ALGOL の一つの特徴になっている。

ALGOL 全般に対する解説書としては (註 1) のようなものがある。

(註 1)

森口繁一編：ALGOL 入門 JUSE 出版

淵一博 伊藤文子：算法言語 ALGOL 60 に関する報告 情報処理 1
巻 3, 4 号 (1960), 2 巻 1 号 (1961)

中島勝也：ALGOL 60 の改訂について 情報処理 4 巻 3 号 (1963)

Ma Cracken, D. D. : A Guide to ALGOL programming, Wiley
(1962)

Dijkstra, E. W. : A Prime of ALGOL 60 Programming,
Academic Press (1962)

Nauer, P. (editor): Revised Report on the Algorithmic
Language ALGOL 60 Comm. ACM Vol. 6
No. 1 (1963)

中村慶一：技術者のための統計解析 11 章 アルゴリズムによるプログラミング
山海堂 (1965)

Wooldridge & Ractliffe : An Introduction to ALGOL
Programming, The English Univ. Press
(1963)

Baumann, Feliciano, Bauer & Samuelson : Introduction to
ALGOL, Prentice-Hall (1964)

岡崎・角山：わかりやすいプログラミング 産報 (1965)

4° 基本定義

4.1 記号

記号としては文字 (letter), 数字 (digit), 論理値 (logical value), 限定詞 (delimiter) がある。

尚アルゴル 60 ではアンダライン又は太文字を用いるが、一般のタイプライタでは実現不能なので普通の書体にしたり 'I' のように ↓ begin と前に ↓ をつけたりする。また 'B' ではすべて大文字を用い 'A' では 'begin' とする。

文字: A から Z および a から z までの 52 文字⁵

数字: 0 から 9 までの数字

論理値: true false^{5'}

限定詞:

算術演算子 (arithmetic operator): + (plus) − (minus)

× (multiply)⁶ / (slant) ÷ (divide)^{6'}

↑ (up arrow)⁷

論理演算子 (logical operator): ≡ (equivalent)⁸

⊃ (imply)⁹ ∨ (or)¹⁰ ∧ (and)¹¹ ¬ (not)¹²

関係演算子 (relational operator): < (less than)

≤ (not more than)¹³ = (equal) ≥ (not less than)¹⁴

> (more than) ≠ (not equal)

逐次演算子 (sequential operator): go to¹⁵ if then

else for do

区切り子 (separator): , (comma) . (decimal point)

: (colon) ; (semi colon) := (colon equal)¹⁶

␣ (space)¹⁷ 10 (small ten)¹⁸ step until while¹⁹

comment

括弧 (bracket): begin end () [] ' ' ^{19'}

宣言子 (declarator): real integer array procedure switch

Boolean*19" Own*20

指定子 (specifier): value label*21 string*22

(明細指定子 specifier としては, label string のほかに
real integer Boolean procedure switch array)*23

標準関数 (standard function): sin cos arctan exp sqrt
sign*24 ln abs

換算関数 (transfer function); entier*25

-
- *5 'PAL', 'H1', 'J', 'B', 'H2' では大文字の26文字に限る。
 - *5' 'H2' では TRUE, FALSE
 - *6 大文字のXと間違えるので, 'PAL' 'H1', 'J', '231', 'T', 'H2' では
* (asterisk) を用いる。
 - *6' 'B' では DIV を用いる。'E', 'J' では div。
 - *7 'B' では (asterisk) を用いる。'B' ではこの他に MOD がある。
割算の余りを求める演算である。
 - *8 'H1', 'J' にはない。'230' では eqv を用いる。'B' では EQV を用いる。
'I' では ← 'E' では equiv。
 - *9 'I' では → を用いる。'J' 'H1', 'E', にはない。'231' では imp を
用いる。'B' では IMP。
 - *10 'E' 'J' 'I' '230' では or。'B' では OR。
 - *11 'E' '230' では and。'B' では AND。'J', 'I' では &。
 - *12 'J' にはない。'E' '231' では not。'B' では NOT。'I' では ~。
 - *13 'J' '230' では <= を用いる。
 - *14 'J' '230' では >= を用いる。
 - *15 '230' では go と to の間は離さなければならない。'B' では GO
TO を用いる。
 - *16 'B' では ← を用いる。
 - *17 'J', 'PAL', 'H1', 'B', 'H2' では * を用いる。

- 18 'B' では 'a' を用いる。
- 19 'J' にはない。
- 19' 'E' では Σ 及び $\int$ を用いる。
- 19'' 'J' にはない。
- 20 '230', 'I', 'H 1', 'J', 'H 2' にはない。
- 21 'J' にはない。
- 22 'B', 'J', 'PAL' にはない。
- 23 型宣言の array は名前の後に添字の限界を伴うが、明細指定の array は名前だけである。
- 24 シグナムと読む。
- 25 アンティエと読む。'PAL' では entire と綴る。'J', 'H 1', 'H 2' では FIX を使い $\text{entier}(A + 0.5)$ となる。

4.2 名 前 (identifier 本文中ではIと略)

変数 (variable), 名札 (label), 手続 (procedure) などに
対してつけられるもので, 文字・²⁶ で始まり, 場合によつてはその後に
文字または数字が続いたものである・²⁷。

名札 (label) としては, 上の名前その他に符号なしの整数を使用しても
よいが・²⁸, いずれにせよ, 名札を宣言するときはその直後に: をつけな
ければならない。

(正) A ABC MI A12B3CD

(たゞし最後のものは・27によりA12B3CEなどとは区別され
ぬものがある)

(誤) 12AB

(たゞし, 名札としてなら12:は許される。)

4.3 数 (number)

数値を読み込ませたり, 式の中に直接書き込んだりすることができるが,
-3.5, 4×10^{-5} などの実数 (real) と 8 などの整数 (integer) があ
る ALGOL における数の表現には

1. 小数点で終つてはならない
2. 10 のべきは整数に限る
3. 10 の直前に小数点があつてはならない

という制約がある。

・26 'H2', 'PAL', 'H1', 'J', 'B' では大文字 (以下も大文字と
読みかえる。)

・27 'H2', 'PAL', 'H1', 'J', 'B', '230', 'I' すべて最初
の6文字しか判別しない。

・28 '230', 'I', 'J', 'H1', 'H2' では名前に限る。

実際のもの	(正)	(誤)
1 2 3	1 2 3	1 2 3.
0.123	0.123 or .123	0, 123
-12 4.3 2 6	-12 4.3 2 6	-12 4, 3 2 6
10 ¹²	10 ¹²	10 ¹²
3.2 1 × 10 ⁻¹⁰	3.2 1 ₁₀ -10 ^{*29}	
10 ^{2.5}	表現できない	10 ^{2.5}
	(10↑2.5は式である)	
35 × 10 ⁻²	35 ₁₀ -2	35 ₁₀ -2 or 35 ₁₀ (-2)

4.4 変数(variable)

単数 { 単純変数 (simple variable $\widetilde{SV}$ と略) : $\widetilde{I}$
添字付変数 (subscripted variable) :

$\widetilde{I} [I \widetilde{A} \widetilde{E}, I \widetilde{A} \widetilde{E} \cdots, I \widetilde{A} \widetilde{E}]$

(正) A ABI

A[K] ABI[K, 3] C[K+L, J]

B[A[K+1], L+2] D[if P>3 then 8 else 9]

5° 宣言 (declaration D と略)

宣言には型宣言, 配列宣言, 手続宣言 (関数指示宣言を含む), スイッチ宣言, 名札宣言がある。名札宣言以外はすべて $\widetilde{S}$ に先立つ宣言部で宣言し, すべての宣言はその宣言の行なわれたブロックに関し局所的である。(すなわち, そのブロックの外では意味を失う) 手続, スイッチ, 名札宣言は別にふれる。

• 29 'B' では 3.2 1 a - 10 と書かれる。

5.1 型宣言 (type declaration)

すべての単変数は型宣言を行なっていないければ使用してはならない。
同じ型の宣言はコンマでつないでよいが、その型の宣言が終わつたらかならずセミコロンを打つ。

型には実数 (real), 整数 (integer), ブール (Boolean) の三つがある。

real は少なくとも小数点か 10 のべきかを含んでいる。

integer はこのいずれも含まぬものである。

Boolean には true と false の二つの値しかない。

(正) real A, B; integer C, K; Boolean P;

(名前として許されるものなら何を用いてもよいが同じブロック内の変数, 名札, 手続につけてはならない。)

5.2 配列宣言 (array declaration)

すべての添字付変数は配列宣言を行なっていないければ使用してはならない。コンマ, セミコロンの使い方は型宣言の場合と同じである。また, 同じ添字の限界をもつ配列はコンマでつないでよい。

array I, J, ..., I(AE:AE, ..., AE:AE),

I, J, ..., I(AE:AE, ..., AE:AE);

integer array "

Boolean array "

単に array と宣言されたときは real array を意味する。

AE:AE³⁰ は添字の下限: 上限を示し, 下限 上限でなければならない。

(正) array A, B(-2:2);

integer array C[1:N, 1:N];

array C, D[1:5], E[2:5];

(誤) array A[3:1];

* 30 'H1', 'H2', 'J', 'PAL' では整数: 整数でなければならない。

'230', 'A' では変数を用いて dynamic に割付け得る。

6° 式 (expression)

6.1 算術演算子 (arithmetic operator)

算術演算子の優先順位は

- | | |
|---------|---|
| 1. 丸カッコ | () |
| 2. ベキ | $\uparrow$ |
| 3. 乗除 | $*$ $/$ $\div$ (今後は大部分の例にしたがい $\times$ の代りに $*$ を用いる) |
| 4. 加減 | $+$ $-$ |

の順であり、同順位のものを並記したときは式の左側の方の演算子が優先する。

$\uparrow$ については $A < 0$ なら $A \uparrow B$ は B が integer のときのみ定義される。また $A = 0$ のとき $B \leq 0$ ならば定義されない^{*31}。 $2 \uparrow N \uparrow K$ は $(2^N)^K$ であり $2 \uparrow (N \uparrow K)$ が 2^{N^K} となる。

$\div$ は整数同志の割算に用い、その結果は絶対値の切捨を行なったものに符号を付したものとなる。たとえば、

$$-5 \div 2 \rightarrow -2$$

で、これを利用して A が偶数か奇数かをチェックするのに、

$$A \div 2 * 2 = A$$

かどうかを調べ、これが成立すれば偶数とすればよい。

一般に $*$ $+$ $-$ の演算子はその両側の式がともに整数のときのみ結果は整数となるが、それ以外のときはすべて結果が実数となる^{*32}。($/$ は常に real の結果を与える、 $\div$ の両辺は整数に限り結果も整数)

(正)	(誤)
$A * B / C + D / A * B \uparrow 2$	$A \frac{B}{C} + \frac{D}{A} B \uparrow 2$
$\frac{A}{B/C}$ に対し $A / (B * C)$	$A / B * C$
または $A / B / C$	
$z + (-E)$	$z + -E$
$A / (-B)$	$A / -B$
$2 \uparrow (-2)$	$2 \uparrow -2$

6.2 関係演算子 (relational operator)

$> \geq = \leq <$ キ という関係演算子はその両側に $\tilde{A} \tilde{E}$ がくることによつて関係を構成し、ブール式の構成要素の一つとなる。また関係演算子は $\tilde{A} \tilde{E}_1 \geq \tilde{A} \tilde{E}_2 \geq \tilde{A} \tilde{E}_3$ という風に、 $\tilde{A} \tilde{E}$ を挟んで2つ以上連続して用いてはならない。

関係演算子は算術演算子に優先されるが、論理演算子には優先する。この関係を表に示すと次のようになる。

優 先 順 位

1.	()	6.	$\neg$
2.	$\uparrow$	7.	$\wedge$
3.	$\cdot \ / \ \div$	8.	$\vee$
4.	$+$ $-$	9.	$\supset$
5.	$> \geq = < \leq =$	10.	$\equiv$

$$(正) \quad X = -2 \quad A + B \leq C * D + E$$

(後者はまず $(A + B)$ と $(C * D + E)$ が計算され後にこの両者が比較される)

$$(誤) \quad A \geq B + C \geq D$$

• 31 'J' では ' $A \downarrow B$ ' は 'B' が非負の整数のときのみ定義。'PAL' では 'A' が integer, B が real のときは定義されない。

• 32 'H1', 'J', 'PAL' では両側のタイプが同じでなければ定義されない。

6.3 論理演算子(logical operator)

論理演算子には $\neg$ (not), $\wedge$ (and), $\vee$ (or), $\supset$ (imply), $\equiv$ (equivalent) があり, この順序に優先する。 $\neg$ 以外の論理演算子はその両側にブール定数, ブール変数, 関係式または $\bar{B}E$ を並べることによつて ($\neg$ のみはその後にくる $\bar{B}E$ を否定する働きをもつ) さらに $\bar{B}E$ を構成する。

A, B を Boolean 型と宣言された変数とすれば,

$\neg A$: A が true のとき false, 逆に false のとき true

$A \wedge B$: A, B ともに true のときのみ true

$A \vee B$: A, B ともに false のときのみ false

$A \supset B$: A が true でかつ B が false のときのみ false

$A \equiv B$: A, B ともに true, または false のとき true

$(A \wedge B \vee \neg A \wedge \neg B)$

で, $\wedge$ は And と関連して記憶するとよい。 $A \supset B$ は if A then B とか, $\neg A \vee B$ と考えるとわかりやすい。

$\neg$ 以外の論理演算子を 2 つ以上続けて使用できない。

(正) $A+B > -5 \wedge Z \rightarrow D > Q \uparrow 2 \wedge \neg (C=0)$

$P \wedge Q \vee X \neq Y \vee \neg (A < 100)$

(誤) $A \vee \wedge B$

6.4 算術式(arithmetic expression $\bar{A}E$ と略)

