CALCULATRICE PROGRAMMABLE

CASIO FX-702P

MODE D'EMPLOI

(1A)

Premièrement, nous désirons vous remercier d'avoir porté votre choix sur ce produit. Il s'agit d'une calculatrice programmable hautement performante dotée de microcircuits permettant d'effectuer des calculs répétitifs et complexes. La particularité essentielle de cet instrument est qu'il utilise le logiciel de programmation BASIC, qui est un langage conversationnel permettant de résoudre des problèmes. Son utilisation est facile, même pour un débutant. De plus, la programmation est très simple et fait appel à des instructions correspondant à une seule touche pour une manipulation plus rationnelle.

Les fonctions de cette calculatrice sont catégorisées comme suit:

1. Calculs manuels
2. Calculs programmés

Cette calculatrice peut exécuter des calculs programmés complexes, comme un ordinateur; elle est en outre prévue pour un usage facile en tant que calculatrice scientifique.

TABLE DES MATIERES

Avant l'emploi 3
Précautions à observer
Alimentation et remplacement des piles

Chapitre 1 Nomenclature et utilisation de chaque section 4

1-1 Nomenclature de chaque section 4
1-2 Interprétation de l'affichage 7
1-3 Réglage du contraste 7

Chapitre 2 Avant de commencer les calculs 8

2-1 Ordre de priorité d'exécution des calculs 8
2-2 Ordre de grandeur d'entrée/sortie et nombre de chiffres de calcul 8
2-3 Calculs de base 9
2-3-1 Symboles de calcul et instructions de fonctions 9
2-3-2 Rappel des résultats de calculs précédents 10
2-3-3 Messages d'erreur 10
2-4 Manipulation des touches 11
2-5 Extension de la mémoire 12
2-6 Coupure automatique de l'alimentation 13
2-7 Nombres excédentaires 13

Chapitre 3 Calculs manuels 14

3-1 Explication des calculs manuels 14
3-2 Opérations en calcul manuel 14

Ce mode d'emploi explique la manière d'utiliser cette calculatrice. Veuillez le lire attentivement avant de l'utiliser pour la première fois. Familiarisez-vous avec chacune de ses fonctions et traitez-la avec soin pour que vous puissiez en profiter longtemps.

* Faire spécialement attention à ne pas endommager l'appareil en le tordant ou en le faisant tomber. Par exemple, ne pas le mettre dans une poche révolver.

3-3 Exemples de calculs manuels **15**
3-3-1 Calculs de base 15
3-3-2 Calculs de fonctions 17
3-3-3 Calculs statistiques 21

Chapitre 4 Calculs programmés **25**

4-1 Description générale d'un programme **25**
4-2 Notions de base de programmation **28**
4-3 Ecriture et exécution de programmes **29**
4-4 Edition de programmes **32**
4-5 Ordres de programmation **39**
4-5-1 Saut et boucle 39
4-5-2 Rangées 53
4-5-3 Ordres d'entrée/sortie 57
4-5-4 Fonctions de caractères 63
4-5-5 Sous-programmes 64
4-5-6 Fonctions générales 68
4-5-7 Traitement statistique 69
4-5-8 Mot de passe 74
4-5-9 Caractéristiques optionnelles 75

Liste des messages d'erreur **79**

Liste des commandes de programme **80**

Fiche technique **83**

Avant l'emploi

Cette calculatrice est le fruit de notre technologie électronique la plus poussée. Lors de sa fabrication elle a été soumise à un contrôle de qualité très strict. De sorte à assurer sa durabilité, veuillez observer les précautions suivantes.

■ Précautions à observer

- Cette calculatrice est faite de composants électroniques de précision. Ne tentez jamais de la démonter. Evitez de la laisser tomber et de la lancer. De la soumettez pas à de fortes variations de température. Ne la rangez pas dans un endroit très chaud, humide ou poussiéreux. A très basse température, la réponse de l'affichage devient plus lente et il pourrait même disparaître. L'affichage normal réapparaîtra dès que la température sera à nouveau normale.
- N'utilisez aucun autre accessoire que ceux prévus en option.
- Lorsque la calculatrice effectue des calculs, le signe "—" est affiché. Durant ce laps de temps, les touches sont inopérantes, sauf pour une section. Par conséquent, observez toujours l'affichage et manipulez les touches avec précautions.
- Concernant les piles: même si vous n'utilisez pas fréquemment la calculatrice, veuillez changer les piles au moins tous les deux ans. Lorsque les piles sont plates, elles ont tendance à couler, ce qui pourrait détériorer l'instrument. En outre, ne laissez jamais des piles usées dans la calculatrice.
- Pour nettoyer la calculatrice, essuyez-la avec un chiffon sec ou imbibé de détergent doux. N'utilisez jamais de dilutif à peinture ou d'essence.
- En cas de défaillance, prenez contact avec le magasin ou vous avez acheté la calculatrice, ou avec l'agence la plus proche.
- Avant de demander une intervention, lisez une nouvelle fois le mode d'emploi, vérifiez l'alimentation, le programme et les erreurs de manipulation.

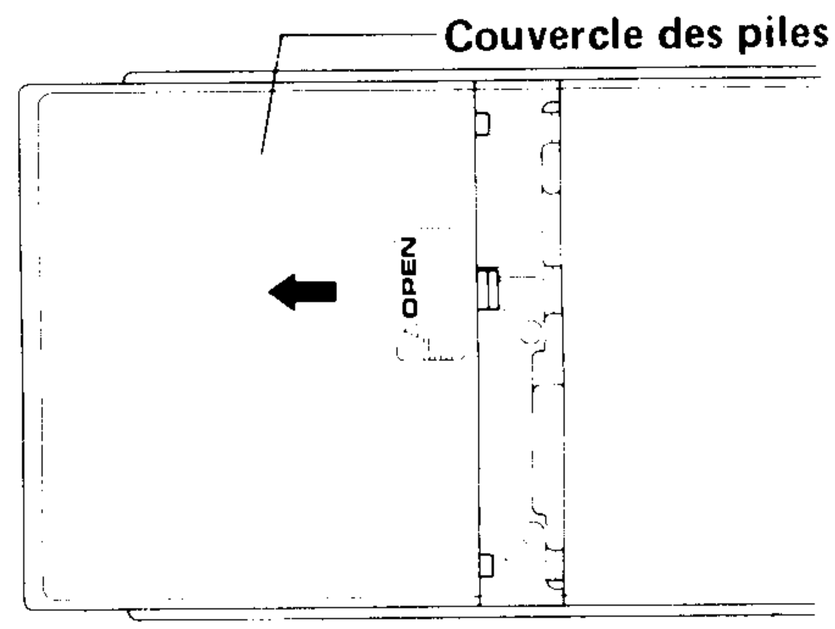
■ Alimentation et remplacement des piles

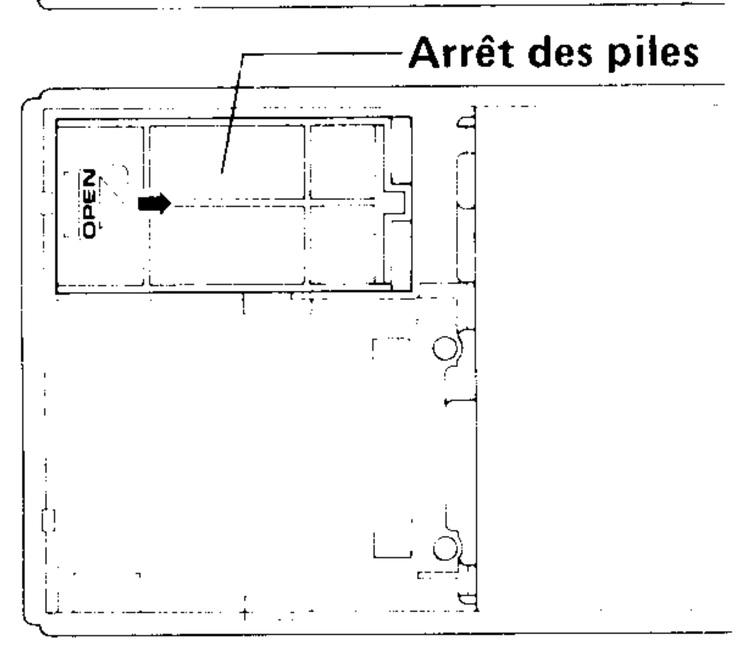
Cette calculatrice est alimentée par deux piles au lithium (CR2032).
Lorsque l'affichage est sombre bien que le réglage de contraste soit tourné à fond, les piles sont plates. Il s'agit dans ce cas de les remplacer de la manière indiquée ci-dessous. Lorsque les piles sont faibles, la calculatrice pourrait faire des erreurs.
Même si la calculatrice fonctionne normalement, assurez-vous de remplacer les piles au moins tous les 2 ans.

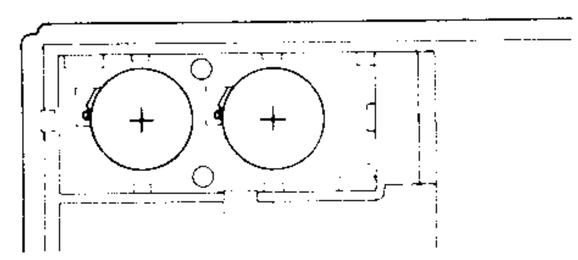
- **Comment remplacer les piles**

(1) Après avoir mis l'interrupteur général sur OFF, faites coulisser le couvercle dans le sens de la flèche et enlevez-le.
(2) Poussez l'arrêt des piles dans le sens de la flèche et enlevez-le.
(3) Otez les deux piles plates. Pour ce faire, frappez doucement l'instrument avec son compartiment dirigé vers le bas.
(4) De la poussière ou des dépôts blancs sur la face des piles peuvent causer un mauvais contact. Essuyez-les bien avec un chiffon sec. Posez les piles neuves avec leur côté ⊕ dirigé vers le haut.
(5) Appuyez les piles au moyen de l'arrêt et refermez le couvercle des piles.

* Tenir les piles hors de portée des enfants. Si on avalait une pile, consulter un médecin immédiatement.







Attention

Après avoir remplacé les piles, assurez-vous d'effectuer les opérations suivantes.
Mettez l'interrupteur général sur ON.

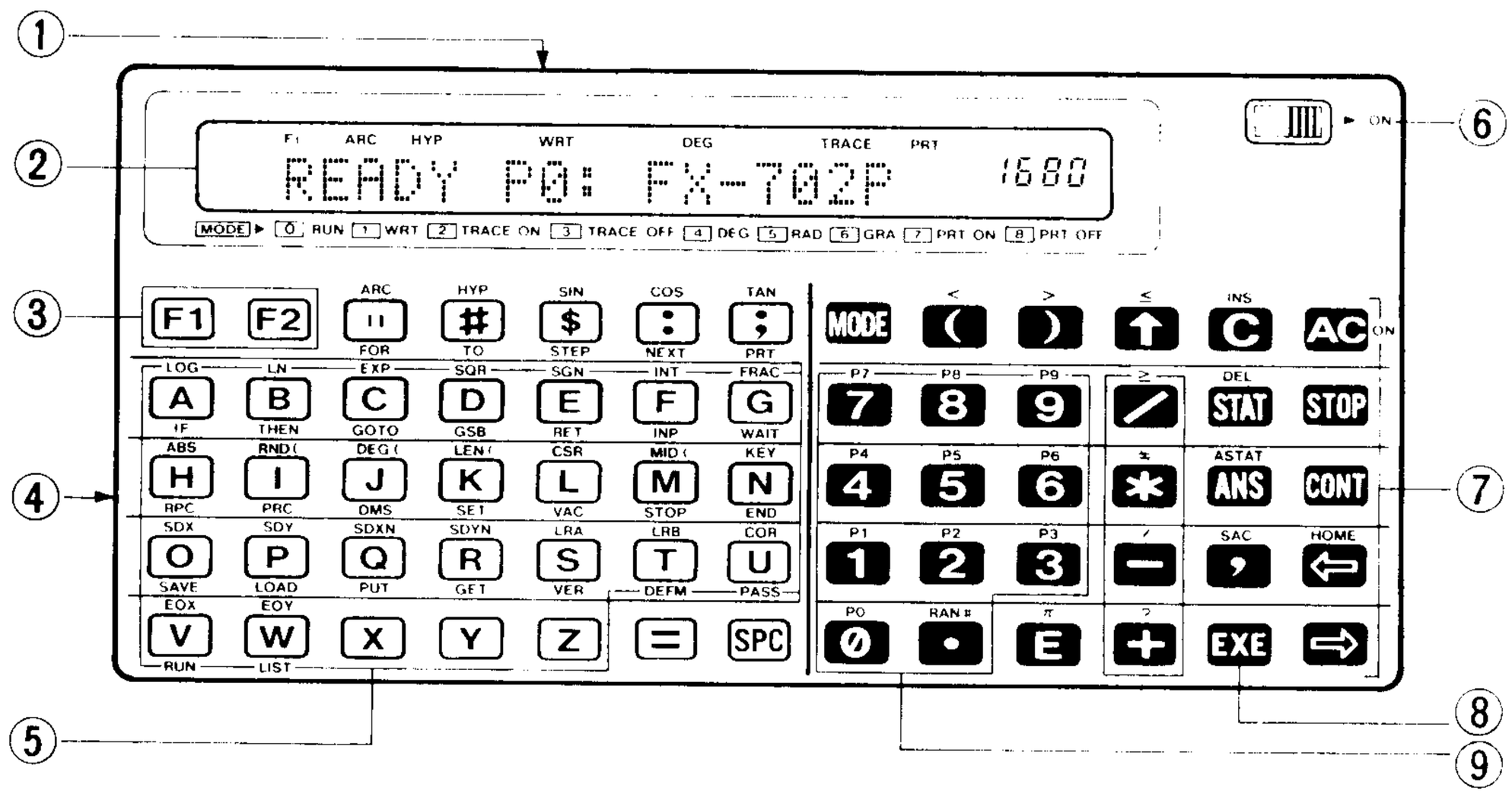
MODE 1 **CLR ALL** EXE

MODE 0 F1 SAC F2 T DEFM 0 EXE **VAC** EXE

ROM 0 EXE AC

Chapitre 1

Nomenclature et utilisation de chaque section



① Réglage de contraste de l'affichage
② Fenêtre d'affichage
③ Touches de fonctions
④ Prise d'adaptateur
⑤ Touches alphabétiques
⑥ Interrupteur général
⑦ Clavier
⑧ Touche d'exécution
⑨ Touches numériques et de virgule décimale

1-1 Nomenclature de chaque section

Chaque touche effectue trois opérations distinctes.
Pressez directement la touche pour obtenir la fonction imprimée sur sa face. Pressez [F1] puis la touche pour obtenir la fonction imprimée au-dessous, et pressez [F2] puis la touche pour la fonction imprimée au-dessous.

Exemple

LOG ..... Mode 1ère fonction
[A] ..... Mode direct
IF ..... Mode 2è fonction

[F1] Touche de 1ère fonction

La pression de cette touche fixe le mode de 1ère fonction ("F1" est alors affiché). La 1ère fonction imprimée au-dessus de chaque touche est alors accessible.

[F2] Touche de 2è fonction

La pression de cette touche fixe le mode de 2è fonction ("F2" est alors affiché). La 2è fonction imprimée au-dessous de chaque touche est alors disponible.

[MODE] Touche de mode

Le fonctionnement de la calculatrice et l'unité de mesure angulaire sont fixés en pressant la touche [MODE] et les touches numériques [0] à [8].

[MODE] [0] .... "RUN est affiché. Des calculs manuels programmés peuvent être exécutés.

[MODE] [1] .... "WRT" est affiché. L'écriture, la vérification et l'édition des programmes peuvent être exécutés.

[MODE] [2] .... "TRACE" est affiché. L'exécution pas à pas est possible. (Voir page 38 pour plus de détails.)

[MODE] [3] .... Effacement du "TRACE" affiché. Le mode d'éxécution pas à pas est annulé.

[MODE] [4] .... "DEG" est affiché. Les mesures angulaires seront données en "degrés".

[MODE] [5] .... "RAD" est affiché. Les mesures angulaires seront données en "radians".

[MODE] [6] .... "GRA" est affiché. Les mesures angulaires seront données en "grades".

[MODE] [7] .... "PRT" est affiché. Si une imprimante est raccordée, la sortie sera imprimée (voir page 78 pour plus de détails).

[MODE] [8] .... Lorsque "PRT" est affiché, il est effacé et la sortie d'impression est annulée.

[AC] Touche d'effacement général

- Lors de l'introduction de données, l'affichage est effacé et les caractères rentrés jusqu'à ce point seront tous annulés.
- Si elle est pressée au cours du déroulement d'un programme, celui-ci s'arrête et la calculatrice est remise à l'état "prêt pour entrée".
- Un message d'erreur est annulé par la pression de cette touche.
- Lorsque la coupure automatique d'alimentation (dispositif économisant les piles, décrit en page 13) est en fonction et que l'affichage s'éteint, presser cette touche pour rétablir l'alimentation.

[INS C] Touche d'effacement (touche de suppression du caractère entré)/touche d'insertion

- Lors de l'introduction de données, le caractère situé juste devant le curseur est annulé. Ceux situés à droite du curseur se déplacent d'une position à gauche.
- En mode 1ère fonction, cette touche réalise l'insertion. Lorsqu'elle est pressée, le caractère situé au-dessus du curseur et ceux à sa droite sont décalés d'un rang à droite pour permettre l'insertion d'un nouveau caractère.

[STOP] Touche d'arrêt

Au cours du déroulement d'un programme, sa pression affiche "STOP". L'exécution du programme s'arrête en fin de ligne; si elle est pressée en exécution pas à pas, l'affichage montre le numéro de programme, le numéro de ligne et le contenu du programme.

[CONT] Touche de reprise

Si elle est pressée après arrêt du programme par la touche [STOP], ou par une instruction d'arrêt, le déroulement reprend à la phrase suivante. Lors d'une exécution pas à pas, presser cette touche pour passer à la phrase suivante.

[DEL STAT] Touche statistique/suppression

- Presser cette touche pour rentrer des données en calcul statistique.
  Exemple: Variance ..... x [STAT]. Calcul de régression ..... x, y [STAT].
- En mode 1ère fonction, presser cette touche pour annuler une donnée lors d'un calcul statistique.

[ASTAT ANS] Touche de réponse/touche de réponse statistique

- En calcul manuel ou programmé, cette touche rappelle le résultat du calcul précédent.
- En mode 1ère fonction, presser cette touche pour afficher le résultat d'un calcul statistique (nombre de données, somme des x, somme des y, somme au carré des x, somme au carré des y, et produit de $x \cdot y$).

[HOME ⇦] Touche de curseur/touche de retour à l'origine

- Presser cette touche une fois pour déplacer le curseur d'un rang à gauche.
- En mode 1ère fonction, elle ramène le curseur à l'origine (sous le premier caractère entré).

[⇨] Touche de curseur

Presser cette touche une fois pour avancer le curseur d'un rang à droite.

[EXE] Touche d'exécution

- Presser cette touche au lieu de "=" pour obtenir le résultat d'un calcul manuel.
- En mode "WRT", presser cette touche pour l'écriture du programme ligne par ligne. Toute écriture est impossible si cette touche n'est pas pressée.
- En mode "RUN", presser cette touche pour effectuér une entrée au cours du déroulement du programme.

Touche de virgule/touche d'effacement mémoire pour statistique

- Presser pour écrire une virgule.
- En mode 1ère fonction, presser pour annuler la mémoire statistique.

Touches de parenthèses/touches de comparaison

- Presser ces touches aux points désirés pour effectuer des calculs de parenthèses.
- En mode 1ère fonction, presser ces touches pour établir une comparaison.

Touches d'opérations/touches de comparaison/touches de caractères

- Symboles Pour addition, soustraction, multiplication et division. Presser pour obtenir les fonctions désirées.
  Presser pour élever des nombres à une puissance.
- En mode 1ère fonction, presser pour établir une comparaison. Presser pour obtenir des résultats factoriels. Presser pour écrire un point d'interrogation.

Touches numériques/touches de numéro de programme

- Presser pour entrer les nombres d'un calcul. Presser pour placer la virgule décimale.
- En mode 1ère fonction, ces touches font usage d'indicateurs de numéro de programme. A l'écriture, le programme peut être démarré.

Touches alphabétiques/touches d'instructions

- Lors de l'écriture d'un programme ou en désignant des instructions d'écriture ou de fonction, la pression de ces touches commande l'affichage de lettres alphabétique.
- En mode 1ère fonction, des instructions de fonction sont affichées.
- En mode 2è fonction, des instructions de programme sont affichées.

Touche égal

Presser cette touche pour inscrire un symbole de substitution ou égal.

Touche d'espacement

Presser pour créer un espace entre caractères lors de l'introduction.

Touche de chaîne/touche d'instruction

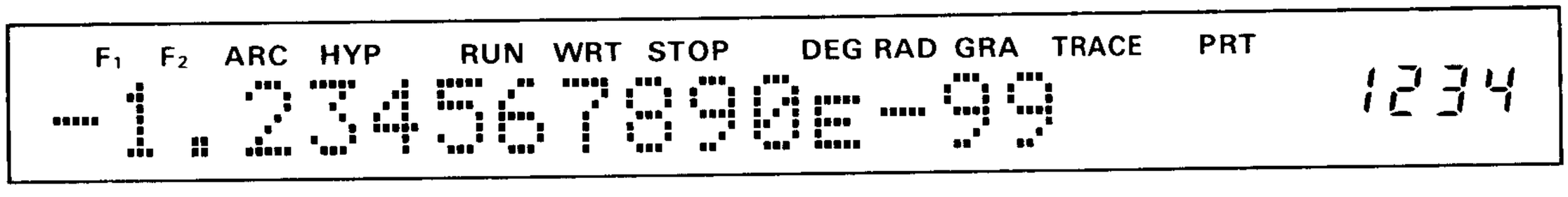
- Presser cette touche au début et à la fin d'une chaîne de caractères lors de l'introduction ou de l'affichage de constantes formées de caractères.
- En mode 1ère fonction, des instructions de fonction obtenues d'une touche sont affichées.
- En mode 2è fonction, des instructions de programme obtenues d'une touche sont affichées.

Touches de caractères/touches d'instructions obtenues d'une touche

- Presser pour utiliser les symboles figurant sur les touches.
- En mode 1ère fonction, des instructions de fonction sont affichées.
- En mode 2è fonction, des instructions de programme sont affichées.

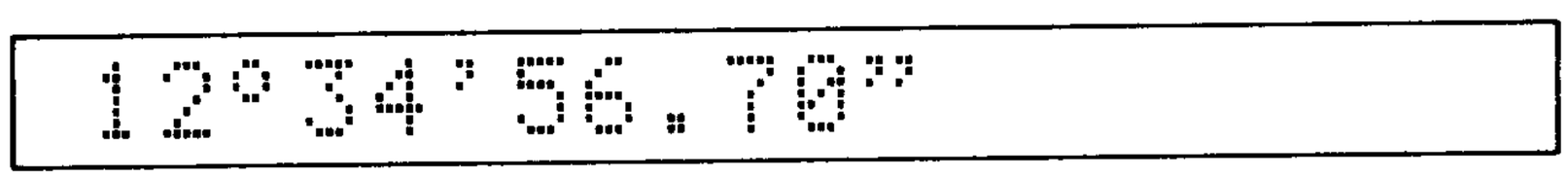
1-2 Interprétation de l'affichage

Affichage matriciel à points

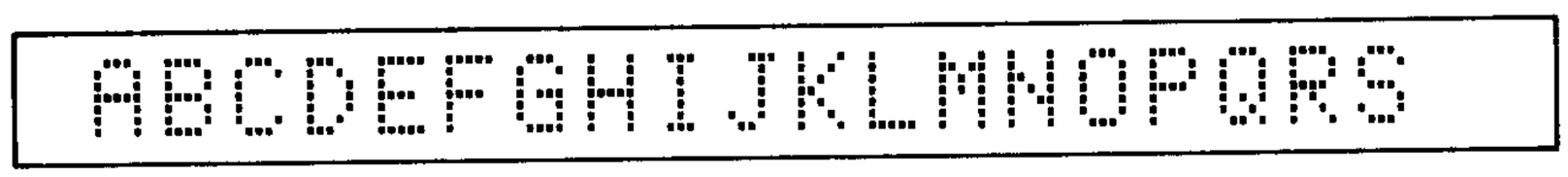


Affiche les valeurs et les résultats de calculs. Chaque caractère est composé de points sur 5 rangées de largeur fois 7 rangées de hauteur. 20 chiffres ou caractères au maximum peuvent être affichés. Le zéro apparaît sous la forme Ø. Si une formule ou une phrase contient plus de 20 caractères, les nombres ou les lettres disparaissent à gauche. 62 caractères au maximum peuvent être entrés. Les 4 positions se trouvant à droite de l'affichage montrent les pas de programme encore disponibles. Au cours d'un calcul, le signe moins est affiché à la droite de ces 4 positions, à l'extrême droite de l'affichage. Y apparaissent également les unités de mesure angulaires, telles que "DEG", "RAD", et "GRA", ou "F1, "F2" (quand la touche [F1] ou [F2] a été pressée), ainsi que "ARC", "HYP" (quand la touche [F1] [ARC] ou [F1] [HYP] a été pressée), "RUN", (en mode RUN), "WRT" (en mode WRT), "TRACE" (en mode TRACE), "PRT" (en mode PRT), etc. Ces symboles sont affichés selon les circonstances. Les chiffres hexadécimaux, lettres alphabétiques et symboles sont affichés comme suit.

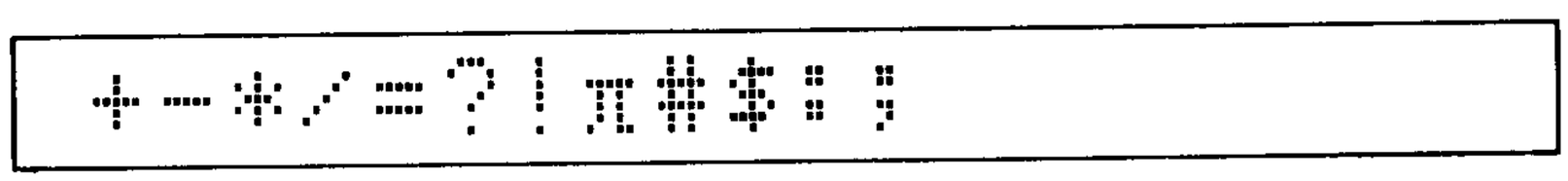
- **Exemple d'affichage sexagésimal**



- **Exemple d'affichage alphabétique**

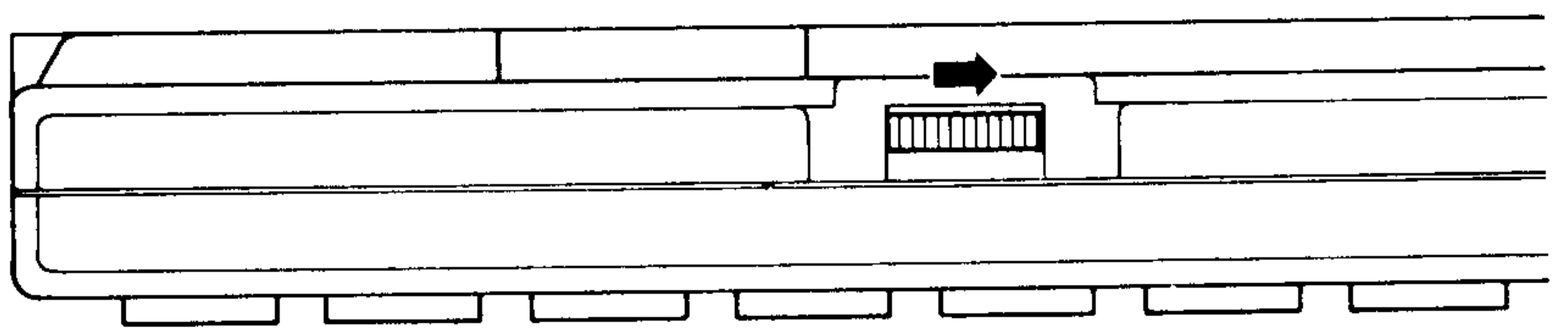


- **Exemple d'affichage de symboles**



1-3 Réglage du contraste

Le réglage du contraste de l'affichage s'effectue au moyen d'une molette se trouvant au sommet de la calculatrice.



Tourner la molette dans le sens de la flèche pour augmenter le contraste, et dans l'autre sens pour le réduire. Ce réglage permet d'établir un compromis entre l'économie des piles et les préférences individuelles.

Chapitre 2

Avant de commencer les calculs

Les calculs manuels et programmés sont exécutés en mode "RUN". (Pour obtenir RUN: [MODE] [0] .) Les unités de mesure angulaire, "DEG", "RAD" et "GRA" peuvent être ignorées si le calcul envisagé ne met pas d'angles en jeu.

2-1 Ordre de priorité d'exécution des calculs (ordre algébrique)

- La calculatrice détermine automatiquement l'ordre d'exécution des calculs et opère selon cet ordre. L'ordre d'exécution est le suivant:

① Fonctions (sin, cos, tan, etc.)
② Puissances, factorielles
③ x, ÷ (*, /)
④ +, −

Lorsque l'ordre de priorité est identique, le calcul s'effectue du début (de la gauche). Si des parenthèses sont utilisées, elles sont résolues en premier.

2-2 Ordre de grandeur d'entrée/sortie et nombre de chiffres de calcul

L'ordre de grandeur d'entrée/sortie de cette calculatrice est de 10 rangs de mantisse et 2 rangs d'exposant. La gamme de calcul s'étend de 1×10^{-99} à $\pm 9{,}999999999 \times 10^{+99}$.
Le nombre de chiffres d'entrée/sortie est de 10 rangs de mantisse et de 2 rangs d'exposant. Toutefois, le nombre de rangs de calcul et de stockage interne en mémoire est de 12 pour la mantisse et de 2 pour l'exposant. Lorsqu'une entrée ou valeur de résultat est supérieure à 10 rangs, jusqu'à 12 rangs sont écrits. Tout ce qui est plus grand sera ignoré. La sortie comporte 10 rangs de mantisse.

Exemple: 1.23456789123×100= | 123.4567891 |

Lorsque le résultat du calcul est supérieur à 10^{10} ou inférieur à 10^{-3}, il est automatiquement affiché en notation exponentielle.

Exemple: 1234567890×10= | 1.23456789E 10 |

(L'affichage de l'exposant vient après la mantisse et le signe exponentiel "E").

Exemple: 1.234÷10000= | 1.234E-04 |

2-3 Calculs de base

2-3-1 Symboles de calcul et instructions de fonctions

Cette calculatrice utilise les symboles BASIC. + et − sont identiques aux symboles ordinaires utilisés pour l'addition et la soustration, mais ∗ et / sont adoptés pour la multiplication et la division.
Par exemple:

2+3−4×5÷6

Est exprimé comme

2+3−4∗5/6

Cette calculatrice comprend également les fonctions suivantes

Nom de fonction		Forme	
Fonction trigonométrique	$\sin x$	SINx	[F1 SIN $]
	$\cos x$	COSx	[F1 COS :]
	$\tan x$	TANx	[F1 TAN :]
Fonction trigonométrique inverse	$\sin^{-1} x$	ASNx	[F1 ARC SIN $]
	$\cos^{-1} x$	ACSx	[F1 ARC COS :]
	$\tan^{-1} x$	ATNx	[F1 ARC TAN :]
Fonction hyperbolique	$\sinh x$	HSNx	[F1 HYP SIN $]
	$\cosh x$	HCSx	[F1 HYP COS :]
	$\tanh x$	HTNx	[F1 HYP TAN :]
Fonction hyperbolique inverse	$\sinh^{-1} x$	AHSx	[F1 ARC HYP SIN $]
	$\cosh^{-1} x$	AHCx	[F1 ARC HYP COS :]
	$\tanh^{-1} x$	AHTx	[F1 ARC HYP TAN :]
Racine carrée	$\sqrt{x}$	SQRx	[F1 SQR D]
Fonction exponentielle	e^x	EXPx	[F1 EXP C]
Logarithme naturel	$\ln x$	LNx	[F1 LN B]
Logarithme commun	$\log x$	LOGx	[F1 LOG A]
Factorielle	$x!$	$x!$	[F1 .]
Changement à entier	INTx	INTx	[F1 INT F]
Annulation de partie entière	FRACx	FRACx	[F1 FRAC G]
Changement à valeur absolue	$\lvert x \rvert$	ABSx	[F1 ABS H]
Changement à symbole	Nombre positif → 1 0 → 0 Nombre négatif → −1	SGNx	[F1 SGN E]
Degrés minutes secondes	(sexagésimal → décimal)	DEG (x, y, z)	[F1 DEG(J]
Degrés minutes secondes	(décimal → sexagésimal)	DMSx	[F2 J DMS]
Conversion de coordonnées	(rectangulaire → polaire)	RPCx, y	[F2 H RPC]
Conversion de coordonnées	(polaire → rectangulaire)	PRCx, y	[F2 I PRC]

* Les coordonnées x, y et r, θ peuvent être substituées aux variables X, Y.

Arrondissement	(10^y de x est arrondi)	RND (x, y)	[F1 RND(1]
Nombres aléatoires		RAN #	[F1 RAN# .]
Calculs statistiques			
Nombre de données n		CNT	—
Ecart standard de x $x\sigma_{n-1}$		SDX	[F1 SDX O]
Ecart standard de y $y\sigma_{n-1}$		SDY	[F1 SDY P]
Ecart standard de x $x\sigma_n$		SDXN	[F1 SDXN Q]
Ecart standard de y $y\sigma_n$		SDYN	[F1 SDYN R]
Moyenne des x $\overline{x}$		MX	—
Moyenne des y $\overline{y}$		MY	—
Somme des x Σx		SX	—

Somme des y	Σy	SY	–
Somme des x au carré	Σx^2	SX2	–
Somme des y au carré	Σy^2	SY2	–
Somme des produits de données	Σxy	SXY	–
Terme constant	A	LRA	[F1 LRA/S]
Coefficient de régression	B	LRB	[F1 LRB/T]
Coefficient de corrélation	r	COR	[F1 COR/U]
Valeur estimée de x	$\hat{x}$	EOXy	[F1 EOX/V]
Valeur estimée de y	$\hat{y}$	EOYx	[F1 EOY/W]

2-3-2 Rappel des résultats de calculs précédents

Le résultat d'un calcul manuel ou programmé est conservé jusqu'à ce que le prochain calcul ait été exécuté. Ce résultat peut être affiché en pressant la touche [ANS].

Exemple: 741+852=1593
2431−1593=838

Opération:

[7][4][1][+][8][5][2]	741+852
[EXE]	1593
[2][4][3][1][−][ANS]	2431− 1593
[EXE]	838

De plus, le nombre affiché en résultat du calcul peut être utilisé tel quel dans le prochain calcul.

Exemple: (suite à ci-dessus)
838×2=1676

Opération:

[*][2]	838*2
[EXE]	1676

2-3-3 Messages d'erreur

Si la formule ou la phrase de substitution n'est pas rédigée correctement conformément aux règles du langage BASIC, ou si la gamme de calcul est dépassée, une erreur se produit au cours du déroulement du programme et un message d'erreur sera affiché. En calcul manuel, les messages d'erreur suivants sont affichés:

ERR−2	(Erreur de structure de la phrase)
ERR−3	(Erreur mathémathique)

Les messages d'erreur suivants sont affichés en calcul programmé:

ERR−2 IN PØ−1Ø

Zone programme Numéro de ligne

(Une erreur de structure de phrase s'est glissée dans la ligne 10 de la zone programme PØ)

ERR−3 IN P2−2Ø

(Une erreur mathématique s'est glissée dans la ligne 20 de la zone programme P2)

Quand à la signification détaillée des messages d'erreur, veuillez vous reporter à la page 79 pour la liste complète.

