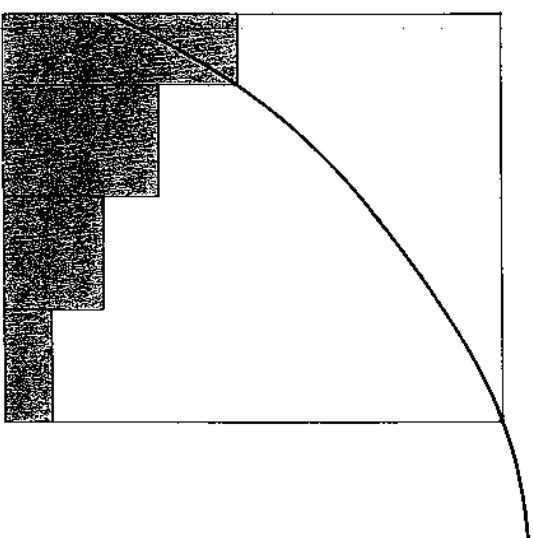


4. CALCULS PAR PROGRAMME

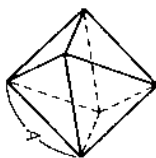


4-1 QU'EST-CE QU'UN PROGRAMME?

Cet appareil possède un dispositif de programmation incorporé qui facilite l'exécution des calculs répétés. Cette caractéristique est utilisée pour exécuter consécutivement des formules de la même manière que la caractéristique d'"instruction multiple", utilisée dans les calculs manuels. Dans cette partie, des programmes sont examinés à l'aide d'exemples explicatifs.

EXEMPLE:

Trouver la surface et le volume d'un octaèdre régulier en connaissant la longueur d'un de ses côtés.



Longueur d'un côté (A)	Surface (S)	Volume (V)
10cm	() cm ²	() cm ³
7	()	()
15	()	()

* Remplir les parenthèses.

① Formules

Si S est la surface, V le volume et A la longueur d'un côté d'un octaèdre régulier, S et V sont donnés par les formules suivantes:

$$S = 2\sqrt{3}A^2 \quad V = \frac{\sqrt{2}}{3}A^3$$

② Programmation

On appelle programmation l'élaboration d'un programme basé sur des formules de calcul. Un programme est élaboré ici à partir des formules ci-dessus. La base d'un programme est un calcul manuel. Aussi, examinons avant tout, la méthode d'exécution d'opérations utilisée pour les calculs manuels.

Surface (S): 2 \times 3 \times Valeur numérique A \rightarrow x^2 \rightarrow EXE

Volume (V): $\sqrt{2}$ 2 \div 3 \times Valeur numérique A \rightarrow x^3 3 \rightarrow EXE

Dans l'exemple ci-dessus, la valeur numérique A est utilisée deux fois. Il peut donc être intéressant de la stocker dans la mémoire A avant d'effectuer les calculs.

Valeur numérique A \rightarrow ALPHA A \rightarrow EXE

2 \times 3 \times ALPHA A \rightarrow x^2 \rightarrow EXE

$\sqrt{2}$ 2 \div 3 \times ALPHA A \rightarrow x^3 3 \rightarrow EXE V

Avec cet appareil, les opérations élaborées pour des calculs manuels peuvent être utilisées telles quelles dans un programme. Une fois que l'exécution d'un programme est lancée, elle se fera dans l'ordre sans s'arrêter. Toutefois, des commandes sont nécessaires pour demander l'entrée de données et pour afficher les résultats. La commande utilisée pour demander l'entrée de données est "?", alors que celle utilisée pour afficher les résultats est "▲". Un "?" placé dans un programme provoque un arrêt temporaire de son exécution et l'apparition d'un "?" sur l'affichage pour signaler que l'appareil est en attente d'entrée de données. Cette commande ne peut pas être utilisée seule mais doit l'être conjointement avec \rightarrow sous la forme "SHIFT ? \rightarrow nom de mémoire". Par exemple, pour stocker une valeur numérique dans la mémoire A:

? \rightarrow A

Quand "?" est affiché, les commandes de calcul et les valeurs numériques peuvent être entrées dans 111 pas.

La commande "▲" provoque un arrêt temporaire de l'exécution du programme et l'affichage du résultat de la dernière formule ou l'affichage de caractères et symboles alphanumériques (voir page 137).

Cette commande est utilisée pour signaler, dans les formules, les endroits auxquels les résultats doivent être affichés. Etant donné que le résultat final est automatiquement affiché lorsque le programme se termine, cette commande peut être omise à la fin d'un programme.

De toute façon, si le mode Base-n est spécifié pour la conversion de base d'un programme, ne pas oublier le "▲" final.

Ces deux commandes sont utilisées ici dans la procédure présentée précédemment:

SHIFT ? \rightarrow ALPHA A \rightarrow 2 \times 3 \times ALPHA A \rightarrow x^2 \rightarrow SHIFT ▲
 \rightarrow Entrée dans la mémoire A \rightarrow Affichage de S

$\sqrt{2}$ 2 \div 3 \times ALPHA A \rightarrow x^3 3 \rightarrow ▲ omis

Le programme est maintenant terminé.

③ Stockage du programme

Le stockage des programmes est effectué en mode WRT, spécifié en appuyant sur **MODE** **2** ("WRT" apparaît sur l'affichage).

Opération

MODE **2**

Affichage

```

sys mode : WRT
cal mode : COMP
angle : Deg
display : Norm

6566 Bytes Free

Prog 0123456789
  
```

Lorsque l'on a appuyé sur **MODE** **2**, "WRT" apparaît sur la partie supérieure gauche de l'affichage pour indiquer que l'appareil est placé en mode WRT. Le nombre de pas restants (voir page 113) est ensuite indiqué sur la partie supérieure droite de l'affichage. Le nombre de pas restants diminue lorsque des programmes sont entrés ou lorsque le nombre de mémoires est étendu. Si aucun programme n'est entré et si le nombre de mémoires est égal à 26 (nombre de mémoires à l'initialisation), le nombre de pas restants est égal à 6566.

Les plus grands chiffres situés au-dessous indiquent les zones de programme (voir page 115). Si les lettres "Prog" sont suivies des chiffres 0 à 9, cela indique qu'il n'y a pas de programmes stockés dans les zones P0 à P9. Le zéro clignotant indique ici que la zone de programme actuelle est P0. Les zones dans lesquelles des programmes ont déjà été stockés sont signalées par des "—" remplaçant les chiffres.

```

sys mode : WRT
cal mode : COMP
angle : Deg
display : Norm

6392 Bytes Free

Prog 01 34 6789
  
```

Le programme mentionné précédemment est ici stocké dans la zone P0 (indiquée par le zéro clignotant).

Opération

EXE (Le stockage commence)

Affichage

```

_

? -> A : 2 X √ 3 X A² 4 _

√ 2 ÷ 3 X A x³ 3

? -> A : 2 X √ 3 X A² 4
√ 2 ÷ 3 X A x³ 3
  
```

Lorsque ces opérations sont terminées, le programme est stocké.

* Le système d'affichage apparaît seulement pendant que la touche **MODE** est enfoncée.

MODE (S'affiche pendant que la touche est enfoncée)

```

**** MODE ****

sys mode : WRT
cal mode : COMP
angle : Deg
display : Norm

Step P0-20
  
```

* Lorsque le programme est stocké, appuyer sur **MODE** **1** pour retourner en mode RUN.

④ Exécution de programmes

Les programmes sont exécutés en mode RUN (**MODE** **1**). La zone de programme devant être exécutée est spécifiée à l'aide de la touche **Prog**.

Pour exécuter P0: **Prog** **0** **EXE**

Pour exécuter P3: **Prog** **3** **EXE**

Pour exécuter P8: **Prog** **8** **EXE**

Le programme de l'exemple qui a été stocké, est ici exécuté. La surface (S) et le volume (V) de l'octaèdre régulier du problème sont calculés de la manière suivante:

Longueur d'un côté (A)	Surface (S)	Volume (V)
10cm	(346.4101615)cm ²	(471.4045208)cm ³
7	(169.7409791)	(161.6917506)
15	(779.4228634)	(1590.990258)

Opération

MODE **1**

Affichage

```

**** MODE ****
sys mode : RUN
cal mode : COMP
angle : Deg
display : Norm
Step      0
  
```

Prog **0** **EXE**

```

7→A:2X√3XA²
√2÷3XA³
Prog 0
?
  
```

10 **EXE**
(Valeur de A)

```

?→A:2X√3XA²
√2÷3XA³
Prog 0
?
10
346.4101615
-Disp-
  
```

Indique la réponse affichée par **▲**.

EXE

Prog **0** **EXE**

```

7→A:2X√3XA²
√2÷3XA³
Prog 0
?
10
346.4101615
471.4045208
  
```

(V lorsque A = 10)

7 **EXE**

(Valeur de A)

```

√2÷3XA³
Prog 0
?
10
346.4101615
471.4045208
Prog 0
?
  
```

(S lorsque A = 7)

EXE

```

10
346.4101615
471.4045208
Prog 0
?
7
169.7409791
-Disp-
  
```

(V lorsque A = 7)

Prog 0 EXE

```

471.4045208
Prog 0
?
7
169.7409791
161.6917506
Prog 0
?