算術式は数, 丸カッコ, if, then, else, 変数, 関数指示子, 算術演算子などよりなり, 数値を計算する規則を与えるもので次の 2 つに分かれる。

単純算術式 (simple $\bar{A}E$, $\bar{S}\bar{A}$ と略)

if then else を含まぬ $\bar{A}E$

条件付算術式 (conditional $\bar{A}E$)

if $\bar{B}E$ then $\bar{S}\bar{A}$ else $\bar{A}E$

条件付算術式は if then else の三つを含んでおり, then と else

の間に $\widetilde{S}A$ に限る。もし、ここに条件付算術式を入れたければ、その全体を丸カッコでつつんで $\widetilde{S}A$ になおせばよい。また、条件付算術式を式の中で用いるときは丸カッコで包むとわかりやすい。丸カッコは対応してさえいれば余分にあつてもさしつかえない。

(正) $Q123 \text{ SUM } A(J+2,3) \text{ COS}(X+Y*3.0)$

$A(J+2,3)+\text{SUM} \text{ COS}(X+Y*3.0)/Q123$

if $Q>0.5$ then $S+3.0*Q/A$ else

$2.0*S$

if $A \geq B$ then (if $P \geq 0$ then $P*(Q+1)$

else $P*(Q-1)$) else $P*Q$

$A+(if X \leq Y \text{ then } C \text{ else } D)$

$A[4]+B(if X>Y \text{ then } 3 \text{ else } 4)$

(誤) if $A \geq B$ then if $A \geq 0$ then $P*(Q+1)$ else $P*(Q-1)$ else $P*Q$

6.5 関係 (relation)

2つの単純の算術式を関係演算子で結んだもので、ブール式の構成要素の一つである。関係演算子は算術式をはさんで、2つ以上連続して用いてはならない。

(正) $\widetilde{S}A_1 \geq \widetilde{S}A_2$

(誤) $\widetilde{S}A_1 = \widetilde{S}A_2 < \widetilde{S}A_3$

6.6 ブール式 (Boolean expression $\widetilde{B}E$ と略)

ブール式はブール定数 (true および false), ブール型変数および関係式を1以外の論理演算子で結んで論理値 (true または false) を計算する規則を与えるもので、次の2つに分かれる。

{ 単純ブール式 (simple $\widetilde{B}E$, $\widetilde{S}B$ と略)
if then else を含める $\widetilde{B}E$

条件付ブール式 (conditional B̄E)

if B̄E then S̄B else B̄E

条件付ブール式は if then else の三つを必ず含んでおり, then と else の間は S̄B に限る。もし, ここに条件付ブール式を入れたければ ĀE 同様丸カッコで包んで S̄B に直せばよい。また条件付ブール式を式の中で用いるときは丸カッコで包むとわかりやすい。丸カッコは対応してさえいれば, 余分にあつてもさしつかえない。

(正) A, B, E を Boolean type の変数, C, D を ĀE とすれば

B C ≠ D → B → A ∨ B A ∨ B ∧ E C > 0 したがって

B := C > 0 ; という形で代入を行なうと → B は C ≤ 0 の結果が代入される。

if A then D = C else → A ∨ B

(誤) A ∧ ∨ B

ブール型の変数に値 (これは true と false の2つしかない) が代入されるのは B := B̄E すなわち, B := C = 0 ∧ D ≠ 0 ; という代入命令により右辺の関係が真なら true が, 偽なら false が代入される。

B := 4 < 5 ; (true が代入される)

B := 8 ≥ 10 ; (false が代入される)

B := false ; (false が代入される)

6.7 指示式 (designational expression D̄E と略)

指示式とは名札 (label) を得る規則であり, 名札, スイッチ指示子, カッコ及び if, then, else がその構成要素で, 次の2つに分かれる。

go to D̄E ; switch A := D̄E₁, D̄E₂, … ;

指示式はレベルの名前の他にスイッチ指示子としては名前の後に ĀE を

[] でくくつたものをつける。

{ 単純指示式 (simple $\overline{DE}$, $\overline{SD}$ と略)
 if then else を含まぬもの
 { 条件付指示式 (conditional $\overline{DE}$)
 if $\overline{BE}$ then $\overline{SD}$ else $\overline{DE}$

後者については $\overline{AE}$, $\overline{BE}$ の場合とまったく同様である。

(正) $L1 \quad 45^{*33} \quad A \quad PQ \quad SW(I)^{*33}$

$if \quad C=0 \quad then \quad L1 \quad else \quad PQ$

(誤) $if \quad C=0 \quad then \quad if \quad D=0 \quad then \quad L1 \quad else \quad A \quad else \quad PQ$

7. 命 令 (statement $\bar{S}$ と略)

命令は計算機に1つの操作を実行させる単位であつて通常その書かれた順序に実行される。1つの命令の末尾には必ず;をつけなければならない(ただし, else の直前ではつけてはならない;は $\bar{S}$ の区切りに用いるがendの前に空命令があるものと考えれば;があつても差し支えない。) * 33

命令には基本命令(代入命令, GOTO命令, 空命令, 手続命令) FOR命令, 条件付命令およびI/ $\bar{O}$ 命令がある。ただしI/ $\bar{O}$ 命令はALGOL 60では規定されない。手続命令およびI/ $\bar{O}$ 命令は別の独立した章で触れる。

7.1 代入命令 (assignment $\bar{S}$)

$\bar{V}_1 := \bar{V}_2 := \dots := \bar{V}_n := \bar{A} \bar{E};$

$\bar{V}_1 := \bar{V}_2 := \dots := \bar{V}_n := \bar{B} \bar{E};$

の形をとり:=の右端の式の結果を左端の変数に代入させる命令で, 左辺は同じ型であればいくつつながつていてもよい。また右辺と左辺とのタイプが

* 33. 'H1', 'H2', 'E', '23G', 'J', 'I' ではレベルは名前に限るので誤り。

* 33. $\bar{I}$ がスイッチ宣言で示された個数の範囲をとるときはそのgo to 命令は無効になる。また整数でないときは最も近い整数をとる。'J' では整数値をとる $\bar{A} \bar{E}$ に限る。

* 33. 'E' では最後のendの後に;が必要。

異なつていても左辺同志のタイプが同じなら自動的に変換を行なう(右辺が実数、左辺が整数のときは四捨五入を行なう)。また左辺は関数指示子の名前であつてもよい。新しい値が代入されたら古い値は失われる。

なか、プログラミングするときにわかっている数値はもちろんそのまま ΔE などに書き込んでよいが、計算時に外から知らせてやる必要のある数値は I/O 命令によつて読み込めばよい。

(正) $SS := SS + A[I];$

(代入命令は再帰的で左右両辺に同じ名前が現われてももちろんかまわない)

$K[Z, M] := L[3, M] = C := 3.5;$

(誤) $A + B := C$

7.2 FOR命令 (for $\bar{S}$, $\bar{FS}$ と略)

命令 S をくり返し実行させたい場合に for $\bar{V} := \langle \text{for list} \rangle \text{do } \bar{S};$ を用いる。

for list としては

$$\begin{cases} \bar{A}\bar{E} \\ \bar{A}\bar{E}_1 \text{ step } \bar{A}\bar{E}_2 *^{3,4} \text{ until } \bar{A}\bar{E}_3 \\ \bar{A}\bar{E} \text{ while } \bar{B}\bar{E} \end{cases}$$

の3種があり *³⁵, これらを、でつないでもよい。

do の後は1つの $\bar{S}$ しか実行しない。したがつて、2つ以上の $\bar{S}$ をくり返し実行させたいときは、これらを begin end でくくつて1つの複合命令 (compound statement) にしてやらなければならない。do の後はいつ

*³⁴ 'PAL' では $\bar{S}\bar{V}$ または数値に限る。

*³⁵ 'B' ではこの他に、 $\bar{A}\bar{E}_1 \text{ STEP } \bar{A}\bar{E}_2 \text{ WHILE } \bar{B}\bar{E}$ がある。

もbegin endでくくる習慣をつけるとエラーが少なくなる。begin endは対応してさえいれば、余分にあつてもなんら害はない。

```
(正)  for A:=10 step 2.0 until 16.0 do
        S:=S+A/(A+B);
    for A:=4.5, 3.0*B while B≤20.0 do
        K:=B↑2*C;
    for I:=1 step 1 until 1012 step 2 until
        2025 step 5 until 100 do
        begin II:=I↑2;
          PRINT('R//', II);
        end;
```

7.3 GO TO命令 (go to $\bar{S}$)

ある $\bar{S}$ に名前をつけたいとき、その直前でレーベルの宣言を行なえばよい。

$\bar{L}_1; \bar{L}_2; \dots; \bar{L}_n; \bar{S};$

名札 (label) はいくつ重ねてもよい。また名札はその宣言がされたブロックに対し局所的である*36。

いくつかの名札に番号をつけて条件によつてはその何番目かを指定したいときはswitch宣言を行なえばよい。

switch I:= $\bar{D}\bar{E}_1, \bar{D}\bar{E}_2, \dots, \bar{D}\bar{E}_n;$

この場合、番号は最初から1, 2と数えられ一種の配列と考えられることになり、そのJ番目を指定するときは

*36 'H1', 'H2', 'PAL' ではブロックだけでなく複合命令に対しても局所的である。'E' ではレーベルはすべて宣言部でスイッチ宣言の右辺リストに記入しなければならない。

I(J)

となる。Jには式を書いて良いが結果がrealのときは自動的に変換され、範囲外の場合はdummy扱いとなる。*36

以上のような名札およびスイッチ宣言をした後にSの実行順序を変更したい場合には次のgo to命令を用いる。

go to DE

DEはレーベル、スイッチはもちろん条件付指示式であつてもよい。

switch P:=LA, LP, Q4;

としてあれば、go to P[J];の命令によつてJが1のときはLAに、2のときはLPに、3のときはQ4に飛越しを行なう。これらの名札はもちろんこのスイッチ宣言の行なわれたブロックまたはその外側のブロックで宣言されていなければならない。

(正) go to if A=0 then L1 else if B≠0
then L2 else if C=0 then (if D≠0 then L3 else L4)
else L5;

7.4 条件付命令 (conditional S)

条件付命令には次の3種がある*37

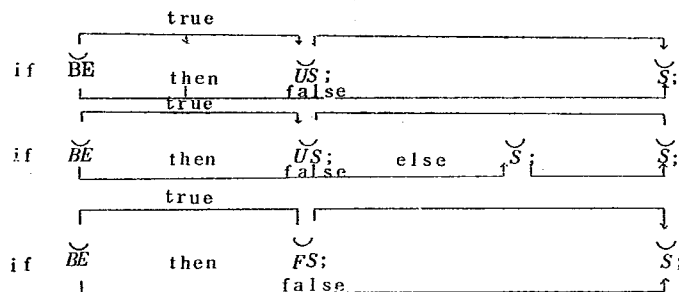
{ if BE then US;
if BE then US else S;
if BE then FS;

一ばん最初のは特にIF命令 (if S) とよばれる。USは代入命令、go to命令、空命令、手続き命令、L/Q命令並びに複合命令及びブロックである。

*36 'H1', 'H2', 'PAL' ではブロックだけでなく複合命令に対しても局所的である。

*37 実際は if BE then FS else S; も考えられるが、もし FS の後の S には if を含むものがあると、次の else はどちらの if に対応するが不明になるので禁止している。たとえば 'PAL', '230', 'H1', 'H2' では else は1番近くの if に対応するものと考えて処理している。

条件付命令による処理の進行は下図のようなものになる。

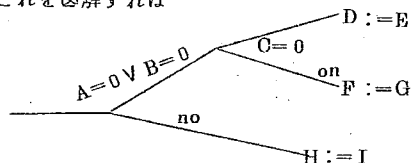


```

(正)  if  $A \geq 0.0 \wedge B \geq 0.0$  then  $X := \text{sqrt}(A) + \text{sqrt}(B)$  ;
      if  $A \geq 0.0 \wedge B = 0.0$  then  $X := \text{sqrt}(A) + \text{sqrt}(B)$  else go
      to END ;
      if  $A = 0 \vee B = 0$  then begin if  $C = 0$  then  $D := E$  else
       $F := G$  ; end else  $H := I$  ;

```

これを図解すれば



(誤) if $A=0$ $B=0$ then if $C=0$ then $D:=E$ else
 $F:=G$; この場合は else $F:=G$ はどの if に対応するが不明である。

7.5 空命令 (dummy $\widetilde{S}$)

何も実行しないという命令があり、これを空命令という。ある条件が成り立てば終りにしたいときに空命令に適當な名札を立てて、その次にプログラムの終りの end をもつてくるという使い方も空命令の利用方法である。

```
if  $A=0$  then go to END;
```

```
   $\widetilde{S}_1$ ;
```

```
   $\widetilde{S}_2$ ;
```

```
END;
```

```
end;
```

この場合、 $A=0$ が真であれば、END に行き、その空命令を実行した後、つぎに行き end で終わる。偽ならばもちろん $\widetilde{S}_1$ を実行する、この場合、
then go to END else $\widetilde{S}_1$; $\widetilde{S}_2$; となつていても同じである。

