Ordinateur Personnel

PB-410/FX-720P/FX-820P

MANUEL DE L'UTILISATEUR

CASIO®

Ordinateur Personnel

PB-410/FX-720P/FX-820P

CASIO®

INTRODUCTION

Ce manuel donne l'explication du fonctionnement de l'ordinateur afin que les programmeurs BASIC débutants aussi bien que les utilisateurs qui ont une complète connaissance du BASIC et qui ont l'intention de l'utiliser pleinement, puissent immédiatement comprendre et utiliser l'ordinateur aisément.

Les utilisateurs pour lesquels la programmation BASIC est une chose nouvelle, doivent lire ce manuel à partir du chapitre 1 et dans l'ordre des chapitres pour maîtriser la programmation. Spécialement au chapitre 3, l'explication de la préparation des programmes et des commandes doit être lue avec attention. L'explication du déroulement d'un programme est donnée au chapitre 3, voir le chapitre 4 "Référence de commandes" pour le format des commandes et une explication détaillée.

Les utilisateurs qui ont une connaissance du BASIC utiliseront l'ordinateur après avoir lu le chapitre 4 "Référence de commandes" et maîtrisé les opérations de base expliquées aux chapitres 1 et 2.

Les utilisateurs qui ont l'intention d'entrer des programmes et de les utiliser immédiatement, peuvent utiliser les programmes du chapitre 5 "Bibliothèque de programmes".

Cette explication est prévue pour les PB-410, FX-720P et FX-820P. Les points différents sont que les FX-720P et FX-820P ont une touche de fonction (touche [F] bleue) et que le FX-820P est muni d'une imprimante à caractères intégrée (pour plus de détails, voir page 12).

AVANT L'UTILISATION

Cet ordinateur d'une technologie électronique de haut niveau, vous a été délivré après avoir subi un processus de tests et un contrôle de qualité sévères chez CASIO. Pour assurer une longue durée de vie à votre ordinateur, veuillez observer les précautions suivantes.

■ Précautions d'utilisation

- Etant donné que cet ordinateur est constitué d'éléments électroniques de précision, ne pas le démonter. Egalement ne pas lui faire subir de choc en le lançant ou en le laissant tomber et ne pas l'exposer à des changements rapides de température. De plus, ne pas le ranger dans un endroit très chaud, très humide ou poussiéreux. Lorsque l'ordinateur est utilisé par basse température, il arrive parfois que l'affichage soit long à s'effectuer ou ne fonctionne pas. Toutefois, lorsque des conditions normales de température sont retrouvées, le fonctionnement de l'ordinateur redevient normal.
- Des soins particuliers doivent être pris afin d'éviter que l'ordinateur ne soit tordu. Par exemple, ne pas le mettre dans une poche révolver.
- Comme équipements optionnels, l'interface cassette FA-3 pour les PB-410, FX-720P et FX-820P et l'imprimante à caractères pour les PB-410 et FX-720P sont prévues. Veuillez ne pas brancher d'appareils autres que ceux-ci au connecteur.
- Etant donné que " – " est affiché pendant l'utilisation de l'ordinateur dans laquelle les manipulations de touches ne sont pas valables excepté pour certaines, toujours contrôler l'affichage avant d'appuyer sur une touche.
- Bien qu'il arrive parfois que l'affichage devienne faible quand le vibreur retentit ceci n'est pas un mauvais fonctionnement de l'appareil. Néanmoins, si l'affichage devient très faible, changer les piles dès que possible.
- Changer les piles de l'ordinateur et de la carte RAM tous les deux ans, même si l'ordinateur n'est pas utilisé. Ne pas laisser de piles épuisées à l'intérieur de l'appareil car des fuites pourraient entraîner des pannes.
- Etant donné que la RAM (mémoire vive) n'est pas incorporée à l'ordinateur, veiller à y insérer une carte RAM.
- Lors du changement des piles de l'ordinateur, il arrive parfois que le contenu de la carte RAM soit modifié. De ce fait, remplacer les piles après avoir enlevé la carte RAM de l'ordinateur.
- Si le bouton de verrouillage de la carte RAM est tourné vers la gauche, l'alimentation est coupée et la carte ne peut donc pas fonctionner. Par conséquent, placer le bouton de verrouillage sur la position LOCK pour pouvoir utiliser l'appareil.