```

15 EXE
(Valeur de A)

```

7
169.7409791
161.6917506
Prog 0
?
15
779.4228634
-Disp-

```

(S lorsque A = 15)

EXE

```

7
169.7409791
161.6917506
Prog 0
?
15
779.4228634
1590.990258

```

(V lorsque A = 15)

* Les calculs programmés sont effectués automatiquement à chaque appui sur la touche EXE après l'entrée d'une donnée ou la lecture d'un résultat.

* Directement après, un programme dans la zone P0 est exécuté en appuyant sur Prog 0 EXE comme dans cet exemple, la commande Prog 0 est mémorisée par la fonction répétition. Par conséquent, les exécutions ultérieures du même programme peuvent être exécutées par un simple appui sur la touche EXE.

Opération

- Prog 0 EXE (Exécution du programme P0)
- 10 EXE (Introduire 10 pour A)
- EXE (Affiche V lorsque A = 10)
- EXE (Réexécute)
- 7 EXE (Introduire 7 pour A)
- EXE (Affiche V lorsque A = 7)
- ⋮

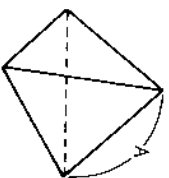
4-2 CONTRÔLE ET ÉDITION DE PROGRAMMES (CORRECTIONS, AJOUTS, SUPPRESSIONS)

On peut rappeler un programme stocké dans le but de vérifier son contenu. Après avoir spécifié la zone de programme désirée à l'aide de $\left[\leftarrow \right]$ ou $\left[\rightarrow \right]$ en mode WRT ($\left[\text{MODE} \right] \left[2 \right]$), le contenu du programme est affiché en appuyant sur la touche $\left[\text{EXE} \right]$. Une fois que le programme est affiché, on utilise la touche $\left[\leftarrow \right]$ (ou $\left[\rightarrow \right]$, $\left[\uparrow \right]$, $\left[\downarrow \right]$) pour le faire avancer pas à pas pour la vérification.

Lorsque le programme n'a pas été correctement stocké, l'édition peut également être effectuée en ajoutant ou en supprimant des parties de programme. Un nouveau programme est ici élaboré en contrôlant et éditant le programme de l'exemple précédent (surface et volume d'un octaèdre régulier).

EXEMPLE:

Trouver la surface et le volume d'un tétraèdre régulier en connaissant la longueur d'un de ses côtés.



Longueur d'un côté (A)	Surface (S)	Volume (V)
10 cm	() cm ²	() cm ³
7.5	()	()
20	()	()

① Formules

Si S est la surface, V le volume et A la longueur d'un côté d'un tétraèdre régulier, S et V sont donnés par les formules suivantes:

$$S = \sqrt{3} A^2 \quad V = \frac{\sqrt{2}}{12} A^3$$

② Programmation

Comme dans l'exemple précédent, la longueur d'un côté est stockée dans la mémoire A, puis le programme est élaboré.

Valeur numérique A → $\left[\text{ALPHA} \right] \left[A \right] \left[\text{EXE} \right]$
 $\left[\sqrt{} \right] \left[3 \right] \left[\text{ALPHA} \right] \left[A \right] \left[x^2 \right] \left[\text{EXE} \right]$ S
 $\left[\sqrt{} \right] \left[2 \right] \left[\div \right] \left[12 \right] \left[\text{ALPHA} \right] \left[A \right] \left[x^3 \right] \left[3 \right] \left[\text{EXE} \right]$ V

Lorsque ce qui précède est créé dans un programme, cela apparaît sous la forme suivante:

$\left[\text{SHIFT} \right] \left[? \right] \left[\rightarrow \right] \left[\text{ALPHA} \right] \left[A \right] \left[\leftarrow \right] \left[\sqrt{} \right] \left[3 \right] \left[\text{ALPHA} \right] \left[A \right] \left[x^2 \right] \left[\text{SHIFT} \right] \left[\leftarrow \right]$
 $\left[\sqrt{} \right] \left[2 \right] \left[\div \right] \left[12 \right] \left[\text{ALPHA} \right] \left[A \right] \left[x^3 \right] \left[3 \right]$

③ Edition du programme

Une comparaison des deux programmes peut être utile.

Octaèdre: $\left[\text{SHIFT} \right] \left[? \right] \left[\rightarrow \right] \left[\text{ALPHA} \right] \left[A \right] \left[\leftarrow \right] \left[2 \right] \left[\times \right] \left[\sqrt{} \right] \left[3 \right] \left[\text{ALPHA} \right] \left[A \right] \left[x^2 \right] \left[\text{SHIFT} \right] \left[\leftarrow \right] \left[\sqrt{} \right] \left[2 \right] \left[\div \right] \left[12 \right]$

Tétraèdre: $\left[\text{SHIFT} \right] \left[? \right] \left[\rightarrow \right] \left[\text{ALPHA} \right] \left[A \right] \left[\leftarrow \right] \left[\sqrt{} \right] \left[3 \right] \left[\text{ALPHA} \right] \left[A \right] \left[x^2 \right] \left[\text{SHIFT} \right] \left[\leftarrow \right] \left[\sqrt{} \right] \left[2 \right] \left[\div \right] \left[12 \right]$
 $\left[\text{ALPHA} \right] \left[A \right] \left[x^3 \right] \left[3 \right]$

Le programme de l'octaèdre peut être changé en un programme de tétraèdre en supprimant les parties marquées de lignes ondulées et en modifiant celles marquées de lignes droites.

Pratiquement, cela peut être effectué de la manière suivante:

Opération

Affichage

Opération	Affichage
$\left[\text{MODE} \right] \left[2 \right]$	sys mode : WRT cal mode : COMP angle : Deg display : Norm 6546 Bytes Free Prog _123456789

$\left[\text{EXE} \right]$

$\left[\sqrt{} \right] \left[2 \right] \left[\div \right] \left[3 \right] \left[\text{ALPHA} \right] \left[A \right] \left[x^3 \right] \left[3 \right]$

Le curseur est situé au début. Appuyer sur $\left[\text{SHIFT} \right] \left[\text{EXE} \right]$ pour amener le curseur à la fin.

$\left[\leftarrow \right] \left[\leftarrow \right] \left[\leftarrow \right] \left[\leftarrow \right]$
 $\left[\text{DEL} \right] \left[\text{DEL} \right]$

$\left[\sqrt{} \right] \left[2 \right] \left[\div \right] \left[3 \right] \left[\text{ALPHA} \right] \left[A \right] \left[x^3 \right] \left[3 \right]$

Placer le curseur à la position à supprimer et supprimer deux caractères.

$\left[\leftarrow \right] \left[\leftarrow \right] \left[\leftarrow \right] \left[\leftarrow \right]$
 $\left[\text{INS} \right] \left[12 \right]$

$\left[\sqrt{} \right] \left[2 \right] \left[\div \right] \left[3 \right] \left[\text{ALPHA} \right] \left[A \right] \left[x^3 \right] \left[3 \right]$

Insérer deux caractères

$\left[\text{DEL} \right]$

$\left[\sqrt{} \right] \left[2 \right] \left[\div \right] \left[3 \right] \left[\text{ALPHA} \right] \left[A \right] \left[x^3 \right] \left[3 \right]$

Supprimer le 3 inutile.

MODE 1

```
**** MODE ****
sys mode : RUN
cal mode : COMP
angle : Deg
display : Norm
Step 0
```

Edition terminée. Re-
tourner en mode RUN.

EXE

4 Exécution de programmes

Ce programme est maintenant exécuté.

Longueur d'un côté (A)	Surface (S)	Volume (V)
10 cm	(173.2050808)cm ²	(117.8511302)cm ³
7.5	(97.42785793)	(49.71844555)
20	(692.820323)	(942.8090416)

Opération

Affichage

MODE 1

```
**** MODE ****
sys mode : RUN
cal mode : COMP
angle : Deg
display : Norm
Step 0
```

Prog 0 EXE

```
?→A:√3XA²
√2÷12XA³
Prog 0
?
```

10 EXE

```
?→A:√3XA²
√2÷12XA³
Prog 0
?
10
173.2050808
— Disp —
```

Prog 0 EXE

```
?→A:√3XA²
√2÷12XA³
Prog 0
?
10
173.2050808
117.8511302
```

7.5 EXE

```
√2÷12XA³
Prog 0
?
10
173.2050808
117.8511302
Prog 0
?
```

EXE

```
10
173.2050808
117.8511302
Prog 0
?
7.5
97.42785793
— Disp —
```

```
10
173.2050808
117.8511302
Prog 0
?
7.5
97.42785793
49.71844555
```