8. ALGOLによるプログラム例 (1)

以上でprocedure に関する部分以外はほとんど説明が終つたので数値解析の基本的な部分を含めて例題を取扱つてみる。

8.1 作 表

$n, n^2, n^3, 1/n, N/$ 表を 1(1)10000 について作れ、初期値 1 キザミ 1 終期値 10000 でこれを一行にならべてプリントする。念のために表頭には $N, N \uparrow 2$ などの字を印字し 50 行打つと 10 行とばして LP (ラインプリンタ) で打たせることとしたい。factorial は FACTN とでもしておいて FACT という変数に 1 を入れておき、次々と N を掛ける方針で進む整数では 13 ケタまで計算するがそれ以上は real にしなければならない。

```
begin integer N;
  real FACT;
  PRINT('↑2/');
  PRINT('6B 'N' 14B 'N↑2' 13B 'N↑3' 13B
    '1/N' 12B 'FACTN' //');
  FACT:=1;
  for N:=1 step 1 until N do
    if N=50*50=N then PRINT('10/');
    begin
      PRINT('3I, 2R/', N, N↑2, N↑3,
        1/N, FACT);
      FACT:=FACT*N;
    end;
end;
```

最初の '↑2/' はまづ用紙が 60 行用紙なので上下 5 行をあけて真中の 50 行に印字する方針で 2 行おくつて次に表頭を印刷し 1 行あけて (//)

50行表を印刷する。

} 2行					
	N	$N \uparrow 2$	$N \uparrow 3$	$1/N$	FACTN
	1	1	1	$10000 \div 10$	$100 \div 10$

} 5行					

50行ごとと10行あけるために

if $N \div 50 * 50 = N$ then PRINT('10/');

としてあるが N が149であれば $N \div 50 = 2$

となり(÷の特性でその計算結果2.98を越さぬ最大の整数2となる)

これに50をかけた100は149に等しくないからこの条件は成立しないので次の命令に移る。

もし更に10, 20, ...のところで1行あけたければif $N \div 50 * 50 = N$ then PRINT('6/')

else if $N \div 10 * 10 = N$ then PRINT('2/');

とすればよい。

表頭の子と下の情報を揃えるには 'I' では16文字 'R' では17文字のスペースを使用することを考慮しておく必要がある。

また1行120字なり135字なりの制限があるのでこれに従って適当な処置をしておく必要がある。一般に120字を越したら自動的にその直前の状態で1行印字し、次の行の先頭に新しい情報を入れるものが多い。

8.2 分 類

コード番号順に1番から給与表が並んでいるものを順数 N を先頭に読み込み、給与額の順にコードを並べ直してコードと給与の表を作れ、但し人数は

高々1000名とする。

```
begin integer N,I,J;
integer array NAME(1:1000);
array HOKYU(1:1000);
READ(N);
for I:=1 step 1 until N do
  NAME(I):=I;
for I:=1 step 1 until N do
  READ(HOKYU(I));
for I:=1 step 1 until N do
  begin
    for J:=I+1 step 1 until N do
      if HOKYU(I)<HOKYU(J) then
        begin
          real BIG; integer K;
          BIG:=HOKYU(J);
          HOKYU(J):=HOKYU(I);
          HOKYU(I):=BIG;
          K:=NAME(J);
          NAME(J):=NAME(I);
          NAME(I):=K;
        end;
      PRINT('I, R/', NAME(I), HOKYU(I)),
    end;
  end;
end;
```

でブロックの中でBIGとKを作業用番地として最大のものとおき換えを行っている。

8.3 マトリクスの取り扱い

M行N列のマトリクスXがMNを先頭に行なりに

$$\begin{array}{ccccccc} X_{11} & X_{12} & X_{13} & \cdots & X_{1m} \\ X_{21} & X_{22} & X_{23} & \cdots & X_{2m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ X_{n1} & X_{n2} & X_{n3} & \cdots & X_{nm} \end{array}$$

M N X_{11} X_{12} $\cdots$ X_{1m} X_{21} $\cdots$ X_{nm} の順に入り、その後にはやりM行N列のYについて同様に入っている。

これを読み込んでマトリクス和

$$Z = X + Y \quad Z(I, J) = X(I, J) + Y(I, J)$$

を計算し、M行N列のマトリクス状に印刷せよ。ただし $N \leq 7$ （つまり1行120以内17×7=119字におさまるものとする） $M < 100$ とする。

```
begin integer M, N, I, J;
  array X, Y, Z(1:100,1:7);
  READ(M, N);
  for I:=1 step 1 until M do
    for J:=1 step 1 until N do
      READ(X(I, J));
  for I:=1 step 1 until M do
    begin
      for J:=1 step 1 until N do
        begin
          READ(Y(I, J));
          Z(I, J) := X(I, J) + Y(I, J);
          PRINT('R', Z(I, J));
        end;
      PRINT('/', );
    end;
end
```

もし和のマトリクスZを印刷するだけが目的ならY, Zというarrayは設定しないで real Yを宣言しておいて

```
begin
  READ(Y);
  PRINT('R', X[I, J]+Y);
end;
```

とすればよい。

最後にJでくくつた次にPRINT('/');を入れてあるのはN個($N \leq 7$)の情報に次々と1行の右の方に入れてちょうどN個になつたらそれまでの分を1行に印刷して1行おくるためである。またXに読み込むための最初の部分は計算機によつては自動的に1つの命令で行なうものがある。(註)

M行N列のマトリクスXとN行L列のマトリクスYを, それぞれ個数を先頭に置いて入れてあるテープから読み込みマトリクス積

$$\begin{aligned} Z &= X, Y \\ N \\ Z[I, J] &= \sum X[I, J] \cdot Y[K, J] \\ K &= 1 \end{aligned}$$

を作りこれを印刷せよ。たゞし $L > 7$ となつたときは自動的に折つて印刷し、次の行とは1行あけるものとする。M, N, Lは高々40とする。

(註) 'PAL' では Readarray (X[1:M, 1:N]);

'230' では宣言で array X[1:M, 1:N]; となつているので
Readarray(X);

'H2'では READARRAY(X, 1, M, 1, N);

```

begin integer M, N, L, I, J, K ;
    array X, Y[1:40, 1:40];
    READ(M, N);
    for I:=1 step 1 until M do
        for J:=1 step 1 until N do
            READ(X[I, J]);
        READ(N, L);
        for I:=1 step 1 until N do
            for J:=1 step 1 until L do
                READ(Y[I, J]);
            for I:=1 step 1 until M do
                begin
                    for J:=1 step 1 until L do
                        begin real S;
                            S:=0;
                            for K:=1 step 1 until N do
                                S:=S+X[I, K]*Y[K, J];
                            PRINT('R', S);
                        end;
                    PRINT('/', '/');
                end;
            end;
        end;
    end;

```

S:=0としたのは御破算に願ひましてはの積りで和を作る時の常道である。
 Lが7の倍数+1のときは自動的に次の行にうつりこれが終ると2行改行する。

3次元以上の配列の取り扱いに当つて間違ひやすいのは2次元では最初の添字で行を次の添字で列を取り扱うが3次元ではこのマトリクスが何枚か重

ねられたときの何枚目かという添字が先頭に、次にマトリクスの行を、最後に列を示すことになる。

$$1 \left\{ \begin{array}{cccc} X_{111} & X_{112} & \cdots & X_{11n} & 1 \\ X_{121} & X_{122} & \cdots & X_{12n} & 2 \\ & \cdots & & & \vdots \\ X_{1m1} & X_{1m2} & \cdots & X_{1mn} & m \end{array} \right.$$

$$2 \left\{ \begin{array}{cccc} X_{211} & X_{212} & \cdots & X_{21n} & 1 \\ X_{221} & X_{222} & \cdots & X_{22n} & 2 \\ & \cdots & & & \vdots \\ X_{2m1} & X_{2m2} & \cdots & X_{2mn} & m \end{array} \right.$$

$$1 \left\{ \begin{array}{cccc} X_{l11} & X_{l12} & \cdots & X_{l1n} & 1 \\ X_{l21} & X_{l22} & \cdots & X_{l2n} & 2 \\ & \cdots & & & \vdots \\ X_{lm1} & X_{lm2} & \cdots & X_{lmn} & m \end{array} \right.$$

1 2 n

これを読み込んで格納するには第1ブロックの第1行の左第1列から第n列まで、次に第2列.....次に第2ブロックの順にデータテープに入れておく。いま2行20列のマトリクス10枚を20点の測点に関する載荷前と載荷後の値としそれを10回くり返して測定したデータとする。

そこでこの前後の値の差を出して各20点について10個の値の平均を印刷させるプログラムは次のようなものになる。データテープは先頭に点の数N, くり返し数L その次にX₁₁₁から順にX_{lmn}まで入れてするものとする。

```

begin integer I, J, K, M, N' L;
  array AA[ 1:10, 1:2, 1:20 ], BA[ 1:20 ];
  READ(N, L);
  for I:=1 step 1 until L do
    for J:=1 step 1 until 2 do
      for K:=1 step 1 until N do
        READ(AA[I, J, K]);
      for I:=1 step 1 until L do
        for K:=1 step 1 until N do
          AA[I, 1, K] := AA[I, 2, K] - AA[I, 1, K];
        for I:=1 step 1 until N do
          begin real S; S:=0;
            for J:=1 step 1 until L do
              S:=S+AA[J, 1, I];
            S:=S/L;
            PRINT(' I 2 B, R/', I, S);
          end;
        end;
      end;
    end;
  end;

```

一般に for 命令は内側のサイクルが終了してはじめてその外側で 1 単位お
くられるものであることは銘記する必要がある。

その度ごとに新しい値を入れるのではなくてコンスタントを配列に代入して
おく場合には数が少いときは

AA[1] := 1.4 0 ; AA[2] := 1.3 2 ;

BA[1] := 1.0 2 ; BA[2] := 1.1 6 ;

などと代入してもよいが数が多いときには大変である。たとえば F 分布表
5% 点の近以値は分母の自由度 ϕ_1 , 分子の自由度 ϕ_2 に対し

$$F_{0.05} = \left(a(\phi_1) + \frac{b(\phi_1)}{\phi_2} \right)^4$$

で a, b

ϕ_1	a	b
1	1.40	1.02
2	1.32	1.16
3	1.27	1.28
4	1.24	1.35
5	1.22	1.40
6	1.20	1.44
7	1.19	1.48
8	1.18	1.51
9	1.17	1.54
10	1.16	1.56

であるとする。このとき ϕ_1, ϕ_2 を読み込んで F005 を印刷するプログラムは

```

begin integer F1, F2, I;
  real FT5, AE;
  array A, B(1:10);
  READ(F1, F2);
  I := 0;
  for AE := 1.40, 1.32, 1.27, 1.24, 1.22, 1.20, 1.19,
    1.18, 1.17, 1.16, 1.02, 1.16, 1.28, 1.35,
    1.40, 1.44, 1.48, 1.51, 1.54, 1.56 do

```

```

begin
  if I < 11
    then A[I+1] := AE
    else B[I-10] := AE;
  I := I+1;
end;

FT5 := (A[F1] + B[F1]/F2) ↑ 4;
PRINT(' 2I, 2BR / ', F1, F2, FT5);
end;

```

2次元の配列への代入も

$A[11] := 1.40, A[12] := 1.32 \dots \dots$

といった形で書くと大変なのでたとえば前の例のA及BというベクトルをA
という配列の第1列及第2列とすれば

```

begin integer I, J, F1, F2;

  real AE, FT5;
  array A(1:10, 1:2);
  READ(F1, F2);
  I := J := 1;

  for AE := 1.40, 1.32, 1.27, ..., 1.56 do
    begin integer K;
      K := J;
      A[I, J] := AE;
      J := if K=2 then 1 else (J+1);
      I := if K=2 then I+1 else I;
    end;
    FT5 := (A[F1, 1] + A[F1, 2]/F2) ↑ 4;
    PRINT(' 2I, 2BR / ', F1, F2, FT5);
  end;
end;

```

でもし 10 行 2 列でなくて 10 行 4 列なら for 命令の中の if $K=2$ とある 2
ヶ所を if $K=4$ に直せばよい。

またその後で $(J+1)$ とカッコでくくつてあるのはもしカッコがないと
計算機は

```
J := (if K=2 then 1 else J)+1;
```

なのか

```
J := if K=2 then 1 else (J+1);
```

なのかを判別できないからであり、その次の $I :=$ の命令は右辺が if then
else の完全な式でしかも最後の I は混乱をおこしようもないのでそのまま
にしてあるもので、カッコは余分にあつても対応さえしていれば差し支えな
いので

```
I := if K=2 then (I+1) else I;
```

としても無論かまわないし、この方が分りやすい。

3 次元の配列に対する代入は

```
begin array A[1:10,1:2,1:20];
  real FT5, AE;
  integer I, J, K, M, N;
  I:=J:=K:=1;
  for AE:=1.40, ..... do
    begin integer M, N;
      M:=I; N:=J;
      A[I, J, K] := AE;
      K:=if M=10 then 1 else(K+1);
      J:=if M=10 then (J+1) else J;
      I:=if N=2 then (I+1) else I;
    end;
  end;
```

ということになる。

8.4 スイッチの取扱い

クリスマスカードを送るのに相手がフランス語を解する人なら1, 日本語を解する人なら2, 英語を解する人なら3を入れてやるとそれぞれ

JOYEUX NOEL

クリスマスおめでとう

MERRY CHRISTMAS

と印刷するプログラムを作れ

```
begin integer N ;
    switch S := P, Q, R ;
        P: PRINT("JOYEUX#NOEL"); go to END;
        Q: PRINT("KURISUMASU#OMEDETO"); go to END;
        R: PRINT("MERRY#CHRISTMAS"); go to END;
        READ(N); go to S[N];
        END: PRINT('/');
end ;
```

3ヶ所に go to END; という命令がくり返されているが ALGOL では後述の procedure の宣言によつて簡便な形にまとめることができる。

また 12月25日以降の配達ときはそれぞれ

BONNE ANNEE

新年おめでとう

A HAPPY NEW YEAR

と印刷させることにして国籍の N の次に受取月日をたとえば 12月23日なら 12.23 として入れておくことにしよう。2月から11月までは1応白紙ということにする。