- Toujours laisser le capuchon de protection de la partie connecteur en place lorsque l'ordinateur est utilisé seul.
- Si une forte électricité statique est appliquée à l'ordinateur ou à la carte RAM, il arrive parfois que le contenu de la mémoire soit modifié ou que les manipulations de touches soient impossibles. Si cela se produit, enlever puis remettre en place les piles.
- Brancher toujours les équipements optionnels après avoir coupé l'alimentation de l'ordinateur.
- Pour nettoyer l'ordinateur, ne pas utiliser de liquides volatils tels que benzine ou diluant, mais un chiffon doux et sec ou humecté avec un détergent neutre.
- Ne pas couper l'alimentation pendant l'exécution d'un programme ou pendant que l'ordinateur effectue des opérations.
- Les programmes d'une carte RAM préparés sur le PB-410, FX-720P ou FX-820P ne peuvent pas être exécutés sur d'autres ordinateurs à carte RAM.
- En cas de panne, contacter le magasin où l'ordinateur a été acheté ou le concessionnaire le plus proche.
- Avant de demander une réparation, relire ce manuel, contrôler l'alimentation, contrôler si le programme ne présente pas d'erreurs de logique, etc.

TABLE DES MATIERES

CHAPITRE 5 BIBLIOTHEQUE DE PROGRAMMES

CHAPITRE 6 DOCUMENTATION

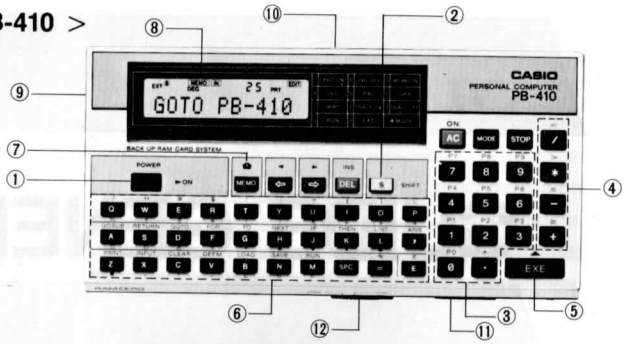
CHAPITRE 1

GUIDE GENERAL

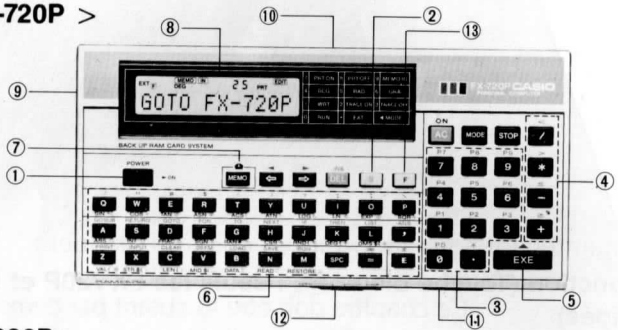
Ce chapitre doit être lu autant par ceux qui n'ont jamais utilisé d'ordinateur que par ceux qui y sont déjà habitués.

1-1. NOMENCLATURE ET OPERATION

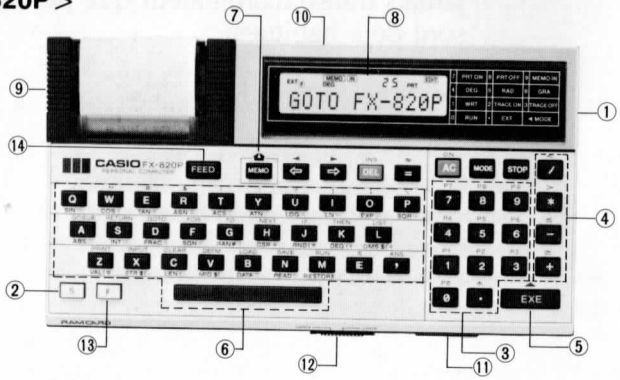
< PB-410 >



< FX-720P >



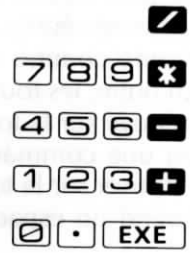
<FX-820P>



① Interrupteur d'alimentation
② Touche de déplacement
③ Touches numériques et touche décimal virgule
④ Touches de calcul
⑤ Touche d'exécution
⑥ Touches alphabétiques, touche d'espacement
⑦ Touche de mémorisation
⑧ Fenêtre d'affichage
⑨ Commande de contraste d'affichage
⑩ Partie connecteur
⑪ Fente d'insertion de la carte RAM
⑫ Bouton de verrouillage de la carte RAM
⑬ Touche de fonction (FX-720P/FX-820P)
⑭ Touche d'avance du papier (FX-820P)