* Lorsque la gamme de calcul est dépassée (±9,999999999E+99), l'état de débordement est réalisé et un message d'erreur est affiché. A moins de 1,0E−99, un état de sous-débordement se produit et le résultat du calcul sera 0.

2-4 Manipulation des touches

Premièrement, mettre la calculatrice sous tension.
"READY P0" est alors affiché et la calculatrice est prête pour l'introduction.

1. Entrée par touches

- **Entrée alphabétique**

Exemple: entrer ABC

Opération: [A][B][C] — ABC

Exemple: Entrer SIN

Opération: [S][I][N] (ou [F1] [SIN/$]) — SIN

* On peut utiliser soit une instruction alphabétique soit une instruction d'une touche.

- **Entrée numérique**

Exemple: Entrer 123

Opération: [1][2][3] — 123

Exemple: Entrer 96,3

Opération: [9][6][·][3] — 96.3

- **Entrée de symboles**

Exemple: Entrer $#?

Opération: [$][#][F1][?] — $ # ?

- **Entrée de nombres avec exposant**

Exemple: Entrer $7{,}896 \times 10^{15}$

Opération: [7][·][8][9][6][π/E][1][5] — 7.896E15

↑ Symbole exponentiel

Exemple: Entrer $-2{,}369 \times 10^{-45}$

Opération: [−][2][·][3][6][9][π/E][−][4][5] — -2.369E-45

2. Correction du contenu d'une entrée (correction, suppression, insertion)

- **Correction**

Amener le curseur à l'endroit où la correction doit être faite (utiliser les touches [⇦] [⇨]). Presser la touche correspondant à la lettre, le chiffre ou le symbole désiré.

Exemple: Remplacer A$ par B$ — A$_

Opération: Déplacer le curseur de 2 rangs à gauche.
[⇦][⇦] — A$
Presser sur la touche [B]. — B$

Exemple: Replacer "LIST" par "RUN" — LIST_

Opération: Déplacer le curseur de 4 rangs à gauche.
[⇦][⇦][⇦][⇦] — LIST
Presser [R][U][N] ou [F2] [V/RUN]. — RUN _

* Dans l'exemple ci-dessus, pour amener le curseur sous la première lettre, presser [F1] [HOME/⇦].

- **Suppression**

Amener le curseur à un rang à droite du caractère à supprimer, puis presser la touche **C**. A chaque pression de cette touche, le caractère situé à gauche du curseur disparaît.

Exemple: Supprimer le "I" de "SIIN"

Opération: Déplacer le curseur d'un rang à gauche.
(Utiliser ⇦)
Presser la touche **C**.

```
SIIN_
SIIN
SIN
```

Exemple: Supprimer "X" de "INP X, Y"

Opération: Déplacer le curseur d'un rang à gauche.
(Utiliser ⇦)
Presser **C** **C**. Les deux caractères seront suprimés.

```
INP X,Y_
INP X,Y
INP Y
```

Exemple: Supprimer ",B" de "PRT A, B"

Opération: Presser **C** **C**. Les deux caractères seront supprimés.

```
PRT A,B_
PRT A_
```

- **Insertion**

Déplacer le curseur à la droite du point où le nouveau caractère doit être inséré. Presser F1 INS/C. Chaque pression de ces touches crée un espace. Presser ensuite la touche désirée pour y placer l'insertion.

Exemple: Insérer "$" en "T = A$" pour former "T$ = A$"

Opération: Déplacer le curseur de 3 rangs à gauche.
(Utiliser ⇦⇦⇦)
Presser F1 INS/C. Un espace est créé au-dessus du curseur.
Presser la touche $.

```
T=A$_
T=A$
T_=A$
T$=A$
```

Exemple: Replacer "PRT X" par "PRT SIN X"

Opération: Déplacer le curseur d'un rang à gauche.
(Utiliser ⇦)
Créer 3 espaces. (utiliser F1 INS/C F1 INS/C F1 INS/C)
Presser S I N.

```
PRT X_
PRT X
PRT _ X
PRT SINX
```

2-5 Extension de la mémoire

Normalement, la calculatrice dispose de 26 mémoires (variables), et simultanément de 1680 pas. Il est possible d'étendre le nombre de mémoires à 226 au maximum. Ceci s'accomplit en convertissant 10 unités de mémoires correspondant à 8 pas chacune.

Nombre de mémoires	Nombre de pas de programme
26 (A ~ Z)	1680 pas
36 (A ~ Z, A0 ~ A9)	1600 pas
46 (A ~ Z, A0 ~ A9, B0 ~ B9)	1520 pas
56 (A ~ Z, A0 ~ A9, B0 ~ B9, C0 ~ C9)	1440 pas
⋮	⋮
226 (A ~ Z, A0 ~ A9, ... , T0 ~ T9)	80 pas

L'instruction DEFM est utilisée pour étendre la mémoire. L'extension est désignée par tranche de 10 unités, qui forment un groupe. Pour une extension de 10, définir "1"; pour une extension de 50, définir "5".

Exemple: Etendre la mémoire à 56, une augmentation de 30.

Opération: Effectué en mode RUN ([MODE] [0]) ou WRT ([MODE] [1]).

DEFM 3 [EXE] | VAR: 56 PRG: 1440 |

Nombre de mémoires — Nombre de pas

* DEFM peut être obtenu en pressant [D] [E] [F] [M] ou [F2] [T/DEFM] .

Exemple: Etendre la mémoire à 166, augmentation de 140.

Opération: **DEFM 14** [EXE] | VAR: 166 PRG: 560 |

* Si un nombre important de programmes est déjà utilisé, un message d'erreur est produit si la zone libre est insuffisante, afin de protéger les programmes existants.

| ERR-1 | ..... indique que le nombre de pas est insuffisant.

* Etant donné que la variable de caractère exclusif ($) est une mémoire spéciale, elle n'est pas inclue dans la capacité de mémoire allouable.

2-6 Coupure automatique de l'alimentation

De sorte à économiser de l'énergie lorsque la calculatrice est laissée sous tension, l'alimentation est automatiquement coupée au bout de 8 minutes (sauf en cours de calcul programmé). Pour réutiliser la calculatrice, soit réenclencher l'interrupteur général, soit presser la touche [AC] .

* La mémoire et les programmes contenus ne sont pas effacés lorsque l'alimentation est coupée. Par contre, la désignation de mesure angulaire ou de mode ("WRT", "TRACE", "PRT", etc.) sera annulée.

2-7 Nombres excédentaires

A l'affichage des résultats de calculs manuels ou programmés, le 11è rang peut être arrondi ou supprimé. Cette instruction est définie par "SET".

SET { 0 / 5 } ("0" pour suppression / "5" pour arrondissement)

Exemple: **11÷3=3.6666.....**

Opération: Pour supprimer [S] [E] [T] [0] [EXE]

[1] [1] [/] [3] [EXE] | 3.666666666 |

* Cette suppression ne concerne que l'affichage. Les calculs internes sont exécutés par arrondissement.

(Suite) [ANS] | 3.666666667 |

- **Arrondissement**

[S] [E] [T] [5] [EXE]

[1] [1] [/] [3] [EXE] | 3.666666667 |

L'instruction SET retourne à l'arrondissement lorsque l'alimentation est coupée automatiquement ou manuellement.
Dans ce manuel, les calculs sont effectués en "SET 5" (arrondissement).

Chapitre 3

Calculs manuels

3-1 Explication des calculs manuels

En calcul manuel, les opérations ne sont pas exécutées automatiquement à l'aide de formules introduites précédemment, comme en calcul programmé. Les opérations pour déplacer les nombres de droite à gauche, effectuer des calculs et appeler le contenu des variables sont dites "calculs manuels".

3-2 Opérations en calcul manuel

- Les additions, soustraction, multiplications et division sont exécutées dans l'ordre algébrique normal. [+], [−], [×] (X), [/] (÷) et [EXE] (=) sont utilisés. La touche [EXE] sert à obtenir le résultat des calculs et fait fonction de touche d'égal "=".

Exemple: 12+36−9×5÷4=36.75

Opération: [1][2][+][3][6][−][9][×][5][/][4] → 12+36−9*5/4

[EXE] → 36.75

- Les calculs de fonctions comprennent l'addition, la soustraction, la multiplication et la division et suivent l'ordre d'algèbre logique. Les données sont écrites après les instructions de fonctions.

Exemple: log 1.23=0.08990511144

Opération: LOG 1[·]23 → LOG1.23

[EXE] → 0.08990511144

* Dans ce manuel, les lettres et chiffres sont représentés sans cadre.

Exemple: [S][I][N][(][1][5][+][8][)][EXE] → SIN(15+8)[EXE]

* Les instructions de fonction et de programmation peuvent être rédigées soit d'une touche ou composées de lettres alphabétiques. Dans ce manuel, elles sont écrites alphabétiquement.

- Les calculs de mémoire utilisés pour stocker des valeurs ou des résultats de calculs et pour obtenir des totaux utilisent des variables. Ces variables sont représentées par des lettres alphabétiques (A - Z), ou des combinaisons de lettres et chiffres (0—9) (quand le nombre de mémoire a été étendu à plus de 26). Pour attribuer une valeur numérique ou le résultat d'un calcul à une variable, utiliser une substitution.

Exemple: Attribuer 1234 à la variable A

Opération: A[=]1234 → A=1234

[EXE] →

Exemple: Ajouter le résultat de 23 x 56 dans la variable K

Opération: K ⊟ K ⊞ 23 ⊠ 56 | K=K+23*56

EXE

Cette méthode est l'équivalent manuel d'une substitution de phrase dans un programme.

- Pour effectuer une correction avant pression de la touche EXE, déplacer le curseur au point à corriger et presser la touche de correction. (voir chapitre 2)
- Pour annuler l'affichage complet, presser la touche AC.

3-3 Exemple de calculs manuels

3-3-1 Calculs de base

■ **Addition, soustration, multiplication et division**

Exemple: $23+4.5-53=-25.5$

Opération: 23 ⊞ 4 ⊡ 5 ⊟ 53 EXE | -25.5

Exemple: $56\times(-12)\div(-2.5)=268.8$

Opération: 56 ⊠ (⊟ 12) ⊘ (⊟ 2 ⊡ 5) EXE | 268.8

(peut être omis)

Exemple: $12369\times7532\times74103=6.903680613\times10^{12}$ (=69036806 13000)

Opération: 12369 ⊠ 7532 ⊠ 74103 EXE | 6.903680613E 12

Exemple: $1.23\div90\div45.6=2.997076023\times10^{-4}$ (=0.0002997076023)

Opération: 1 ⊡ 23 ⊘ 90 ⊘ 45 ⊡ 6 EXE | 2.997076023E-04

* Si le résultat est supérieur à 10^9 ou inférieur à 10^{-3}, il est affiché en notation exponentielle.

Exemple: $\underline{7\times8}+\underline{4\times5}=76$

Opération: 7 ⊠ 8 ⊞ 4 ⊠ 5 EXE | 76

Exemple: $12+\underline{(2.4\times10^5)\div42.6}-\underline{78\times36.9}=2767.602817$

Opération:

12 ⊞ 2 ⊡ 4 π/E 5 ⊘ 42 ⊡ 6 ⊟ 78 ⊠ 36 ⊡ 9 EXE | 2767.602817

■ Calculs de mémoire

Exemple: 12×45=540
12×31=372
75÷12=6.25

Opération:

A [=] 12 [EXE]	
A [×] 45 [EXE]	540
A [×] 31 [EXE]	372
75 [÷] A [EXE]	6.25

Exemple:

23+9=32
53−6=47
−) 45×2=90
99÷3=33
Total 22

Opération:

M [=] 23 [+] 9 [EXE]
M [=] M [+] 53 [−] 6 [EXE]
M [=] M [−] 45 [×] 2 [EXE]
M [=] M [+] 99 [÷] 3 [EXE]
M [EXE] 22

* En utilisant cette méthode, le résultat des calculs individuels ne peut pas être vérifié. Si les résultats individuels sont désirés, procéder comme suit:

23 [+] 9 [EXE]	32
M [=] [ANS] [EXE]	
53 [−] 6 [EXE]	47
M [=] M [+] [ANS] [EXE]	
45 [×] 2 [EXE]	90
M [=] M [−] [ANS] [EXE]	
99 [÷] 3 [EXE]	33
M [=] M [+] [ANS] [EXE]	
M [EXE]	22

3-3-2 Calculs de fonctions

■ **Fonctions trigonométriques (sin, cos, tan) et trigonométriques inverses ($\sin^{-1}$, $\cos^{-1}$, $\tan^{-1}$)**

- Lors d'utilisation de fonctions trigonométriques et trigonométriques inverses, s'assurer de désigner la mesure angulaire.
  (Si la mesure angulaire n'est pas modifiée, il n'est pas nécessaire d'effectuer une nouvelle désignation.)

Exemple: 14°25′36″=14.42666667°

Opération: DEG [(] 14 [.] 25 [.] 36 [)] [EXE]

14.42666667

* "DEG" peut être entré soit d'une touche soit alphabétiquement. (de même pour la suite)

Exemple: 14.2536°=14°15′12.96″

Opération: DMS 14 [.] 2536 [EXE]

14° 15′ 12.96″

Exemple: sin12.3456°=0.2138079201

Opération: [MODE] [4] → "DEG"

SIN 12 [.] 3456 [EXE]

0.2138079201 DEG

(ou [F1] [SIN] 12 [.] 3456 (de même pour ci-dessous))

Exemple: 2·sin45°×cos65°6′=0.5954345575

Opération:

2 [×] SIN 45 [×] COS DEG [(] 65 [.] 6 [)] [EXE]

0.5954345575 DEG

Exemple: $\sin^{-1}0.5=30°$ (Quel est x quand $\sin x° = 0{,}5$?)

Opération: ASN 0 [.] 5 [EXE]

30 DEG

(ou [F1] [ARC] [SIN] 0 [.] 5 (de même pour ci-dessous))

Exemple: $2.5\times(\sin^{-1}0.8-\cos^{-1}0.9)=68.22042398$

Opération:

2 [.] 5 [×] [(] ASN 0 [.] 8 [−] ACS 0 [.] 9 [)] [EXE]

68.22042398 DEG

Exemple: $\cos(\frac{\pi}{3}\text{rad})=0.5$

Opération: [MODE] [5] → "RAD"

COS [(] [F1] [π] [÷] 3 [)] [EXE]

0.5 RAD

Exemple: $\cos^{-1}\frac{\sqrt{2}}{2}=0.7853981634$

Opération: ACS [(] SQR 2 [÷] 2 [)] [EXE]

0.7853981634 RAD

Exemple: tan(−35gra)= −0.6128007881

Opération: [MODE] [6] → "GRA"

TAN [−] 35 [EXE]

−0.6128007881 GRA

■ **Fonctions logarithmiques (log, ln) et exponentielles (e^x, x^y)**

Exemple: $\log 1.23\ (=\log_{10} 1.23) = 0.08990511144$

Opération: LOG 1 [·] 23 [EXE]

0.08990511144

Exemple: $\ln 90\ (=\log_e 90) = 4.49980967$

Opération: LN 90 [EXE]

4.49980967

Exemple: $\log 456 \div \ln 456 = 0.4342944819$

Opération: LOG 456 [÷] LN 456 [EXE]

0.4342944819

Exemple: $e^{4.5} = 90.0171313$

(Quel est l'antilogarithme du logarithme naturel 4,5?)

Opération: EXP 4 [·] 5 [EXE]

90.0171313

Exemple: $10^{1.23} = 16.98243652$

(Quel est l'antilogarithme du logarithme commun 1,23?)

Opération: 10 [↑] 1 [·] 23 [EXE]

16.98243652

Exemple: $5.6^{2.3} = 52.58143837$

Opération: 5 [·] 6 [↑] 2 [·] 3 [EXE]

52.58143837

Exemple: $123^{\frac{1}{7}}\ (=\sqrt[7]{123}) = 1.988647795$

Opération: 123 [↑] [(] 1 [/] 7 [)] [EXE]

1.988647795

Exemple: $(78-23)^{-12} = 1.305111829 \times 10^{-21}$

Opération: [(] 78 [−] 23 [)] [↑] [−] 12 [EXE]

1.305111829E-21

Exemple: $2^2 + 3^3 + 4^4 = 287$

Opération: 2 [↑] 2 [+] 3 [↑] 3 [+] 4 [↑] 4 [EXE]

287

Exemple: $\log \sin 40° + \log \cos 35° = -0.2785679838$

L'antilogarithme est 0,5265407845 (calcul logarithmique de sin 40° x cos 35°)

Opération: [MODE] [4] (désignation DEG)

LOG SIN 40 [+] LOG COS 35 [EXE]

10 [↑] [ANS] [EXE]

-0.2785679838
0.5265407845

■ **Fonctions hyperboliques (sinh, cosh, tangh) et hyperboliques inverses ($\sinh^{-1}$, $\cosh^{-1}$, $\tanh^{-1}$)**

Exemple: sinh 3.6=18.28545536

Opération: HSN 3 [·] 6 [EXE] | 18.28545536 |

(ou [F1] [HYP] [SIN] 3 [·] 6 , de même pour ci-dessous)

Exemple: tanh 2.5=0.9866142981

Opération: HTN 2 [·] 5 [EXE] | 0.9866142981 |

Exemple: cosh 1.5−sinh 1.5=0.2231301602

Opération: HCS 1 [·] 5 [−] HSN 1 [·] 5 [EXE] | 0.2231301602 |

Exemple: $\sinh^{-1}30=4.094622224$

Opération: AHS 30 [EXE] | 4.094622224 |

(ou [F1] [ARC] [HYP] [SIN] 30, (de même pour ci-dessous))

Exemple: **Quel est x si tanh $4x$ = 0,88?** $x=\frac{\tanh^{-1}0.88}{4}=0.3439419141$

Opération: AHT 0 [·] 88 [/] 4 [EXE] | 0.3439419141 |

■ **Autres fonctions $\sqrt{\ }$, $x!$, SGN, RAN#, RND, ABS, INT, FRAC)**

Exemple: $\sqrt{2}+\sqrt{5}=3.65028154$

Opération: SQR 2 [+] SQR 5 [EXE] | 3.65028154 |

Exemple: $8!\ (=1\times2\times3\times\cdots\cdots\times7\times8)=40320$

Opération: 8 [F1] [ǐ] [EXE] | 40320 |

Exemple: **Attribuer "1" à un nombre positif, "−1" à un nombre négatif et "0" à zéro.**

Operation:

SGN 6 [EXE]	1
SGN 0 [EXE]	0
SGN [−] 2 [EXE]	−1

Exemple: **Génération de nombres aléatoires (nombre pseudoaléatoire de 0 < RAN# < 1)**

Opération: RAN [#] [EXE] | 0.904186914 |

Exemple: **Le résultat de 12,3 x 4,56 est arrondi à 10^{-2}** $12.3\times4.56=56.088$

Opération: RND [(] 12 [·] 3 [×] 4 [·] 56 [,] [−] 2 [)] [EXE] | 56.1 |

Exemple: $|-78.9\div5.6|=14.08928571$

Opération: ABS [(] [−] 78 [·] 9 [/] 5 [·] 6 [)] [EXE] | 14.08928571 |

Exemple: **La partie entière de $\frac{7800}{96}$ est 81**

Opération: INT [(] 7800 [/] 96 [)] [EXE] | 81 |

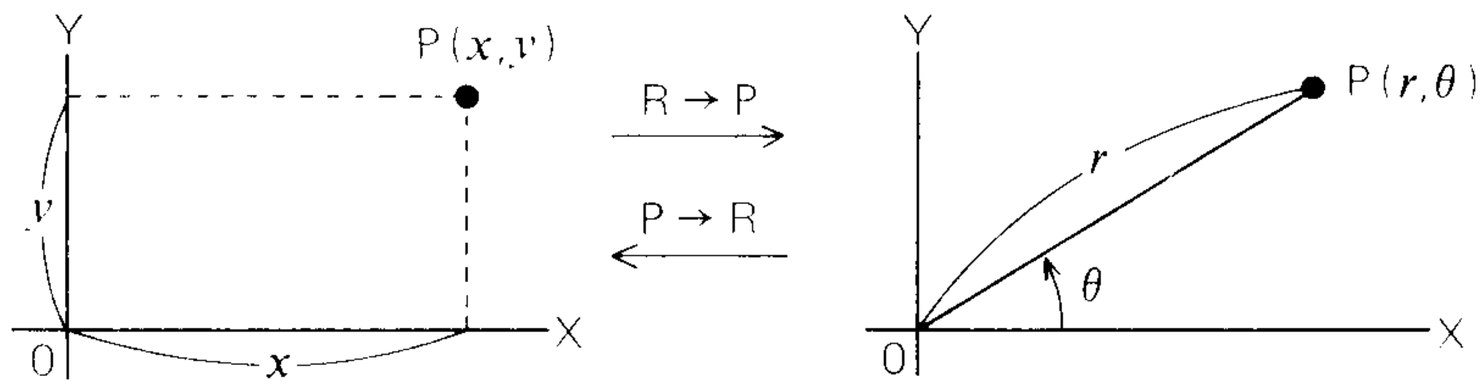
* Cette instruction ne dépassera pas la valeur du nombre original.

Exemple: **La partie décimale de $\frac{7800}{96}$ est 0,25**

FRAC [(] 7800 [/] 96 [)] [EXE] | 0.25 |

■ Conversion de coordonnées (R → P, P → R)

● Coordonnées rectangulaires **● Coordonnées polaires**



Le θ de R → P peut être obtenu dans la gamme de $-180° < \theta \leq 180°$.
(même gamme pour "RAD" et "GRA")

Exemple: Quels sont r et θ si $x = 14$ et $y = 20{,}7$?

Opération: [MODE] [4] (désignation DEG)

RPC 14 [·] 20 [·] 7 [EXE]

(r)	**X [EXE]**	24.98979792
(θ)	**DMS Y [EXE]**	55°55'42.20"

Exemple: Quels sont r et θ si $x = 7{,}5$ et $y = -10$?

Opération: [MODE] [5] (désignation RAD)

RPC 7 [·] 5 [·] [−] 10 [EXE]

(r)	**X [EXE]**	12.5
(θ)	**Y [EXE]**	-0.927295218

Exemple: Quels sont x et y si $r = 25$ et $\theta = 56°$?

Opération: [MODE] [4]

PRC 25 [·] 56 [EXE]

(x)	**X [EXE]**	13.97982259
(y)	**Y [EXE]**	20.72593931

Exemple: Quels sont x et y si $r = 4{,}5$ et $\theta = 2/3\ \pi$ rad?

Opération: [MODE] [5]

PRC 4 [·] 5 [·] 2 [/] 3 [×] [F1] [π] [EXE]

(x)	**X [EXE]**	-2.25
(y)	**Y [EXE]**	3.897114317

■ **Désignation du nombre de chiffres et du nombre de décimales considéré**

La désignation du nombre de chiffres et du nombre de décimales s'effectue à l'aide de l'instruction "SET".

Désignation du nombre de chiffres considéré . . . . . SET E n (n = 0 à 9)
Désignation des décimales . . . . . SET F n
Annulation de désignation. . . . . SET N

* En ce qui concerne la désignation du nombre de chiffres, "SET E 0" correspond à 10 chiffres.
* Lorsqu'une telle désignation est effectuée, la fonction des chiffres excédentaires (voir page 13) cause la suppression et l'arrondissement de l'affichage un chiffre avant l'unité désignée. Toutefois, le nombre original est conservé dans la section de calcul et dans la mémoire.

Exemple: 100÷6=16.66666666.....
Opération: SET [E] 4 [EXE] (nombre de chiffres désigné = 4)

100 ◪ 6 [EXE] | 1.667E 01

Exemple: 123÷7=17.57142857.....
Opération: SET [F] 2 [EXE] (nombre de décimales désigné = 2)

123 ◪ 7 [EXE] | 17.57

Exemple: 1÷3=0.3333333333.....
Opération: SET [N] [EXE] (suppression de la désignation)

1 ◪ 3 [EXE] | 0.3333333333

3-3-3 Calculs statistiques

- Avant de commencer un calcul statistique, s'assurer de presser [F1] [SAC] pour effacer la mémoire statistique.

■ **Calcul d'écart standard**

- Presser la touche [STAT] une fois après chaque donnée entrée.
  Donnée [STAT]
- Lorsque plusieurs données identiqués se suivent, presser la touche [STAT] le nombre de fois désiré, ou entrer la donnée [;] nombre de répétitions [STAT].
- Ecart standard

$$\sigma_{n-1} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} = \sqrt{\frac{\Sigma x^2 - (\Sigma x)^2/n}{n-1}}$$

[Mettre la donnée pechantillon dans un groupe et désigner l'écart standard de celui-ci.]

- Moyenne

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} = \frac{\Sigma x}{n}$$

Exemple: Les données sont **55, 54, 51, 55, 53, 53, 54, 52**
Opération: [F1] [SAC]

55 [STAT] 54 [STAT] 51 [STAT] 55 [STAT]
53 [STAT] [STAT] 54 [STAT] 52 [STAT] | STAT 52
(écart standard σ_n) SDXN [EXE] | 1.316956719
(écart standard σ_{n-1}) SDX [EXE] | 1.407885953
(moyenne $\bar{x}$) MX [EXE] | 53.375
(donnée n) CNT [EXE] | 8
(somme des x Σx) SX [EXE] | 427
(somme des x au carré Σx^2) SX2 [EXE] | 22805

Exemple: Parmi les valeurs ci-dessous, quels sont σ_n et σ_{n-1} ?

Numéro	Grade	Degré
1	110	10
2	130	31
3	150	24
4	170	2
5	190	3

Opération:

F1 SAC

110 [;] 10 STAT

130 [;] 31 STAT

150 [;] 24 STAT

170 STAT STAT

190 STAT STAT STAT

F1 ASTAT ANS

STAT 190
CNT= 70
SX=9640
SY=0
SX2=1351000
SY2=0
SXY=0
18.2968716
18.42898069

SDXN EXE

SDX EXE

* Suppression et correction d'une donnée fausse (opération correcte: 51 STAT)
(1) 50 STAT → ensuite F1 DEL STAT puis effectuer l'opération correcte.
(2) 49 STAT (quelques données auparavant) → 49 F1 DEL STAT puis effectuer l'opération correcte.
La suppression et la correction peuvent être accomplies de la même manière pour plusieurs points.
49 [;] 12 STAT (quelques points avant) → 49 [;] 12 F1 DEL STAT , puis effectuer l'opération correcte.

■ Calcul de régression

- Entrer donnée x [,] donnée y STAT pour donnée.
- S'il existe plusieurs paires de données identiques, presser la touche STAT le nombre de fois qu'il y a de données, ou donnée x [,] donnée y [;] nombre de répétitions STAT .

■ Calcul de régression linéaire

- L'équation de régression est $y = A + Bx$
  Les coefficients A et B sont calculés au moyen des formules suivantes:

Coefficient de régression ou équation de régression

$$B = \frac{n \cdot \Sigma xy - \Sigma x \cdot \Sigma y}{n \cdot \Sigma x^2 - (\Sigma x)^2}$$

Terme constante de l'équation de régression

$$A = \frac{\Sigma y - B \cdot \Sigma x}{n}$$

- Le coefficient de corrélation r des paires de données entrées est calculé à l'aide de la formule suivante.

$$r = \frac{n \cdot \Sigma xy - \Sigma x \cdot \Sigma y}{\sqrt{\{n \cdot \Sigma x^2 - (\Sigma x)^2\}\{n \cdot \Sigma y^2 - (\Sigma y)^2\}}}$$

Exemple: ● Température et longueur d'une barre d'acier

Température	Longueur
10°C	1003mm
15	1005
20	1010
25	1008
30	1014

Obtenir l'équation de régression et le coefficient de corrélation à partir du résultat mesuré. Calculer la longueur à 18°C et la température à 1000mm.

Opération: [F1] [SAC]

10 [.] 1003 [STAT]	STAT 10, 1003
15 [.] 1005 [STAT]	STAT 15, 1005
20 [.] 1010 [STAT]	STAT 20, 1010
25 [.] 1008 [STAT]	STAT 25, 1008
30 [.] 1014 [STAT]	STAT 30, 1014
(terme constant A) **LRA** [EXE]	998
(coefficient de régression B) **LRB** [EXE]	0.5
(coefficient de corrélation r) **COR** [EXE]	0.9190182776
(longueur à 18°C) **EOY 18** [EXE]	1007
(température à 1000mm) **EOX 1000** [EXE]	4

* Pour supprimer ou corriger une entrée fausse (opération correcte: 10 [.] 1003 [STAT])

(1) 11 [.] 1003 [STAT] → ensuite [F1] [DEL STAT] , puis effectuer l'opération correcte.

(2) 10 [.] 1030 [STAT] → ensuite [F1] [DEL STAT] , puis effectuer l'opération correcte.

(3) 11 [.] 1003 [STAT] (plusieurs points avant) → 11 [.] 1003 [F1] [DEL STAT] , puis effectuer l'opération correcte.
Des quantités peuvent également être supprimées ou corrigées en procédant de même.
11 [.] 1003 [;] 10 [STAT] (plusieurs points avant) → 11 [.] 1003 [;] 10 [F1] [DEL STAT] , puis effectuer l'opération correcte.

■ Calculs de régression logarithmique

- L'équation de régression est $y = A + B \cdot \ln x$. La donnée x est introduite en tant que (ln) de x. La donnée y est introduite de même qu'en régression linéaire.
- Pour obtenir le coefficient de régression ou pour corriger, procéder de même qu'en régression linéaire, mais $\Sigma \ln x$ est requis pour Σx, $\Sigma(\ln x)^2$ pour Σx^2 et $\Sigma \ln x \cdot y$ pour Σxy.

Exemple:

x_i	y_i
29	1.6
50	23.5
74	38.0
103	46.4
118	48.9

Déterminer la régression logarithmique de ces données et tirer le terme constant, le coefficient et la corrélation de l'équation de régression, puis calculer le coefficient déterminant (r^2).

Opération: [F1] [SAC]

LN 29 [.] 1 [.] 6 [STAT]	TAT 3.36729583, 1.6
LN 50 [.] 23 [.] 5 [STAT]	T 3.912023005, 23.5
LN 74 [.] 38 [.] 0 [STAT]	TAT 4.304065093, 38
LN 103 [.] 46 [.] 4 [STAT]	T 4.634728988, 46.4
LN 118 [.] 48 [.] 9 [STAT]	T 4.770684624, 48.9
(terme constante A de l'équation de régression) **LRA** [EXE]	-111.1283976
(coefficient de régression B) **LRB** [EXE]	34.02014749
(coefficient de corrélation r) **COR** [EXE]	0.9940139464
(coefficient déterminant r^2) [↑] 2 [EXE]	0.9880637256

■ Calcul de régression exponentielle

- L'équation de régression est $y = A \cdot e^{B \cdot x}$ ($\ln y = \ln A + B \cdot x$).
  La donnée y est introduite en tant que logarithme (ln) de y. La donnée x est introduite comme en régression linéaire.
- La méthode de correction est pareille qu'en régression linéaire, mais lnA est requis pour le terme constant A, $\Sigma \ln y$ pour somme des y (SY), $\Sigma(\ln y)^2$ pour somme des y^2 (SY2) et $\Sigma x \cdot \ln y$ pour somme des $x \cdot y$ (SXY).

Exemple:

x_i	y_i
6.9	21.4
12.9	15.7
19.8	12.1
26.7	8.5
35.1	5.2

Etablir la régression exponentielle de ces données et obtenir l'équation de régression et le coefficient de corrélation.

Opération:

F1 SAC	
6 · 9 · LN 21 · 4 STAT	AT 6.9, 3.063390922
12 · 9 · LN 15 · 7 STAT	T 12.9, 2.753660712
19 · 8 · LN 12 · 1 STAT	T 19.8, 2.493205453
26 · 7 · LN 8 · 5 STAT	T 26.7, 2.140066163
35 · 1 · LN 5 · 2 STAT	T 35.1, 1.648658626
(terme constant A) EXP LRA EXE	30.49758742
(coefficient B) LRB EXE	-0.04920370831
(coefficient de corrélation r) COR EXE	-0.9972473519

■ Calcul de régression à une puissance

- L'équation de régression est $y = A \cdot x^B$ ($\ln y = \ln A + B \ln x$).
  Les deux données x et y sont introduites sous la forme de logaritme (ln).
- La méthode de correction est pareille qu'en régression linéaire, mais lnA est requis pour le terme constant A, $\Sigma \ln x$ pour Σx, $\Sigma(\ln x)^2$ pour Σx^2, $\Sigma \ln y$ pour Σy, $\Sigma(\ln y)^2$ pour Σy^2, et $\Sigma(\ln x, \ln y)$ pour Σxy.

Exemple:

x_i	y_i
28	2410
30	3033
33	3895
35	4491
38	5717

Etablir la régression à la puissance de ces données et obtenir l'équation de régression et le coefficient de régression.

Opération:

F1 SAC	
LN 28 · LN 2410 STAT	3220451, 7.787382026
LN 30 · LN 3033 STAT	1197382, 8,017307508
LN 33 · LN 3895 STAT	6507561, 8,267448958
LN 35 · LN 4491 STAT	5348061, 8.409830673
LN 38 · LN 5717 STAT	3758616, 8.651199471
(terme constant A) EXP LRA EXE	0.2388010829
(coefficient B) LRB EXE	2.771866148
(coefficient de corrélation r) COR EXE	0.9989062562

Chapitre 4

Calculs programmés

Cet instrument utilise le langage de programmation BASIC. BASIC est une abbréviation de "Begginer's All-purpose Symbolic Instruction Code". Il est considére que c'est le langage de base le plus facile à utiliser pour les néophytes de la programmation.

Particularité du langage BASIC

1. C'est un langage orienté vers la résolution de problèmes d'une excellente efficacité de programmation.
2. Nullement restreint à un domaine spécifique, au contraire ce langage est à buts multiples et peut être appliqué à de nombreux secteurs, notamment les scienes naturelles, la sociologie et le commercial.
3. C'est un langage du type conversationnel; l'utilisateur rédige son programme en conversant avec l'ordinateur et utilise ses données pour effectuer les entrées.
4. Il est doté de nombreuses possibilités et caractéristiques propres au FORTAN, mais sans ses nombreuses restrictions.
5. De nombreuses fonctions essentielles sont prêtes à l'emploi, et la gamme de calcul est étendue.

4-1 Description générale d'un programme

Calcul programmé **1)** Programme le contenu des calculs à exécuter **2)** la calculatrice stocke le programme et **3)** à l'aide de ce programme, introduire simplement les données et les résultats sont automatiquement obtenus.

