

 Olympia

OP 545

LA PROGRAMMATION
N'EST QU'UN JEU D'ENFANT

 Olympia

Olympia OP 545

LA PROGRAMMATION N'EST QU'UN JEU D'ENFANT



 Olympia

TABLE DES MATIERES

CHAPITRE 1 GUIDE GENERAL

1-1	NOMENCLATURE ET OPERATION	12
1-2	ALIMENTATION	19
1-3	MODULE D'EXTENSION DE MEMOIRE VIVE (RAM) (OPTION)	21
1-4	EXTENSION MEMOIRE	22
1-5	AVANT D'EFFECTUER DES CALCULS	24

CHAPITRE 2 PASSONS A L'EMPLOI

2-1	PASSONS A L'EMPLOI DE L'ORDINATEUR	28
2-2	PRATIQUE FONCTION BANQUE DE DONNEES	32
2-3	POUR COMMENCER, DES CALCULS SIMPLES	33
2-4	CALCULS DE FONCTIONS — UN POINT IMPORTANT DE CET ORDINATEUR	35

CHAPITRE 3 PROGRAMMATION EN BASIC

3-1	QU'EST-CE QU'UN PROGRAMME?	42
3-2	PREPARATION D'UN PROGRAMME	45
3-3	DEVELOPPEMENT DU PROGRAMME	64
3-4	EQUIPEMENTS OPTIONNELS PRATIQUES	105
3-5	UTILISATION D'UN PROGRAMME DE L'OP544	112

CHAPITRE 4 REFERENCE DE COMMANDES

NEW [ALL]	119
RUN	120
LIST	121
PASS	122

TABLE DES MATIERES

SAVE [ALL]	123
LOAD [ALL]	124
VERIFY	125
CLEAR	125
END	126
STOP	126
LET	126
REM	127
INPUT	127
KEY\$	128
PRINT	129
CSR	130
GOTO	131
ON — GOTO	132
IF — THEN	133
FOR — NEXT	134
GOSUB	135
RETURN	136
ON — GOSUB	136
DATA	137
READ	138
RESTORE	139
PUT	140
GET	140
BEEP	141
DEFM	142
MODE	143

SET	144
LEN	145
MID\$	146
VAL	147
STR\$	148
SIN, COS, TAN	149
ASN, ACS, ATN	150
LOG, LN	150
EXP	151
SQR	151
ABS	152
SGN	152
INT	152
FRAC	153
RND	153
RAN #	154
DEG	154
DM\$	155
COMMANDES DE BANQUE DE DONNEES	156
NEW #	156
LIST #	156
SAVE #	157
LOAD #	157
READ #	158
RESTORE #	159
WRITE #	161

CHAPITRE 5 BIBLIOTHEQUE DE PROGRAMMES

1. CONVERSION DE NOMBRES DECIMAUX EN NOMBRES HEXADECIMAUX	164
2. CALCULS DE PRET (MENSUALITES EGALES)	166
3. EXERCICES D'ARITHMETIQUE	169
4. ANALYSE DE REGRESSION DU SECOND DEGRE	171
5. JEU DE REMISE EN ORDRE	173

CHAPITRE 6 DOCUMENTATION

6-1 TABLEAU DE MESSAGES D'ERREUR	176
6-2 TABLEAU DES CODES DE CARACTERES	177
6-3 SYMBOLES D'ORGANIGRAMME	178
6-4 TABLE DES ELEMENTS DE TABLEAU	180

INDEX DES COMMANDES/FONCTIONS	181
-------------------------------------	-----

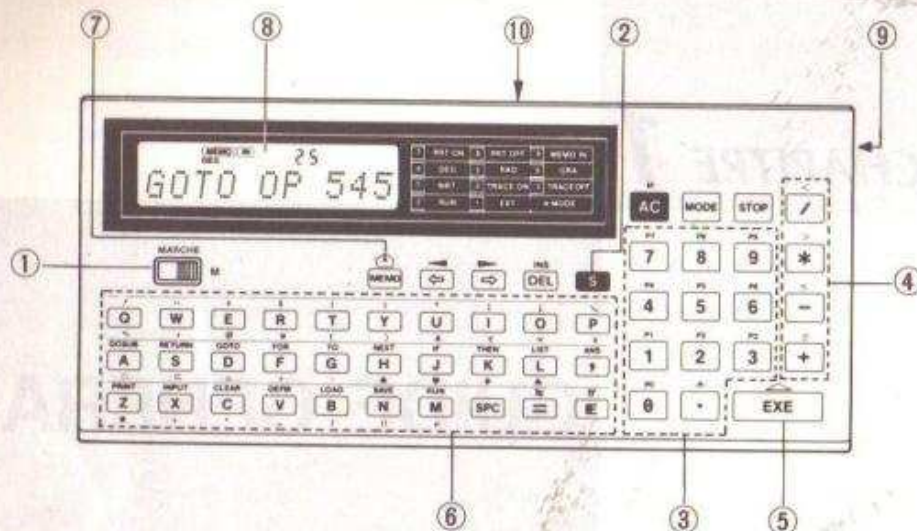
CARACTERISTIQUES	182
------------------------	-----

CHAPITRE 1

GUIDE GENERAL

Ce chapitre doit être lu autant par ceux qui n'ont jamais utilisé d'ordinateur que par ceux qui y sont déjà habitués.

1-1. NOMENCLATURE ET OPERATION



- | | |
|--|-------------------------------------|
| ① Interrupteur d'alimentation | ⑥ Touches alphabétiques |
| ② Touche de déplacement | ⑦ Touche de mémorisation |
| ③ Touches numériques et de point décimal | ⑧ Fenêtre d'affichage |
| ④ Touches de calcul | ⑨ Commande de contraste d'affichage |
| ⑤ Touche d'exécution | ⑩ Partie connecteur |

Comparé à des calculatrices ordinaires, cet ordinateur possède beaucoup plus de touches. La fonction de chaque touche est maintenant expliquée.

• Interrupteur d'alimentation

La mise sous tension est effectuée lorsque cet interrupteur est tourné vers la droite et l'alimentation est coupée lorsqu'il est tourné vers la gauche.

• Touche de déplacement (Touche rouge S)

Si on appuie sur cette touche, le mode déplacement est sélectionné ("S" est affiché) et la commande ou le symbole écrit au-dessus de chaque touche peut être affiché. Lorsque l'on appuie de nouveau sur cette touche, le mode déplacement est annulé et "S" disparaît. (Pour distinguer cette touche de la touche alphabétique S, elle sera écrite à partir de maintenant sous la forme **S** dans ce manuel.)

• Touches numériques, touche de point décimal, touches de calculs et touche d'exécution

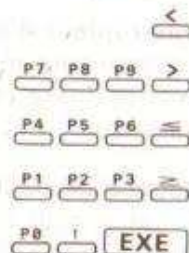
Examiner ce tableau de touches avec attention. C'est le même que sur une calculatrice ordinaire. Cette partie est utilisée pour effectuer les quatre opérations arithmétiques (addition, soustraction, multiplication, division). Néanmoins, il existe les différences suivantes. Les touches **x** (multiplication) et **÷** (division) sont différentes et il n'y a pas de touche **=** alors qu'il y a une touche **EXE** (exécution). Cela vient du fait qu'un ordinateur utilise un * (astérisque) et / (barre de fraction) respectivement pour $\times$ et $\div$ et la réponse est obtenue en appuyant sur la touche **EXE** au lieu de la touche **=**.



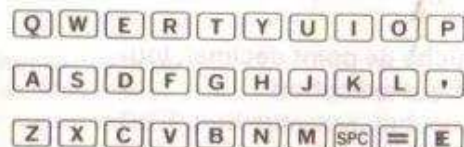
Par exemple, une opération effectuée par une calculatrice ordinaire serait $12 \times 4 \div 3 + 7 - 5 =$ alors qu'en utilisant cet ordinateur elle devient $12 * 4 / 3 + 7 - 5 =$.

Cet ordinateur peut être utilisé comme une calculatrice ordinaire comme montré ci-dessus.

Après avoir appuyé sur la touche **DEF**, l'appui sur une des touches numériques (0 à 9) permet de spécifier une zone de programme de P0 à P9, l'appui sur la touche **□** permet d'effectuer des calculs de puissance ($x^y \rightarrow x \uparrow y$) et l'appui sur les touches **+**, **-**, ***** et **/** permet d'entrer des opérateurs de comparaison ($\geq, \leq, >, <$).



• Touches alphabétiques, touche d'espacement

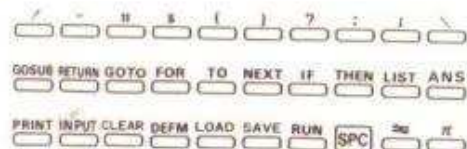


A l'aide de ces touches, des commandes peuvent être entrées ou des programmes écrits. Chacune des 26 touches alphabétiques de **A** à **Z** fonctionne comme une mémoire (pour les emplacements de stockage).

En outre, les touches de **A** à **Z** ont une autre fonction. Lorsqu'après avoir appuyé sur la touche **DEF**, on appuie sur une de ces touches, un symbole ou une commande BASIC sont affichés.

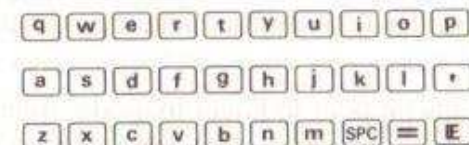
Appuyer sur la touche **SPC** quand un espace est nécessaire.

Exemple) **DEF A** → GOSUB, **DEF U** → ?

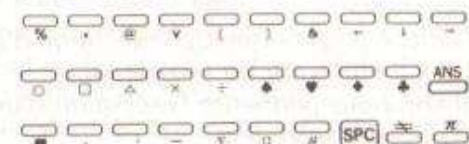


De plus, les touches alphabétiques ont une autre utilisation en mode extension (lorsque l'on a appuyé sur **□** après avoir appuyé sur la touche **MOD**, "EXT" est affiché). Lorsque l'on appuie dessus directement, des lettres minuscules sont affichées et lorsque l'on appuie dessus après avoir appuyé sur la touche **DEF**, des symboles spéciaux sont affichés.

Fonctions en mode extension:



Fonctions fournies lorsque l'on a appuyé sur une touche alphabétique après avoir appuyé sur la touche **DEF** en mode extension:



Pour annuler le mode extension, appuyer de nouveau sur **MOD** **□**.

• Touche égale (=)

La touche égale n'est pas utilisée pour fournir la réponse à un calcul, mais pour une instruction d'affectation et pour un test de condition dans une instruction IF (voir page 46).

En outre, quand on a appuyé sur cette touche après avoir appuyé sur la touche **DEF**, un symbole $\neq$ (différent) est affiché.

• Touche exposant/Pi (π)

Lorsque l'on appuie sur cette touche directement, elle permet de fournir un exposant. Par exemple, appuyer sur les touches **1** **□** **2** **3** **E** **4** pour obtenir $1,23 \times 10^4$. Quand l'exposant est négatif, appuyer sur la touche **-** après avoir appuyé sur cette touche. Par exemple, appuyer sur les touches **7** **□** **4** **1** **E** **-** **9** pour obtenir $7,41 \times 10^{-9}$. Lorsque l'on a appuyé sur cette touche après avoir appuyé sur la touche **DEF**, Pi (le rapport entre la circonférence d'un cercle et son diamètre) est affiché.

• **Touche réponse (ANS)**

Lorsque l'on a appuyé sur cette touche après avoir appuyé sur la touche **DEF**, le résultat du calcul manuel ou effectué par programme immédiatement avant est affiché.

• **Touche mode (MOD)**

Cette touche est utilisée conjointement avec les touches **□** et de **□** à **□** pour spécifier l'état de l'ordinateur ou l'unité d'angle.

MOD **□**..... "EXT" est affiché pour indiquer le mode extension dans lequel peuvent être utilisés des minuscules et des symboles spéciaux. Pour annuler le mode extension, appuyer de nouveau sur cette touche.

MOD **□**..... "RUN" est affiché pour permettre l'exécution de calculs manuels ou par programme.

MOD **1**..... "WRT" est affiché pour permettre l'exécution d'insertion, de contrôle et de mise en forme de programme.

MOD **2**..... "TR" est affiché pour permettre l'exécution d'une analyse. (Pour plus de détails, voir page 63).

MOD **3**..... Lorsque "TR" est affiché, le mode analyse est annulé et "TR" disparaît.

MOD **4**..... "DEG" est affiché pour indiquer que le degré est spécifié comme unité d'angle.

MOD **5**..... "RAD" est affiché pour indiquer que le radian est spécifié comme unité d'angle.

MOD **6**..... "GRA" est affiché pour indiquer que le grade est spécifié comme unité d'angle.

MOD **7**..... "PRT" est affiché pour permettre l'exécution d'une impression quand une imprimante est connectée.

MOD **8**..... Lorsque "PRT" est affiché, le mode impression est annulé et "PRT" disparaît.

MOD **9**..... " **MEM** **IN** " est affiché pour indiquer le mode entrée pour la fonction banque de données (voir page 32). Pour annuler ce mode, appuyer sur **MOD** **□**).

• **Touche de mémorisation (MEM)**

Employée pour utiliser la fonction banque de données. Employée également pour effectuer des rappels séquentiels en mode RUN (appuyer sur **MOD** **□**) ou en mode entrée (appuyer sur **MOD** **9**) ou pour effectuer un rappel après avoir appuyé sur un caractère spécifié.

• **Touche de curseur (← →)**

Ces touches sont utilisées pour déplacer le curseur (clignotant sous la forme "—" dans la fenêtre d'affichage) vers la gauche ou vers la droite. Cette commodité permet d'effectuer la correction d'un caractère affiché. Lorsque l'on appuie une fois sur ces touches, le curseur se déplace d'un caractère et lorsque l'on les maintient enfoncées, le curseur se déplace continuellement à l'intérieur de la suite de caractères écrits.

• **Touche d'effacement total (AC)**

Cette touche efface tous les affichages. En outre, elle est utilisée lorsqu'une erreur se produit ou lorsque l'affichage s'éteint par suite d'une coupure automatique d'alimentation (voir page 20). Quand un programme est en train d'être exécuté, en appuyant sur cette touche, l'exécution est suspendue.

• **Touche suppression/insertion (INS DEL)**

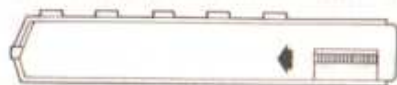
Cette touche est utilisée pour supprimer le caractère au-dessous duquel clignote le curseur. Après la suppression, les caractères placés à droite du curseur se déplacent vers la gauche. Lorsque l'on appuie sur cette touche après avoir appuyé sur la touche **DEF**, le caractère au-dessous duquel clignote le curseur se déplace vers la droite, ce qui provoque l'insertion d'un espace.

• **Touche arrêt (STOP)**

Lorsque l'on appuie sur cette touche pendant l'exécution d'un programme, celle-ci est temporairement arrêtée. Pour la reprendre, appuyer sur la touche **EXE**.

• Réglage du contraste de l'affichage

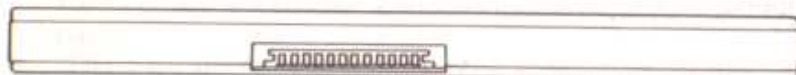
Quand l'affichage est trop sombre ou flou, selon l'état des piles ou l'orientation de l'appareil, régler son contraste à l'aide de la commande située sur le côté droit de l'ordinateur.



L'affichage devient plus sombre lorsque la commande est tournée dans la direction de la flèche et plus pâle dans le sens contraire. Si l'affichage reste pâle quand cette commande est placée sur la position la plus sombre, les piles sont faibles et doivent être remplacées.

• Partie connecteur

Ce connecteur est utilisé pour brancher un appareil optionnel. Lorsqu'une impression est nécessaire, brancher à celui-ci la MI547 et lorsque l'on doit stocker un programme sur bande, brancher la CI546.



Ne pas brancher d'autre appareils que la MI547 et la CI546 à ce connecteur. Lorsque ces appareils optionnels ne sont pas branchés, mettre toujours en place le couvercle de protection du connecteur.

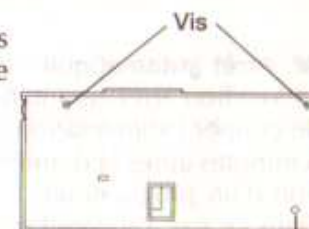
1-2. ALIMENTATION

L'ordinateur est alimenté par deux piles au lithium (CR2032). Lorsque l'ordinateur seul est utilisé, la durée de vie des piles est d'environ 140 heures. Néanmoins, celle-ci est plus courte si le vibreur est souvent utilisé. Si l'affichage est pâle même après avoir réglé le contraste (voir page 18), ceci provient de la faiblesse des piles qui doivent donc être remplacées dès que possible. Toujours remplacer les deux piles en même temps.

- * Remplacer les piles tous les deux ans même si elles n'ont pas été utilisées car une fuite pourrait se produire.

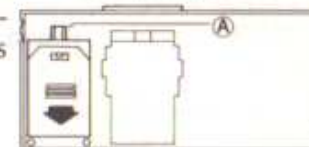
■ Comment changer les piles

- (1) Après avoir coupé l'alimentation, enlever les deux vis du panneau arrière puis enlever ce panneau.



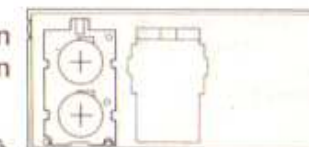
Bouton de remise à zéro générale
(Après avoir changé les piles, appuyer sur ce bouton avec un objet pointu.)

- (2) Tout en appuyant sur (A), faire coulisser le couvercle du compartiment à piles dans le sens de la flèche pour l'enlever.



- (3) Enlever les vieilles piles.
(Cette opération est plus facile si vous tapez doucement sur l'instrument avec le compartiment à piles orienté vers le bas.)

- (4) Bien essuyer les piles neuves avec un chiffon sec puis les mettre en place avec le côté + en haut.



- (5) Remonter le couvercle du compartiment à piles.

- (6) Remonter le panneau arrière et serrer ses vis.
Après avoir mis l'appareil sous-tension, appuyer sur le bouton de remise à zéro générale (R. A Z.).

- * Il est dangereux de jeter des vieilles piles au feu; elles pourraient exploser.
- * Etre sûr que les polarités des piles ne sont pas inversées.
- * Tenir les piles hors de portée des enfants. Si on avalait une pile, consulter un médecin immédiatement.

Conserver les piles dans un endroit situé hors de portée des enfants. Si elles ont été avalées, contacter immédiatement un médecin.

■ Arrêt automatique

La fonction arrêt automatique évite de gaspiller l'énergie quand on oublie de couper l'alimentation. Celle-ci sera automatiquement coupée environ 6 minutes après la dernière manipulation de touche (sauf pendant l'exécution d'un programme).

Dans ce cas, l'appareil sera remis sous tension en ouvrant puis refermant l'interrupteur d'alimentation ou en appuyant sur la touche **AC**.

* Bien que le contenu de la mémoire ne soit pas effacé lorsque l'alimentation est coupée, les spécifications d'angle et de mode ("RAD", "WRT", "TR", "PRT", etc.) sont annulées.

1-3. MODULE D'EXTENSION DE MEMOIRE VIVE (RAM) (OPTION)

La zone (mémoire) RAM standard de cet appareil est de 544 pas/26 mémoires. Toutefois, cette capacité peut être portée à un maximum de 1.568 pas/222 mémoires par l'ajout du bloc RAM EM545 optionnel. Cette zone RAM étendue peut être utilisée de la même manière que la zone standard et permet une augmentation du nombre de pas et de la capacité mémoire (voir page 22).

■ Comment monter le module RAM

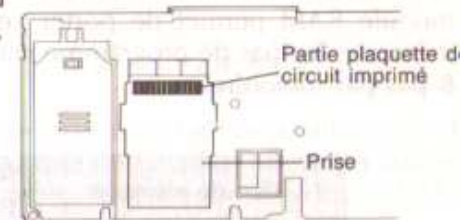
(Préparation)

Etant donné que les circuits internes peuvent être endommagés par l'électricité statique, avant de manipuler le module, ne pas oublier de toucher un objet métallique tel que poignée de porte afin de décharger toute électricité statique.

(Procédure)

(1) Couper l'alimentation.

(2) Desserrer les deux vis du panneau arrière puis enlever ce panneau.



(3) Insérer le module dans la prise du corps de l'appareil puis mettre le fermoir en position de blocage.

* Ne jamais toucher la partie connecteur du module RAM ni la partie plaquette de circuit imprimé du corps de l'appareil.



(4) Remonter le panneau arrière et serrer ses vis.

- Après avoir monté ou enlevé le module RAM, ne pas oublier d'appuyer sur le bouton de remise à zéro générale (R. A Z.) avec un objet pointu après avoir mis l'appareil sous tension. Si on n'appuie pas sur le bouton de remise à zéro générale, le contenu de la mémoire peut être changé ou un affichage qui n'a pas de sens peut apparaître.
- Prendre garde à ne pas salir ou toucher la partie connecteur du module ou la partie plaquette de circuit imprimé de l'appareil; cela entraînerait un mauvais contact.
- Quand le module est enlevé, ne pas oublier de le remettre dans son boîtier; et le ranger à l'abri de la poussière.

1-4. EXTENSION MEMOIRE

Normalement, il y a 26 mémoires (variables). Dans ce cas, le nombre de pas est de 544.

Le nombre maximal de mémoires standard est de 94. Le montage d'un module RAM permet de porter ce maximum à 222. Pour l'extension mémoire, les pas de programme sont convertis en mémoires à raison de 8 pas par mémoire.

Nombre de mémoires	Nombre de pas de programme	
	Standard	Avec extension
26	544	1568
27	536	1560
28	528	1552
⋮	⋮	⋮
46	384	1408
⋮	⋮	⋮
94	0	1024
⋮	⋮	⋮
200	—	176
⋮	⋮	⋮
222	—	0

L'extension mémoire est faite par paliers d'une unité à l'aide d'une commande DEFM.

Exemple:

Etendre de 30 pour porter à 56.

Opération:

Sélectionne le mode RUN (appuyer sur ) ou le mode WR1 (appuyer sur ) .

DEFM 30 

***VAR:56

- * DEFM peut être entré en appuyant sur  ou en appuyant sur  .

Une commande DEFM est aussi utilisée pour contrôler le nombre de mémoires désignées.

Exemple:

56 mémoires sont désignées.

DEFM 

***VAR:56

- Quand un grand nombre de pas de programme est déjà utilisé, afin de protéger le programme existant, si une désignation pouvant entraîner un insuffisant nombre de pas est tentée, une erreur se produit. (ERR 1 nombre de pas insuffisant)
- La variable à caractères exclusive (\$) étant une mémoire spéciale, elle n'est pas comptée lors de la désignation.

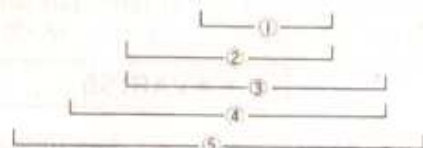
■ Ordre de priorité des calculs

Il existe une règle "d'ordre de priorité" pour les calculs dans laquelle la multiplication et la division sont effectuées avant l'addition et la soustraction. Cet ordinateur est muni d'une fonction qui distingue automatiquement l'ordre des priorités. Cette fonction est si pratique qu'elle permet d'obtenir une réponse correcte en entrant une expression numérique comme elle est. L'ordre des priorités des calculs est déterminé de la façon suivante.

- ① Fonctions (SIN, COS, etc.)
- ② Puissance (↑)
- ③ X (*), ÷ (/)
- ④ +, -

Un calcul est effectué en fonction de l'ordre des priorités. Si deux opérations d'un même niveau de priorité sont présentes, celle de gauche a priorité et si des parenthèses sont utilisées, les opérations placées entre parenthèses ont priorité.

Exemple) $2 + 3 * \text{SIN} (17 + 13) \uparrow 2 = 2.75$



■ Nombre de chiffres en entrée/sortie et nombre de chiffres dans les calculs

Le nombre de chiffres entrés peut être de 12 pour une mantisse et de 2 pour un exposant. Les opérations internes sont également effectuées en utilisant 12 chiffres pour une mantisse et 2 pour un exposant.

Bien que le nombre de chiffres en sortie soit généralement de 10 pour une mantisse et de 2 pour un exposant, il diffère suivant qu'il s'agit d'afficher le résultat d'un calcul manuel ou celui d'un calcul effectué par programme. Dans le cas d'un calcul manuel, le résultat comprenant jusqu'à 12 positions incluant mantisse, exposant et signe moins, est affiché. Alors que dans le cas d'un calcul effectué par programme, 10 chiffres de mantisse et deux d'exposant sont affichés.

Toutefois, si le résultat dépasse 12 chiffres, les 12 premiers sont affichés d'abord, puis le reste est affiché séquentiellement en déplaçant l'affichage vers la gauche.

Exemple)

Calcul manuel

1 □ 2345678912 **EX**
 12345678912 * 100 **EX**
 12345678912 * -100 **EX**

1.234567891
1.2345678 E12
-1.234567 E12

Calcul effectué par programme

Pour PRINT 12345678912 * -100

-1.234567891	E12	
-	1.234567891 E	12
-1	.234567891 E1	2
-1.	234567891 E12	

Disparaît de
l'affichage.

N'a pas été affiché.

Est automatiquement
déplacé.

CHAPITRE 2

PASSONS A L'EMPLOI

Vous devez utiliser cet ordinateur afin de vous habituer à lui, mais une mauvaise utilisation ne risque pas d'entraîner de panne. Tout d'abord, essayer une opération simple en accord avec le proverbe "il vaut mieux faire l'expérience pratique des choses qu'uniquement consulter des livres."

2-1. PASSONS A L'EMPLOI DE L'ORDINATEUR

Apprenez à utiliser l'ordinateur en vous en servant.
Pour la mise en marche, prendre l'ordinateur et le mettre sous tension en faisant glisser l'interrupteur d'alimentation vers la droite. Alors les messages suivants sont affichés.

RUN DEG
READY P0

D'abord effacer ce message en appuyant sur la touche **AC**. "READY P0" disparaît. Ensuite le caractère " _ " clignote à l'extrême gauche de l'écran. C'est le curseur, à cet endroit un caractère peut être écrit.

RUN DEG
_

Lorsque le curseur clignote, on dit que l'ordinateur est en "état d'attente d'entrée (de données)", il attend l'entrée d'une opération ou d'une instruction. Ordinairement le curseur clignote sous la forme " _ ", mais lorsqu'on entre des caractères, il clignote également sous la forme " █ ". On peut entrer jusqu'à 62 caractères dans une ligne. Le caractère " █ " apparaît comme signal d'avertissement lorsqu'on a entré au moins 56 caractères. Les affichages "RUN" et "DEG" indiquent respectivement le mode marche au cours duquel des opérations ou des programmes peuvent être exécutés et l'unité d'angle "degré". L'unité d'angle comprend également un mode radian ("RAD" est affiché) et un mode grade ("GRA" est affiché) que l'on peut utiliser en appuyant respectivement sur les touches **MODE** **5** et **MODE** **6**. Une unité d'angle est exigée lorsqu'on utilise des fonctions trigonométriques. Lorsqu'on met l'ordinateur sous tension, le message "DEG" s'affiche.

D'autres affichages indiquent un mode insertion de programmes ("WRT" est affiché), un mode analyse ("TR" est affiché), un mode impression ("PRT" est affiché), un mode entrée pour la fonction banque de données (**MEMO** **IN** est affiché) et un mode extension ("EXT" est affiché) que l'on utilise en appuyant respectivement sur les touches **MODE** **1**, **MODE** **2**, **MODE** **7**, **MODE** **9**, et **MODE** **+**.

Vous apprendrez à vous servir de ces différents modes au fur et à mesure que vous utiliserez l'ordinateur.

Passons réellement à l'emploi de l'ordinateur pour comprendre les affichages. Si un message reste affiché après avoir cessé d'utiliser le mode correspondant, ouvrir puis refermer l'interrupteur d'alimentation.

Commençons par une opération simple.

Exemple) $123+456=579$

_

Appuyer sur la touche **AC**.

Appuyer sur les touches appropriées pour rentrer l'expression numérique.

1**2****3****+****4****5****6**

123+456_

Puis une réponse est obtenue en appuyant sur la touche **EXE** au lieu de **=**.

EXE

579

Une autre opération.

Exemple) $45 \times 6 + 89 = 359$

Considérons que l'on ait tapé 46 par erreur au lieu de 45.

4**6*********6****+****8****9**

46*6+89_

Vous vous apercevez alors que 46 a été tapé par erreur. Placez le curseur à l'endroit de l'erreur en utilisant calmement la touche de déplacement du curseur **←**.

←←←←←

4~~6~~*6+89

Le curseur et le chiffre 6 clignotent.

Entrer alors un **5**.

5

45*6+89

Puisque l'expression est maintenant correcte, pour obtenir une réponse appuyer sur la touche **EXE**.

EXE

359

Lorsqu'une erreur a été faite, il est facile de la corriger en utilisant la touche de déplacement du curseur. Toutefois, si on a appuyé sur la touche **EXE**, il faut recommencer depuis le début.

Maintenant entrons des caractères en utilisant les touches alphabétiques. L'ordre des touches alphabétiques est le même que celui d'une machine à écrire de type QWERTY. Etant donné que la plupart des ordinateurs ont un clavier de type QWERTY, se souvenir de l'emplacement des caractères même si ce n'est pas facile pour un débutant.

D'abord entrer des caractères.

Exemple) Les caractères utilisés sont "ABCXYZ".

Entrer ABC.

ABC

ABC_

Entrer ensuite XYZ.

XYZ

ABCXYZ_

Maintenant mettre un espace entre ABC et XYZ.
Placer le curseur sous le X.

←←←

ABCXYZ

Introduire un espace.

SHIFT INS

ABC_XYZ

Lorsqu'un espace doit être inséré entre des caractères, placer le curseur à l'endroit où l'insertion est désirée et appuyer sur SHIFT INS. Répéter cette procédure si d'autres espaces doivent être insérés.

Cet ordinateur possède des caractères spéciaux en plus des caractères alphanumériques, ils servent pour les jeux ou comme symboles scientifiques. Voir page 15 les différentes sortes de caractères spéciaux.

Affichons certains de ces caractères.

Exemple) Affichage des symboles ♠♥♦♣.

D'abord mettre en mode extension.

AC EXT

EXT est affiché
EXT

Pour afficher ces symboles, appuyer sur les touches alphabétiques et la touche SHIFT.

SHIFT ♠ SHIFT ♥ SHIFT ♦ SHIFT ♣

♠♥♦♣_

Exemple) Afficher les symboles Σ , Ω , μ .

Comme le mode extension est utilisé, effectuer seulement les opérations suivantes.

SHIFT Σ SHIFT Ω SHIFT μ

♠♥♦♣ $\Sigma\Omega\mu$ _

Comme l'ordinateur possède ces symboles, veuillez les utiliser. Pour retourner au mode dans lequel les lettres majuscules sont affichées, appuyer de nouveau sur AC pour annuler "EXT". L'utilisation des touches est maintenant acquise. Pendant une utilisation prolongée de cet ordinateur, "ERR2" est parfois affiché et les touches ne fonctionnent plus. Ce n'est pas une panne de l'ordinateur mais un message, appelé "message d'erreur", qui indique une instruction erronée. Si cela survient, appuyer calmement sur la touche AC. Le message disparaît alors et l'ordinateur se remet à fonctionner. Pour plus de détails, voir pages 58 et 176.

2-2. PRATIQUE FONCTION BANQUE DE DONNEES

Cet ordinateur est muni d'une fonction banque de données qui permet de stocker et de rappeler des données en utilisant seulement la touche $\boxed{\text{MEM}}$. Elle peut être utilisée de différentes façons.

Par exemple, elle peut être utilisée comme répertoire téléphonique, horaire, emploi du temps, tableau, etc.

En outre, comme l'extraction, l'accès et l'insertion de données peuvent être effectués par un programme BASIC, on peut utiliser la banque de données comme liste de clients, liste de produits, calculs d'estimation, catalogue de livres, etc.