Noter que les FX-720P et FX-820P ont une touche [F] (fonction) qui est utilisée conjointement avec les touches alphabétiques quand des fonctions sont entrées, alors que le PB-410 n'en est pas muni. Noter aussi que le FX-820P a une imprimante à caractères intégrée.
Comparé à des calculatrices ordinaires, cet ordinateur possède beaucoup plus de touches. La fonction de chaque touche est maintenant expliquée.

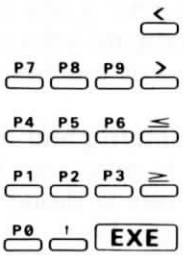
- **Interrupteur d'alimentation**
  La mise sous tension est effectuée lorsque cet interrupteur est tourné vers la droite et l'alimentation est coupée lorsqu'il est tourné vers la gauche.
- **Touche de déplacement (Touche rouge [S])**
  Si on appuie sur cette touche, le mode déplacement est sélectionné (" [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 [SHIFT] dans ce manuel.)
- **Touche de fonction (Touche bleue [F] : seuls les FX-720P et FX-820P en sont équipés)**
  Si on appuie sur cette touche, le mode fonction est sélectionné (" [F] " est affiché) et la fonction écrite au-dessous de chaque touche peut être affichée. Lorsque l'on appuie de nouveau sur cette touche, le mode fonction est annulé et " [F] " disparaît. (Pour distinguer cette touche de la touche alphabétique [F] , elle sera écrite à partir de maintenant sous la forme [FUNC] dans ce manuel.)
- **Touches numériques, touche décimal virgule, 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 [×] (multiplication) et [÷] (division) sont différentes et il n'y a pas de touche [=] alors qu'il y a une touche [EXE] (exécution). Celà vient du fait qu'un ordinateur utilise un * (astérisque) et / (barre de fraction) respectivement pour X et ÷ 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 × 4 ÷ 3 + 7 − 5 = alors qu'en utilisant cet ordinateur elle devient 12 * 4 / 3 + 7 − 5 EXE .

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

Après avoir appuyé sur la touche SHIFT , 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 ($\geqq, \leqq, >, <$).



- **Touches alphabétiques, touche d'espacement**

<PB-410/FX-720P>

Q W E R T Y U I O P
A S D F G H J K L "
Z X C V B N M SPC = E

<FX-820P>

Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M E "
[ ]

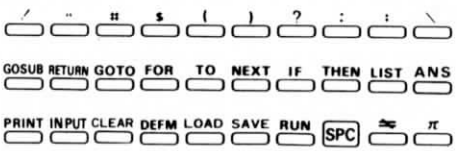
A l'aide de ces touches, des commandes peuvent être entrées ou des programmes écrits. Chacune des 26 touches alphabétiques de A à Z fonctionne comme une mémoire (pour les emplacements de stockage).

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

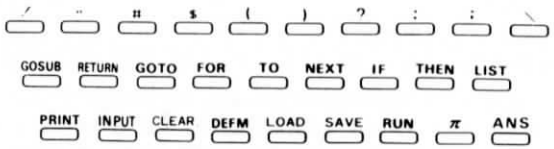
Appuyer sur la touche d'espacement (PB-410/FX-720P: SPC , FX-820P: [ ]) quand un espace est nécessaire.

Exemple) [SHIFT][A] → GOSUB 、 [SHIFT][U] → ?

<PB-410/FX-720P>



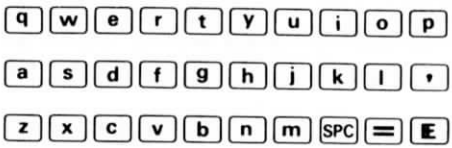
<FX-820P>



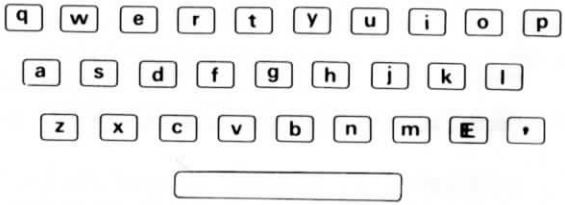
De plus, les touches alphabétiques ont une autre utilisation en mode extension (lorsque l'on a appuyé sur [•] après avoir appuyé sur la touche [MODE], "EXT" est affiché). Lorsque l'on appuie dessus directement, des lettres minuscules sont affichées et lorsque l'on appuie dessus après avoir appuyé sur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