この場合はスイッチを2重に制御して次のようなプログラムになる。

```

begin integer I, N;
    switch S1:=P1, P2;
    switch S2:=Q1;
    switch S3:=R1, R2;
    switch S:=S1[J], S2[J], S3[J];
    real D;
    procedure G:go to END;
    P1:PRINT("JOYEUX#NOEL");G;
    P2:PRINT("BONNE#ANNEE");G;
    Q1:PRINT("KURISUMASU#OMEDETO");G;
    Q2:PRINT("SINNEN#OMEDETO");G;
    R1:PRINT("MERRY#CHRISTMAS");G;
    R2:PRINT("A#HAPPY#NEW#YEAR");G;
    READ(N,D);
    I:=if D>2.1 V D<1.3 0
        then(if D≥12.01∧D≤12.25
            then 1 else 2 )else 2;
    go to S[N];
    END;
    PRINT('/');
end;

```

I には 1 2 月に入ってから 1 2 月 2 5 日までなら 1 がそれ以降 1 月 1 杯は 2 が入っているからフランス語なら N=1 で S1[2] go to 命令で S(1)つまり S1[J] 1 2 月 2 4 日なら J=1 で結局 P2 という名札のとこに行き BONNE ANNEE という印刷準備をし G; つまり go to END; という命令に従がって END つまり PRINT('/'); という命令を実行し 1 行に印刷して 1 行おくつて終りである。

8.5 一元配置の分散分析

因子数各因子に属するデータサイズ $N_1, \dots, N_k$ を先頭に第1因子 $MX_{11} \dots X_{1n_1}$ 第2因子 $X_{21} \dots X_{2n_2} \dots$ が入っているときこれを読み込んで

各因子ごとの平均 MX 不偏分散の平方根 RVX を印刷し、次に分散分析表

A O V TABLE

SV	SS	DF	MS	F
A	SSA	NA	FA	F $\bar{O}$
E	SSE	NE	FE	
$\bar{O}$	SS $\bar{O}$	NO		

を印刷するプログラムを作れ、たゞし

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

$$SSO = \sum_i \sum_j X_{ij}^2 - (\sum \sum X_{ij})^2 / \sum N_i = \sum \sum X_{ij}^2 - CF$$

$$SSA = (\sum_j X_{ij})^2 / N_i - CF$$

$$SSE = SSO - SSA$$

$$NA = M - 1$$

$$NO = \sum N_i - 1$$

$$NE = NO - NA$$

$$FA = SSA / NA$$

$$FE = SSE / NE$$

$$FO = FA / FE$$

とする。

実際はこの FO を F 分布表 5% 点と比較して A という分類に意味があるかどうかを検定するわけであるが、この手続きについては後で触れることとして、此処では $A O V$ 表を印刷するところまでとする。

```

begin integer M, I, J, NO, NA, NE;

integer array IA [ 1 : 10 ];
real SSA, SSE, SSO, CF, FA, FO, FE ;
array XA [ 1 : 20, 1 : 10 ], SX, MX, SSX, RV [ 1 : 10 ];
READ ( M );

for I:=1 step 1 until M do
    READ ( IA [ I ] );
for I:=1 step 1 until M do
    for J:=1 step 1 until IA [ I ] do
        READ ( XA [ I, J ] );
PRINT ( ' 5 B ' MEAN ' 15 B ' RV ' 15 B ' N ' / ' );
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
                S:=S+XA [ I, J ];
                SS:=SS+XA [ I, J ] 2;
            end;
        MX [ I ]:=S/IA [ I ];
        SSX [ I ]:=SS;
        RV [ I ]:=sqrt((SS-S2/IA [ I ])/(IA [ I ]-1));
        SX [ I ]:=S;
        PRINT ( ' 2 R, I / ', MX [ I ], RV [ I ], I );
    end;
SO:=SSO:=SSA:=0; NO:=NA:=0;
for I:=1 step 1 until M do

```

begin

SSO:=SSO+SSX(I);

SSA:=SSA+SX(I)²/IA(I);

NO:=NO+IA(I);

SO:=SO+SX(I);

end;

NO:=NO-1;

NA:=M-1;

NE:=NO-NA;

CF:=SO²/(NO+1);

SSO:=SSO-CF;

SSA:=SSA-CF;

SSE:=SSO-SSA;

FA:=SSA/NA;

FE:=SSE/NE;

FO:=FA/FE;

PRINT('10B'AOV#TABLE'/');

PRINT('5B'SV'14B'SS'14B'MS'15B
'F'/');

PRINT('A'R,I,2R/',SSA,NA,FA,FO);

PRINT('E'RI/',SSE,NE,FE);

PRINT('20(' - ')/');

PRINT('O'R,I/',SSO,NO);

end;

9 手 続(procedure)

同じような手続で計算を行なう部分が何箇所あるとき、そのたびにくり返してほとんど同じプログラムを書くことを避け、この手続全体に適当な名前をつけて、この名前を呼び出すことによつて実行させようとするものでサブルーチンに当る。この場合、ある演算の結果、一つの数値が得られる場合は、その結果を手続全体につけた名前のところに入れておいてもよい。後者の場合を関数指示子といい、使用するときは実行命令としてでなく式の中で変数と同様に用いる。いずれの場合も宣言部において宣言してから使用する。

9.1 手続命令(procedure S)

手続命令を使用するに先立つて、その手続の名前やパラメタとその内容を宣言しておかなければならない。このための手続宣言は次の形をとる。

```
procedure  $\bar{I}$  ( $\bar{F}\bar{P}$ ,  $\bar{F}\bar{P}$ , ...,  $\bar{F}\bar{P}$ )  
  <value part> ; <specification part>;  
  <procedure body> ;
```

名前の次の($\bar{F}\bar{P}$, ..., $\bar{F}\bar{P}$) はパラメタを使用しない手続の場合は不用である。value で呼ぶパラメタはすべてその型を specification part において明細指定しておかなければならない。^{*38} また、外からその変数の値そのものを受取つて来る変数は明細指定部に先行する value part で value 指定をする必要がある。値で呼んでならぬパラメタは、たとえば、その手続の結果値が得られるようなもので^{*39}、値で呼ばれるパラメタに対し名前で呼ばれるパラメタと云う。

* 38 'H1' 'H2' 'J' 'PAL' では名前で呼ぶパラメタも指定を必要とする。

* 39 'H1' 'H2' '230' 'J' 'PAL' では他に array などがある。

何回も本体の中で使うパラメタで、値で呼ぶことのできるものは value 指定した方が計算が早くなる（名前で呼ぶ場合はいちいちそのパラメタの番地を頼りに情報を受取りに行かれない）。value 指定できるのは $\langle \text{type} \rangle$, array, $\langle \text{type} \rangle$ array, label である *39。

手続命令は上述の手続宣言がしてあるものについて

$\bar{I}(\bar{AP}, \bar{AP}, \dots, \bar{AP});$

として使用する。もちろんパラメタのないものはカッコの部分が不用であり、ここに書かれた $\bar{AP}$ （実パラメタ）の名前なり値なりが対応する $\bar{FP}$ のところに置き換えられて手続が実行される。

明細指定子 (specifier) としては integer, real, Boolean, array, integer array, Boolean array, label, switch, procedure, integer procedure, real procedure, Boolean procedure, string *40がある。

この場合宣言のときとよく似ているが、この明細指定は宣言とは異なつたものであり、たとえば array, procedure など名前だけを書くべき、宣言の時のように $[\dots]$, $(\dots)$ をつけてはならない。

たとえば Simpson 公式によつて

$$\int_{LL}^{UL} f(x) dx$$

をきざみ n や LL, UL および関数 $f(x)$ を指定して計算させる手続は、いま $n = 100$, きざみの幅を h , 答を S として

$$S = \frac{h}{3} (f_0 + 4f_1 + 2f_2 + 4f_3 + 2f_4 + 4f_{99} + f_{100})$$

であるから、きざみ M , 下限 L , 上限 U を何組か読み込んで

* 39 'H1', 'H2', 'I', 'J', 'PAL' 230' では $\langle \text{type} \rangle$ に限るし, 'J' ではさらに Boolean を除く。また $\langle \text{type} \rangle$ とは real, integer, Boolean を意味する。

* 40 'H1', 'J', 'PAL' では string がない。

$$f(x) = \log(1+x)$$

を計算し印刷するプログラムはデータの終了マークを0とすれば

```
begin comment SIMPSON RULE;
  real L,U; integer M;
L 3 : real procedure FUNC(X);
    value X; real X,
    FUNC := 0.4342944 * ln(1.0 + X);
L 6 : procedure SIMPSON(N, LL, UL, F);
    value N, LL, UL; real LL, UL;
    integer N; real LL, UL;
    real procedure F;
    begin integer I;
      real S, H, FN, FI;
L 12 : FN := FLOAT(N);
      S := FUNC(LL);
      H := (UL - LL) / FN;
L 16 : for I := 1 step 2 until N - 1 do
        begin FI := FLOAT(I);
          S := S + 4.0 * F(LL + FI * H)
            + 2.0 * F(LL + (FI + 1.0)
              * H);
        end;
      S := S / 3.0 * (S - F(LL + FN * H));
      PRINT(' 2R/' , N, S);
    end of SIMPSON;
```

```

L24:START:READ(M,L,U);
      if M=0 then go to END; PRINT('10/');
      SIMPSON(M,L,U,FUNC);
      go to START;
END:

```

end SIMPSON RULE;

ということになる^{*40}。

この中で real procedure FUNC(X); は次のべる関数指示子の宣言を行っているもので、この場合は FUNC という名の所に答が入り、それが、17, 18 行目の式の中で使われているので、Procedure のパラメタとして形式的に F という名の関数指示子として明細指定しておいたわけである。

L 3 等のレーベルはエラーチェックの段階で、エラーが何行目にあるかを分りやすくするためにつけたもので、S の頭になら、いくらつけても別段害にならない。

本体の中で 26 行目で M, L, U, FUNC という実際のパラメタを指定して手続命令が出されているが、これによつて宣言部の形式的パラメタ N, LL, UL, F とそれぞれ値および名前のやりとりをして宣言部の手順を実行する。

* 40 'J', 'PAL' では procedure の名前をパラメタにしたときは、それを proceduer 中で使うときの実パラメタは name でなければならないから、一たん 10 行目で real K, A, B; としておいて、16 行の後で A:=LL+FI*H; B:=A+H; と受けておいて、S:=S+4.0*F(A)+2.0*F(B); また 19 行の前で K:=LL+FN*H; として 19 行も書き直す必要がある。また 'H 1' では actual parameter に式を書くためには formal parameter は value 指定を要する。

string の使い方は $A = \times \times \times$ $B = \times \times \times$ と次々印刷したい場合にその計算結果をある procedure によつて実行し、字を印刷する部分だけ取り替えなければ

```

procedure PR(Q,ST);
  value Q; real Q; string ST;
begin real L;
  S; S; ... ; L := A E;
  PRINT( 'STR//', L );
end;

```

としておいて本体のところで Q が 2.5 のとき、 A を、 Q が 4.8 のとき B をプリントしたければ

```

PR( 2.5, 'A=' );
PR( 4.8, 'B=' );

```

とすればよい。

同じ手続きだが場合によつて LP と LS に接続したいときには明細指定子の label を用い

```

procedure XY(A, LA);
  value A; integer A; label LA;
begin S; S; ... ; S; go to LA; end;

```

しておいて本体のところで

```

XY( 1, LP );
XY( 2, LS );

```

などとすればよい。

switch をパラメタに用いるのは上の例であらかじめ

```

switch P: = L1, L2;

```

```

procedure XY(A, P);
  value A; integer A; switch P;
  begin S; S;
    .....
    S;
    go to P[A];
  end;

```

としておき、本体の計算途中で I にしたがってどちらかに行きたければ

```
XY(I, P);
```

としておくだけで、その時の I が 1 ならば P_1 に、2 なら P_2 に行つてくれるわけである。

パラメタに他の手順の名前を指定しても良いが、自分の名前を書くことは再帰的呼出し (recursive call) として禁止されている。このことは関数指示子の場合でも同様である。

この手続と関数指示子はその使用頻度が多ければそれ自身を機械に記憶させておいて、使用するときは無宣言であたかも辞書を引かせるようなやり方で気軽に使用することも可能で、土木研究所でもこの方法によるプログラミングの簡略化を目論んでいる。もちろん、そのときは指定された名前はもはや reserved word として他の変数や手続、名札などの名前として使用することはできない。

9.2 関数指示子 (function designator)

前節にのべたように一つの結果が得られる手続でその結果を手続自身の名前の所に入れておき、これを式の中で使おうとするもので ALGOL 60 の

```

sin (A E)   cos (A E)   arctan (A E)   ln (A E)
exp (A E)   sqrt (A E)   sign (A E)   abs (A E)

```

`entier(  $\widetilde{AE}$  ) *40`

は実はある意味で関数指示子に外ならないわけである(これらに限つて無宣言で使用して良い)。sin, cos, arctan はラジアンで取扱う。

sign はシグナムと読み $\widetilde{AE}$ が正なら 1, 0 なら 0, 負なら -1 が入り, abs は絶対値, entier はアンティエと読み $\widetilde{AE}$ を越さぬ最大の整数をとる。したがつて entier(4.3) は 4 であるが entier(-4.3) は -5 となる。

以上はいずれも無宣言で使つて良いが、これと同じ働きをするものを関数指示子宣言によつて作り出すことができる。やり方は手続の場合と大体同じであるが違ふのは宣言子が

real procedure, integer procedure, Boolean procedure と前に <type> がつくことと procedure body の中で少なくとも 1 回自分自身の名前 $\widetilde{I}$ に対して