■ Bases de la programmation

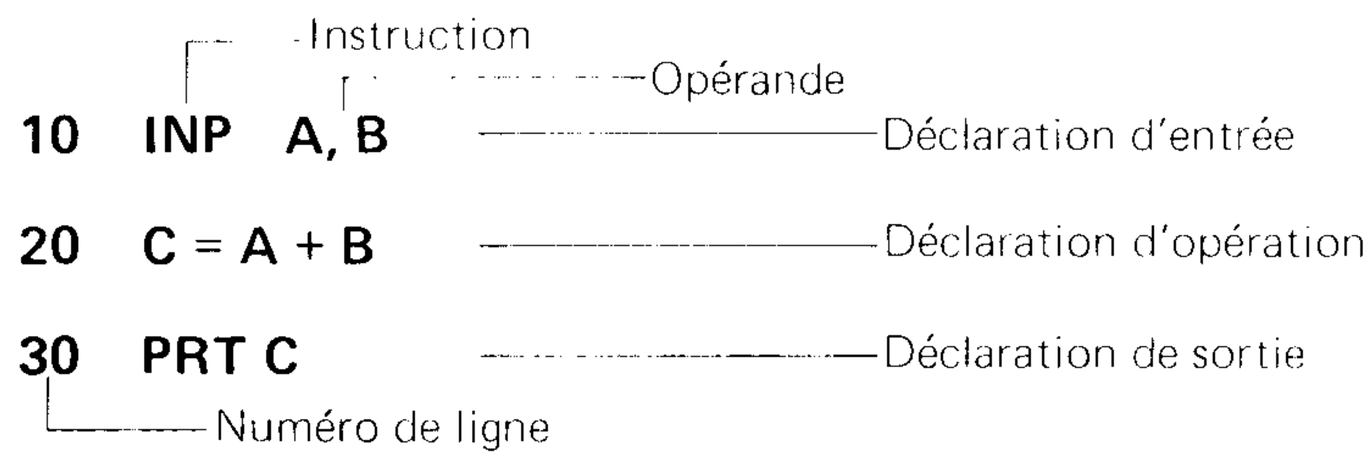
Examinons la programmation nécessaire pour faire exécuter à l'ordinateur un calcul donné, ainsi que le concept et la procédure de programmation.

● Programmes et programmation

Lorsqu'on utilise un ordinateur pour résoudre un problème, des instructions assimilables par l'ordinateur doivent être utilisées. Ces instructions forment un programme, et la composition de ces instructions est dite la programmation.

● Qu'est-ce qu'un programme?

Un programme est soumis à un grand nombre de règles grammaticales, dont le détail sera expliqué plus avant. Premièrement, voyons ce qu'est un programme, et quelle est sa forme. Pour cela, prenons un exemple de programme élémentaire.



Le programme ci-dessus représente un programme élémentaire et se compose d'une déclaration d'entrée, d'une déclaration d'opération, et d'une déclaration de sortie, chacune dotée d'un numéro de ligne. En clair, la déclaration d'entrée permet d'entrer une donnée. Sur la base de cette donnée, la déclaration d'opération exécute une opération quelconque, et la déclaration de sortie délivre les résultats de l'opération. Chaque ligne comporte un numéro de ligne qui la prècède. Ces instructions ne sont pas limitées à une seule, mais peuvent être exécutées plusieurs fois. Avec des instructions de décision, le programme peut devenir long et compliqué. Néanmoins, les bases de programmation restent identiques.

Sur une ligne, à la suite de son numéro, on trouve une INSTRUCTION qui indique à la calculatrice ce qu'elle doit faire ensuite. Elle se compose de caractères alphabétiques. Celle-ci est suivie d'OPERANDES qui donnent les informations nécessaires pour l'instruction à exécuter.

Ceci est un PROGRAMME sous sa forme élémentaire.

- **Nombre de pas de programme**

Les pas de programme sont comptés comme suit.

1) Commande de programme . . . . . 1 pas/1 commande
2) Commande de fonction . . . . . 1 pas/1 commande
3) Numéro de ligne . . . . . 2 pas/1 numéro de ligne
4) Caractère . . . . . 1 pas/1 caractère
5) Appui sur la touche [EXE] après chaque entrée par touche de numéro de ligne pour stocker dans la calculatrice . . . . . 1 pas

Exemple:

1 INP A [EXE] . . . . . 5 pas
(2 1 1 1)

10 B = SIN A [EXE] . . . . . 7 pas
(2 1 1 1 1 1)

100 PRT " B = " ; B [EXE] . . . . . 10 pas
(2 1 1 1 1 1 1 1 1)

Total: 22 pas

Note: L'affectation d'un mot de passe nécessite 6 pas. (Voir page 74.)

■ Ordre de programmation

La programmation se déroule dans l'ordre suivant:

1) Analyse du problème
2) Préparation d'un organigramme
3) Rédaction du programme sur une feuille de codage
4) Mise au point du programme

Explication

1ère étape: Analyse du problème donné de façon à déterminer les étages nécessaires pour sa résolution.

2è étape: Rédaction d'un organigramme pour représenter la suite logique menant à la solution du problème. L'organigramme est composé de symboles représentant le traitement ainsi que les éléments de décision.

3è étape: Sur la base de l'organigramme, rédaction du programme sur des feuilles de codage ou formulaires semblables à l'aide des instructions de langage BASIC.

4è étape: Détection des erreurs de programme pour sa mise au point.

Ces 4 étapes représentent la procédure d'élaboration d'un programme.

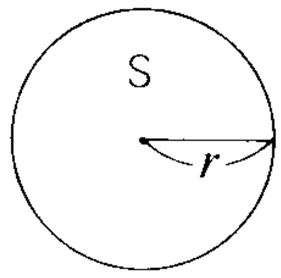
■ Organigramme

Les symboles les plus couramment utilisés sont les suivants.

Symbole	Désignation	Signification
	Point terminal	Début, fin.
	Entrée/sortie	Fonctions d'entrée/sortie.
	Traitement	Diverses fonctions de traitement.
	Traitement prédéterminé	Groupe d'instructions définies par ailleurs, par exemple par un sous-programme.
	Décision	Décision sur le choix de la voie empruntée parmi plusieurs voies possibles.

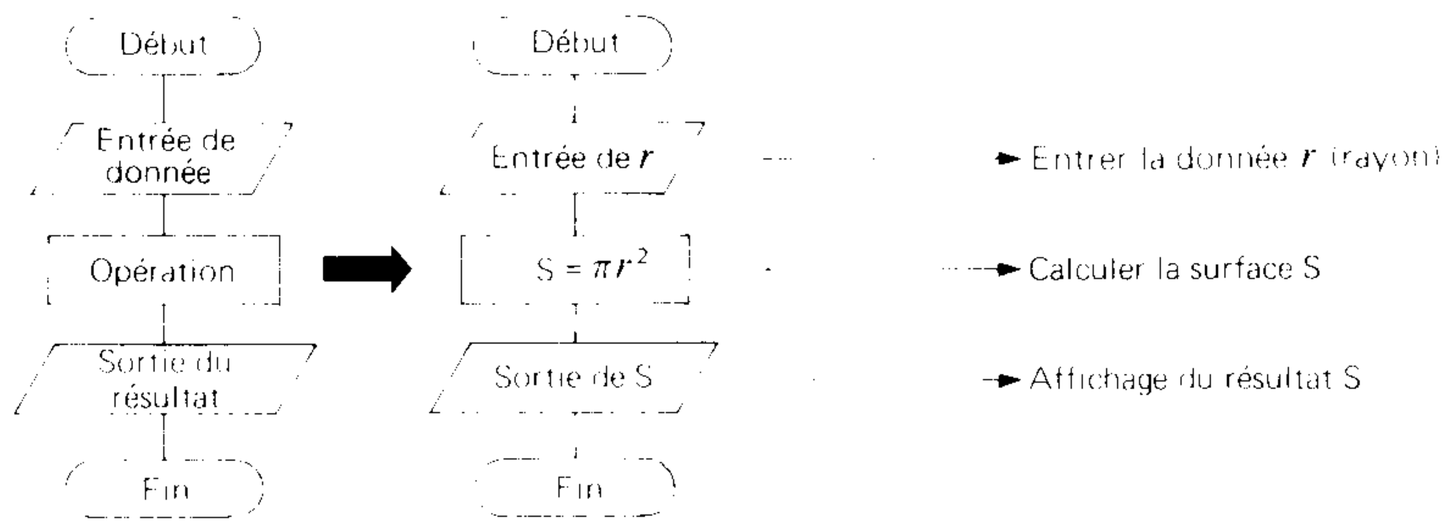
- **Exemple d'organigramme**

Prenons comme exemple un programme simple pour déterminer la surface d'un cercle.



$S = \pi r^2$

Considérons isolément l'entrée de donnée, l'opération et la sortie du résultat, comme illustré ci-dessous.



L'organigramme préparé de cette manière montre le suivi du programme; il est composé en regardant le programme dans son ensemble.
En programmant, prendre comme habitude d'établir un organigramme. De cette manière, on évitera toute confusion lors de l'élaboration de programmes plus conséquents.

■ Codage

Le codage est le processus d'écriture de programmes sur des feuilles de codage ou autres formulaires semblables après son assemblage. Lors de l'assemblage de programmes, certains opérateurs sont nécessaires. On trouvera ci-dessous les symboles d'opérateurs les plus simples.
Les quatre symboles d'opérations arithmétiques
+ et − sont exprimé sous forme " + " et " − ".
x et ÷ sont exprimés sous forme " ✱ " et " / "
Les puissances, telles que x^2 et x^3 sont exprimées sous forme "x↑2" et "x↑3".
Le symbole " = " utilisé dans les instructions d'affectation est brièvement expliqué ici. Il sera repris en détail ultérieurement. Par exemple, le " = " de S = πr^2 signifie qu'on attribue à S le résultat du calcul πr^2, par opposition à sa signification "égale" mathématique.
Rédigeons un programme pour déterminer la surface du cercle définie auparavant.

1) Rédiger une déclaration d'entrée de la donnée r.

Il existe plusieurs instructions d'entrée, mais en général on utilise l'instruction "INP" pour entrer les données frappées sur le clavier au cours de l'exécution du programme. L'instruction d'entrée utilisée pour introduire la donnée r devient INP R. Si l'on y ajoute un numéro de ligne pour la compléter, elle devient 10 INP R.
Les numéros de ligne peuvent être attribués de 1 à 9999, mais dans un même programme il est impossible d'utiliser deux fois le même numéro. Dans un tel cas, le numéro de ligne attribué en dernier prendrait la précédence. Il est en principe plus pratique d'attribuer des numéros de ligne par accroissements de 10 de sorte à faciliter l'addition, la correction et la suppression. Lorsqu'un programme est démarré, les instructions sont exécutées dans l'ordre des numéros de ligne, du plus petit au plus grand. Il s'agit par conséquent d'attribuer les numéros de ligne dans l'ordre d'exécution du programme.

2) Ensuite, faire une déclaration d'affectation pour attribuer à S le résultat du calcul avec la donnée d'entrée.

Etant donné que πr^2 signifie $\pi \times r^2$,
S = π✱R↑2 (X est représenté par ✱ et R^2 par R↑2)
En y ajoutant le numéro de ligne comme ci-dessus, on obtient
20 S = π✱R↑2

3) Rédiger une déclaration de sortie pour délivrer (afficher) le résultat de l'opération.

Ce qu'on a fait jusqu'à présent ne concerne que le calcul et ne commande pas l'affichage. Utiliser "PRT" pour ordonner l'affichage.
La déclaration de sortie pour afficher le résultat S devient:
PRT S
En y ajoutant le numéro de ligne:
30 PRT S
Ainsi, le programme complet pour déterminer la surface d'un cercle devient:
10 INP R
20 S = π✱R↑2
30 PRT S
Il n'est pas nécessaire d'utiliser la feuille de codage spéciale pour la programmation, toutefois son utilisation facilite l'analyse du problème et la rédaction de l'organigramme.

4-2 Notions de base de programmation

■ Constantes et variables

Les caractères utilisés en langage BASIC sont les lettres majuscules de l'alphabet (A à Z) et les chiffres (0 à 9) ainsi que certains caractères spéciaux (tels que les symboles).

• Constantes

Une constante est une valeur fixe qui peut être directement incorporée au programme.

Exemple: L'expression $S = \pi r^2$ devient S = π✱R↑2; 2 est constant.

• Variables

Une variable est une valeur qui est représentée dans le programme mais qui est introduite sur le clavier pendant son exécution. Elle est également utilisée pour le résultat du calcul car initialement il est inconnu et est attribué au cours de l'exécution.
Une variable est représentée par une lettre majuscule (A à Z) ou une combinaison d'une lettre majuscule et d'un seul chiffre (A0, A1 à A9, B0 à T9 . . . . lorsque la mémoire est étendue). L'affectation est faite à l'intérieur de cette gamme.

Exemple: Dans l'expression $S = \pi r^2$, qui devient S = π✱R↑2, R est la variable.

Exemple:

```
Y = 2 ✱ X ↑ 2 + 3 ✱ X + 4
|   |   |   |   |   |   |
V   C   V   C   C   V   C
```

V: Variable

C: Constante

Ainsi, les variables et constantes algébriques sont ici également des variables et constantes. De plus, il existe des constantes de caractères et des variables de caractères. Les premières sont des chaînes de caractères qui sont directement incorporées au programme, telles que "ABC" et "END", enfermées entre des guillemets et définies lettre à lettre. Les variables de caractères ne sont pas des valeurs numériques, mais des variables qui acceptent une chaîne de caractères. Chaque fois qu'une chaîne de caractères lui est attribuée, son contenu change.
Une chaîne de caractères est composée de caractères enfermés entre des guillemets, comme par exemple "123", et ne possède pas de valeur numérique. Ainsi, "123" n'est que la suite des chiffres 1, 2 et 3, et est considéré comme égal à "ABC" entre guillemets.
Les variables de caractères sont des variables d'ordre général (telles que A, B, X et Y) avec un signe $ attaché. Elles peuvent être sélectionnées dans cette gamme.

Exemple: A$, B$, C$, X$, Y$

Les variables de caractères peuvent être comparées ou ajoutées l'une à l'autre, mais les autres opérations (telles que soustraction, multiplication et division) sont proscrites.

Exemple: Si A$ = "123" et B$ = "456",
C$ = A$ + B$
devient C$ = "123456"
(Si C$ = B$ + A$, C$ devient "456123")

Une telle variable de caractères peut recevoir une chaîne de jusqu'à 7 caractères.
En plus des variables de caractères, il y a les variables de caractères exclusives. Celles-ci sont affichées avec le signe $. Une telle chaîne peut recevoir jusqu'à 30 caractères.

Exemple: $ = "1234567890ABCDEFG"

Ces variables de caractères exclusives peuvent utiliser les fonctions de caractères (fonctions MID) qui seront décrites plus loin. Elles peuvent être utilisées de manière plus pratique que les variables de caractères conventionnelles.

■ Instructions d'affectation

En BASIC, les instructions d'affectation ont la forme suivante:

Variable = expression numérique

En ce qui concerne l'instruction d'affectation en BASIC, un terme doté d'une opération arithmétique (+, −, ✱, ÷) du côté droit est dit une expression numérique.

Exemple: En Y = 2✱X+3, le terme 2✱X+3 de droite est une expression numérique. Le signe " = " ne signifie pas égal, mais "affecté à".

Exemple: En Y = 2✱X+3, le terme gauche est la variable et le droit l'expression numérique. Ainsi, cela ne signifie pas, comme d'habitude en mathématiques, que le terme gauche Y est égal au terme droit 2✱X+3. Cela signifie que le résultat de 2✱X+3 est affecté à Y. (Il est plus facile de comprendre Y = 2✱X+3 si l'on écrit Y ← 2✱X+3)

Exemple d'instructions d'affectation

A = B . . . . . Affecte la valeur de B à A (l'ancienne valeur de A est annulée)
N = 2✱M . . . . . Affecte la double valeur de M à N
X = Y + Z . . . . . Affecte la somme de Y et Z à X
I = I + 1 . . . . . Affecte la valeur de I + 1 à I (la valeur numérique en I est incrémentée de 1)

4-3 Ecriture et exécution de programmes

■ Ecriture de programmes

Le stockage du programme dans la mémoire de la calculatrice est appelé écriture du programme.
Cette opération s'effectue en frappant les entrées sur le clavier.

1. Désignation du modo WRT
2. Désignation de la zone de programme
3. Entrée du programme ligne par ligne (écriture)
4. La zone programme peut être divisée en dix parties de P0 à P9. Les programmes s'écrivent quelque part dans cette zone.

(1) Désignation du mode WRT

L'écriture du programme est effectuée en mode WRT.
Frapper [MODE] [1] et "WRT" sera affiché.

(2) Désignation de la zone de programme

Pour désigner la zone de programme, frapper la touche [F1] puis une touche numérique de [0] à [9].

[F1] [P0/0] → **P0**
[F1] [P1/1] → **P1**
[F1] [P2/2] → **P2**
[F1] [P3/3] → **P3**
[F1] [P4/4] → **P4**
[F1] [P5/5] → **P5**
[F1] [P6/6] → **P6**
[F1] [P7/7] → **P7**
[F1] [P8/8] → **P8**
[F1] [P9/9] → **P9**

(3) Entrée du programme (écriture)

L'écriture du programme s'effectue ligne par ligne. Une ligne peut comprendre jusqu'à 62 caractères, y compris le numéro de ligne. En dernier lieu, frapper la touche [EXE].

*** Rôle de la touche [EXE]**

La touche [EXE] est utilisée pour écrire les programmes, entrer les données et obtenir les résultats en calcul manuel. Lors de l'écriture de programme, frapper la touche [EXE] après que la ligne ait été rédigée pour la stocker dans la calculatrice. Les changements, additions, suppressions dans le programme sont achevés par la pression de la touche [EXE]. Si le contenu de l'affichage est modifié et que la touche [EXE] n'est pas pressée, le contenu interne ne sera pas modifié.

Exemple: Ecrire le programme suivant en P0

```
10 INP A,B
20 V=A+B
30 W=A-B
40 PRT V,W
50 END
```

Opération:

1. **Désigner le mode WRT.**

MODE 1 — READY P6: 012345_789 1228

Ecrit en P6

Zone de programme présentement désignée — Zone de programme non écrite — Nombre de pas

* Le nombre de pas est affecté par le nombre de mémoires et le volume des programme déjà présents.

2. **Désigner la zone de programme P0.**

F1 P0 — READY P0: 012345_789 1228

3. **S'il y reste un programme antérieur, l'effacer.**

CLR EXE — READY P0: 012345_789 1228

Disparaît par la suite

(Si rien n'y est écrit, supprimer cette opération)

4. **Ecrire la ligne numéro 10.**

10 INP A . B EXE — 10 INP A,B 1221

Signifie 1 espace libre (peut être omis)

* Ne pas oublier de frapper en changeant de ligne

5. **Ecrire la ligne numéro 20.**

20 V = A + B EXE — 20 V=A+B 1213

6. **Ecrire la ligne numéro 30.**

30 W = A − B EXE — 30 W=A-B 1205

7. **Ecrire la ligne numéro 40.**

40 PRT V . W EXE — 40 PRT V,W 1198

8. **Ecrire la ligne numéro 50.**

50 END EXE — 50 END 1194

- Utiliser la commande "END" pour terminer un programme. La commande "END" peut être omise dans le cas d'un programme comme ci-dessus, mais elle doit être écrite pour clarifier la fin d'un programme lorsqu'on utilise des instructions GOTO ou des instructions GSB.
- Un espace est créé entre le numéro de ligne et les instructions et opérandes pour faciliter la lecture de l'affichage. En langage BASIC, cet espace n'a pas de signification particulière, sauf pour des messages tels que PRT.
- Les numéros de ligne ont été choisis par intervalles de 10. Toutefois, ceux-ci peuvent être librement choisis entre 1 et 9999. Cet intervalle de 10 permet par la suite d'effectuer des insertions et additions. Le programme est exécuté dans l'ordre des numéros de ligne, en commençant par le plus petit. Il s'agit par conséquent d'attribuer les numéros de ligne dans l'ordre de déroulement du programme.
- Utiliser la commande "CLR" comme ci-dessus pour effacer un programme précédemment écrit, et utiliser la commande "CLR ALL" pour effacer tous les programmes précédemment écrits (P0 à P9).

■ Exécution du programme

Le programme est exécuté dans le mode RUN. (Pour obtenir RUN, frapper [MODE] [0]). Il existe deux méthodes d'exécution des programmes écrits.

(1) Méthodes d'exécution des programmes

1. Exécution par zone de programme

Dans cette méthode, l'exécution commence dès la désignation de la zone de programme.

[F1] { [P0 0] ∫ [P9 9] } (Frapper l'une des touches [P0 0] à [P9 9] après [F1].)

Exemple: Débuter le programme donné à l'exemple précédent.

Mode RUN (disparaît par la suite)

Opération:

[F1] [P0 0]

RUN
?

* Ce signe "? " existe parce que l'instruction INP est présente au début.

2. Exécution par ordre RUN

RUN[EXE]

(RUN peut être rentré en frappant [R] [U] [N] ou [F2] [V RUN].)

?

* En reprenant l'exemple précédent, "? " est affiché. Dans l'attente d'une entrée, ce signe ne sera pas effacé par la pression de [AC]. Après la frappe de [MODE] [0], l'opération 2 est exécutée.
Pour obtenir une exécution à mi-programme, introduire le numéro de ligne désiré après l'ordre RUN puis frapper la touche [EXE].

Exemple: Commencer à la ligne 20

Opération: **RUN 20 [EXE]**

* Avec la méthode 1, il n'est pas nécessaire de désigner la zone programme à exécuter. Toutefois, il est nécessaire, pour la méthode 2, de désigner la zone de programme à exécuter. (Si la zone de programme était différente, le programme écrit dans cette zone serait exécuté.)

(2) Entrée sur clavier lors de l'exécution du programme

L'entrée au clavier durant l'exécution du programme utilise l'instruction INP et la fonction KEY. L'entrée au clavier par la fonction KEY consiste en 1 seule touche, mais si aucune touche n'est frappée, l'exécution continue.
Pour l'entrée au clavier utilisant l'instruction INP, un "? " est affiché et le programme attend l'entrée.
Lorsque la donnée a été entrée, l'exécution reprend lorsque la touche [EXE] est frappée.

Exemple: Exécuter le programme écrit en P0 dans l'exemple précédent

Opération: Exécuter le programme.

[F1] [P0 0]

?

● Dans ce programme, deux variables sont entrées. On introduit d'abord la valeur de la variable A.

47 [EXE]

?

● On introduit ensuite la valeur de la variable B.

69 [EXE]

116

[CONT]

−22

Pour l'entrée au clavier lors de l'exécution d'un programme avec instruction INP, la donnée est introduite au moyen de la touche [EXE]. Si lors de l'attente d'une entrée avec instruction INP la touche [EXE] est frappée sans introduction d'un nombre, "STOP" sera affiché et le déroulement sera à l'état d'arrêt. Dans cette situation, il est possible d'effectuer d'autres opérations, telles que calculs manuels, et de reprendre l'exécution du programme arrêté en frappant la touche [CONT].
Si dans l'état d'attente d'une entrée on désire arrêter l'exécution du programme et qu'on frappe directement [EXE] [AC], un arrêt se produit.

4-4 Edition de programmes

- L'édition de programme sert à créer un programme logiquement correct. Cette opération permet d'apporter des changements, des additions, et de récrire les lignes complètes.
- L'édition de programme est effectué à l'aide d'instructions LIST qui appellent une ligne à la fois.
- L'instruction LIST est utilisable à la fois en mode RUN et WRT, mais si le mode RUN est utilisé, le contenu du programme sera affiché. Le mode WRT permet d'éditer le programme.

(1) Affichage de la liste du programme en mode RUN

(L'affichage reste 2 s.environ)

Opération: LIST [EXE]

(LIST peut être écrit [L][I][S][T] ou [F2] [W/LIST])

```
10 INP A,B
20 V=A+B
30 W=A-B
40 PRT V,W
50 END
```

Si la liste n'est pas nécessaire depuis le début, désigner le numéro de ligne.
Pour lister depuis la ligne numéro 30:

Opération: LIST 30 [EXE]

```
30 W=A-B
40 PRT V,W
50 END
```

* Lors de l'exécution de l'instruction LIST, l'affichage se fait à la suite jusqu'à la fin. Pour l'arrêt presser la touche [STOP].
Pour reprendre l'instruction LIST après un arrêt, frapper la touche [CONT].

(2) Changement, addition et suppression en mode WRT

1. Changement

En utilisant l'instruction LIST, chaque ligne est successivement affichée à partir du numéro de ligne désigné chaque fois que la touche [EXE] est frappée. Si le numéro de ligne de départ n'est pas désigné, l'affichage commence depuis la première ligne.

a. Changement partiel

Exemple: Remplacer "+" par "*" sur la ligne 20 de l'exemple précédent.

Opération:

- Au cas où la zone de programme désignée n'est pas P0, désigner P0.

[F1] [P0/0]

```
READY P0: _12345_789 1194
```

- Appeler la ligne numéro 20 à l'aide de l'instruction LIST.

LIST 20 [EXE]

```
20 V=A+B_                1194
```

- Déplacer le curseur pour l'amener à la position du changement (soit "+").

[⇦][⇦]

```
20 V=A+B                 1194
```

* Un déplacement accéléré peut être obtenu en appuyant sur la touche de curseur ([⇦] , [⇨]) pendant plus d'une seconde.

- Effectuer la correction.

[*] [EXE]

```
30 W=A-B_                1194
```

* Ne pas manquer de frapper la touche [EXE] sinon seul l'affichage sera changé, mais le contenu du programme écrit ne changera pas.

- A présent, la ligne numéro 30 est changée. L'opération est terminée par la pression de la touche AC pour effacer l'affichage.

AC

–	1194

* La manipulation de touches quand on se trouve sur une ligne ne nécessitant pas de changement pourrait altérer son contenu. Par conséquent, ne frapper aucune touche autre que EXE et AC.

- Vérifier si le changement a été opéré à l'aide de l'instruction LIST.

MODE 0
LIST EXE

READY P0
10 INP A,B
20 V=A*B
30 W=A-B
40 PRT V,W
50 END

b. Changement d'une ligne entière

Entrer la ligne à changer. (De cette manière, la ligne entrée précédemment est remplacée par la nouvelle).

Exemple: Remplacer "W=A–B" par "W=V/2" sur la ligne numéro 30

Opération: MODE 1

READY P0: _12345_789 1194

- Ecrire la nouvelle ligne numéro 30.

30 _ W = V / 2 EXE

30 W=V/2	1194

- Vérifier le programme par LIST.

MODE 0
LIST EXE

READY P0
10 INP A,B
20 V=A*B
30 W=V/2
40 PRT V,W
50 END

2. Addition

Pour ajouter une ligne complète, utiliser un numéro de ligne compris entre les deux numéros où l'addition est désirée.

Exemple: Ajouter "U=V*2" entre la ligne 30 et la ligne 40 dans l'exemple précédent, et changer la ligne 40 en "PRT V, W, U".

Opération: MODE 1

READY P0: _12345_789 1194

- Pour placer une ligne entre 30 et 40, utiliser la ligne numéro 35.

35 _ U = V * 2 EXE

35 U=V*2	1186

* Choisir un numéro de ligne compris entre 31 et 39 pour une entrée entre les lignes 30 et 40.

- Pour changer la ligne 40, l'appeler à l'aide d'une instruction LIST et ajouter "U".

LIST 40 EXE
, U EXE
AC

40 PRT V,W	1186
50 END	1184
_	1184

● Vérifier le programme au moyen de LIST pour vérifier les additions.

MODE 0	READY P0
LIST EXE	10 INP A,B
	20 V=A*B
	30 W=V/2
	35 U=V*2
	40 PRT V,W,U
	50 END

3. Suppression

a. Suppression partielle

Exemple: Supprimer "V" de la ligne numéro dans l'exemple précédent

Opération:

MODE 1	READY P0: _12345_789 1184

● Comme méthode de suppression partielle, on appelle la ligne 40 en question à l'aide de l'instruction LIST.

LIST 40 EXE	40 PRT V,W,U_ 1184

● Déplacer le curseur et l'amener sous le "W" (à droite du rang à supprimer).

⇦ ⇦ ⇦	40 PRT V,W,U 1184

● Frapper la touche C pour supprimer "V".

C C	40 PRT W,U 1184
EXE	50 END 1186

* Le contenu du programme ne sera pas modifié si l'on ne frappe pas la touche EXE.

AC	_ 1186

* S'assurer de frapper la touche AC pour annuler la ligne numéro 50.

● Vérifier le programme au moyen de LIST pour s'assurer que la suppression a été correctement effectuée.

MODE 0	READY P0
LIST EXE	10 INP A,B
	20 V=A*B
	30 W=V/2
	35 U=V*2
	40 PRT W,U
	50 END

b. Suppression d'une ligne complète

Entrer le même numéro de ligne que celui de la ligne à supprimer et elle sera entièrement annulée.

Exemple: Supprimer la ligne 30

Opération:

MODE 1	READY P0: _12345_789 1186

● Entrer le numéro de la ligne à supprimer, c'est-à-dire 30.

30 EXE	_ 1194

● Vérifier que la suppression a été accomplie.

MODE 0	READY P0
LIST EXE	10 INP A,B
	20 V=A*B
	35 U=V*2
	40 PRT W,U
	50 END

4. Correction d'un numéro de ligne

Exemple: Le programme suivant est contenu en P2.

```
10 INP N
20 M=N↑2
30 L=N↑0.5
40 PRT M,L
50 END
```

Amener la ligne numéro 20 entre les lignes 30 et 40.

Opération: MODE 1 F1 P2/2

```
READY P2: _1_345_789 1160
```

- Appeler la ligne numéro 20 par une instruction LIST.

LIST 20 EXE

```
20 M=N↑2_ 1160
```

- Amener le curseur sous le 2 de ligne 20.

F1 HOME ⇦ ⇨ ⇨

```
20 M=N↑2 1160
```

- Changer 20 pour 35 à l'affichage et entrer.

35 EXE

```
30 L=N↑0.5 1152
```

- Pour achever le changement, frapper AC pour annuler l'instruction.

AC

```
_ 1152
```

- A l'aide de LIST vérifier le contenu du programme modifié.

MODE 0
LIST EXE

```
READY P2
10 INP N
20 M=N↑2
30 L=N↑0.5
35 M=N↑2
40 PRT M,L
50 END
```

- Maintenant, le contenu de la ligne 20 a été déplacé entre les lignes 30 et 40, mais 20 est toujours là. Il s'agit par conséquent de supprimer la ligne inutile.

MODE 1
20 EXE

```
READY P2: _1_345_789 1152
_ 1160
```

- Ceci termine le déplacement de ligne. Vérifier le résultat à l'aide de LIST.

MODE 0
LIST EXE

```
READY P2
10 INP N
30 L=N↑0.5
35 M=N↑2
40 PRT M,L
50 END
```

■ Mise au point du programme

(1) Système de mise au point du programme

Le système de mise au point de cette calculatrice se divise grossièrement en mise au point structurelle et mise au point par conversation avec le programme via l'affichage.

Systèmes de mise au point

- **I. Mise au point structurelle**
  - **a. Mise au point complète**
    Vérification de la structure logique du programme
  - **b. Mise au point partielle**
    Vérification par ligne de programme
- **II. Mise au point par conversation avec le programme via l'affichage**
  A l'aide de l'affichage, le déroulement du programme et les erreurs de grammaire de language BASIC en utilisant le système d'auto-vérification de la calculatrice.

La mise au point structurelle est exécutée au cours de la programmation.
Nous décrirons ici la méthode de mise au point conversationnelle via l'affichage.

(2) Mise au point conversationnelle

Toute erreur de rédaction du programme sera signalée au cours de l'exécution par un message d'erreur apparaissant sur l'affichage. Ces erreurs sont affichées par ligne et leur type est identifié par la convention du BASIC. Sur la base du message d'erreur affiché, la mise au point est faite par conversation en utilisant ce manuel.
En ce qui concerne la signification des messages d'erreur, voir la liste des messages d'erreur en page 79.

Exemple:

```
10 INP X
20 Y=X↑2+3*X+15
30 PRT Y
40 END
```

Y=X↑2+3X+15 a été introduit par erreur sur la ligne 20 de ce programme.

Opération: Si l'on commande l'exécution de ce programme, le signe "?" est affiché lorsqu'on utilise la ligne 10 INP.

RUN EXE	?

- Maintenant, entrer 45

45 EXE	ERR-2 IN P0-20

- Le message d'erreur indique qu'il s'est glissé une erreur de structure à la ligne 20 et que le contenu du programme doit être revu.

AC MODE 1	READY P0: _123456789
LIST 20 EXE	20 Y=X↑2+3X+15_

- Etant donné que "*" manque entre "3" et "X" sur la ligne 20, effectuer la correction par la méthode d'édition de programme.

⇦ ⇦ ⇦ ⇦ F1 INS/C	20 Y=X↑2+3_X+15
* EXE	30 PRT Y

(3) Mise au point du programme pendant son exécution

Par mise au point conversationnelle on entend la mise au point effectuée d'après les informations délivrées par la calculatrice, par voie d'un message d'erreur. Toutefois, si aucun message d'erreur n'est affiché, mais que le résultat des calculs n'est pas celui escompté, reprendre l'exécution du programme et effectuer la mise au point en vérifiant les opérations pas à pas.
Cette méthode utilise l'ordre STOP pour arrêter le déroulement du programme, et le mode TRACE pour la mise au point ligne par ligne.

• Mise au point à l'aide de l'ordre STOP.

Exemple: Le programme ci-dessous a été écrit.

```
10 Y=0
20 INP N,X
30 FOR I=1 TO N
40 Y=Y+X↑2
50 NEXT I
60 PRT Y
70 END
```

Pour découvrir la valeur de Y dans la boucle FOR•NEXT, obtenir le résultat de chaque boucle à l'aide de l'ordre STOP.

Opération: Le meilleur endroit où insérer l'ordre STOP est immédiatement après la formule de calcul, c'est à dire entre les lignes 40 et 50.

MODE 1	READY P0: 123456789
45 STOP EXE	45 STOP

• Par cette instruction, le déroulement du programme est arrêté à la fin du calcul sur la ligne 40, et on peut voir le résultat.

MODE 0	READY P0
RUN EXE	?
4 EXE	?
87 EXE	_

Le curseur clignote

• Quelle est la valeur de Y au moment de l'arrêt?

Y EXE	7569

• Lorsque le programme est redémarré, il s'arrêtera à l'instruction STOP suivante, et sortira à nouveau la valeur de Y.

CONT	_
Y EXE	15138

• En répétant cette opération, on peut se rendre compte du processus de calcul.
Cet exemple montre un programme simple; lorsqu'il s'agit d'un programme complexe, il devient très difficile de vérifier le processus de calcul à l'aide de la mise au point structurelle. Si l'on vérifie la variable comme expliqué ci-dessus à l'aide d'une instruction STOP, il est facile de comprendre et corriger les erreurs lorsqu'elles apparaissent.

• Mise au point en mode TRACE

En mode TRACE (frapper [MODE] [2]), le programme est exécuté ligne par ligne. Le déroulement s'arrête après chaque ligne pour faciliter la mise au point. A l'aide de l'instruction STOP précédente, effectuons une mise au point en mode TRACE.

Opération:

"RUN", "MODE" et "TRACE" sont omis par la suite

Désigner le mode "RUN" ..... [MODE] [0]	RUN READY P0
Désigner le mode "TRACE" ..... [MODE] [2]	RUN TRACE READY P0
RUN [EXE]	RUN STOP TRACE P0-10
Examiner le contenu du programme ... [STOP]	RUN STOP TRACE P0-10 Y=0
[CONT]	P0-20
	?
4 [EXE]	?
87 [EXE]	_
[STOP]	P0-20 INP N,X
[CONT]	P0-30
[STOP]	P0-30 FOR I=1 TO N
[CONT]	P0-40
[STOP]	P0-40 Y=Y+X↑2
Valeur de Y ..... Y [EXE]	7569
[CONT]	P0-45
[STOP]	P0-45 STOP
[CONT]	P0-50
[STOP]	P0-50 NEXT I

Reprendre ces opérations

Cette méthode de mise au point en mode TRACE est la mieux appropriée pour l'examen du déroulement global du programme et est pratique pour découvrir l'endroit où les erreurs se sont produites.

4-5 Ordres de programmation

4-5-1 Saut et boucle

• Instruction de saut

Les instructions de saut peuvent être classifiées en deux groupes: l'instruction GOTO est un saut inconditionnel. L'autre est un saut conditionnel associé à une instruction IF.