Fonctions en mode extension:

<PB-410/FX-720P>

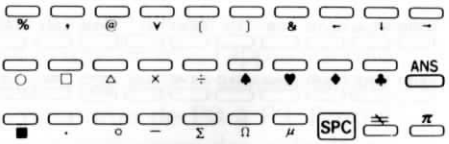


<FX-820P>

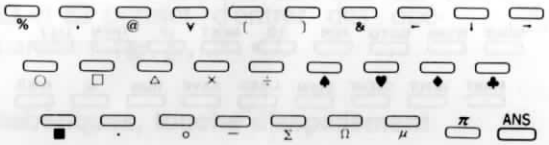


Fonctions fournies lorsque l'on a appuyé sur une touche alphabétique après avoir appuyé sur la touche [SHIFT] en mode extension:

<PB-410/FX-720P>



<FX-820P>

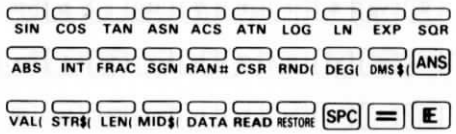


Pour annuler le mode extension, appuyer de nouveau sur [MODE] [•].

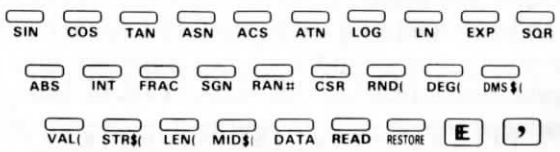
Les FX-720P et FX-820P sont munis d'une touche [FUNC]. Lorsque l'on appuie sur une touche après avoir appuyé sur la touche [FUNC], une des fonctions suivantes est affichée.

Exemple) [FUNC] [Q] → SIN

<PB-410/FX-720P>



<FX-820P>



Des lettres majuscules sont affichées en mode extension.

- **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 74).
  En outre, quand on a appuyé sur cette touche après avoir appuyé sur la touche [SHIFT], un symbole ≠ (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][EXP][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][EXP][(-)][9] pour obtenir $7{,}41 \times 10^{-9}$.
  Lorsque l'on a appuyé sur cette touche après avoir appuyé sur la touche [SHIFT], 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 [SHIFT], le résultat du calcul manuel ou effectué par programme immédiatement avant est affiché.

- **Touche mode (MODE)**
  Cette touche est utilisée conjointement avec les touches [.] et de [0] à [9] pour spécifier l'état de l'ordinateur ou l'unité d'angle.

[MODE][.] ..... "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.

[MODE][0] ..... "RUN" est affiché pour permettre l'exécution de calculs manuels ou par programme.

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

[MODE][2] ..... "TR" est affiché pour permettre l'exécution d'une analyse. (Pour plus de détails, voir page 69.)

[MODE][3] ..... Lorsque "TR" est affiché, le mode analyse est annulé et "TR" disparaît.

[MODE][4] ..... "DEG" est affiché pour indiquer que le degré est spécifié comme unité d'angle.

[MODE][5] ..... "RAD" est affiché pour indiquer que le radian est spécifié comme unité d'angle.

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

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

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

MODE 9 ..... " MEMO IN " est affiché pour indiquer le mode entrée pour la fonction banque de données (voir page 38). Pour annuler ce mode, appuyer sur MODE 0 .

- **Touche de mémorisation (MEMO)**
  Employée pour utiliser la fonction banque de données. Employée également pour effectuer des rappels séquentiels en mode RUN (appuyer sur MODE 0) ou en mode entrée (appuyer sur MODE 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 continûment 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 21). Quand un programme est en train d'être exécuté, en appuyant sur cette touche, l'exécution est suspendue.

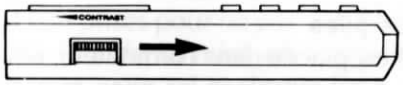
- **Touche suppression/insertion (INS DEL)**
  Cette touche est utilisée pour supprimer le caractère au-dessous duquel clignote le curseur. Après la suppression, les caractères placés à droite du curseur se déplacent vers la gauche. Lorsque l'on appuie sur cette touche après avoir appuyé sur la touche SHIFT , 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 .