$\widetilde{I} := \widetilde{AE};$

という代入命令が来なければならぬ点である。

また使用する際は手続命令と違いそれ自身が命令になるのでなく前節の 17, 18 行目のように算術式の中で使用される。

(正)

```
real procedure INNER PRODUCT(N, A, B);  
  value N; integer N; array A, B;
```

*40 " もちろん大文字しか使えない 'J', 'PAL', 'B', 'H1', 'H2' ではこれらの標準関数もすべて大文字である。'J'では FIX, '230'では fix があり四捨五入(負の場合も)行なう。'H1', 'H2'には FIXがあり切りすてを行う。'I'では arcsin, arccos, tan がある。また'B'では REAL, BOOLEAN があり、実数型とブール型の変換を行なう。'PAL', '230'には float, 'J', 'H1', 'H2'には FLOAT があり整数を実数に直す。

```

begin real S; integer I;
  S := 0.0;
  for I = 1 step 1 until N do
    S := S + A[I] * B[I];
  INNER PRODUCT := S;
end;

```

と宣言しておき、本文中でたとえばZにベクトルXとY（その大きさはM）の内積を代入するならば（M, X, Y）：

Z := INNER PRODUCT;

10 その他

10.1 コメント (comment)

今まで何度も出て来たがプログラムのメモ用として

- 1 begin および; の後に comment と書けばその後; が出て来るまでの間何を書いても機械は無視する。
- 2 end の後は次の; が出てくるまでの間 end, else 以外は何か書いても機械は無視する。

という2つの場合を利用することができる。そこで一つの手順が end で終つたら何が終つたかを書いておいたり、途中でこここの段階でどうなつていなければならぬかという自分のメモを comment ; の間にはさんだり、begin の後にこのプログラムはどのような内容のもので、後で使う人がどのように注意しなければならぬかを書いて; でしめくくつておくことなどに利用すると便利である。

実は; は S の区切りに用いるので end の前は不要であり最後の end の後は無視されるので; を書く必要はないが、後でつなぎ合わせる時に落したり end の前の; を省略して良いということを逆に end の後で一般的に省略できるように誤解して end の次に S が来るのに; を落したりするので

else の直前以外は必ず ; を入れることにしておく。

10.2. FOR $\bar{S}$ の出口

FOR $\bar{S}$ の実行が終るのは do 以下の $\bar{S}$ が処理し終ったときか、または do 以下の $\bar{S}$ の中で $G\bar{O} \quad T\bar{O} \quad \bar{S}$ が成立したときである。前者の場合はつきつきと変えられて来たループ変数の値はきまらない、後者の場合は $G\bar{O} \quad T\bar{O} \quad \bar{S}$ の実行直前の値になっている。

10.3. own

変数の値はそれが宣言されたブロックから外に出た時に失われてしまう。しかし外でもその値を保持したいときは、もちろんその外側のブロックで宣言しておけば良いが、それ以外にブロック内でタイプ宣言するとき
own real A ; のように前につけておけばよい *41。

10.4. 条件付 FOR 命令

ALGOL 60 にはこのような言葉がない。5.4 で条件は命令の3つの型をのべたがオ3のものについて else 以下があつたらどうなるかを考察してみよう。

```
if A > 0 then for I: = 1 step 1 until N do
    S := S + P * I else S := P ;
```

というプログラムには別段あいまいな所はないから

```
if BE then  $\bar{FS}$  else  $\bar{S}$  ;
```

は認められて良さそうなものである。しかし do の後に条件付命令が来ると事情は変わってくる。do の後の $\bar{S}$ が else を含まない場合はそれでよいが、たとえば、else を含むものとして前の例を変えて

*41 'I', 'J', '2 3 0', 'H1', 'H2' には own がない。

```

if A > 0 then for I := 1 step 1 until N do
  if B = 0 then S := S + P * I
    else S := P ;

```

とすれば3行目の else はどの if に対応するのかというあいまいさを持つこととなる。このゆえに then $\overline{FS}$ の後に else が来ることを禁止したものと考えられるがここで無条件 FOR 命令という概念を設定して do の後は条件付命令が来てはならないことにすればよい。今のままの文法でも大半のものはおそらく then $\overline{FS}$ の後に else が来たらエラーと判定することの技術的困難さから、たとえば else は一ばん近くの if に対応するものとして計算を進めるもの^{*42}などもある。手つとり早い方法は then の後で for $\overline{S}$ 全体を begin end でくくってしまう方法で、このように多少あいまいだと感じた時はたとえ無駄であつても statement なら begin end で、式なら丸カッコでくくって単純命令、単純式にしていまえばよい。このようなカッコは無駄なものであつたとしても全く無害である。

10.5 push down list

記憶容量の制限から、たとえば 'PAL' ではコンスタントは50個に限るとか入れ子(nest)は25段に限るといった制約がついている。これはコンパイラ製作者に聞かないとわからぬことが多い。

*42 '230', 'PAL', 'H1', 'H2' など

11 I/O 命令 (input -- output S)

11.1 OKITAC 5090H用 NEW ALGOL-H 'H2'

READ(I, I, ..., I);

テープより順次数値を読み、対応するIという名前のところに格納する。

変数の型は何であつてもIに合せて変換する。

READARRAY(I, i_{a11} , i_{a21} , i_{b11} , i_{b21} , ...);

テープよりIという名の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);

である。

array Aの宣言された通りに読み込んで良いなら、READ(A);
でよい。

PRINT("XX...X");

XX...Xをそのまま印刷する準備をする。外側の'はフォーマットを示すもの。内側の'はその中のものがそのまま印字したいstringであることを示すもので、後の一般のプリントフォーマットの中に書き込んでもよい。(' 3 ('XX ...X) '); とすれば3回くり返される。

PRINT('5B');

スペース(ブランク)5個をおくもので、符号のない整数なら何を書いてもよい。また後述のXを用いてXBとして変数で個数を指定することもできる。

PRINT('2/');

2回復帰改行する。2のところはBの場合と同じ何個でもよい。

/がくると今迄の情報を1行に印刷して指定された行数改行する。

また120字なり、135字なりのLP(ラインプリンタ)がある

情報が入ることによつてオーバーすると、その最後の情報の前まで
で1行分印刷し、最後の情報は新しい行の先頭に入る。

PRINT('2↑');

↑がくるとLP用紙の先頭までおくれられ、今の場合は1枚白紙をお
いて2枚目の先頭から印刷が開始される。

PRINT('R', I); PRINT('I', I); PRINT('L', I);

簡単な形で夫々 real, integer, Boolean型の情報を印字準備さ
せるものである

'R' □ □ ± . $\underbrace{XX \cdots X}_{9 \text{ ケタ}}_{10} \underbrace{\pm XX}_{2 \text{ ケタ}}$
(ゼロサプレス)

'I' □ □ □ ± $XX \cdots X$
13ケタ(ゼロサプレス)

'L' □ □ TRUE □ 又は □ □ FALSE

でこれらは整数個のくり返しを指定してもよいし、前の/B↑
とくみあわせてもよい。ただしR, I, Lは次のF, D, Z, A,
Sと同様カンマで切る必要がある。

integer B1の右にスペースを2個入れ、real A1, C2, RN
を用紙の先頭から印字し、A1の左にDATA=という字を入れ1
行改行したいならば、

PRINT('↑I2B, 'DATA='3R/', B1, A1, C2, RN
) ; となる。

PRINT(' -4Z.3D, 2B-8Z, 2B-, 7D₁₀-2D', I₁, I₂, I₃);

Zはゼロサプレスする桁

Dはゼロサプレスしない桁

-又は+は符号をつける要求

・ 10はその位置への印字

で、今の場合 $\bar{I}_1$ はrealで小数点上4ケタ、下3ケタ、 $\bar{I}_2$ はintegerで8ケタ、 $\bar{I}_3$ は $\pm.XX \dots \times 10 \pm XX$ の形の印刷だが何れも間にblankを入れないと情報がつながってしまうおそれがある。

これらはカッコにつつんで、3(-4Z, 3D)とくり返させることもできる。

```
PRINT('5F',  $\bar{I}$ );
```

Boolean型の印刷で、TRUE又はFALSEを印字する'F'とすればT又はFを印字。'P'とすれば1又は0を印字する。

```
PRINT('7A',  $\bar{I}$ );
```

$\bar{I}$ がアルファモードのときその文字内容をそのまま7文字打つ(1語は7文字以内である)。アルファモードの読み込みは後にのべるが、プログラム中で代入するときは、integer Kに対しP39Aという文字を代入したければ

```
K := EQUIV(P39A);
```

としておいて

```
PRINT( 4A , K);
```

でよい。

```
PUT(<address>,  $\bar{I}$ );
```

```
GET(<address>,  $\bar{I}$ );
```

MT(マグネティックテープ)に対するデータの書き込み及び読み出しであつて配列ならば $\bar{I}$ のところにarrayの名前を書く。

addressは先頭番地で1番から使用した方が早い。

データの作り方はREAD命令に対応してデータをカンマでつなぎ最後に⑩⑩を入れておけばよい。もし同じ12.5というデータが10個連続するときは10*12.5でよい。またカンマだけがあると0が読み込まれる。文字の読み込みは=YOKOHAM, =A, =TOKYO, =, ...等とす

る。＝の後は文字として読み込み(7字まで)何もなければブランクを読み込む。但し文字の間では#をもつてブランクとする。

読み込みの1時停止はhalt; ⑩⑪である。論理定数の読み込みはF又はT(後に字がついていてよい)によつて行う。フォーマット中のSARIL DZF及びB/↑等を変数でくり返し個数指定を行いたいとき、たとえば↑をN回、RをM個、IをP個、/をQ個でその順に用いるなら

```
REPX(N, M, P, Q);
```

としておいて

```
PRINT('X↑XR, XIX/',  $\tilde{I}$ , .....,  $\tilde{I}$ );
```

と書く。

11.2 OKITAC 5090 H 'H1'

```
READ( $\tilde{I}$ ,  $\tilde{I}$ , .....,  $\tilde{I}$ );
```

テープより順次数値を読み込み対応する $\tilde{I}$ という名の場所に格納、変数の型は何でもよい。

```
PRINT(n, AE);
```

AEの結果を有効数字nケタで印刷(ただし、ラインプリンタではこの情報をつぎつぎと1行分のスペースにたくわえ、次のCRLFで印刷)

実数型 $\square \pm . \underbrace{XX \cdots XX}_n \square \pm XX \square$

整数型 $\square \pm \underbrace{XX \cdots XX}_n \times$ (初めの0は消す)

ただし□は空白の意

```
PRINT('XX .....X');
```

' 'でつつまれた中のものをそのまま印刷(ただし、ラインプリンタではその情報をたくわえるだけ)スペース記号は#とする。

```
PRINT(7, B̃E);
  TRUE 又は FALSE
PRINT(m1, n2, ÃE);
  実数型  [ ± X ..... X . X ..... X ]
                n1          n2
```

SPACE(IÃE);

整数値の結果となる算術式により、IÃE個のスペースをとる(ライ
ンプリンタでは情報をたくわえる)。

CRLF(IÃE);

復帰改行(carriage return line feed)の略で、今までたく
わえられた120字以下の情報をプリントし、IÃE行おくる。もし
すでに120字を越すときは自動的に120字を越すことになった
命令までの分を1行印刷し、1行おくつて次の行の最初から今の最
後の命令分の情報をたくわえる。

MTに対しては、PUT(<address>, Ĩ); GET(<address>, Ĩ); 読込中止はhalt; ⑩⑩

データはカンマでつないで最後に; ⑩⑩を入れる。

11.3 JUSE ALGOL 'J'

READREAL(Ĩ);

実数を1組テープより読み込みĨに格納

READINTEGER(Ĩ);

整数を1組テープより読み込みĨに格納

PRINTREAL(ÃE);

ÃEを実数型で印刷

PRINTINTEGER(ÃE);

ÃEを整数型で印刷

PRINTSTRING('XX X');

でつづまれた中のものをそのまま印刷。

CRLF;

復帰改行を1回。

11.4 ISSP ALGOL 'I'

readreal($\tilde{I}$);

実数を1組テープより読み、 $\tilde{I}$ に格納

readinteger($\tilde{I}$);

整数を1組テープより読み、 $\tilde{I}$ に格納

READREAL n ($n, \tilde{I}_1, \tilde{I}_2, \dots, \tilde{I}_n$);

実数を n 組テープより読み対応する $\tilde{I}$ に格納

READ INTEGER n ($n, I_1, I_2, \dots, I_n$);

整数を n 組テープより読み対応する $\tilde{I}$ に格納

printreal ($n, \tilde{AE}$);

符号, 小数点, 指数, 空白を含め $n+7$ ケタ印刷

printinteger ($n, \tilde{AE}$);

符号空白を含め $n+2$ ケタ印刷

printfree ($n_1, n_2, \tilde{AE}$);

符号, 小数点以下 n_1 ケタ, 小数点, 小数点以下 n_2 ケタと空白を印刷

printstring (' $\times \times \dots \times$ ');

' ' でつづまれた中のものをそのまま印刷

space (n);

空白 n 個

crlf;

復帰改行

11.5 FACOM 230 ALGOL '230'

Readreal (I);

1語テープより読みIに代入

Readinteger (I);

1語テープより読みIに代入

Readarray (I);

配列を列なりに読みIに順次代入

Printreal (AE, n);

実数AEをnケタfloatingで印刷

Printinteger (AE);

整数AEを印刷

Printfix (AE, n1, n2);

実数AEを小数点以上n1ケタ以下n2ケタ印刷

Print (AE);

実数を10ケタfloatingで印刷

Printx(AE, n);

整数をnケタ印刷

Printstring('××...×');