• Instruction GOTO

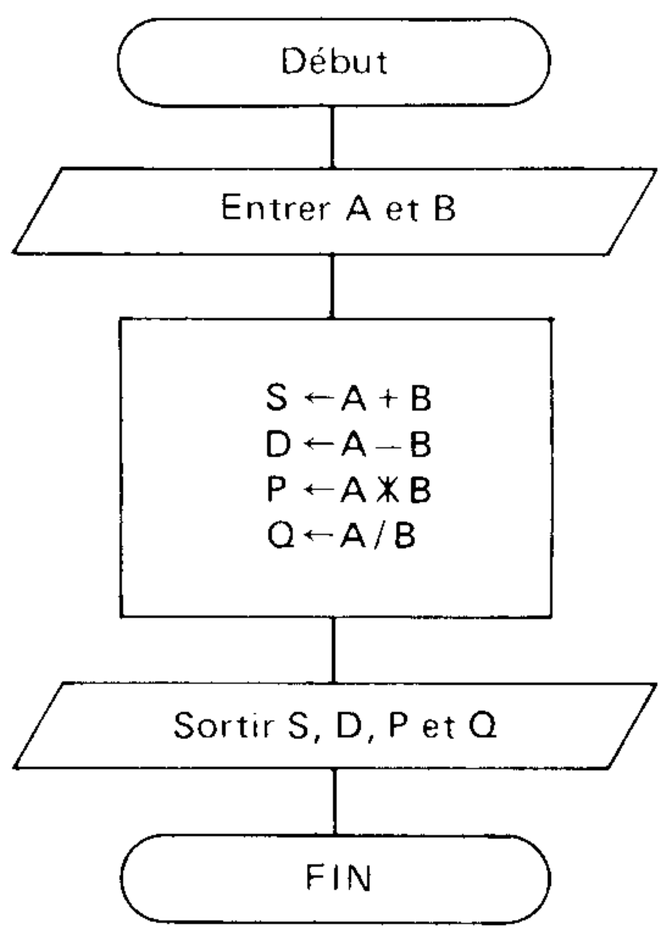
L'instruction GOTO est appelée un saut inconditionnel car elle commande sans condition l'avance du programme vers un point désigné (numéro de ligne).

Exemple 1: Ajouter une instruction GOTO au programme pour déterminer la somme, la différence, le produit et le quotient des données A et B.

Le programme élémentaire est donné ci-dessous:

```
10  INP  A,B
20  S=A+B
30  D=A-B
40  P=A*B
50  Q=A/B
60  PRT  S,D,P,Q
70  END
```

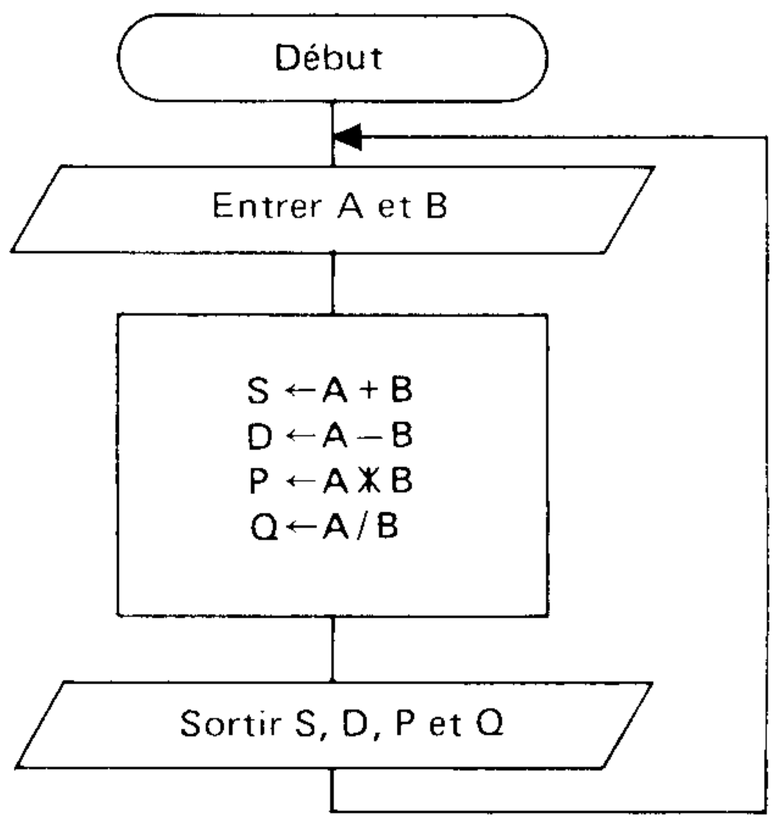
Organigramme



Chaque fois qu'il a été exécuté, le programme a besoin d'un "RUN" pour pouvoir réintroduire de nouvelles données. Si l'on ajoute une instruction GOTO à la ligne 70 au lieu du END, le déroulement reprend à l'endroit où les données sont entrées (à la ligne 10 de ce programme).

Cette instruction GOTO (GOTO 10) commande le saut inconditionnel vers la ligne 10. Le programme devient tel que ci-dessous.

```
10  INP  A,B
20  S=A+B
30  D=A-B
40  P=A*B
50  Q=A/B
60  PRT  S,D,P,Q
70  GOTO  10
```



Lorsqu'une donnée est entrée, le programme entre en attente (? est affiché), et la donnée suivante peut être introduite immédiatement après.

Exécution: Entrer les données 15 et 3, puis 903 et 43

Opération:

RUN [EXE]	?
15 [EXE]	?
3 [EXE]	18
[CONT]	12
[CONT]	45
[CONT]	5
[CONT]	?
903 [EXE]	?
43 [EXE]	946
[CONT]	860
[CONT]	38829
[CONT]	21

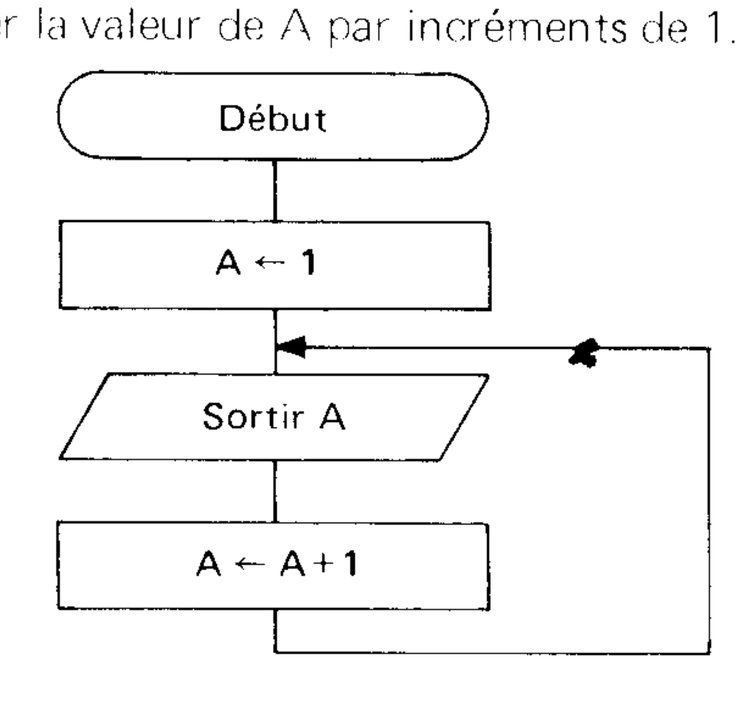
Dans ce programme, l'instruction GOTO placée à la fin commande le retour à la ligne 10 "INP A, B". De même, comme dans l'exemple ci-dessous, le saut s'effectue vers le numéro de ligne inscrit après GOTO. Toutefois, si au lieu d'un numéro de ligne on écrit # et "0" à "9", le saut s'effectue vers le numéro de programme désiré.

Exemple:

GOTO 10 (saut à la ligne numéro 10 et exécution à partir de ce point)
GOTO #5 (Saut à la zone de programme P5 et exécution du programme P5)

Exemple 2: Créer un programme pour augmenter la valeur de A par incréments de 1.

```
10  A=1
20  PRT  A
30  A=A+1
40  GOTO  20
```



Explication:

Etant donné que la valeur de A est successivement augmentée par incréments de 1, il est nécessaire d'attribuer à A une valeur initiale, soit "A=1" sur ligne 10.
Ensuite PRT (sortie) A
En ligne 30, le résultat de 1 ajouté à A est affecté à A, soit "A=A+1". Ensuite le programme effectue un saut par instruction GOTO à la ligne numéro 20, et non au début. On obtient donc GOTO 20.
De cette manière, l'instruction GOTO oblige le programme à sauter inconditionnellement au point désiré.

N.B.: Lorsqu'on utilise l'instruction GOTO, il est nécessaire de désigner correctement le numéro de ligne. Si l'on oubliait ce numéro à la suite de GOTO, une erreur se produirait.

Application de l'instruction PRT

L'instruction PRT utilisée dans ce programme modifie l'affichage en fonction du séparateur suivant les opérandes.
Par exemple, 1, S, D, P et Q sont séparés par des " , " (virgule).
L'affichage s'arrête après S. Pour obtenir D, frapper la touche [CONT]. Ainsi, si les articles sont séparés par une virgule, les résultats sont affichés un par un. Si la " , " était remplacée par " ; ", quel affichage obtiendrait-on?
Les résultats seraient affichés ainsi:

Opération:

RUN EXE	?
15 EXE	?
3 EXE	18 12 45 5
CONT	?
903 EXE	?
43 EXE	946 860 38829 21

Le " ;" (point-virgule) est utilisé lorsqu'il y a plusieurs résultats, qui sont alors affichés l'un à la suite de l'autre.
Un espace est intercalé entre chaque résultat sur l'affichage. Le signe "+" n'est pas affiché dans la colonne signe, mais est tout de même présent.

␣18␣12␣45␣5 — Colonne de signe

Si l'exemple 2 est exécuté tel quel, le résultat sera affiché comme ci-dessous.

Opération:

RUN EXE	1
CONT	2
CONT	3
CONT	4
⋮	⋮

Toutefois, si un " ;" était placé à la suite de "PRT A" en ligne 20,

20 PRT A;

Le résultat sera le suivant.

RUN EXE	1 2 3 4 5 6 7 8 9 1
	1 2 3 4 5 6 7 8 9 10
	2 3 4 5 6 7 8 9 10
	2 3 4 5 6 7 8 9 10 1
	3 4 5 6 7 8 9 10 11
	⋮

Pour obtenir des résultats continuellement, il faut donc utiliser " ;". De plus, il existe une instruction "WAIT" (attente) pour figer l'affichage pendant un certain temps après l'affichage PRT. Pour plus de détails, prière de se reporter en page 59.

- **Désignation indirecte de l'instruction GOTO**

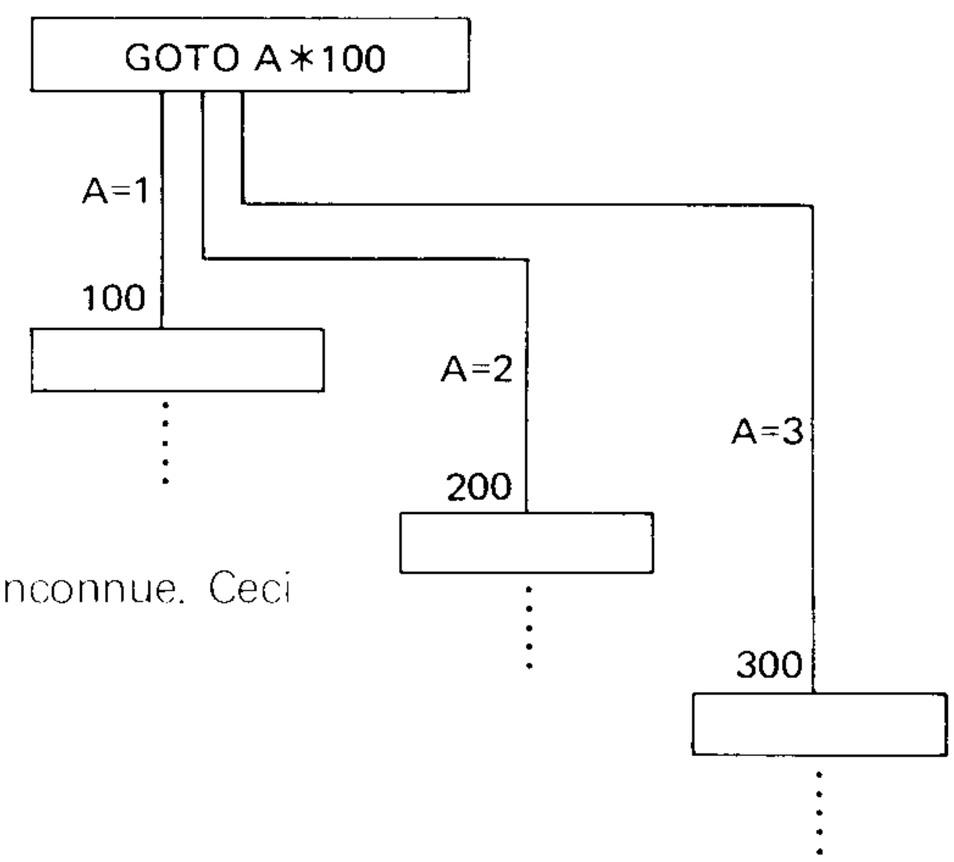
L'instruction GOTO décrite précédemment détermine directement la destination du saut. Dans une désignation indirecte, la destination est fonction de la valeur de la variable. Elle est utilisée lorsque la destination du saut ne peut être écrite au début du programme, car le mode d'opération dépend de la donnée.
La désignation indirecte se compose de:

GOTO variable ou expression numérique
GOTO # variable ou expression numérique

Ici on utilise la valeur de la variable (A, B, X, Y, etc.) ou une expression numérique (A+B, X+10, etc.) pour déterminer à quel numéro de ligne ou à quelle zone de programme l'exécution va sauter.

Exemple: GOTO A*100

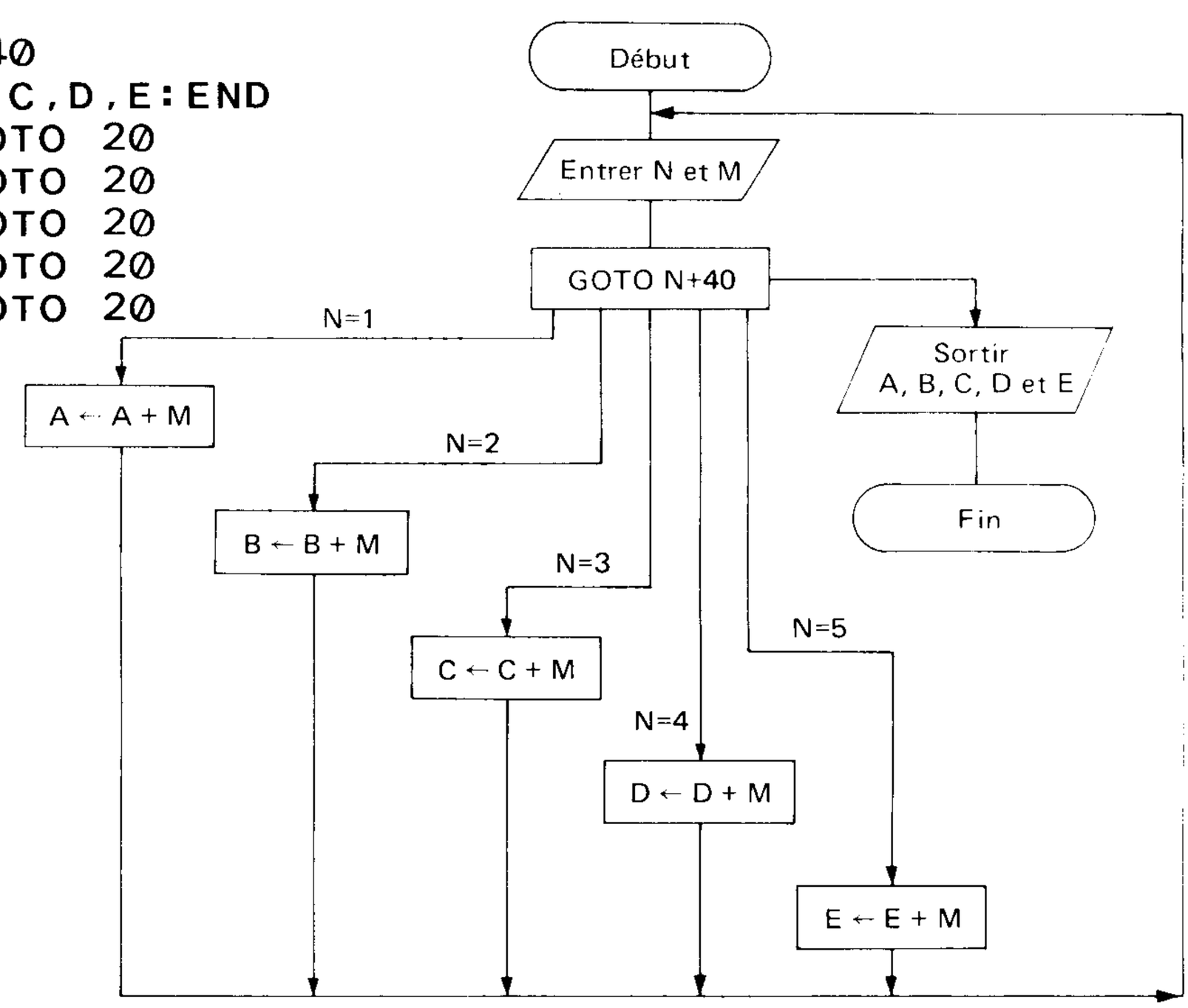
Dans ce programme, si A est 1, 2, ou 3,
Pour A=2, GOTO 100 signifie saut à la ligne numéro 100
Pour A=2, GOTO 200 signifie saut à la ligne numéro 200
Pour A=3, GOTO 300 signifie saut à la ligne numéro 300



Si A est différent de 1, 2 ou 3, la destination du saut est inconnue. Ceci cause une erreur ou un saut vers un autre numéro de ligne.

Exemple 2: Créer un programme de tri à l'aide de la désignation indirecte (en 5 classes).
Le programme prend la forme suivante:

```
10 VAC
20 INP N,M
30 GOTO N+40
40 PRT A,B,C,D,E:END
41 A=A+M:GOTO 20
42 B=B+M:GOTO 20
43 C=C+M:GOTO 20
44 D=D+M:GOTO 20
45 E=E+M:GOTO 20
```



L'instruction "VAC" sur ligne 10 est un ordre d'effacement de la mémoire de donnée (pour la rendre égale à 0). Dans ce programme, l'effacement est nécessaire pour totaliser les données entrées sur les lignes 41 à 45. La prochaine instruction permet d'entrer le numéro code (N) et les montants de revenu.
Grâce à l'instruction 30 GOTO, si le numéro code (N) est 1, saut à la ligne 41, c'est-à-dire pour la totalisation des revenus de code 1.
Pour le numéro code 2, saut à la ligne 42, De cette manière, les revenus (M) sont distribués par numéros codes de 1 à 5.

N=1 : GOTO 41 → 41	**A=A+M : GOTO 20 Exécuter**
N=2 : GOTO 42 → 42	**B=B+M : GOTO 20 Exécuter**
N=3 : GOTO 43 → 43	**C=C+M : GOTO 20 Exécuter**
N=4 : GOTO 44 → 44	**D=D+M : GOTO 20 Exécuter**
N=5 : GOTO 45 → 45	**E=E+M : GOTO 20 Exécuter**

Par conséquent, quand N est égal à 0, il saute à la ligne 40. Les résultats A, B, C, D et E triés sont affichés et terminés. Dans ce programme, N entré doit être compris entre 0 et 5. Tout autre chiffre causera une erreur.
Faisons une démonstration avec des valeurs concrètes.
Les revenus ne sont pas classés dans l'ordre. Les entrer et trier les totaux (en cinq classes).

Code	Revenu
3	2870
2	1960
5	3850
5	1250
1	2500
2	2310
3	1850
5	4370
3	5360
1	2220
2	1450
4	6120
1	3100

Code	Revenu
1	7820
2	5720
3	10080
4	6120
5	9470

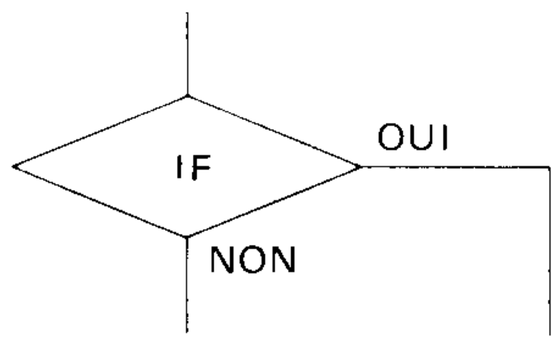
Opération:

RUN EXE	?
3 EXE	?
2870 EXE	?
2 EXE	?
1960 EXE	?
⋮	⋮
1 EXE	?
3100 EXE	?
0 EXE	?
0 EXE	7820
CONT	5720
CONT	10080
CONT	6120
CONT	9470

* Le " : " utilisé dans les lignes numéro 40 à 45 est appelé "multi-instruction" et autorise l'écriture de plusieurs instructions sur une même ligne. Si les instructions se déroulent en continu, les numéros de ligne peuvent ainsi être supprimés et on économise la mémoire. De nombreuses instructions peuvent être combinées en utilisant cette multi-instruction. Toutefois, le nombre de caractères qu'une ligne peut contenir est limité à 62.

• Instruction IF

Une instruction IF est appelée saut inconditionnel en raison de sa nature. Elle ordonne l'exécution du saut uniquement lorsque certaines conditions sont remplies, vers une certaine destination désignée.
Si on inlut cette instruction dans un organigramme, on obtient:

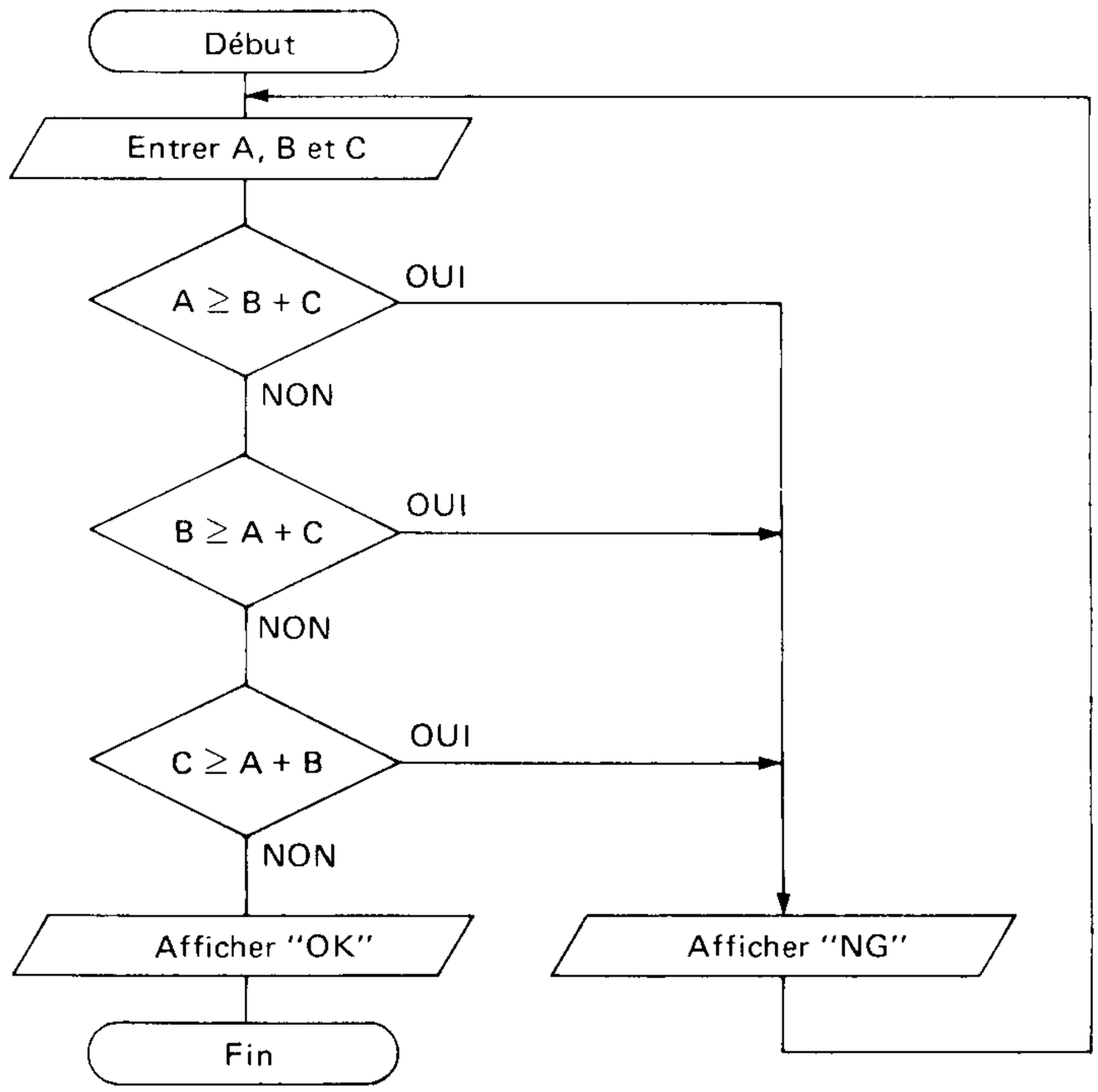


Ce qui signifie que si l'instruction IF est vérifiée, l'exécution va vers OUI. Sinon, elle continue vers NON.
En d'autres termes, l'instruction IF indique un branchement auquel une décision est faite pour déterminer la suite de l'opération en foncition du résultat.
L'instruction IF est utilisée pour terminer une boucle lorsque le nombre de données est inconnu, ou lorsque l'opération suivante est modifiée en fonction du résultat d'une opération, etc.

Exemple 1: Entrer la longueur de trois côtés et déterminer s'il est possible de former un triangle avec ceux-ci.

Le programme est donné ci-dessous:

```
10 INP A,B,C
20 IF A≥B+C THEN 70
30 IF B≥A+C THEN 70
40 IF C≥A+B THEN 70
50 PRT "OK"
60 END
70 PRT "NG"
80 GOTO 10
```



Dans ce programme, trois données sont entrées. De par les caractéristiques du triangle, la somme de la longueur de deux côtés est supérieure à la longueur du côté restant.
Chaque côté est ainsi comparé, et s'il est impossible de former un triangle, "NG" est affiché, et le programme retourne à l'état d'introduction de données. Si le triangle peut être formé, "OK" est affiché et le programme est terminé.

L'instruction IF est composée des termes suivants:

IF expression de comparaison THEN numéro de ligne (expression numérique) ou #*n* (*n* = 0 à 9)
ou
IF expression de comparaison; instruction ou affectation

L'expression de comparaison suivant "IF" compare le terme droit d'un signe d'inégalité ou d'égalité avec le terme gauche. Si OUI, on va après "THEN" ou " ;". Si NON, on va à la ligne suivante. En d'autres termes, pour la ligne 20, si A et plus grand ou égal à la somme de B et C, le triangle ne peut pas être formé, et par conséquent, "THEN 70" c'est-à-dire exécution du saut à la ligne 70. Ce "THEN" inclut la fonction de l'instruction GOTO.

Si un numéro de ligne est écrit à la suite de "THEN", le saut s'effectue au numéro désigné. Si "#" puis 0 à 9 sont écrits, le saut va à la zone de programme (P0 à P9).
Si le résultat de l'expression de comparaison est OUI, un " ;" à la place de "THEN" est utilisé si l'on désire une déclaration d'affectation au lieu d'un saut.
Dans ces comparaisons, on peut utiliser des constantes, variables, expressions numériques, constantes de caractères et variables de caractères.

A > 10	variable et constante (OUI si A est plus grand que 10)
X ≥ Y	variable et variable (OUI si X est plus grand ou égal à Y)
N = L + 3	variable et expression numérique (OÚI si N est égal à la somme de L et 3)
A$ = "XYZ"	variable de caractères et constante de caractères (OUI si la chaîne de caractères en A$ égale à "XYZ")
P$ ≠ Q$	variable de caractères et variable de caractères (OUI si la chaîne de caractères en P$ est différente de la chaîne de caractères en Q$)

* La comparaison de variables numériques et de variables de caractères est impossible.
* La comparaison de chaînes de caractères s'applique au code ASCII.

"THEN" ou " ;" peuve être utilisés de manière différente, suivant ce qui vient après.

THEN 150	(numéro de ligne)	; PRT A
THEN #9	(zone de programme)	; Z = X + Y

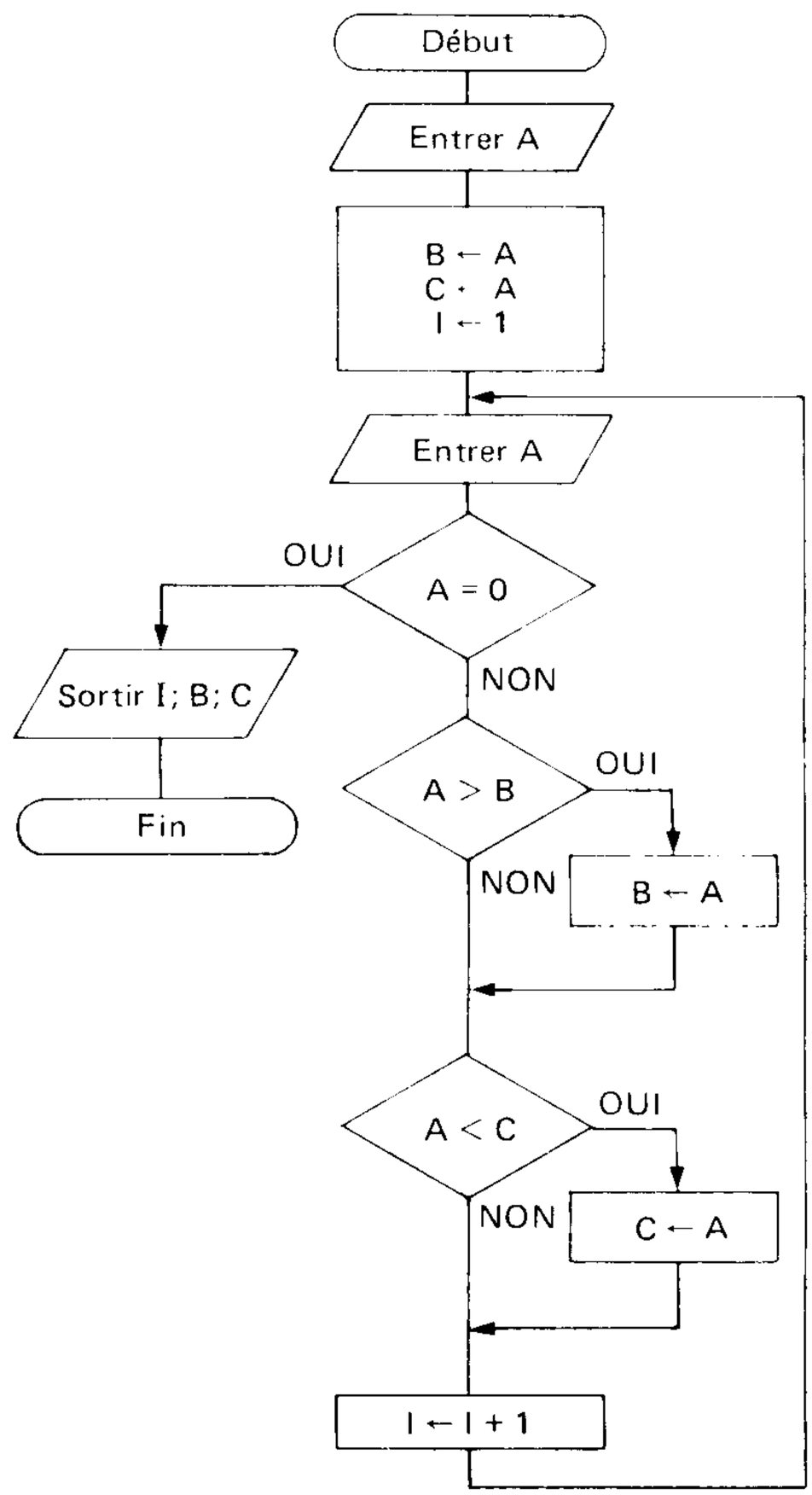
Application de l'instruction PRT
Dans le programme précédent, "OK" et "NG" sont affichés pour montrer le résultat de la comparaison. C'est ici la raison d'être de l'instruction PRT.
Si l'on écrivait PRT A, la valeur numérique de la variable A serait affichée.
Si l'on écrivait PRT "A", la lettre A serait affichée telle quelle.
En d'autres termes, l'article entre guillemets est traité comme le caractère et est affiché tel quel.

Exemple:			
	PRT A	PRT "A"	PRT "REPONSE"
	⇓	⇓	⇓
	10 (si A = 10)	A	REPONSE
	PRT X	PRT "X"	PRT "N =" ;N
	⇓	⇓	⇓
	23 (si X = 23)	X	N = ␣15 (si N = 15)

Exemple 2: Le programme pour obtenir la valeur minimum et la valeur maximum est montré ci-dessous.

```
 10 INP A
 20 B=A
 30 C=A
 40 I=1
 50 INP A
 60 IF A=0 THEN 110
 70 IF A>B;B=A
 80 IF A<C;C=A
 90 I=I+1
100 GOTO 50
110 PRT I;B;C
120 END
```

L'instruction INP de la ligne 10 sert à entrer la donnée initiale. Celle-ci est à la fois la valeur maximum et la valeur minimum. Elle est par conséquent traitée ensemble avec la donnée sur ligne 20 et ligne 30, et la valeur maximum B et la valeur minimum C lui est attribuée.
La ligne 40 utilise la variable I pour compter le nombre de données. "1" est entré comme valeur initiale. L'instruction d'entrée de la ligne 50 est utilisée pour entrer la seconde donnée et les suivantes. Ce processus est repris par l'instruction GOTO de la ligne 100. Si la donnée "0" est entrée à l'aide de l'instruction IF de la ligne 60, le déroulement du programme est terminé. Ainsi, si "0" est entré à l'aide de l'instruction INP de la ligne 50, le déroulement saute à l'instruction PRT pour sortir le résultat sur la ligne 110.
Les instructions IF des lignes 70 et 80 déterminent si la donnée d'entrée est oui ou non plus grande que la valeur maximum de la donnée déjà entrée, ou moins grande que la valeur minimum, et si oui effectue la substitution.
Lorsque toutes les données sont entrées, le nombre de données et les valeurs maximum et minimum sont affichées sur la ligne 110.