⟨Sommaire⟩

117.8511302

Prog 0

١٤٤

7.5

97.42785793

49.7184455

Prog 0

٢٠

20 EXE

7.5

97.42785793

49.7184455

Prog 0

2

20

692.820323

- Disp -

EXE

7.5

97.42785793

49.7184455

Prøg

2

20





692.820323

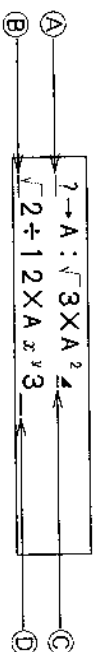
942.8090416





Opération		Touches utilisées	
Contrôle du programme	<ul style="list-style-type: none"> • Spécification du mode WRT • Spécification de la zone de programme (omise si P0) • Commencer la vérification 	<div>MODE [2]</div> <div>← →</div>	
	<ul style="list-style-type: none"> • Vérification du contenu 	<div>EXE</div> <div>← → ↕ ↕</div>	
Correction	<ul style="list-style-type: none"> • Placer le curseur à la position à corriger. • Appuyer sur les touches adéquates. 	<div>← → ↕ ↕</div>	
	<ul style="list-style-type: none"> • Placer le curseur à la position à supprimer. 	<div>← → ↕ ↕</div>	
Suppression	<ul style="list-style-type: none"> • Supprimer 	<div>DEL</div>	
	<ul style="list-style-type: none"> • Placer le curseur à la position à laquelle l'insertion doit être effectuée. • Spécifier le mode inséré. • Appuyer sur les touches souhaitées. 	<div>← → ↕ ↕</div>	<div>SHIFT INS</div>

<REFERENCE>

Mouvement du curseur

Appuyer sur les touches de curseur (, , , ) entraîne le déplacement du curseur suivant:



Position du curseur				
(A) invalide		1 position à droite	invalide	1 ligne en dessous (B)
(B) 1 position à gauche	(C) 1 position à droite	1 position à droite	1 ligne au dessus (A)	Fin de ligne (D)
(C) 1 position à gauche	1 position à droite (B)	Début de ligne (A)	1 ligne en dessous (D)	
(D) 1 position à gauche	invalide	1 ligne au dessus (C)	invalide	

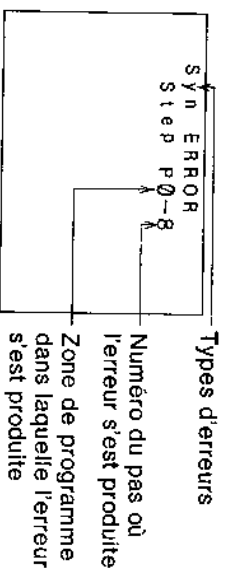
4-3 MISE AU POINT DE PROGRAMMES (CORRECTION DES ERREURS)

Après qu'un programme ait été élaboré et entré, son exécution génère parfois des messages d'erreurs ou donne des résultats inattendus. Cela indique qu'il y a des erreurs dans le programme. Ces dernières doivent être corrigées.

La correction de telles erreurs de programmation est appelée mise au point.

■ Mise au point lorsqu'un message d'erreur est généré

Un message d'erreur est affiché de la manière suivante:



Le message d'erreur informe l'opérateur de la zone de programme (P0 à P9) dans laquelle l'erreur s'est produite.

Il mentionne également le type de l'erreur, ce qui donne une idée des mesures appropriées à prendre pour la corriger. Le numéro de pas indiqué à quel pas de la zone de programme l'erreur s'est produite. Il y a six messages d'erreurs au total.

■ Messages d'erreur

Il y a six messages d'erreur au total.

① Syn ERROR (Erreur de syntaxe)

Indique une erreur dans la formule ou une mauvaise utilisation des commandes de programmation.

② Ma ERROR (Erreur mathématique)

Indique un résultat de calcul d'une expression numérique dépassant 10^{100} , une opération illogique (p.ex. une division par zéro), l'entrée d'un argument dépassant le domaine de définition de la fonction.

③ Go ERROR (Erreur de saut)

Indique qu'une commande GOTO ne renvoie à aucune commande Lb1 (voir page 120) ou pour la commande Prog (voir page 128), que la zone de programme (voir page 115) ne contient pas de programme.

④ Ne ERROR (Erreur d'emboîtement)

Indique un dépassement du nombre d'emboîtements de sous-programmes par la commande Prog.

⑤ Stk ERROR (Erreur de pile)

Indique que le calcul effectué dépasse la capacité de la pile des valeurs numériques ou de celle des commandes (voir page 16).

⑥ Mem ERROR (Erreur de mémoire)

Indique une tentative d'utilisation d'un nom de mémoire tel que Z[5] sans que le nombre de mémoires n'ait été étendu.

⑦ Arg ERROR: Erreur d'argumentation

S'affiche lorsque l'argument de commande ou de spécification dans un programme excède la plage d'entrée (ex.Sci 10, Goto 11).

La suite d'une opération devient impossible lorsqu'un message d'erreur est affiché.

Appuyer sur $\left[\text{AC} \right] \Rightarrow$ ou $\left[\Rightarrow \right]$ pour supprimer l'erreur.

Une pression sur $\left[\text{AC} \right]$ supprime l'erreur et l'utilisation des touches est à nouveau possible. Pendant cette opération, le mode RUN est conservé.

Une pression sur $\left[\Rightarrow \right]$ ou $\left[\Rightarrow \right]$ supprime l'erreur et change le mode système en mode WRT. Le curseur est situé à l'emplacement où l'erreur s'est produite pour permettre une modification du programme et éliminer l'erreur.

■ Points de contrôle de chaque type d'erreur

Les points de contrôle de chaque type d'erreur sont les suivants:

- ① **Syn ERROR**
Vérifier de nouveau qu'il n'y a pas d'erreur dans le programme.
- ② **Ma ERROR**
En ce qui concerne les calculs qui requièrent l'utilisation des mémoires, contrôler que les valeurs numériques stockées dans les mémoires ne dépassent pas le domaine de définition des arguments. Ce type d'erreur apparaît souvent lors d'une division par zéro ou d'un calcul de racine carrée d'une valeur négative.
- ③ **Go ERROR**
Contrôler qu'il y a bien une commande Lbl *n* correspondante lorsqu'un Goto *n* est utilisé. Contrôler également que le programme de la zone *P n* a été correctement entré lorsque Prog *n* est utilisée.
- ④ **Ne ERROR**
Contrôler qu'une commande Prog n'est pas utilisée dans la zone de programme à laquelle l'exécution a été branchée pour la faire revenir à la première zone de programme.
- ⑤ **Stk ERROR**
Contrôler que la formule n'est pas trop longue, provoquant ainsi un débordement de la pile. Si c'est le cas, la formule doit être divisée en au moins deux parties.
- ⑥ **Mem ERROR**
Contrôler que le nombre de mémoires a été correctement étendu à l'aide de "MODE [2] n EXE" (Defm). Lorsque des mémoires de type tableau (voir page 00) sont utilisées, contrôler que les indices sont corrects.
- ⑦ **Arg ERROR**
Vérifier que les valeurs indiquées par "MODE [7] (Fix) ou [8] (Sci) sont contenues dans la plage de 0 à 9. Vérifier aussi que les valeurs indiquées par Goto, Lbl, ou des programmes de commande sont contenus entre 0 et 9. S'assurer aussi à l'aide de "MODE [3] (Defm) que l'extension de mémoire est inférieure ou égale au nombre de pas restants, et que la valeur utilisée pour l'extension n'est pas négative.

4-4 COMPTAGE DU NOMBRE DE PAS

La capacité de cet appareil est de 6566 pas. Le nombre de pas indique le montant de l'espace de stockage disponible pour les programmes. Il diminue lorsque des programmes sont entrés. Le nombre de pas restants diminue également lorsque des pas sont convertis en mémoires. (Voir page 25.)

Il y a deux méthodes utilisées pour déterminer le nombre actuel de pas restants:

- ① Lorsque l'on appuie sur "MODE [3] EXE" en mode RUN, le nombre de pas restants est affiché conjointement avec le nombre de mémoires.

Exemple:

Defm	
Program : 19	← Nombre de pas utilisés pour la programmation
Memory : 26	← Nombre de mémoires
6547 Bytes Free	← Nombre de pas restants

- ② Spécifier le mode WRT "MODE [2]" pour faire apparaître le nombre de pas restants sur la partie supérieure droite de l'affichage. A cet instant l'état des zones de programme peut également être examiné.

MODE [2]	
sys mode : WRT	
cal mode : COMP	
angle : Deg	
display : Norm	
6547 Bytes Free	← Nombre de pas restants
Prog : 123456789	

Fondamentalement, une fonction ne requiert qu'un seul pas, mais il y a quelques commandes dans lesquelles une fonction en requiert deux.

- Une fonction/un pas: sin, cos, tan, log, {, }, :, A, B, 1, 2, 3, etc.
- Une fonction/deux pas: Lbl 1, Goto 2, Prog 8, etc.

Chaque pas peut être contrôlé en déplaçant le curseur:

Exemple:

Position actuelle du curseur →

2 → A : $\sqrt{3 \times A^2}$
 $\sqrt{2 + 12 \times A \times 3}$

A cet instant, chaque appui sur une touche de déplacement du curseur (↵ ou ←) provoque un déplacement de ce dernier au pas précédent ou suivant. Par exemple:

2 → A : $\sqrt{3 \times A^2}$
 $\sqrt{2 + 12 \times A \times 3}$ même pas

L'affichage montrera à quel pas du programme le curseur est actuellement situé aussi longtemps que la touche [EXE] est enfoncée

[EXE] ~
 (Maintenir enfoncée)

```

**** MODE ****
sys mode : WRT
cal mode : COMP
angle : Deg
display : Norm
Step P0-9
  
```

Indique la position du curseur au même pas

4-5 ZONES DE PROGRAMME ET MODES DE CALCUL

Cet appareil comporte un total de 10 zones de programme (P0 à P9) utilisées pour le stockage des programmes.