Il y a beaucoup d'autres manières d'utiliser la banque de données.

Pour plus de détails, voir le "Manuel de référence de banque de données."

2-3. POUR COMMENCER, DES CALCULS SIMPLES

Les calculs simples sont effectués de la manière suivante. Toutefois, si vous n'avez jamais utilisé de calculatrice scientifique, faites attention car cet ordinateur est muni de fonctions algébriques logiques qui effectuent multiplications et divisions avant additions et soustractions.

Exemple 1) $23 + 4.5 - 53 = -25.5$

Opération) $23 \boxed{+} 4.5 \boxed{-} 53 \boxed{=}$

-25.5

* Les touches numériques sont maintenant représentées sans être entourées comme ci-dessus.

Exemple 2) $56 \times (-12) \div (-2.5) = 268.8$

Opération) $56 \boxed{\times} 12 \boxed{\div} 2.5 \boxed{=}$

268.8

* Pour entrer un nombre négatif, appuyer sur la touche $\boxed{-}$ avant le nombre.

Exemple 3) $7 \times 8 - 4 \times 5 = 36$

Opération) $7 \boxed{\times} 8 \boxed{-} 4 \boxed{\times} 5 \boxed{=}$

36

* Les multiplications sont effectuées d'abord, ensuite la soustraction.

Exemple 4) $(4.5 \times 10^{75}) \times (-2.3 \times 10^{-78}) = -0.01035$

Opération) $4.5 \boxed{\text{E}} 75 \boxed{\times} 2.3 \boxed{\text{E}} 78 \boxed{=}$

-0.01035

* Appuyer sur la touche $\boxed{\text{E}}$ puis entrer l'exposant.

Il y a d'autres calculs algébriques qui utilisent la mémoire. Celle-ci est pratique quand une même valeur numérique est reprise dans plusieurs calculs.

Par exemple,

$$\begin{aligned} 3x + 5 &= \\ 4x + 6 &= \\ 5x + 7 &= \end{aligned}$$

Si la valeur de x dans ces calculs est 123456, il peut être pénible de la retaper plusieurs fois. Y a-t-il une manière d'effectuer ces calculs en évitant de retaper plusieurs fois la valeur de x ? La solution est d'utiliser une mémoire appelée variable. Dans cet exemple, les calculs sont effectués en utilisant la variable X .

D'abord affecter la valeur 123,456 à la variable X.

X = 123,456 EXE

"=" ne signifie pas le signe =, mais que la variable X prend la valeur 123,456. Faisons ces calculs.

3 * X + 5 EXE

4 * X + 6 EXE

5 * X + 7 EXE

375.368
499.824
624.28

Ils peuvent être effectués facilement.

Comme cet ordinateur possède 26 variables de A à Z, de nombreuses valeurs numériques peuvent être mémorisées.

Dans ces exemples, la valeur numérique de X est constante et les opérations différentes. Qu'en est-il du cas dans lequel pour une même opération la variable prend des valeurs différentes?

Si l'opération est " $3x + 5 =$ " et si la variable x prend les valeurs 123, 456, 789, suivre la procédure décrite ci-dessus sera pénible. En fait, l'opération sera mémorisée par l'ordinateur et il sera seulement nécessaire de changer la valeur de la variable X. Cette méthode de calcul pratique est appelée "calcul programmé". Un point fort de cet ordinateur est le calcul programmé. Un calcul manuel, étape préalable dans laquelle un programme de calcul est utilisé, est décrit au chapitre 3. PROGRAMMATION EN BASIC.

2-4. CALCULS DE FONCTIONS — UN POINT IMPORTANT DE CET ORDINATEUR

Cet ordinateur est muni de fonctions scientifiques en plus des quatre fonctions de base.

Toutes ces fonctions peuvent être utilisées par programme ou manuellement comme expliqué ci-dessous.

Les fonctions dont dispose cet ordinateur sont les suivantes.

Noms des fonctions

Fonctions trigonométriques

Fonctions trigonométriques inverses

Racine carrée

Logarithme

Logarithme népérien

Fonction exponentielle

Puissance

Conversion décimal/sexagésimal

Conversion sexagésimal/décimal

Valeur entière

Partie fractionnaire

Valeur absolue |x|

Signe

Arrondi

Nombre aléatoire

$\sin x$

$\cos x$

$\tan x$

$\sin^{-1} x$

$\cos^{-1} x$

$\tan^{-1} x$

$\sqrt{x}$

$\log x$

$\ln x$

e^x

x^y

|x|

$\left\{ \begin{array}{l} \text{nombres positifs} \rightarrow 1 \\ \quad \quad \quad 0 \rightarrow 0 \\ \text{nombres négatifs} \rightarrow -1 \end{array} \right\}$

$\left\{ \begin{array}{l} \text{la valeur de } x \text{ est} \\ \text{arrondie à la position } 10^y \end{array} \right\}$

Format

SIN x

COS x

TAN x

ASN x

ACS x

ATN x

SQR x

LOG x

LN x

EXP x

x↑y

DMS \$ (x)*

DEG (x,y,z)*

INT x

FRAC x

ABS x

SGN x

RND (x,y)*

RAN #

* Pour les fonctions DMS \$, DEG et RND les arguments doivent être placés entre parenthèses.

Effectuons des calculs utilisant des fonctions.

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

Lorsque les fonctions trigonométriques ou trigonométriques inverses sont utilisées, toujours s'assurer de bien spécifier l'unité d'angle (DEG, RAD, GRA).

Exemple) $\sin 12.3456^\circ = 0.2138079201$

Opération) $\text{MOD} \rightarrow * \text{DEG}^*$

$\text{SIN} 12.3456 \text{ EXE}$

0.2138079201

* A partir de maintenant, les touches alphabétiques et les touches numériques seront représentées sans cadre.

Exemple) $\cos 63^\circ 52' 41'' = 0.4402830847$

Opération) $\text{COS DEG} \text{ MOD} \rightarrow 63 \text{ MOD} \rightarrow 52 \text{ MOD} \rightarrow 41 \text{ EXE}$

0.4402830847

Exemple) $2 \cdot \sin 45^\circ \times \cos 65.1^\circ = 0.5954345575$

Opération) $2 \text{ MOD} \rightarrow \text{SIN} 45 \text{ MOD} \rightarrow \text{COS} 65.1 \text{ EXE}$

0.5954345575

Exemple) $\sin^{-1} 0.5 = 30^\circ$

Opération) $\text{ASN} 0.5 \text{ EXE}$

30

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

Opération) $\text{MOD} \rightarrow * \text{RAD}^*$

$\text{COS MOD} \rightarrow \text{MOD} \rightarrow 3 \text{ MOD} \rightarrow \text{EXE}$

0.5

Exemple) $\cos^{-1} \frac{\sqrt{2}}{2} = 0.7853981634 \text{ rad}$

Opération) $\text{ACS MOD} \rightarrow \text{SQR} 2 \text{ MOD} \rightarrow 2 \text{ MOD} \rightarrow \text{EXE}$

0.7853981634

Exemple) $\tan(-35 \text{ gra}) = -0.612800788$

Opération) $\text{MOD} \rightarrow * \text{GRA}^*$

$\text{TAN MOD} \rightarrow 35 \text{ EXE}$

-0.612800788

• Fonctions logarithmiques (log, ln) et fonctions exponentielles (e^x , x^y)

Exemple) $\log 1.23 (= \log_{10} 1.23) = 0.0899051114$

Opération) $\text{LOG} 1.23 \text{ EXE}$

0.0899051114

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

Opération) $\text{LN} 90 \text{ EXE}$

4.49980967

Exemple) $e^5 = 148.4131591$

Opération) $\text{EXP} 5 \text{ EXE}$

148.4131591

Exemple) $10^{1.23} = 16.98243652$

(L'antilogarithme de 1,23 est obtenu.)

Opération) $10 \text{ MOD} \rightarrow 1.23 \text{ EXE}$

16.98243652

Exemple) $5.6^{2.3} = 52.58143837$

Opération) $5.6 \text{ MOD} \rightarrow 2.3 \text{ EXE}$

52.58143837

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

Opération) $123 \text{ MOD} \rightarrow \text{MOD} \rightarrow 1 \text{ MOD} \rightarrow 7 \text{ MOD} \rightarrow \text{EXE}$

1.988647795

* Quand $x < 0$, y est un nombre naturel.

Exemple) $\log \sin 40^\circ + \log \cos 35^\circ = -0.278567983$

Le nombre est 0,5265407845 (calcul logarithmique de $\sin 40^\circ \times \cos 35^\circ$).

Opération) $\text{MOD} \rightarrow * \text{DEG}^*$

$\text{LOG SIN} 40 \text{ MOD} \rightarrow \text{LOG COS} 35 \text{ EXE}$

-0.278567983

$10 \text{ MOD} \rightarrow \text{MOD} \rightarrow \text{ABS} \text{ EXE}$

0.5265407845

• Autres fonctions ($\sqrt{\quad}$, SGN, RAN#, RND, ABS, INT, FRAC)

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

Opération) $\text{SQR} 2 \text{ MOD} \rightarrow \text{SQR} 5 \text{ EXE}$

3.65028154

Exemple) Donne "1" si le nombre est positif, "-1" s'il est négatif et "0" s'il est nul.

Opération) SGN 6 **EXE**
SGN 0 **EXE**
SGN -2 **EXE**

1
0
-1

Exemple) Génération de nombres aléatoires (pseudo nombres aléatoires compris dans la plage de $0 < \text{RAN\#} < 1$)

Opération) RAN **EXE**

0.7903739076

Exemple) Arrondir la réponse de $12,3 \times 4,56$ à une décimale.

$$12,3 \times 4,56 = 56,088$$

Opération) RND **EXE** 12.3 **EXE** 4.56 **EXE** 2 **EXE**

* Quand RND (x, y) est employée,
 $|y| < 100$.

56.1

Exemple) $| -78,9 \div 5,6 | = 14,08928571$

Opération) ABS **EXE** 78.9 **EXE** 5.6 **EXE**

14.08928571

Exemple) Partie entière de $7800/96 \dots 81$

Opération) INT **EXE** 7800 **EXE** 96 **EXE**

81

* Le plus grand entier qui ne dépasse pas la valeur numérique initiale est obtenu par cette fonction.

Exemple) Partie fractionnaire de $7800/96 \dots 0,25$

Opération) FRAC **EXE** 7800 **EXE** 96 **EXE**

0.25

• Désignation du nombre de chiffres significatifs et désignation du nombre de décimales

Ces désignations se font à l'aide d'une commande "SET".

Désignation du nombre de chiffres significatifs SET E n (n = 0 à 8)

Désignation du nombre de décimales SET F n (n = 0 à 9)

Annulation de désignation de nombre de chiffres SET N

* Dans un calcul manuel, "SET E0" correspond à une désignation de 8 chiffres significatifs.

* Même si une désignation est faite, la valeur numérique initiale reste dans la mémoire.

Exemple) $100 \div 6 = 16,66666666\dots$

SET E 4 **EXE** (Désignation de 4 chiffres significatifs)

Opération) 100 **EXE** 6 **EXE**

1.667E01

Exemple) $123 \div 7 = 17,57142857\dots$

Opération) SET F 2 **EXE** (Désignation de 2 décimales)

123 **EXE** 7 **EXE**

17.57

Exemple) $1 \div 3 = 0,3333333333\dots$

Opération) SET N **EXE** (Annulation de désignation)

1 **EXE** 3 **EXE**

0.3333333333

• Conversion décimal $\leftrightarrow$ sexagésimal (DEG, DMS \$)

Exemple) $14^{\circ}25'36'' = 14,42666667$

Opération) DEG **EXE** 14 **EXE** 25 **EXE** 36 **EXE**

14.42666667

Exemple) $12.3456^\circ = 12^\circ 20' 44.16''$

Opération) DMS $\square$ $\square$ 12.3456 $\square$ $\square$ $\square$ 12° 20' 44.16

Exemple) $\sin 63^\circ 52' 41'' = 0.897859012$

Opération) $\square$ 4 $\square$ SIN DEG $\square$ 63 $\square$ 52 $\square$ 41 $\square$ $\square$ 0.897859012

CHAPITRE 3

PROGRAMMATION EN BASIC

Dans ce chapitre, nous allons étudier la programmation en BASIC. Nous conseillons aux utilisateurs qui ne sont pas habitués à la programmation de lire ce chapitre attentivement et plusieurs fois si nécessaire. Une telle pratique leur permettra de maîtriser les rudiments de la programmation.

3-1. QU'EST-CE QU'UN PROGRAMME?

Le mot "PROGRAMME" peut paraître correspondre à quelque chose de difficile. Néanmoins, il existe des programmes très simples et d'autres plus compliqués. Par exemple, un calcul dans lequel des expressions algébriques sont mémorisées et des valeurs numériques affectées à ces expressions est aussi un programme.

3-1-1 Un programme est pratique

Nous avons souvent affaire à de nombreuses et différentes sortes de calculs; gestion financière, mesure, comptabilité, etc. Bien que ces calculs ne soient pas très fastidieux s'ils ne sont exécutés qu'une seule fois, ils peuvent le devenir s'ils doivent être répétés en changeant les valeurs numériques à maintes reprises. Dans ce dernier cas, l'emploi d'un ordinateur s'avère très pratique.

Par exemple, avec l'expression $y = 2x^2 + 5x + 13$, le même calcul, permettant d'obtenir la valeur de y , doit être répété pour chaque valeur de x . Pour éliminer cette contrainte, l'expression suivante est mise en mémoire.

```
10 INPUT X
20 Y=2*X^2+5*X+13
30 PRINT Y
```

Dans ce programme une expression numérique est mémorisée. L'expression de calcul est écrite à la deuxième ligne. Une explication détaillée sera donnée plus loin. Ce programme permet d'effectuer le calcul très facilement. Des programmes simples peuvent être commodément employés en mémorisant une expression de calcul comme mentionné ci-dessus. Voyons les instructions de ce programme.

3-1-2 Structure du programme

Etudions la structure de notre programme.

```
10 INPUT X
20 Y=2*X^2+5*X+13
30 PRINT Y
```

Ce programme peut être grossièrement décomposé en 3 parties comme suit.

```
10 INPUT X ..... Entrée
20 Y=2*X^2+5*X+13 ..... Calcul
30 PRINT Y ..... Sortie
```

En premier lieu, la partie entrée est employée pour entrer (INPUT) des données (telles que valeurs numériques destinées aux calculs) dans l'ordinateur. Cette partie peut être étendue en fonction du volume de données à traiter. Ensuite, la partie calcul est employée pour effectuer le calcul désiré. Enfin, la partie sortie est employée pour afficher le résultat dudit calcul. Un ordinateur effectue toutes les tâches nécessaires si des commandes (instructions) correctes lui sont fournies. Dans notre exemple, une commande d'entrée (INPUT) et une commande d'affichage (PRINT) sont mémorisées. Chacune des trois parties (entrée, calcul et sortie) peut être décomposée comme suit.

N° de ligne	Commande	Opérande
10	INPUT	X

Les numéros de ligne indiquent l'ordre d'exécution des différentes instructions du programme. Etant donné qu'un ordinateur lit et exécute les instructions dans l'ordre croissant des numéros de ligne, affecter ces numéros en fonction du déroulement escompté. De plus, il est préférable d'affecter ces numéros par incréments de 10 (10, 20, 30, ...) afin de pouvoir faire des ajouts par la suite. Pour les numéros de ligne, on ne peut pas employer de nombres avec décimales.

Les éléments qui suivent le numéro de ligne constituent la commande que l'ordinateur doit exécuter. Il existe de nombreuses commandes différentes utilisées pour spécifier l'opération à exécuter. Bien qu'il soit souhaitable de se souvenir de toutes les commandes, apprendre d'abord le minimum nécessaire, puis le reste progressivement. Voir le chapitre 4 "REFERENCE DE COMMANDES" page 117 et suivantes pour les différentes commandes et leurs fonctions.

Le rôle de l'opérande d'une commande est de la compléter. Certaines commandes ont un opérande, d'autres non. Dans notre exemple, la commande INPUT indique l'entrée de donnée. Un opérande spécifie la mémoire dans laquelle cette donnée est placée. Dans ce cas, il indique que la donnée est affectée à la variable X.

Dans la ligne suivante,

N° de ligne	Instruction d'affectation
20	Y=2*X^2+5*X+13

20 est le numéro de ligne. L'instruction d'affectation signifie que la valeur de l'expression qui se trouve à droite du signe égal (=) est affectée à la variable qui se trouve à gauche. Si la commande LET est ajoutée à l'affectation comme suit,

20 LET Y=2*X+2+5*X+13

LET est une commande et l'instruction d'affectation un opérande. La ligne 30 est composée, comme la ligne 10, d'un numéro de ligne, d'une commande et d'un opérande.

30 PRINT Y

N° de ligne Commande Opérande

Un programme est constitué, comme mentionné ci-dessus, de commandes, d'opérandes et de numéros de ligne.

3-1-3 Préparation facile d'un programme

Lorsque la structure d'un programme est comprise, il n'est pas si difficile d'en préparer un.

Lorsque les trois principales parties (entrée, calcul et sortie) sont comprises, on peut les utiliser pour préparer un programme. Il n'est pas nécessaire d'apprendre toutes les commandes en même temps. Il est recommandé d'essayer de simples calculs en utilisant uniquement les commandes "INPUT", "PRINT" et une instruction d'affectation.

Le secret de la maîtrise d'une programmation rapide n'est pas de simplement se rappeler un programme, mais en fait de préparer un programme en sélectionnant les problèmes qui peuvent aisément être mis en équation comme traitement de factures ou gestion financière, effectués de manière répétitive dans une entreprise, ou mesures, gestion ménagère et calculs de temps pour enregistrement sur cassette souvent effectués à la maison. La meilleure façon de maîtriser la programmation est d'abord de préparer un programme pour le problème que l'on veut traiter, d'apprendre les commandes nécessaires, puis d'améliorer le programme.

Un autre secret est de ne pas préparer le programme entier en même temps, mais d'en préparer des parties et de les assembler ultérieurement.

Préparer un programme lentement et progressivement en suivant ces principes. Lorsque la structure fondamentale d'un programme est comprise, on peut en préparer un en se référant aux fonctions des commandes décrites dans le chapitre 4 "REFERENCE DE COMMANDES" et en les utilisant.

3-2. PREPARATION D'UN PROGRAMME

Ce paragraphe et les suivants indiquent comment préparer un programme; ils traitent du point le plus important, LA PROGRAMMATION EN BASIC.

3-2-1 Préparation d'un organigramme

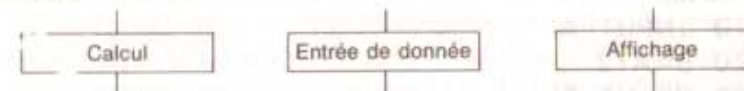
Vous ne devez pas être habitué à un organigramme! C'est un graphique qui décrit les étapes d'un travail.

On entend souvent qu'un organigramme n'est pas nécessaire pour programmer en BASIC, c'est vrai pour celui qui est habitué à utiliser un ordinateur. Néanmoins, comme la totalité de la méthode de travail est difficile à comprendre pour un débutant, il est important de faire un organigramme simple ne serait-ce que pour voir le déroulement du programme.

Bien qu'il existe des symboles formels et spécialisés pour faire un organigramme, il n'est pas nécessaire de se les rappeler. Placer simplement chaque élément dans un rectangle et les relier par des lignes.

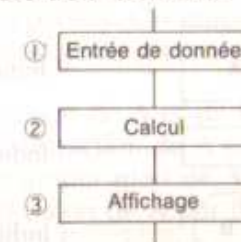
Préparons un programme en expliquant chaque élément.

Par exemple, préparons un programme qui calcule le carré d'un nombre que l'on a entré. Une partie calcul est nécessaire. Etant donné qu'un nombre est entré et qu'une réponse est affichée, une partie entrée et une partie sortie sont nécessaires. Les trois parties sont placées dans des rectangles comme suit.



Pour mettre dans l'ordre ces trois éléments, on peut facilement deviner que le dernier est l'affichage de la réponse. Ensuite, qu'il est nécessaire de faire le calcul pour obtenir une réponse et également d'entrer une donnée pour faire le calcul.

On a maintenant sûrement compris dans quel ordre ces trois éléments doivent être placés. Relier ces trois éléments.

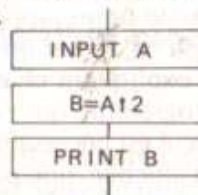


Bien qu'ainsi l'organigramme soit complet, mettons le sous une forme qui soit plus proche de celle d'un programme.

Le premier élément est une instruction d'entrée de donnée. Comme en utilisant une commande INPUT, une donnée est entrée dans une variable, déterminer cette variable. Si la variable A est utilisée, le contenu de (1) est "INPUT A".

Dans le calcul effectué dans le second élément, la valeur entrée dans la variable A est élevée au carré et la réponse affectée à une autre variable. Si cette variable est B, le calcul s'écrit " $B=A^2$ ". Cette expression de calcul est appelée une instruction d'affectation qui s'écrit sous la forme "LET $B=A^2$ ". Cependant, étant donné que LET peut être omis, on peut l'écrire " $B=A^2$ ".

La réponse est affichée dans le troisième élément. Etant donné qu'on affiche la variable B qui contient la réponse par une commande PRINT, le contenu de ce dernier élément est donc "PRINT B". Replaçons les trois éléments dans l'organigramme.



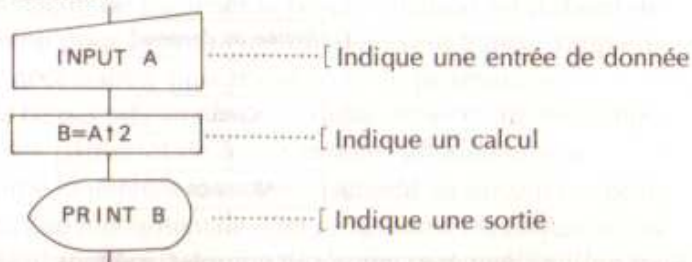
Le programme sera complété en plaçant des numéros de ligne devant ces trois éléments.

```

10 INPUT A
20 B=A2
30 PRINT B
  
```

On peut facilement et rapidement écrire un programme en construisant un organigramme après avoir préparé chaque élément comme mentionné ci-dessus.

Un véritable organigramme comporte des symboles formels qui ont des significations spéciales. Faisons l'organigramme de notre exemple en utilisant ces symboles formels.

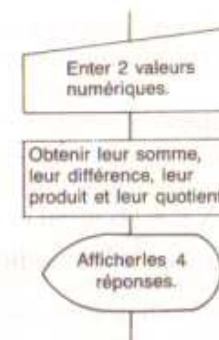


Voir à la fin de ce manuel les symboles d'organigramme.

3-2-2 Préparation d'un programme

Pour donner un exemple simple, entrer 2 valeurs numériques et obtenir leur somme, leur différence, leur produit et leur quotient.

Préparer un organigramme basé sur les parties entrée, calcul et sortie qui sont les trois éléments importants.



Utiliser une instruction INPUT (instruction qui permet d'entrer des données dans une variable à partir du clavier) pour entrer 2 valeurs numériques. Les éléments qui doivent être notés sont les variables dans lesquelles sont affectées les données entrées et les réponses. L'utilisation des variables est assez compliquée. Les variables comprennent celles composées d'un caractère alphabétique de A à Z et les tableaux qui ont un élément appelé indice représentées de la manière A(3). Bien qu'une variable puisse être choisie parmi celles-ci, il est recommandé de les utiliser dans l'ordre alphabétique tant que l'on est pas habitué à la programmation.

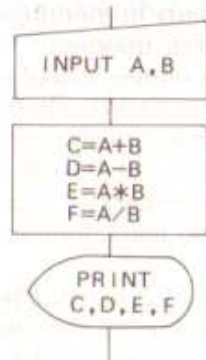
Deux valeurs numériques sont entrées dans notre exemple; on choisira A et B et l'on écrira "INPUT A, B". Plusieurs variables peuvent être placées dans une commande INPUT; les séparer par des virgules.

Qu'en est-il de la partie calcul? Quatre calculs sont faits dans notre exemple; on obtiendra quatre réponses. On utilisera C, D, E, F pour désigner les variables dans lesquelles les quatre réponses seront affectées.

Au préalable, Pour l'addition de A et de B, on utilisera $C=A+B$.
 Pour la soustraction de A et de B, on utilisera $D=A-B$.
 Pour la multiplication de A par B, on utilisera $E=A*B$.
 Pour la division de A par B, on utilisera $F=A/B$.

Puisque les calculs sont effectués, on affichera ensuite les réponses. La commande d'affichage est PRINT; on écrira "PRINT C, D, E, F".

On peut construire l'organigramme du programme de la manière suivante.



Compléter le programme en plaçant les numéros de lignes.

```

10 INPUT A,B
20 C=A+B
30 D=A-B
40 E=A*B
50 F=A/B
60 PRINT C,D,E,F
  
```

Ensuite ajouter "END" pour indiquer la fin du programme.

```

70 END
  
```

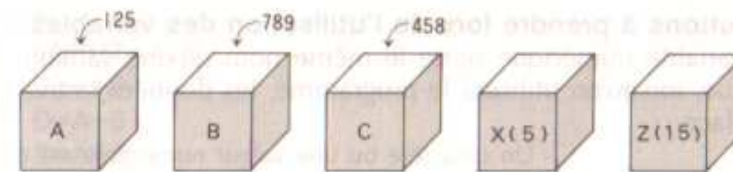
Avec l'instruction ci-dessus, le programme est terminé. Il peut être aisément écrit s'il est assemblé séquentiellement.

Dans un premier temps, préparer un programme simple et pratique au lieu d'un compliqué qui utilise plusieurs commandes différentes.

REMARQUE

VARIABLES

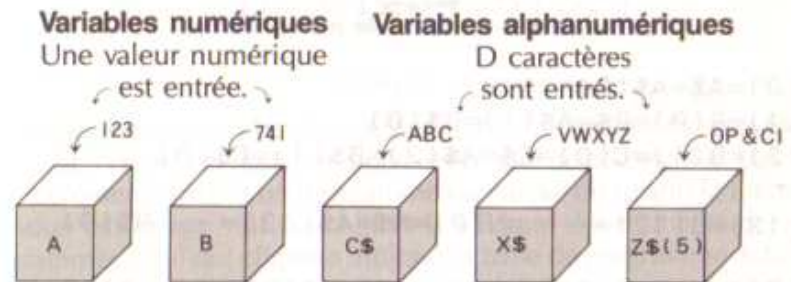
Les variables sont des éléments importants pour la préparation des programmes. Les variables sont exactement comme des boîtes dans lesquelles les données entrées ou calculées sont stockées; elles portent toutes un nom. Les variables comprennent celles standards de A à Z et celles dont le nom possède un indice lié à la lettre (A à Z). Ces dernières sont appelées des tableaux comme A(5) ou B(50).



Il y a également deux types de variables, les numériques dans lesquelles sont entrées des valeurs numériques et les alphanumériques qui servent à entrer des chaînes de caractères.

Les variables que nous avons utilisées précédemment et où nous avons entré des valeurs numériques pour effectuer des calculs sont des variables numériques. En plus de celles-ci, il y a des variables alphanumériques dont le nom s'écrit avec un \$ lié à la lettre de A à Z, telles que A\$, B\$, C\$, et une exclusive variable alphanumérique, \$.

On peut entrer une valeur numérique d'au plus 10 chiffres (10 chiffres pour la mantisse et 2 pour l'exposant) dans une variable numérique et une chaîne de 7 caractères au plus dans une variable alphanumérique. En outre, jusqu'à 30 caractères peuvent être entrés dans l'exclusive variable alphanumérique.



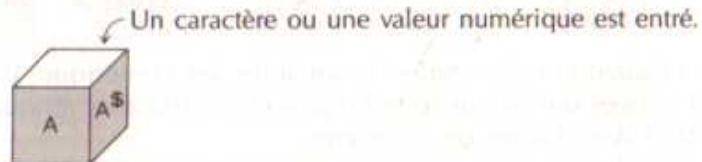
Etant donné que les éléments entrés dans ces deux types de variables sont différents, les caractères tels que "ABC" ne peuvent être entrés dans une variable numérique, de même des valeurs numériques ne peuvent être affectées pour des calculs à des variables alphanumériques.

L'utilisation de ces variables est différente. Utiliser les variables numériques pour entrer des valeurs numériques servant à faire des calculs et des variables alphanumériques pour entrer des messages ou des symboles.

Les tableaux sont pratiques pour stocker des données utilisant plusieurs variables. Elles sont distinguées par l'indice qui indique la 1^{ère} variable, la 2^e, etc. On expliquera les tableaux en les utilisant dans un programme.

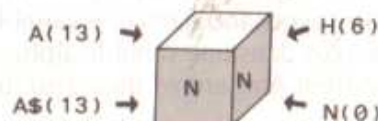
<Précautions à prendre lors de l'utilisation des variables>

Si une variable numérique porte le même nom qu'une variable alphanumérique, lorsqu'on utilisera le programme, les données entreront à la même place.



En fait, la variable numérique A et la variable alphanumérique A\$ ne peuvent être utilisées simultanément dans un même programme.

Egalement, lorsqu'un tableau est utilisé, des précautions doivent être prises pour éviter de donner plusieurs noms équivalents au même élément.



Tous les noms sont pour le même élément.

A=A(0)=A\$=A\$(0)

B=A(1)=B(0)=B\$=A\$(1)=B\$(0)

C=A(2)=B(1)=C(0)=C\$=A\$(2)=B\$(1)=C\$(0)

⋮

N=A(13)=B(12)=.....=N(0)=N\$=A\$(13)=.....N\$(0)

⋮

Z=A(25)=B(24)=C(23)=.....=Z\$=A\$(25)=.....Z\$(0)

Les mêmes précautions doivent être prises quand la mémoire est étendue par une instruction DEFM.

3-2-3 Entrée de programmes

Stocker (entrer) un programme. Utilisons le précédent programme dans lequel quatre calculs de base sont effectués après avoir entré deux valeurs numériques.

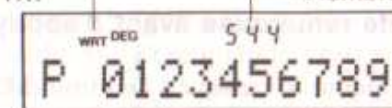
Programme

```
10 INPUT A,B
20 C=A+B
30 D=A-B
40 E=A*B
50 F=A/B
60 PRINT C,D,E,F
70 END
```

Quand l'interrupteur d'alimentation est fermé, le mode RUN dans lequel des calculs manuels ou des programmes peuvent être exécutés est spécifié. Passer de ce mode au mode WRT (mode stockage de programmes). Etant donné que l'on doit également insérer pour stocker un programme, le mode WRT est appelé mode "insertion". Appuyer sur **MOD** **1** pour passer en mode WRT.

Indication de mode WRT

Nombre de pas restants



Zones de programme

Cet affichage montre un état dans lequel aucun programme n'est stocké. Le nombre à 3 chiffres ci-dessus indique le nombre de pas restants. Ce nombre est décrémenté quand un programme est stocké ou quand la mémoire est étendue (voir page 89). Les chiffres 0 à 9 sont des numéros de zone de programme; celui qui clignote indique la zone de programme actuellement spécifiée. Quand un programme est stocké dans cet état, il est stocké dans la zone de programme P0. Différents programmes peuvent être mémorisés dans les 10 zones de programme P0 à P9.

Si un programme est stocké, le numéro de zone de programme n'apparaît pas, mais le curseur, " — ", est affiché. Pour effacer tous les programmes et stocker un programme dans la zone de programme P0, entrer

NEW ALL **ON**

Nous avons ici une commande qui efface tous les programmes. Stocker un programme en procédant comme suit.

10 INPUT A,B — Appuyer sur cette touche à la fin de chaque ligne.

Cette opération peut aussi être exécutée en appuyant sur les touches alphabétiques INPUT.

```
20 C=A+B
30 D=A-B
40 E=A*B
50 F=A/B
60 PRINT C,D,E,F
70 END
```

Après l'appui sur la touche **END**, un espace d'un caractère est ouvert après le numéro de ligne pour faciliter la lecture de l'affichage.

Le programme a-t-il été correctement stocké?