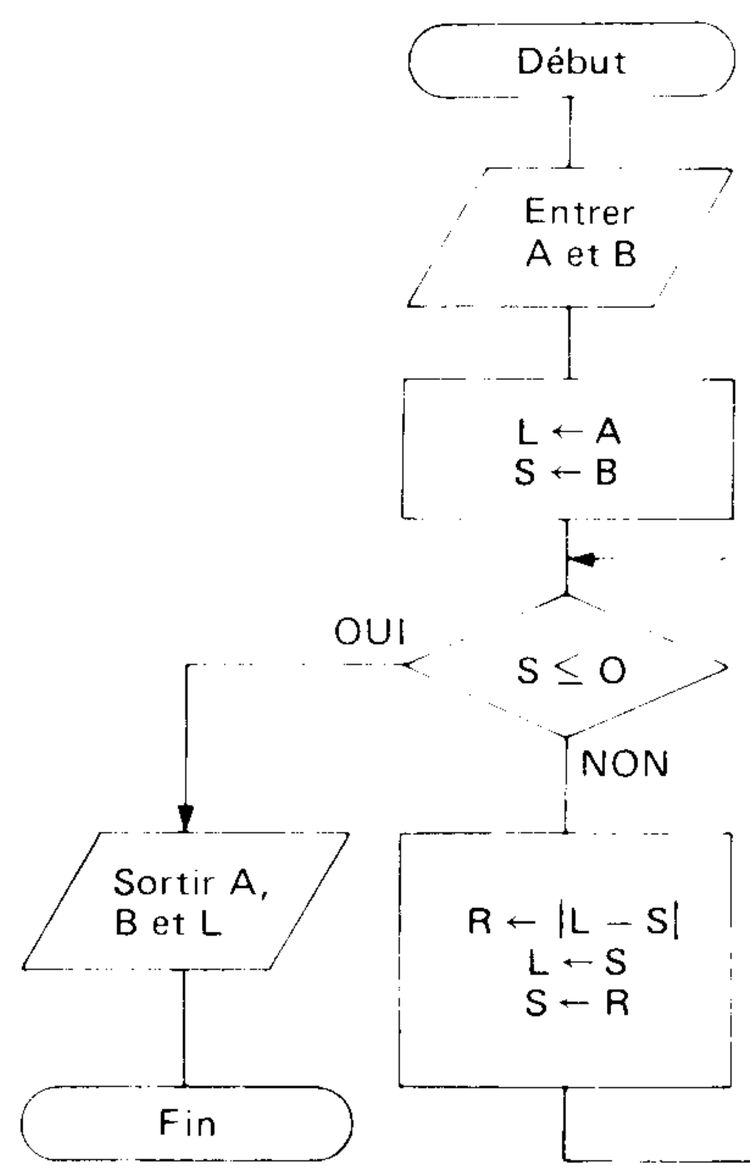
• Diverses instructions GOTO et IF

Examinons maintenant quelques exemples d'application des instructions GOTO et IF.

Exemple 1: Programme pour déterminer la plus grande commune mesure selon l'algorithme d'Euclide.

```
 10  INP  A,B
 20  L=A
 30  S=B
 40  IF  S≤0  THEN  90
 50  R=ABS(L-S)
 60  L=S
 70  S=R
 80  GOTO  40
 90  PRT  A;B;L
100  END
```

Organigramme



Selon l'algorithme d'Euclide, le second bloc de données est soustrait du premier bloc. Ensuite, ce résultat (la différence) est soustrait du second bloc. Soustraire ensuite cette différence du résultat précédent, et répéter ceci jusqu'à ce que la différence soit nulle. Quand cela se produit, la différence précédente est la plus grande commune mesure. La plus grande mesure de 63 et 36 est montrée ci-dessous.

Variable						
L	63	36	27	9	18	$\boxed{9}$ ←Plus grande commune mesure
S	36	27	9	18	9	9
R (différence)	27	9	18	9	9	$\boxed{0}$ ←Quand 0

Dans ce programme, les variables pour obtenir les différences sont désignées par L et S. La différence est affectée provisoirement à R, puis L et S sont respectivement remplacés par S et R et la même opération est reprise sans fin (boucle).
L'instruction IF de la ligne 40 détermine si S, c'est-à-dire la différence après substitution, est zéro ou non. Si c'est zéro, L représente la plus grande commune mesure, et l'affichage montre les données A et B ainsi que leur commune mesure. Si ce n'est pas zéro, la différence est continuellement prise selon l'instruction GOTO de la ligne 80.
L'instruction IF est utilisée pour décider quelle instruction (opération) va être exécutée sur la base du résultat du calcul.
Les opérations de substitution des lignes 50 et 70 méritent qu'on leur accorde une attention particulière. Les données A et B sont à l'origine attribuées à L et S et leur différence est provisoirement affectée à R. La différence entre L et S est à nouveau prise, L et S étant remplacés respectivement par S et R. Ce processus se répète jusqu'à qu'une différence nulle soit obtenue.

La relation entre L, S et R est montrée ci-dessous.

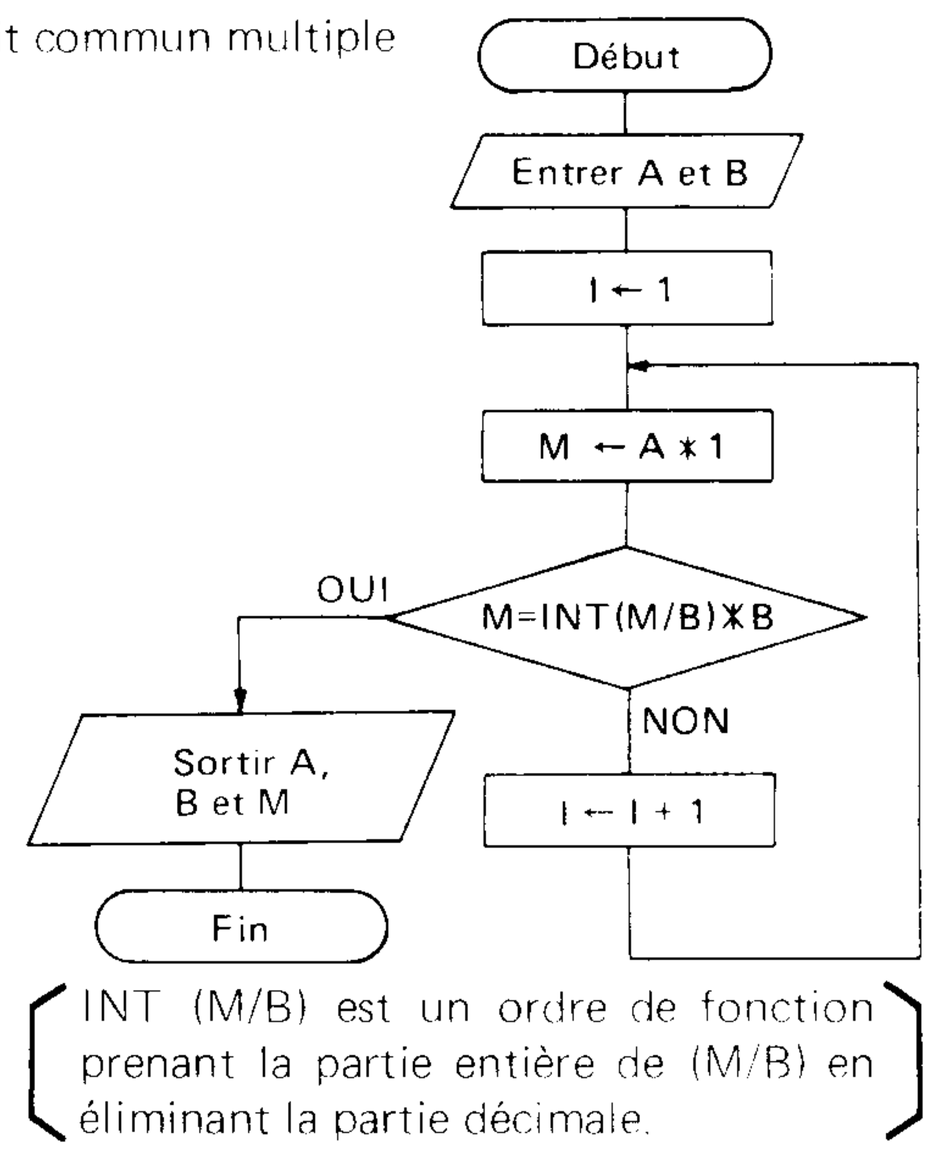
	L	S	R	
Initial	63	36	0	
Soustraction (1ère fois)	63	36	27	R = ABS (L−S)
Substitution (1ère fois)	36	27	27	L = S , S=R
Soustraction (2è fois)	36	27	9	R = ABS (L−S)
Substitution (2è fois)	27	9	9	L = S , S=R
Soustraction (3è fois)	27	9	18	R = ABS (L−S)
Substitution (3è fois)	9	18	18	L = S , S=R
⋮		⋮		⋮
Soustraction (6è fois)	9	9	0	R = ABS (L−S)
Substitution (6è fois)	$\boxed{9}$	0	0	L = S , S=R

↑
Plus grande commune mesure

N.B.: Si l'ordre est modifié dans le processus ci-dessus, les opérations de substitution ci-dessus n'auront plus de signification.

Exemple 2: Créer un programme pour déterminer le plus petit commun multiple

```
10 INP A,B
20 I=1
30 M=A*I
40 IF M=INT(M/B)*B THEN 70
50 I=I+1
60 GOTO 30
70 PRT A,B,M
80 END
```



INT (M/B) est un ordre de fonction prenant la partie entière de (M/B) en éliminant la partie décimale.

Dans ce programme, les données A et B sont d'abord entrées et la valeur initiale de A est multipliée par un, deux, trois et ainsi de suite jusqu'à ce que le résultat de cette multiplication devienne égal au plus petit multiple de la valeur B, tel que vérifié par l'instruction IF.
L'instruction IF de la ligne 40 compare la variable "M" avec l'expression numérique "INT (M/B)*B". En d'autres termes, la variable "M" est comparée au résultat de l'expression numérique "INT (M/B)*B". Cette méthode est identique à:

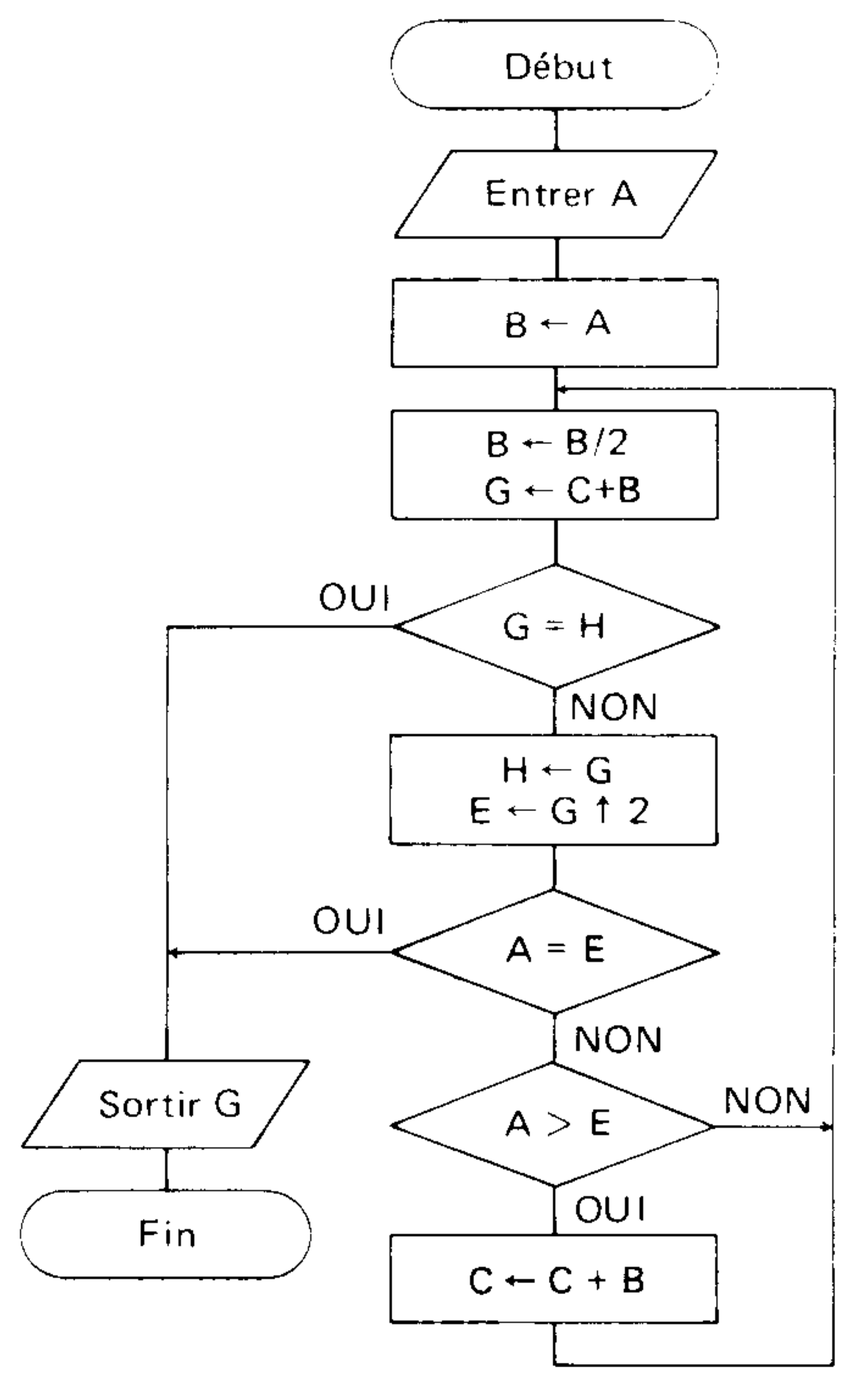
```
N=INT(M/B)*B
IF M=N THEN 70
```

Cependant, la méthode que nous utilisons économise des lignes et de la mémoire.

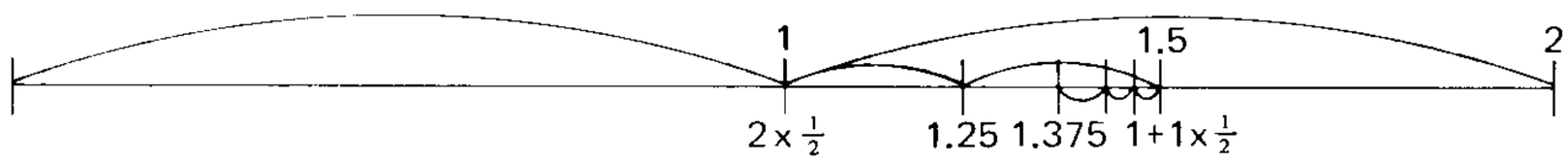
Ensuite, le multiplicateur I est augmenté d'une unité, et par l'instruction GOTO de la ligne 60, est à nouveau multiplié et soumis à la décision.
Les variables utilisées ici sont A, B, M et I. A et B sont deux données. M est un multiple de A, et il sera plus facile de l'expliquer par la suite.
Le multiplicateur I est incrémenté de 1 pour produire un nouveau multiple. Il est souvent utilisé en tant que variable qui est incrémentée chaque fois qu'une boucle est accomplie.

Exemple 3: Créer un programme pour déterminer la racine carrée selon la méthode dichotomique. Le programme est le suivant:

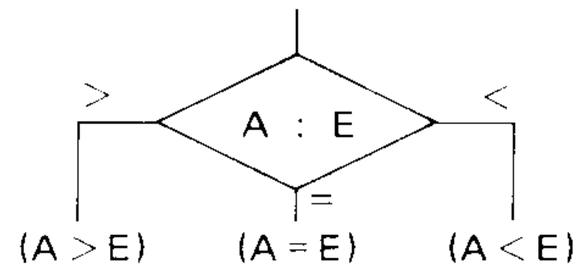
```
10 VAC
20 INP A
30 B=A
40 B=B/2
50 G=C+B
60 IF G=H THEN 120
70 H=G
80 E=G↑2
90 IF A=E THEN 120
100 IF A>E;C=C+B
110 GOTO 40
120 PRT G
130 END
```



Selon la dichotomie, on prend d'abord la moitié de la donnée A et on détermine si le carré de cette moitié est égal à la valeur de A ou non. Si ce n'est pas le cas, on prend la moitié de la moitié et on l'élève au carré jusqu'à obtenir la valeur approximative de A. Le processus de calcul est le suivant.



Dans ce programme, l'instruction "VAC" de la ligne 10 sert à effacer la mémoire. Ensuite, on introduit la donnée A. Cette donnée est utilisée par la suite pour la comparaison de l'instruction IF. Il n'est pas possible de faire des changements directs. Ceux-ci sont opérés par affectation à B.
L'instruction IF est utilisée ici pour déterminer la valeur approximative à la ligne 60.
Le nombre limite de chiffres de calcul est déterminé. G et H sont utilisés pour déterminer le point auquel la limite supérieure est atteinte, et l'exécution se termine quand le résultat est égal au précédent.
L'instruction IF de la ligne 90 est pareille à celle de l'exemple 2. Si la racine carrée est divisible, le carré du résultat est déterminé, la donnée A sera égale et le résultat sera affiché.
L'instruction IF de la ligne 100 détermine laquelle des deux parties également divisées se rapproche de la solution (valeur approximative).
Dans ce programme, les instructions IF sont utilisées de manière différente à précédemment, mais la fonction de chaque instruction IF reste pareille.
Le programme est quelque peu compliqué, car plusieurs instructions IF en combinaison sont nécessaires. L'instruction IF des lignes 90 et 100 compare A et E. L'organigramme de cette opération est montré ci-dessous:



Toutefois, le langage BASIC ne prévoit pas d'instruction IF à trois branches. On doit par conséquent utiliser une combinaison de deux instructions IF.
Suivant leur emplacement dans le programme, les instructions IF ont des effets différents, comme on peut le voir dans les exemples 1 et 2.
Dans l'exemple 1, une instruction IF est utilisée au début de la boucle pour causer un saut à END avant l'exécution de l'opération.
Dans l'exemple 2, les opérations sont exécutées puis la comparaison de décision est effectuée. Ces deux méthodes montrent que, suivant la donnée, l'instruction IF peut être placée au début d'une boucle, comme dans l'exemple 1, ou à la fin d'une opération, comme dans l'exemple 2.

■ Instruction FOR·NEXT

L'instruction FOR·NEXT est utilisée lorsque le nombre de répétitions d'une même boucle est connu.

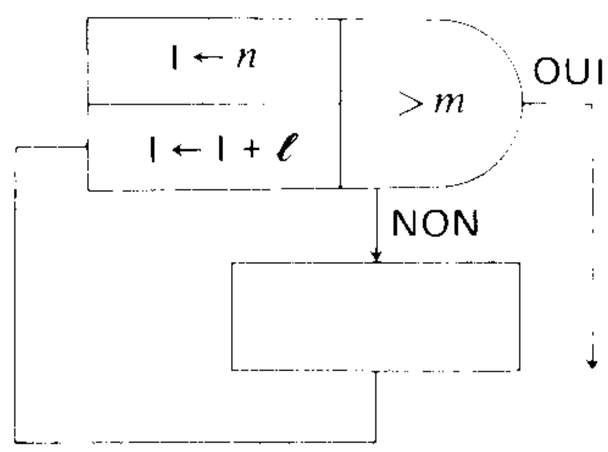
● Fonction de l'instruction FOR·NEXT

L'instruction FOR·NEXT est composée comme ci-dessous:

FOR variable = *n* TO *m* STEP *ℓ*

valeur initiale — *n*
valeur finale — *m*
Incrément ⇒ variable ou expression numérique — *ℓ*

Organigramme



Ordre d'opération d'expression numérique, etc.

NEXT variable

Cette instruction commande la répétition de l'opération spécifiée entre "FOR" et "NEXT" avec une variable passant de n à m incrémentée successivement de l Quand la variable atteint m, le déroulement du programme passe à "NEXT".
Par exemple, la programme ci-dessous montre comment exécuter une action donnée lorsque la variable **I** passe de 1 à 10 par incréments de 2.

```
FOR I = 1 TO 10 STEP 2
.......................
NEXT I
```

Ce STEP peut être omis si l'incrément de la variable est de 1.

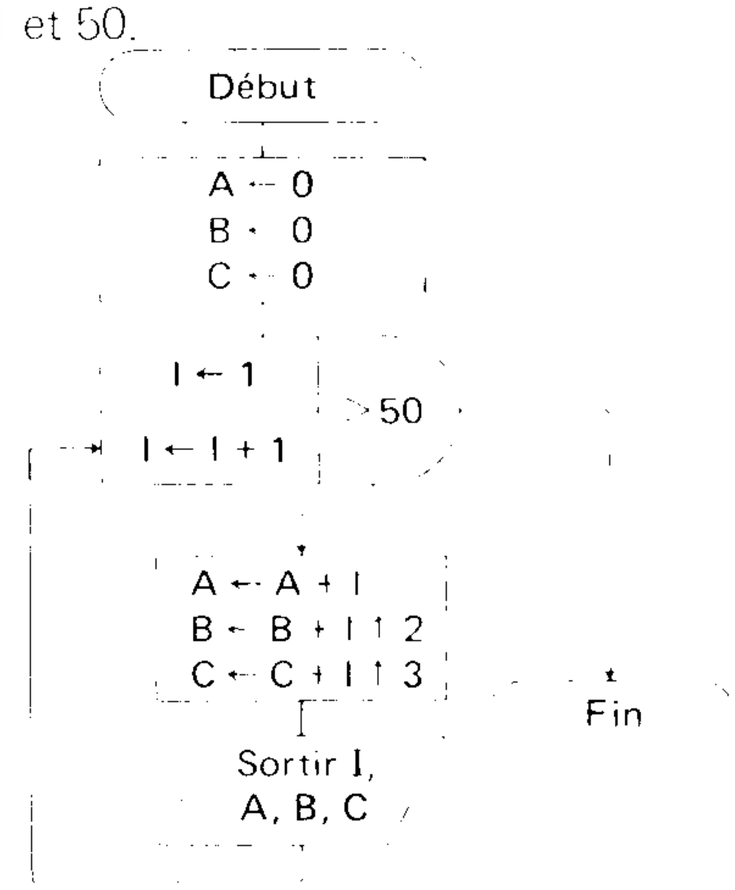
FOR I = 1 TO 10 STEP 1 est équivalent à
FOR I = 1 TO 10

* L'instruction FOR•NEXT n'augmente la variable que de la valeur de l'incrément spécifié, et ne peut pas la décrémenter. Si la variable doit être décrémentée de 10 à 1, il faut alors l'écrire sous forme de nombre négatif, comme "STEP 1".

Exemple 1: Créer un tableau de $\sum_{i=1}^{n} i$, $\sum_{i=1}^{n} i^2$, $\sum_{i=1}^{n} i^3$, avec n compris entre 1 et 50.

Ce programme est montré ci-dessous.

```
10 VAC
20 FOR I=1 TO 50 STEP 1
30 A=A+I
40 B=B+I↑2
50 C=C+I↑3
60 PRT I,A,B,C
70 NEXT I
80 END
```



Cet exemple peut uniquement être réalisé au moyen d'une instruction FOR•NEXT. Les totaux de i, i^2 et i^3 ($\sum_{i=1}^{n}$) sont affichés tandis que la variable n est augmentée de 1 à 50 par incréments de un.
De ce fait, l'opération spécifiée entre "FOR . . . . " de la ligne 20 et "NEXT . . . . " de la ligne 70 est répétée 50 fois tandis que I est augmenté de 1 à 50 par incréments de 1.
Cet exemple peut être réalisé avec une instrucion IF.

```
10 VAC
20 I=1
30 A=A+I
40 B=B+I↑2
50 C=C+I↑3
60 PRT I,A,B,C
70 I=I+1
80 IF I=51;END
90 GOTO 30
```

Nous pourrons clairement comprendre la fonction de l'instruction FOR•NEXT si nous comparons l'exemple utilisant celle-ci et l'exemple avec l'instruction IF.

En d'autres termes, l'instruction FOR•NEXT combine les fonctions de test d'une instruction IF et d'une instruction GOTO, tout en assurant une fonction d'incrémentation.

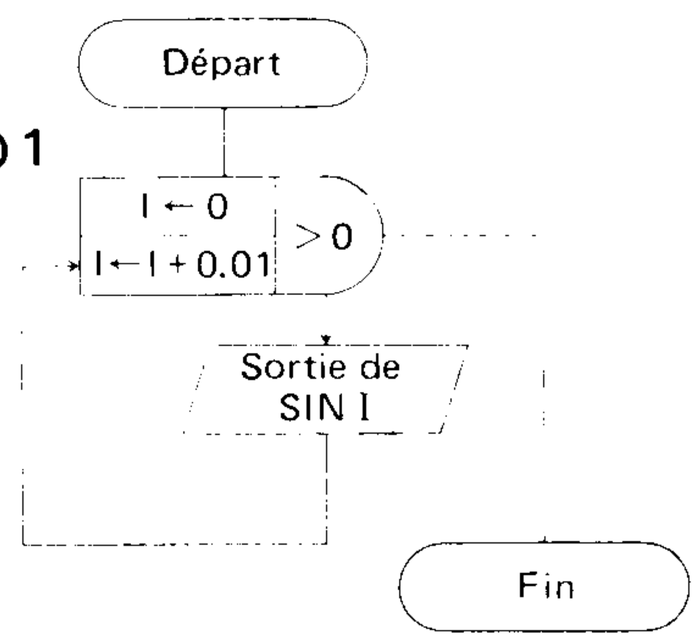
```
20 FOR I=1 TO 50 STEP 1 ⇨ { 20 I=1
                          { 70 I=I+1
80 NEXT I ⇨ { 80 IF I=51;END
            { 90 GOTO 30
```

L'instruction FOR•NEXT possède à la fois des fonctions d'incrémentation et de test, de sorte qu'elle est d'un usage très souple et pratique lorsque le nombre de répétition d'une boucle est connu.

Exemple 2: Créer un programme pour produire une table des fonctions sinus de 0 à 1 par incréments de 0,01.

Ce programme est le suivant:

```
10 MODE 4
20 FOR I=0 TO 1 STEP 0.01
30 PRT SIN I
40 NEXT I
50 END
```



Cet exemple détermine les fonctions sinus de 0 à 1 par incréments de 0,01. On peut utiliser l'instruction FOR•NEXT en posant son incrément égal à 0,01. Cette incrémentation est possible car les circuits de cette calculatrice utilisent un système décimal. SIN I est obtenu et entré en utilisant l'instruction de la ligne 30. "MODE 4" de la ligne 10 spécifie la mesure angulaire "degrés". MODE 5 et MODE 6 spécifient respectivement "radians" et "grades".

• Diverses boucles

Exemple: Déterminer le *n* ième nombre d'une série de Fibonacci.

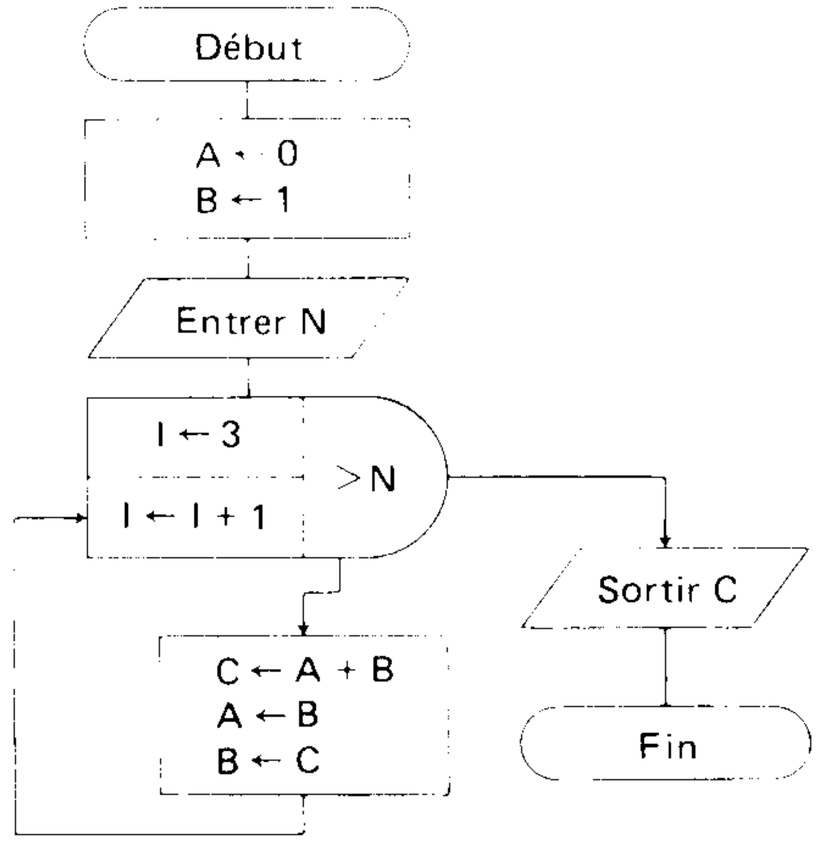
Une série de Fibonacci est une série d'entiers qui sont chacun égaux à la somme des deux précédents. En d'autres termes, la somme du premier et second nombres est égale au troisième, la somme des deuxième et troisième est égal au quatrième, et ainsi de suite.

0 , 1 , 1 , 2 , 3 , 5 , 8 , 13, 21, 34,

0 + 1 1 + 1 1 + 2 2 + 3

Ce programme est le suivant:

```
10 A=0:B=1
20 INP N
30 FOR I=3 TO N
40 C=A+B
50 A=B
60 B=C
70 NEXT I
80 PRT C
90 END
```



A, B et C représentent les variables de la série. A est la valeur du nombre précédent, B celle du suivant et C est le total (le nombre suivant B).
Une valeur initiale 0 est attribuée à A et une valeur initiale de 1 à B. L'opération commence donc au troisième rang.
L'instruction FOR•NEXT commence la boucle à 3 et la continue jusqu'au point désiré (*n* ième valeur). La valeur initiale de l'instruction FOR•NEXT n'est pas forcément 1.

Les opérations de substitution des lignes 40 à 60 exigent une attention particulière. Cet ordre ne peut être inversé.
Lorsque la somme de A et de B est introduite en C, B est placé en A et C dans B.
Si cet ordre d'opération n'est pas suivi, la substitution ne s'effectue pas correctement.
Ce mode d'utilisation de l'instruction FOR•NEXT diffère des exemples précédents. Par consqeuent, la variable N est utilisé car la valeur terminale est modifiée sur la base des entrées suivantes.

Cet exemple peut être réalisé avec l'instruction IF.
Un modèle de programme utilisant l'instruction IF apparait ci-dessous.

```
10 A=0: B=1: I=3
20 INP N
30 C=A+B
40 A=B
50 B=C
60 IF I≠N; I=I+1: GOTO 30
70 PRT C
80 END
```

Dans ce programme, l'instruction IF de la ligne 60 incrémente la variable I tout en remplissant la fonction de l'instruction GOTO. Les valeurs initiales sont fixées à l'aide d'une multi-instruction sur la ligne 10.

• Emboîtement

Les boucles FOR•NEXT peuvent être emboîtées à jusqu'à 8 niveaux.

```
FOR ....................
FOR ....................
FOR ....................
FOR ....................
    ....................
NEXT ...................
NEXT ...................
NEXT ...................
NEXT ...................
```

Voila un exemple d'emboîtement à 4 niveaux.
Une instruction FOR•NEXT est insérée à l'intérieur d'une autre instruction FOR•NEXT, de la manière indiquée.

Lors d'un tel emboîtement, il faut prendre garde à l'instruction NEXT correspondant à l'instruction FOR, et aux variables concernées.

De même, l'emboîtement doit être réalisé comme ci-dessus, c'est-à-dire que les boucles doivent être entièrement inclues dans d'autres. Le recouvrement de boucles FOR•NEXT, c'est-à-dire qu'une partie d'une boucle sort d'une autre, n'est pas autorisé. L'exemple ci-dessous montre un emboîtement erroné.

Ce genre de boucle FOR•NEXT n'est pas utilisable.

```
FOR ....................
FOR ....................
    ....................
NEXT ...................
    ....................
NEXT ...................
```

De plus, le déroulement peut sortir de la boucle FOR•NEXT, mais ne peut y entrer.

```
X
FOR ....................
FOR ....................
    ....................
IF ...... THEN..........
NEXT ...................
NEXT ...................
O
```

4-5-2 Rangées

Les rangées tombent dans deux catégories: les rangées unidimensionnelles (listes) et bidimensionnelles (tableaux) suivant la disposition des éléments; elle peuvent être utilisées toutes deux.
Une rangée unidimensionnelle est représentée par A(i) ou B(j), (i) et (j) étant les indices inférieurs. Une rangée bidimensionnelle est représentée par $x(i, j)$ ou $y(n, m)$, (i, j) et (n, m) étant des indices inférieurs à deux variables.

On trouvera des exemples de ces deux types ci-dessous.

Rangée unidimensionnelle (liste)

i	A
0	A0
1	A1
2	A2
3	A3
4	A4
5	A5
6	A6
7	A7
8	A8
9	A9
10	B0
⋮	

Rangée bidimensionnelle (tableau)

i \ j	0	1	2	3	4	5	6	7	8	9
0	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9
1	B0	B1	B2	····	····	····	····	····	B8	B9
2	C0	C1	····	····	····	····	····	····	····	C9
3	D0	····	····	····	····	····	····	····	····	D9
4										
5										
6										
7										
8										
9										
10										
⋮										

Pour les listes et rangées, cette calculatrice utilise les mémoires A0 à A9, à T9, dont la zone normale a été étendue.
La rangée unidimensionnelle (liste) est représentée par A(0) à A(199).
La rangée bidimensionnelle (tableau) est représentée par A(0, 0) à A(19, 9).
En d'autres termes, A(0) et A0 occupent la même mémoire, A(10) et B0 la même mémoire, A(1, 0) et B0 la même mémoire, et A(18, 7) et S7 la même mémoire.
De plus, seul A peut être utilisé comme variable de la rangée.

* Lorsqu'on utilise les rangées, il est nécessaire d'étendre le nombre de mémoire suivant leur ampleur. Cette opération est effectuée en manuel.

DEFM n (n est compris entre 0 et 19, et doit être 1 ou supérieur lorsqu'utilisé dans une rangée)
[EXE]
(Voir page 12 pour plus de détails)

Cette opération doit être effectuée. Si le nombre de mémoire est inférieur au nombre d'éléments de la rangée, une erreur se produit lorsque l'opération est exécutée.

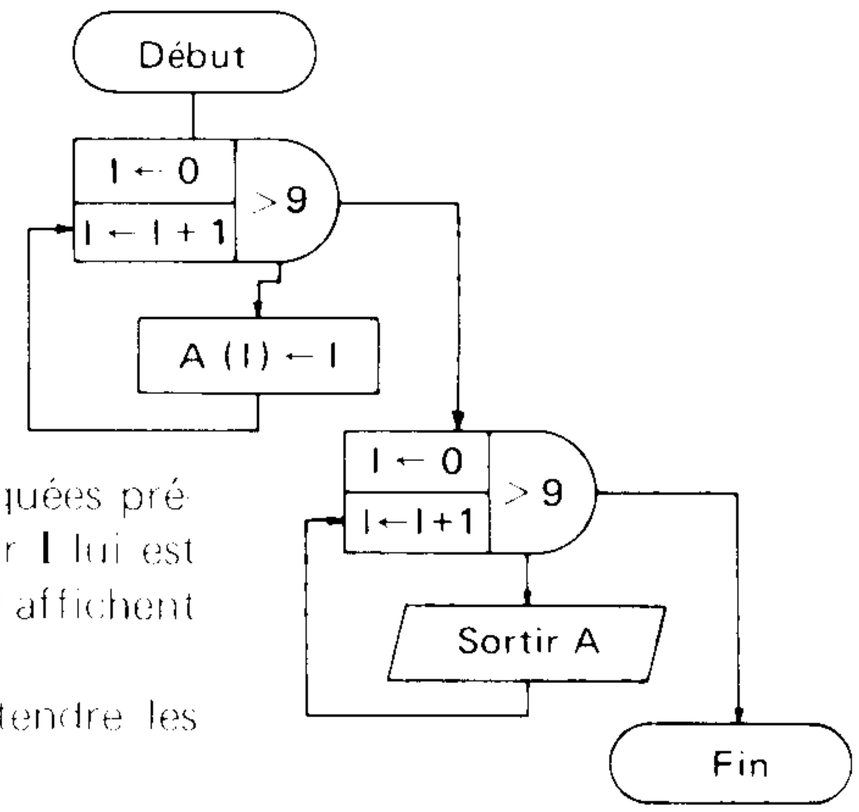
■ Rangées unidimensionnelles (listes)

Une rangée unidimensionnelle est une disposition de 0 à 199 éléments avec un indice inférieur.