Ces zones de programme sont toutes utilisées de la même manière, 10 programmes indépendants peuvent y être entrés. Un programme principal et plusieurs programmes secondaires (sous-programmes) peuvent également être stockés.

Le nombre total de pas utilisables pour le stockage dans les zones de programme P0 à P9 est de 6566 au maximum.

La spécification d'une zone de programme est effectuée de la manière suivante:

Mode RUN: Appuyer sur une touche de 0 à 9 après avoir appuyé sur la

touche [Prog]. Appuyer ensuite sur [EXE].

Exemple: P 0 [Prog] [0] [EXE]
 P 8 [Prog] [8] [EXE]

* Dans ce mode, l'exécution d'un programme commence lorsque l'on a appuyé sur [EXE].

Mode WRT: Utiliser [↵] ou [←] pour déplacer le curseur sous le numéro de la zone de programme devant être spécifiée puis appuyer sur [EXE].

Seuls les numéros des zones de programme ne contenant pas encore de programme sont affichés. Les symboles "—" indiquent les zones de programme qui contiennent déjà un programme.

Exemple:

```

sys mode : WRT
cal mode : COMP
angle : Deg
display : Norm
6547 Bytes Free
Prog 123 67 9
  
```

Des programmes sont déjà mémorisés dans ces zones de programme.

■ Zone de programme et spécification de mode de calcul en mode WRT

Mis à part les calculs normaux de fonctions pour effectuer des calculs binaires, octaux, décimaux et hexadécimaux, des conversions, des calculs d'écarts-types et de régressions dans un programme, un mode de calcul doit être spécifié. Les spécifications du mode de calcul et de la zone de programme sont effectuées en même temps.

Le mode WRT (**MODE** **2**) est spécifié le premier, puis le mode de calcul.

Ensuite, la zone de programme est spécifiée et, lorsque l'on a appuyé sur la touche **EXE**, le mode de calcul est mémorisé dans la zone de programme. D'ores et déjà, les programmes stockés seront accompagnés du mode de calcul.

Exemple: Mémoriser le mode Base-n dans la zone P2

MODE **2**

```
sys mode : WRT
cal mode : COMP
angle : Deg
display : Norm
6566 Bytes Free
Prog 0123456789
```

(En supposant que rien n'est stocké.)

→ **→**

```
sys mode : WRT
cal mode : COMP
angle : Deg
display : Norm
6566 Bytes Free
Prog 0123456789
```

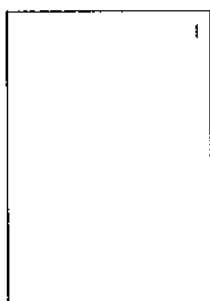
Spécifier P2

MODE **=**

```
sys mode : WRT
cal mode : Base-n
Dec
6566 Bytes Free
Prog 0123456789
```

(Spécifier le mode Base-n.)

EXE



Comme indiqué ci-dessus, le mode de calcul est mémorisé dans une zone de programme.

■ Avertissement concernant les modes de calcul

La totalité des opérations utilisables dans chaque mode de calcul peut être stockée en tant que programmes, mais certaines commandes ou fonctions ne peuvent pas être utilisées dans tous les modes de calculs.

Mode Base-n

- Les calculs de fonctions ne peuvent pas être effectués.
- Les unités de mesure d'angle ne peuvent pas être spécifiées.
- Toutes les commandes de programmation peuvent être utilisées.
- Contrôler l'insertion du "▲" à la sortie du résultat final pour revenir au mode de calcul précédent lorsque le programme est exécuté. L'oubli pourrait entraîner un affichage décimal ou une erreur.

Mode SD1, SD2

- Parmi les fonctions, Abs et $\sqrt{\quad}$ ne peuvent pas être utilisées.
- Parmi les commandes de programmation, Dsz, >, et, < ne peuvent pas être utilisées.

Mode LR1, LR2

- Parmi les fonctions, Abs et $\sqrt{\quad}$ ne peuvent pas être utilisées.
- Parmi les commandes de programmation, \Rightarrow , =, \neq , Isz, \geq , \leq , Dsz, > et < ne peuvent pas être utilisées.

4-6 EFFACEMENT DE PROGRAMMES

L'effacement de programmes est effectué en mode PCL. Appuyer sur **MODE** **3** pour spécifier le mode PCL. "PCL" apparaît sur l'affichage. Il y a deux méthodes utilisées pour effacer des programmes: effacer le programme d'une seule zone de programme et effacer tous les programmes.

■ Effacement d'un seul programme

Pour effacer le programme d'une seule zone de programme, spécifier le mode PCL et appuyer sur **AC** après avoir spécifié ladite zone.

Exemple: Effacer seulement le programme de la zone P3.

Opération

Affichage

MODE **3**

sys mode	:	PCL
cal mode	:	COMP
angle	:	Deg
display	:	Norm
6468 Bytes Free		
Prog	:	2_45678

P0, P3 et P9 contiennent déjà des programmes.

⇒ **⇒** **⇒**

sys mode	:	PCL
cal mode	:	COMP
angle	:	Deg
display	:	Norm
6468 Bytes Free		
Prog	:	1_2_45678

Aligner le curseur avec P3.

AC

sys mode	:	PCL
cal mode	:	COMP
angle	:	Deg
display	:	Norm
6511 Bytes Free		
Prog	:	1_2345678

Le nombre trois apparaît après la suppression.

MODE **1**

Revenir au mode RUN

**** MODE ****		
sys mode	:	RUN
cal mode	:	COMP
angle	:	Deg
display	:	Norm
Step	:	0

■ Effacement de tous les programmes

Pour effacer tous les programmes stockés dans les zones de programme 0 à 9, spécifier le mode PCL, appuyer sur **SHIFT** puis sur **DEL**.

Exemple: Supprimer les programmes stockés dans les zones P0, P4, P8, et P9.

Opération

Affichage

MODE **3**

sys mode	:	PCL
cal mode	:	COMP
angle	:	Deg
display	:	Norm
6439 Bytes Free		
Prog	:	123_567

SHIFT **DEL**

sys mode	:	PCL
cal mode	:	COMP
angle	:	Deg
display	:	Norm
6566 Bytes Free		
Prog	:	0_123456789

MODE **1**

**** MODE ****		
sys mode	:	RUN
cal mode	:	COMP
angle	:	Deg
display	:	Norm
Step	:	0

4-7 COMMANDES DE PROGRAMMATION PRATIQUES

Les programmes destinés à cet appareil sont basés sur les calculs manuels. Néanmoins, des commandes de programmation spéciales sont utilisables pour permettre de choisir une formule et d'exécuter répétitivement la même formule.

Quelques unes de ces commandes sont décrites dans ce paragraphe pour permettre d'élaborer des programmes plus pratiques.

■ Commandes de saut

Les commandes de saut sont utilisées pour modifier le déroulement de l'exécution d'un programme. Les programmes sont exécutés dans l'ordre dans lequel ils sont entrés (à partir du plus petit numéro de pas) jusqu'à ce que la fin du programme soit atteinte. Cette méthode n'est pas très pratique lorsque des calculs répétés doivent être effectués ou lorsqu'il est souhaitable de faire passer l'exécution à une autre formule. C'est par conséquent dans ces cas que les commandes de saut sont très efficaces. Il y a trois types de commande de saut : un simple saut inconditionnel à une destination de branchement, un saut conditionnel qui décide de la destination de branchement en vérifiant qu'une condition est vraie ou non et un saut déterminé par un compteur, qui augmente ou diminue de un le contenu d'une mémoire particulière, puis décide de la destination de branchement après avoir contrôlé si la valeur stockée est égale à zéro ou non.

◆ Saut inconditionnel

Le saut inconditionnel est composé d'un "Goto" et d'un "Lbl". Lorsque l'exécution du programme atteint l'instruction "Goto n" (dans laquelle n est un chiffre de 0 à 9), elle passe alors à l'instruction "Lbl n" (n a la même valeur que dans l'instruction "Goto n"). Le saut inconditionnel est souvent utilisé dans les programmes simples pour reprendre l'exécution à partir du début, pour effectuer des calculs répétés ou pour répéter des calculs à partir d'un point du programme. Les sauts inconditionnels sont également utilisés en combinaison avec sauts conditionnels et déterminés par compteur.

Exemple: Le programme précédent utilisé pour trouver la surface et le volume d'un tétraèdre régulier est réécrit en utilisant "Goto 1" et "Lbl 1" pour permettre d'effectuer des calculs répétés.

Le précédent programme contenait :

? → A, ; √, 3, X, A, x², ▲,
√, 2, ÷, 1, 2, X, A, x³, 3

19 pas

* *Ci-après, des virgules (,) sont utilisées pour séparer les pas par souci de clarté.*

Ajouter "Goto 1" à la fin du programme et "Lbl 1" au début comme destination de branchement.

Néanmoins, si on laisse le programme dans cet état, le volume ne sera pas affiché et l'exécution passera immédiatement à l'entrée d'un côté au début du programme. Pour éviter cette situation, insérer une commande d'affichage (▲) avant le "Goto 1".