Appuyer sur les touches lentement et fermement, même si c'est fastidieux. Quand on appuie sur une mauvaise touche, une correction peut être faite en procédant comme suit.

• Une erreur a été remarquée avant d'appuyer sur la touche **END**.

Si ceci se produit, mettre le curseur à l'endroit de l'erreur à l'aide des touches **←** et **→** puis corriger.

Exemple 1) 10 INPUT S_ "S" a été entré au lieu de "A".

Mettre le curseur sous le "S" en appuyant une fois sur la touche **←**.

← 10 INPUT S

Appuyer sur la touche correcte.

A 10 INPUT A_

Continuer l'entrée puis appuyer sur la touche **END**.

A 10 INPUT A,B

Exemple 2) 40 EE=A*B_ On a entré un "E" de trop.

Appuyer 5 fois sur la touche **←** pour mettre le curseur sous le deuxième "E".

←←←←← 40 EE=A*B

Vu qu'on doit supprimer un caractère, appuyer une fois sur la touche **DEL**.

DEL 40 E=A*B

Une fois que la suppression est faite, appuyer sur la touche **END**.

40 E=A*B

Exemple 3) RINT C,E,F_ Le "D" de la ligne 60 a été oublié.

Appuyer 4 fois sur la touche **←** pour mettre le curseur à côté de l'endroit où l'insertion doit être faite.

←←←← 60 PRINT C_

Ouvrir un espace d'un caractère.

← 60 PRINT C_

Insérer "D".

D 60 PRINT C,D_

Une fois que la correction est terminée, appuyer sur la touche **END**.

END 60 PRINT C,D

• Une erreur a été remarquée une fois qu'on a appuyé sur la touche **END**.

Vu qu'une ligne est stockée comme partie d'un programme une fois qu'on a appuyé sur la touche **END**, la rappeler à l'aide de la commande LIST si on doit la corriger.

Exemple) La ligne 50, "50 F=A/N", erronée, a été entrée.

Rappeler la ligne 50 à l'aide de la commande LIST.

LIST 50 **END** 50 F=A/N_

L'appui sur **LIST** donne le même résultat.

Appuyer une fois sur la touche **←** pour mettre le curseur sous le "N".

← 50 F=A/N

Appuyer sur la touche correcte.

B 50 F=A/B_

Une fois que la correction est terminée, appuyer sur la touche **END**.

END

Si les lignes 60 et suivantes sont stockées, la ligne 60 est affichée.

60 PRINT C,D

Si aucune autre correction n'est nécessaire, appuyer sur la touche **AC** pour effacer l'affichage.

AC

Une fois qu'on a appuyé sur la touche **END**, une correction peut être faite en rappelant la ligne en question à l'aide de la commande LIST.

Quand une nouvelle ligne est stockée avec un numéro de ligne déjà employé, la ligne stockée en dernier a priorité, et la vieille ligne est effacée. Le stockage d'un programme se fait comme mentionné ci-dessus. Une fois que le stockage est terminé, appuyer sur **MODE** **←** pour repasser au mode RUN.

Si on ne quitte pas le mode WRT, le programme stocké risque d'être effacé ou récrit. Par conséquent, ne pas oublier de revenir au mode RUN une fois que le stockage est terminé.

REMARQUE

ZONES DE PROGRAMME

Cet ordinateur est muni de 10 zones de programme, P0 à P9, où des programmes indépendants peuvent être stockés. Toutes ces zones de programme peuvent être utilisées de la même façon. S'il n'y avait pas ces zones indépendantes et si 3 programmes, par exemple, devaient être utilisés très souvent, il faudrait les charger chaque fois à partir d'une cassette.

Grâce à ladite fonction partage, ces 3 programmes peuvent être stockés dans les zones de programme P0, P1 et P2, par exemple.

Bien que la fonction partage de mémoire soit très pratique, des précautions doivent être prises en ce qui concerne le nombre de pas. La mémoire peut être partagée, mais le nombre total de pas employés dans toutes les zones de programme ne peut pas dépasser la capacité maximale.

Une zone de programme peut être spécifiée en appuyant sur une touche numérique, de 0 à 9, après avoir appuyé sur la touche **WRT**. Cette spécification peut se faire autant en mode RUN qu'en mode WRT. Quand elle est faite en mode RUN, le programme stocké dans la zone spécifiée démarre automatiquement. Quand elle est faite en mode WRT, le programme ne démarre pas. Les zones de programme doivent être correctement utilisées. Quand un programme est stocké ou sauvegardé ou chargé sur ou à partir d'une cassette, si une zone de programme erronée est spécifiée, l'opération ne peut pas être faite correctement.

Lors de la mise sous-tension de l'appareil, la zone de programme P0 est spécifiée automatiquement. Ceci peut être contrôlé par le chiffre suivant "READY" une fois qu'on a appuyé sur **WRT**.

Exemple) **WRT** 0 → READY P3 Zone de programme P3

3-2-4 Exécution d'un programme

Effectuer des calculs à l'aide du programme précédemment stocké. Appuyer sur **WRT** et s'assurer que l'ordinateur est en mode RUN ("RUN" est affiché).

Il y a deux manières différentes d'exécuter un programme.

- (1) Exécution en spécifiant la zone de programme.

Appuyer sur la touche numérique de 0 à 9 qui spécifie la zone de programme après avoir appuyé sur la touche **WRT**. Si un programme est stocké, il démarrera.

Exemple) **WRT** 0

- (2) Exécution à l'aide de la commande RUN.

Exemple) **WRT** **RUN** (de la même manière que **RUN**) **EXE**

La différence entre ces deux méthodes d'exécution est la suivante. Avec la méthode (1), l'exécution commence toujours au début de la zone de programme, tandis qu'avec la méthode (2), elle peut commencer au début ou à partir de n'importe quel numéro de ligne.

Exécuter un programme en utilisant la méthode (1).

Opération

WRT 0

Entrer 2 données.

Exemple)

45 **EXE**

36 **EXE**

Affichage

?

?

81

Lorsque l'instruction PRINT est exécutée, "STOP" est affiché.

Lorsqu'on a entré 2 données, la somme est affichée. Appuyer sur la touche **EXE** pour afficher la réponse suivante.

EXE

EXE

EXE

9

1620

1.25

Maintenant, exécuter un programme à l'aide de la commande RUN. Si on entre RUN **EXE**, le résultat sera le même que celui obtenu avec la méthode (1). Donc, l'exécuter à partir de la ligne 20.

Opération

WRT 20 EXE	81
(ou RUN 20 EXE .)	
EXE	9
EXE	1620
EXE	1.25

Si aucun numéro de ligne n'est spécifié lorsqu'on utilise la commande RUN, l'exécution du programme commencera à partir du début, sinon elle commencera à la ligne spécifiée comme mentionné ci-dessus. En fait, il y a une différence entre les deux méthodes.

Lorsque la commande RUN est utilisée, le programme de la zone spécifiée à ce moment est exécuté. D'autre part, si le programme de la zone P0 est à exécuter et la zone P5 spécifiée, quelle est la marche à suivre? La solution est d'appuyer sur **WRT** **P0**.

Après qu'un programme ait été préparé et stocké, l'exécuter. Si une erreur (ERR est affiché) apparaît après l'exécution, n'en soyez pas découragé. Dans ce cas, trouver la cause de l'erreur (mettre au point) en se référant au paragraphe suivant.

REMARQUE

COMMENT COMPTER LE NOMBRE DE PAS

Cet ordinateur a une capacité mémoire standard de 544 pas et de 1568 pas avec le module d'extension de RAM EM545 (option).

Le pas est l'unité qui indique la capacité de la mémoire dans laquelle un programme peut être stocké. Quand un programme est stocké, le nombre de pas restants s'en trouve réduit.

Quand le mode WRT est spécifié en appuyant sur **MOD** **1**, le nombre de pas restants est affiché.

Exemple)

MOD 1	WRT 544	Nombre de pas restants
	P 0123456789	

On compte le nombre de pas de la manière suivante.

- Numéro de ligne 2 pas par numéro de 1 à 9999.
- Commande..... 1 pas
- Fonction..... 1 pas
- Caractère..... 1 pas par caractère.
- De plus, le fait d'appuyer sur la touche **EXE** pendant le stockage utilise 1 pas.

Exemple)

1	INPUT	A	EXE	5 pas
2	1	1		
10	B=SIN	A	EXE	7 pas
2	1 1	1		
100	PRINT	"B=" ; B	EXE	10 pas
2	1	4	1 1	
Total: 22 pas				

- 8 pas sont utilisés pour chaque extension de la mémoire, le cas échéant.

Exemple) Etat initial..... 26 mémoires 544 pas
DEFM 10 **EXE** 36 mémoires 464 pas

3-2-5 Correction des erreurs (Mise au point)

Après qu'un programme ait été préparé et exécuté, il arrive souvent qu'une erreur soit affichée et que le résultat ne puisse donc pas être obtenu. Ne pas se décourager car la cause de cette erreur peut être trouvée.

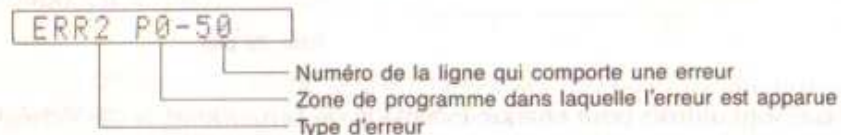
Corriger les erreurs s'appelle "mettre au point".

La méthode de correction des erreurs dépend de leurs causes. Dans certains cas, une erreur est affichée pendant l'exécution, dans d'autres cas, aucune erreur n'est affichée mais le résultat ne peut pas être obtenu comme il devrait l'être. Quand une erreur est affichée pendant l'exécution, sa position et son type sont indiqués. On peut alors aisément en trouver la cause. Mais quand un résultat correct n'est pas obtenu et aucune erreur affichée, c'est plus ennuyeux.

Corrigeons progressivement les erreurs.

(1) Correction avec affichage de l'erreur.

L'affichage de l'erreur comporte un "message d'erreur" composé de trois éléments.



Le type de l'erreur est indiqué par le numéro de code qui suit "ERR". Ce dernier va de 1 à 9 et détermine les types d'erreurs tels que "ERR1" pour un dépassement mémoire et "ERR2" pour une erreur de syntaxe. Voir page 190 le paragraphe "Table de messages d'erreur" pour la signification de ces numéros de code.

La zone de programme dans laquelle l'erreur est apparue, est indiquée. De même que le numéro de ligne qui comporte une erreur.

Ces trois éléments permettent donc de déterminer la position et le type de l'erreur.

Prenons un exemple.

Une erreur qui apparaît souvent est "ERR2" qui correspond à une erreur de syntaxe. Elle survient lorsqu'un programme n'est pas correctement stocké.

Par exemple, le programme utilisé dans le précédent exemple n'est pas correctement stocké.

Opération

STO 1
20 C=A+B
30 C=AB
40
AC END

Affichage

P	123456789
20	C=A+B
20	C=AB
30	C=A-B
READY P0	

Dans cet exemple, "C=A+B" de la ligne 20 a été entré par erreur sous la forme "C=AB". Maintenant exécutons le programme.

Opération

STO P0
45
12

Affichage

?
?
ERR2 P0-20

Un message d'erreur est affiché et indique qu'une erreur de syntaxe est apparue à la ligne 20 dans la zone de programme P0.

Contrôler la ligne 20.

Opération

AC Sortie d'erreur
STO 1
20 C=AB

Affichage

P	123456789
20	C=AB

Contrôler si la ligne est correctement écrite. Comme "+" a été oublié entre A et B, faire la correction.

Opération

STO
20 C=A+B
30 C=A-B
AC END

Affichage

20	C=AB
20	C=A+B
30	C=A-B
READY P0	

Etant donné que "ERR2" survient le plus souvent à cause d'une erreur faite au moment de l'entrée du programme, contrôler la ligne du programme dont le numéro est affiché dans le message d'erreur. De même, lorsque des données sont chargées par un ordre READ (voir page 92), si des données alphanumériques sont lues dans une variable numérique, "ERR2" est affiché comme précédemment. Lorsqu'une erreur survient à l'endroit d'une instruction comportant un ordre READ, contrôler également comment les données ont été déclarées.

Les contrôles correspondant aux différentes erreurs sont les suivants.

- EER1: Manque de place mémoire. Dépassement.
S'assurer du nombre de pas restants. Contrôler si la mémoire n'a pas été étendue par erreur au delà de la capacité par une commande DEFM. Contrôler si les expressions de calcul ne sont pas trop compliquées.
- ERR2: Erreur de syntaxe
Contrôler si il y a des erreurs dans le programme stocké.
- ERR3: Erreur mathématique
Contrôler si un résultat arithmétique ou une expression numérique n'est pas supérieur à 10^{99} ou si le domaine des fonctions n'est pas dépassé. Quand des variables sont utilisées, contrôler leur contenu. (La division par zéro ou l'extraction de la racine carrée d'une valeur négative arrivent souvent.)
- ERR4: Numéro de ligne inconnu
La spécification du numéro de ligne contenu dans un ordre GOTO, GOSUB ou RESTORE est incorrecte. Vérifier ce numéro de ligne.
- ERR5: Erreur d'argument
Contrôler la valeur de l'argument ou du paramètre d'une commande ou d'une fonction. Quand des variables sont utilisées, contrôler leur contenu.
- ERR6: Erreur de variable
Quand un tableau est utilisé, contrôler si la mémoire a été étendue par un ordre DEFM. Contrôler également si la même variable n'est pas utilisée à la fois comme variable numérique et comme variable alphanumérique.
- ERR7: Erreur d'emboîtement
Si la ligne à laquelle l'erreur survient est une instruction RETURN ou une instruction NEXT, contrôler si la correspondance avec les instructions GOSUB ou FOR est correcte. En outre, si la ligne à laquelle l'erreur survient est une instruction GOSUB ou une instruction FOR, contrôler s'il n'y a pas plus de 8 emboîtements pour l'instruction GOSUB et plus de 4 pour l'instruction FOR.
- ERR8: Erreur de mot de passe
Quand un mot de passe est spécifié, contrôler si un autre n'a pas été entré ou si LIST, NEW, NEW ALL, etc. ont été utilisées.

- ERR9: Erreur d'option
Contrôler si l'interface cassette CI546 ou la mini-imprimante MI547 sont bien branchés. Contrôler si l'accumulateur de la MI547 est chargé et si le papier peut correctement se dérouler. Egalement régler le volume ou la tonalité du magnétophone à cassette connecté à la CI546, nettoyer les têtes du magnétophone et changer la cassette. Ou bien, utiliser le magnétophone en ne branchant que la fiche blanche lors de l'enregistrement et que la noire lors de la lecture.

Les opérations mentionnées ci-dessus seront expliquées plus loin. Néanmoins, pour plus de détails, consulter le paragraphe "REFERENCE DE COMMANDES" page 117.

(2) Bien qu'aucune erreur ne soit affichée, le résultat désiré ne peut être obtenu.

Etant donné que cela apparaît souvent quand une expression de calcul ou une variable d'un programme ont été incorrectement utilisés, contrôler les opérations des expressions de calcul et les variables. Spécialement lorsqu'on n'obtient pas un résultat correct, comparer l'expression originale avec celle du programme.

Quand l'exécution d'un programme ne s'arrête pas ou s'arrête sans que le travail ait été effectué, vérifier le fonctionnement de la variable qui contrôle le déroulement du programme. En ce qui concerne une expression de calcul, vérifier son emplacement en mode WRT (appuyer sur  (1)). Le déroulement du programme peut être contrôlé en l'arrêtant par une commande STOP après avoir entré des données dans la variable de contrôle ou en affichant la valeur d'une variable par une commande PRINT. Stocker le programme suivant.

```

10 INPUT A
20 B=1
30 FOR C=1 TO A
40 B=B*C
50 C=C+1
60 NEXT C
70 PRINT B
80 END

```


Ce programme obtient la factorielle des données entrées par une instruction INPUT. En fait, la ligne 50 n'est pas nécessaire et la variable utilisée dans la boucle FOR ne doit pas être changée.

Les instructions FOR-NEXT des lignes 30 et 60 forment une boucle dans laquelle le calcul est exécuté plusieurs fois. Ces instructions FOR-NEXT seront expliquées plus loin, pour l'instant stocker simplement le programme.

Opération

MODE 1
NEW EX

Commande d'effacement du programme.

10 INPUT A EX
20 B=1 EX
30 FOR C=1 TO A EX
40 B=B*C EX
50 C=C+1 EX
60 NEXT C EX
70 PRINT B EX
80 END EX

Affichage

P 123456789
P 123456789
P 123456789

10 INPUT A
20 B=1
30 FOR C=1 TO A
40 B=B*C
50 C=C+1
60 NEXT C
70 PRINT B
80 END

Appuyer sur AC MODE pour spécifier le mode RUN.

Opération

Exemple) MODE 1
12 EX

Affichage

?
10395

La réponse correcte est 479001600.

Contrôler l'expression de calcul. Elle ne comporte pas d'erreur. Ensuite, contrôler le déroulement de la boucle FOR-NEXT.

Insérer une commande STOP après la ligne 50 pour arrêter le programme à chaque fois.

Opération

MODE 1
55 STOP EX
AC MODE

Affichage

P 123456789
55 STOP
READY P1

Etant donné qu'une commande STOP doit être insérée après la ligne 50, placer un numéro de ligne entre 50 et 60. Choisir, par exemple, 55. Exécuter le programme.

Opération

MODE 1
12 EX

Affichage

?
STOP

Contrôler la valeur de la variable de contrôle de boucle C.

C EX

2

Continuer l'exécution

EX

STOP

Contrôler de nouveau la valeur de la variable C.

C EX

4

Vu que la valeur de la variable C doit être incrémentée de 1 à chaque fois, elle l'est de 2. On a donc trouvé que le déroulement de la boucle FOR-NEXT est incorrect. Contrôler de nouveau l'incrément de la variable C. On a trouvé que le problème vient de la ligne 50 qui n'est pas nécessaire. De ce fait, supprimer la ligne 50 et la commande STOP qui a été ajoutée à la ligne 55.

Opération

MODE 1
50 EX
55 EX
AC MODE

Affichage

P 123456789
-
-
READY P1

La mise au point est terminée.

Il y a une autre manière de corriger les erreurs que celle qui consiste à insérer une commande STOP. C'est la correction dans le mode analyse (appuyer sur MODE 2). "TR" est affiché). En ce qui concerne la correction des erreurs dans le mode analyse, un programme est exécuté et s'arrête après chaque instruction. Etant donné que la valeur d'une variable, etc. est contrôlée pendant que le programme est arrêté, on poursuit la correction en appuyant sur EX pour passer à l'instruction suivante.

Essayer cela avec le précédent exemple. Chaque fois qu'on appuie sur EX, la zone de programme et le numéro de ligne sont affichés.

Appuyer sur MODE 3 pour quitter le mode analyse, "TR" disparaît.

Etant donné que la correction des erreurs peut être effectuée comme mentionné ci-dessus, quand une erreur est affichée ou quand le résultat ne peut être obtenu, ne pas se décourager mais tenter la correction.

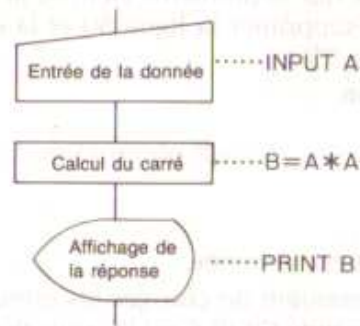
3-3. DEVELOPPEMENT DU PROGRAMME

Il est maintenant certain que la configuration d'un programme a été comprise grâce aux explications précédentes. Les trois parties d'un programme sont entrée, calcul et sortie. De nombreux programmes différents peuvent être préparés avec ces trois éléments. Néanmoins, un programme sera plus facile à utiliser si on apprend les commandes expliquées dans ce paragraphe.

3-3-1 Changement du déroulement d'un programme (instruction GOTO)

En plus des trois parties de base d'un programme, les instructions GOTO sont très pratiques lorsqu'un calcul doit être fait plusieurs fois, ou pour continuer le déroulement d'un programme à une ligne arbitraire au lieu de l'exécuter dans l'ordre des numéros de ligne.

Par exemple, préparons un programme qui calcule le carré d'une certaine valeur. Ce programme peut être décomposé en trois parties qui sont: "entrée de la donnée", "calcul du carré" et "affichage de la réponse". Faisons un organigramme.



Préparons le programme correspondant à cet organigramme.

```
10 INPUT A
20 B=A*A
30 PRINT B
40 END
```

Par exemple, on veut obtenir les carrés de 15 et de 43.

Opération

```
RUN [EXE]
15 [EXE]
RUN [EXE]
45 [EXE]
```

Affichage

?
225
?
1849

Etant donné que le programme doit être exécuté à chaque fois, ce n'est pas très pratique lorsqu'il y a beaucoup de données à entrer.

Pensez-vous qu'il soit plus pratique que le programme exécute plusieurs calculs à la suite? C'est l'instruction GOTO qui le permet. La fonction de cette instruction est de brancher le programme au numéro de ligne ou à la zone de programme indiqué par la valeur numérique qui suit GOTO. Ajouter une instruction GOTO au programme en remplaçant l'instruction END de la ligne 40 par une instruction GOTO.

```
40 GOTO 10
```

Cela signifie que le déroulement du programme se poursuivra à la ligne 10 après la ligne 40.

Exécuter le programme modifié par ce branchement.

Opération

```
RUN [EXE]
15 [EXE]
[EXE]
43 [EXE]
```

Affichage

?
225
?
1849

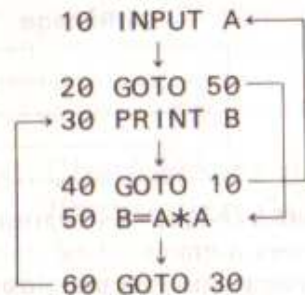
Comme mentionné ci-dessus, une instruction GOTO est pratique pour effectuer des calculs répétitifs.

Etant donné qu'une instruction GOTO peut provoquer le retour au début du programme pour une nouvelle exécution et également provoquer un branchement à un endroit quelconque, il y a beaucoup de manières pratiques de l'utiliser.

Par exemple,

```
10 INPUT A
20 GOTO 50
30 PRINT B
40 GOTO 10
50 B=A*A
60 GOTO 30
```


Le déroulement de ce programme est le suivant.



Etant donné qu'une instruction GOTO provoque un branchement inconditionnel à un numéro de ligne spécifié comme montré ci-dessus, elle est appelée "branchement inconditionnel".

Un branchement à une zone de programme peut être réalisé par une instruction GOTO de la même manière qu'à un numéro de ligne. On spécifie une zone de programme en ajoutant "#" et un chiffre de 0 à 9 après GOTO.

Exemple) GOTO #1 branchement à la zone de programme P1.
GOTO #9 branchement à la zone de programme P9.

Lorsqu'on provoque un branchement dans une zone de programme, l'exécution continue à partir du début du programme de cette zone.

REMARQUE

INSTRUCTIONS PRINT

Une instruction PRINT permet d'afficher le contenu d'une variable, une chaîne de caractères ou une valeur numérique. Les variables numériques comme les variables alphanumériques peuvent être utilisées.

Exemple) Lorsque A=123 PRINT A → 123
Lorsque B\$="ABC" PRINT B\$ → ABC

Etant donné qu'une chaîne de caractères placée entre guillemets est affichée telle quelle elle peut être utilisée comme un message.

Exemple) PRINT "SIGNE" → SIGNE

Lorsqu'on doit afficher plusieurs éléments, ils peuvent être écrits à la suite séparés par des virgules ou des points-virgules.

Exemple) PRINT A,B,Z\$
PRINT "TOTAL=";T

Noter que lorsqu'on utilise une virgule, cela provoque l'affichage d'un "STOP" après l'affichage du premier contenu et l'arrêt de l'exécution. L'affichage suivant est obtenu en appuyant sur la touche . Mais, lorsqu'un point-virgule est utilisé, l'exécution n'est pas arrêtée; l'affichage continue.

Exemple) Essayer cela avec le programme suivant.

```

10 A=123
20 B$="ABC"
30 PRINT A,B$ ..... Après affichage du contenu de A, le
                        contenu de B$ est affiché si on ap-
                        puye sur la touche .
40 PRINT A;B$ ..... Le contenu de B$ est affiché à la
                        suite du contenu de A.
50 PRINT B$; ..... Après affichage du contenu de B$,
                        l'exécution du programme se
                        poursuit.
60 PRINT A ..... Après affichage du contenu de A, le
70 END ..... programme s'arrête.
  
```

Ce programme est exécuté de la manière suivante.

Opération	Affichage
RUN	123
	ABC
	123ABC
	ABC 123
	} PRINT A,B\$
	...PRINT A;B\$
	...PRINT B\$;
	...PRINT A

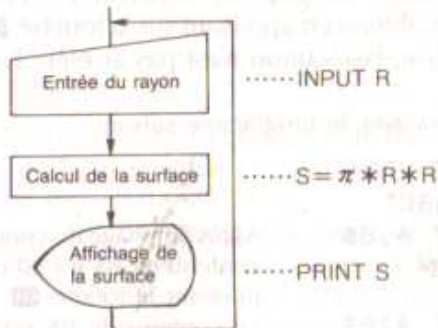
Il y a un espace avant la valeur numérique 123. C'est la place du signe (+ ou -). Etant donné que le signe plus est toujours omis, un espace est ouvert.

[EXERCICE]

Préparer un programme qui calcule la surface de cercles dont on entre les rayons. Utiliser une instruction GOTO.

Expression: $S = \pi r^2$ (Appuyer sur π pour entrer Pi.)

L'organigramme est le suivant. S et R sont utilisées comme variables en concordance avec l'expression.

**PROGRAMME**

```

10 INPUT R
20 S=pi*R*R
30 PRINT S
40 GOTO 10

```

ou

```

10 INPUT R
20 S=pi*R^2
30 PRINT S
40 GOTO 10

```

"12" est également utilisé pour un calcul de carré.

3-3-2 Tests (instruction IF-THEN)

Il serait pratique qu'un programme puisse déterminer une grandeur et effectuer automatiquement des contrôles.

Une instruction IF prend une décision dans un programme; elle effectue un test sur une expression conditionnelle.

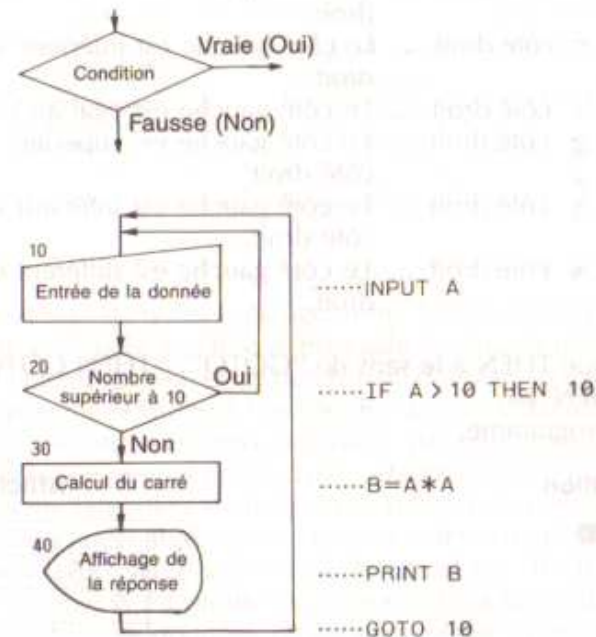
IF condition THEN { Numéro de ligne ou de zone de programme }
Instruction

Si la condition est vraie, un branchement au numéro de ligne ou à la zone de programme qui suit THEN est effectué ou l'instruction exécutée. Si la condition est fausse, l'exécution du programme se poursuit à la ligne suivante.

Contrôlons la fonction de l'instruction IF.

Exemple) Entrer un nombre arbitraire. S'il est supérieur à 10, revenir en demander un autre, s'il est inférieur ou égal à 10, calculer son carré, afficher la réponse et revenir en demandant un autre.

Ce programme est composé de 4 parties (Entrée, Test d'une condition, Calcul et Affichage). Le symbole suivant est utilisé pour un test dans un organigramme.



Les nombres sur la gauche de l'organigramme sont les numéros de ligne.

```

10 INPUT A
20 IF A>10 THEN 10
30 B=A*A
40 PRINT B
50 GOTO 10

```


La ligne 20 comporte une instruction IF. Si la condition est vraie, l'élément qui suit THEN est exécuté. Dans ce cas, le programme passe à la ligne 10.

Les opérateurs relationnels suivants sont utilisés dans des expressions conditionnelles.

- Côté gauche > côté droit ... Le côté gauche est supérieur au côté droit.
 Côté gauche < côté droit ... Le côté gauche est inférieur au côté droit.
 Côté gauche = côté droit ... Le côté gauche est égal au côté droit.
 Côté gauche $\geq$ côté droit ... Le côté gauche est supérieur ou égal au côté droit.
 Côté gauche $\leq$ côté droit ... Le côté gauche est inférieur ou égal au côté droit.
 Côté gauche $\neq$ côté droit ... Le côté gauche est différent du côté droit.

Etant donné que THEN a le sens de "GOTO", "THEN GOTO 10" peut être écrit "THEN 10".
 Exécuter ce programme.

Opération

RUN [1]
 5 [2]
 [3]
 12 [4]
 9 [5]

Affichage

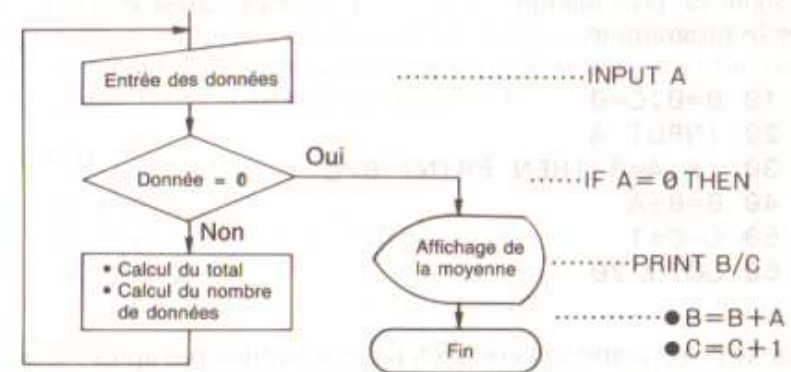
?
25
?
?
81

Les données peuvent être sélectionnées par une instruction IF comme mentionné ci-dessus.

Exemple) Quand on entre "0" après avoir entré plusieurs données, calculer leur moyenne.

Ce programme peut comporter les parties suivantes: "Entrée", "Test de condition", "Calcul" et "Affichage". Le calcul de la moyenne comprend trois parties (faire le total, compter le nombre de données et obtenir la moyenne). Etant donné qu'on ne calcule la moyenne que lorsque zéro est entré, ce calcul suit la partie test.

Faisons un organigramme basé sur cette constatation.



Comme on le voit dans cet organigramme, l'instruction IF contrôle si la donnée entrée dans la variable A est égale à zéro. Si elle est différente de zéro, les calculs du total et du nombre de données sont effectués, après quoi le programme retourne à l'entrée des données. Remarquer que si la première donnée entrée est zéro, la division par zéro provoque une erreur. Cet exemple est un peu compliqué par rapport à l'exemple précédent. En ce qui concerne la partie calcul, la méthode utilisée consiste à ajouter la donnée entrée à la variable utilisée pour calculer le total. Etant donné que B est la variable utilisée pour le total, le calcul est "B = B + A". (Le contenu de la variable A est ajouté au contenu de la variable B). En ce qui concerne le nombre de données, quand une donnée est entrée, 1 est ajouté au compteur (C dans ce cas); "C = C + 1" est effectué.

En ce qui concerne la partie test d'une condition, qui est la partie principale, si A est égal à zéro, la moyenne est affichée par une instruction PRINT.

Etant donné qu'une instruction peut également être écrite après le THEN d'une instruction IF, "PRINT B/C" est écrit après THEN dans cet exemple; le résultat de cette expression est affiché.