Exemple: A(0), A(1), ... A (199)

Exemple 1: Créer un programme pour afficher la valeur de *i* dans la liste A(*j*) alors qu'il change de 0 à 9.

Organigramme



Ce programme est le suivant:

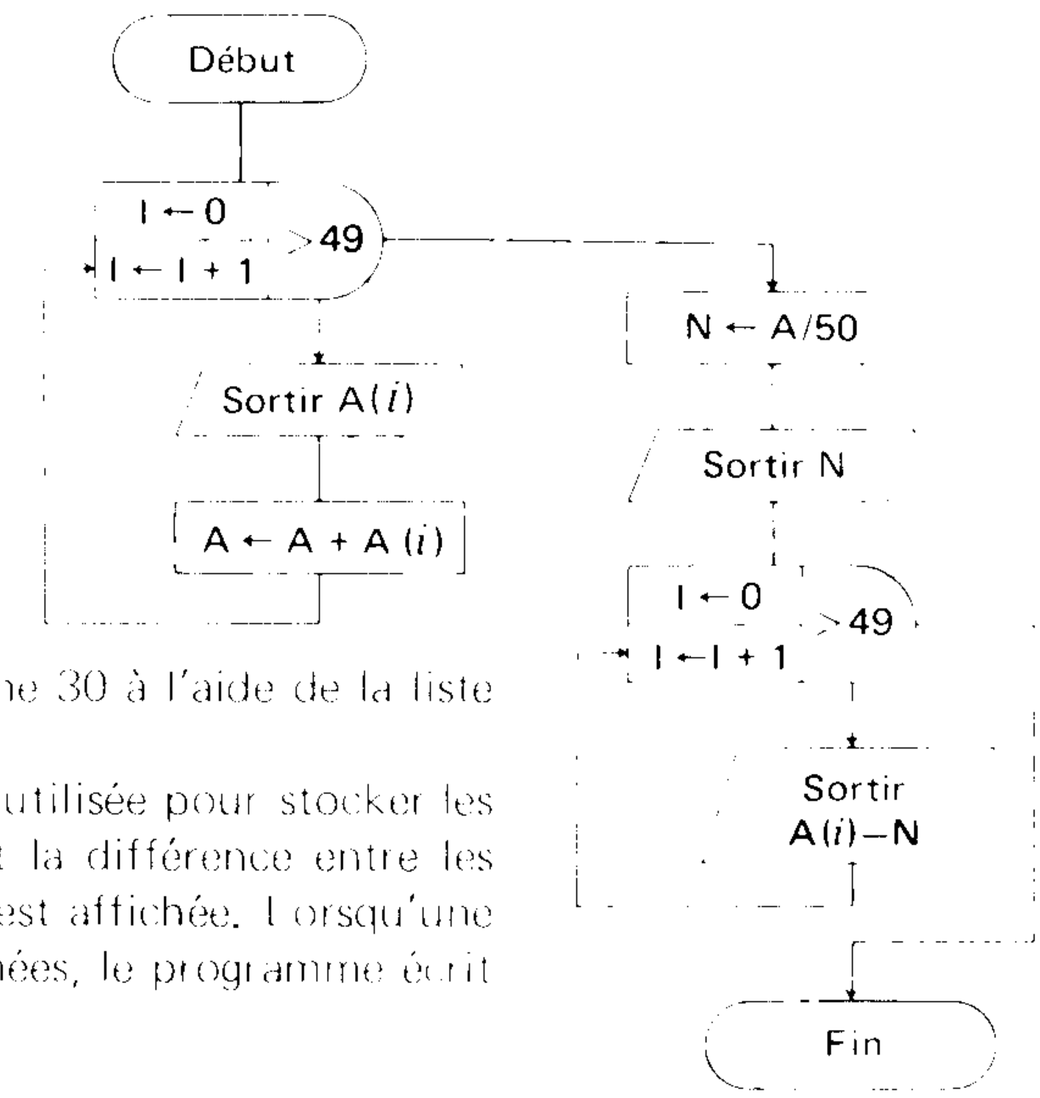
```
10 FOR I=0 TO 9
20 A(I)=I
30 NEXT I
40 FOR I=0 TO 9
50 PRT A(I)
60 NEXT I
70 END
```

Les instructions FOR•NEXT des lignes 10 à 30 ont été expliquées précédemment. Toutefois, A(I) est utilisé en ligne 20 et une valeur I lui est attribuée. De même, l'instruction FOR•NEXT des lignes 40 à 60 affichent la valeur de la liste en utilisant l'instruction PRT de la ligne 50.
De plus, avant d'exécuter ce programme, il est nécessaire d'étendre les mémoires de 20 unités.

DEFM 2 EXE

De cette manière, la liste stocke les données en tant que variables séparées simplement en changeant les éléments, sans changer de désignation de variable. C'est donc une fonction pratique lorsqu'utilisée en association avec l'instruction FOR•NEXT.

Exemple 2: Créer un programme pour afficher la différence entre la moyenne et les résultats individuels en introduisant les résultats d'examen de 50 élèves.




```
 10 A=0
 20 FOR I=0 TO 49
 30 INP A(I)
 40 A=A+A(I)
 50 NEXT I
 60 N=A/50
 70 PRT N
 80 FOR I=0 TO 49
 90 PRT A(I)-N
100 NEXT I
110 END
```

Dans ce programme les résultats sont introduits en ligne 30 à l'aide de la liste A(I).
La somme totale est obtenue en ligne 40. Une liste est utilisée pour stocker les entrées (séparées). Ensuite, la moyenne est calculée et la différence entre les données stockées (résultats individuels) et la moyenne est affichée. Lorsqu'une liste est déclarée pour entrer un grand nombre de données, le programme écrit sera très court.

Si l'on utilisait pas de liste, on obtiendrait ceci:

```
A(0) ←——→ A
A(1) ←——→ B
A(2) ←——→ C
  ⋮        ⋮
```

et la déclaration d'entrée serait:

```
INP A,B,C, . . . . . . . . . . . .
```

La séparation des données nécessiterait beaucoup de temps.

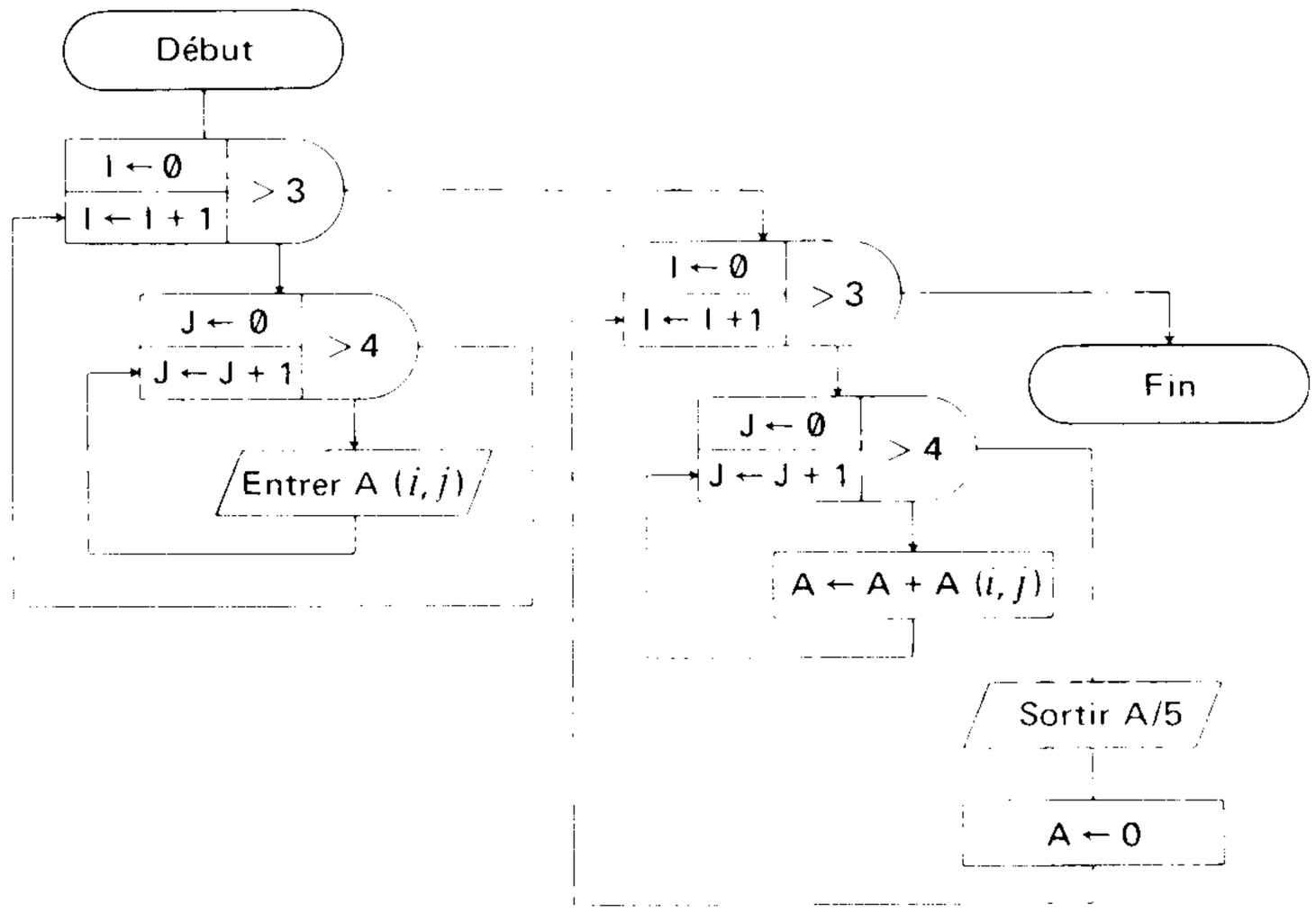
■ Rangées bidimensionnelles (tableaux)

Une rangée bidimensionnelle (tableau) est une combinaison de plusieurs rangées unidimensionnelles (listes). Son format est A(i, j), où A est la désignation du tableau, i et j étant ses éléments. Les variables utilisées sont semblables à celles utilisées dans la liste. Les éléments sont i et j, des expressions numériques.

Exemple 1: Créer un programme pour obtenir la différence des résultats de chaque élève, en utilisant la somme des résultats de 5 étudiants en langues, mathématiques, physique et sociologie.

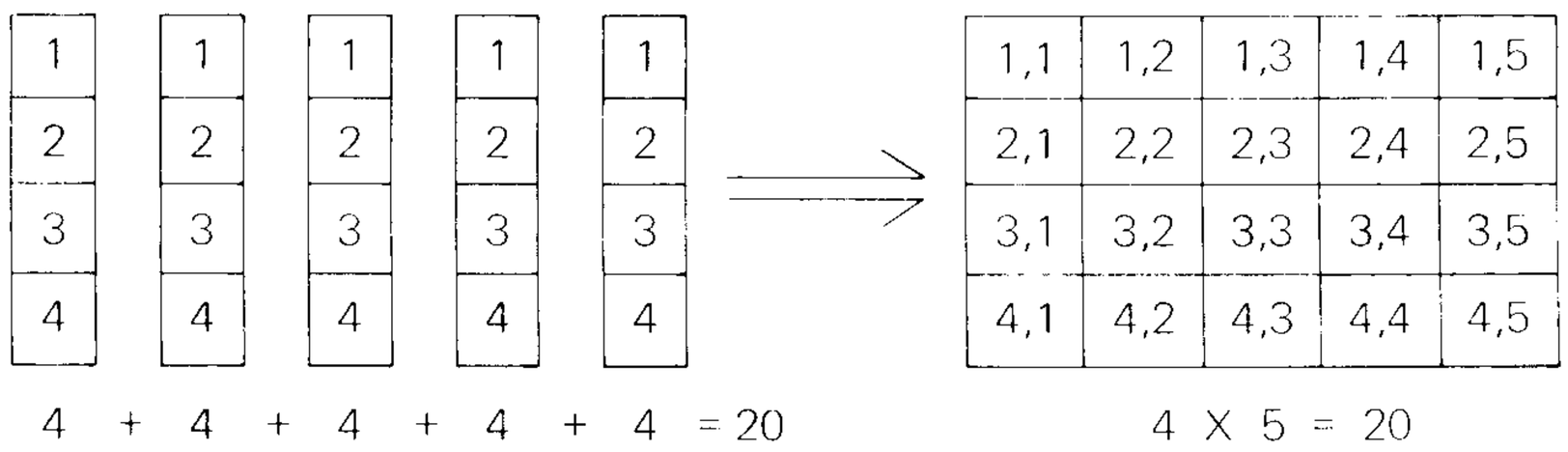
Le programme est le suivant:

```
 10 VAC
 20 FOR I=0 TO 3
 30 FOR J=0 TO 4
 40 INP A(I,J)
 50 NEXT J
 60 NEXT I
 70 FOR I=1 TO 4
 80 FOR J=1 TO 5
 90 A=A+A(I,J)
100 NEXT J
110 PRT J
120 A=0
130 NEXT I
140 END
```



La partie entrée et la partie sortie peuvent être séparées en deux. L'instruction FOR•NEXT est utilisée à deux reprises en lignes 20 à 60 et exécute l'entrée dans le tableau. L'instruction FOR•NEXT est utilisée à nouveau deux fois en lignes 70 à 130 et la moyenne est sortie à cet endroit.

En fait, la rangée bidimensionnelle (tableau) est une combinaison de deux rangées unidimensionnelles (listes).



Les deux variables qui forment les éléments du tableau peuvent être changées soit simultanément soit isolément. C'est pourquoi il est pratique d'utiliser l'instruction FOR•NEXT à deux reprises. Dans ce programme, on utilise i et j. D'abord, déterminer i puis changer j. Ensuite, i est changé puis j est changé à nouveau. Cette opération est reprise sans fin. On pourra mettre en premier i ou j, mais alors l'ordre d'entrée des données sera différent.

```
FOR I=0 TO 3
FOR J=0 TO 4 ---------→ A(0,0)
                        A(0,1)
                        A(0,2)
                        A(0,3)
                        A(0,4)
                        A(1,0)
                          ⋮
                        A(3,3)
                        A(3,4)
```

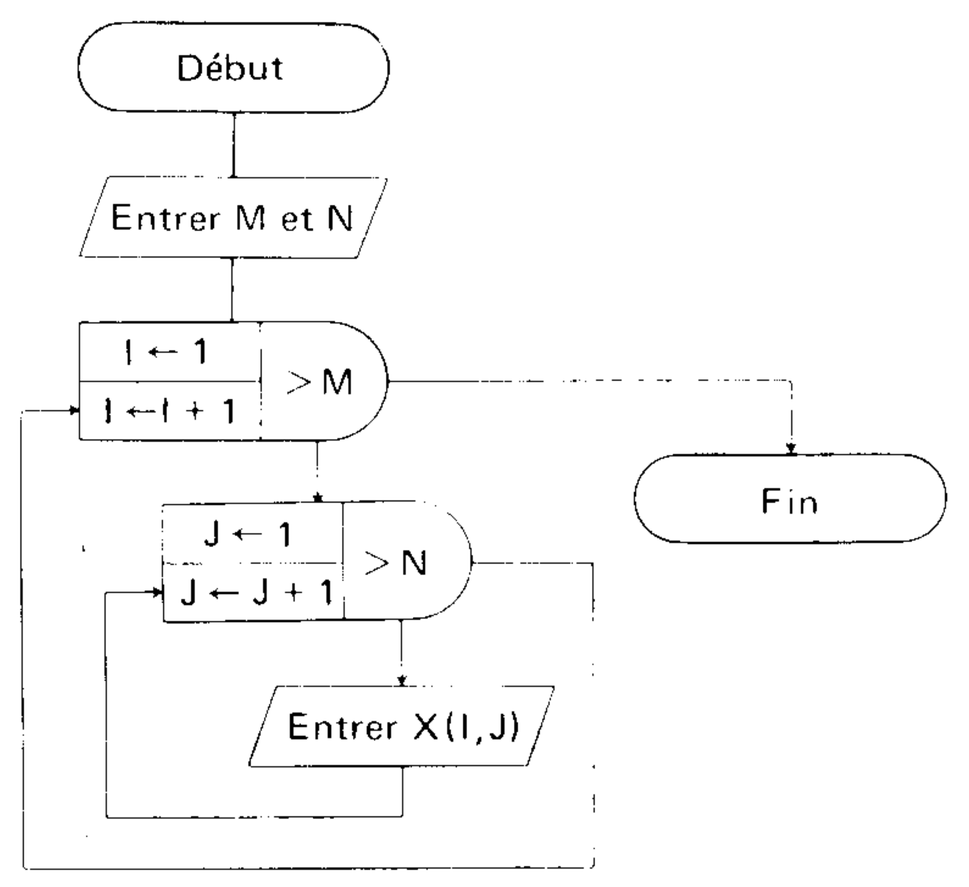
```
FOR J=Ø TO 4
FOR I=Ø TO 3 ──────→ A(Ø,Ø)
                     A(1,Ø)
                     A(2,Ø)
                     A(3,Ø)
                     A(Ø,1)
                       ⋮
                     A(2,4)
                     A(3,4)
```

Considérons un tableau avec deux variables telles que A(1, 1) ou A(2,3) disposées en une matrice.

Exemple 2: Créer un programme pour la collecte de données et obtenir la somme de la matrice. Le nombre de données sera entré ultérieurement.

Ce programme est utilisé pour stocker toutes les données afin d'obtenir la somme de la matrice dans un tableau.

```
10 VAC
20 INP N,M
30 FOR I=1 TO M
40 FOR J=1 TO N
50 INP A(I,J)
60 NEXT J
70 NEXT I
80 END
```

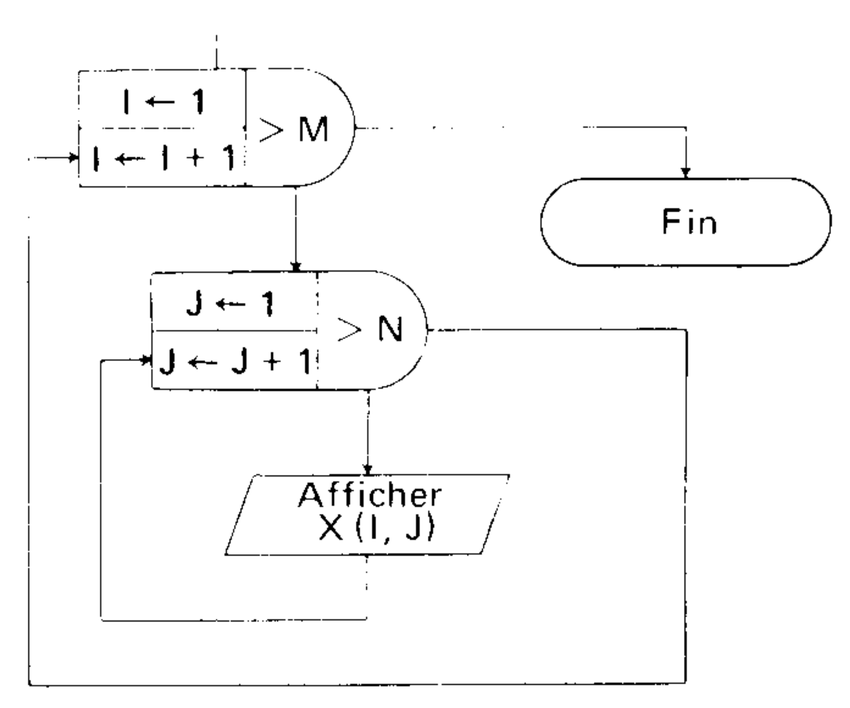


Ce programme ne vaut que pour l'entrée. Il ne se termine pas à la ligne 80; il continue.
L'instruction FOR•NEXT des lignes 30 à 70 est utilisée pour entrer les données de la même manière qu'à l'exemple 1.

Etant donné que ce programme ne considère que l'entrée, ajoutons-y celui de calcul et de sortie.

Par exemple, pour afficher toutes les données:

```
10 VAC
20 INP N,M
30 FOR I=1 TO M
40 FOR J=1 TO N
50 INP A(I,J)
60 NEXT J
70 NEXT I
80 FOR I=1 TO M
90 FOR J=1 TO N
100 PRT A(I,J)
110 NEXT J
120 NEXT I
130 END
```



Ceci est ajouté à la suite de la ligne 80. La méthode d'utilisation est pareille que pour les instructions FOR•NEXT des lignes 30 à 70. La seule différence est que la sortie est remplacée par l'entrée.

Dans la forme ci-dessus, le tableau est une sorte de variable. Il comporte deux éléments qui sont des indices inférieurs fixés à la désignation de variable.
Par exemple, la variable est déterminée par la valeur numérique de i et j en A(i,j).

4-5-3 Ordres d'entrée/sortie

Ce chapitre décrit de manière les ordres d'entrée/sortie utilisés avec cette caclulatrice.

■ Ordres d'entrée

Un ordre d'entrée est une instruction qui permet d'entrer la donnée au cours de l'exécution du programme. Il met en jeu une instruction INP et KEY.

• Instruction INP

L'instruction INP est utilisée pour attribuer manuellement une valeur à une variable lorsque " ?" est affiché au cours de l'exécution du programme.
L'instruction INP est composée de:

INP ["chaîne de caractères"] ,variable, ["chaîne de caractères"] , variable . . . . . .

Ces chaînes de caractères peuvent être omises; toutefois, si elles sont utilisées, elles sont affichées lors de l'état d'attente d'une entrée lors de l'exécution du programme. Elles peuvent prendre la forme d'un message. Les variables peuvent être des variables numériques, des variables de caractères, ou des variables de caractères exclusives ($).

Exemple:

Quand **INP A**	?
Quand **INP "A=" , A**	A=?

Le processus passe à l'étape suivante quand la donnée est entrée et la touche [EXE] frappée lors de l'état d'attente d'une entrée.

Exemple: Ajouter une message à l'instruction INP dans le programme pour déterminer la somme, la différence et le quotient de deux variables.

Le programme est montré ci-dessous.

```
10 INP "A=",A,"B=",B
20 S=A+B
30 D=A-B
40 P=A*B
50 Q=A/B
60 PRT S,D,P,Q
70 END
```

L'exécution se passe ainsi:

Opération:

RUN [EXE]	A=?
45 [EXE]	B=?
23 [EXE]	68
[CONT]	22
[CONT]	1035
[CONT]	1.956521739

De cette manière, la chaîne de caractères qui a été incorporée à l'instruction INP est affichée en tant que message, et l'entrée au clavier s'en trouve facilitée.

Avec cette instruction INP, si l'on entre une donnée composée de caractères dans une variable numérique, une erreur sera déclarée. Si l'on efface l'erreur à l'aide de la touche AC, "? " est affiché à nouveau et l'on retrouve l'état d'attente d'une entrée.
L'entrée pour une variable numérique peut être faite sous forme d'une expression numérique.
Etant donnée que l'état d'entrée ne peut pas être supprimé par simple pression de la touche AC, la procédure d'annulation de cet état est la suivante:

1: Si la touche EXE est directement pressée sans avoir fait aucune entrée, "STOP" est affiché et l'exécution du programme s'arrête. Si l'on presse maintenant la touche AC, l'état d'attente d'une entrée est supprimé. Pour continuer le programme, frapper la touche CONT.
2. L'état d'attente est annulé si l'on frappe MODE 0 pour redésigner le mode RUN. Dans ce cas, il n'est pas possible de continuer le programme.

• Fonctions KEY

Cette fonction lit un caractère pour chaque touche frappée durant l'exécution du programme. Contrairement à l'instruction INP, cette fonction ne peut être accomplie durant l'état d'attente d'une entrée. Le programme se poursuit comme prévu, et s'il n'y a pas d'entrée, rien ne se passe.
La fonction KEY se compose ainsi:

Variable de caractères = KEY

Le caractère lu par la fonction KEY est attribué à la variable de caractères.

Exemple:

```
10 A$=KEY:IF A$=" " THEN 10
20 IF A$="1" THEN 100
30 IF A$="2" THEN 200
40 IF A$="3" THEN 300
50 GOTO 10
     ⋮
     ⋮
```

Ce programme ne sert qu'à l'entrée de la donnée.
La fonction KEY de la ligne 10 vérifie si le caractère lu à été frappé sur le clavier à l'aide d'une instruction IF. Même si la touche EXE n'est pas pressée, cette fonction KEY est capable de lire le premier caractère entré, mais celui-ci seulement.
Cependant, elle n'arrête pas le déroulement comme le fait l'instruction INP. Toutefois, en la combinant avec une instruction IF, on obtient un état d'attente d'entrée. De même, les lignes 20 à 40 comparent le contenu des variables de caractère lu et déterminent la destination des sauts. Ainsi, la fonction KEY ne peut lire qu'un seul caractère frappé sur une touche.

■ Ordres de sortie

Dans un ordre de sortie, il y a neu instruction PRT pour afficher les résultats d'un calcul ou des données et une instruction DMS pour afficher la valeur de l'expression numérique après conversion en hexadécimal, degrés, minutes, secondes. Il y a également une instruction WAIT pour déterminer la durée de l'affichage.

• Instruction PRT

Une instruction PRT se compose ainsi:

PRT [**fonction de contrôle de sortie** { , ; }] { **expression numérique** / **expression de caractères** } [{ , ; } . . . . .]

{ } L'un ou l'autre de ces contenus est utilisable
[ ] Ces contenus peuvent être omis

L'instruction PRT affiche la valeur de l'expression numérique ou la variable numérique, et la chaîne de caractères mise entre guillements " " ou le contenu de la variable de caractères.

Exemple:

```
10 INP "A=",A,"B=",B
20 C=INT (A/B)
30 D=A-B*C
40 PRT A;"/";B;"=";C;"...";D
50 GOTO 10
```

Ce programme calcule le restant lorsque par exemple A est divisé par B. La valeur des variables est affichée de la manière indiquée en ligne 40.
Les articles entre guillements " " sont affichés tels quels, comme caractères.
De plus, la manière dont " , " et " ; " sont affichés en tant que variables et chaînes de caractères est différente.
Dans le cas de " ; ", la donnée suivante est affichée après la précédente.
Dans le cas de " , ", la donnée suivante est affichée après que l'affichage précédent ait été effacé.
Lorsque la durée d'affichage n'est pas spécifiée au moyen d'une instruction WAIT dans le cas de " , ", "STOP" est affiché et le traitement s'arrête. Pour continuer l'affichage, frapper la touche [CONT].

• Instruction DMS

L'instruction DMS sert à afficher la valeur de l'expression numérique ou de la variable après conversion en hexadécimal. L'affichage des valeurs décimales est exprimé en degrés, minutes et secondes.

Exemple:

```
10 INP N
20 DMS N
30 GOTO 10
```

Ce programme rentre une certaine donnée qui est convertie en hexadécimal. Même sans instruction "PRT", l'affichage est obtenu uniquement par l'instruction DMS. L'affichage ne reste que pendant un certain temps puis la calculatrice procède au programme suivant.

L'affichage est le suivant:

000°00'00.00"
- **Secondes (0,00 à 59,99)**
- **Minutes (0 à 59)**
- **Degrés (0 à 99999)**

De plus, lorsque la valeur de l'expression numérique est supérieure à 99999, elle est exprimée en décimal.
La partie décimale des secondes est affichée avec arrondissement à la troisième décimale.

• Instruction WAIT

L'instruction est un ordre déterminant la durée d'affichage obtenu par instruction PRT ou DMS. Il n'est pas nécessaire de presser la touche [CONT] pour conserver l'affichage.
L'instruction WAIT possède la forme suivante:

WAIT expression numérique (0 ≤ expression numérique < 1000)

La partie décimale de l'expression numérique est supprimée. L'affichage reste pendant la durée désignée par multiplication de la valeur par 0,05 seconde environ.
Lorsqu'aucune désignation n'est faite au moyen de l'instruction WAIT ou lorsque la valeur de l'expression numérique est supérieure à 1000, le déroulement s'arrête. "STOP" est alors affiché et le programme se met à l'état d'attente.

Exemple:

```
10 WAIT 5
20 FOR I=1 TO 100
30 PRT I
40 NEXT I
```

Ce programme affiche la valeur de 1 à 100. Lorsqu'aucune désignation n'est faite par l'instruction WAIT, le processus s'arrête après affichage de la valeur de I. Pour obtenir l'affichage de la prochaine valeur, frapper la touche [CONT]. Toutefois, si la durée d'affichage est désignée par l'instruction WAIT, celle-ci sera de 0,25 seconde environ.
La désignation au moyen de l'instruction WAIT est supprimée lorsque la calculatrice est mise hors tension (également lors du fonctionnement de la coupure automatique), ou lors du début d'exécution du programme.

■ Fonctions de contrôle de sortie

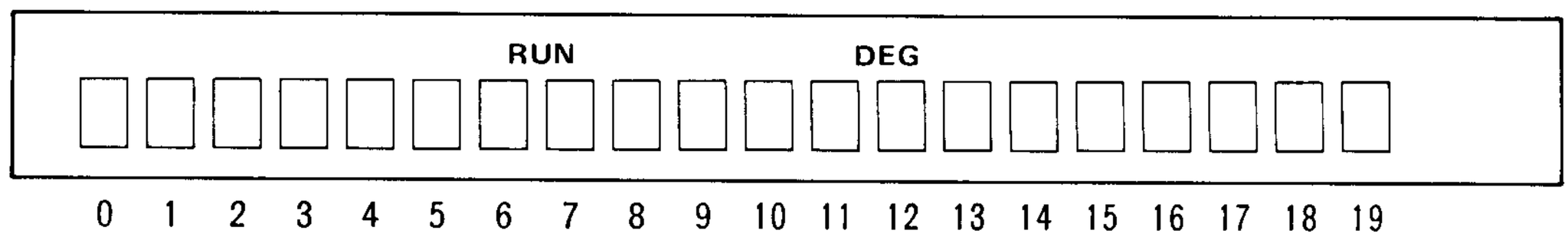
Les fonctions de contrôle de sortie désignent le lieu de sortie et son type.

● Fonction CSR

C'est une fonction de contrôle utilisée dans une instruction PRT. Elle fixe la position de la sortie sur l'affichage (20 positions) au moyen de PRT.
Le format de la fonction CSR est le suivant:

PRT CSR { , ; } **expression numérique** (la partie décimale de l'expression numérique est coupée et sa valeur est comprise entre 0 et 19)

Elle utilise la valeur de l'expression numérique pour déterminer à quel rang à partir de la gauche l'affichage de la donnée de sortie commence.
La méthode de comptage du rang d'affichage est illustrée ci-dessous:



Exemple:

```
10 INP X
20 PRT X;CSR 8;X↑2
30 GOTO 10
```

Ce programme élève au carré la donnée déjà entrée et affiche le résultat ainsi que la donnée. La fonction CSR est utilisée dans l'instruction PRT de la ligne 20.
La donnée "X" est affichée normalement tout à gauche, et la suivante "X↑2" à partir du 8è rang à partir de la gauche par la fonction CSR.

Ce programme est exécuté de la manière suivante:

Opération:

RUN [EXE]	?	
7 [EXE]	7	49
[CONT]	?	
45 [EXE]	45	2025
[CONT]	?	
852 [CONT]	852	725904

Etant donné que deux ces types de sorties sont toujours affichées à des positions distinctes, la lecture en est facilitée.

* S'il y a un " ; " à la suite de CSR, cette donnée est affichée à la position désirée; par contre si l'on utilise un " , ", l'affichage ne sera obtenu qu'après qu'il ait été effacé, L'affichage ne sera donc pas continu.

• Instruction SET

L'instruction SET a été expliquée dans la section calcul manuel pour fixer le nombre de chiffres effectif et le nombre de décimales. Ici nous allons voir comment s'effectue cette désignation dans un programme.
Dans l'exemple d'instruction SET ci-dessous, E fixe le nombre de chiffres effectifs, F les décimales et N annule ces désignations.

SET $\left\{\begin{matrix} \mathbf{E}n \\ \mathbf{F}n \\ \mathbf{N} \end{matrix}\right\}$ (n est 0 à 9)

Exemple 1:

```
10 SET E4
20 MODE 4:WAIT 10
30 FOR X=0 TO 180 STEP 5
40 PRT SIN X
50 NEXT X
60 SET N
70 END
```

Ce programme donne sin x de 0 à 180 degrés par incréments de 5 degrés. Afin d'obtenir les valeurs avec 4 chiffres effectifs dans cet exemple, sin x est affecté de "SET E4".
Si l'instruction SET est placée au début du programme, tous les affichages ultérieurs (sortie de résultats) seront donnés avec 4 chiffres effectifs.
Si les calculs sont pousuivis, les résultats obtenus n'auront que 4 chiffres. Par conséquent, on utilise "SET" N" en ligne 60 pour annuler cette désignation.

Exemple 2: Fixer le nombre de décimales à 4 dans le programme de l'exemple 1.

```
10 SET F4
20 MODE 4:WAIT 10
30 FOR X=0 TO 180 STEP 5
40 PRT SIN X
50 NEXT X
60 SET N
70 END
```

De cette manière, le nombre de chiffres en sortie est réglable tout comme en calcul manuel.

• Mise en format de la sortie

Les résultats de calcul pourraient ne pas toujours être affichés selon un format spécifique en raison de la position de la virgule décimale ou du nombre de chiffres qui varient selon les résultats. Dans un tel cas, on peut utiliser une mise en format de la sortie pour faciliter la lecture.
Les symboles utilisés pour la mise en format sont "#" et "↑". # s'utilise pour spécifier le nombre de chiffres de la mantisse et "↑" pour spécifier l'affichage de l'exposant.

 # #.# # #↑ **(# est un nombre de 10 ou moins)**

- **Désignation d'affichage exponentiel**
- **Nombre de chiffres de la partie décimale**
- **Position de la virgule décimale**
- **Nombre de chiffres de la partie entière**

Le "#" situé à gauche de la virgule décimale fixe le nombre de chiffres de la partie entière.
Le "#" à droite de la virgule décimale fixe le nombre de décimales et le "↑" de la fin désigne l'affichage de l'exposant.

Format	Explication	Affichage quand X = 0,123456789 Y = 12,3456789	
① #	Partie entière à 1 chiffre, sans décimales	X = 0	Y = #
② # . # # #	Partie entière à 1 chiffre, 3 décimales	X = 0.123	Y = # . # # #
③ # # . # # # # #	Partie entière à 2 chiffres, 5 décimales	X = 0.12346	Y = 12.34568
④ # # . # # ↑	Partie entière à 2 chiffres, 2 décimales, avec exponentielle	X = 1.23E −01	Y = 1.23E 01
⑤ # . # # # ↑	Partie entière à 1 chiffre, 3 décimales, avec exponentielle	X = 1.235E −01	Y = 1.235E 01

* En ce qui concerne les exemples ① et ②, quand il y a débordement du nombre de chiffres de la mantisse, l'afichage devient impossible et c'est le format lui-même qui apparaît en sortie.

Exemple:

```
10 MODE 4:WAIT 10
20 FOR X=30 TO 180 STEP 30
30 PRT SIN X;COS X
40 NEXT X
50 END
```

Ce programme est utilisé pour obtenir sin x et cos x tous les 30 degrés. Ces deux valeurs sont affichées en continu. Toutefois, s'il est laissé tel quel, les valeurs seront affichées de manière irrégulière, et leur lecture sera difficile. Dans un tel cas, pour simplifier l'affichage lorsqu'on n'a pas besoin de mantisses à 10 chiffres, on effectue une mise en format.
Lorsque cette mise in format est ajoutée, le programme devient le suivant:

```
10 MODE 4:WAIT 10
20 FOR X=30 TO 180 STEP 30
30 PRT ##.##↑;SIN X;CSR 10;COS X
40 NEXT X
50 END
```

Grâce à cette mise en format, les deux résultats seront affichés simultanément et leur position sur l'affichage sera régulière.
L'affichage sera ainsi d'une lecture bien plus facile.