Le programme terminé par le saut inconditionnel ajouté se présente comme suit :

Lbl, 1, ; ? → A, ; √, 3, X, A, x², ▲,
√, 2, ÷, 1, 2, X, A, x³, 3, ▲, Goto, 1

25 pas

Essayons maintenant d'exécuter ce programme.

* *Pour plus de détails concernant l'entrée et l'édition de programmes, voir les paragraphes 4-1 et 4-2.*

* *Désormais les affichages ne montreront que la sortie du résultat du calcul.*

Opération		Affichage	
Prog	EXE	?	Stocké dans la zone P0
10	EXE	173.2050808	La longueur du côté est égale à 10
EXE		117.8511302	
EXE		?	
7.5	EXE	97.42785793	La longueur du côté est égale à 7.5
EXE		49.71844555	
EXE		?	

Étant donné que le programme est une boucle sans fin, il continuera de s'exécuter. Pour terminer son exécution, appuyer sur **MODE** **LT**.

MODE **LT**

```

**** MODE ****
sys mode : RUN
cal mode : COMP
angle : Deg
display : Norm
Step 0

```

Outre le début du programme, des destinations de branchement peuvent être spécifiées à des points quelconques du programme.

Exemple: Calculer $y = ax + b$ lorsque la valeur de x change à chaque passage tandis que les valeurs de a et b peuvent également changer en fonction du calcul.

Programme

```

? , →, A, :, ?, →, B, :, Lbl, 1, :, ?, →, X, :,
A, X, X, +, B, ▴, Goto, 1
23 pas

```

Lorsque ce programme est exécuté, les valeurs de a et de b sont stockées respectivement dans les mémoires A et B. Après cela, seule la valeur de x peut être changée.

Pour cela, un saut inconditionnel est fait en fonction de "Goto" et "Lbl" et le déroulement de l'exécution du programme est changé. Lorsqu'il n'y a pas de "Lbl n " correspondant à un "Goto n ", une erreur (Go ERROR) se produit.

◆ Saut conditionnel

Le saut conditionnel compare une valeur numérique d'une mémoire avec une constante ou une valeur numérique d'une autre mémoire. Si la condition est vraie, l'instruction suivant le "⇒" est exécutée et si la condition est fausse, l'exécution saute cette instruction et reprend après le "⇐", le "⇐" ou le "▴" suivant.

Les sauts conditionnels ont la forme suivante :

Partie gauche	Opérateur de relation	Partie droite	⇒	Instruction { : } *	Instruction
---------------	-----------------------	---------------	---	---------------------	-------------

* ⇐ représente la fonction de retour de chariot (voir page 128).
* N'importe lequel peut être utilisé.

Des noms de mémoire (un caractère alphabétique de A à Z), des valeurs numériques constantes ou des formules de calcul ($A \times 2$, $D - E$, etc.) peuvent être utilisés dans les parties gauche et droite.

L'opérateur de relation est un symbole de comparaison. Il y a six types d'opérateurs de relation : =, ≠, ≥, ≤, >, <.

Partie gauche = partie droite (la partie gauche est égale à la partie droite)

Partie gauche ≠ partie droite (la partie gauche n'est pas égale à la partie droite)

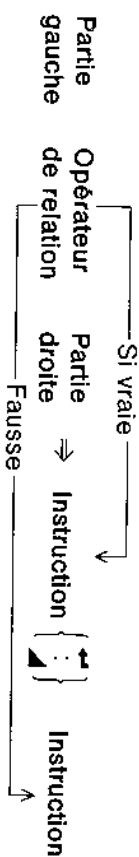
Partie gauche ≥ partie droite (la partie gauche est supérieure ou égale à la partie droite)

Partie gauche ≤ partie droite (la partie gauche est supérieure à la partie droite)

Partie gauche > partie droite (la partie gauche est supérieure à la partie droite)

Partie gauche < partie droite (la partie gauche est inférieure à la partie droite)

Le "⇒" est affiché lorsque l'on a appuyé sur **SHIFT** **▽**. Si la condition est vraie, l'exécution passe à l'instruction située après le "⇒". Si la condition est fausse, l'instruction située après le "⇒" est sautée et l'exécution passe à l'instruction située après le "⇐", "⇐" ou le "▴" suivant.



Une instruction est une formule de calcul ($\sin A \times 5$, etc.) ou une commande de programmation (Goto, Prog, etc.)
Tout ce qui précède le "⇐", "⇐" ou le "▴" suivant est considéré comme une instruction.

Exemple: Si une valeur numérique supérieure ou égale à zéro est entrée, calculer la racine carrée de cette valeur. Si la valeur numérique entrée est inférieure à zéro, entrer une nouvelle valeur.

Programme

```

Lbl, 1, :, :, ? , →, A, :, A, ≥, 0, ⇒, √, A, ▴, Goto, 1 16 pas
Dans ce programme, la valeur numérique entrée est stockée dans la mémoire A, puis est testée pour déterminer si elle est supérieure, égale ou inférieure à zéro.

```

Si le contenu de la mémoire A est supérieur ou égal à zéro (pas inférieur à zéro), l'instruction (formule de calcul) située entre " \Rightarrow " et " \blacktriangleleft " est exécutée, puis "Goto 1" renvoie l'exécution à "Lbl 1". Si le contenu de la mémoire A est inférieur à zéro, l'exécution passe à l'instruction située après le " \blacktriangleleft " suivant et est renvoyée à "Lbl 1" par "Goto 1".

Exemple: Calculer la somme des valeurs numériques entrées. Le total doit être affiché si un zéro est entré.

Programme

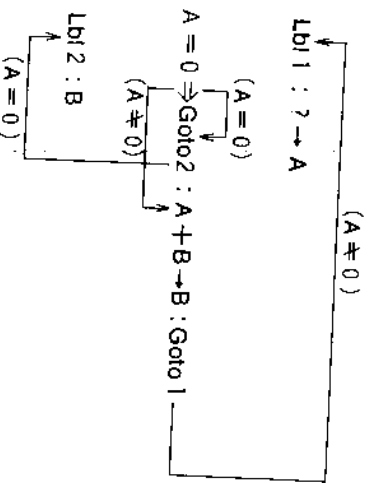
```
0, →, B, ;  
Lbl, 1, ;, ?, →, A, ;, A, =, 0, ⇒, Goto, 2, ;,  
A, +, B, →, B, ;, Goto, 1, ;,  
Lbl, 2, ;, B
```

31 pas

Dans ce programme, un zéro est d'abord entré dans la mémoire B pour l'initialiser afin d'effectuer le calcul de la somme. Ensuite, la valeur entrée par "? → A" est stockée dans la mémoire A et par "A = 0 ⇒", on détermine si la valeur stockée dans la mémoire A est égale à zéro ou non. Si A = 0, "Goto 2" provoque un saut de l'exécution à "Lbl 2". Si la mémoire A n'est pas égale à zéro, "Goto 2" est sautée et la commande A + B → B suivant ";" est exécutée. "Goto 1" renvoie ensuite l'exécution à "Lbl 1".

L'exécution à partir de "Lbl 2" affiche la somme qui a été stockée dans la mémoire B. La commande d'affichage " \blacktriangleleft " est placée à la suite de B, mais ici, elle peut être omise.

L'illustration suivante indique le déroulement du programme :



▣ Saut déterminé par compteur

Le saut déterminé par compteur provoque l'incrément ou la décrémentation de 1 de la valeur de la mémoire spécifiée. Si cette valeur n'est pas égale à zéro, l'instruction suivante est sautée et celle située après le " \blacktriangleleft ", ";" ou le " \blacktriangleleft " suivant est exécutée. La commande "lsz" est utilisée pour incrémenter de 1 la valeur de la mémoire et décider de l'exécution ultérieure tandis que la commande "Dsz" est utilisée pour décrémenter cette valeur de 1 et décider de l'exécution ultérieure.

Contenu de la mémoire $\neq 0$

lsz	Nom de la mémoire	: Instruction	\blacktriangleleft	Instruction
		Contenu de la mémoire $= 0$		

Contenu de la mémoire $\neq 0$

Dsz	Nom de la mémoire	: Instruction	\blacktriangleleft	Instruction
		Contenu de la mémoire $= 0$		

Exemple: Incrémenter la mémoire A de 1 lsz A

Décrémenter la mémoire A de 1 Dsz B

Exemple: Déterminer la moyenne de 10 valeurs numériques entrées.

Programme

```
1, 0, →, A, ;, 0, →, C, ;  
Lbl, 1, ;, ?, →, B, ;, B, +, C, →, C, ;,  
Dsz, A, ;, Goto, 1, ;, C, ÷, 1, 0
```

32 pas

Dans ce programme, 10 est d'abord stocké dans la mémoire A et 0 dans la mémoire C. La mémoire A est utilisée comme "compteur" et une décrémentation est effectuée le nombre de fois spécifié par la commande Dsz. La mémoire C est utilisée pour stocker la somme des valeurs entrées et doit donc d'abord être initialisée à zéro. La valeur numérique entrée en réponse à "?" est stockée dans la mémoire B, puis est ajoutée à la somme des valeurs entrées dans la mémoire C par "B + C → C". L'instruction Dsz A décrémenté ensuite de 1 la valeur stockée dans la mémoire A. Si le résultat est différent de zéro, l'instruction suivante, "Goto 1", est exécutée. Si le résultat est égal à zéro, l'instruction suivante, "Goto 1", est sautée et l'exécution passe à "C ÷ 10".