En ce qui concerne les variables B et C, si les contenus de ces variables ne sont pas égaux à zéro au début du programme, elles continueront à être incrémentées et on obtiendra un mauvais résultat. De ce fait, elles doivent être mises à zéro ("B = 0", "C = 0"). Bien que des numéros de ligne séparés puissent être utilisés pour ces deux expressions d'affectation,

il sera plus facile d'utiliser une instruction multiple (écrite sous la forme "B=0: C=0" sur une seule ligne en séparant les deux instructions par le signe de ponctuation ":". Stocker le programme.

```
10 B=0:C=0
20 INPUT A
30 IF A=0 THEN PRINT B/C
40 B=B+A
50 C=C+1
60 GOTO 20
```

Bien que le programme soit complet, il ne se termine pas après l'affichage de la moyenne. Alors, ajouter une instruction END après l'instruction PRINT de la ligne 30 en utilisant une instruction multiple.

```
30 IF A=0 THEN PRINT B/C:END
```

Comme indiqué ci-dessus, dans une instruction IF, un test est effectué à l'aide de l'expression conditionnelle.

• Applications de l'instruction IF

Dans l'exemple ci-dessus, le déroulement du programme était conditionné par un test. Mais s'il y a plusieurs tests de condition et que ces dernières doivent être toutes vérifiées, quelle est la solution?

Par exemple, on doit entrer une valeur numérique arbitraire et on doit sélectionner les valeurs numériques comprises entre 1 et 9. En d'autres termes, étant donné que les valeurs numériques à sélectionner doivent être supérieures à zéro et inférieures à 10, deux conditions sont nécessaires ("0 < variable" et "variable < 10"). Cela peut être écrit sur une ligne de la manière suivante.

```
IF 0 < variable THEN IF variable < 10 THEN .....
```

Bien que le même type d'instruction soit utilisé lorsqu'il y a trois conditions ou plus, il est recommandé de n'en mettre que deux au maximum car l'emploi de plus de deux conditions est trop difficile et la ligne devient trop longue.

REMARQUE

INSTRUCTION MULTIPLE (:)

Une instruction multiple est très pratique lorsque de courtes expressions d'affectation sont disposées sur une ligne ou lorsqu'il y a plusieurs commandes après THEN dans une instruction IF.

Exemple 1)

```
10 A=1
20 B=2
30 C=3
...
⇒ 10 A=1:B=2:C=3
...

```

Exemple 2)

```
50 C=A+B
60 D=A-B
70 E=A*B
80 F=A/B
...
⇒ 50 C=A+B:D=A-B:E=A*B:F=A/B
...

```

Lorsqu'une instruction multiple est utilisée après THEN dans une instruction IF, elle n'est exécutée que lorsque l'expression conditionnelle est vraie. A cause de cela des précautions doivent être prises.

Exemple 3)

```
IF A<0 THEN A=10:B=20:.....
```

Exécutée que lorsque la condition est vraie.

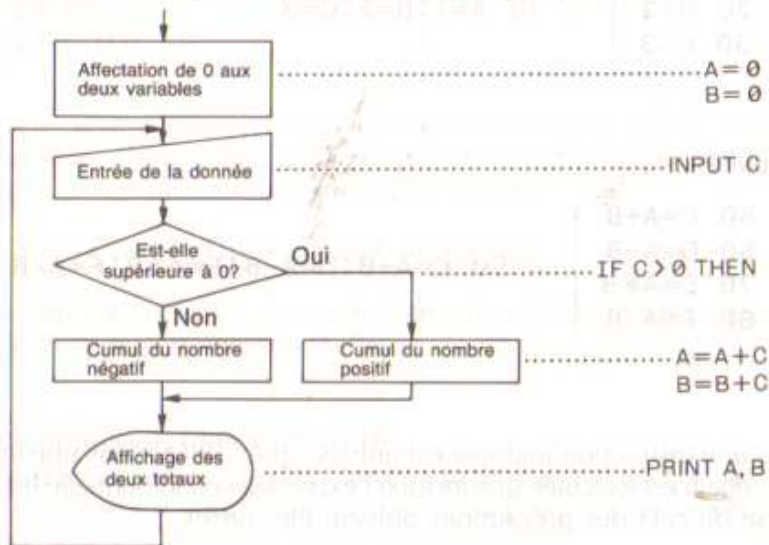
Une instruction multiple est pratique pour ordonner un programme. Néanmoins, comme une ligne n'est pas facile à lire si elle est trop longue, faire en sorte qu'elle soit d'une longueur appropriée et écrire les éléments restants sur la ligne suivante.

[EXERCICE]

Préparer un programme qui sépare en deux groupes les valeurs numériques entrées (supérieures et inférieures à zéro) et effectue le total de chacun de ces groupes.

< SUGGESTION >

Après l'entrée d'une valeur numérique, une instruction IF détermine si le total doit être fait dans l'une ou l'autre variable. Egalement, les variables servant à faire les totaux doivent être initialisées à zéro.

**PROGRAMME**

```

10 A=0:B=0
20 INPUT C
30 IF C>0 THEN A=A+C:GOTO 50
40 B=B+C
50 PRINT A:B
60 GOTO 20
  
```

L'instruction IF de la ligne 30 détermine si la valeur entrée (valeur de la variable C) est supérieure à zéro ou pas. Si elle est supérieure à zéro, elle est ajoutée à la variable A après THEN et si elle est inférieure à zéro, elle est ajoutée à la variable B à la ligne 40. A la ligne 50, les deux totaux sont affichés chaque fois qu'une valeur est entrée.

3-3-3 Répétition d'un programme (instruction FOR-NEXT)

Lorsqu'une instruction GOTO et une instruction IF combinées sont répétées un certain nombre de fois, le programme devient trop long et trop compliqué. Quand le nombre de répétitions est connu, le programme peut être arrangé de manière plus simple. La commande qui effectue cette répétition est l'instruction FOR-NEXT.

Dans une instruction FOR-NEXT, le calcul situé entre l'instruction FOR et l'instruction NEXT est répété un nombre de fois spécifié.

Le format d'une instruction FOR est le suivant.

FOR variable de contrôle = valeur initiale TO valeur finale STEP incrément.

Le format d'une instruction NEXT est le suivant.

NEXT variable de contrôle

Une variable numérique d'un caractère, telle que A ou B, peut seulement être utilisée comme variable de contrôle dans une instruction FOR; un tableau ne peut pas être utilisé. La valeur initiale, la valeur finale et l'incrément peuvent être une expression numérique ou une variable numérique. Pour passer de la valeur initiale à la valeur finale, la valeur de la variable de contrôle est changée de la valeur de l'incrément. L'incrément peut être omis; la valeur prise par défaut est 1.

Stocker et exécuter le programme suivant pour comprendre facilement le fonctionnement de l'instruction FOR-NEXT.

```

10 INPUT A
20 FOR B=1 TO 10 STEP A
30 PRINT B
40 NEXT B
50 GOTO 10
  
```


Ce programme est exécuté de la manière suivante.

Opération

```

RUN [EXE]
3 [EXE]
[EXE]
[EXE]
[EXE]
[EXE]
[EXE]
0.8 [EXE]
[EXE]
[EXE]
[EXE]
...
```

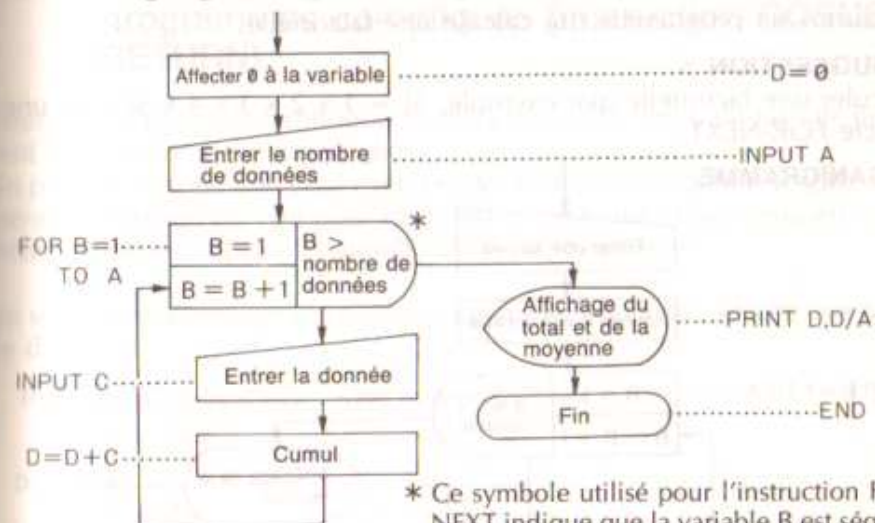
Affichage

?
1
4
7
10
?
1
1.8
2.6
3.4
...

Exemple) Préparer un programme qui effectue le total des données entrées et leur moyenne quand un certain nombre de données est entré.

Pour ce programme, entrer d'abord le nombre de données, puis entrer chaque donnée à l'aide d'une instruction FOR-NEXT et obtenir le total. Une fois que l'entrée de données est terminée, le total et la moyenne sont affichés.

Faisons l'organigramme.



* Ce symbole utilisé pour l'instruction FOR-NEXT indique que la variable B est séquentiellement incrémentée de 1 à partir de 1 et quand sa valeur devient supérieure au nombre de données, le programme quitte la boucle FOR-NEXT et effectue un transfert au travail suivant.

Préparer le programme basé sur cet organigramme.

```

10 D=0
20 INPUT A
30 FOR B=1 TO A
40 INPUT C
50 D=D+C
60 NEXT B
70 PRINT D,D/A
80 END
```

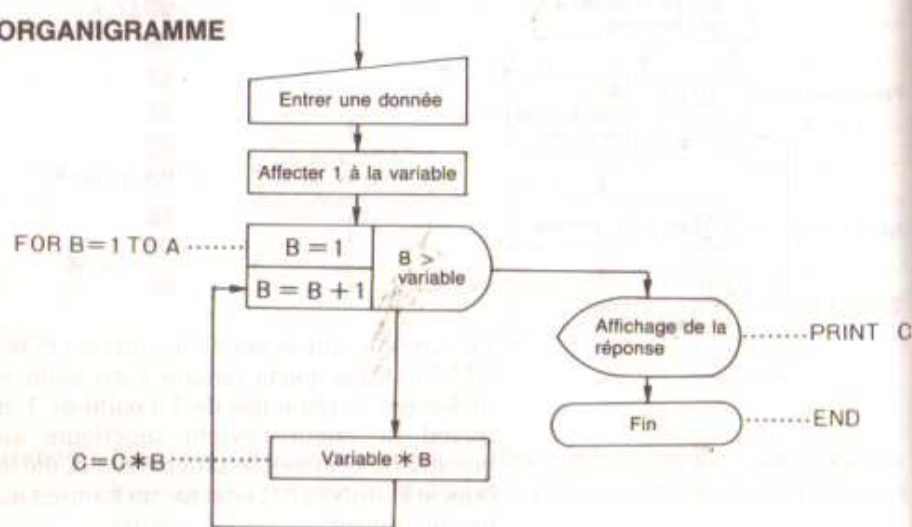
Lorsque le nombre de données est connu comme ci-dessus, l'entrée d'une donnée et le calcul peuvent être répétés par une instruction FOR-NEXT. Le nombre de données peut être directement écrit au lieu de A à la ligne 30 sans entrer ce nombre par une instruction INPUT comme indiqué à la ligne 20. Dans ce cas, la ligne 20 n'est pas nécessaire.

[EXERCICE]

Préparons un programme qui calcule une factorielle.

< SUGGESTION >

Calculer une factorielle (par exemple, $5! = 1 \times 2 \times 3 \times 4 \times 5$) avec une boucle FOR-NEXT.

ORGANIGRAMME**PROGRAMME**

```

10 INPUT A
20 C=1
30 FOR B=1 TO A
40 C=C*B
50 NEXT B
60 PRINT C
70 END
  
```

A la ligne 20, la valeur 1 est affectée à la variable C qui calcule la factorielle car on obtiendrait un résultat faux si la variable C n'était pas initialisée à 1.

Le calcul de factorielle est effectué par l'instruction FOR-NEXT (lignes 30 à 50) dans laquelle la valeur de la variable B est incrémentée de 1, ainsi on obtient la factorielle en calculant $1 \times 2 \times 3 \times 4 \times \dots$. Le programme est terminé après un calcul par l'instruction END de la ligne 70. Néanmoins, si on devait faire plusieurs calculs de manière répétitive, la ligne 70 devrait être transformée en "GOTO 10".

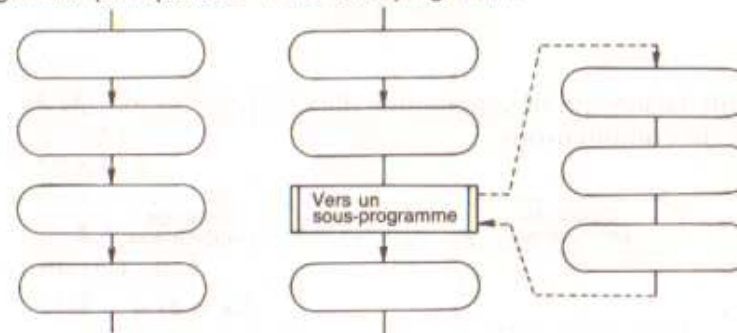
3-3-4 Un sous-programme pratique pour des programmes compliqués (instruction GOSUB-RETURN)

Vous êtes maintenant habitués à la préparation des programmes, ils peuvent devenir longs et compliqués.

Un programme pratique pour exécuter des traitements répétitifs et spécialement utile lorsqu'on compose de longs programmes dans lesquels une même partie est répétée, s'appelle un sous-programme.

Un sous-programme appelé par un programme principal exécute une partie du travail.

Programme principal seul Avec sous-programme



Contrôler la fonction d'un sous-programme en se servant d'un exemple.

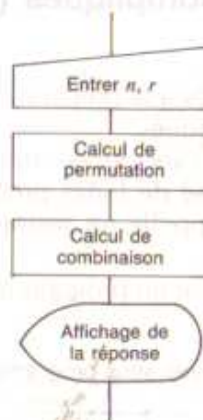
Exemple) Préparer un programme qui calcule permutations et combinaisons.

Expression: Permutations ${}_nP_r = \frac{n!}{(n-r)!}$

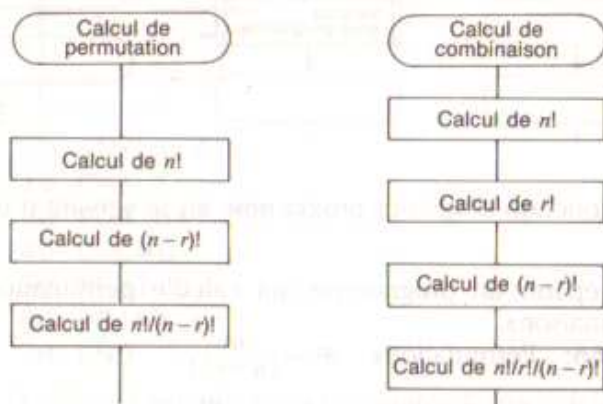
Combinaisons ${}_nC_r = \frac{n!}{r!(n-r)!}$

Ce programme calcule les permutations et les combinaisons de deux données entrées (n, r).

Faisons un organigramme simple.



Maintenant, faisons un organigramme détaillé pour les calculs de permutations et de combinaisons.



Les calculs de $n!$ et $(n-r)!$ sont communs aux calculs de permutation et de combinaison. Egalement, trois calculs de factorielle différents sont exécutés.

EXEMPLE DE PROGRAMME (1)

```

10 INPUT N,R
20 A=1
30 FOR B=1 TO N
40 A=A*B
50 NEXT B
60 E=A
70 A=1
80 FOR B=1 TO N-R
90 A=A*B
100 F=A
110 NEXT B
120 P=E/F
130 A=1
140 FOR B=1 TO R
150 A=A*B
160 NEXT B
170 G=A
180 C=E/G/F
190 PRINT P,C
200 END
  
```

Annotations du programme :

- Calcul de $n!$: lignes 20 à 60
- Calcul de $(n-r)!$: lignes 70 à 110
- Calcul de $\frac{n!}{(n-r)!}$: ligne 120
- Calcul de $r!$: lignes 140 à 170
- Calcul de $\frac{n!}{r!(n-r)!}$: ligne 180

CONTENU DES VARIABLES

N : n
 R : r
 P : Permutation
 C : Combinaison
 A : Pour la factorielle
 B : Pour la boucle FOR-NEXT
 E : $n!$
 F : $(n-r)!$
 G : $r!$

Dans ce programme, trois calculs de factorielle utilisent un programme commun excepté pour la valeur finale d'une instruction FOR. Si cette valeur finale pouvait être donnée par une variable commune, les trois calculs pourraient être faits en commun. Dans ce cas, utiliser un sous-programme est vraiment efficace.

En vue d'effectuer les trois calculs de factorielle en commun, utiliser la variable H pour les valeurs finales. Il est nécessaire d'entrer séquentiellement dans la variable H les valeurs de n , $n-r$ et r . Maintenant ce programme est modifié et comporte un sous-programme; une commande qui transfère le travail à un sous-programme et une commande qui provoque un retour après la fin du travail sont nécessaires. Ces commandes sont "GOSUB" et "RETURN".

EXEMPLE DE PROGRAMME (2)

```

10 INPUT N,R
20 H=N
30 GOSUB 150
40 E=A
50 H=N-R
60 GOSUB 150
70 F=A
80 P=E/F
90 H=R
100 GOSUB 150
110 G=A
120 C=E/G/F
130 PRINT P,C
140 END
150 A=1
160 FOR B=1 TO H
170 A=A*B
180 NEXT B
190 RETURN

```

Sous-programme

Un programme qui comporte une partie commune peut être simplement préparé en utilisant un sous-programme comme indiqué ci-dessus. Bien que ce programme soit plus court de seulement une ligne, cela a une fonction importante quand un programme est plus compliqué ou plus long.

[EXERCICE]

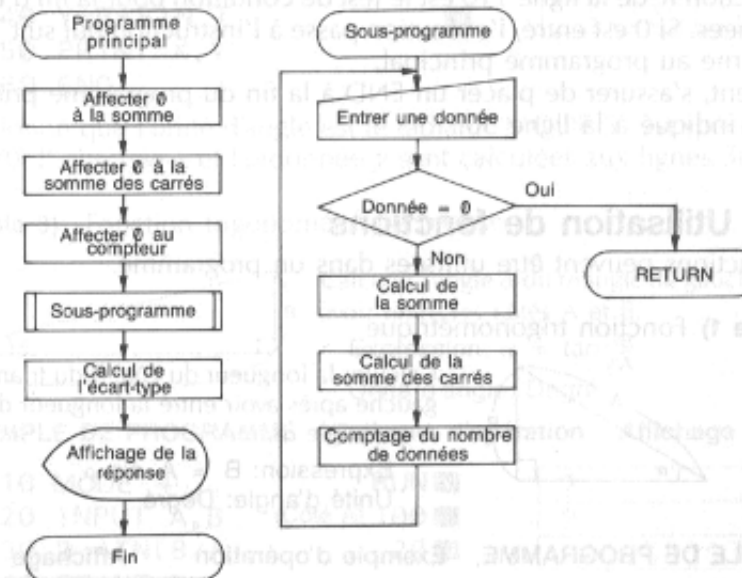
Préparer un programme qui calcule un écart-type. L'entrée des données, le calcul de la somme, le calcul de la somme des carrés et le comptage du nombre de données sont effectués dans un sous-programme.

Expression: $\sigma_n = \sqrt{\frac{\sum x^2 - (\sum x)^2/n}{n}}$

$\sum x$: Somme
 $\sum x^2$: Somme des carrés
 n : Nombre de données

< SUGGESTION >

Après l'entrée de données, si "0" est détecté par une instruction IF, un retour est opéré au programme principal pour obtenir l'écart-type. SQR est utilisé pour la racine carrée.



PROGRAMME

```

10 B=0:C=0:D=0
20 GOSUB 100 ..... Vers le sous-programme
30 E=SQR((C-B*B/D)/D) ..... Ecart-type
40 PRINT E
50 END
100 INPUT A
110 IF A=0 THEN RETURN
120 B=B+A ..... Somme
130 C=C+A*A ..... Somme des carrés
140 D=D+1 ..... Nombre de données
150 GOTO 100

```

Sous-programme

Dans ce programme, l'exécution est transférée au sous-programme à la ligne 20; l'entrée des données, le calcul de la somme, le calcul de la somme des carrés et le comptage du nombre de données sont effectués à partir de la ligne 100 de ce sous-programme.

L'instruction IF de la ligne 110 est le test de condition pour la fin d'entrée de données. Si 0 est entré, l'exécution passe à l'instruction qui suit THEN et retourne au programme principal.

Egalement, s'assurer de placer un END à la fin du programme principal comme indiqué à la ligne 50.

3-3-5 Utilisation de fonctions

Des fonctions peuvent être utilisées dans un programme.

Exemple 1) Fonction trigonométrique



Calculer la longueur du côté B du triangle de gauche après avoir entré la longueur du côté A et l'angle α .

Expression: $B = A \cdot \sin \alpha$
Unité d'angle: Degré

EXEMPLE DE PROGRAMME

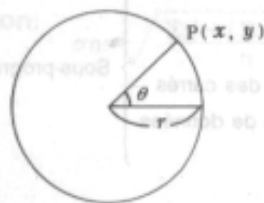
```
10 MODE 4
20 INPUT A,D
30 B=A*SIN D
40 PRINT B
50 END
```

Exemple d'opération
(Côté A) 15
(Angle α) 30

Affichage
?
?
7.5

L'unité d'angle est spécifiée à la ligne 10. Etant donné que le calcul est effectué en degrés, "MODE 4" est entré. La fonction trigonométrique est utilisée pour effectuer le calcul à la ligne 30.

Exemple 2) Fonction trigonométrique



Calculer les coordonnées du point P du cercle de gauche après avoir entré le rayon et l'angle θ .

Expressions: $x = r \cdot \cos \theta$
 $y = r \cdot \sin \theta$

Unité d'angle: Radian

EXEMPLE DE PROGRAMME

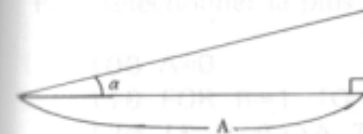
```
10 MODE 5
20 INPUT R,T
30 X=R*COS T
40 Y=R*SIN T
50 PRINT X,Y
60 END
```

Exemple d'opération
(Rayon) 5
(Angle θ) $\pi/3$

Affichage
?
?
2.5
4.330127019

Etant donné que l'unité d'angle est le radian, "MODE 5" est spécifié à la ligne 10. L'abscisse x et l'ordonnée y sont calculées aux lignes 30 et 40.

Exemple 3) Fonction trigonométrique inverse



Calculer l'angle α du triangle de gauche après avoir entré les côtés A et B.

Expression: $\alpha = \tan^{-1} \frac{B}{A}$

Unité d'angle: Degré

EXEMPLE DE PROGRAMME

```
10 MODE 4
20 INPUT A,B
30 D=ATN(B/A)
40 PRINT D
50 END
```

Exemple d'opération
(Côté A) 100
(Côté B) 20

Affichage
?
?
11.30993247

Exemple 4) Conversion décimal $\leftrightarrow$ sexagésimal

Préparer un programme qui effectue un calcul de temps.

EXEMPLE DE PROGRAMME

```
10 T=0
20 INPUT D,E,G
30 S=SGN D
40 D=ABS D
50 T=T+S*DEG(D,E,G)
60 PRINT DMS$(T)
70 GOTO 20
```

Exemple d'opération
(Heures) 1
(Minutes) 25
(Secondes) 36

Affichage
?
?
?
1° 25' 36"
?
?
?
3° 40' 41"

A la ligne 20, entrer heures, minutes et secondes respectivement dans trois variables D, E et G.

Les lignes 30 et 40 sont utilisées pour la soustraction. Si les heures sont entrées avec un signe moins, le temps entré est soustrait du résultat précédent. Si l'addition seule est exécutée, ces lignes ne sont pas nécessaires. Le total est obtenu à la ligne 50. 1 ou -1 sont entrés dans la variable S pour effectuer respectivement l'addition ou la soustraction. La fonction DEG convertit du sexagésimal (heures, minutes et secondes) au décimal et le total est effectué en décimal.

La ligne 60, pour l'affichage, utilise la fonction DMS\$ qui convertit du décimal au sexagésimal.

Exemple 5) Génération de nombres aléatoires
Générer des nombres aléatoires de 1 à 99.

EXEMPLE DE PROGRAMME

```
10 R=INT(RAN#*99)+1
20 PRINT R
30 GOTO 10
```

Un nombre aléatoire est généré à la ligne 10. Multiplier le nombre aléatoire généré par 99 puis ajouter 1 au résultat pour obtenir un nombre de 1 à 99 (voir page 154).

Quand le nombre aléatoire

est compris entre 5 et 9 $\rightarrow$ $\text{INT}(\text{RAN}\#*5)+5$

Quand le nombre aléatoire

est compris entre 10 et 20 $\rightarrow$ $\text{INT}(\text{RAN}\#*11)+10$

3-3-6 Utilisation d'un tableau

Différente des variables courantes de A à Z, une variable de tableau est gérée par un nombre (indice).

Un caractère alphabétique peut être utilisé comme nom de variable de tableau avec un indice qui lui est rattaché.

Exemple) A(1)

Indice
Nom de variable

Un tableau est pratique quand on utilise un grand nombre de données. Par exemple, pour entrer 10 données:

```
10 FOR A=1 TO 10
20 INPUT N(A)
30 NEXT A
```

Dans ce cas, le tableau est ordonné de la manière suivante.

Variables courantes	O	P	Q	R	S	T	U	V	W	X
Variables de tableau	N(1)	N(2)	N(3)	N(4)	N(5)	N(6)	N(7)	N(8)	N(9)	N(10)

Pour sélectionner la plus grande donnée parmi 10:

```
100 A=0
110 FOR B=1 TO 10
120 IF N(B)>A THEN A=N(B)
130 NEXT B
140 PRINT A
```

Lorsqu'un tableau est utilisé, des précautions doivent être prises en ce qui concerne la manière de l'ordonner. Certains éléments de tableau sont communément utilisés comme ceux de variables courantes. Par exemple, l'élément A(0) est le même que A et l'élément A(1) le même que B. Ainsi si A(0) et A sont utilisés dans un même programme, le contenu de la variable est modifié.

Les parties communes sont les suivantes.

A	B	C	D	X	Y	Z
A(0)	A(1)	A(2)	A(3)	A(23)	A(24)	A(25)
B(0)	B(1)	B(2)	B(22)	B(23)	B(24)	
				X(0)	X(1)	X(2)
				Y(0)	Y(1)	
						Z(0)

*Pour plus de détails, voir page 180.

Par exemple, effectuer les opérations suivantes.

Affecter 7 à la variable I.

I=7 **EXE**

Confirmer le contenu de la variable I.

I **EXE**

Affecter 10 au 9^e élément de la variable de tableau A.

A(8)=10 **EXE**

Confirmer le contenu de la variable I.

I **EXE**

Le contenu de la variable I a été modifié. Cela vient du fait que l'élément de la variable I et celui de la variable de tableau A(8) sont les mêmes. Quand un tableau est utilisé dans un programme, garder les variables pour les boucles FOR-NEXT et les affectations et ensuite déterminer les noms des variables de ce tableau.

Exemple 1) Quand 10 éléments de tableau sont nécessaires,

Variables de tableau: A(0) — A(9)

Variables courantes: K — Z

Exemple 2) Quand un tableau de 50 éléments et 15 variables sont nécessaires,

Variables courantes: A — O

Variables de tableau: P(0) — P(49) [Entrer DEFM 39 **EXE**]

Exemple 3) Quand la variable A est utilisée pour une boucle FOR-NEXT, la variable B pour le cumul et un tableau de 50 éléments est nécessaire,

10 DEFM 26 Augmenter le nombre de variables de
26 pour en avoir 52.

20 B=0

30 FOR A=0 TO 49

40 INPUT C(A)

50 B=B+C(A)

60 NEXT A

..... Un tableau de C(0) à C(49) peut être utilisé.

Quand de nombreuses données sont gérées par un tableau, des précautions doivent être prises afin que les variables de ce tableau ne recouvrent les autres variables générales nécessaires.

Des précautions doivent aussi être prises en ce qui concerne l'extension mémoire quand un tableau est utilisé. Quand les variables générales (A à Z) sont suffisantes, l'extension est inutile. Quand ces variables ne suffisent pas, ne pas oublier d'étendre la mémoire.

L'extension de la mémoire est effectuée par une commande DEFM qui spécifie la taille de cette extension.

Par exemple, quand une extension de 10 variables est effectuée,

Manuellement: DEFM 10 **EXE**

Dans un programme: Numéro de ligne DEFM 10

On peut utiliser cette commande manuellement ou dans un programme. L'écrire au début du programme quand un tableau est utilisé. Analyser un programme qui utilise un tableau.

Exemple) Entrer des nombres aléatoires compris entre 0 et 59 dans les éléments d'un tableau allant de G(0) à G(59) et les ordonner de manière décroissante pour affichage.

EXEMPLE DE PROGRAMME

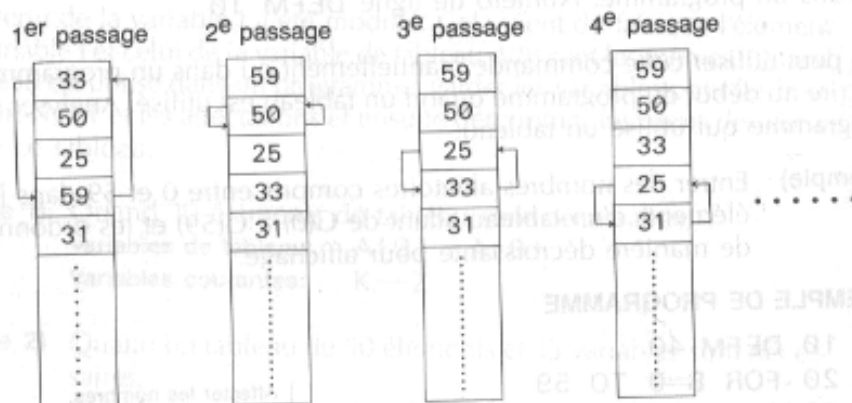
```
10 DEFM 40
20 FOR B=0 TO 59
30 G(B)=INT(RAN#*60)
40 NEXT B
50 BEEP 1
60 FOR B=0 TO 59
70 A=-1
80 FOR C=B TO 59
90 IF G(C)>A THEN A=G(C):D=C
100 NEXT C
110 E=G(B):G(B)=G(D):G(D)=E
120 NEXT B
130 BEEP 0
140 FOR B=0 TO 59
150 PRINT G(B)
160 NEXT B
170 END
```

Affecter les nombres aléatoires compris entre 0 et 59 dans les éléments G(0) à G(59).

Ordonner les éléments de manière décroissante.

Les afficher séquentiellement.

Ce programme est constitué de trois parties. Dans la première, des nombres aléatoires dont les valeurs sont comprises entre 0 et 59, sont générés et affectés dans les éléments d'un tableau allant de G(0) à G(59). Dans la deuxième partie, les éléments sont ordonnés de manière décroissante. Les instructions BEEP des lignes 50 et 130 sont utilisées pour générer un son. Un son aigu est généré par "BEEP 1" et un son grave par "BEEP 0". Etant donné que ces deux lignes ne sont utilisées que pour générer un son qui indique la fin de la préparation des données et du tri, elles peuvent être supprimées du programme. Les lignes 60 à 120 sont exécutées répétitivement pour mettre les données dans l'ordre décroissant.