L'exécution du programme est la suivante:

Opération:

RUN EXE

5.00E-01	8.66E-01
8.66E-01	5.00E-01
1.00E 00	0.00E 00
8.66E-01	-5.00E-01
5.00E-01	-8.66E-01
0.00E 00	-1.00E 00

4-5-4 Fonctions de caractères

Les fonctions de caractères sont associées aux variables de caractères (A$, B$, $, etc.) Leur maniement est différent puisqu'elles sont utilisées pour entrer des caractères au lieu de valeurs numériques. Ces fonctions déterminent la dimension des variables de caractères et peuvent extraire certains caractères hors de chaînes de caractères dans des variables.

LEN et MID

Une fonction LEN est un ordre de compter le nombre de caractères dans une variable de caractères. Il est ainsi possible de connaître la dimension de la variable de caractères.
Une fonction MID extrait certains caractères de la variable de caractères exclusive ($). Elle est utilisée pour redisposer les variables de caractères.
Etant donné que la fonction LEN peut indiquer la dimension d'une variable de caractères, elle est pratique pour lui réserver un espace en sortie, ou pour la diviser lorsque sa dimension est variable.

Exemple:

```
10 A$="123"
20 B$="456"
30 C$=A$+B$
40 D=LEN(C$)
50 PRT D
60 END
```

Ce programme affiche la dimension (nombre de caractères) d'une variable de caractères en ajoutant les variables de caractères A$ et B$.
Seule l'addition est possible avec les variables de caractères. Puisque la dimension de la variable de caractères est inconnue après cette opération, on utilise la fonction LEN pour déterminer la dimension après addition.

Seules les variables de caractères peuvent utiliser la fonction LEN.

Le format de la fonction MID est le suivant:

MID (*m*[,*n*]) (*m*, *n*: expression numérique)

Elle extrait les *n* caractères à partir du mième dans la chaîne de caractères stockée dans la variable exclusive ($).
Si l'on oublie *n*, tous les caractères après le *m* ième seront extraits.
De plus, les décimales de *m* et *n* sont ignorées, et leur valeur peut être de 1 à 30. Si la somme de *m* et de *n* est plus grande que le nombre de caractères stockés dans la variable plus 1, une erreur est produite.

Exemple: $ = "ABCDEFGHIJ"

Pour extraire quatre caractères à partir du troisième → MID (3,4) . . . . CDEF
Pur extraire deux caractères à partir du début → MID (1,2) . . . . AB
Pour extraire tous les caractères à partir du sixième → MID (5) . . . . EFGHIJ

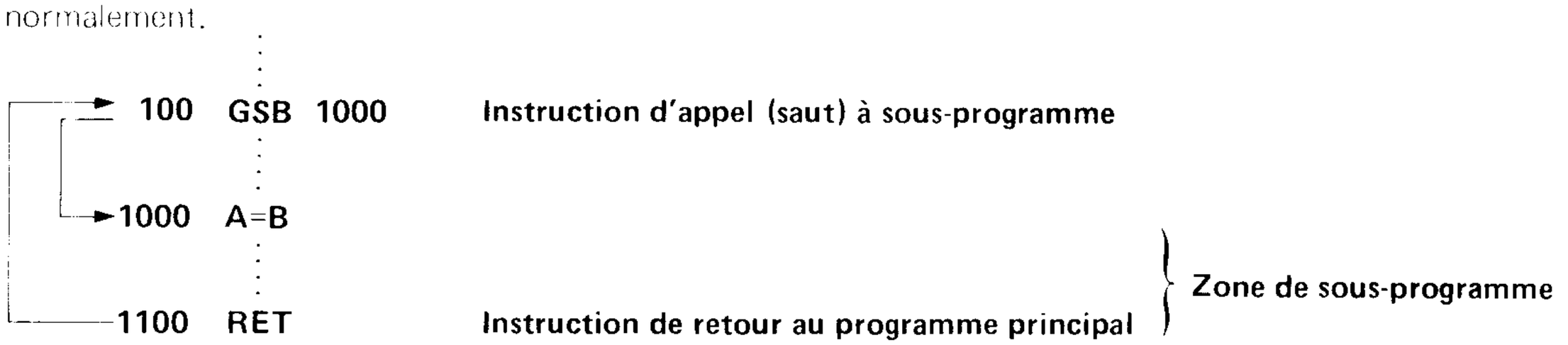
Exemple:

```
10 INP $
20 N=LEN($)/2
30 X$=MID(1,N):Y$=MID(N+1)
40 PRT X$,Y$
50 END
```

Ce programme divise les chaînes de caractères entrées en deux à l'aide des fonctions LEN et MID et les affiche.
Ceci détermine la dimension de la variable de caractères exclusive $ de la ligne 20. Ces deux moitiés sont attribuées à N, la première et la seconde moitiés étant divisées par la fonction MID.
Ainsi, si la dimension de la chaîne de caractères entrée n'est pas fixe, les moitiés ou les tiers ne peuvent pas être extraits; par conséquent, leur dimension est déterminée par la fonction LEN.
De plus, la chaîne de caractères pouvant être introduite en $ est limitée à 30 caractères. Toutefois, si le nombre de caractères en X$ et Y$ est limité à 7, le nombre de caractères introduits en $ est alors limité à 14.

4-5-5 Sous-programmes

Un sous-programme se déroule de manière différente d'un programme principal comme il a été décrit précédemment. Le sous-programme est appelé par le programme principal, et lorsque l'opération désirée est terminée, l'exécution reprend au point du programme principal abandonné auparavant.
En d'autres termes, lorsqu'une instruction (GSB) est donnée, l'exécution saute du programme principal au sous-programme à partir de l'endroit désigné. Lorsque l'opération du sous-programme est terminée, l'exécution reprend au point (instruction GSB) abandonné du programme, et ce dernier continue normalement.



Les ordres nécessaires pour mettre le sous-programme en fonction sont les instructions "GSB" et "RET". L'ordre de saut (appel) au sous-programme est l'instruction GSB, et l'ordre de retour au programme principal est RET. L'instruction GSB peut prendre les formes suivantes:

(1) **GSB expression numérique** (l'expression numérique numérique est coupée aux décimales et peut utiliser les numéros de ligne 1 à 9999).

(2) **GSB # expression numérique** (l'expression numperique numérique est coupée aux décimales et peut utiliser $0 \leq n < 10$).

La première méthode utilise la désignation directe pour écrire directement le numéro de ligne à la suite de "GSB"; la désignation indirecte de la position du sous-programme est possible en utilisant la valeur de la variable.
La seconde méthode consiste à utiliser un sous-programme contenu dans une autre zone programme (P0 à P9) pour une désignation directe ou indirecte.
Le retour au programme principal n'est exécuté que si l'instruction RET est écrite à la fin du sous-programme.

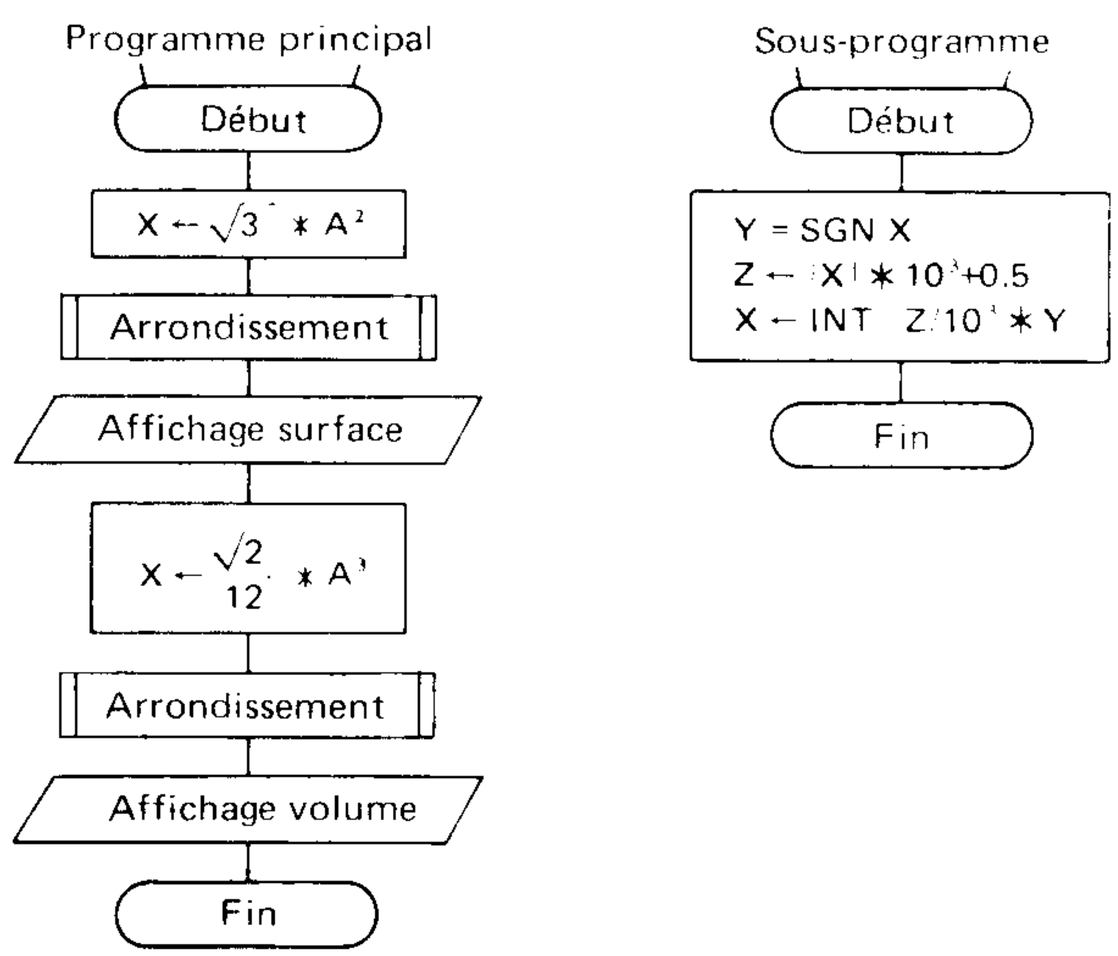
■ Programmes élémentaires de sous-programmes

Nous expliquerons ici la manière de composer des sous-programmes en utilisant des exemples pratiques.

Exemple 1: Créer un programme pour déterminer la surface et le volume d'un tétrahèdre régulier, dont le côté est A, calculés avec trois décimales. La procédure d'arrondissement est inclue dans un sous-programme.

$$\left[\begin{array}{l} S = \sqrt{3}\,a^2 \\ V = \dfrac{\sqrt{2}}{12}\,a^3 \end{array}\right]$$

```
10   INP A
20   X=SQR 3*A↑2
30   GSB 1000
40   PRT X
50   X=SQR 2/12*A↑3
60   GSB 1000
70   PRT X
80   END
1000 Y=SGN X
1010 Z=ABS X*10↑3+0.5
1020 X=INT Z/10↑3*Y
1030 RET
```



Le programme principal de calcul de la surface du tétrahèdre régulier est facile à comprendre, car il est similaire à ceux que nous avons vu jusqu'à présent.
La différence est le programme d'arrondissement inclu dans le sous-programme, à partir de la ligne 1000. La fonction de ce sous-programme est de calculer la surface et le volume, déterminés aux lignes 20 à 50, avec trois décimales et arrondissement.

Le sous-programme peut être utilisé autant de fois que désiré.
Dans ce programme, il est utilisé deux fois, la première pour arrondir la valeur de surface, et la seconde pour arrondir le volume.
Si l'on n'utilisait pas de sous-programme, le programme global serait plus long, comme on le voit ci-dessous.

```
10  INP  A
20  X=SQR  3*A↑2
30  Y=SGN  X
40  Z=ABS  X*10↑3+0.5
50  X=INT  Z/10↑3*Y
60  PRT  X
70  X=SQR  2/12*A↑3
80  Y=SGN  X
90  Z=ABS  X*10↑3+0.5
100 X=INT  Z/10↑3*Y
110 PRT  X
120 END
```

Ce programme et le précédent comportant un sous-programme exécutent le même traitement (les mêmes opérations), mais ce dernier est plus court et plus simple.
En d'autres termes, le sous-programme est utilisé pour simplifier le programme par élimination des opérations répétitives.

* Dans cet exemple, puisque l'arrondissement est inclus dans le sous-programme, "PRT X" ne fait pas partie de ce dernier. Dans le programme ci-dessus, les lignes 30 à 60 et les lignes 80 à 110 sont pareilles, et ce sont elles qui sont incorporées au sous-programme.

Ce sous-programme fait partie du programme principal, mais il est placé après la fin ("END"). Il doit par conséquent être terminé par une instruction RET. Par contre, s'il était inséré au milieu du programme principal, il serait lu de la même manière que ce dernier, et traité de manière répétitive. Lors de l'exécution de l'instruction RET, une erreur serait produite.
Par conséquent, la zone de sous-programme soit être séparée, comme dans cet exemple, en commençant à la ligne 1000 ou 5000. Le sous-programme est similaire à la fonction GOTO (instruction GOTO), et est considéré comme un ordre de saut. Il en diffère par le fait que l'instruction RET est toujours placée à la fin de sorte à retourner à l'instruction GSB après on exécution.

Exemple 2: Créer un programme pour déterminer la surface et le volume d'un tétrahèdre régulier et d'un octahèdre régulier dont le côté est A, calculés avec quatre décimales. Le programme principal et le sous-programme de calcul de la surface de la surface et du volume sont placés en P0, et le sous-programme d'arrondissement en P1.

Tétrahèdre régulier

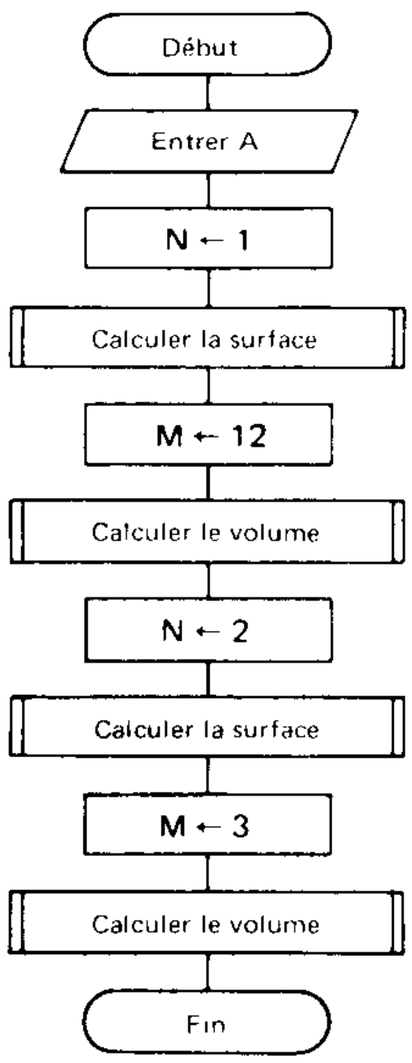
$$S = \sqrt{3}a^2,\ V = \frac{\sqrt{2}}{12}a^3$$

Octahèdre régulier

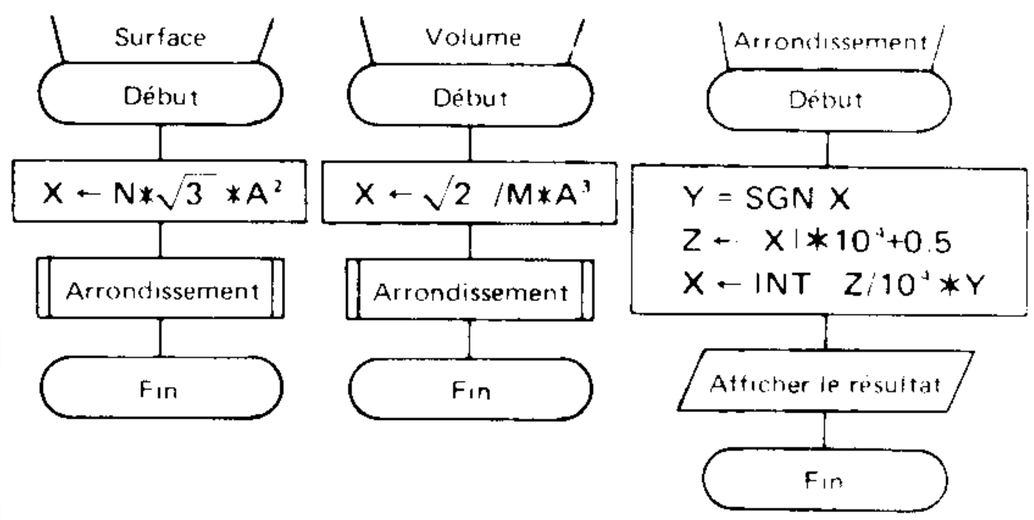
$$S = 2\sqrt{3}a^2,\ V = \frac{\sqrt{2}}{3}a^3$$

```
P0  10   INP  A
    20   N=1
    30   GSB  1000
    40   M=12
    50   GSB  2000
    60   N=2
    70   GSB  1000
    80   M=3
    90   GSB  2000
   100   END
  1000   X=N*SQR  3*A↑2
  1010   GSB  #1
  1020   RET
  2000   X=SQR  2/M*A↑3
  2010   GSB  #1
  2020   RET
```

P0
Organigramme



P1



```
P1      10 Y=SGN X
        20 Z=ABS X*10↑4+0.5
        30 X=INT Z/10↑4*Y
        40 PRT X
        50 RET
```

Dans ce programme, trois sous-programmes sont incorporés; deux ordinaires pour calculer la surface et le volume, et un troisième pour l'arrondissement.

En utilisant cette méthode, on place les opérations de calcul communes dans un sous-programme pour une exécution répétée en changeant simplement les valeurs de N et M. Si l'on place les parties communes dans un sous-programme, on économise de la mémoire et de plus la méthode de calcul est plus facile à comprendre.

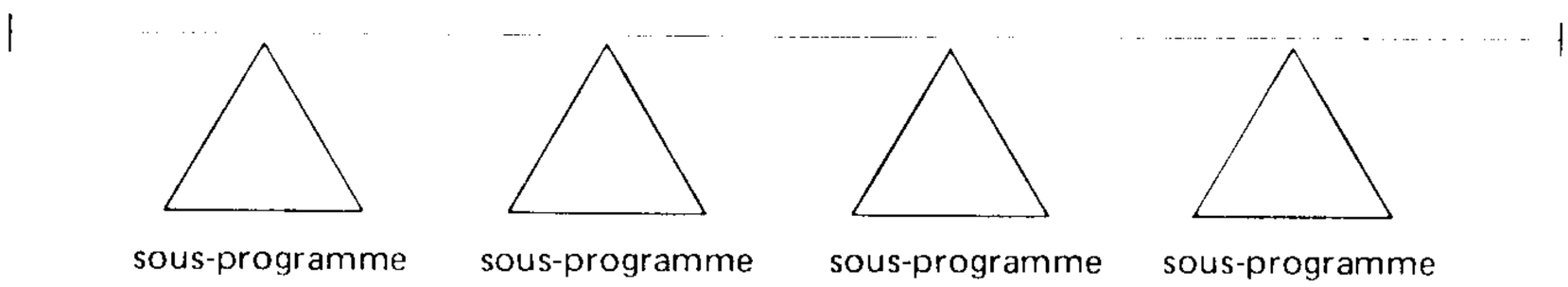
Par conséquent, sur les lignes 1010 et 2010 de ce programme, un second sous-programme est appelé par ce sous-programme. Cette méthode est dite "emboîtement"; on appelle un sous-programme à partir d'un sous-programme, ce qui est une opération similaire à l'appel d'un sous-programme par un programme.

10 niveaux d'emboîtement sont possibles, et au-delà une erreur se produit. L'exécution va du programme principal au sous-programme de 1er niveau. Ce dernier agit en tant que programme principal et appelle le second sous-programme (2è niveau). Même dans ces circonstances, l'instruction RET sur laquelle le sous-programme est basé doit être placée à la fin de chaque sous-programme.
Comme dans le cas du sous-programme d'arrondissement, celui-ci peut être logé dans une autre zone de programme (ici, P1). Cette méthode est pratique car ce sous-programme peut être utilisé communément avec un autre programme entièrement différent (écrit dans une autre zone de programme).

De cette manière, on peut utiliser de nombreux sous-programmes dans le cadre d'un programme.

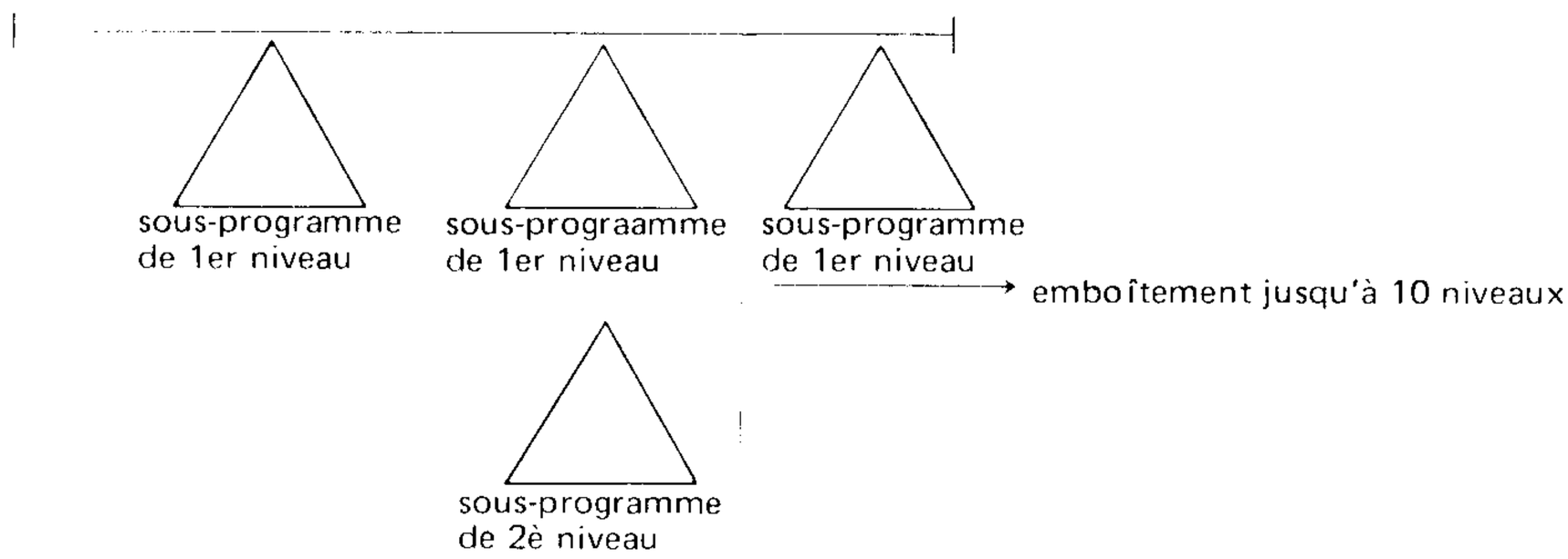
Si l'on utilise l'emboîtement, 10 niveaux de sous-programmes sont possibles.

Programme principal



On peut ainsi utiliser de nombreux sous-programmes

Programme principal



Il est possible d'éviter une partie commune en utilisant un sous-programme. De même, en présence d'un programme complexe, un sous-programme peut être utilisé lorsque ses parties sont séparées en groupes.

L'instruction GSB dans un sous-programme peut être considérée comme équivalente à une instruction GOTO. Elle détermine la zone de sous-programme appelée par le programme principal, puis le retour au point de départ après l'opération.

De plus, lorsque le sous-programme est rédigé dans un même programme, il est nécessaire d'écrire END à la fin du programme principal, et si elle est répétée, il faut écrire une instruction GOTO.

■ Désignation indirecte d'un sous-programme

Comme pour le cas de l'instruction GOTO, la désignation indirecte peut également être utilisée pour un sous-programme.
Le mode d'utilisation est identique à celui de l'instruction GOTO. Le saut s'effectue au sous-programme désigné, puis le retour au point d'origine.

Exemple 1:

```
 10  INP  N
 20  GSB  N+100
 30  PRT  X*6
 40  END
101  X=5:RET
102  X=10:RET
103  X=15:RET
104  X=20:RET
105  X=25:RET
106  X=30:RET
107  X=35:RET
108  X=40:RET
109  X=45:RET
110  X=50:RET
```

Dans ce programme, le sous-programme est indirectement désigné en utilisant la valeur de l'entrée N. Si N est 1 à 10, on va au sous-programme entre 101 et 110, où la valeur de la variable X est déterminée et le résultat du calcul X ✱ 6 est affiché.

Ici, la désignation indirecte de l'instruction GSB détermine la destination du saut (sous-programme) en utilisant la valeur de la variable. Si cette dernière est égale à 1, le saut s'effectue au 1er sous-programme (ligne 101.)
Si la variable est 2, le second sous-programme est appelé (ligne 102) est ainsi de suite. La destination dépend donc de la valeur de la variable.

Exemple 2: Utiliser l'instruction GSB avec une désignation indirecte pour trier les totaux. Ce programme avait été donné précédemment en exemple pour démontrer la désignation indirecte de l'instruction GOTO. (5 classes)

```
 10  VAC
 20  INP  I
 30  GSB  I+100
 40  GOTO 20
100  PRT  A,B,C,D,E:END
101  INP  J:A=A+J:RET
102  INP  J:B=B+J:RET
103  INP  J:C=C+J:RET
104  INP  J:D=D+J:RET
105  INP  J:E=E+J:RET
```

Ce programme est semblable à l'exemple précédent avec l'instruction GOTO à désignation indirecte. Ici, GOTO de la ligne 30 est remplacé par l'instruction GSB
Ici également on entre les numéros de code 1 à 5 en ligne 20, utilise la désignation indirecte en ligne 30 pour l'aiguillage sur les sous-programmes en lignes 101 à 105. Chaque poste est accumulé dans les sous-programmes. Pour ce programme également, chaque sous-programme est créé sur une ligne à l'aide d'une multi-instruction. Chaque ligne représente donc un sous-programme, ce qui facilite la lecture globale. Ici aussi, les numéros de code 1 à 5 sont entrés sur la ligne 20. Si l'on utilisait un autre nombre, une erreur serait causée car la destination correspondante n'existe pas.

4-5-6 Fonctions générales

Les fonctions générales comprennent les fonctions trigonométriques telles que le sinus, cosinus et tangente, et les fonctions incorporées. Celles-ci sont appelées par combinaison de caractères frappés sur le clavier. Elles peuvent également être obtenues par une "fonction d'une touche" en frappant une seule touche, et sont aussi faciles à utiliser que sur une calculatrice scientifique ordinaire.
Les fonctions générales peuvent être utilisées manuellement ou être incorporées à un programme.

Nom de fonction		Forme	Exemple
Fonctions trigonométriques	$\sin x$ $\cos x$ $\tan x$	SIN x COS x TAN x	SIN **30** SIN **A** SIN (**N+5**) COS **3.14** COS **I** COS (**L−3**) TAN **70** TAN **F** TAN (**F ✳ 2**)
Fonctions trigonométriques inverses	$\sin^{-1} x$ $\cos^{-1} x$ $\tan^{-1} x$	ASN x ACS x ATN x	ASN **0.07** ASN **P** ASN (**Z+Y**) ACS π/**5** ACS **D** ACS (**C−X**) ATN **6.5** ATN **V** ATN (**Q+0.5**)
Fonctions Hyperboliques	$\sinh x$ $\cosh x$ $\tanh x$	HSN x HCS x HTN x	HSN **26.8** HSN **T** HSN (**T+E**) HCS **1.45** HCS **0** HCS (**R/2**) HTN **0.861** HTN **H** HTN (**U+0.9**)
Fonctions hyperboliques inverses	$\sinh^{-1} x$ $\cosh^{-1} x$ $\tanh^{-1} x$	AHS x AHC x AHT x	AHS **76.2** AHS **G** AHS (**J+1**) AHC **16** AHS **W** AHS (**S ✳ 2**) AHT **0.91** AHT **M** AHT (**H−E**)
Racine carrée	$\sqrt{x}$	SQR x	SQR **69** SQR **A** SQR (**R ✳ S**)
Fonctions exponentielles	e^x	EXP x	EXP **8** EXP **F** EXP (**D+L**)
Logarithme naturel	$\log_e x$	LN x	LN **43** LN **P** LN (**U+T**)
Logarithme commun	$\log_{10} x$	LOG x	LOG **24.6** LOG **R** LOG (**G+15**)
Intégration (plus grand entier ne dépassant pas x)	INT x	INT x	INT **347.457** INT **V** INT (**Q+U**) INT **−45.43**
Fractionnalisation (x sans sa partie entière)	FRAC x	FRAC x	FRAC **73.54** FRAC **N** FRAC (**H+B**)
Valeur absolue	$\lvert x \rvert$	ABS x	ABS **−9.43** ABS **L** ABS (**K/P**)
Signe (Si $x > 0$, 1) (Si $x = 0$, 0) (Si $x < 0$, −1)		SGN x	SGN **79** SGN **E** SGN (**P−0**)
Conversion sexagésimal à décimal		DEG (degrés, minutes, secondes)	DEG (**39**, **34**, **15**) DEG **T** DEG (**R+5**) degrés minutes secondes
Conversion décimal à séxagésimal		DMS x	DMS **12.582** DMS **Z** DMS (**P+0.05**)
Conversion de coordonnées Rectangulaires à polaires Polaires à rectangulaires		 RPC x, y PRC r, θ	 RPC **14,20.7** RPC A,B PRC **4.5,2/**π PRC F,F

Nom de fonction		Forme	Exemple
Désignation des chiffres significatifs (x déterminé au 10^yème rang par arrondissement)		RND (x, y)	RND (**123.456, 2**) RND (**A, C**) RND (**F+E, G−5**)
Génération de nombres aléatoires (génération de nombres aléatoires dans la gamme $0 < x < 1$)		RAN #	RAN #
Factorielle	x !	x !	8 ! A ! (I+J) !
Unité de mesure angulaire	Degré	MODE 4	Angle droit = 90 degrés
	Radian	MODE 5	Angle droit = $\frac{\pi}{2}$ radians
	Grade	MODE 6	Angle droit = 100 grades

* x et y sont des constantes, variables ou expressions numériques.

Etant donné que ces fonctions sont incorporées, elles peuvent être utilisées telles quelles dans les programmes.

Exemple 1: Créer un programme pour obtenir la longueur d'un côté d'un triangle en utilisant l'angle formé par les deux autres côtés.
[$C = \sqrt{a^2 + b^2 - 2ab\cos\theta}$]

```
10 MODE 4
20 INP A,B,C
30 D=SQR (A↑2+B↑2-2*A*B*COS DEG C)
40 PRT D
50 GOTO 20
```

Etant donné qu'on utilise la mesure angulaire "degrés" dans les fonctions trigonométriques, MODE 4 apparaît sur la ligne 10 de ce programme. Ensuite, la longueur des deux côtés et l'angle sont entrés.
Selon la formule, la racine carrée SQR et le cosinus COS sont écrits dans l'expression numérique.

Exemple 2: Créer un programme pour obtenir le gain en dB d'un amplificateur, la tension d'entrée étant X et la tension de sortie Y.
[$dB = 20 \cdot \log_{10} \frac{Y}{X}$]

```
10 INP X,Y
20 Z=20*LOG (Y/X)
30 PRT Z
40 GOTO 10
```

Si la tension d'entrée est X et la tension de sortie Y, la formule de la ligne 20 peut être posée pour obtenir le gain Z.

Exemple 3: Créer un programme pour obtenir des nombres aléatoires de trois chiffres à l'aide d'une fonction de génération et une fonction de désignation de chiffres significatifs.

```
10 N=RND(RAN#,-4)*1000
20 PRT N
30 END
```

Etant donné que les nombres aléatoires peuvent être générés avec un mantisse de 10 chiffres dans la gamme $0 < x < 1$, trois chiffres doivent être pris comme chiffres significatifs et multipliés par 1000 pour extraire des nombres à trois chiffres.

4-5-7 Traitement statistique

Les fonctions statistiques sont d'un emploi facile car elles sont intégrées à la calculatrice.
Les possibilités de traitement statistique comprennent le calcul de l'écart standard, de régression pour la déterminaison des coefficients de corrélation, etc.

■ Calcul de l'écart standard

Le calcul de l'écart standard peut être exécuté manuellement, ou rédigé dans un programme. Ce chapitre décrit le calcul de l'écart standard au moyen de programmes.
En ce qui concerne le calcul manuel, prière de se reporter au chapitre calculs manuels.

Exemple 1: Créer un programme pour obtenir la différence entre chaque donnée et la moyenne, par stockage de toutes les données dans une liste (rangée unidimensionnelle).

```
 10 SAC
 20 INP "N=",N
 30 FOR I=1 TO N
 40 INP D
 50 STAT D
 60 A(I)=D
 70 NEXT I
 80 PRT "MX=";MX
 90 FOR I=1 TO N
100 PRT "NO.";I;"=";A(I)-MX
110 NEXT I
120 END
```

L'ordre SAC sur la ligne 10 de ce programme sert à annuler (mettre à "0") la mémoire de cumul statistique. L'ordre VAC n'efface pas la mémoire statistique. Avant d'entreprendre tout calcul statistique, il est indispensable d'utiliser cet ordre. L'ensemble des données est introduit en ligne 20 et le nombre de données entrées est déterminé au moyen de l'instruction FOR•NEXT.

Dans l'instruction FOR•NEXT suivante, les données sont cumulées et stockées dans la liste A(i), en utilisant l'ordre d'entrée de données STAT pour le calcul de l'écart standard.

Après l'entrée des données, MX est l'ordre de sortie de la valeur moyenne. A l'aide d'une instruction FOR•NEXT, la valeur moyenne est soustraite de chaque donnée stockée dans la liste et le résultat est affiché.

Prenons un exemple d'exécution de ce programme.

Exemple: nombre de données = 8

Données: 55, 54, 51, 55, 53, 53, 54, 52

L'exécution du programme avec ces données donne les résultats suivants:

Opération:

DEFM1 [EXE]	VAR: 36 PRG: 1600
RUN [EXE]	N=?
8 [EXE]	?
55 [EXE]	?
54 [EXE]	?
⋮	⋮
52 [EXE]	MX= 53.375
[CONT]	NO. 1= 1.625
[CONT]	NO. 2= 0.625
[CONT]	NO. 3=-2.375
[CONT]	NO. 4= 1.625
[CONT]	NO. 5=-0.375
[CONT]	NO. 6=-0.375
[CONT]	NO. 7= 0.625
[CONT]	NO. 8=-1.375

* Lorsqu'on utilise une liste de cette manière, ne pas oublier auparavant d'étendre la capacité de la mémoire

De plus, les ordres d'appel du résultat de l'écart standard dans cet exemple sont les suivants:

CNT . . . . . **nombre de données (n)**
SX . . . . . **Somme (Σx)**
SX2 . . . . . **Somme des carrés (Σx^2)**
MX . . . . . **Moyenne ($\overline{x}$)**
SDX . . . . . **Ecart standard (σ_{n-1})**
SDNX . . . . . **Ecart standard (σ_n)**

Exemple 3: Créer un programme pour obtenir la moyenne des résultats d'examen, l'écart standard σ_n et l'écart du résultat de chaque étudiant.