Exemple: Déterminer, à des intervalles d'une seconde, l'altitude d'une balle lancée en l'air à une vitesse initiale de $V_{m/s}$ et sous un angle de S° . La formule est la suivante: $h = V \sin \theta t - \frac{1}{2}gt$, avec $g = 9,8$ et les effets de la résistance de l'air ignorés.

Programme

Deg, :, 0, →, T, :, ?, →, V, :, ?, →, S, :,
Lbl, 1, :, Isz, T, :, V, X, sin, S, X, T, -,
9, ., 8, X, T, x^2 , ÷, 2, ▴, Goto, 1

38 pas

Dans ce programme, l'unité de mesure d'angle est spécifiée et la mémoire T est initialisée (effacée). La vitesse initiale et l'angle sont en "Lbl 1" est placée au début du programme pour effectuer des calculs répétés. La valeur numérique stockée dans la mémoire T est initialement de 1 par Isz T. Dans ce cas, la commande Isz est utilisée uniquement dans le but d'incrémenter la valeur stockée dans la mémoire T et le saut ultérieur ne dépend d'aucune comparaison ou décision. La commande Isz peut être utilisée de la même manière que la commande Dsz pour effectuer des sauts nécessitant des décisions, mais aussi, valeurs. Si au lieu de la commande Isz, une autre méthode telle que "T + 1 → T" est utilisée, cinq pas sont nécessaires au lieu de deux requis par la méthode (Isz T) indiquée ici. De telles commandes sont pratiques pour conserver l'espace mémoire.

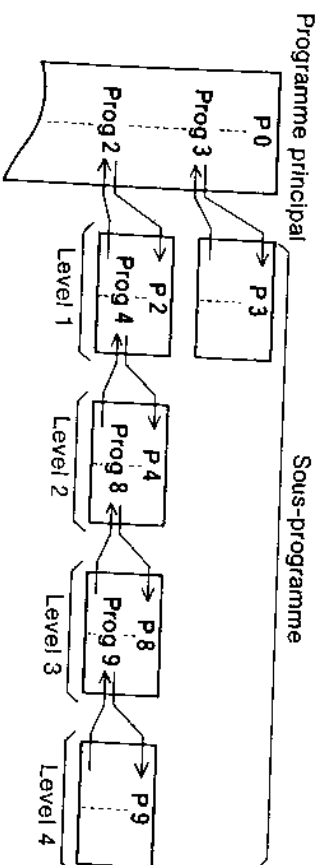
Chaque fois que la mémoire T est incrémentée, le calcul est effectué d'après la formule et l'attitude est affichée. Il est à noter que ce programme n'a pas de fin. Aussi, lorsque la valeur requise est obtenue, appuyer sur **MODE** **□** pour terminer le programme.

(Sommaire)

Commande	Formule	Opération
Saut inconditionnel	Lbl n Goto n (entier naturel de n = 0 à 9)	Effectue un saut inconditionnel à la commande "Lbl n" correspondant à la commande "Goto n"
Saut conditionnel	Partie gauche de relation droite Instruction { : } Instruction (Opérateurs de relation: =, ≠, >, <, ≥, ≤)	Les parties gauche et droite sont comparées. Si l'expression conditionnelle est vraie, l'instruction située après ⇒ est exécutée. Si elle est fausse, l'exécution passe à l'instruction située après le "←", " " ou le "▴" suivant. Les instructions peuvent contenir des expressions numériques, des commandes Goto, etc.
Saut déterminé par comparateur	Isz Nom de mémoire : Instruction { : } Instruction Dsz Nom de mémoire : Instruction { : } Instruction (Un nom de mémoire est constitué d'une seule lettre de A à Z, A [], etc.)	La valeur numérique stockée dans la mémoire est incrémentée (Isz) ou décrémentée (Dsz) de un. Si le résultat est égal à zéro, un saut est effectué à l'instruction située après le "←", " " ou le "▴" suivant. Les instructions peuvent contenir des expressions numériques, des commandes Goto, etc.

■ Sous-programmes

Un programme contenu dans une seule zone de programme est appelé un programme principal. Les parties de programme fréquemment utilisées et stockées dans d'autres zones de programme sont appelées des sous-programmes. Les sous-programmes peuvent être utilisés de différentes façons pour faciliter les calculs. Ils peuvent être utilisés pour stocker des formules dans le but d'effectuer des calculs répétés, sous la forme d'un bloc auquel on se branche à chaque calcul, ou pour stocker des formules ou des opérations fréquemment utilisées pour pouvoir les appeler au moment voulu.



La commande de sous-programme est "Prog" suivie d'un chiffre de 0 à 9 qui indique la zone de programme.

Exemple: Prog 0 Saut à la zone de programme 0
Prog 2 Saut à la zone de programme 2

Après avoir effectué le saut à l'aide de la commande "Prog", l'exécution se poursuit à partir du début du programme stocké dans la zone de programme spécifiée. Lorsque la fin du sous-programme est atteinte, l'exécution retourne à l'instruction suivant la commande "Prog n" du premier programme. Des sauts peuvent être effectués d'un sous-programme à un autre. Cette procédure est appelée "emboîtement". Il est possible d'effectuer jusqu'à 10 niveaux d'emboîtement. Toute tentative de dépasser cette limite provoque une erreur (Ne ERROR). Tenter, à l'aide d'une commande "Prog", d'effectuer un saut dans une zone de programme ne contenant pas de programme provoque également une erreur (Go ERROR).

* Une commande "Goto n" contenue dans un sous-programme effectue un saut à la commande "Lbi n" correspondante de cette zone de programme.

Exemple : Exécuter simultanément les deux programmes présentés précédemment pour calculer les surfaces et volumes d'un octaèdre et d'un tétraèdre réguliers.

Exprimer les résultats avec trois décimales.

Cet exemple utilise les deux programmes expliqués précédemment. La première étape consiste à entrer le nombre spécifié de décimales (MODE [7] [3]).

Revenons maintenant les deux premiers programmes.

Octaèdre régulier

P0 Fix, 3, :, ?, → A, :, 2, X, √, 3, X, A, x², ▲,

23 pas

Tétraèdre régulier

P1 Fix, 3, :, ?, → A, :, √, 3, X, A, x², ▲,

22 pas

Total: 45 pas

Si l'on compare les deux programmes, il est évident que les parties soulignées sont identiques. Si ces parties sont intégrées dans un même sous-programme, les programmes sont simplifiés et le nombre de pas nécessaires réduit.

En outre, les parties signalées par des lignes ondulées ne sont pas identiques telles qu'elles figurent, mais si P1 est modifiée en : √, 2, ÷, 3, X, A, x², 3, ÷, 4, elles le deviennent.

Les parties soulignées par des lignes droites sont maintenant stockées dans la zone P9 sous la forme d'un programme indépendant et celles soulignées par les lignes ondulées le sont dans la zone P8.

P9 Fix, 3, :, ?, → A, :, √, 3, X, A, x² 12 pas
P8 √, 2, ÷, 3, X, A, x², 3 8 pas

Après que les parties communes aient été enlevées, le reste de la formule de l'octaèdre régulier est stocké dans la zone P0 et celui de la formule du tétraèdre régulier dans la zone P1. Les commandes "Prog 9" et "Prog 8" doivent bien sûr être ajoutées pour permettre d'effectuer des sauts aux sous-programmes P9 et P8.

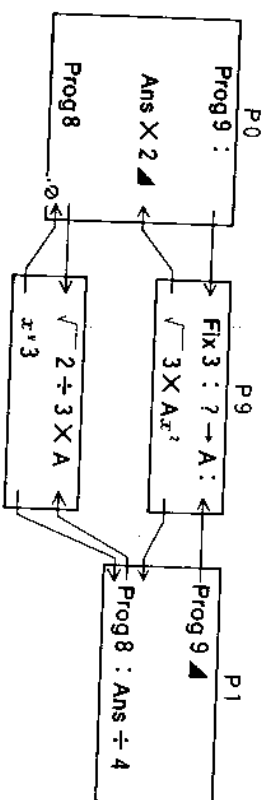
P0 Prog, 9, :, Ans, X, 2, ▲, Prog, 8 9 pas
P1 Prog, 9, ▲, Prog, 8, :, Ans, ÷, 4 9 pas

Total: 38 pas

Avec cette configuration, l'exécution passe au programme P9 au début des programmes P0 et P1, trois décimales sont spécifiées, la valeur d'un côté est entrée et la surface du tétraèdre est calculée. L'expression "2X" de la première formule de l'octaèdre a été omise dans P9, ainsi lorsque l'exécution retourne à P0, "AnsX2" permet d'obtenir la surface de l'octaèdre. Dans le cas de P1, le résultat de P9 ne demande pas d'autre modification et est donc immédiatement affiché au moment de revenir à P1.