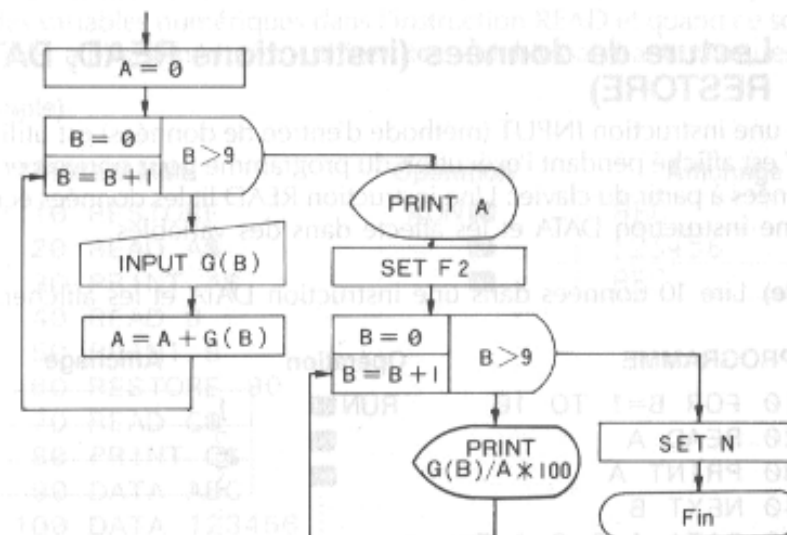


L'instruction DEFM de la ligne 10 est utilisée pour étendre la mémoire. Etant donné qu'il reste 20 variables de G à Z et qu'un tableau de 60 éléments de G(0) à G(59) est nécessaire, une extension mémoire de 40 variables doit être faite.

Un tableau est pratique lorsqu'une grande quantité de données est traitée.

[EXERCICE]

Entrer 10 données et calculer le taux de composition du total de chacune d'elles. Calculer ce taux (pourcentage) avec un maximum de deux décimales.



PROGRAMME

```

10 A=0
20 FOR B=0 TO 9
30 INPUT G(B)
40 A=A+G(B)
50 NEXT B
60 PRINT A
70 SET F2
80 FOR B=0 TO 9
90 PRINT G(B)/A*100
100 NEXT B
110 SET N
120 END

```


Les lignes 20 à 50 sont utilisées pour entrer les données dans G(0) à G(9) par une instruction FOR-NEXT. A la ligne 70, "SET F2" spécifie deux décimales pour le taux de composition. Ce taux est affiché aux lignes 80 à 100. La spécification du nombre de décimales est annulée à la ligne 110.

3-3-7 Lecture de données (instructions READ, DATA, RESTORE)

Quand une instruction INPUT (méthode d'entrée de données) est utilisée, un "?" est affiché pendant l'exécution du programme pour pouvoir entrer les données à partir du clavier. Une instruction READ lit les données écrites dans une instruction DATA et les affecte dans des variables.

Exemple) Lire 10 données dans une instruction DATA et les afficher.

PROGRAMME	Opération	Affichage
10 FOR B=1 TO 10	RUN	1
20 READ A		2
30 PRINT A		3
40 NEXT B		...
50 DATA 1,2,3,4,5,		...
6,7,8,9,10		...
60 END		10

Une instruction READ doit être utilisée conjointement avec une instruction DATA étant donné qu'une donnée provenant de l'instruction DATA est affectée à la variable qui suit "READ". Plusieurs variables peuvent être écrites après un "READ" en les séparant par des virgules.

Exemple)

PROGRAMME	Opération	Affichage
10 READ A\$,B\$	RUN	VILLEMATIN
20 PRINT A\$;B\$		
30 END		
40 DATA VILLE, MATIN		

Une instruction DATA peut être placée à n'importe quel endroit d'un programme. Après l'exécution du programme, les données sont affectées séquentiellement dans des variables à partir de la première donnée de l'instruction DATA qui possède le plus petit numéro de ligne. Quand les données écrites dans une instruction DATA sont des valeurs numériques, utiliser des variables numériques dans l'instruction READ et quand ce sont des données alphanumériques, utiliser des variables alphanumériques.

Exemple)

PROGRAMME	Opération	Affichage
10 RESTORE	RUN	ABC
20 READ A\$		123456
30 PRINT A\$		ABC
40 READ B		
50 PRINT B		
60 RESTORE 90		
70 READ C\$		
80 PRINT C\$		
90 DATA ABC		
100 DATA 123456		
110 END		

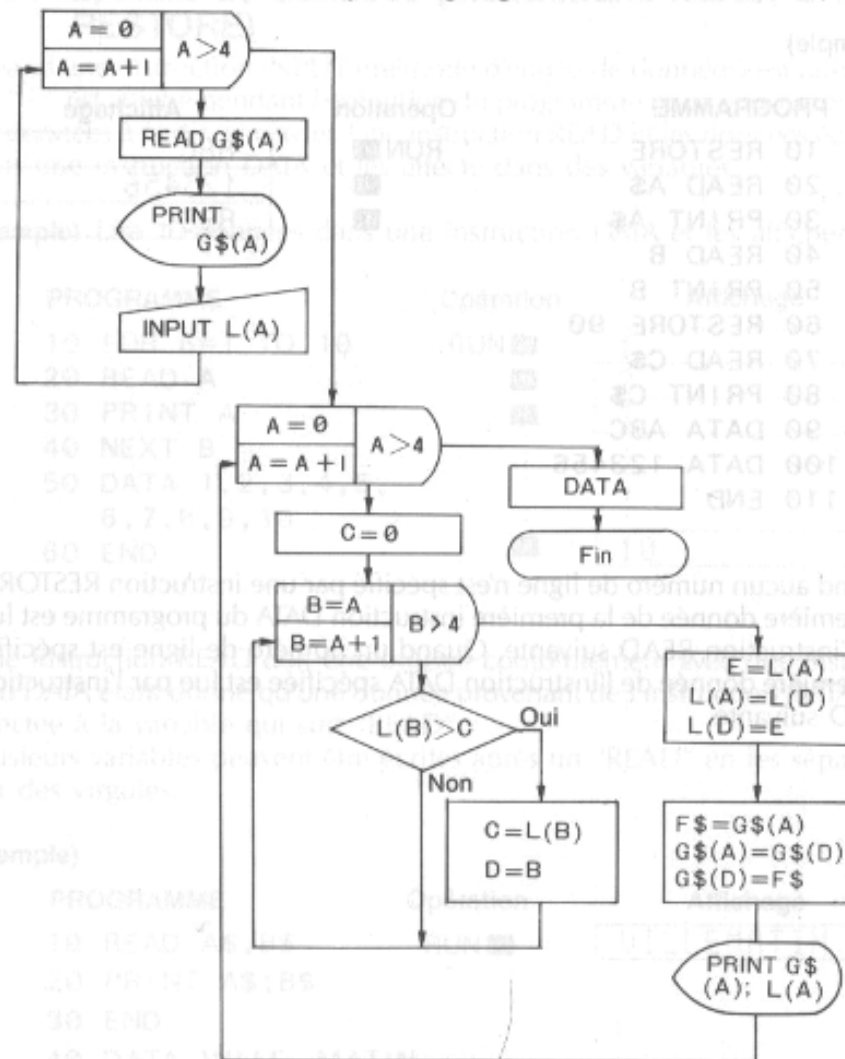
Quand aucun numéro de ligne n'est spécifié par une instruction RESTORE, la première donnée de la première instruction DATA du programme est lue par l'instruction READ suivante. Quand un numéro de ligne est spécifié, la première donnée de l'instruction DATA spécifiée est lue par l'instruction READ suivante.



[EXERCICE]

Lire les noms CHICAGO, LONDON, PARIS, ROME et TOKYO et les ordonner dans l'ordre décroissant de leurs données connexes.

Remarquer que les noms sont affectés dans les éléments de tableau de G\$(0) à G\$(4) et les données entrées dans les éléments de L(0) à L(4).

Exemple d'organigramme**EXEMPLE DE PROGRAMME**

```

10 FOR A=0 TO 4
20 READ G$(A)
30 PRINT G$(A);
40 INPUT L(A)
50 NEXT A
60 FOR A=0 TO 4
70 C=0
80 FOR B=A TO 4
90 IF L(B)>C THEN C=L(B):D=B
100 NEXT B
110 E=L(A):L(A)=L(D):L(D)=E
120 F$=G$(A):G$(A)=G$(D):G$(D)=F$
130 PRINT G$(A);L(A)
140 NEXT A
150 DATA CHICAGO, LONDON, PARIS, ROME, TOKYO

```

VARIABLES

A : Utilisée dans une instruction FOR-NEXT
 B : Utilisée dans une instruction FOR-NEXT
 C : Valeur maximale
 D : Nombre de valeur maximale
 E : Utilisée pour le tri
 F\$: Utilisée pour le tri
 G\$(0) — G\$(4) : Nom de villes
 L(0) — L(4) : Données

Ce programme peut être divisé en deux parties qui sont une partie entrée, des lignes 10 à 50 et une partie tri, des lignes 60 à 140. Dans la partie entrée, les noms de villes sont lus en utilisant une instruction DATA tandis qu'une boucle est effectuée 5 fois par une instruction FOR-NEXT, et les données sont entrées en même temps. L'instruction PRINT de la ligne 30 affiche un nom de ville comme message avant qu'une donnée ne soit entrée par l'instruction INPUT suivante. Les noms de villes et les données sont ordonnés simultanément aux lignes 110 et 120. L'instruction DATA de la ligne 150 peut être placée n'importe où dans le programme.

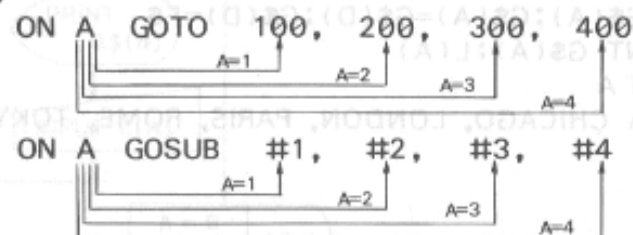
3-3-8 Spécification indirecte (instructions ON-GOTO, ON-GOSUB)

Bien que les spécifications des instructions GOTO et GOSUB soient établies en écrivant le numéro de ligne ou la zone de programme directement dans un programme, le branchement peut parfois dépendre du résultat d'un calcul arithmétique ou d'une donnée pour être effectué. Dans ce cas, tester la condition avec une instruction IF n'est pas pratique.

Les commandes qui effectuent ce type de branchements sont des spécifications indirectes (ON-GOTO et ON-GOSUB).

La fonction d'une instruction ON-GOTO est similaire à celle d'une instruction ON-GOSUB.

Exemple)



Le branchement est effectué au premier emplacement si la valeur de A est 1, au deuxième si la valeur de A est 2, etc. Il dépend donc de la valeur de la variable numérique ou du résultat d'une expression de calcul qui suit "ON". Quand le nombre d'emplacements de branchement est inférieur à la valeur de la variable ou de l'expression de calcul ou quand un emplacement de branchement n'existe pas, l'exécution du programme passe à la ligne suivante ou à la commande suivante dans le cas d'une instruction multiple.

Exemple)

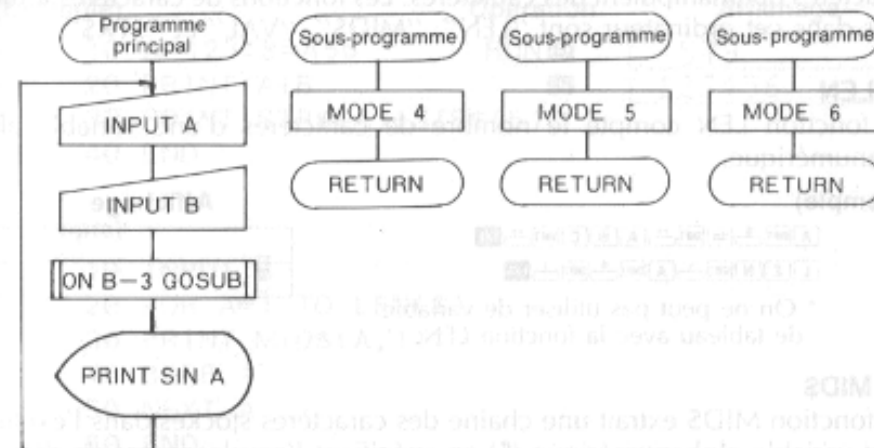
```

10 INPUT A
20 ON A GOTO 100,200,300,400,500
30 PRINT " NO"
40 END
100 PRINT "LINE 100" :END
200 PRINT "LINE 200" :END
300 PRINT "LINE 300" :END
400 PRINT "LINE 400" :END
500 PRINT "LINE 500" :END
  
```

[EXERCICE]

Entrer un angle et une valeur numérique entre 4 et 6, se brancher à un sous-programme qui spécifie l'unité d'angle par une spécification indirecte et calculer le sinus de cet angle.

Organigramme



PROGRAMME

```

10 INPUT "ANGLE",A
20 INPUT "UNIT",B
30 ON B-3 GOSUB 100,200,300
40 PRINT SIN A
50 GOTO 10
100 MODE 4
110 RETURN
200 MODE 5
210 RETURN
300 MODE 6
310 RETURN
  
```

Vu que deux données sont entrées, un message est ajouté à chaque instruction INPUT pour rendre la saisie plus facile. A la ligne 10, l'angle est entré dans la variable A et à la ligne 20, 4, 5 ou 6 est entré dans la variable B pour spécifier l'unité d'angle (voir page 143). A la ligne 30, l'emplacement de branchement est déterminé en convertissant la valeur numérique 4, 5 ou 6 en 1, 2 ou 3 et en utilisant un ON-GOSUB.

Dans les sous-programmes les unités d'angle sont spécifiées.

3-3-9 Fonctions de manipulation de caractères (LEN, MID\$, VAL, STR\$)

De même que SIN et COS sont appelées des fonctions numériques puisqu'elles manipulent des valeurs numériques, il y a des fonctions de caractères qui manipulent des caractères. Les fonctions de caractères fournies dans cet ordinateur sont "LEN", "MID\$", "VAL" et "STR\$".

• LEN

La fonction LEN compte le nombre de caractères d'une variable alphanumérique.

Exemple)

A SHIFT [=] SHIFT A B C SHIFT EXE
LEN SHIFT A SHIFT SHIFT EXE

Affichage

3

* On ne peut pas utiliser de variable de tableau avec la fonction LEN.

• MID\$

La fonction MID\$ extrait une chaîne des caractères stockés dans l'exclusive variable alphanumérique (\$) en spécifiant l'emplacement de début d'extraction et le nombre de caractères à extraire.

Exemple)

	Opération	Affichage
10 \$="VILLE MATIN"		
20 PRINT \$	RUN EXE	VILLE MATIN
30 PRINT MID\$(1,5)	EXE	VILLE
40 PRINT MID\$(7,5)	EXE	MATIN
50 END		

• VAL

La fonction VAL convertit les chiffres stockés dans une variable alphanumérique en valeur numérique.

Exemple)

	Opération	Affichage
10 A\$="123":B\$="456"		
20 PRINT A\$+B\$	RUN EXE	123456
30 PRINT VAL(A\$)+VAL(B\$)	EXE	579
40 END		

* Une variable de tableau ne peut pas être utilisée avec la fonction VAL.

• STR\$

La fonction STR\$ convertit les valeurs numériques stockées dans une variable numérique en une chaîne de caractères; ce qui est l'inverse de la fonction VAL.

Exemple)

	Opération	Affichage
10 A=123:B=456	RUN EXE	579
20 PRINT A+B	EXE	123456
30 PRINT STR\$(A)+STR\$(B)		
40 END		

Exemple)

```

10 INPUT $
20 FOR A=1 TO LEN($)+1
30 PRINT MID$(A,1);
40 BEEP 1
50 NEXT A
60 END
  
```

Ce programme extrait un caractère entré dans l'exclusive variable alphanumérique (\$) à l'aide de la fonction MID\$. Un caractère est extrait chaque fois. L'emplacement d'extraction est spécifié par une instruction FOR-NEXT, la valeur finale est déterminée par la fonction LEN. Un point-virgule est ajouté à la fin de la ligne 30 pour que le programme ne s'arrête pas car un affichage continu est désiré.

Exemple)

```

10 A=1:B=0
20 PRINT "<";STR$(A);">";
30 INPUT $
40 IF $="END" THEN 100
50 B=B+VAL($)+1
60 A=A+1
70 GOTO 20
100 PRINT B/(A-1)
110 END
  
```


Ce programme calcule la moyenne d'un nombre de données inconnu. L'entrée des données est terminée en entrant END ("fin") et la moyenne est affichée en se branchant à la ligne 100.

La ligne 20 fournit un message qui rend la saisie plus facile.

À la ligne 50, étant donné qu'un caractère est entré dans l'exclusive variable alphanumérique, le total est effectué après conversion en valeur numérique.

Egalement étant donné que l'entrée des données se termine en entrant END une erreur (ERR2) se produit si l'on entre autre chose que ces trois caractères.

3-3-10 Fonctions de contrôle d'entrées/sorties (KEY\$, CSR)

Bien que la fonction KEY\$ soit utilisée pour entrer des données, elle diffère de l'instruction INPUT comme indiqué ci-dessous.

Instruction INPUT	Fonction KEY\$
• Une mantisse de 12 chiffres et un exposant de 2 chiffres pour une valeur numérique. • Jusqu'à 7 caractères pour une variable alphanumérique et jusqu'à 30 pour l'exclusive variable alphanumérique.	• Lit le caractère de la touche sur laquelle on a appuyé et l'affecte dans une variable alphanumérique.
• L'attente d'entrée de données est indiquée par l'affichage d'un "?".	• Aucun affichage d'attente d'entrée de données n'apparaît.

Exemple)

```

10 INPUT A
20 PRINT A
30 B$=KEY$
40 IF B$=" " THEN 30
50 PRINT B$
60 END

```

La ligne 10 utilise une instruction INPUT tandis que les lignes 30 et 40 utilisent la fonction KEY\$. Bien que la fonction KEY\$ accepte l'entrée d'un caractère à partir d'une touche du clavier, aucun affichage d'attente d'entrée de données n'apparaît et l'exécution n'est pas stoppée qu'on appuie ou pas sur une touche. C'est pour cette raison que la fonction KEY\$ est combinée avec une instruction IF comme indiqué à la ligne 40. Si l'entrée d'un caractère n'est pas effectuée, le programme retourne à la ligne 30. On utilise donc la fonction KEY\$ en combinaison avec une instruction IF.

Exemple)

```

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

```

Dans l'exemple précédent un contrôle d'appui sur une touche est fait, dans celui-ci un contrôle d'entrée de 1, 2 ou 3 est effectué. Si une condition est vraie, le programme passe à la tâche suivante.

Quand une fonction KEY\$ est utilisée au début d'un programme comme dans cet exemple, porter une attention particulière au lancement du programme. Il y a deux méthodes différentes de lancement de programme. Quand la méthode de lancement de programme  est utilisée, étant donné que l'entrée d'un caractère est effectuée si on continue d'appuyer sur la touche numérique , le chiffre 1 est entré; "LINE100" est alors affiché.

Quand la fonction KEY\$ est utilisée au début d'un programme, ajouter les lignes suivantes.

```

5 A$=KEY$
6 IF A$=" " THEN 5
10 A$=KEY$
  ...

```

} Attend que la touche enfoncée soit relâchée.

La fonction CSR spécifie l'emplacement d'affichage d'une donnée et est utilisée dans une instruction PRINT.

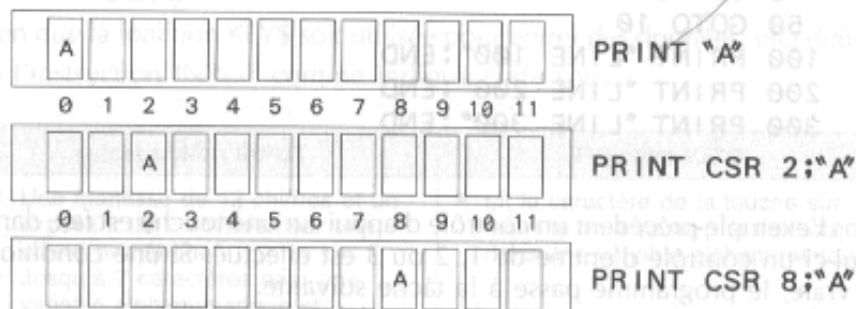
Exemple)

```
10 PRINT "A"
20 PRINT CSR 2;"A"
30 PRINT CSR 8;"A"
40 END
```

On comprendra le fonctionnement de la fonction CSR en exécutant ce programme.

Quand cette fonction est utilisée, l'affichage démarre à l'endroit spécifié (0, 1, 2... ou 11 à partir de la gauche).

Quand elle n'est pas utilisée, l'affichage démarre à l'extrême gauche.



Exemple)

```
10 A=INT(RAN#*12)
20 PRINT CSR A;"↑"
30 GOTO 10
```

Ce programme génère une valeur numérique allant de 0 à 11 en utilisant la fonction RAN# et affiche "↑" à l'aide de la fonction CSR. Après un certain temps, "↑" est affiché à plusieurs endroits différents. Un jeu intéressant peut être préparé en combinant les fonctions KEY\$ et CSR.

Exemple)

```
10 D=0:$=" 0123456789"
20 FOR B=1 TO 10
30 IF KEY$=" " THEN 30
40 A=INT(RAN#*10)
50 PRINT MID$(1,A+1);"↑";MID$(A+3):
60 FOR C=1 TO 30
70 E$=KEY$
80 IF E$=" " THEN 100
90 NEXT C
100 IF E$<"0" THEN 130
110 IF E$>"9" THEN 130
120 IF VAL(E$)=A THEN D=D+1:BEEP 1:GOTO 140
130 BEEP 0
140 PRINT
150 NEXT B
160 PRINT CSR 2;"RIGHT":D:
170 IF D=10 THEN 210
180 FOR B=1 TO 10
190 BEEP 1:BEEP 0
200 NEXT B
210 END
```


Ceci est un programme de jeu. Des chiffres de 0 à 9 sont affichés, un de ceux-ci est affiché sous la forme "↑". Appuyer sur la touche numérique (de 0 à 9) qui correspond à l'emplacement de "↑". A la ligne 50, "↑" est affiché à l'emplacement qui correspond au chiffre de 0 à 9 généré par la fonction RAN# de la ligne 40.

La ligne 30 est utilisée pour attendre que la touche enfoncée soit relâchée. La ligne 60 et les suivantes testent si on a appuyé sur une touche, la répétition cesse alors et un contrôle est effectué pour voir si on a appuyé sur la touche correcte. Les lignes 100 et 110 contrôlent si l'on a appuyé sur une touche numérique. L'entrée d'un caractère autre qu'un chiffre de 0 à 9, provoque un branchement à la ligne 130; le vibreur est activé pour signaler l'erreur.

La ligne 120 est utilisée pour contrôler si la réponse est correcte. Etant donné que KEY\$ lit des caractères, la comparaison est effectuée après conversion du caractère en valeur numérique à l'aide d'une fonction VAL.

Lorsque le contrôle d'enfoncement ou non d'une touche est simplement effectué comme indiqué à la ligne 30, le test peut être fait sans affecter KEY\$ à une variable alphanumérique. Toutefois, quand une réponse correcte est contrôlée par plusieurs tests comme indiqué aux lignes 70 à 120, KEY\$ est affectée à une variable alphanumérique. Stocker ce programme.

Exemple d'opération

Affichage
01↑3456789
01234567↑9
0123↑56789

Opération

Appuyer sur 2.
Appuyer sur 9.
Appuyer sur 4.

Si la vitesse à laquelle l'affichage change est trop rapide, mettre un nombre plus grand que 30 comme valeur finale de l'instruction FOR de la ligne 60.

3-4. EQUIPEMENTS OPTIONNELS PRATIQUES

Bien que cet ordinateur soit déjà pratique, il peut être équipé d'une interface magnétophone à cassette (CI546) et d'une imprimante à caractères (MI547) qui sont des équipements optionnels.

La CI546 permet de stocker des programmes et données de l'ordinateur à une cassette, ou de les charger d'une cassette à l'ordinateur.

La MI547 peut imprimer des programmes, des données et des résultats arithmétiques.

La fonction de chaque appareil est expliquée ci-dessous.

3-4-1 Stockage d'un programme ou de données.

La CI546 est nécessaire pour stocker un programme ou des données sur cassette. Pour brancher la CI546 à l'ordinateur et à un magnétophone à cassette, voir le mode d'emploi fourni avec la CI546.

■ Stockage et chargement de programme

Vu que les programmes sont mémorisés dans l'ordinateur, il peut arriver que le programme suivant ne puisse pas être mémorisé du fait de la capacité mémoire. Si le programme précédent est effacé, il ne pourra plus être utilisé. Dans ce cas, la CI546 est très pratique.

Les commandes servant à stocker un programme sur bande sont "SAVE" et "SAVE ALL". "SAVE" ne peut seulement stocker un programme d'une zone de programme tandis que "SAVE ALL" peut stocker simultanément tous les programmes des 10 zones de programmes.

Commande SAVE



READY P_n

Le programme de cette zone de programme peut être stocké.

Commande SAVE ALL

Tous les programmes des 10 zones de programme peuvent être stockés. Les commandes SAVE et SAVE ALL sont exécutées manuellement.

Exemple)

```
SAVE [EXE]
SAVE "ABCDE" [EXE]
SAVE ALL [EXE]
SAVE ALL "PB" [EXE]
```

Les caractères entre guillemets situés après SAVE et SAVE ALL constituent un nom de fichier qui est placé avec le programme stocké pour permettre son chargement en spécifiant ce nom. Jusqu'à 8 caractères ou chiffres écrits entre guillemets peuvent constituer un nom de fichier.

LOAD ou LOAD ALL sont utilisées pour charger un programme dans l'ordinateur à partir d'une bande. L'emploi correct de ces commandes dépend de l'utilisation de SAVE ou de SAVE ALL lors du stockage des programmes.

Chargement Stockage	LOAD	LOAD "Nom de fichier"	LOAD ALL	LOAD ALL "Nom de fichier"
SAVE	○	×	×	×
SAVE "Nom de fichier"	○	○	×	×
SAVE ALL	×	×	○	×
SAVE ALL "Nom de fichier"	×	×	○	○

* Les éléments marqués d'un "○" peuvent être chargés, pas ceux marqués d'un "×".

* Les noms de fichier doivent être identiques.

Exemple)

```
LOAD [EXE]
LOAD "Nom de fichier" [EXE]
LOAD ALL [EXE]
LOAD ALL "Nom de fichier" [EXE]
```

Lorsque les programmes sont chargés par LOAD ou LOAD ALL, un affichage dépendant de la manière avec laquelle ils ont été stockés apparaît.

Stockage	Affichage
SAVE	PF:
SAVE "Nom de fichier"	PF: Nom de fichier
SAVE ALL	AF:
SAVE ALL "Nom de fichier"	AF: Nom de fichier

Lorsqu'un programme stocké par une commande SAVE est chargé par une commande LOAD, la zone de programme du stockage peut être différente de celle du chargement.

Exemple) Stocker le programme de la zone P0.

↓
Le charger dans la zone P9.

Précautions à prendre:

Il arrive parfois qu'un programme ne puisse pas être stocké ou chargé. Dans ce cas, contrôler les points suivants.

- "ERR9" est affiché pendant le stockage.

[Point de contrôle]

Contrôler si l'ordinateur est correctement branché à la CI546.

- "ERR9" est affiché pendant le chargement.

[Points de contrôle]

Si la bande est détendue suite à un mauvais rangement, la remplacer. Si les têtes du magnétophone sont sales, les nettoyer. Placer le contrôleur de tonalité du magnétophone sur la position moyenne.

- Aucune erreur n'est affichée mais le chargement ne s'effectue pas.

[Points de contrôle]

Si le volume de sortie du magnétophone est bas, l'augmenter et le placer près de MAX.

Les caractéristiques de sortie du magnétophone ne sont pas compatibles avec celles de la CI546.

■ Stockage et chargement de données pour la fonction banque de données

Lorsque les données pour la banque de données sont transférées à une autre unité, elles doivent être stockées sur bande.

"SAVE #" est utilisée pour stocker toutes les données en même temps.

SAVE # "Nom de fichier"

Jusqu'à 8 caractères.

Jusqu'à 8 caractères peuvent être placés entre guillemets pour constituer un nom de fichier de la même manière que lorsqu'un programme est stocké.

Exemple)

SAVE # "MEMO" [EX]

Pour charger dans l'ordinateur les données pour banque de données stockées sur bande, LOAD # est utilisée.

Lorsque de nouvelles données sont chargées les anciennes sont effacées.

LOAD # "Nom de fichier"

Jusqu'à 8 caractères.

Exemple)

LOAD # "MEMO" [SE]

Un des affichages suivants apparaît lorsque les données pour banque de données sont chargées.

Stockage	Affichage
SAVE #	MF:
SAVE # "Nom de fichier"	MF: Nom de fichier

■ Stockage et chargement de données

Un programme utilise toujours des données; il peut être ennuyeux de les entrer à chaque fois à partir du clavier.

Essayer une méthode qui permette de stocker et de recharger des données en mémoire.

"PUT" est utilisée pour stocker des données.

La commande PUT est exécutée avec des noms de variables spécifiés. Un nom de fichier peut aussi être spécifié.

PUT "Nom de fichier" \$, A, Z

Jusqu'à 8 caractères.

Stocke 26 variables de A à Z.

Pour constituer un nom de fichier, jusqu'à 8 caractères sont placés entre guillemets de la même manière que lorsqu'un programme est stocké.

Spécifier d'abord l'exclusive variable alphanumérique (\$) si elle est utilisée. Ensuite deux noms de variables sont spécifiés pour indiquer la première et la dernière variable qui doivent être stockées.

Exemple)

Stocker le contenu de l'exclusive variable alphanumérique (\$) et des 13 variables A à M.

PUT \$, A, M

Stocker le contenu des 36 variables A à Z(10) avec "CDEFG" comme nom de fichier.

PUT "CDEFG" A, Z(10)

Etant donné que les noms de variables spécifient d'où à où le stockage doit être effectué, ils sont placés dans l'ordre alphabétique A, Z. Une spécification telle que "Z, A" ne peut pas être exécutée.

Quand les variables sont des variables alphanumériques, "A, Z" peut être utilisée au lieu de "\$, Z\$".

"GET" est utilisée pour charger des données dans l'ordinateur à partir d'une bande. La commande GET est exécutée avec des noms de variables spécifiés. Un nom de fichier peut être spécifié.

GET "Nom de fichier" \$, M, W

Jusqu'à 8 caractères.

Charge dans les variables M à W.

Exemple)

Charger des données dans l'exclusive variable alphanumérique (\$) et dans les trois variables X à Z.

```
GET $,X,Z
```

Charger des données du fichier "OP" dans les variables G(0) à G(59).

```
GET "OP" G(0), G(59)
```

Les affichages suivants apparaissent quand des données sont chargées par une commande GET.

Stockage	Affichage
PUT \$, A, Z	DF:
PUT "Nom de fichier" G, P	DF: Nom de fichier

3-4-2 Conservation d'un relevé

Il peut être pratique d'imprimer un programme après l'avoir préparé pour effectuer la mise au point ou modifier son contenu. De même, il peut être pratique de conserver un résultat arithmétique une fois qu'il est imprimé. Effectuer l'impression à l'aide de l'imprimante MI547.

Se mettre en mode impression ("PRT" est affiché) pour effectuer cette dernière.

On spécifie le mode impression en appuyant sur  et on l'annule en appuyant sur .

Pour imprimer le contenu d'un programme, exécuter la commande LIST après avoir appuyé sur .

Exemple)

Entrer le programme suivant.

```
10 INPUT A
20 PRINT A*A
30 GOTO 10
```

Ensuite, effectuer l'impression.

```
 LIST 
```

Exemple d'impression

```
LIST
10 INPUT A
20 PRINT A*A
30 GOTO 10
```

Lorsque le contenu des 10 zones de programme P0 à P9 doit être imprimé, exécuter.

```
LIST ALL 
```

Annuler le mode impression en appuyant sur  après avoir effectué l'impression.

Pour imprimer un résultat arithmétique, spécifier le mode impression en appuyant sur  ou en écrivant "MODE 7" dans le programme. Lorsque l'on a appuyé sur , toutes les manipulations de touches sont imprimées. Par conséquent, si seulement une partie doit être imprimée, il est plus pratique d'écrire "MODE 7" dans le programme.

* Dans ce cas, l'entrée doit être effectuée en appuyant sur les touches  au lieu de la touche .