' 'の中ものを印刷

Space (n);

n個空白をとる。

CRLF;

1行復帰改行

LFEEED;

10回復帰改行(ラインプリンタではHome Positionまで)。

11.6 OKIPAL PAL

数値を print out するためには必ず宣言部において、フォーマット宣言がされていなければならない。

```
Format I [-0.810 -2, 'A#', 54, 'B', ('#')↑53, -710-1,
          'C*D', (-5, '#', -2)↑3];
```

これは Format 名が I でそれぞれ次の形の情報が 1 行に印刷される。

```
±XXXXXXXXXX10±XXALJLJXXXXXXXX
. XXXXBLLJLLJLJLJ XXX. XXXXXXXX
10±XC*D±XXXXXX ±XX±XXXXXX
LJ±XX±XXXXXLLJ±XX
```

これは 8 8 字分であるが 1 2 0 字以下でなければならない。

```
Read n (I1, I2, ..., In);
```

テープより n 組の数値を読み込み対応する I に格納する。

```
Readarray (I[n1:n2, ..., m1:m2]);
```

テープより行なりに配列の要素を読み込み格納

```
Print (I, AE1, ..., AEm);
```

フォーマット I にしたがひ順次 AE を 1 行に印刷し 1 行 feed。

```
Print out;
```

フォーマットに満たなくても印刷

```
Print string(XX...X);
```

' ' の中のものを 1 行に印刷し 1 行 feed。

```
Feed (n);
```

n 行おくり

```
Read tape (I[n1:n2, ..., m1:m2], S, R);
```

磁気テープより読み出し。S は機種番号, R は SV で示した先頭番地。

```
Write tape (I[n1:n2, ..., m1:m2], S, R);
```

磁気テープへの書き込み。

11.7 ALCORグループ用 'A'

read($\overline{I}$, $\overline{I}$, ..., $\overline{I}$);

print($\overline{E}$, $\overline{E}$, ..., $\overline{E}$);

但し $\overline{E}$ は算術式及ブール式である。

12° アルゴリズムによるプログラム例 (2)

12.1 危険度の評価

例として計画洪水量の危険度の評価の計算をとり上げてみよう。過去の N 年間の最大洪水量のデータが何組もあり、今後 M 年間に $M-1$ 個は下廻わるはず (この予測の信頼度は 95%) の値は

$$P = 1/N$$

$$\text{平均値} \quad MLX = (\sum \log X / N)$$

不偏分散の平方根

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

とすれば

$$Q(P) = MLX + H(P) \cdot PV LX$$

$$H(P) = \frac{K_p + \sqrt{\left(1 - \frac{K_{0.05}^2}{2(n-1)}\right) \left(K_p^2 - \frac{K_{0.05}^2}{n}\right)}}{1 - \frac{K_{0.05}^2}{2(n-1)}}$$

で K_p は正規偏差で

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

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

$$K_p = \eta - \left\{ \frac{2.30753 + 0.27061\eta}{1 + 0.99229\eta + 0.04481\eta^2} \right\}$$

$$\eta = \sqrt{\ln \frac{1}{p^2}}$$

がある。

テープより N_i 個のデータ群を読み込み $M=2$ 年, 5 年, 10 年, 20 年, 50 年, 100 年, 200 年, 500 年に対する値を印刷させるプログラムを作りたい。ただし一群のデータの終りには 0.0 が全部の終りには 1010 が入っているものとする。

```
begin real X, LX, MLX, SLX, SLXX, RVLX, FN, P, H;
```

```
integer I;
```

```
real procedure KPF(P);
```

```
value P; real P;
```

```
begin real K;
```

```
  K:=sqrt(ln(1.0/P↑2));
```

```
  KPF:=K-(230753+0.27061*K)
```

```
    /(1.0+0.99229*K  
      +0.04481*K↑2);
```

```
end;
```

```
START 1:SLK:=SLXX:=0; I:=0;
```

```
START 2:READ(X);
```

```
  if X=0.0 then go to START 3;
```

```
  if X=10.10 then go to END;
```

```
  LX:=ln(X);
```

```
  SLX:=SLX+LX;
```

```
  SLXX:=SLXX+LX↑2;
```

```
  I:=I+1;
```

```
  go to START 2;
```

```
START 3:FN:=I; PRINT(' I\' );
```

```
  MLX:=SLX/FN;
```

```
  KP:=KPF(P);
```

```
  K5:=KPF(0.05);
```

```
  RVLX:=sqrt((SLXX-MLX*SLX)
```

```
    /(FN-1.0));
```

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

```

begin H:=(KP+sqrt(KP↑2-(1.0
-K5↑2/2.0/(FN-1.0))
*(KP↑2-K5↑2/FN)))/
(1.0-K5↑2/2.0/(FM-1.0));
PRINT ('/', EXP(M LX+H*RV LX));

end;

PRINT ('5/');
go to START 1;
END:

end;

```

12.2 一元配置の分散分析(2)

8.5 であげた一元配置の分散分析を一つの procedure にまとめあげ、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, 及び F 値 $F\bar{O}$ を計算し $A\bar{O}V$ 表を印刷する procedure $A\bar{O}V1$ を作り、これを用いるプログラムを作ってみよう。

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。

今度は procedure $A\bar{O}V1$ の外で必要なのは形式的パラメタに対応する (名前は違っても良いが面倒なので形式的パラメタと同じ名前にした) M, IA, XA, MX, RV, $F\bar{O}$ のほかには読み込みの個数制御に用いる I, J 位のものである。

procedure $A\bar{O}V1$ の中からは READ 命令を外し、プログラム本体で先づ M と次に IA 次に XA を次々と前の情報を利用して読み込み、やりたいことは $A\bar{O}V1(\dots)$; と 1 行ですましてしまうことができる。尚プログラムは他の言葉への翻訳の例として 'H1' によつて書いてみよう。

```

begin integer I, J, M; real MX, RV,  $\overline{FO}$ ;

integer array IA[1:10];
array XA[1:20, 1:10], MX, RV[1:10];
procedure AOV1(M, IA, XA, MX, RV,  $\overline{FO}$ );
  value M;
  integer M; integer array IA;
  array XA, MX, RV; real  $\overline{FO}$ ;
begin
  integer I, J,  $\overline{NO}$ , NA, NE;
  real SSA, SSE,  $\overline{SSO}$ , CF, FA,  $\overline{FO}$ ;
  array SX, SSX[1:10];
  PRINT(' #####MEAN#####RV#####
        #N'); CRLF(1);
  for I:=1 step 1 until M do
    begin
      real S, SS;
      S:=SS:=0;
      for J:=1 step 1 until IA[I] do
        begin
          S:=S+XA[I, J];
          SS:=SS+XA[I, J]2;
        end;
      MX[I]:=S/FL $\overline{O}$ AT(IA[I]);
      SSX[I]:=SS;
      SX[I]:=S;
      RV[I]:=SQRT((S-S2/FL $\overline{O}$ AT(IA[I])
                    /(FL $\overline{O}$ AT(IA[I])-1.0)));
    end;
end;

```

```

PRINT( 5.5, MX(I));
PRINT( 5.5, RV(I));
PRINT( 2, I);
CRLF(1);
end;
S $\bar{O}$ :=SS $\bar{O}$ :=SSA:= $\bar{O}$ ; N $\bar{O}$ :=NA:= $\bar{O}$ ;
for I:=1 step 1 until M do
begin
SS $\bar{O}$ :=SS $\bar{O}$ +SSX(I);
SSA:=SSA+SX(I) $\uparrow$ 2/
      FLOAT(IA(I));
NO:=N $\bar{O}$ +IA(I);
SO:=S $\bar{O}$ +SX(I);
end;
N $\bar{O}$ :=N $\bar{O}$ -1;
NA:=M-1;
NE:=N $\bar{O}$ -NA;
CF:=S $\bar{O}$  $\uparrow$ 2/FLOAT(N $\bar{O}$ +1);
SS $\bar{O}$ :=SS $\bar{O}$ -CF;
SSA:=SSA-CF;
SSE:=SS $\bar{O}$ -SSA;
FA:=SSA/FLOAT(NA);
FE:=SSE/FLOAT(NE);
F $\bar{O}$ :=FA/FE;
PRINT( '#####A $\bar{O}$ V#TABLE' );
CRLF(1);
PRINT( '##SV#####SS#####DF####'

```

```

      ###MS#####F '); CRLF(1);
PRINT(' ##A ');
PRINT(4.4, SSA);
PRINT(3, NA);
PRINT(3.3, FA);
PRINT(3.3, F $\bar{O}$ ); CRLF(1);
PRINT(' ##E ');
PRINT(4.4, SSE); PRINT(3, NE); CRLF(1);
PRINT(3.3, FE)
PRINT('-----');
CRLF(1);
PRINT(' ## $\bar{O}$  '); PRINT(4.4, SS $\bar{O}$ );
PRINT(3, N $\bar{O}$ ); CRLF(1);
end;
READ(M);
for I:=1 step 1 until M do
  READ(IA[I]);
  for I:=1 step 1 until M do
    for J:=1 step 1 until IA[I] do
      READ(XA[I, J]);
    A $\bar{O}$ V1(M, IA, XA, MX, RV, F $\bar{O}$ );
  end;
end;

```

12.3 平均及び不偏分散の平方根の手続

3.3 例4にあげた不偏分散の平方根を求めるプログラムを個数N, 配列Aを与えられて平均をMX, 不偏分散の平方根をRVに格納する手続を作り, 個数を先頭に何組かのデータを読んでプリントさせるプログラ

μを 'H 2' で作つてみる。但し最後に 0 を入れておいてデータの終りを判定させることにする。各組のデータ数は高々 100 個としよう。

```

begin integer I, N;
  real MX, RV;
  array A[1:100];
  procedure READA1(M, AA);
    value M;
    integer M; array AA;
    begin integer I;
      for I:=1 step 1 until M do
        READ(AA[I]);
      end;
  procedure MXRV(N, A, MX, RV);
    value N;
    integer N; real MX, RV; array A;
    begin integer I; real SX, SXX;
      SX:=SXX:=0;
      for I:=1 step 1 until N do
        begin
          SX:=SX+A[I];
          SXX:=SXX+A[I]2;
        end;
      MX:=SX/N;
      RV:=SQRT((SXX-SX*MX)/(N-1));
    end;
  ST:READ(N);
  if N=0 then go to END;

```

```

READA1(N, A);
AOV1(N, A, MX, RV);
PRINT('I. 2R', I, MX, RV);
go to ST;
END;
end;

```

12.4 積和平方和マトリクス及相関マトリクスを求める手続

3.3 例5で相関係数を求めたが

$$\text{積和を } S(XY) = \sum XY - (\sum X)(\sum Y)/M$$

と書けば積和平方和マトリクスは

$$\begin{pmatrix} S(X_1 X_1) & S(X_1 X_2) & \cdots & S(X_1 X_n) \\ & S(X_2 X_2) & \cdots & S(X_2 X_n) \\ & & \ddots & \\ & & & S(X_n X_n) \end{pmatrix}$$

という対称マトリクスであり相関マトリクスは

$$R_{12} = \frac{S(X_1 X_2)}{\sqrt{S(X_1 X_1) S(X_2 X_2)}}$$

であるから対応する対角要素の積の平方根で交点の要素を割つたものが相関マトリクスとなり、

$$\begin{pmatrix} 1 & R_{12} & \cdots & R_{1n} \\ & 1 & \cdots & R_{2n} \\ & & \ddots & \\ & & & 1 \end{pmatrix}$$

という形をとる。

procedureとしてはSUM12(N, A1, A2, P)という関数指示子を作りPが1のときはA1だけの1次の和, Pが2のときはA1 A2の内

積を作らせてそれをSUMという名のところに格納させることにする。
 データはM行N列の配列に入れこの各列を1次元のarrayに移してはSUMを行なうこととし、TRANという移動用のprocedureも使つておく
 因子数Nは10以下データ組数Mは100以下としておく。

```
begin integer I, J, M, N, P;
  array XA[1:100, 1:10], SS[1:10, 1:10], A1,
    A2[1:100];
  procedure TRAN(M, N, XA, A);
    value M, N;
    integer M, N; array XA, A;
    begin
      integer I;
      for I:=1 step 1 until M do
        A[I]:=XA[I, N];
      end;
    real procedure SUM12(M, A1, A2, P);
      value M, P;
      integer M, P;
      begin
        integer I; real S;
        S:=0;
        for I:=1 step 1 until M do
          S:=S+A1[I]*(if P=1 then 1 else
            A2[I]);
          SUM12:=S;
        end;
```

```

READ(M, N);
for I:=1 step 1 until M do
  for J:=1 step 1 until N do
    READ(XA(I, J));
for I:=1 step 1 until N do
  begin
    for J:=1 step 1 until N do
      PRINT('17B');
    for J:=1 step 1 until N do
      begin
        real A;
        TRAN(M, I, XA, A1);
        TRAN(M, J, XA, A2);
        A:=SUM12(M, A1, A2, 2)-SUM12(M,
          A1, A1, 1)+SUM12(M, A2, A2, 2)
          /M;
        PRINT('R', A);
      end;
    PRINT('//');
  end;
for I:=1 step 1 until N do
  begin
    for J:=1 step 1 until I-1 do
      PRINT('17B');
    for J:=I step 1 until N do
      begin
        real A;

```

```

        A:=SS[I, J]/SQRT(SS[I, I]*SS[J,
            J]);
        PRINT('R', A);
    end;
    PRINT('//');
end;
end;