Le calcul des volumes est effectué de la même manière. Après avoir effectué un saut à P8 pour faire le calcul, l'exécution retourne aux programmes principaux. Dans la zone P0, le programme prend fin après que le volume de l'octaèdre soit affiché. Néanmoins, obtenir le volume du tétraèdre. En utilisant des sous-programmes de cette manière, on a réduit le nombre de pas et rendu les programmes nets et faciles à lire.

L'illustration suivante indique le déroulement du programme dont on vient de faire état.



En isolant les parties communes des deux premiers programmes et en les stockant dans des zones de programmes séparées, on a réduit le nombre de pas et donné une configuration claire aux programmes.

■ Fonction retour à la ligne

Avec la fonction retour à la ligne, [EXE] est utilisée à la place de [↵] pour séparer les commandes afin d'avoir un affichage plus facile à lire.

```

Deg : 0 -> T : ? -> V : ? -> S :
Lb l 1 : 1 : 1 s z T : V X s i
n SXT - 9 . 8 X T² ÷ 2
Goto 1
  
```

L'emploi de la fonction retour à la ligne dans le programme indiqué ci-dessus produit la suite de l'affichage suivant

```

Deg V
0 -> T : ? -> V : ? -> S :
Lb l 1 : 1 : 1 s z T : V X s i
n SXT - 9 . 8 X T² ÷ 2
Goto 1
  
```

La touche [EXE] est enfoncée dans ces deux endroits. Rien n'est affiché au point où elle est enfoncée, et l'affichage passe à la ligne suivante.

Ceci rend l'établissement des opérations d'unité d'angle et de boucle plus facile à suivre.

Procédure d'utilisation

[MODE] [4] [EXE] (appuyer à la place de [↵])
 [0] [→] [ALPHA] [7] [↵] [SHIFT] [7] [→] [ALPHA] [V] [↵] [SHIFT] [7] [→] [ALPHA] [S] [EXE]
 [SHIFT] [Lb l] [1] [1] [↵] ...
 * Pour inclure la fonction retour à la ligne dans un programme qui a déjà été entré, appuyer tout d'abord sur [SHIFT] [INS] pour spécifier le mode d'insertion, puis appuyer sur [EXE]. Ensuite, supprimer le " : ".

```

Deg : 0 -> T : ? -> V : ? -> S :
Lb l 1 : 1 : 1 s z T : V X s i
n SXT - 9 . 8 X T² ÷ 2
Goto 1
  
```

Aligner le curseur avec le " : " suivant "Deg" et appuyer sur [SHIFT] [INS] [EXE].

```

Deg
[↵] 0 -> T : ? -> V : ? -> S : Lb l
1 : 1 s z T : V X s i n S
XT - 9 . 8 X T² ÷ 2
Goto 1
  
```

Supprimer le " : ".

```

Deg
0 -> T : ? -> V : ? -> S : Lb l
1 : 1 s z T : V X s i n S
T - 9 . 8 X T² ÷ 2
Goto 1
  
```

Aligner le curseur avec le " : " suivant "? -> S". Comme ci-dessus, insérer tout d'abord [EXE], puis supprimer le " : ".

```

Deg
0 -> T : ? -> V : ? -> S
[Lb l] 1 : 1 : 1 s z T : V X s i
n SXT - 9 . 8 X T² ÷ 2
Goto 1
  
```

* La fonction retour à la ligne peut être utilisée dans les opérations manuelles en appuyant sur [SHIFT] [EXE].

4-8 MEMOIRES DE TYPE TABLEAU

■ En utilisant des mémoires de type tableau

Toutes les mémoires utilisées jusqu'ici ont été désignées par un seul caractère alphabétique tel que A, B, X, ou Y.

Dans le cas de la mémoire de type tableau introduite ici, un indice tel que [1] ou [2] est joint au nom de mémoire (un caractère alphabétique de A à Z).

* Les crochets sont entrés par **ALPHA** et **ALPHA** **EXP**.

Mémoire standard	Mémoire de type tableau
A	A[0]
B	A[1]
C	A[2]
D	A[3]
E	A[4]
	C[-2]
	C[-1]
	C[0]
	C[1]
	C[2]

L'utilisation appropriée des indices raccourcit les programmes et les rend plus facile à utiliser. Les indices négatifs sont considérés comme étant ramenés à 0 et donc utilisés par la mémoire 0 comme indiqué ci-dessus.

Exemple: Entrer les nombres 1 à 10 dans les mémoires A à J.

En utilisant des mémoires standard

```
1, → A, ; 2, → B, ; 3, → C, ; 4, → D, ;
5, → E, ; 6, → F, ; 7, → G, ; 8, → H, ;
9, → I, ; 1, 0, → J
```

40 pas

En utilisant des mémoires de type tableau

```
0, → Z, ; Lbl, 1, ; Z, +, 1, → A, [Z,], ;
Isz, Z, ; Z, <, 1, 0, → Goto, 1
```

26 pas

Dans le cas de l'utilisation de mémoires standard, entrer les valeurs une par une dans les mémoires n'est pas très efficace et constitue une perte de temps. Que se passe-t-il si l'on veut voir une valeur stockée dans une mémoire particulière ?

En utilisant des mémoires standard

```
Lbl, 1, ; ? , → Z, ;
Z, =, 1, → A, ▴, Z, =, 2, → B, ▴,
Z, =, 3, → C, ▴, Z, =, 4, → D, ▴,
Z, =, 5, → E, ▴, Z, =, 6, → F, ▴,
Z, =, 7, → G, ▴, Z, =, 8, → H, ▴,
Z, =, 9, → I, ▴, Z, =, 1, 0, → J, ▴,
Goto, 1
```

70 pas

En utilisant des mémoires de type tableau

```
Lbl, 1, ; ? , → Z, ; A, [Z, - 1,], ▴,
Goto, 1
```

16 pas

La différence saute aux yeux. Lorsque l'on utilise des mémoires standard, la valeur entrée est comparée aux valeurs affectées à chaque mémoire (p. ex. A=1, B=2, ...).

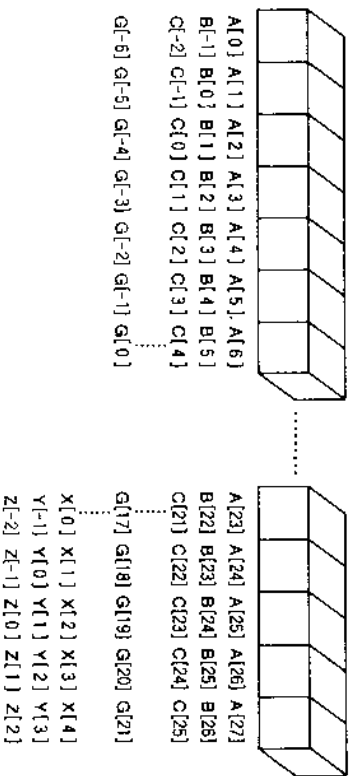
Avec les mémoires de type tableau, la valeur entrée est immédiatement stockée dans la mémoire adéquate déterminée par "Z-1".

Des formules (Z-1, A+10, etc.) peuvent même être utilisées comme indice.

■ Précautions à prendre lors de l'utilisation de mémoires de type tableau

Lorsque l'on utilise des mémoires de type tableau, un indice est joint au caractère alphabétique de A à Z qui représente une mémoire standard. Par conséquent, il faut prendre soin de pas provoquer de dépassement du nombre de mémoires.

La relation est la suivante:



Le point suivant présente un cas dans lequel des mémoires de type tableau provoquent un dépassement du nombre de mémoires standard. Cette situation doit toujours être évitée.

Exemple: Stocker les valeurs numériques 1 à 5 dans les mémoires A[1] à A[5].

```
5, →, C, :, Lbl, 1, :, C, →, A, [, C, ], :,
Dsz, C, :, Goto, 1, :,
A, [, 1, ], ▲, A, [, 2, ], ▲, A, [, 3, ], ▲,
A, [, 4, ], ▲, A, [, 5, ],
```

44 pas

Dans ce programme, les valeurs 1 à 5 sont stockées dans les mémoires de type tableau A[1] à A[5] et la mémoire C est utilisée comme compteur. Lorsque ce programme est exécuté, le résultat suivant est obtenu:

Opération

Affichage

Prog 0	EXE	1.
EXE		0.
EXE		3.
EXE		4.
EXE		5.

Comme on peut le voir, la deuxième valeur affichée (qui doit être 2) provenant de A[2] n'est pas correcte. Ce problème se produit car A[2] et C constituent la même mémoire.

A B C D E F
A [1] A [2] A [3] A [4] A [5]

Le contenu de la mémoire C (A[2]) est passé de 5 à 0 par décrétement de 1. Par conséquent, l'affichage du contenu de la mémoire A[2] est 0.

■ Applications des mémoires de type tableau

Il est parfois nécessaire de traiter deux différents types de données en un seul groupe.

Exemple: Stocker des données x et y dans des mémoires.

Lorsqu'une valeur x est entrée, la valeur correspondante y est affichée. Il y a un total de 15 données.