Exemple)

Seul le résultat arithmétique est imprimé.

```
10 INPUT A
20 MODE 7
30 PRINT A*A
40 MODE 8
50 GOTO 10
```

Dans ce programme, le mode impression est spécifié après l'entrée de données et est annulé après l'impression pour retourner à l'attente d'entrée de données.

L'affichage par l'instruction PRINT n'est pas arrêté pendant l'impression. Le programme passe à la commande suivante après l'impression.

Annuler le mode impression en faisant MODE 8.

3-5. UTILISATION D'UN PROGRAMME DE L'OP544

Les programmes préparés pour les OP544 et OP644 peuvent être utilisés par cet ordinateur. Cet ordinateur possède plus de commandes que les OP544 et OP644; son utilisation est plus pratique.

Le langage BASIC utilisé par cet ordinateur est presque le même que celui utilisé par les OP544 et OP644.

■ Différences

• Commandes supplémentaires

PASS (Protection de programmes)
BEEP (Tonalité)
READ (Lecture de données à partir d'une instruction DATA)
DATA (Ecrit des données)
RESTORE (Spécifie les données à lire)
ON-GOTO (Spécification indirecte d'une instruction GOTO)
ON-GOSUB (Spécification indirecte d'une instruction GOSUB)
REM (Instruction de commentaire)

• Fonctions supplémentaires

DEG (Conversion décimal → sexagésimal)
DMS\$ (Conversion sexagésimal → décimal)
STR\$ (Convertit une valeur numérique en une chaîne de caractères)

• Commandes modifiées

Cet ordinateur	OP544/OP644
NEW (NEW ALL) CLEAR IF-THEN SAVE ALL LOAD ALL VERIFY DEFM (Peut être écrite dans un programme)	CLEAR (CLEAR ALL) VAC IF-; SAVE A LOAD A VER DEFM (Ne peut être exécutée que manuellement.)

• Fonctions modifiées

Cet ordinateur	OP544/OP644
KEY\$ MID\$	KEY MID

Malgré ces différents points, un programme préparé pour les OP544/OP644 peut être utilisé fondamentalement par cet ordinateur.

Néanmoins, il est préférable de réécrire les programmes pour cet ordinateur afin qu'ils puissent être utilisés plus facilement ou revus plus tard sans difficultés.

Exemple 1)

Programme de l'OP544

```

10 VAC
20 FOR A=1 TO 20
30 INPUT Z(A)
40 IF Z(A)>80;B=B+1;GOTO 90
50 IF Z(A)<60;C=C+1;GOTO 90
60 IF Z(A)>40;D=D+1;GOTO 90
70 IF Z(A)>20;E=E+1;GOTO 90
80 F=F+1
90 NEXT A

```

Cet exemple est une partie de programme permettant d'entrer des données et de les répartir en fonction de leur taille. Bien que ce programme puisse être utilisé tel quel, effectuer les corrections suivantes.
Changer "VAC" à la ligne 10 en "CLEAR".

```

10 CLEAR

```

Changer les ";" aux lignes 40 à 70 en "THEN".

```

40 IF Z(A)>80 THEN B=B+1;GOTO 90

```

Etant donné qu'une extension de mémoire est nécessaire dans ce programme, écrire au début du programme la commande, DEFM qui est exécutée manuellement sur l'OP544/OP644.

```
5 DEFM 20
```

Exemple 2)

Programme de l'OP544

```
10 INPUT "I=1/O=2/P=3",N
20 IF N<1 THEN 10
30 IF N>3 THEN 10
40 GOTO N*100
```

Ce programme est utilisé pour déterminer les emplacements de branchement en fonction du travail à effectuer. Pour l'adapter à cet ordinateur, le modifier de la manière suivante en utilisant une instruction ON-GOTO.

```
10 INPUT "I=1/O=2/P=3",N
20 ON N GOTO 100,200,300
30 GOTO 10
```

L'utilisation d'une instruction ON-GOTO simplifie le programme; le test de la donnée N est supprimé.

Les programmes et les données stockés sur bande par les OP544 ou OP644 peuvent être chargés tels quels dans cet ordinateur. Toutefois, l'opération inverse n'est pas toujours possible. Par conséquent, des précautions doivent être prises. Les relations entre les ordinateurs sont indiquées ci-dessous.

Cet ordinateur → OP544, OP644

LOAD \ SAVE	PF	AF	MF	Avec mot de passe		
				PF	AF	MF
LOAD	○	△	△	△	△	△
LOAD ALL	△	○	△	△	△	△

○ : peut être chargé.

△ : Ne peut pas être chargé.

[PRECAUTIONS A PRENDRE]

- Quand un programme préparé par cet ordinateur est transféré sur l'OP544 ou l'OP644, les commandes READ #, WRITE # et RESTORE # ne doivent pas être utilisées. KEY\$ et MID\$ doivent être changées en KEY et MID pour les OP544 et OP644.
- Quand un programme préparé par l'OP544 ou l'OP644 est exécuté avec cet ordinateur, il peut arriver que l'exécution ne puisse pas être effectuée correctement comme indiqué ci-dessous.
 - * Si une expression numérique est utilisée pour un emplacement de branchement dans une instruction IF-THEN, une erreur se produit. Dans ce cas, effectuer la correction en utilisant une instruction IF-THEN-GOTO emplacement de branchement.

CHAPITRE 4

REFERENCE DE COMMANDES

* Ci-après suit la description des symboles et termes employés dans la syntaxe.

- { $\times \times \times \times$ } Un des éléments entre { } doit être sélectionné.
- [$\circ \circ \circ \circ$] L'élément entre [] peut être omis.
- $\circ \circ \circ \circ^*$ L'élément marqué d'un * en haut à droite peut être utilisé à plusieurs reprises.
- Expression numérique
Valeur numérique telle que 10, 2+3, A, S*Q, expression de calcul et variable numérique.
- Expression alphanumérique
Constante de caractères telle que "ABC", X\$, N\$+M\$, variable alphanumérique et expression de traitement de caractères.
- Paramètre Un élément qui accompagne une commande.
- (P) Peut être uniquement exécuté dans un programme.
- (M) Peut être uniquement exécuté manuellement.
- (A) Peut être exécuté et manuellement et dans un programme.
- (F) Fonction qui peut être exécutée et manuellement et dans un programme.

< Exemple > DATA [donnée] [, [donnée]]*

Vu que toutes les données sont entre crochets [], il est aussi possible de n'écrire que "DATA". Vu que ", [donnée]" est spécifié avec [], cet élément peut être écrit à plusieurs reprises. L'instruction ci-dessus peut par conséquent être écrite "DATA donnée, donnée, ...". Si on omet le premier élément [donnée], cette instruction peut aussi être écrite "DATA, donnée, donnée, ...".

GOTO { Numéro de ligne.
numéro de zone de programme. }

Il y a deux différentes façons d'écrire cette instruction comme indiqué ci-dessous.

- (1) GOTO Numéro de ligne.
- (2) GOTO # numéro de zone de programme.

NEW [ALL]

Fonction

Effacement de programme. Efface les programmes et les variables.

Paramètre

Lorsque ALL est spécifiée, tous les programmes des zones P0 à P9 et les variables sont effacés.

Explication

- (1) Si ALL n'est pas spécifiée, le programme de la zone de programme spécifiée à ce moment est effacé. Les variables ne sont pas effacées.
- (2) Si ALL est spécifiée, les programmes et les variables de toutes les zones de programme sont effacés. La commande DEFM est annulée et le nombre de mémoires initialisé.
- (3) Ne peut être exécutée quand un mot de passe est spécifié.
- (4) Ne peut pas être utilisée dans un programme.
- (5) Ne peut être exécutée qu'en mode WRT.

* NEW ALL peut être abrégée sous la forme NEW A.

Exemple

0001 NEW 000

RUN

[Ligne de début d'exécution]
numéro de ligne

Fonction

Exécution d'un programme.

Paramètre

Ligne de début d'exécution: numéro de ligne.

Explication

- (1) Exécute un programme à partir d'une ligne spécifiée (si le numéro de ligne est omis, l'exécution commence au début du programme)
- (2) Quand le numéro de ligne spécifié n'existe pas, l'exécution démarre à la ligne de numéro supérieur le plus proche.
- (3) Les variables ne sont pas effacées.

Exemple

```
10 PRINT "LINE 10"  
20 PRINT "LINE 20"  
30 END
```

RUN **EXE**
RUN 20 **EXE**

LINE 10
LINE 20

LIST

{Numéro de ligne}
ALL

Fonction

Affiche le contenu d'un programme.

Paramètre

Numéro de ligne: numéro de la première ligne à afficher.

ALL: Affiche séquentiellement le contenu de tous les programmes des zones P0 à P9.

Explication

I. Mode RUN.

- (1) Affiche séquentiellement le contenu d'un programme à partir d'un numéro de ligne s'il est spécifié ou à partir du début s'il est omis.
- (2) Etant donné que le contenu d'un programme est automatiquement affiché séquentiellement, appuyer sur la touche **STOP** pour arrêter l'affichage. Appuyer sur la touche **EXE** pour afficher les lignes suivantes.
- (3) En mode PRINT (lorsque "PRT" est affiché) avec une imprimante branchée, l'affichage n'est pas arrêté mais exécuté séquentiellement à vitesse élevée.

II. Mode WRT.

- (1) Affiche le contenu d'un programme à partir d'un numéro de ligne s'il est spécifié ou à partir du début s'il est omis.
 - (2) Etant donné qu'en mode WRT, chaque ligne est affichée pour permettre sa mise en forme, si celle-ci n'est pas requise, appuyer sur la touche **EXE** pour passer à la ligne suivante. Egalement, si l'on a appuyé sur la touche **SHIFT** avant d'appuyer sur la touche **EXE**, la ligne précédente est affichée.
- Lorsque ALL est spécifiée, le contenu de tous les programmes des zones P0 à P9 est affiché séquentiellement. Dans ce cas, les programmes défilent séquentiellement, même en mode WRT, on ne peut donc pas procéder à la mise en forme.
 - Cette commande ne peut pas être utilisée lorsqu'un mot de passe est spécifié.

* LIST ALL peut être abrégée sous la forme LIST A.

Exemple

LIST **EXE**
LIST 30 **EXE**

PASS

"Mot de passe"
Chaîne de caractères



Fonction

Spécifie ou annule un mot de passe.

Paramètre

Mot de passe: chaîne de 1 à 8 caractères.

Explication

- (1) Si un mot de passe n'est pas spécifié, l'exécution de cette commande spécifie un mot de passe pour toutes les zones de programme (P0 à P9).
- (2) Si cette commande est exécutée alors qu'un mot de passe est spécifié, ce dernier n'est annulé que si le nouveau mot de passe lui correspond. Si les deux mots de passe ne sont pas identiques, une erreur de protection est provoquée (ERR8).
- (3) Un mot de passe consiste en une chaîne de caractères qui peut comprendre des espaces, des caractères alphabétiques, des chiffres, des symboles spéciaux, etc. Néanmoins, (") ne peut pas être utilisé.
- (4) Quand un mot de passe est spécifié, les commandes telles que LIST, LIST ALL, LIST #, NEW, NEW ALL et NEW # ne peuvent pas être utilisées. Également, aucune écriture (mode WRT) ne peut être faite. Si elle était tentée, une erreur (ERR8) se produirait.
- (5) Ne peut pas être utilisée dans un programme.
- (6) Un mot de passe peut être conservé quand l'alimentation est coupée.
- (7) Si un programme est stocké sur cassette par une commande SAVE ou SAVE ALL quand un mot de passe est utilisé, ce dernier est également stocké. Quand un programme comprenant un mot de passe est chargé à partir d'une cassette par une commande LOAD ou LOAD ALL, le mot de passe est également chargé. De plus, quand le mot de passe actuellement spécifié dans l'unité centrale est différent de celui du programme devant être chargé à partir d'une cassette, ce dernier ne peut être chargé (ERR8).

Précaution à prendre

Si un mot de passe a été oublié après qu'il ait été spécifié, appuyer sur le bouton de remise à zéro générale (R. A. Z.) situé au dos de l'ordinateur pour effacer tous les programmes et la mémoire.

Exemple

PASS "OLYMPIA"

SAVE [ALL]

["Nom de fichier"]
Chaîne de caractères



Fonction

Stocke un programme sur une cassette.

Paramètre

ALL: stocke les programmes de toutes les zones de programme.
Nom de fichier: Une chaîne de 1 à 8 caractères. Peut être omis.

Explication

- (1) Lorsque ALL est omis, le contenu de la zone de programme spécifiée actuellement est stocké.
- (2) Lorsque ALL est utilisé, le contenu de toutes les zones de programme P0 à P9 est stocké.
- (3) Lorsqu'un mot de passe est utilisé, le stockage est exécuté avec ce mot de passe. De ce fait, lorsqu'un programme est chargé par une commande LOAD, il comporte le même mot de passe que lorsqu'il a été stocké.

* SAVE ALL peut être abrégée sous la forme SAVE A.

Exemple

```
SAVE   
SAVE "OLYMPIA"   
SAVE ALL "OP" 
```


LOAD [ALL]

["Nom de fichier"]
Chaîne de caractères



Fonction

Charge un programme à partir d'une cassette.

Paramètre

ALL: Charge les programmes dans toutes les zones de programme.
Nom de fichier: Une chaîne de 1 à 8 caractères.
Peut être omis.

Explication

- (1) Lorsque ALL est omis, un programme stocké par une commande SAVE est chargé dans la zone de programme actuellement spécifiée.
 - (2) Lorsque ALL est utilisé, les programmes stockés par une commande SAVE ALL sont chargés dans les zones de programme P0 à P9.
 - (3) Lorsqu'un programme stocké avec un mot de passe est chargé à partir d'une cassette, ce mot de passe est également chargé.
- * LOAD ALL peut être abrégée sous la forme LOAD A.

Relations entre SAVE et LOAD

	LOAD	LOAD "Nom de fichier"	LOAD ALL	LOAD ALL "Nom de fichier"
SAVE	O	x	x	x
SAVE "Nom de fichier"	O	O	x	x
SAVE ALL	x	x	O	x
SAVE ALL "Nom de fichier"	x	x	O	O

* Les noms de fichier sont identiques. O---peut être chargé.
x---ne peut pas être chargé.

VERIFY

["Nom de fichier"]
Chaîne de caractères



Fonction

Contrôle l'état d'un programme et des données stockés sur une cassette.

Paramètre

Nom de fichier: Une chaîne de 1 à 8 caractères. Peut être omis.

Explication

- (1) Lorsqu'un nom de fichier est spécifié, le fichier portant ce nom est contrôlé.
- (2) Lorsque le nom de fichier est omis, contrôle le premier fichier qui apparaît sur la cassette.
- (3) Le système de contrôle de parité est utilisé pour contrôler le format de stockage.

Exemple

VERIFY **END**
VERIFY *PROG1 * **END**

CLEAR



Fonction

Efface toutes les variables.

Explication

- (1) Efface toutes les variables, les numériques sont mises à zéro et les alphanumériques vidées.
 - (2) Cette commande peut être utilisée dans un programme et manuellement.
 - (3) Etant donné que les variables de contrôle sont également effacées dans une boucle FOR-NEXT (voir page 134), une erreur survient pendant l'exécution de l'instruction NEXT.
- * La commande CLEAR fonctionne de la même façon que la commande VAC.

END

Fonction

Termine l'exécution d'un programme.

Explication

Etant donné que l'exécution du programme est terminée, s'il en existe un autre, il ne sera pas exécuté.

STOP

Fonction

Suspend temporairement l'exécution d'un programme.

Explication

- (1) Suspend temporairement l'exécution d'un programme et affiche "STOP" puis une attente d'entrée se produit.
- (2) Après la suspension, on reprend l'exécution en appuyant sur la touche **EXE**.
- (3) Si l'on appuie sur la touche **STOP** pendant que l'exécution est arrêtée par une commande STOP, le numéro de la zone de programme et le numéro de ligne sont affichés.
- (4) Pendant la suspension de l'exécution d'un programme par une commande STOP, des calculs manuels peuvent être exécutés.

[LET] { Variable numérique = expression numérique
Variable alphanumérique = expression alphanumérique } ^(P)

Fonction

Affecte la valeur de l'expression de droite à la variable de gauche.

Explication

- (1) Une expression numérique correspond à une variable numérique et une expression alphanumérique à une variable alphanumérique.
- (2) LET peut être omise.

Exemple

```
10 LET X=12
20 LET Y=X+2+2*X-1
30 PRINT Y
40 A$="ABCDE"
50 B$="OP & CI"
60 PRINT A$;B$
70 END
```

REM

Commentaire
Chaîne de caractères

Fonction

Instruction qui formule un commentaire.

Explication

- (1) Dans un programme, le contenu qui se trouve après REM est traité comme une instruction de commentaire et n'est donc pas exécuté.
- (2) Quand une commande qui doit être exécutée est écrite sur la même ligne, placer deux-points (:) avant la commande REM.

Exemple

```
10 INPUT "R",R
20 S=PI*R^2:REM AREA
30 PRINT S
40 END
```

INPUT ["Instruction de message"; Nom de variable] , ["Instruction de message"; Nom de variable] ^(P)

Fonction

Entre des données dans une variable à partir du clavier.

Paramètre

Message: Chaîne de caractères.

Nom de variable: Nom de variable numérique ou alphanumérique.

Explication

- (1) Entre des données dans une variable spécifiée à partir du clavier.
- (2) Lorsqu'un message existe, il est affiché et suivi d'un "?".
- (3) Quand il n'y a pas de message seul le "?" est affiché.
- (4) Appuyer sur la touche **OK** après avoir entré les données.
- (5) Lorsque des données alphanumériques sont entrées dans une variable numérique, une erreur (ERR2) se produit et les données sont de nouveau demandées par l'affichage du "?" après que l'on ait appuyé sur la touche **OK**. Quand une expression numérique est entrée, le résultat de cette expression est affecté. Quand un caractère alphabétique est entré, la valeur de la variable correspondant à ce caractère est affectée.
- (6) Quand on appuie sur la touche **OK** pendant l'attente d'entrée de données, cela provoque une entrée vide. Une erreur (ERR2) se produit si la variable est une variable numérique.

Exemple

```
10 INPUT A
20 INPUT "B$=", B$
30 INPUT "C$=", C$, "D$=", D$
```

KEY\$

(P)

Fonction

Fonction qui permet d'entrer un caractère à partir du clavier.

Explication

- (1) L'entrée d'un caractère seulement est permise à partir du clavier.
- (2) Des chiffres, des caractères alphabétiques et des symboles peuvent être entrés.
- (3) Etant donné qu'un "?" n'est pas affiché et qu'une attente d'entrée ne se produit pas, KEY\$ est habituellement combinée avec une instruction IF.

* KEY\$ peut être abrégée sous la forme KEY.

Exemple

```
10 PRINT "INPUT<6>";
20 A$=""
30 K$=KEY$
40 IF K$="" THEN 30
50 A$=A$+K$
60 IF LEN(A$)<6 THEN 30
70 PRINT A$
80 END
```

* Six caractères sont acceptés à partir du clavier.

PRINT [Elément de sortie] [{ : } [Elément de sortie]] * (P)

Fonction

Affiche un élément de sortie.

Paramètre

Elément de sortie: Fonction de contrôle de sortie (CSR), expression numérique, expression alphanumérique.

Explication

- (1) Affiche un élément de sortie. Lorsqu'une fonction de contrôle de sortie est jointe, l'élément est affiché à l'endroit déterminé par celle-ci.
- (2) Les valeurs des expressions numériques et alphanumériques sont affichées.
- (3) Lorsqu'un élément de sortie est une expression numérique, un signe (+, -) est placé avant la valeur. Toutefois, le signe + est affiché en blanc.

• Affichage de caractères

élément de sortie

• Affichage de valeur numérique

signe élément de sortie

- (4) Lorsqu'un élément de sortie est une expression numérique et lorsque la mantisse comporte plus de 10 chiffres, le 11^e et les suivants sont supprimés. Egalement, lorsqu'un signe et un exposant existent, un signe d'exposant (E) et un exposant de deux chiffres sont affichés.
- (5) On peut utiliser ",", " " et ";" comme ponctuation pour séparer les éléments de sortie. Lorsqu'une virgule est utilisée, un arrêt se produit après l'affichage du premier élément de sortie (STOP est affiché), puis le suivant est affiché en appuyant sur la touche **OK**. Lorsqu'un point-virgule est utilisé, l'élément de sortie suivant est affiché immédiatement après le premier.
- (6) Quand aucun élément n'est spécifié (seul PRINT est écrit), l'affichage est effacé et n'est pas arrêté.
- (7) L'affichage n'est pas arrêté pendant l'impression en mode PRINT (**MODE** **PRINT**).
- (8) Les valeurs numériques peuvent être mises en forme par une instruction SET.

Exemple

```
10 PRINT 1/3
20 PRINT "A="; A
30 PRINT "SIN 30", SIN 30
40 PRINT "END";
50 PRINT
60 END
```


Fonction

Affiche un élément de sortie à partir d'un emplacement spécifié.

Paramètre

Spécification de l'emplacement de sortie: Expression numérique. Les chiffres après la virgule sont supprimés.

$$0 \leq \text{spécification} < 12$$

Explication

- (1) Utilisé dans une instruction PRINT pour spécifier l'emplacement d'un élément de sortie.
- (2) 0 correspond à l'emplacement de sortie du bord gauche.

**Exemple**

```
10 FOR I=0 TO 11
20 PRINT CSR1; "A"; CSR 11-I; "B"
30 NEXT I
40 END
```

- Les caractères A et B sont décalés respectivement à partir de la gauche et de la droite à chaque appui sur la touche **OK**.

Fonction

Effectue un branchement inconditionnel à un emplacement spécifié.

Paramètre

Numéro de ligne de branchement: Numéro de ligne de 1 à 9999.
 Numéro de zone de programme: Un caractère de 0 à 9.

Explication

- (1) Effectue un branchement à un emplacement spécifié.
- (2) Lorsque l'emplacement de branchement est un numéro de ligne, effectue un branchement à la ligne spécifiée de la zone de programme dans laquelle on se trouve et exécute le programme. Lorsqu le numéro de ligne de branchement n'existe pas, une erreur (ERR4) se produit.
- (3) Lorsque l'emplacement de branchement est un numéro de zone de programme, effectue un branchement à la zone de programme spécifiée et exécute le programme à partir du début.

* Une expression numérique peut être utilisée comme numéro de ligne de branchement et comme numéro de zone de programme.

Exemple

```
10 PRINT "START";
20 GOTO 100
30 PRINT "LINE 30"
40 END
100 PRINT "LINE 100"
110 GOTO 30
```


ON ^(P) Condition de branchement
Expression numérique **GOTO** [, [Emplacement de branchement]]
* Emplacement de branchement { Numéro de ligne de branchement
N : numéro de zone de programme

Fonction

Effectue un branchement à un emplacement en accord avec la condition.

Paramètre

Condition de branchement: Expression numérique.

Les chiffres après la virgule sont supprimés.

Numéro de ligne de branchement: Numéro de ligne de 1 à 9999.

Numéro de zone de programme: Un caractère de 0 à 9.

Explication

- (1) Effectue un branchement en accord avec la partie entière de la valeur de l'expression de condition de branchement. Les emplacements de branchement sont alloués séquentiellement pour les valeurs correspondantes de l'expression 1, 2,

ON A GOTO $\frac{100}{A=1}, \frac{200}{A=2}, \frac{300}{A=3}, \dots\dots\dots$

- (2) Lorsque la valeur de l'expression est inférieure à 1, ou lorsqu'un emplacement de branchement n'existe pas, l'instruction suivante est exécutée. Il n'y a donc pas de branchement.
- (3) On peut écrire autant d'emplacements de branchement que peut en contenir une ligne.

Exemple

```
10 INPUT A
20 ON A GOTO 100,200,300
30 PRINT "OTHER"
40 GOTO 10
100 PRINT "LINE 100":GOTO 10
200 PRINT "LINE 200":GOTO 10
300 PRINT "LINE 300":GOTO 10
```

- * Lorsqu'on entre les chiffres de 1 à 3, cela provoque les branchements respectifs de 100 à 300, dans le cas contraire "OTHER" est affiché.

IF ^(P) Condition de branchement
Expression de comparaison **THEN** { Instruction [: instruction]*
Emplacement de branchement }
* Emplacement de branchement { Numéro de ligne de branchement
N : numéro de zone de programme

Fonction

Lorsque la condition de branchement est réalisée, les instructions qui suivent THEN sont exécutées. Egalement, lorsque l'instruction qui suit THEN est un emplacement de branchement, le branchement est effectué.

Paramètre

Condition de branchement: Expression de comparaison.

Numéro de ligne de branchement: Numéro de ligne de 1 à 9999.

Numéro de zone de programme: Un caractère de 0 à 9.

Explication

- (1) Lorsque la condition de branchement est réalisée, les instructions qui suivent THEN sont exécutées ou le branchement effectué.
- (2) Lorsque la condition de branchement n'est pas réalisée, la ligne suivante est exécutée.
- (3) La condition de branchement est déterminée à l'aide d'une expression de comparaison (=, ≠, <, >, ≤, ≥).
= L'élément de gauche est égal à celui de droite.
≠ L'élément de gauche est différent de celui de droite.
< L'élément de gauche est inférieur à celui de droite.
> L'élément de gauche est supérieur à celui de droite.
≤ L'élément de gauche est inférieur ou égal à celui de droite.
≥ L'élément de gauche est supérieur ou égal à celui de droite.
- (4) Lorsque deux conditions de branchement ou plus existent, plusieurs instructions IF-THEN peuvent être écrites séquentiellement.

IF — THEN IF — THEN

- * Lorsqu'une instruction est présente après THEN, on peut utiliser ";" au lieu de THEN.

Exemple

```
10 N=6
20 PRINT CSR N: "1 ";
30 K$=KEY$
40 IF K$="4" THEN N=N-1:IF N<0 THEN N=0
50 IF K$="6" THEN N=N+1:IF N>11 THEN N=11
60 PRINT
70 GOTO 20
```

- * "1" est décalé vers la gauche quand on appuie sur la touche  et vers la droite quand on appuie sur la touche .

FOR $\frac{\text{Nom de variable de contrôle} = \text{Valeur initiale}}{\text{Expression numérique}}$ **TO** $\frac{\text{Valeur finale}}{\text{Expression numérique}}$ **[STEP** $\frac{\text{Incrément}}{\text{Expression numérique}}$ **]** **NEXT** $\frac{\text{Nom de variable de contrôle}}{\text{Expression numérique}}$

Fonction

Répète les instructions situées entre les instructions FOR et NEXT un nombre de fois spécifié par une variable de contrôle dont la valeur passe d'une valeur initiale à une valeur finale en variant à chaque itération de la valeur de l'incrément.

Paramètre

Nom de variable de contrôle: Simple nom de variable.
Une variable de tableau ne peut pas être utilisée.

Valeur initiale: Expression numérique.

Valeur finale: Expression numérique.

Incrément: Expression numérique.

La valeur 1 est prise par défaut.

Explication

- (1) Répète les instructions situées entre les instructions FOR et NEXT un nombre de fois spécifié par une variable de contrôle dont la valeur passe d'une valeur initiale à une valeur finale en variant à chaque itération de la valeur de l'incrément. Lorsque la valeur de la variable de contrôle dépasse la valeur finale, la répétition est terminée.
- (2) Lorsque la valeur initiale est supérieure à la valeur finale, l'exécution des instructions situées entre FOR et NEXT n'a lieu qu'une seule fois.
- (3) Un nombre négatif peut être utilisé comme incrément.
- (4) Une instruction NEXT doit toujours correspondre à une instruction FOR. Egalement, l'instruction NEXT qui correspond à une instruction FOR doit être située après cette dernière.
- (5) Des boucles FOR-NEXT peuvent avoir la structure d'emboîtement suivante.

```
10 FOR I=1 TO 10
20 FOR J=11 TO 20
30 PRINT I;" ";J
40 NEXT J
50 NEXT I
60 END
```

- (6) L'emboîtement des boucles FOR-NEXT est possible jusqu'à 4 niveaux.

- (7) Lorsque l'exécution d'une boucle FOR-NEXT est terminée, la valeur de la variable de contrôle dépasse la valeur finale de celle de l'incrément.
- (8) Bien qu'un branchement à partir d'une boucle FOR-NEXT puisse être effectué, un branchement provoqué par une instruction IF ou une instruction GOTO renvoyant l'exécution du programme à l'intérieur d'une boucle provoque une erreur.

GOSUB $\left\{ \begin{array}{l} \text{Numéro de ligne de branchement} \\ \text{Numéro de ligne} \\ \text{\# numéro de zone de programme} \end{array} \right\}$

Un caractère de 0 à 9

Fonction

Effectue un branchement à un sous-programme.

Paramètre

Numéro de ligne de branchement: Numéro de ligne de 1 à 9999.

Numéro de zone de programme: Un caractère de 0 à 9.

Explication

- (1) Effectue un branchement à un sous-programme. Un retour du sous-programme est effectué par l'exécution de l'instruction RETURN.
 - (2) L'emboîtement de sous-programmes est possible jusqu'à 8 niveaux.
 - (3) L'instruction RETURN provoque un retour à l'instruction qui suit le GOSUB ayant appelé le sous-programme.
 - (4) Prendre garde que le retour soit effectué par une instruction RETURN et non en sortant du sous-programme par une instruction IF ou une instruction GOTO.
 - (5) Lorsque le numéro de ligne de branchement n'existe pas, une erreur (ERR4) se produit.
- * Une expression numérique peut également être utilisée comme numéro de ligne de branchement ou comme numéro de zone de programme.

Exemple

```

10 PRINT "MAIN 10"
20 GOSUB 100
30 PRINT "MAIN 30"
40 END
100 PRINT "SUB 100"
110 GOSUB 200
120 RETURN
200 PRINT "SUB 200"
210 RETURN

```

RETURN

Fonction

Provoque un retour du sous-programme au programme principal.

Explication

Retourne à l'instruction située juste après l'instruction qui appelle le sous-programme.

ON Condition de branchement
Expression numérique **GOSUB** [Emplacement de branchement]
[, [Emplacement de branchement]]

★ Emplacement de branchement { Numéro de ligne de branchement
numéro de zone de programme

Fonction

Branche à un sous-programme en accord avec une condition de branchement.

Paramètre

Condition de branchement: Expression numérique.
Les décimales ne sont pas prises en compte.
Numéro de ligne de branchement: Numéro de ligne de 1 à 9999.
Numéro de zone de programme: Un caractère de 0 à 9.

Explication

- (1) Effectue le branchement à un sous-programme en fonction de la valeur entière d'une expression de condition de branchement. Les emplacements de branchement sont séquentiellement alloués à partir du début pour les valeurs de l'expression de 1, 2, 3

ON B GOSUB $\frac{1000}{B=1}$, $\frac{2000}{B=2}$, $\frac{3000}{B=3}$

- (2) Lorsque la valeur de l'expression est inférieure à 1 ou lorsqu'un emplacement de branchement approprié n'existe pas, l'instruction suivante est exécutée, il n'y a donc pas de branchement.
- (3) On peut écrire autant d'emplacements de branchement que peut en contenir une ligne.

Exemple

```

10 INPUT A
20 ON A GOSUB 100,200,300
30 GOTO 10
100 PRINT "SUB 100":RETURN
200 PRINT "SUB 200":RETURN
300 PRINT "SUB 300":RETURN

```

- Lorsqu'une valeur de 1 à 3 est entrée, un branchement au sous-programme correspondant est effectué.

DATA

[Données] **[, [Données]]***

Constante Constante

Fonction

Stocke des données.

Paramètre

Données: Constante numérique ou alphanumérique.

Explication

- (1) Est utilisée pour écrire des données qui seront lues par une instruction READ.
- (2) Plusieurs données séparées par des virgules peuvent être écrites.
- (3) Si seule une instruction DATA est exécutée, sans une instruction READ, cela n'a aucune fonction utile.
- (4) Lorsqu'une constante alphanumérique comprend une virgule, elle doit être placée entre guillemets.

DATA $\frac{ABC}{1^{ère}}$, $\frac{DEF}{2^{e}}$, $\frac{"GHI,JKL"}{3^{e}}$,.....