```

で、さらにSUMA(M, A, K, L, P);をarray AのK列とL列の積和(P=2のとき)及びK列の和(P=1のとき)とする関数指示子としTRANを省略すると同時にSS及Rの印刷手続をPR(F);とひとまとめにしてFのところに関数形を入れる様になす次のようになる。

```

begin integer I, J, M, N, P, K, L; real RES;
    array XA[1:100, 1:10], SS[1:10, 1:10];
    procedure PR(F, N);
        value N; integer N;
        procedure F;
            begin integer I, J;
                for I:=1 step 1 until N do
                    begin
                        for J:=1 step 1 until I-1 do
                            PRINT('17B');
                        for J:=I step 1 until N do
                            begin
                                F;
                                PRINT('R', RES);
                            end;
                    end;
            end;
        end;
    end;
end;

```

```

        PRINT('//');
    end;
end;

real procedure SUMA(M, A, K, L, P);
    value M, K, L, P;
    integer M, K, L, P; array A;
begin
    integer I, J; real S;
    S:=0;
    for I:=1 step 1 until M do
        S:=S+A[I, K]*(if P=1 then 1
            else A[I, L]);
    SUMA:=S;
end;

procedure SSF;
begin RES:=SUMA(M, XA, I, J, 2)
    -SUMA(M, XA, I, I, 1)
    *SUMA(M, XA, J, J, 1)/M;
    SS[I, J]:=RES;
end;

procedure RF;
begin RES:=SS[I, J]/SQRT(SS[I, I]
    *SS[J, J]);
    R[I, J]:=RES;
end

READ(M, N);
for I:=1 step 1 until M do

```

```

        for J:=1 step 1 until N do
            READ(XA[I, J]);
        PR(SSF, N);
        PR(RF, N);
    end;

```

となり, 更にこの全体をSSRというprocedureにしてプログラムの中
で使つてみると,

```

begin integer I, J, M, N;
    array XA[1:100, 1:10], SS, R[1:10, 1:10];
    procedure SSR(M, N, XA, SS, R);
        value M, N;
        integer M, N; array XA, SS, R;
        begin
            integer K, L, P; real RES;
            procedure PR(F, N);
                value N;
                integer N; procedure F;
                begin
                    integer I, J;
                    for I:=1 step 1 until N do
                        begin
                            for J:=1 step 1 until I-1 do
                                PRINT('17B');
                            for J:=I step 1 until N do
                                begin F;
                                    PRINT('R', RES);
                                end;
                        end;
                end;
            end;
        end;
    end;

```

```

        PRINT(' 2/' );
    end;
end;

real procedure SUMA(M, A, K, L, P);
    value M, K, L, P;
    integer M, K, L, P;    array A;
begin
    integer I, J;    real S;
    S:=0;
    for I:=1 step 1 until M do
        S:=S+A[I, K]*(if P=1
                        then 1 else A[L, L]);
    SUMA:=S;
end;

procedure SSF;
begin RES:=SUMA(M, XA, I, J, 2)
      -SUMA(M, XA, I, I, 1)
      +SUMA(M, XA, J, J, 1)
      /M;
    SS[I, I]:=RES;
end;

procedure RF;
begin
    RES:=SS[I, J]/SQRT(SS[L, I]*SS
                        [J, J]);
    R[I, J]:=RES;
end;

```

```

        PR(SSF, N);
        PR(RE, N);
    end;

    READ(M, N);
    for I:=1 step 1 until M do
        for J:=1 step 1 until N do
            READ(XA(I, J));
            SSR(M, N, XA, SS, R);
        end;
    end;

```

であり、これを I/O 命令の一部が異なり、関数指示子がなく、real と integer の混合演算を許さぬ 'H1' に直してみると、

- procedure PR 中の PRINT('17B'); を SPACEN(17); に、
PRINT('R', RES); を PRINT(9, RES); に PRINT('21');
を CRLF(2); に修正。

real procedure SUMA(M, A, K, L, P); を

- procedure SUMA(M, A, K, L, P, SUM); とし、specification
part に real SUM を追加し、本体の中の

SUMA:=S; を SUM:=S; に修正

- procedure SSF; を

```

procedure SSF;
begin real SUM1, SUM2, SUM3;
    SUMA(M, XA, I, J, 2, SUM1);
    SUMA(M, XA, I, I, 1, SUM2);
    SUMA(M, XA, J, J, 1, SUM3);
    RES:=SUM1-SUM2*SUM3/FLOAT(M);
    SS(I, J):=RES;
end;

```

に修正

- procedure RF中のsqrtをSQRTに修正ということになる。

またI/O命令のみ異なる'230'に直すには

- procedure PR中の

PRINT('17B');をSpace(17);に PRINT('R', RES);

をprintreal(RES,9);に, PRINT('2/');を, CRLF;

CRLF;に修正

- READ(M,N);を Readreal(M); Readreal(N);に修正して先頭を

```
begin integer M, N;  
  Readreal(M); Readreal(N);  
  begin  
    array XA[1:M, 1:N], SS, R[1:N, 1:N];  
    Readarray(XA);  
    begin integer I, J;  
      procedure SSR(M, N, XA, SS, R);  
        以下同じ  
        SSR(M, N, XA, SS, R);  
      end;  
    end;  
  end;
```

とすればよい。

13. アルゴリズムによる有用な諸手続

13. 1 フーリエ係数

$$a_k = \frac{1}{n} \sum_{i=0}^{2n-1} f\left(a + i \frac{b-a}{2n}\right) \cdot \cos\left(\frac{2\pi j}{b-a} \left(a + i \frac{b-a}{2n}\right)\right)$$

$$b_k = \frac{1}{n} \sum_{i=0}^{2n-1} f\left(a + i \frac{b-a}{2n}\right) \cdot \sin\left(\frac{2\pi j}{b-a} \left(a + i \frac{b-a}{2n}\right)\right)$$

であるから限界 A, B , 区割数 N , 関数形 FUNC , $\sin$ 又は $\cos$, SICO , 答 ANS をパラメタとして procedure を作つてみる。

procedure $\text{FOURIER}(A, B, N, \text{FUNC}, \text{SICO}, \text{ANS})$,

value A, B, N ;

real A, B ; integer N ;

real procedure FUNC, SICO ;

array ANS ;

begin

integer I, J ;

real $A1, A2, C, S$;

$A1 := (B-A) / 2/N$;

$A2 := 6.2831853/(B-A)$;

for $J := 0$ step 1 until $N-1$ do

begin $C := J * A2$; $S := 0$;

for $I := 0$ step 1 until $2 * N - 1$ do

$S := S + \text{FUNC}(A + I * A1) *$

$\text{SICO}(C + (A + I * A1))$;

$\text{ANS}[J] := S/N$;

end;

end;

としておいて関数形を何でも

```
real procedure P (A) ;  
  value A; .  
  real A;  
  begin P:=Aを用いた関数形 ;  
end;
```

と宣言しておいてQ1 からQ2まで20分割に対し a_n b_n を夫々 array
 A, B に格納するなら

```
FOURIER (Q1, Q2, 20, COS, A);  
FOURIER (Q1, Q2, 20, SIN, B);
```

とすればよい。

13. 2 最大, 最小, 絶対値の最大を求める手続

2つの値のうち大きい方を求める手続BIGGERを利用して考えられる
最小値をMAXに考えられる最大値をMINに入れておいて次々比較して大
きいときはMAXに代入するやり方で全部について行方方針をとる。

```
procedure EXTREME(N, A, MAX, MIN, AB);
```

```
  comment N: data size,
```

```
  A: data array,
```

```
  MAX: maximum value,
```

```
  MIN: minimum value,
```

```
  AB : absolutely maximum value;
```

```
  value N;
```

```
  real MAX, MIN, AB; integer N;
```

```
  array A;
```

```
  begin
```

```
    integer I;
```

```
    real procedure BIGGER(X, Y)
```

```

value X,Y;
real X,Y;
BIGGER := if x > Y then X else Y;
MAX := -10 50;
MIN := 10 50;
AB := 0;
for I := 1 step 1 until N do
begin
MAX = BIGGER(A[I], MAX);
MIN = BIGGER(-A[I], MIN);
AB = BIGGER(ABS(A[I]), AB);
end;
MIN = -MIN;
end;

```

13. 3 arcsine の手続

$$\sin^{-1} x = x + \frac{x^3}{6} + \frac{1}{2} \frac{3}{4} x^5 + \frac{1}{2} \frac{3}{4} \frac{5}{6} \frac{x^7}{7} + \frac{1}{2} \frac{3}{4} \frac{5}{6} \frac{7}{8} \frac{x^9}{9} + \dots$$

であるから x の 0.1 (0.1) 0.9 について精度 5×10^{-7} まで求めるには

$$N \text{ 項} = (N-1) \text{ 項} \times \frac{(2N-3)^2}{(2N-2)} \frac{x^2}{(2N-1)}$$

であることを利用すればよい。

```

begin real X,DIF;
real procedure AROSIN(X);
value X;
real X;
begin integer N; real DIF;

```

```

N:=1;
DIF:=ARCSIN:=X;
for N:=N+1 while ABS(DIF)
    ≥ 510-7 do
begin
    DIF:=DIF*((2*N-3)*X)
        ↑ 2/((2*N-2)*(2*N-1));
    ARCSIN:=ARCSIN+DIF;
end;
end;
PRINT('↑');
for X:= 0.1 step 0.1 until 0.91 do
    PRINT('2R/' ,X,ARCSIN(X));
end;

```

ここにXの終期値を0.91としたのは2進10進の換算で0.1ずつ増加するうちに0.90をわづかでも越して実行しないで終ることをみそれたものである。

13.4 DEGREE より RADIAN への変換

```

real procedure RADIAN(DEG,MIN,SEC);
value DEG,MIN,SEC;
integer DEG,MIN,SEC;
begin
    RADIAN:=(DEG+MIN/60+SEC/3600)
        *3.14159265/180;
end;

```

13.5 RADIAN より DEGREE への変換

```

procedure DEGREE(X,P);
  value X,P;
  real X; integer P;
begin
  integer DEG, MIN, SEC;
  real D,A,B,C;
  D := 180*X/3.14159265;
  DEG := FIX(D);
  A := D - DEG;
  B := A*60;
  MIN := FIX(B);
  C := B - MIN;
  SEC := FIX(C*60+0.5);
  PRINT('D',I'M',I'S', DEG, MIN, SEC);
  if P = 0 then PRINT('/');
end;

```

たとえばプログラム中 $A = 4D, 12M, 36S$; $B = 2D, 12M, 18S$; と A 及び B を換算して印刷しなければ

```

PRINT('A=');
DEGREE(A,1);
PRINT('B=');
DEGREE(B,0); となろう。

```

13.6 2次方程式の根を求める手続

$$AX^2 + BX + C = 0$$

の根は $A \neq 0$ として

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

であるから実部は RE ，虚部は IM と格納すると同時に

$A = \times \cdots \times \cdots B = \times \times \cdots \times \quad C = \times \times \cdots \times \quad \times 1 = \times \cdots \times \quad \times 2 = \times \cdots \times$

又は $A = \times \times \cdots \times \quad B = \times \cdots \times \quad C = \times \cdots \times \quad RE = \times \cdots \times \quad IM = \times \cdots \times$

と印刷させる手続を 'H 2' で作つてみる。

```

procedure QUADEQ (A,B,C,RE,IM);
  value A,B,C;
  real A,B,C,RE,IM
begin
  real S;
  Boolean CR;
  procedure PR(L,X,P);
    value X,P;
    real X; string L; integer P;
    begin
      PRINT('L R',X);
      if P=0 then PRINT('/');
    end;
  RE := -B/(2*A);
  S := RE↑2 - C/A;
  CR := S < 0;
  IM := Sqrt(Abs(S));
  PR('A=',A,1); PR('B=',B,1); PR('C=',C,1);
  if ¬ CR then PR('X1=',RE+IM,1);
    else PR('RE=',RE,1);

```

```

      if  $\neg$ CR then PR('X2=', RE-IM, 1)
      else PR('IM=', IM, 0);
end;

```

14 其他の特殊な取り扱い

14.1 アルファモードの取り扱い

ALGOL 60 にアルファモードに関する規定はないが、情報を文字として取り扱って A の内容 TANI という字を打ちたい場合がよくある。これに対して Burroughs B 5500 では type declaration として ALPHA という宣言子があり 6 桁 (? を除く) の文字を格納することができる。

OKITAC 5090H 用の NEW ALGOL-H 'H2' でも integer に対しデータタイプ上 =TANI, =TOKYO, ... などと前に = をつけた文字数字記号等 (上段で = を除いたものと空白を示す #) 7 桁まで読み込むか

```
A:=EQUIV(TANI);
```

```
B:=EQUIV(TOKYO);
```

などと代入して

```
PRINT('7A', A);
```

という命令で印刷させることができる。

今 TOKYO, KITAKYUSYU, YKOHAMA などの 15 大都市名と人口をその順に読み込んで人口順に名前と人口を印刷するプログラムを 'H2' で書いてみる。ただし名前は 7 文字をこえるものがある、ので、integer array としては 15 行 3 列のものをとり第 1 列と第 2 列に名前を第 3 列に人口を入れることにする。

データテーブルとしてはたとえば次のような形で作っておく。

=YOKOHAM, =A, 1364821,=TOKYO,=,10364321,
=NAGOYA,=, ...