Programme d'exemple 1

La mémoire A est utilisée comme mémoire de gestion des données et la mémoire B pour le stockage temporaire des données x. Les données x sont stockées dans les mémoires C[1] (mémoire D) à C[15] (mémoire R) et les données y dans les mémoires C[16] (mémoire S) à C[30] (mémoire Z(7)).

```
1, →, A, :, Defm, 7, :,
Lbl, 1, :, ?, →, C, [, A, ], :,
?, →, C, [, A, +, 1, 5, ], :,
Isz, A, :, A, =, 1, 6, ⇒, Goto, 2, :, Goto, 1, :,
Lbl, 2, :, 1, 5, →, A, :, ?, →, B, :,
B, =, 0, ⇒, Goto, 5, :,
Lbl, 3, :, B, =, C, [, A, ], ⇒, Goto, 4, :,
Dsz, A, :, Goto, 3, :, Goto, 2, :,
Lbl, 4, :, C, [, A, +, 1, 5, ], ▲, Goto, 2, :,
Lbl, 5
```

98 pas

Dans ce programme, les mémoires sont utilisées de la manière suivante:

Données x

C [1] C [2] C [3] C [4] C [5] C [6] C [7] C [8]
D E F G H I J K
C [9] C [10] C [11] C [12] C [13] C [14] C [15]
L M N O P Q R

Données y

C [16] C [17] C [18] C [19] C [20] C [21] C [22] C [23]
S T U V W X Y Z
C [24] C [25] C [26] C [27] C [28] C [29] C [30]
Z (1) Z (2) Z (3) Z (4) Z (5) Z (6) Z (7)

Programme d'exemple 2

Les mêmes mémoires que dans l'exemple 1 sont ici utilisées, mais deux types de noms de mémoires sont employés et les données x et y sont séparées.

```
1, →, A, :, Defm, 7, :
Lb1, 1, :, ?, →, C, [, A, ], :,
?, →, R, [, A, ], :
lsz, A, :, A, =, 1, 6, ⇒, Goto, 2, :, Goto, 1, :,
Lb1, 2, :, 1, 5, →, A, :, ?, →, B, :,
B, =, 0, ⇒, Goto, 5, :
Lb1, 3, :, B, =, C, [, A, ], ⇒, Goto, 4, :,
Dsz, A, :, Goto, 3, :, Goto, 2, :,
Lb1, 4, :, R, [, A, ], ▲, Goto, 2, :,
Lb1, 5
```

92 pas

Les mémoires sont utilisées de la manière suivante:

Données x

```
C [1] C [2] C [3] C [4] C [5] C [6] C [7] C [8]
D      E      F      G      H      I      J      K
C [9] C [10] C [11] C [12] C [13] C [14] C [15]
L      M      N      O      P      Q      R
```

Données y

```
R [1] R [2] R [3] R [4] R [5] R [6] R [7] R [8]
S      T      U      V      W      X      Y      Z
R [9] R [10] R [11] R [12] R [13] R [14] R [15]
Z (1) Z (2) Z (3) Z (4) Z (5) Z (6) Z (7)
```

De cette façon, les noms des mémoires peuvent être changés. Néanmoins, étant donné que lesdits noms sont limités aux lettres de A à Z, les mémoires étendues MODE □ ne peuvent être utilisées que sous la forme de mémoires de type tableau.

* La commande d'extension de la mémoire (Defm) peut être utilisée dans un programme.

Exemple: Etendre le nombre de mémoires de 14 pour pouvoir en utiliser 40.

```
Defm, 1, 4, ,...
```

4-9 AFFICHAGE DE CARACTERES ET DE SYMBOLES ALPHANUMERIQUES

Des caractères alphabétiques, des chiffres, des symboles de commande de calcul, etc. peuvent être affichés sous la forme de messages. Ils sont placés entre guillemets ALPHA Prog.

■ Caractères et symboles alphanumériques

● Caractères et symboles affichés lorsque l'on appuie sur leurs touches à la suite de ALPHA:

```
[ ], K, m, μ, n, p, f, space (espace),
A, B, C, D, E, F, G, H, I, J, K, L, M, N,
O, P, Q, R, S, T, U, V, W, X, Y, Z
```

● Autres nombres, symboles, commandes de calcul, commandes de programme.

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
(, ), √, ε, +, −, X, ÷, ...
sin, cos, tan, log, ln, ...
=, ≠, ≥, ≤, >, <, ...
A, IB, C, ID, IE, F, d, h, b, o
Neg, Not, and, or, xor
x̄, ȳ, xσn, xσn-1, ...
0(SHIFT MODE 24), 1(SHIFT MODE 5), 9(SHIFT MODE 5)
```

* Tous les caractères indiqués ci-dessus peuvent être utilisés de la même manière que les caractères alphabétiques.

Dans l'exemple précédent nécessitant l'entrée de deux types de données (x , y), l'affichage "?" ne donne aucune information concernant le type de l'entrée souhaitée. Un message peut être inséré avant le "?" pour contrôler le type de la donnée que l'on désire entrer.

```
Lb1, 1, :, ?, →, X, :, ?, →, Y, :, ...
```

Les messages "X=" et "Y=" sont insérés dans ce programme.

```
Lb1, 1, :, "X, =", ?, →, X, :,
"Y, =", ?, →, Y, :, ...
```

Si des messages sont insérés comme indiqué ci-dessus, l'affichage est le suivant:

(En supposant que le programme est stocké dans la zone P1)

Prog 1 EXE
10 EXE

X = ?
Y = ?

Les messages sont aussi pratiques lors de l'affichage de résultats dans des calculs programmés.

Exemple:

```
Lbl, 0, :, "N = ", ? , → B, ~, C, :,
0, → A, :,
Lbl, 1, :, C, ÷, 2, → C, :, Frac, C, ÷, 0, →, Goto, 3,
:, Isz, A, :, C, =, 1, →, Goto, 2, :, Goto, 1, :,
Lbl, 2, :, "X = ", A, →, Goto, 0, :,
Lbl, 3, :, "N, O", →, Goto, 0
```

70 pas

Ce programme calcule la puissance x ième de 2. Un message "N=?" apparaît pour entrer des données. Le résultat est affiché en appuyant sur [EXE] lorsque "X=" est affiché. Lorsqu'une donnée entrée n'est pas une puissance x ième de 2, l'affichage "NO" apparaît et l'exécution revient au début du programme pour demander une nouvelle entrée.

En supposant que le programme est stocké dans la zone P2 :

Prog 2 EXE
4096 EXE
EXE
EXE
3124 EXE
EXE
512 EXE
EXE

N = ?
X =
12.
N = ?
NO
N = ?
X =
9.

Les chaînes de plus de 16 caractères sont affichées sur deux lignes. Quand les caractères alphabétiques sont affichés à la fin de la première ligne, l'affichage entier se déplace vers le haut et la ligne supérieure disparaît de l'écran.

Prog 0

123+45	168.
852-87	765.
968+125-65	1028.
Prog 0	

123+45	168.
852-87	765.
968+125-65	1028.
Prog 0	
ABCDEFGHIJKLMN	

↑ Après un instant.

852-87	168.
968+125-65	765.
	1028.
Prog 0	
ABCDEFGHIJKLMN	
QRSTUVWXYZ	

4-10 UTILISATION DE LA FONCTION GRAPHÉ DANS DES PROGRAMMES

L'utilisation de la fonction graphé dans des programmes donne la possibilité de représenter des équations longues et complexes et de superposer des graphes de façon répétitive.

Généralement, toutes les commandes de graphé (exceptée la fonction tracé) peuvent être incluses dans des programmes sans modification.

Ex. 1) Déterminer graphiquement le nombre de solutions (racines réelles) qui répondent à chacune des deux équations suivantes.

$$y = x^4 - x^3 - 24x^2 + 4x + 80$$

$$y = 10x - 30$$

Les valeurs de plage sont comme suit :

```

Range
Xmin:-10.
max:10.
sol:2.
Ymin:-120.
max:150.
sol:50.

```

Tout d'abord, programmer les domaines de plage. Noter que les valeurs sont séparées les unes des autres par des virgules.

Range, (-), 1, 0, 1, 1, 0, 1, 2, 1, (-), 1, 2, 0, 1, 5, 0, 1, 5, 0

Maintenant, programmer l'équation pour le premier graphe.

Graph, X, x⁴, 4, -, X, x³, 3, -, 2, 4, X, x², +, 4, X, +, 8, 0

Enfin, programmer l'équation pour le second graphe.

Graph, 1, 0, X, -, 3, 0

49 pas

Lorsque l'on rentre ce programme, appuyer sur **EXE** après l'entrée des plages et la première équation.

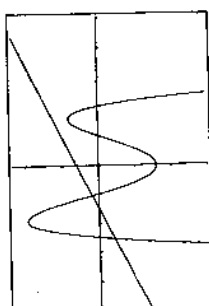
```

Range -10,10,2,-
120,150,50
Graph Y=X4-X3-24X2+4X+80
Graph Y=10X-30

```

L'affichage suivant doit apparaître lorsque le programme est exécuté.

Prog 0 **EXE**



Un "▲" peut être entré au lieu d'appuyer sur **EXE** après la première équation pour suspendre l'exécution après que le premier graphe soit produit. Pour continuer l'exécution du graphe suivant, appuyer sur **EXE**.

La procédure décrite ci-dessus peut être utilisée pour produire une grande variété de graphes. La bibliothèque située à la fin de ce manuel inclut un nombre d'exemples de programmation de graphe.