- (5) Lorsqu'une donnée est omise, une chaîne de caractère d'une longueur de 0 est prise par défaut.

```

DATA A, ,B → DATA A,"",B
DATA , → DATA "",
DATA → DATA ""

```


READ (P)

Nom de variable [, [Nom de variable]]*

Fonction

Lit le contenu d'une instruction DATA.

Paramètre

Nom de variable: Variable numérique ou alphanumérique. Une variable de tableau peut être utilisée.

Explication

- (1) Affecte les données de l'instruction DATA actuellement spécifiée séquentiellement dans des variables spécifiées.
- (2) On ne peut affecter que des données de type numérique dans une variable numérique.
- (3) Les données d'une instruction DATA sont lues séquentiellement à partir du début et les instructions DATA séquentiellement à partir du plus petit numéro de ligne.
- (4) Après que les données nécessaires aient été lues par une instruction READ, les données suivantes seront lues par l'instruction READ suivante.
- (5) La première donnée d'une zone de programme dans laquelle existe une instruction READ est lue lors de la première exécution de cette instruction après quoi, les données de l'actuelle zone de programme sont lues séquentiellement.
- (6) La spécification des données à lire peut être changée par une instruction RESTORE.
- (7) Lorsque le nombre de données est inférieur au nombre de variables de l'instruction READ, une erreur (ERR4) se produit.
- (8) Lorsqu'un espace existe au début d'une donnée, il est ignoré.

Exemple

```
10 DATA 1,2,3
20 READ A,B
30 PRINT A;B
40 DATA 4,5
50 READ C,D,E
60 PRINT C;D;E
70 END
```

- Lit séquentiellement les données d'une instruction DATA et les affiche.

RESTORE (P)

[Numéro de ligne]
Expression numérique

Fonction

Spécifie l'emplacement des données à lire par une instruction READ.

Paramètre

Numéro de ligne: Expression numérique. Les décimales ne sont pas prises en compte.

$$1 \leq \text{Numéro de ligne} \leq 9999$$

Explication

- (1) Spécifie l'instruction DATA dans laquelle existent les données à lire par une instruction READ.
- (2) Lorsqu'un numéro de ligne est omis, la spécification des données est annulée. Après cela, les premières données de la zone de programme dans laquelle une instruction READ existe, sont spécifiées et lues par la première instruction READ exécutée.
- (3) Lorsqu'un numéro de ligne de la zone de programme est spécifié par une instruction RESTORE, les données de l'instruction DATA portant ce numéro de ligne seront lues séquentiellement par l'instruction READ.
- (4) Lorsqu'un numéro de ligne spécifié n'existe pas ou lorsqu'une instruction DATA n'existe pas au numéro de ligne spécifié ou aux lignes suivantes, une erreur (ERR4) se produit.

Exemple

```
10 DATA 1,2,3
20 DATA 4,5
30 READ A,B,C,D,E
40 RESTORE 10
50 READ F,G
60 RESTORE 20
70 READ H,I
80 PRINT A;B;C;D;E;F;G;H;I
90 END
```


PUT

["Nom de fichier"] variable 1 [, variable 2]*
Chaîne de caractères

Fonction

Stocke des données sur cassette.

Paramètre

Nom de fichier: Chaîne de 1 à 8 caractères. Peut être omis.
Variable 1, Variable 2: Spécification de la variable à stocker.

Explication

- (1) Stocke le contenu de variables sur cassette.
- (2) Les spécifications de variable sont écrites de la manière suivante.

PUT A Contenu de la variable A.

PUT A,Z Contenu des variables de A à Z.

PUT A,A (50) Contenu des variables ou éléments de tableau de A à A (50).

PUT \$,D,W Contenu de l'exclusive variable alphanumérique \$ et des variables de D à W.

Lorsque le contenu de l'exclusive variable alphanumérique doit être stocké, écrire cette dernière en premier.

- (3) Peut être exécutée manuellement ou écrite dans un programme.

GET

["Nom de fichier"] variable 1 [, variable 2]*
Chaîne de caractères

Fonction

Charge des données stockées sur cassette dans une variable.

Paramètre

Nom de fichier: Une chaîne de 1 à 8 caractères. Peut être omis.
Variable 1, Variable 2: Spécification de la variable à charger.

Explication

- (1) Charge des données stockées sur cassette dans une variable spécifiée.
- (2) Les spécifications de variable sont écrites de la manière suivante.

GET A Lit et affecte dans la variable A.

GET A,Z Lit et affecte dans les variables de A à Z.

GET A,A (50) Lit et affecte dans les variables ou les éléments de tableau de A à A (50).

GET \$,D,W Lit et affecte dans l'exclusive variable alphanumérique \$ et dans les variables de D à W.

- (3) Un nom de variable stocké par PUT peut être différent de celui lu par GET.
- (4) Lorsque le nombre de données stockées est inférieur au nombre de variables à lire, les données sont lues et affectées séquentiellement dans les premières variables.
- (5) Peut être exécutée manuellement ou dans un programme.

BEEP

(A)

Fonction

Fait retentir le vibreur.

Paramètre

0: Son grave.

1: Son aigu.

0 est pris par défaut.

Explication

- (1) Génère un son grave ou aigu.
- (2) Peut également être utilisée manuellement.

Exemple

```
10 $=" ABCDEFGHIJKLMNOPQRSTUVWXYZ":N=0
20 FOR I=1 TO 10
30 A$=MID$(RAN#*26+1,1)
40 PRINT CSR4;"<";A$;">";
50 FOR J=1 TO 30
60 K$=KEY$:IF K$=" "THEN80
70 NEXT J
80 IF K$=A$ THEN BEEP 1:N=N+1:GOTO 100
90 BEEP 0
100 PRINT:NEXT I
110 PRINT N:
120 IF N>10 THEN END
130 FOR I=1 TO 10
140 BEEP 0:BEEP 1
150 NEXT I
```

- Appuyer sur les touches alphabétiques qui correspondent aux caractères affichés.

DEFM

[Taille de l'extension mémoire]

Expression numérique

(A)

Fonction

Fournit une extension mémoire.

Paramètre

Taille de l'extension mémoire: Expression numérique.
Les décimales ne sont pas prises en compte. Peut être omis.

$0 \leq \text{Taille de l'extension mémoire} < 69$

Explication

- (1) Etend la mémoire (zone de variable).
- (2) Un nombre arbitraire peut être spécifié en accord avec le nombre de pas de programme restants.
- (3) Etant donné que 8 pas sont utilisés pour chaque extension mémoire, le nombre de pas restants dans la zone de mémoire s'en trouve réduit.
- (4) Lorsque le nombre d'extensions mémoire est omis, le nombre de mémoires couramment spécifié est affiché.
- (5) Peut être exécutée manuellement ou écrite dans un programme. Lorsque cette commande est exécutée manuellement, le nouvel état de spécification (nombre d'extensions mémoire + 26 mémoire de base) est affiché. Lorsqu'elle est écrite dans un programme, le nouvel état de spécification n'est pas affiché.
- (6) Lorsqu'on tente de faire une extension supérieure au nombre de pas de programme restants, une erreur (ERR1) se produit.
- (7) Spécifier DEFM 0 pour annuler l'extension mémoire et retourner aux 26 mémoires de base.

Exemple

```
DEFM 10 [EXE] ***VAR:36
```

```
DEFM [EXE] ***VAR:36
```

```
10 DEFM 10  
20 FOR I=1 TO 10  
30 INPUT Z(I)  
40 NEXT I
```

MODE

Expression numérique

(P)

Fonction

Définit l'état de l'ordinateur.

Paramètre

Expression numérique: Les décimales ne sont pas prises en compte.

$4 \leq \text{Expression numérique} < 9$

Explication

- (1) Définit l'unité angulaire, le mode impression ou annule ce dernier en fonction de l'expression numérique utilisée.
- (2) Les définitions sont les suivantes.
MODE4 Définit l'unité d'angle en degrés.
MODE5 Définit l'unité d'angle en radians.
MODE6 Définit l'unité d'angle en grades.
MODE7 Affiche "PRT" et définit le mode impression.
MODE8 Annule le mode impression.
- (3) Même définition qu'en utilisant la touche **MODE**. Toutefois, les modes RUN et WRT ne peuvent pas être définis par cette commande. En outre, elle ne peut pas être entrée en appuyant sur la touche **MODE** mais sur les touches **M O D E**.

Exemple

```
10 MODE 4  
20 A=SIN 30  
30 MODE 7  
40 PRINT A  
50 MODE 8  
60 END
```


Fonction

Spécifie le format de sortie des données numériques.

Paramètre

F_n : Spécifie le nombre de décimales.

E_n : Spécifie le nombre de chiffres significatifs.

N : Annule une spécification.

Explication

- (1) Spécifie le nombre de décimales ou de chiffres significatifs.
- (2) Le nombre de décimales est spécifié (F_n) par un chiffre de 0 à 9.
- (3) Le nombre de chiffres significatifs est spécifié (E_n) par un chiffre de 0 à 9. "SET E0" indique une spécification de 10 chiffres.
- (4) Les deux spécifications sont annulées par "SET N".
- (5) Peuvent être exécutées manuellement ou écrites dans un programme.

Exemple

```
10 INPUT N
20 SET F5:PRINT N
30 SET E5:PRINT N
40 SET N:GOTO 10
```

```
DEFM 10
DEFM 10
10 DEFM 10
20 FOR I=1 TO 10
30 INPUT Z(I)
40 NEXT I
```

LEN

(Simple variable alphanumérique)

Fonction

Donne la longueur de la chaîne de caractères affectée à une simple variable alphanumérique.

Paramètre

Simple variable alphanumérique: Un variable de tableau ne peut pas être utilisée.

Explication

- (1) Compte le nombre de caractères d'une simple variable alphanumérique.
- (2) La variable alphanumérique utilisée est une simple variable (A\$, Y\$, etc.); un élément de tableau (B\$ (3), etc.) ne peut pas être utilisé.

Exemple

```
10 INPUT A$
20 PRINT LEN(A$)
30 GOTO 10
```

```
10 $="*ABCDEFGHIJKLMNORSTUVWXYZ"
20 INPUT M,N
30 PRINT MID$(M,N)
40 END
```

MID\$ (Emplacement [, Nombre de caractères]) (F)

Expression numérique Expression numérique

Fonction

Extrait un nombre spécifié de caractères à partir d'un emplacement spécifié de l'exclusive variable alphanumérique (\$).

Paramètre

Emplacement: Expression numérique. Les décimales ne sont pas prises en compte.

$$1 \leq \text{Emplacement} < 101$$

Nombre de caractères: Expression numérique.

Les décimales ne sont pas prises en compte.

$$1 \leq \text{Nombre de caractères} < 101$$

Lorsqu'il est omis, tous les caractères qui suivent l'emplacement spécifié sont extraits.

Explication

- (1) Extrait un nombre spécifié de caractères à partir d'un emplacement spécifié de l'exclusive variable alphanumérique (\$).
- (2) Quand l'emplacement spécifié est hors de la chaîne de caractères, un vide est obtenu.
- (3) Quand la longueur de la chaîne de caractères qui restent après l'emplacement spécifié est inférieure au nombre de caractères spécifié, tous les caractères qui suivent l'emplacement sont extraits.

* MID\$ peut être abrégée sous la forme MID.

Exemple

```
10 $="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
20 INPUT M,N
30 PRINT MID$(M,N)
40 END
```

VAL (Simple variable alphanumérique) (F)

Fonction

Convertit les caractères d'une simple variable alphanumérique en valeur numérique.

Paramètre

Simple variable alphanumérique: Une variable de tableau ne peut pas être utilisée.

Explication

- (1) Convertit les caractères d'une simple variable alphanumérique en valeur numérique.
- (2) Lorsque le contenu d'une variable alphanumérique comprend +, -, *, π ou π^- , il est converti en valeur numérique tel quel.

Quand A\$ = "-12.3", VAL(A\$) → -12.3

- (3) Lorsque le contenu d'une variable alphanumérique commence par un autre caractère qu'un chiffre, +, - ou *, une erreur se produit.

Quand A\$ = "A45", VAL(A\$) → -erreur (ERR2)

- (4) Lorsque un caractère autre qu'un chiffre est inséré au milieu, seule la partie située avant ce caractère est convertie en valeur numérique.

Quand A\$ = "78A9", VAL(A\$) → 78

Exemple

```
10 INPUT A$
20 PRINT VAL(A$)
30 END
```


Fonction

Convertit la valeur d'une expression numérique en chaîne de caractères.

Paramètre	Unité	Unité	Unité
α	mm	mm	mm
β	mm	mm	mm
γ	mm	mm	mm
δ	mm	mm	mm
ϵ	mm	mm	mm
ζ	mm	mm	mm
η	mm	mm	mm
θ	mm	mm	mm
ι	mm	mm	mm
κ	mm	mm	mm
λ	mm	mm	mm
μ	mm	mm	mm
ν	mm	mm	mm
ξ	mm	mm	mm
$\omicron$	mm	mm	mm
π	mm	mm	mm
ρ	mm	mm	mm
σ	mm	mm	mm
τ	mm	mm	mm
υ	mm	mm	mm
ϕ	mm	mm	mm
χ	mm	mm	mm
ψ	mm	mm	mm
ω	mm	mm	mm
φ	mm	mm	mm
ϑ	mm	mm	mm
ϖ	mm	mm	mm
ϱ	mm	mm	mm
ς	mm	mm	mm
ζ	mm	mm	mm
η	mm	mm	mm
θ	mm	mm	mm
ι	mm	mm	mm
κ	mm	mm	mm
λ	mm	mm	mm
μ	mm	mm	mm
ν	mm	mm	mm
ξ	mm	mm	mm
$\omicron$	mm	mm	mm
π	mm	mm	mm
ρ	mm	mm	mm
σ	mm	mm	mm
τ	mm	mm	mm
υ	mm	mm	mm
ϕ	mm	mm	mm
χ	mm	mm	mm
ψ	mm	mm	mm
ω	mm	mm	mm
φ	mm	mm	mm
ϑ	mm	mm	mm
ϖ	mm	mm	mm
ϱ	mm	mm	mm
ς	mm	mm	mm
ζ	mm	mm	mm
η	mm	mm	mm
θ	mm	mm	mm
ι	mm	mm	mm
κ	mm	mm	mm
λ	mm	mm	mm
μ	mm	mm	mm
ν	mm	mm	mm
ξ	mm	mm	mm
$\omicron$	mm	mm	mm
π	mm	mm	mm
ρ	mm	mm	mm
σ	mm	mm	mm
τ	mm	mm	mm
υ	mm	mm	mm
ϕ	mm	mm	mm
χ	mm	mm	mm
ψ	mm	mm	mm
ω	mm	mm	mm
φ	mm	mm	mm
ϑ	mm	mm	mm
ϖ	mm	mm	mm
ϱ	mm	mm	mm
ς	mm	mm	mm
ζ	mm	mm	mm
η	mm	mm	mm
θ	mm	mm	mm
ι	mm	mm	mm
κ	mm	mm	mm
λ	mm	mm	mm
μ	mm	mm	mm
ν	mm	mm	mm
ξ	mm	mm	mm
$\omicron$	mm	mm	mm
π	mm	mm	mm
ρ	mm	mm	mm
σ	mm	mm	mm
τ	mm	mm	mm
υ	mm	mm	mm
ϕ	mm	mm	mm
χ	mm	mm	mm
ψ	mm	mm	mm
ω	mm	mm	mm
φ	mm	mm	mm
ϑ	mm	mm	mm
ϖ	mm	mm	mm
ϱ	mm	mm	mm
ς	mm	mm	mm
ζ	mm	mm	mm
η	mm	mm	

Expression numérique: Valeur numérique, expression de calcul, variable numérique de tableau numérique.

Explication

- (1) Convertit la valeur d'une expression numérique en chaîne de caractères.
- (2) Lorsque l'expression numérique est une expression de calcul, le résultat est converti en chaîne de caractères.
- (3) Lorsqu'une expression numérique est positive, le signe est supprimé et seuls les chiffres sont convertis.

Example

```
10 PRINT STR$(123)
20 PRINT STR$(45+78)
30 A=963
40 PRINT STR$(A)
50 END
```

SIN

Argument
Expression numérique

COS

Argument
Expression numérique

TAN

Argument
Expression numérique

Fonction

Pour un argument, donne la valeur d'une fonction trigonométrique.

Paramètre	Unité	Unité	Unité
α	mm	mm	mm
β	mm	mm	mm
γ	mm	mm	mm
δ	mm	mm	mm
ϵ	mm	mm	mm
ζ	mm	mm	mm
η	mm	mm	mm
θ	mm	mm	mm
ι	mm	mm	mm
κ	mm	mm	mm
λ	mm	mm	mm
μ	mm	mm	mm
ν	mm	mm	mm
ξ	mm	mm	mm
$\omicron$	mm	mm	mm
π	mm	mm	mm
ρ	mm	mm	mm
σ	mm	mm	mm
τ	mm	mm	mm
υ	mm	mm	mm
ϕ	mm	mm	mm
χ	mm	mm	mm
ψ	mm	mm	mm
ω	mm	mm	mm
φ	mm	mm	mm
ϑ	mm	mm	mm
ϖ	mm	mm	mm
ϱ	mm	mm	mm
ς	mm	mm	mm
ζ	mm	mm	mm
η	mm	mm	mm
θ	mm	mm	mm
ι	mm	mm	mm
κ	mm	mm	mm
λ	mm	mm	mm
μ	mm	mm	mm
ν	mm	mm	mm
ξ	mm	mm	mm
$\omicron$	mm	mm	mm
π	mm	mm	mm
ρ	mm	mm	mm
σ	mm	mm	mm
τ	mm	mm	mm
υ	mm	mm	mm
ϕ	mm	mm	mm
χ	mm	mm	mm
ψ	mm	mm	mm
ω	mm	mm	mm
φ	mm	mm	mm
ϑ	mm	mm	mm
ϖ	mm	mm	mm
ϱ	mm	mm	mm
ς	mm	mm	mm
ζ	mm	mm	mm
η	mm	mm	mm
θ	mm	mm	mm
ι	mm	mm	mm
κ	mm	mm	mm
λ	mm	mm	mm
μ	mm	mm	mm
ν	mm	mm	mm
ξ	mm	mm	mm
$\omicron$	mm	mm	mm
π	mm	mm	mm
ρ	mm	mm	mm
σ	mm	mm	mm
τ	mm	mm	mm
υ	mm	mm	mm
ϕ	mm	mm	mm
χ	mm	mm	mm
ψ	mm	mm	mm
ω	mm	mm	mm
φ	mm	mm	mm
ϑ	mm	mm	mm
ϖ	mm	mm	mm
ϱ	mm	mm	mm
ς	mm	mm	mm
ζ	mm	mm	mm
η	mm	mm	mm
θ	mm	mm	mm
ι	mm	mm	mm
κ	mm	mm	mm
λ	mm	mm	mm
μ	mm	mm	mm
ν	mm	mm	mm
ξ	mm	mm	mm
$\omicron$	mm	mm	mm
π	mm	mm	mm
ρ	mm	mm	mm
σ	mm	mm	mm
τ	mm	mm	mm
υ	mm	mm	mm
ϕ	mm	mm	mm
χ	mm	mm	mm
ψ	mm	mm	mm
ω	mm	mm	mm
φ	mm	mm	mm
ϑ	mm	mm	mm
ϖ	mm	mm	mm
ϱ	mm	mm	mm
ς	mm	mm	mm
ζ	mm	mm	mm
η	mm	mm	

Argument: Expression numérique

- $-1440^\circ < \text{argument} < 1440^\circ$ (degrés)
- $-8\pi < \text{argument} < 8\pi$ (radians)
- $-1600 < \text{argument} < 1600$ (grades)

Toutefois, pour TAN, " $| \text{Argument} | = (2n - 1) * 1 \text{ angle droit}$ " est exclu.

$$1 \text{ angle droit} = 90^\circ = \frac{\pi}{2} \text{ radians} = 100 \text{ grades.}$$

Explication

- (1) Pour un argument, donne la valeur d'une fonction trigonométrique.
- (2) La valeur dépend de l'unité d'angle définie (par la touche  ou la commande MODE).

ASN Argument
Expression numérique

ATN Argument
Expression numérique

Fonction

Fonction trigonométrique inverse qui obtient un angle pour un argument donné.

Paramètre

Argument: Expression numérique.

Pour ASN, ACS, $-1 \leq \text{argument} \leq 1$

Explication

- (1) Fonction trigonométrique inverse qui obtient un angle pour un argument donné.
- (2) La valeur dépend de l'unité d'angle définie (par la touche **MODE** ou la commande MODE).
- (3) Les valeurs de ces fonctions sont comprises entre les éléments suivants.

$$-90^\circ \leq \text{ASN } X \leq 90^\circ$$

$$0^\circ \leq \text{ACS } X \leq 180^\circ$$

$$-90^\circ < \text{ATN } X < 90^\circ$$

LOG Argument
Expression numérique

LN Argument
Expression numérique

Fonction

Donne la valeur d'une fonction logarithmique.

Paramètre

Argument: Expression numérique.

$0 < \text{argument}$.

Explication

Donne la valeur d'une fonction logarithmique.

- LOG Fonction logarithme $\log_{10} x, \log x$
- LN Fonction logarithme népérien $\log_e x, \ln x$

EXP Argument
Expression numérique

Fonction

Donne la valeur d'une fonction exponentielle.

Paramètre

Argument: Expression numérique

$-227 \leq \text{argument} \leq 230$

Explication

Donne la valeur d'une fonction exponentielle.

$$\text{EXP } e^x$$

SQR Argument
Expression numérique

Fonction

Donne la racine carrée d'un argument.

Paramètre

Argument: Expression numérique.

$0 \leq \text{argument}$

Explication

Donne la racine carrée d'un argument.

$$\text{SQR } \sqrt{x}$$

ABS

Argument
Expression numérique

Fonction

Donne la valeur absolue d'un argument.

Paramètre

Argument: Expression numérique.

Explication

Donne la valeur absolue d'un argument.

ABS | x |

SGN

Argument
Expression numérique

Fonction

Donne une valeur qui correspond au signe d'un argument.

Paramètre

Argument: Expression numérique.

Explication

Donne une valeur qui correspond au signe d'un argument.

- 1, quand un argument est positif.
- 0, quand un argument est 0
- 1, quand un argument est négatif.

INT

Argument
Expression numérique

Fonction

Donne le plus grand entier qui n'est supérieur à l'argument.

Paramètre

Argument: Expression numérique.

Explication

Donne le plus grand entier qui n'est pas supérieur à l'argument.

INT 12.56 → 12

INT -78.1 → -79

FRAC

Argument
Expression numérique

Fonction

Donne la partie fractionnaire d'un argument.

Paramètre

Argument: Expression numérique.

Donne la partie fractionnaire d'un argument.

Le signe est en concordance avec celui de l'argument.

RND

(Argument , Position)
Expression numérique Expression numérique

Fonction

Donne la valeur d'un argument arrondie à la position spécifiée.

Paramètre

Argument: Expression numérique.

Position: Expression numérique. Les décimales ne sont pas prises en compte.

$-100 < \text{position} < 100$

Explication

- (1) Donne la valeur d'un argument arrondie à la position spécifiée.
- (2) L'argument est arrondi à la 3^e décimale.

→ **RND** (x , -3)

L'argument est arrondi à la position des centaines.

→ **RND** (x , 2)

RAN

Fonction

Donne un nombre aléatoire compris entre 0 et 1.

Explication

(1) Donne un nombre aléatoire compris entre 0 et 1.

$$0 < \text{nombre aléatoire} < 1$$

(2) Le nombre aléatoire possède 10 chiffres.

Exemple

Fournit un chiffre aléatoire compris entre 0 et 9.

$$\text{INT}(\text{RAN} \# * 10)$$

Fournit un chiffre aléatoire compris entre 1 et 5.

$$\text{INT}(\text{RAN} \# * 5) + 1$$

Fournit un nombre aléatoire de deux chiffres compris entre 10 et 99.

$$\text{INT}(\text{RAN} \# * 90) + 10$$

DEG

(Degrés [, Minutes [, Secondes]])
Expression numérique Expression numérique Expression numérique

Fonction

Convertit le sexagésimal en décimal.

Paramètre

Degrés: Expression numérique.

Minutes: Expression numérique.

Secondes: Expression numérique.

$$| \text{DEG}(\text{degrés, minutes, secondes}) | < 10^{100}$$

Explication

Convertit le sexagésimal exprimé en degrés, minutes et secondes en décimal.

Exemple

DEG(12, 34, 56)

12.58222222

```
10 INPUT A,B,C
20 PRINT DEG(A,B,C)
30 END
```

DMS\$

Argument
Expression numérique

Fonction

Convertit le décimal en sexagésimal.

Paramètre

Argument: Expression numérique.

$$| \text{Expression numérique} | < 10^{100}$$

Explication

(1) Convertit le décimal en sexagésimal.

(2) Le résultat de la conversion est une chaîne de caractères.

Exemple

DMS\$(45.678)

45°40'40.8

```
10 INPUT A
20 $=DMS$(A)
30 PRINT$
40 END
```


NEW

Fonction

Efface les données pour la fonction banque de données.

Explication

- (1) Efface toutes les données stockées.
- (2) Ne peut pas être utilisée quand un mot de passe est spécifié.
- (3) Ne peut être exécutée qu'en mode WRT.

Exemple

MODE 1
NEW # EXE

LIST

Fonction

Affiche toutes les données pour la fonction banque de données.

Explication

- (1) Affiche séquentiellement les données à partir du début.
- (2) Les affichages contiennent un numéro de séquence et des données.
- (3) Etant donné que les données sont automatiquement affichées séquentiellement, appuyer sur la touche **STOP** pour arrêter l'affichage. Appuyer sur la touche **EXE** pour procéder à l'affichage des données suivantes.
- (4) En mode impression (**MODE** **7**), l'affichage n'est pas arrêté mais est effectué à grande vitesse.
- (5) Ne peut pas être exécutée quand un mot de passe est spécifié.
- (6) Ne peut être exécutée dans le mode entrée pour la fonction banque de données (**MODE** **9**).

Exemple

LIST# EXE

SAVE

["Nom de fichier"]
Chaîne de caractères

Fonction

Pour la fonction banque de données, stocke les données sur une cassette.

Paramètre

Nom de fichier: Chaîne de 1 à 8 caractères. Peut être omis.

Explication

- (1) Stocke les données sur une cassette.
- (2) Comme pour la fonction banque de données, les données ne sont pas stockées avec SAVE ou SAVE ALL, s'assurer d'utiliser SAVE#.
- (3) Si un mot de passe a été spécifié, le stockage est effectué avec ce mot de passe. De ce fait, le même mot de passe devra être spécifié lors du chargement par la commande LOAD#.
- (4) Pour la fonction banque de données, ne peut pas être exécutée en mode entrée.

Exemple

SAVE # EXE
SAVE # "OLYMPIA" EXE

LOAD

["Nom de fichier"]
Chaîne de caractères

Fonction

Pour la fonction banque de données, charge les données à partir d'une cassette.

Paramètre

Nom de fichier: Chaîne de 1 à 8 caractères. Peut être omis.

Explication

- (1) Charge les données stockées sur une cassette en mémoire.
- (2) Quand des données qui ont été stockées avec un mot de passe, sont chargées, ce mot de passe doit être spécifié.
- (3) Si des données sont présentes dans l'ordinateur, elles sont effacées puis les nouvelles sont chargées.
- (4) Pour la fonction banque de données, ne peut pas être exécutée en mode entrée.

Exemple

LOAD # EXE
LOAD # "OLYMPIA" EXE

Fonction

Pour la fonction banque de données, lit les données.

Paramètre

Nom de variable: Variable numérique ou alphanumérique. Un variable de tableau peut également être utilisée.

Explication

- (1) Lit séquentiellement les données stockées et les affecte dans une variable.
- (2) Seules des données numériques peuvent être lues et affectées dans une variable numérique. Si des données alphanumériques sont utilisées, une erreur (ERR2) se produit.
- (3) Après qu'une première instruction READ# ait lu des données, l'instruction READ# suivante lira les données suivantes.
- (4) Quand les données sont séparées par une virgule, elles sont lues et affectées dans l'ordre dans lequel elles sont écrites.

Exemple) DATA

N°1 A, X, Y

N°2 B, Z

N°3 C



Ordre de lecture

A→X→Y→B→Z→C

- (5) Lorsque les données à lire n'existent pas, une erreur (ERR4) se produit.
- (6) L'ordre des données à lire peut être modifié par RESTORE# (voir page 159)
- (7) Quand un espace existe au début d'une donnée, il n'est pas pris en compte.
- (8) Quand une donnée est placée entre guillemets, la chaîne de caractères entre guillemets est lue.

Exemple

⟨Données⟩

⟨Programme⟩

N°1 1, 2, 3

10 A=0

N°2 4, 5, 6

20 READ#\$

N°3 7, 8, 9

30 IF \$="" THEN 60

N°4 10,

40 A=A+VAL(\$)

50 GOTO 20

60 PRINT "Σx="; A

70 END

- Lit des données numériques pour obtenir une somme.

Fonction

Pour la fonction banque de données, recherche des données et spécifie l'ordre des données à lire par READ#.

Paramètre

Chaîne de caractères recherchée: Expression alphanumérique. Quand une chaîne de caractères est utilisée, la placer entre guillemets.

Numéro de ligne: Expression numérique.

$0 < \text{numéro de ligne} < 10000$

Numéro de zone de programme: Expression numérique.

$0 \leq \text{numéro de zone de programme} < 10$

Explication

- (1) Recherche des données et spécifie l'ordre des données à lire par l'instruction READ# suivante.
- (2) La relation entre un paramètre et la recherche des données est la suivante.

① RESTORE#

Quand la chaîne de caractères recherchée et les paramètres suivants sont omis, les données sont lues à partir du début par l'instruction READ# suivante.

② RESTORE# "chaîne de caractères recherchée"

Recherche la donnée qui commence par la chaîne de caractères recherchée et cette donnée est lue par l'instruction READ# suivante.

③ RESTORE# "chaîne de caractères recherchée" { 0 }
1

Lorsque 0 est spécifié, cela revient au même que pour le point ② ci-dessus. Lorsque 1 est spécifié, la première donnée de la ligne qui comprend les données recherchées est lue par l'instruction READ# suivante.

④ RESTORE# "chaîne de caractères recherchée", [{ 0 }] ,
[{ 1 }] ,
{ numéro de ligne
numéro de zone de programme }

Lors de l'exécution de la recherche, si une donnée appropriée n'existe pas, un branchement est effectué à la ligne spécifiée ou à la zone de programme.

- * Dans les cas ② et ③, quand une donnée appropriée n'existe pas, une erreur (ERR4) se produit.
- * Dans le cas ④, quand un numéro de ligne de branchement n'existe pas ou quand il n'y a pas de programme dans la zone de programme, une erreur (ERR4) se produit.

Exemple

<Donnée>

No.1 FOSTER,347-4811,NEW YORK
No.2 SMITH,045-211-0821,CHICAGO
No.3 JONES,06-314-2681,SAN FRANCISCO
No.4 BROWN,075-351-1161,LOS ANGELES

<Programme>

```
10 RESTORE#
20 READ# $
30 PRINT $
40 RESTORE# "S"
50 READ# $
60 PRINT $
70 RESTORE# "LO",1
80 READ# $
90 PRINT $
100 RESTORE# "AA",1,200
110 READ# $
120 PRINT $
130 END
200 PRINT "END"
210 END
```

La donnée stockée au début est affichée.

La donnée qui commence par la lettre S est affichée.

Recherche les données dont les deux premières lettres sont LO et affiche la première donnée sur la ligne qui comprend ces données.

Quand aucune donnée ne commence par les deux lettres AA, se branche à la ligne 200.

RUN
EXE
EXE
EXE
EXE

FOSTER
SMITH
BROWN
END

WRITE#

[Données] [Données]*]
Expression Expression

Fonction

Pour la fonction banque de données, réécrit ou supprime des données.

Paramètre

Données: Expression numérique ou alphanumérique. Quand une chaîne de caractères est utilisée, la placer entre guillemets.