(=の後はアルファモード扱いで,=,となつて後に何もないとブラン
クが入る)

```
begin integer array INA[1:11,1:3];
  integer I,J,N;
  READ(N);
  for I:=1 step 1 until N do
    for J:=1 step 1 until 3 do
      READ( INA[I,J] );
  for I:=1 step 1 until N do
  begin
    integer L ;
    for J:=I+1 step 1 until N do
    begin
      if INA[I,3] < INA[J,3]
      then
      begin
        real BIG;
        integer K;
        BIG:=INA[J,3];
        INA[J,3]:=INA[I,3];
        INA[I,3]:=BIG;
      end;
    end;
    PRINT('I 2B',I);
    for L:=1 step 1 until 2 do
      PRINT('7A', INA[I,L]);
```

```

      PRINT('2B I/' , INA(I,3)),
    end;
end;

```

14. 2. 倍精度演算

FORTRAN では DOUBLE PRECISION という宣言があるが ALGOL には規定がない。しかし Burroughs B 5506 では DOUBLE という宣言子が設けてあるし、また倍精度可変長語の計算機である FACOM 230-30, 230 では最初の begin の直後に

```
comment integer length:=14, reallength:=14;
```

というように長さを指定できる。

14. 3 特殊な条件付算術式及び条件付命令

コンソール上から適当な信号を入れることによつてチェック計算時には計算を簡略化して行うことがよくある。前置パラメタを読んでその内容によつて判断することになるが、'H2' では

条件付算術式に

```
if BE then AS else AE
```

の他に

```
on <unsigned integer> then SA else AE
```

条件付命令に

```
if BE then US;
```

```
if BE then US else S;
```

```
if BE then FS;
```

の他に

```
on <unisged integer> then US;
```

```

on < unsigned integer > then  $\overline{US}$  else  $\overline{S}$  ;
on < unsigned integer > then  $\overline{FS}$  ;

```

を追加してある。

何れもコンソール上の 1 ~ 10 のボタンに対応してこれが押してあつたら then 以下に行くようにさせてある。

15° プログラムエラーのチェック

ALGOL によつてプログラムを書けば機械がプログラムの文法的エラーをチェックしてわれわれにエラーメッセージを知らせてくる。しかし高価な機械を初等的なミスの検出に空費するのはもつたない話であるし、また文法的なエラーとは一見みなされないがプログラマの意図したものと違つた手順を指示した場合などもあるので機械にかける前に一応次のチェックポイントは見直しておいた方がよい。

計算機にプログラムをかけてみてどうも思つた答が出ないというのは次のどれかの場合であろう。

- A 計算機の故障の場合
- B 文法的ミスを機械が指摘して来た場合
- C 計算途中でストップした場合
- D 計算途中で同じ動作を無限にくり返し始めた場合
- E 答は出たが予想したものと違う場合

A の機械的ミスに対しては

- i 入力テープが引掛つた場合は最初から入れ直しを行なう。
- ii マグネティックテープにキズがあるか読取りヘッドにゴミのついた場合、テープをとり替へヘッドを清掃する。
- iii 電気回路の故障の場合は専門技術者にまかせる。

B の文法的ミスは機械の打つて来たエラー番号などからエラーの原因を調べるがこれを避けるには徹減にかける前に次の点についてチェックし

てみる方法がよい。

i 算術演算関係

1. 掛算の * がおちていないか？

特に $(A+B)C$ とか $4B$ とかに注意

2. 割算の分母に 0 が入るようになっていないか？

これはそのつもりでなくても何も代入や読込をしていない変数をいきなり使用すると 0 が入っているおそれがある。

3. 負または 0 の log をとっていないか？

4. 負の sqrt をとっていないか？

5. $A < 0$ B real の状態で $A \uparrow B$ を作っていないか？

6. real の数値で、終っているものや $10^{2.5}$ などを書いていないか？

7. 2 つ以上の算術演算子が並んでいないか？

$A \uparrow -5$ $3.0 / -4.8$ などに注意

ii 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外でそのブロック内と同じ意味に使用されていないか？

3. 同じブロック内で変数・名札・手続・関数指示子・スイッチ等の名前に同じものを使っていないか？

4. 変数はそれに代入命令かまたは読込命令によつて値を与えられる前に使用されていないか？

iii デリミタ

1. デリミタの綴りに誤りはないか特にタイプライタの上下段コードを誤つて小文字を大文字などにしていないか

2. begin, end, (), [], ' ' などのカッコが対応しているか？

IV 式

1. 条件付算術式，ブール式，指示式は if, then, else の3つを含んでいるか？

2. 条件付の式で then の次に if がくるときは if 以下の式を () でつつんであるか？

V 命令

1. 命令は else の直前を除きすべて ; で終っているか？

特に begin end でつつんだ複合命令の場合に end の後の ; を落していないか？

(ブロックの場合も同様)

2. 代入命令の左辺の変数の型は皆同じものか？

3. 条件付命令で then の後に if がくるときは if 以下の命令を begin end でつつんであるか？

4. for 命令の do の後のくり返すべき命令は最初の ; (begin がくるときは end の後の ;) までであるがそれでよいか？

5. for 命令の中で初めにもどつて初期値からやり直していることはないか？

6. for 命令のコントロール変数について指定されたとおり計算終了してから次の命令の中でこのコントロール変数の値を利用していないか？

7. 手続命令の実パラメタが未宣言でないか？

(形式的パラメタは手続宣言中の局所的なものに過ぎない)

VI コメント

1. comment は ; で終っているか？

2. end の後の ; を落したため次の命令が comment 扱いにされて読みとばされていないか？

○の計算途中でストップした場合は

i 標準関数が指定された範囲外の数値を扱っていないか？

ii 割算の分母に無限小または 0 が来っていないか？

Ⅲ 単純変数または array でそれが使用されるまでに値を与えてないために前に入つたわけの分らない情報を使っているのではないか？

Dの無限のくり返し動作を続けた場合は for命令からぬけ出せなくなつたか条件がいつまでも成立しないときで

i 毎回得られた答を格納し忘れていないか？

ii 毎回同じ初期値をセットしていないか？

iii 判定条件が間違っていないか？

iv 初期値が間違っていないか？

Eの答は出たが予期したものと違う場合は

i 無限小が出て来たらそれに対応する変数またはそれに用いた変数を始めて式の右边で使したときまでに代入命令にかつて与えられていないのではないか？

ii 全く予期しない値のときも i と同様

iii 全く予期しない値のとき array で次々と置き替えてゆく段階で置き替え前の値を他の変数に受けることを忘れていないか？

iv end の後の ; を忘れたためその次の命令が comment 扱いとなつて読み飛ばされたため全く違つた答になつたのではないか？

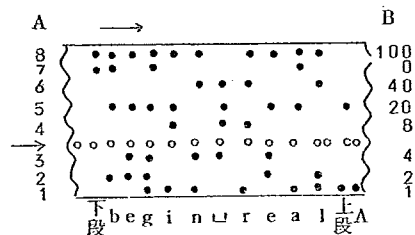
v for命令の外に書くべき READ命令をうつかり中に入れて読むべきデータが2回目以降なくなつたのではないか？

vi procedure 中で for 命令に用いる I, J 等を local に宣言しなかつたため本体の I, J と混同を起したのではないか？
以上の諸点についてプログラムをチェックしてから計算機にかけることが望ましい。

16. アルゴリズムの作り方

プログラムシートに書かれたアルゴリズムプログラムはタイプライターによつてテープに作られる。このテープ上の穿孔位置は機械によつてまちまちであるが此処では1例として OKITYPE R による8単位コード例をあげる。

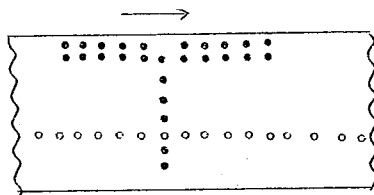
但しエラーチェックのため孔から読み取るときは



右側の B マークによつて読みたとえ一番左は $0+100=100$ で表の 100 から下段コードを読み、これから後上段コード（矢印の孔のみのもの）がくるまではすべて下段であることを了解し次の $2+20+0+100=122$ によつて 6 以下 begin スペース real と読み上段コードの次は $1+20=21$ で大文字の A と了解する。

またテープスプライサや補修紙を用いて字を訂正するとき n という字が必要ならば A コードによつて 13 68 と承知してこの位置にパンチすればよい。

土木研究所においては土研のテープであることを示すために begin の前に下段 5 復帰改行下段 5 のマークを入れる。



というパターンによつて土研のテープの先頭であることを容易に分らせるようにしている。

上 段

	B	A
1	1	1,7
2	2	2,7
3	3	1,2
4	4	3,7
5	5	1,3
6	6	2,3
7	7	1,2,3,7
8	8	4,7
9	9	1,4
=	10	2,4
10	11	1,2,4,7
+	20	5,7
A	21	1,5
B	22	2,5
C	23	1,2,5,7
D	24	3,5
E	25	1,3,5,7
F	26	2,3,5,7
G	27	1,2,3,5
H	28	4,5
I	29	1,4,5,7
.	30	2,4,5,7
)	31	1,2,4,5
-	40	6,7
J	41	1,6
K	42	2,6
L	43	1,2,6,7
M	44	3,6
N	45	1,3,6,7
O	46	2,3,6,7
P	47	1,2,3,6
Q	48	4,6
R	49	1,4,6,7
;	50	2,4,6,7
*	51	1,2,4,6
O	60	5,6
/	61	1,5,6,7
S	62	2,5,6,7
T	63	1,2,5,6
U	64	3,5,6,7
V	65	1,3,5,6
W	66	2,3,5,6
X	67	1,2,3,5,6,7
Y	68	4,5,6,7
Z	69	1,4,5,6
,	70	2,4,5,6
(	71	1,2,4,5,6,7
•SP	72	3,4,5,6
•CR	74	2,3,4,5,6,7
Tm	14	2,3,4,7

下 段

	B	A
△	101	1,8
▽	102	2,8
:	103	1,2,7,8
≡	104	3,8
≡	105	1,3,7,8
≡	106	2,3,7,8
□	107	1,2,3,8
⌈	108	4,8
✱	109	1,4,7,8
'	110	2,4,7,8
ˆ	111	1,2,4,8
↑	120	5,8
a	121	1,5,7,8
b	122	2,5,7,8
c	123	1,2,5,8
d	124	3,5,7,8
e	125	1,3,5,8
f	126	2,3,5,8
g	127	1,2,3,5,7,8
h	128	4,5,7,8
i	129	1,4,5,8
≡	130	2,4,5,8
≠	131	1,2,4,5,7,8
÷	140	6,8
j	141	1,6,7,8
k	142	2,6,7,8
l	143	1,2,6,8
m	144	3,6,7,8
n	145	1,3,6,8
o	146	2,3,6,8
p	147	1,2,3,6,7,8
q	148	4,6,7,8
r	149	1,4,6,8
v	150	2,4,6,8
Λ	151	1,2,4,6,7,8
∕	160	5,6,7,8
Σ	161	1,5,6,8
s	162	2,5,6,8
t	163	1,2,5,6,7,8
u	164	3,5,6,8
v	165	1,3,5,6,7,8
w	166	2,3,5,6,7,8
x	167	1,2,3,5,6,8
y	168	4,5,6,8
z	169	1,4,5,6,7,8
⌋	170	2,4,5,6,7,8
⌋	171	1,2,4,5,6,8
•\\	175	1,2,3,4,5,6,7,8

• 印は上下段共通

おわりに

ALGÖL は何回ものべたように問題を持つものが直接機械に話しかけたり、何人かで手分してプログラムを作ったり、他の人の書いた手順を容易に理解したりするための国際共通語である。機械語や中間語と人間の言葉とを一直線に並べてみれば ALGOL は非常に人間語に近いものと云うことができる。

今まで電子計算機の言葉を学ぶという段になると、どうしても機械の構造に対する知識が必要で、中間語の段階でもインデックスレジスタとかアキュムレータといった耳なれない言葉や機械がどのようにして演算を進めるかといったことに関する知識を要求された。しかし、モータを動かすにはどのようにスイッチを入れるかと云った知識以外にモータの構造原理に関する知識は一般に不必要であると同様に、ALGÖL では電子計算機の構造原理に関する知識は一般的にいつて不必要である。

われわれの言葉に近い表現で書いてやれば機械自らがその文法的なチェックを行い正しい命令を自分自身の言葉に翻訳編集して後それにしたがって計算を実行する。

ただしわれわれの言葉はある程度冗長度が大きいので `go to A ;` といつても `jump to A ;` といつても意味が通ずるが ALGÖL では一つの表現に統一されてしまうことはやむを得ないわけで、このためにも他の人の開発したプログラムを互いに利用しあうことが必要になつてくる。

このようにプログラミングを容易にし、協同開発を促進し、さらにおたがいの情報交換の道具として ALGÖL の重要性はますます増大するものと考えられるので、さらに次の文献などによつて日常の言葉としての ALGOL に親しまれることをおすすめしたい。

参 考 文 献

1. 森口繁一編：ALGOL入門，JUSE 出版
2. 高須達編：電子計算機のプログラミング，日刊工業新聞社
3. 富士通信機：FACOM - ALGOL への入門Ⅱ
4. 沖電気：OKI-PAL 説明書
5. 高千穂交易：INTRODUCTION TO ALGOL 60
6. Nauer: Revised Report on the Algorithmic Language
ALGOL 60
ACM vol. 6 No. 1 (1963)
7. 中村慶一：技術者のための統計解析
11章 アルゴリズムによるプログラミング 山海堂 (1965)
8. 加瀬滋男：コンピュータの常識 日刊工業新聞 (1963)
9. 岡崎・角山：わかりやすいプログラミング 産報 (1965)
10. Mcracken, D.D.: A Guide to ALGOL
Programming, Wiley (1962)
11. Dijkstra, E.W.: A Prime to ALGOL 60
Programming, Academic Press
12. Wooldridge & Ractliffe: An Introduction to ALGOL
Programming,
The English Univ. Press (1963)
13. Beumann et al: Introduction to ALGOL, Prentice-Hall
(1964)
14. 山内・森口一松：電子計算機のための数値計算法 I II III (1965)
15. 一松信：数値解析 至文堂 (1963)