[Voir l'exemple 1

Valeur de l'écart SS= $\dfrac{(x-\overline{x})\cdot 10}{\sigma_n}+50$]

```
 10 SAC
 20 INP N
 30 FOR I=1 TO N
 40 INP D
 50 STAT D
 60 A(I)=D
 70 NEXT I
 80 PRT "MX=";MX,"SDXN=";SDXN
 90 FOR I=1 TO N
100 S=(A(I)-MX)*10/SDXN+50
110 PRT "NO.";I;"=";S
120 NEXT I
130 END
```

Dans ce programme, les lignes numéro 10 et 70 sont identiques à celles de l'exemple 1. Toutefois, le programme de calcul et d'affichage à partir de la ligne 80 est différent.
La valeur d'écart SS est obtenue à la ligne 100. Celle-ci est obtenue pour chaque étudiant à l'aide d'une instruction FOR•NEXT.

Ainsi, en appliquant la fonction d'écart standard incorporée, la moyenne ou l'écart standard peut être directement utilisée dans le programme.
De plus,
lors de l'entrée de fréquences, le mode opératoire est similaire au calcul manuel. Lors de l'entrée de données dans le programme, la suppression et la correction de données fausses doivant être accomplies manuellement.

■ Régression et coefficients de corrélation

En plus de l'écart standard, la fonction d'analyse de régression (statistiques à 2 variables) est également incorporée.

Pour le calcul de régression, seule la régression linéaire est inclue. Toutefois, l'utilisation de la régression linéaire permet d'obtenir les régressions logarithmique, exponentielle et de puissance à l'aide d'un programme.

Calcul de régression linéaire

- La formule de régression linéaire est y = A+Bx. Le terme constant A et le coefficient de régression B sont calculés comme suit:

$$A = \frac{\Sigma y - B\cdot\Sigma x}{n} \qquad B = \frac{n\cdot\Sigma xy - \Sigma x\cdot\Sigma y}{n\cdot\Sigma x^2 - (\Sigma x)^2}$$

- Le coefficient de corrélation r de la paire de données entrées est calculé par la formule ci-dessous:

$$r = \frac{n\cdot\Sigma xy - \Sigma x\cdot\Sigma y}{\sqrt{\{n\cdot\Sigma x^2 - (\Sigma x)^2\}\{n\cdot\Sigma y^2 - (\Sigma y)^2\}}}$$

- Ceux-ci sont traités dans le programme de la même manière que pour le calcul de l'écart standard.

Exemple: L'entrée des données se fait manuellement. A l'aide du résultat de calcul en sortie, créer un programme pour calculer le terme constant A, le coefficient de régression B, le coefficient de corrélation r et le coefficient déterminant (r^2), ainsi que la covariance $\left(\frac{\Sigma xy - n \cdot \overline{x} \cdot \overline{y}}{n-1}\right)$.

```
10 PRT "A=";LRA,"B=";LRB,"R=";COR,"R↑2=";COR↑2
20 X=(SXY-CNT*MX*MY)/(CNT-1)
30 PRT X
40 END
```

Etant donné que le programme n'opère qu'un simple affichage, l'entrée des données s'effectue manuellement.

La méthode d'entrée des données est la suivante:

donnée x, donnée y [STAT]
ou
donnée x, donnée y; fréquence [STAT]

Ici, "LRA" est le terme constant A, "LRB" le coefficient de régression B, et "COR" le coefficient de corrélation r. De même, le coefficient déterminant et la covariance sont calculés à l'aide des formules puis affichés.

Les fonctions d'affichage des résultats de l'analyse de régression sont indiqués ci-dessous.

CNT	nombre de données (n)
SX	somme des x (Σx)
SY	somme des y (Σy)
SX2	somme des x^2 (Σx^2)
SY2	somme des y^2 (Σy^2)
SXY	somme du produit des données (Σxy)
MX	moyenne des x ($\overline{x}$)
MY	moyenne des y ($\overline{y}$)
SDX	écart standard de x ($x\sigma_{n-1}$)
SDY	écart standard de y ($y\sigma_{n-1}$)
SDXN	écart standard de x ($x\sigma_n$)
SDYN	écart standard de y ($y\sigma_n$)
LRA	terme constant (A)
LRB	coefficient de régression (B)
COR	coefficient de corrélation (r)
EOX	valeur estimée de x ($\hat{x}$)
EOY	valeur estimée de y ($\hat{y}$)

L'analyse de régression étant effectuée automatiquement sur la base des seules données d'entrées, elle est d'un usage facile tant en programme complet ou en programme d'affichage.
Lorsque les données sont entrées manuellement, s'assurer de frapper [F1] [SAC] au préalable.

Calcul de régression logarithmique

- La formule de régression logarithmique est y=A+B•ln$x$.
- La régression logarithmique est effecutée à l'aide de la fonction de calcul de la régression linéaire. La méthode d'entrée des données est ici différente, et s'effectue plus aisément avec un programme.

Exemple: Créer un programme appliquant la fonction de de régression linéaire à la régression logarithmique exécuté par seule entrée des données x et y.

La sortie des résultats s'effectue manuellement, et n'est pas inclue dans le programme.

```
10 SAC
20 INP X,Y
30 STAT LN X,Y
40 GOTO 20
```

Ce programme est très simple et ne comprend que l'entrée. Il se compose de l'instruction SAC qui annule la mémoire statistique, l'instruction INP et l'instruction STAT d'entrée des données.

Ce STAT utilise le logarithme naturel ln de x. Puisque la formule de régression est y=A+B•ln$x$, y peut être entrée telle quelle. Toutefois, x doit être introduit sous forme de son logarithme.
Ce programme n'est assemblé que pour la partie entrée. Après l'introduction des données, il est nécessaire d'arrêter ce programme (frapper [MODE][0]) et de frapper la touche de fonction de sortie.
De même, le nombre de données doit être introduit au début pour obtenir l'achèvement automatique et pour afficher les résultats, comme dans l'exemple d'écart standard.
De plus, ce programme possède une fréquence de 1. Pour pouvoir entrer les fréquences, le modifier comme suit:

```
10 SAC
20 INP X,Y,Z
30 STAT LN X,Y;Z
40 GOTO 20
```

Puisque dans ce programme la fréquence doit être entrée, même si elle est de 1, s'assurer d'entrer "1".

Calcul de régression exponentielle

- La forme de régression exponentielle est $y=A\cdot e^{B\cdot x}$ ($\ln y=\ln A+B\cdot x$).
- La régression exponentielle est exécutée à l'aide de la fonction de régression linéaire, comme pour la régression logarithmique.

Exemple: Créer un programme d'entrée/sortie pour la régression exponentielle. En sortie l'affichage livre les termes constants, les coefficients et la corrélation.

```
10 SAC
20 INP "DATA=",N
30 FOR I=1 TO N
40 INP "X=",X,"Y=",Y
50 STAT X, LN Y
60 NEXT I
70 PRT "A=";EXP LRA,"B=";LRB,"R=";COR
80 END
```

Les lignes 10 à 60 de ce programme utilisent une forme identique à celle des précédents. Toutefois, la donnée Y n'est pas prise telle quelle, mais est introduite sous forme du logarithme naturel ln. La formule de régression étant $\ln y=\ln A+B\cdot x$, y doit être introduit en tant que logarithme de y ($\ln y$). A l'entrée, le nombre de données initiales est obtenu au moyen d'une instruction FOR•NEXT.
Pour qu'il puisse être utilisé tel quel, le terme constant A est calculé en tant que ln A. Pour tirer A, on cherche l'antilog de lnA et calcule EXP A (e^A) à l'aide de la fonction exponentielle.

Le résultat devient le terme constant A.
Le coefficient de régression B et le coefficient de corrélation r peuvent être utilisés tels quels.

Calcul de régression à la puissance

- La formule de régression à la puissance est $y=A\cdot x^B$ ($\ln y=\ln A+B\cdot \ln x$).
- Cette régression peut également être traitée à l'aide de la fonction de régression linéaire.

Exemple: Créer un programme d'analyse de régression à la puissance affichant les termes constants, les coefficients de régression et les coefficients de corrélation en sortie et ainsi obtenir les valeurs estimées.

```
10 SAC
20 INP "DATA=",N
30 FOR I=1 TO N
40 INP "X=",X,"Y=",Y
50 STAT LN X,LN Y
60 NEXT I
70 PRT "A=";EXP LRA,"B=";LRB,"R=";COR
80 INP "X OR Y",Z$
```

```
 90 IF Z="Y" THEN 130
100 INP "Y=",Y
110 PRT EXP EOX LN Y
120 GOTO 80
130 INP "X=",X
140 PRT EXP EOY LN X
150 GOTO 80
```

Les lignes 10 à 70 ce ce programme possèdent le même genre d'entrée/sortie, mais l'entrée des données sur ligne 50 par instruction STAT est différente des cas précédents.
Les deux données x et y sont entrées sous forme de logarithme naturel ln. Puisque la formule de régression est $\ln y = \ln A + B \cdot \ln x$, il est nécessaire d'introduire le logarithme des données x et y.

En sortie, A est également délivré sous forme de logarithme lnA, et il faut tirer l'antilog A. Le programme de calcul de valeur estimée présente deux cas; l'un estime $\hat{Y}$ de la donnée x. L'autre estime $\hat{X}$ de la donnée y. La décision est effectuée sur les lignes 80 et 90. En d'autres termes, si Z$ est $\hat{X}$, entrer la donnée y pour obtenir $\hat{X}$. Si Z$ est Y, entrer la x pour obtenir $\hat{Y}$.

Ce mode opératoire permet d'effectuer facilement même le calcul de régression à la puissance en utilisant un programme. Si l'on effectue une erreur lors de la composition de la partie entrée du programme, il est nécessaire soit de procéder manuellement ou d'inclure dans le programme une opération compensatrice.

4-5-8 Mot de passe

Cette calculatrice permet d'attribuer des mots de passe pour protéger des programmes écrits. Un mot de passe est un "code". Lorsque l'auteur du programme ou son utilisateur désire garder secret son contenu, ou qu'il ne désire pas qu'il soit altéré par d'autres, il peut y joindre un mot de passe. Les instructions "LIST" ou "CLR" seront alors inopérantes.

Un mot de passe prend la forme suivante:

PASS "ABC"

Les lettres ou nombres entre guillemets, "ABC" par exemple, constituent le mot de passe. Toutes les lettres et chiffres sont utilisables, et le mot peut comprendre jusqu'à 8 caractères.

Exemples: PASS "CASIO"
PASS "#$"
PASS "AZ−1"

Etant donné que les mots de passe sont efficaces pour chaque zone de programme (P0 à P9), il ne peuvent être annexés qu'à P0 ou P9.
Le mot de passe peut être effacé en annulant le programme entier par l'ordre "CLR ALL" ou en réintroduisant le même mot de passe.
Si le mot de passe est entrée à la suite de "LIST", l'ordre LIST peut être momentanément effacé.

Exemple:

PASS "CASIO" [EXE] . . . . . Le mot de passe est fixé

LIST "CASIO" [EXE] . . . . . Le mot de passe est provisoirement supprimé (ou LIST 20 "CASIO" [EXE])

Cette suppression temporaire du mot de passe n'est efficace que pour l'ordre LIST au moment de l'entrée.
Si le mot de passe n'est pas réintroduit par la suite, une erreur mot de passe est déclarée.

4-5-9 Caractéristiques optionnelles

■ CMT (bande magnétique en cassette)

Un enregistreur à bande magnétique peut être raccordé à cette calculatrice pour former une mémoire externe. Cet enregistreur peut conserver aussi bien des programmes que des données, ce qui permet de stocker ses programmes importants. Un autre type d'enregistreur peut également être utilisé pour cette fonction. Si cet appareil possède une borne de télécommande, il pourra être commandé à distance à partir de la calculatrice, par l'intermédiaire de l'adaptateur FA-2. En ce qui concerne le raccordement de la calculatrice et de l'enregistreur, prière de se reporter au mode d'emploi du FA-2.

L'enregistreur dont il est question ici est un enregistreur à cassette doté d'une borne de télécommande. Si l'enregistreur utilisé ne possède pas cette fonction, il devra être être manipulé en fonction de l'utilisation de la calculatrice.

L'avance et l'arrêt en lecture comme en enregistrement peuvent être télécommandés, mais pas l'avance rapide et le rebobinage.

● Sauvegarde/chargement de programme SAVE/LOAD

L'instruction pour la sauvegarde du programme est indiquée ci-dessous.

SAVE [# "nom de fichier"] (n = 0 à 9, les caractères entre guillemets peuvent être omis)

#n désigne la zone de programme. Si c'est #0, le programme contenu en P0 est sauvegardé. Si c'est #3, le programme contenu en P3 le sera.

Pour le nom de fichier, on peut utiliser jusqu'à 8 caractères identiques, lettres, chiffres ou symboles.

Exemple: "A"
"B"
"✱✱✱✱✱✱✱✱"

Ce #n et le nom de fichier peuvent être omis. Si #n est omis, c'est le programme se trouvant dans la zone présentement utilisée qui sera sauvegardé sur bande lorsque l'instruction SAVE est exécutée.
Le programme sauvegardé par l'instruction SAVE est celui se trouvant dans la zone désignée. Si l'on désire sauvegarder simultanément les programmes de toutes les zones (P0 à P9), utiliser l'instruction SAVE ALL (voir page 77).

Mode opératoire:

(1) Charger une cassette et noter le numéro du compteur de bande.
(2) Mettre l'enregistreur en mode enregistrement.
(3) Mettre en oeuvre SAVE [#n "nom de fichier"] EXE .
(4) Lorsque l'enregistrement des programmes est terminé, la bande s'arrêtera automatiquement. (Arrêter la bande manuellement si l'enregistreur ne possède pas de borne de télécommande.)

* L'instruction SAVE ne peut être utilisée que manuellement. Elle ne peut pas être incorporée à un programme.

L'instruction nécessaire pour charger un programme enregistré est la suivante:

LOAD [#n "nom de fichier"] (n entre 0 et 9, caractères entre guillemets pouvant être omis)

#n indique la zone de programme à rappeler. Si c'est P1, il sera chargé en P1. Le nom de fichier peut être omis, comme dans le cas de SAVE. Si c'est le cas, le premier programme découvert après la mise en marche sera chargé.

En outre, si #n est omis, le programme correspondant à la zone utilisée sera chargé lorsque l'instruction LOAD sera exécutée.

L'instruction LOAD peut être exécutée manuellement ou écrite dans le programme.

Mode opératorie:

Opération manuelle:

(1) Amener la bande juste avant l'endroit du début à l'aide du compteur (2 ou trois chiffres avant).
(2) Mettre l'enregistreur en marche en mode de lecture avec la télécommande.
(3) Mettre en oeuvre LOAD [#*n* "nom de fichier"] [EXE].
(4) Si l'enregistreur utilisé ne possède pas de borne de télécommande, le mettre en mode de lecture après l'instruction LOAD.
(5) Lorsque le transfert du programme est terminé, la bande s'arrête automatiquement. (arrêter l'enregistreur manuellement s'il ne possède pas de borne de télécommande.)

Affichage durant le chargement du programme . . . . .

```
PF: AAA
```

Fiche de programme — nom de fichier

Affichage lorsque le chargement est terminé . . . .

```
READY Pn
```

* Même s'il existe un programme précédent dans la zone où l'on désire en charger un autre, celui sur bande sera tout de même chargé correctement. Dans ce cas, le programme précédent sera annulé depuis le numéro de ligne initial du nouveau programme tandis que celui-ci est chargé.

Exemple:

Programme restant en P0

```
10  INP A
     ⌇
90  PRT B
100 FOR I=1 TO 100
     ⌇
300 PRT C
```

+

Nouveau programme chargé

```
100 INP X
     ⌇
200 PRT Y
```

→

Etat des programmes après LOAD

```
10  INP A
     ⌇
90  PRT B
100 INP X
     ⌇
200 PRT Y
```

Pour écrire une instruction LOAD dans le programme:

(1) Ecrire l'instruction LOAD dans le programme.
(2) Mettre l'enregistreur en marche en mode de lecture à l'aide de la télécommande.
(3) Le programme comportant l'instruction LOAD est chargé dans le mode RUN ([MODE] [0]).
(4) Si l'enregistreur ne comporte pas de borne de télécommande, le démarrer en mode de lecture après le début du programme.
(5) Lorsque le transfert du programme est terminé, la bande s'arrête automatiquement et le programme chargé commence.

Lorsqu'une instruction LOAD est incorporée au programme, son exécution commence dès que LOAD est terminé.

Exemple:

Programme d'exécution	Nom de fichier "A"	Nom de fichier "B"	Nom de fichier "C"
P0: 10 LOAD#0"A"	**100 INP X**	**100 INP Y**	**100 INP Z**
20 LOAD#0"B"	⌇	⌇	⌇
30 LOAD#0"C"	**500 GOTO 20**	**500 GOTO 30**	**500 END**

Avec ce genre de programme, si la ligne numéro 10 est exécutée, le programme portant le nom de fichier "A" est lu. Lorsque cela est terminé, le programme "A" est exécuté. Lorsque celui-ci est terminé, l'exécution reprend à la ligne 20 et le programme portant le nom "B" est lu puis exécuté. Lorsque le programme est écrit de cette manière, la lecture et l'exécution se produisent séquentiellement.

- **Stockage et rappel de données PUT/GET**

Le stockage de données s'effectue au moyen de l'ordre PUT.

PUT ["nom de fichier"] variable (l'article entre parenthèses peut être omis)

Les variables sont $, A et B à T9.

Pour stocker des variables de A à Z, écrire "A , Z".

Exemples: Pour SAVE $, A, B, C, D . . . . . PUT ["nom de fichier"] $, A , D
Pour SAVE A à Z . . . . . PUT ["nom de fichier"] A , Z
Pour SAVE A à Z et A0 à T9 . . PUT ["nom de fichier"] A , T9
ou
PUT ["nom de fichier"] A , A (199)

* Pour l'écriture de variables, "A , Z" signifie dans ce cas de A à Z. Le stockage des données de A à Z s'effectue dans l'ordre, et il n'est pas possible d'écrire une séquence du plus grand au plus petit, comme Z à A.

Exemple: PUT H , A ——————→ erreur

Le rappel de données s'effectue au moyen d'un ordre GET.

GET ["nom de fichier"] variable (l'article entre paranthèses peut être omis)

Cette variable est écrite sous forme "$, A" ou "$, A , Z" de la même manière que dans l'exemple PUT.

Les ordres PUT et GET peuvent être exécutés manuellement ou être incorporés au programme.

Le mode opératoire manuel est similaire à l'opération SAVE et LOAD.

Pour l'exécution par le programme, la mise en marche et l'arrêt de l'enregistreur à cassette peut être effectuée automatiquement s'il dispose d'une prise de télécommande. Avec un appareil sans cette prise, il s'agit de bien synchroniser les opérations.

Exemple: pour stocker

```
490 PUT "D" A , Z
500 END
```

pour rappeler

```
10 GET "D" A , Z
```

Pour un programme tel que celui-ci, le stockage de données s'accomplit ainsi:

(1) Charger la bande cassette et noter le numéro du compteur.
(2) Mettre l'enregistreur en mode d'enregistrement.
(3) Demarrer le programme comportant l'ordre PUT.
(4) Lors de la lecture de PUT, la bande démarre automatiquement et s'arrête quand l'opération est terminée.

Le rappel des données s'accomplit ainsi:

(1) Charger la bande et l'enrouler jusqu'à un point précédant légèrement le début à l'aide du compteur.
(2) Mettre l'enregistreur en mode de lecture.
(3) Démarrer le programme comportant l'ordre GET.
(4) Lorsque le GET est lu, la bande démarre automatiquement et s'arrête lorsque le rappel est terminé.

Affichage durant LOAD . . . . . **VF : D**

fichier de données — nom de fichier

Comme pour programmer SAVE/LOAD, le nom du fichier peut également être omis.

- **SAVE/LOAD de programme/données complets**

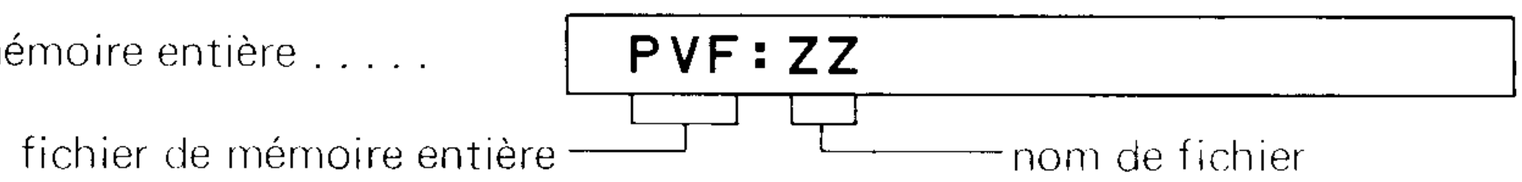
Pour le SAVE/LOAD décrit précédemment, le programme figurant dans une zone de programme spécifique est appelé. Si le programme occupe plusieurs zones, par exemple P0 à P3, ou P2 à P6, ou si les données requises pour celui-ci sont dans une autre zone, l'ensemble programme/données peut être sauvegardé et chargé.

Les ordres de transfert pour la mémoire entière sont décrits ci-dessous:

SAVE ALL ["nom de fichier"]
LOAD ALL ["nom de fichier"]
(les articles entre parenthèses peuvent être omis.)

Cette opération est semblable à SAVE/LOAD de programme.

L'affichage durant LOAD pour mémoire entière . . . . .



* Lorsque ce LOAD ALL est exécuté, tous les programmes et données stockés sur la bande seront transférés. L'ordre DEFM utilisé pour étendre la mémoire de données sera également exécuté s'il est présent.

- **Vérification de fichiers sur bande**

Pour vérifier l'exactitude du programme ou des données présents sur la bande, un ordre VER est utilisé.

VER ["nom de fichier"] (l'article entre parenthèses peut être omis)

Le mode opératoire est similaire à LOAD de programme.

■ Imprimante

La mini-imprimante (FP-10) peut être raccordée à cette calculatrice. Celle-ci permet d'obtenir le listing du programme et des données. De même, les sorties de résultats, etc. peuvent être obtenues lors d'exécution de programmes.

En ce qui concerne le raccordement et la manipulation de l'imprimante, prière de consulter le mode d'emploi FP-10.

Pour imprimer un listing de programme ou de données, frapper [MODE] [7] et le mode PRT sera choisi (affichage "PRT").
Pour l'impression des résultats de calculs, le mode PRT peut être désigné manuellement, ou [MODE] [7] peut être incorporé au programme pour obtenir une impression partielle.

Le calcul en mode PRT s'obtient manuellement en frappant [MODE] [8] ou en incorporant [MODE] [8] au programme.

- **Listing de programme**

Pour imprimer le listing d'un programme, exécuter l'instruction LIST en mode RUN.

LIST { **numéro de ligne** / **#0 à 9** } **["Mot de passe"]** (les articles entre parenthèses [ ] peuvent être omis)

Lorsque le numéro de ligne est indiqué, l'impression se produit à partir de cette ligne. Si #0 à 9 est écrit, la zone de programme désignée (P0 à P9) sera imprimée.
De plus, si un mot de passe est inclus, il doit être indiqué. S'il n'y en a pas, il n'est pas requis.

- **Listing de données**

Pour imprimer le listing de données, exécuter un ordre LIST en mode RUN.

LIST V

Cette instruction imprime la liste du contenu de toutes les variables (données).

- **Listing de l'ensemble des programmes/données**

Pour imprimer en même temps le listing des programmes en P0 à P9 et des données, exécuter l'instruction LIST ALL en mode RUN.

LIST ALL

Ce LIST ALL commande l'impression du listing entier P0 à P9. Toutefois, les zones ne comportant pas d'entrées ne seront pas imprimées.
Pour les programmes comportant un mot de passe, PASS sera imprimé.

- **Résultats de calculs**

Si le programme est exécuté après la sélection du mode PRT, les résultats de calcul seront imprimés au moment où ils sont affichés à l'aide d'une instruction PRT et d'une programme DMS.
Si la désignation de mode PRT est inclue dans le programme, seul le résultat désiré sera imprimé.
De plus, pour les résultats de sortie, si la désignation de fonction CSR est située devant la position actuelle d'impression, ils seront placés à la suite, car la tête imprimante ne peut pas reculer.

Liste des messages d'erreur

Code d'erreur	Signification	Cause	Correction
1	Débordement de mémoire ou de piles	● Nombre de pas insuffisant. Le programme est trop long. ● Débordement de piles.	● Annuler les programmes inutiles ou réduire le nombre de mémoires. ● Diviser les formules pour les rendre plus simples.
2	Erreur grammaticale	● Erreur de forme du programme. ● Les termes de gauche et de droite diffèrent par leur instruction d'affectation, etc.	● Corriger l'entrée erronée dans le programme.
3	Erreur mathématique	● Le résultat de calcul d'une expression numérique dépasse 10^{100}. ● Hors de gamme d'entrée d'argument de fonction numérique ● Résultat indéfini ou impossible	● Corriger la formule de calcul ou les données. ● Vérifier les données.
4	Numéro de ligne erroné ou indéfini	● Instruction GOTO ou GSB non assortie d'un numéro de ligne	● Désigner correctement le numéro de ligne
5	Erreur d'argument	● L'argument d'une fonction en nécessitant un est hors de la gamme d'entrée ● L'argument de rangée est hors de 0 – 199 pour une liste ou hors de (0, 0) à (19, 9) pour un tableau	● Corriger l'erreur d'argument.
6	Erreur de Variable	● Utilisation d'une variable non définie par une instruction DEFM	● Augmenter la variable si nécessaire
7	Erreur d'emboîtement	● L'instruction RET sort quand le sous-programme n'est pas exécuté. ● L'instruction NEXT sort quand elle n'est pas dans la boucle FOR ● Plus de 10 niveaux de sous-programmes ● Plus de 8 niveaux de boucles FOR•NEXT	● Eliminer les instructions RET ou NEXT inutiles ● Faire en sorte que les sous-programmes ou les boucles d'instruction FOR•NEXT soient contenus dans le nombre de piles prescrit.
8	Erreur de mot de passe	● Le programme comporte un mot de passe et un ordre interdit tel que CLR ou LIST a été utilisé. ● Le programme comporte un mot de passe et un autre a été déclaré à l'entrée.	● Utiliser le mot de passe correct.
9	Erreur d'option	● Aucune imprimante ou enregistreur n'est raccordé et le mode PRT ou un ordre optionnel tel que SAVE est donné.	● Raccorder une imprimante ou un enregistreur à cassette ● Annuler le mode PRT

Liste des commandes de programme

Classification	Nom de la commande	Format	Fonction
Instruction d'entrée	INP	INP chaîne de variables	Entraîne l'entrée des données à partir du clavier pendant l'exécution d'un programme. L'exécution du programme est arrêtée jusqu'à la fin de l'entrée. **p. 57**
	KEY	Variable de caractère = KEY	Lit un caractère entré pendant l'exécution d'un programme et l'affecte à une variable de caractère. Etant donné que le programme n'est pas arrêté par cette commande, rien n'est affecté à la variable de caractère si aucune entrée n'est faite par le clavier. **p. 58**
Instruction de sortie	PRT	PRT fonction de contrôle de sortie {;/,} élément de sortie [{;/,} ...]	Sort un élément de sortie spécifié sous un format spécifié. **p. 58**
	DMS	DMS variable	Convertit une valeur de variable en notation sexagégimale, et l'affiche. **p. 59**
	WAIT	WAIT n * $0 \leq n < 1000$	Détermine le temps d'affichage pour une instruction PRT ou une instruction DMS. **p. 59**
Branchement	GOTO	GOTO {numéro de ligne / variable}	Entraîne le saut au numéro de ligne spécifié. **p. 39**
	IF...{THEN / ;}...	IF comparaison {THEN numéro de ligne / ; commande}	Entraîne le saut au numéro de ligne suivant THEN, ou exécute la commande suivant " ;", si le résultat de la comparaison est vrai. Entraîne le passage au numéro de ligne suivant si le résultat de la comparaison est faux. **p. 44**
	GSB	GSB {numéro de ligne / variable}	Appelle le sous-programme avec le numéro de ligne spécifié pour exécution. Une fois que le sous-programme est exécuté, la commande revient à l'instruction GSB par l'instruction RET pour passer à la commande suivant cette instruction. **p. 64**
	RET	RET	Signifie la fin d'un sous programme; renvoie la commande à un numéro de ligne suivant l'instruction GSB. **p. 64**

Classification	Nom de la commande	Format	Fonction
Boucle	FOR	FOR $v = e_1$ TO e_2 [STEP e_3] * v signifie une variable numérique, et e_1, e_2 et e_3 représentent respectivement une expression numérique.	Déclare le début d'une boucle dans laquelle la valeur numérique v change de la valeur initiale e_1 à la valeur finale e_2 par incréments de e_3. La boucle est répétée "$\left(\frac{e_2 - e_1}{e_3}\right) + 1$" fois entre les instructions FOR et NEXT. Si l'incrément e_3 est omis, il est considéré égal à "1". **p. 49**
	NEXT	NEXT	Signifie la fin d'une boucle FOR. Si le résultat de v plus e_3 est égal ou inférieur à e_2, la boucle est encore répétée. Si ce résultat est supérieur à e_2, la commande passe à la ligne suivant l'instruction NEXT. **p. 49**
Arrêt d'exécution	STOP	STOP	Arrête temporairement l'exécution d'un programme pour amener le système dans un état d'attente d'entrée par le clavier. L'exécution peut être reprise par appui sur la touche [CONT]. **p. 37**
Fin d'exécution	END	END	Signifie la fin d'un programme, le système revenant à son état de pré-exécution. L'exécution d'un programme, une fois terminée, ne peut pas être reprise par appui sur la touche [CONT]. **p. 30**
Effacement de données	VAC	VAC	Efface toutes les données variables pour un programme. **p. 42**
Inventaire de programme	LIST	LIST [numéro de ligne]	Affiche en descendant le contenu de toutes les instructions d'un programme à partir du numéro de ligne spécifié. **p. 32**
Inventaire de données	LIST V	LIST V	Affiche les données situées dans les mémoires. **p. 78**
Inventaire de programme/ données	LIST ALL	LIST ALL	Affiche le contenu de toutes les instructions d'un programme et toutes les données situées dans les mémoires. **p. 78**
Exécution de programme	RUN	RUN [numéro de ligne]	Entraîne le démarrage d'un programme à partir du numéro de ligne spécifié. **p. 31**
Effacement de programme	CLR	CLR	Efface la zone programme spécifiée d'un programme. **p. 30**
	CLR ALL	CLR ALL	Efface tous les programmes. **p. 30**

Classification	Nom de la commande	Format	Fonction
Protection de programme	PASS	PASS "mot de passe"	Ouand un mot de passe est entré, les opérations telles que LIST et LOAD ne peuvent pas être exécutées. Si un mot de passe a déjà été entré, il est annulé. **p. 74**

Fiche technique

■ **Type**

FX-702P

■ **Fonctions de calcul élémentaires**

Nombre négatifs, exposants, opérations arithmétiques y compris paranthèses (addition, soustraction, multiplication et division, avec fonction discriminatoire d'ordre de priorité – logique algébrique véritable)

■ **Fonctions incorporées**

Fonctions trigonométriques et trigonométriques inverses (mesures angulaires en degrés, radians ou grades), fonctions hyperboliques et hyperboliques inverses, logarithmiques et exponentielles, factorielle, racine carrée, puissances, conversion de coordonnées, réduction à l'entier, suppression de l'entier, valeur absolue, symbolisation, conversion décimal ↔ sexagésimal, désignation du nombre de chiffres significatifs, désignation de décimales, nombres aléatoires, π.

■ **Fonctions statistiques**

Ecart standard: nombre de données, somme, somme des carrés, moyenne, écarts standard (2 types)

Régression linéaire: nombre de données, somme des x, somme des y, somme des carrés de x, somme des carrés de y, somme du produit des données, moyenne de x, moyenne de y, écart standard de x (2 types), écart standard de y (2 types), terme constant, coefficient de régression, coefficient de corrélation, valeur estimée de x, valeur estimée de y.

■ **Instructions**

INP, KEY, PRT, DMS, IF-THEN, GOTO, FOR, NEXT, GSB, RET, WAIT, STAT, DEL, SAC, MODE, SET, STOP, END, SAVE, SAVE ALL, LOAD, LOAD ALL, GET, PUT, VER, PASS, RUN, LIST, LIST V, LIST ALL, CLR, CLR ALL

■ **Fonctions de caractères**

LEN, MID

■ **Fonctions de contrôle de sortie**

CSR

■ **Gamme de calcul**

$\pm 1 \times 10^{-99}$ à $\pm 9{,}999999999 \times 10^{99}$ et 0

Calculs internes à mantisse à 12 chiffres

Capacité des fonctions	Gamme d'entrée	Précision du résult
$\sin x$, $\cos x$, $\tan x$	$\lvert x \rvert < 1440°$ (8π rad, 1600gra)	10è chiffre ±1
$\sin^{-1} x$, $\cos^{-1} x$	$\lvert x \rvert \leqq 1$	– " –
$\tan^{-1} x$		– " –
$\sinh x$, $\cosh x$	$\lvert x \rvert \leqq 230$	– " –
$\tanh x$		– " –
$\sinh^{-1} x$	$\lvert x \rvert \leqq 10^{99}$	– " –
$\cosh^{-1} x$	$1 \leqq x \leqq 10^{99}$	– " –
$\tanh^{-1} x$	$\lvert x \rvert < 1$	– " –
$\log x$, $\ln x$	$x > 0$	– " –
e^x	$\lvert x \rvert \leqq 230$	– " –
$\sqrt{x}$	$x \geqq 0$	– " –
$x!$	$x \leqq 69$	– " –
x^y ($x \uparrow y$)	Quand $x < 0$, y est un nombre naturel	– " –
R → P	$\lvert x \rvert \leqq 10^{99}$, $\lvert y \rvert \leqq 10^{99}$	– " –
P → R	$\lvert \theta \rvert < 1440°$ (8π rad, 1600gra)	– " –
Décimal → sexagésimal	Dans les ±99999,9999999	– " –

■ **Système de programmation**

Système à stockage

■ **Langage de programmation**

BASIC

■ **Nombre de pas**

De 80 au maximum de 1680 pas (avec alimentation)

■ **Nombre de programmes enregistrés**

Maximum de 10 groupes (P0 à P9)

■ **Nombre de mémoires**

De 26 au maximim de 226 mémoires plus variable de caractères exclusive ($) (avec alimentation)

■ **Nombre de piles**

Sous-programme — 10 niveaux
Boucle FOR•NEXT — 8 niveaux
Valeur numérique — 10 niveaux
Eléments de calcul — 20 niveaux

■ **Système et méthode d'affichage**

Mantisse à 10 chiffres (y compris signe –), partie exponentielle à 2 chiffres, affichage à cristaux liquides, affichage sexagésimal, affichage d'état F1, F2, ARC, HYP, RUN, WRT, STOP, DEG, RAD, GRA, TRACE, PRT.

■ **Eléments d'affichage**

Affichage à points matriciels à 20 caractères (cristaux liquides)

■ **Principaux composants**

C-MOS I SI et autres

■ **Consommation**

0,01W (max.)

■ **Alimentation**

2 piles au lithium (CR2032)
L'autonomie de la calculatrice en fonctionnement continu sur les piles CR2032 est d'environ 240 heures (200 heures avec options)

■ **Coupure automatique de l'alimentation**

L'alimentation est automatiquement coupée 8 minutes environ après la dernière opération.

■ **Plage de temérature ambiante**

0 à 40°C (32 à 104°F)

■ **Dimensions**

17H x 165L x 82mmP (5/8"H x 6-1/2"L x 3-1/4"P)

■ **Poids**

176g (6,2 oz) avec piles

CASIO®

024E SA Printed in Japan
Imprimé au Japon