Explication

- (1) Ecrit des données dans la ligne couramment spécifiée par RESTORE#.
- (2) Les données sont nouvellement écrites sans aucune relation avec l'existence de données dans la zone de données appropriée.
- (3) Quand toutes les données sont omises, s'il y avait des données sur une ligne, la ligne entière est supprimée.
- (4) Quand plusieurs données existent, elles peuvent être écrites en les séparant par des virgules.
- (5) Après qu'une première instruction WRITE# ait écrit des données, les données suivantes seront écrites lors de l'exécution de l'instruction WRITE# suivante.

Exemple

```
10 REM WRITE
20 RESTORE#
30 WRITE# "A,B,C"
40 RESTORE#
50 FOR I=1 TO 3
60 READ# $:PRINT $
70 NEXT I
80 PRINT " "
90 REM CHANGE
100 RESTORE#
110 FOR I=1 TO 3
120 WRITE# STR$(I)
130 NEXT I
140 RESTORE#
150 FOR I=1 TO 3
```

De nouvelles données sont écrites.

Les données sont réécrites.

```

160 READ# $: PRINT $:
170 NEXT I
180 PRINT " "
190 REM CLEAR
200 RESTORE#
210 WRITE#
220 RESTORE#
230 READ# $

```

Efface les données.

Opération

RUN 



Affichage

ABC
123
ERR4 P0-230

Manque de données par suite de l'effacement.

```

100 RESTORE#
110 READ# $
120 PRINT $
130 END
140 PRINT "END"
150 END

```

```

10 REM WRITE
20 RESTORE#
30 WRITE# "ABC"
40 RESTORE#
50 FOR I=1 TO 3
60 READ# $:PRINT $
70 NEXT I
80 PRINT " "
90 REM CHANGE
100 RESTORE#
110 FOR I=1 TO 3
120 WRITE# STR$(I)
130 NEXT I
140 RESTORE#
150 FOR I=1 TO 3

```

CHAPITRE 5

BIBLIOTHEQUE DE PROGRAMMES

1. Conversion de nombres décimaux en nombres hexadécimaux
2. Calculs de prêt (Mensualités égales)
3. Exercices d'arithmétique
4. Analyse de régression du second degré
5. Jeu de remise en ordre

1. CONVERSION DE NOMBRES DECIMAUX EN NOMBRES HEXADECIMAUX

Ce programme peut être utilisé pour convertir des nombres décimaux en nombres hexadécimaux, et vice versa. Il convient donc très bien pour les calculs d'adresse.

Exemples)

1. Quel est l'équivalent décimal de l'hexadécimal 14AF?
2. Quel est l'équivalent décimal de l'hexadécimal A540?
3. Quel est l'hexadécimal équivalent du décimal 4582?
4. Quel est l'hexadécimal équivalent du décimal 18657030?

Opération

Affichage

- Lancer le programme.

RUN 

10+1-0-2+16

- Le menu est affiché. Pour obtenir l'équivalent décimal d'un nombre hexadécimal, appuyer sur la touche . Pour obtenir l'équivalent hexadécimal d'un nombre décimal, appuyer sur la touche . Pour terminer le programme, appuyer sur la touche . Maintenant, appuyer sur la touche .



HEX?

- Entrer la donnée hexadécimale.

14AF 

5295

- Entrer la donnée suivante.

A540 

HEX?

42304

- Si "HEX" apparaît après l'obtention de l'équivalent décimal d'un nombre hexadécimal, appuyer uniquement sur la touche .



HEX?

10+1-0-2+16

- Ensuite, appuyer sur la touche  pour obtenir l'équivalent hexadécimal du nombre décimal.



DEC?

- Entrer la donnée décimale.

4582 

11E6

- Entrer la donnée suivante.

18657030 

DEC?

11CAF06

- Pour terminer le programme, revenir au menu et appuyer sur la touche .



DEC?

10+1-0-2+16



----END----

Liste du programme

```
10 FOR I=0 TO 5: R
  EAD A$(I): NEXT
  I
20 DATA A,B,C,D,E,
  F
30 PRINT "10+1-0-2
  +16":
40 $=KEY$: IF $="
  " THEN 40
50 BEEP: PRINT
60 IN VAL($): GOTO
  30,220
70 PRINT "----END-
  ----": END
80 INPUT "HEX", $
90 IF $="" THEN 30
100 Q=LEN($): R=0
110 FOR I=0 TO Q-1
120 P$=MID$(Q-I,1)
130 IF P$="9" THEN
  Q=VAL(P$): 50T
  0 100
140 FOR J=0 TO 5
150 IF A$(J)=P$ THE
  N Q=J+10: GOTO
  160
160 NEXT J
170 BEEP: GOTO 100
180 R=R+16*I*Q
190 NEXT I
```

```
200 BEEP 1: PRINT X
210 GOTO 40
220 INPUT "DEC", $
230 IF $="" THEN 30
240 S=VAL($): P=S:0
  =0: $=""
250 0=FRAC(S/16)*1
  6+S=0
260 IF P<16 THEN 29
  0
270 P=INT(P/16): Q=
  Q+1
280 GOTO 260
290 IF Q=0 THEN 350
300 FOR I=0 TO 1: ST
  EP -1
310 P=INT(S/16*I)
320 Z=P: GOSUB 500
330 S=INT( FRAC(S/
  16*I)*16*I+.5)
340 NEXT I
350 Z=0: GOSUB 500
360 BEEP 1: PRINT $
370 GOTO 220
500 IF Z>9 THEN $=$
  +A$(Z-10): RETU
  RN
510 $=$+STR$(Z): R
  ETURN
Total: 507 pas
```

2. CALCULS DE PRET (MENSUALITES EGALES)

Ce programme est utilisé pour calculer une des trois données suivantes, mensualité, montant du prêt et nombre de mensualités, quand le taux d'intérêt annuel et les deux autres données sont connus.

Entrée	Affichage
• Taux d'intérêt annuel, mensualité et nombre de mensualités	→ Montant du prêt
• Taux d'intérêt annuel, mensualité et montant du prêt	→ Nombre de mensualités
• Taux d'intérêt annuel, montant du prêt et nombre de mensualités	→ Mensualité

Entrer le taux d'intérêt annuel sous forme de pourcentage.
La mensualité et le montant du prêt sont affichés en francs et le nombre de mensualités est en mois. Les résultats des calculs sont arrondis.

Formules)

P : Mensualité PV : Montant du prêt
 i : Taux d'intérêt mensuel (En fait, le taux d'intérêt annuel est entré)
 n : Nombre de mensualités

$$P = PV \frac{i}{1 - (1 + i)^{-n}}$$

$$PV = P \frac{1 - (1 + i)^{-n}}{i}$$

$$n = - \frac{\ln\left(1 - \frac{i \cdot PV}{P}\right)}{\ln(1 + i)}$$

Exemples)

1. Si un prêt est accordé sur la condition que le remboursement est fait en mensualités de F200 pendant 24 mois à un taux d'intérêt annuel de 7,5%, quel est le montant de ce prêt?
2. Si un prêt de F5.000 est accordé à un taux d'intérêt annuel de 6,5% en 36 mensualités (3 ans), quel sera le montant d'une mensualité?
3. Si un prêt de F25.000 est accordé à un taux d'intérêt annuel de 5,8% et des mensualités de F250, quel sera le nombre de mensualités?

Opération	Affichage
• Lancer le programme	RUN Payment?
• Entrer la mensualité. Dans l'exemple 1 ci-dessus, le montant est de F200.	200 Loan?
• Dans cet exemple, le problème consiste à obtenir le montant du prêt. Appuyer donc sur la touche .	Rate?
• Entrer le taux d'intérêt annuel.	7.5 Months?
• Entrer le nombre de mensualités.	24 Loan 4444 * Le montant du prêt est affiché.
• Essayons l'exemple 2.	RUN Payment? Loan?
• Le montant du prêt est de F5.000.	5000 Rate?
• Le taux d'intérêt annuel est de 6,5%.	6.5 Months?
• Le nombre de mensualités est de 36.	36 Payment 153 * Le montant d'une mensualité est affiché.
• Voyons maintenant l'exemple 3.	RUN Payment?

- Le montant d'une mensualité est de F250.

250 **EXE**

- Le montant du prêt est de F25.000.

25000 **EXE**

- Le taux d'intérêt annuel est de 5,8%.

5.8 **EXE**

- Pour obtenir le nombre de mensualités, appuyer sur la touche **EXE**.

EXE

* Le nombre de mensualités est affiché.

■ Liste du programme

```

10 CLEAR
20 RESTORE
30 DATA Payment,Lo
  an,Rate,Months
40 FOR I=0 TO 3
50 READ E$(I): PRI
  NT E$(I):
60 INPUT $
70 IF $="" THEN AC
  (I)=VAL($):Y=Y+
  1
80 NEXT I
90 IF Y<3 THEN BEE
  P 0: GOTO 10
100 K=C/1200
110 FOR I=0 TO 3
120 IF A(I)=0 THEN
  150
130 NEXT I
140 BEEP 0: GOTO 10
150 ON I GOTO 170,1
  40,180
160 A=INT(B*K/(1-(
  1+K)*↑-0)+.5): G
  0 TO 190
170 B=INT(A*(1-(1+
  K)*↑-0)/K+.5): G
  0 TO 190
180 D=INT(-LN(1-K
  *B/A)/LN(1+K)+
  .5)
190 BEEP 1
200 PRINT E$(I):A(I)
  )

```

Total: 287 pas

3. EXERCICES D'ARITHMETIQUE

Ce programme présente des problèmes arithmétiques d'addition, de soustraction, de multiplication et de division.

Pour ces quatre opérations, on peut choisir d'effectuer des calculs soit avec des nombres d'un chiffre soit avec des nombres de deux chiffres.

Dix problèmes font un ensemble. Un score de 10 points est marqué pour chaque problème résolu correctement. Les divisions sont calculées jusqu'à deux décimales.

Opération	Affichage	
RUN EXE	Push + - * /	... Appuyer sur + , - , * ou / pour effectuer des additions, des soustractions, des multiplications ou des divisions.
+	1 or 2	... Sélectionner des nombre d'un ou deux chiffres.
1	1 > 3+9=?	... Problème 1
12 EXE	2 > 2+7=?	... Problème 2
9 EXE	3 > 2+6=?	... Problème 3
	10 > 6+7=?	... Problème 10
13 EXE	SCORE:100	... Indication du score

Chaque fois qu'une réponse correcte est faite, une tonalité apparaît. Un sans-faute est salué par une fanfare.

■ Liste du programme

```

10 E=0
20 PRINT "Push + -
  + /";
30 K$= KEY$
40 IF K$="+" THEN
  M=1: GOTO 90
50 IF K$="-" THEN
  M=2: GOTO 90
60 IF K$="*" THEN
  M=3: GOTO 90
70 IF K$="/" THEN
  M=4: GOTO 90
80 GOTO 30
90 BEEP 1
100 PRINT : PRINT "
  1 or 2";
110 K$= KEY$
120 IF K$="1" THEN
  N=9: GOTO 150
130 IF K$="2" THEN
  N=99: GOTO 150
140 GOTO 110
150 BEEP 1
160 FOR I=1 TO 10
170 PRINT
180 A= INT( RAN#*N+
  1)
190 B= INT( RAN#*N+
  1)
200 ON M GOSUB 300,
  400,500,600
210 PRINT "(<"; STR$
  (I);">";$;: INP
  UT 0
220 BEEP 0:0= INT(0
  *100)/100
230 IF C=0 THEN E=E
  +1: BEEP 1: BEE
  P 1
240 NEXT I
250 PRINT "SCORE:":
  E*10;
260 IF E=10 THEN FO
  R I=1 TO 10: BE
  EP 1: BEEP 0: N
  EXT I
270 END
300 $= STR$(A)+"*"+
  STR$(B)+"="
310 C=A*B
320 RETURN
400 IF A<B THEN F=A
  :A=B:B=F
410 $= STR$(A)+"*"+
  STR$(B)+"="
420 C=A*B
430 RETURN
500 $= STR$(A)+"*"+
  STR$(B)+"="
510 C=A*B
520 RETURN
600 IF A<B THEN F=A
  :A=B:B=F
610 $= STR$(A)+"*"+
  STR$(B)+"="
620 C= INT(A/B*100)
  /100
630 RETURN

```

Total: 540 pas

4. ANALYSE DE REGRESSION DU SECOND DEGRE

Ce programme est utilisé pour obtenir une valeur estimée de y par l'analyse de régression du second degré des données x et y .

Les calculs sont effectués sur la base $y = ax^2 + bx + c$ à l'aide des formules suivantes:

$$a = \frac{\sum(x^2 \cdot y) \cdot \sum(x \cdot x) - \sum(x \cdot y) \cdot \sum(x \cdot x^2)}{\sum(x \cdot x) \sum(x^2 \cdot x^2) - |\sum(x \cdot x^2)|^2}$$

$$b = \frac{\sum(x \cdot y) \sum(x^2 \cdot x^2) - \sum(x^2 \cdot y) \sum(x \cdot x^2)}{\sum(x \cdot x) \cdot \sum(x^2 \cdot x^2) - |\sum(x \cdot x^2)|^2}$$

$$c = \frac{\sum y_i}{n} - b \frac{\sum x_i}{n} - a \frac{\sum x_i^2}{n}$$

Le programme de la zone P0 est utilisé pour l'entrée de données. Entrer les données x et y ; le programme obtient alors $\sum x$, $\sum y$, $\sum x^2$, $\sum xy$, $\sum x^3$, $\sum x^4$ et $\sum x^2 \cdot y$.

Le programme de la zone P1 est utilisé pour calculer a , b et c à partir des $\sum x$, $\sum y$, $\sum x^2$, $\sum xy$, $\sum x^3$, $\sum x^4$ et $\sum x^2 \cdot y$ obtenus à l'aide du programme de la zone P0. Ensuite, on obtient une estimation de y correspondant au x entré.

Exemple)

	1	2	3	4	5	6	7
x	1	5	8	11	15	18	22
y	21	30	41	54	70	①	②

Par une analyse de régression des données ci-dessus, obtenir les estimations de ① et ②.

Opération

GET P0
 1 EXE
 21 EXE
 5 EXE

Affichage

x	1?
y	1?
x	2?
y	2?

Entrer les données suivantes dans l'ordre.

70 EXE

y	5?
x	6?

Après avoir entré toutes les données, exécuter le programme de la zone P1.

Opération	Affichage
P1	a = 0.08967a
OK	b = 2.14288b
OK	c = 18.23781c
OK	x = ?
18 OK	y = 85.86245 Estimation de y
OK	x = ?
22 OK	y = 108.78103 Estimation de y

En utilisant le programme de la zone P1 de cette façon, *a*, *b*, *c* et l'estimation de *y* peuvent être obtenus.

Ici, les résultats des calculs sont arrondis à cinq décimales. Si l'arrondissement est inutile ou si on veut arrondir à un autre nombre de décimales, la ligne n° 90 du programme de la zone P1 doit être changée.

■ Liste du programme

P0	P1	Contenus des variables
10 CLEAR	10 D=P-M*2/M	A $\sum x^2$
20 PRINT "x":M+1::	20 E=R-M*0/M	B $\sum x^4$
INPUT X	30 F=R-M*P/M	C $x^2 \cdot y$
30 PRINT "y":M+1::	40 G=C-P*0/M	I a
INPUT Y	50 H=B-P*2/M	J b
40 N=M+X:0=0+Y	60 I=(G*0-E*F)/(D*H-F*2)	K c
50 P=P+X*2:R=R+X*Y	70 J=(E*M-G*F)/(D*H-F*2)	N $\sum x$
60 A=A+X*3:B=B+X*4	80 K=0/M-J*M/M-I*P/M	M Nombre de données
:C=C+X*2*Y		O $\sum y$
70 M=M+1	90 SET FS	P $\sum x^2$
80 GOTO 20	100 PRINT "a=":I,"b=":J,"c=":K	R $\sum x \cdot y$
	110 INPUT "x=":X	
	120 Y=I*X*2+J*X+K	
	130 PRINT "y=":Y	
	140 GOTO 110	

Total: 309 pas

5. JEU DE REMISE EN ORDRE

Dans ce jeu très amusant, il faut remettre des chiffres aléatoires, de 0 à 9, dans l'ordre numérique (0123456789) le plus rapidement possible. On utilise les touches **←**, **→** et **+**. L'appui sur la touche **←** déplace l'astérisque clignotant d'une position vers la gauche, l'appui sur la touche **→** le déplace d'une position vers la droite. Quand on appuie sur la touche **+**, le chiffre de l'astérisque clignotant est échangé avec le chiffre situé dans le panier.

Lecture de l'affichage

5710*93642>
Chiffres aléatoires Contenu du panier

Etant donné que le score représente le temps passé à la remise en ordre, plus il est petit, meilleur est le résultat. Quand on utilise le jeu pour la première fois ou après l'exécution d'un autre programme, ne pas oublier d'entrer CLEAR **EXE**.

Opération	Affichage
CLEAR EXE	Hi-Sc:0 ... Le meilleur score précédent est affiché.
RUN EXE	<<WAIT!>> ... Le problème est en cours de préparation.
	3721*94650>
+	3721*94650>8 ... Mettre le 8 dans le panier.
←←←←	3721 946*0>8 ... Amener l'astérisque où il devrait être.
+	3721 946*0>5 ... Remplacer le 5 par le 8.
←←←	3721 *4680>5 ... Amener l'astérisque au 9.
+	3721 *4680>9 ... Remplacer le 5 par le 9.
Répéter les opérations ci-dessus.	...
+	0123*56789>4
	[0123456789] ... Remise en ordre terminée.
	SCORE 132 ... Score

Pour une nouvelle partie, entrer RUN **EXE**.

* Quelle est votre dextérité?

Score	Evaluation
Moins de 50	Vous êtes un génie! Vous avez aussi beaucoup de chance.
50—69	Vous avez d'excellents réflexes.
70—99	Moyen. Un peu d'entraînement est nécessaire pour faire mieux.
100—149	Vous manquez encore d'entraînement.
150 ou plus	Quelque chose ne va pas. Essayer de nouveau.

■ Liste du programme

```

10 PRINT "Hi-Sc:";
  H:= FOR I=1 TO
    10: BEEP 1: NEX
    I I
20 $=""
30 PRINT : PRINT "
  << WAIT ! >>":
40 A$= STR$( INT(
  RAN#*10))
50 FOR I=1 TO LEN(
  $)
60 IF A$= MID$(I,1
  ) THEN 40
70 NEXT I
80 $=$+A$
90 IF LEN($)<10 TH
  EN 40
100 PRINT
110 X=5:B$="" :N=0
120 PRINT CSR0:$;"
  ":B$:
130 PRINT CSRX:"*":
140 K$= KEY$
150 IF K$="4" THEN
  BEEP 0:X=X-1: I
  F X<0 THEN X=0
160 IF K$="6" THEN
  BEEP 0:X=X+1: I
  F X>9 THEN X=9
170 IF K$="+" THEN
  GOSUB 500
180 N=N+1
190 IF $*"012345678
  9" THEN 120
200 IF H=0 THEN H=N
210 IF H=N THEN H=N
220 PRINT : PRINT "
  [":$:"I":
230 FOR I=1 TO 10
240 BEEP 0: BEEP 1
250 NEXT I
260 PRINT
270 PRINT "SCORE":N
  ;
280 END
500 BEEP 1
510 IF X=0 THEN C$=
  MID$(1,1):B$=
  + MID$(2): GOTO
  540
520 C$= MID$(X+1,1)
530 $= MID$(1,X)+B$
  + MID$(X+2)
540 B$=C$
550 RETURN
Total: 444 pas

```

CHAPITRE 6

DOCUMENTATION

6-1. TABLEAU DE MESSAGES D'ERREUR

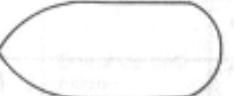
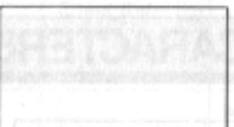
Code erreur	Signification	Cause	Mesure corrective
1	Dépassement de mémoire ou de niveaux d'opérateurs.	- Le nombre de pas est insuffisant. Un programme ne peut pas être entré. - Dépassement de niveaux d'opérateurs.	- Effacer les programmes inutiles ou réduire le nombre de mémoires. - Diviser les formules et les rendre plus simples.
2	Erreur de syntaxe	- Erreur de format de commande dans un programme, etc. - Le format de la partie gauche diffère de celui de la partie droite d'une instruction d'affectation, etc.	Corriger l'erreur dans le programme entré, etc.
3	Erreur mathématique	- Le résultat de calcul d'une expression numérique dépasse $\pm 1 \times 10^{100}$. - Argument hors du domaine de définition d'une fonction numérique. - Le résultat est indéterminé ou impossible.	- Corriger la formule de calcul ou les données. - Vérifier les données.
4	Erreur de définition	- Pas de numéro de ligne dans une instruction GOTO ou GOSUB. - Une instruction READ ou READ# est exécutée alors qu'il n'y a pas de donnée à lire.	- Désigner un numéro de ligne. - Contrôler la relation entre les instructions READ et DATA. Créer des données à affecter à la variable dans l'instruction.
5	Erreur d'argument	Pour une commande ou une fonction qui requiert un argument, celui-ci est hors du domaine de définition.	Corriger l'argument.
6	Erreur de variable	- On a essayé d'utiliser une mémoire qui n'a pas été étendue. - On a essayé d'utiliser la même mémoire à la fois pour une variable numérique et une variable alphanumérique.	- Étendre la mémoire de manière appropriée. - Ne pas utiliser la même mémoire à la fois pour une variable numérique et pour une variable alphanumérique.
7	Erreur d'emboîtement	- On exécute une instruction RETURN alors que l'on n'est pas dans un sous-programme. - Une instruction NEXT ne correspond à aucune instruction FOR. - Le nombre de niveaux d'emboîtement des sous-programmes dépasse 8. - Le nombre de niveaux d'emboîtement des boucles FOR-NEXT dépasse 4.	- Enlever les instructions RETURN ou NEXT inutiles. - Corriger le nombre de niveaux d'emboîtement des sous-programmes ou des boucles FOR-NEXT.

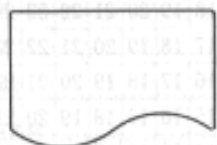
Code erreur	Signification	Cause	Mesure corrective
8	Erreur de mot de passe	Lorsqu'un mot de passe est spécifié, a) un autre mot de passe est spécifié b) une commande telle que LIST ou NEW qui ne peut pas être utilisée, est exécutée.	Supprimer le mot de passe.
9	Erreur d'option	- Les commandes SAVE ou PUT sont exécutées sans qu'un magnétophone soit branché. - Le signal d'entrée utilisé par les commandes LOAD ou GET ne peut pas être lu. - La batterie de l'imprimante est déchargée. - Il y a un bourrage de papier dans l'imprimante.	- Brancher un magnétophone. - Baisser le volume du magnétophone. - Placer la commande de tonalité du magnétophone en position moyenne. - Changer la bande. - Nettoyer les têtes du magnétophone. - Charger la batterie. - Débarrasser le papier.

6-2. TABLEAU DES CODES DE CARACTERES

	Espace	+	-	*	/	↑	↓	"	#	\$	>	≥	=	≤	<	≠
Nombres	0	1	2	3	4	5	6	7	8	9	.	π	)	(	E	E
Majuscules	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Minuscules	Q	R	S	T	U	V	W	X	Y	Z						
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
	q	r	s	t	u	v	w	x	y	z						
Symboles	?	,	:	:												
Symboles graphiques	○	Σ	◊	△	@	×	÷	♠	←	♥	♦	♣	μ	Ω	↓	→
	%	¥	□	[	&	-	'	.	]	■	\					

Les caractères et symboles de la table ci-dessus sont placés dans l'ordre, l'espace étant le plus petit et " \ " le plus grand. (On peut afficher " \ " en appuyant sur **SHIFT** **P**.)

Symbole	Signification
	Point d'entrée ou de sortie (Début, retour, fin, etc.)
	Entrée de données à partir du clavier
	Sortie (résultat)
	Traitement général
	Traitement dans un sous-programme
	Test (condition)

Symbole	Signification
	Sortie sur imprimante
	Ligne de liaison
	Transfert ou point de continuation

6-4. TABLE DES ELEMENTS DE TABLEAU

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
B	-	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
C	-	-	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
D	-	-	-	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
E	-	-	-	-	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
F	-	-	-	-	-	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
G	-	-	-	-	-	-	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
H	-	-	-	-	-	-	-	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
I	-	-	-	-	-	-	-	-	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
J	-	-	-	-	-	-	-	-	-	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
K	-	-	-	-	-	-	-	-	-	-	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L	-	-	-	-	-	-	-	-	-	-	-	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
M	-	-	-	-	-	-	-	-	-	-	-	-	0	1	2	3	4	5	6	7	8	9	10	11	12	13
N	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	2	3	4	5	6	7	8	9	10	11	12
O	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	2	3	4	5	6	7	8	9	10	11
P	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	2	3	4	5	6	7	8	9	10
Q	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	2	3	4	5	6	7	8	9
R	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	2	3	4	5	6	7	8
S	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	2	3	4	5	6	7
T	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	2	3	4	5	6
U	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	2	3	4	5
V	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	2	3	4
W	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	2	3
X	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	2
Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1
Z	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
	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

Cette table indique l'équivalence entre les variables.

Exemple) $H(0) \sim H(9) \rightarrow H \sim Q$

INDEX DES COMMANDES/FONCTIONS

ABS	38, 152	MIDS	98, 146
ACS	35, 150	MODE	143
ASN	35, 150	NEW(ALL)	119
ATN	35, 150	NEW#	156
BEEP	90, 141	ON-GOSUB	96, 136
CLEAR	125	ON-GOTO	96, 132
COS	36, 149	PASS	122
CSR	100, 130	PRINT	46, 66, 129
DATA	92, 137	PUT	109, 140
DEFM	89, 142	RAN#	38, 154
DEG	39, 154	READ	92, 138
DM\$	40, 155	READ#	158
END	126	REM	127
EXP	37, 151	RESTORE	92, 139
FOR-TO-NEXT	75, 134	RESTORE#	159
FRAC	38, 153	RETURN	79, 136
GET	109, 140	RND	38, 153
GOSUB	79, 135	RUN	55, 120
GOTO	64, 131	SAVE(ALL)	106, 123
IF-THEN	68, 133	SAVE#	108, 157
INPUT	46, 127	SET	39, 144
INT	38, 152	SGN	38, 152
KEY\$	100, 128	SIN	36, 149
LEN	98, 145	SQR	37, 151
LET	44, 126	STOP	61, 126
LIST	121	STR\$	98, 148
LIST#	156	TAN	36, 149
LN	37, 150	VAL	98, 147
LOAD(ALL)	106, 124	VERIFY	125
LOAD#	108, 157	WRITE#	161
LOG	37, 150		

- **Type**
OP545
- **Fonctions élémentaires de calcul**
Additions, soustractions, multiplications et divisions avec nombres négatifs, exposants et parenthèses (avec fonction de jugement d'ordre de priorité (vraie logique algébrique)).
- **Fonctions incorporées**
Fonctions trigonométriques/trigonométriques inverses (unités angulaires — degrés/radians/grades), fonctions logarithmiques/exponentielles, racines carrées, puissances, conversion en entier, suppression de partie entière, valeur absolue, désignation du nombre de chiffres significatifs, désignation du nombre de décimales, nombres aléatoires, π , conversion décimal $\leftrightarrow$ sexagésimal.
- **Commandes**
INPUT, PRINT, GOTO, ON-GOTO, FOR-NEXT, IF-THEN, GOSUB, ON-GOSUB, RETURN, READ, DATA, RESTORE, STOP, END, REM, LET, BEEP, PASS, RUN, LIST, LIST ALL, MODE, SET, CLEAR, NEW, NEW ALL, DEFN, SAVE, SAVE ALL, LOAD, LOAD ALL, PUT, GET, VERIFY, NEW #, LIST #, LOAD #, SAVE #, READ #, WRITE #, RESTORE #.
- **Fonctions de programme**
KEY\$, CSR, LEN, MID\$, VAL, STR\$
- **Plage de calcul**
 $\pm 1 \times 10^{-99}$ à $\pm 9,999999999 \times 10^{99}$ et 0 (Les calculs internes sont faits avec une mantisse de 12 chiffres)
- **Système de programmation**
Méorisé
- **Nombre de pas**
Maximum de 544 pas (maximum de 1.568 pas quand le module RAM optionnel est utilisé)
- **Capacité en programmes**
Maximum de 10 programmes (dans les zones P0 à P9)
- **Nombre de variables**
26 de base, étendable à 94 (maximum de 222 variables quand le module RAM optionnel est utilisé) et une exclusive variable à caractères (\$).
- **Emboîtement**
Sous-programmes — 8 niveaux
Boucles FOR-NEXT — 4 niveaux
Valeurs numériques — 6 niveaux
Opérateurs — 12 niveaux
- **Affichage**
10 positions de mantisse (en incluant le signe moins) ou 8 positions de mantisse (7 positions pour un nombre négatif) et 2 positions d'exposant.
- **Éléments d'affichage**
Affichage de 12 digits par matrice par points (cristaux liquides)

- **Composants principaux**
C-MOS VLSI et autres
- **Alimentation**
2 piles au lithium (CR2032)
- **Consommation**
0,03 W au maximum.
- **Longévité des piles (Utilisation continue)**
Unité centrale seule — approximativement 140 heures
Avec des appareils optionnels branchés — approximativement 70 heures
- **Arrêt automatique**
L'alimentation est automatiquement coupée environ 6 minutes après la dernière opération.
- **Plage de température ambiante**
0°C à 40°C (32°F à 104°F)
- **Dimensions**
9,8 mmH x 165 mmL x 71 mmP
- **Poids**
119g, piles comprises

13672 AUBAGNE (Marseille) (42) 03.60.40

Z.I. Saint Mitre B.P. 62

04-05-06-13-20-30-34-83-84-Monaco *

33300 BORDEAUX (56) 39.77.88

Avenue de Suffren, Z.I. Alfred-Daney

16-17-19-24-33-40-64-87 *

92142 CLAMART (siège social) (11) 630.21.42

10, avenue Réaumur B.P. 209

10-14-18-27-28-45-50-51-58-60-61-75-76-77

78-89-91-92-93-94-95 *

59006 LILLE (20) 07.33.55

1, rue Deleplanque

02-08-59-62-80 *

69362 LYON (78) 58.32.21

60, route de Vienne

01-03-07-23-26-38-39-42-43-63-69-71-73-74 *

54500 VANDOEUVRE (Nancy) (8) 351.61.61

Centre Cial < Les Nations >, bd de l'Europe

21-25-52-54-55-57-67-68-70-88-90 *

44880 SAUTRON (Nantes) (40) 63.32.66

B.P. 7, Route de Vannes

22-29-35-36-37-41-44-49-53-58-72-79-85-86 *

31200 TOULOUSE (61) 47.90.24

142, chemin du Sang-de-Serp

09-11-12-15-31-32-46-47-48-65-66-81-82 Andorre *

* Départements rattachés à la succursale.

CONDITIONS DE GARANTIE

Ce modèle est garanti 6 mois

La garantie ne couvre pas la détérioration provoquée par l'usage abusif de ce matériel, la mauvaise manipulation, les chutes ou accidents.

La durée de garantie ne peut être prolongée, ni renouvelée et ne couvre pas les accidents de transport. Pendant cette durée, les réparations sont effectuées gratuitement dans nos ateliers, seuls les frais de port sont à la charge du client.



OLYMPIA FRANCE S.A.

BON DE GARANTIE

à envoyer avec le matériel en cas de réparation

Modèle _____

No matricule _____

Date de la vente _____

Cachet du revendeur:

Emballage

Sous-programmes -- 8 niveaux

Boucles FOR-NEXT -- 4 niveaux

Valeurs numériques -- 6 niveaux

Opérateurs -- 12 niveaux

(A remplir en majuscules S.V.P.)

Société _____

Nom _____

Service/profession _____

No _____ Rue _____

Code postal _____ Ville _____

N.B. Si possible, veuillez indiquer la nature de la panne.