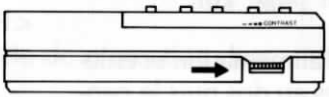
- **Touche d'avance de papier (FEED : uniquement sur le FX-820P)**
  Appuyer sur cette touche pour faire défiler le papier du rouleau.

- **Commande de contraste d'affichage**
  Lorsque l'affichage est sombre ou pâle du à l'état des piles ou à l'angle sous lequel on le voit, le régler au contraste désiré à l'aide de la commande située sur le côté droit de l'ordinateur.

  <**PB-410/FX-720P**>



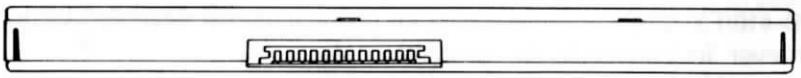
  <**FX-820P**>



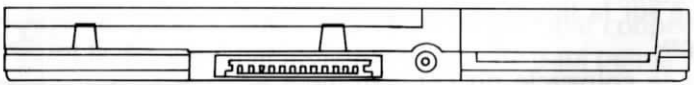
  L'affichage devient plus sombre lorsque la commande est tournée dans la direction de la flèche et plus pâle dans le sens contraire. Si l'affichage reste pâle quand cette commande est placée sur la position la plus sombre, les piles sont faibles et doivent être remplacées.

- **Partie connecteur**
  Quand le stockage de programmes sur une cassette est nécessaire, la FA-3 est connectée; quand l'impression est nécessaire avec le PB-410 ou le FX-720P, la FP-12S est connectée.

  <**PB-410/FX-720P**>



  <**FX-820P**>



  Le PB-410 ou le FX-720P peut être connecté aux FP-12S et FA-3; le FX-820P peut être connecté à la FA-3.
  Ne pas brancher d'autre appareils que la FP-12S et la FA-3 à ce connecteur. Lorsque ces appareils optionnels ne sont pas branchés, mettre toujours en place le couvercle de protection du connecteur.

1-2. ALIMENTATION DE L'UNITE CENTRALE

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 19), ceci provient de la faiblesse des piles qui doivent donc être remplacées dès que possible. Toujours remplacer les deux piles en même temps.

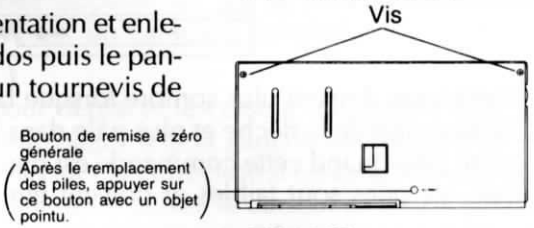
* Remplacer les piles tous les deux ans même si elles n'ont pas été utilisées car une fuite pourrait se produire.

■ Remplacement des piles

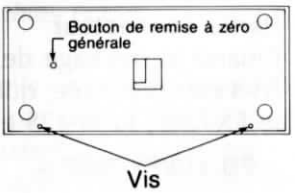
Lorsque la carte RAM est placée dans l'ordinateur, remplacer les piles après l'avoir enlevée. Après avoir remplacé les piles, remettre la carte RAM dans sa fente d'insertion (voir page 26).

① Ouvrir l'interrupteur d'alimentation et enlever les deux vis situées au dos puis le panneau de celui-ci. (Utiliser un tournevis de précision)

<PB-410/FX-720P>



<FX-820P>



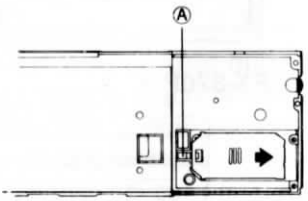
② **PB-410/FX-720P:**

Enlever le couvercle de protection des piles en le faisant glisser dans le sens de la flèche tout en appuyant sur Ⓐ comme indiqué sur la figure de droite.

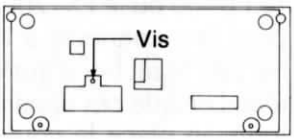
FX-820P:

Ouvrir le couvercle du compartiment à pile en enlevant la vis à l'aide d'un tournevis.

<PB-410/FX-720P>



<FX-820P>



③ Enlever les deux piles usagées (Elles peuvent être facilement enlevées en tapant légèrement sur le compartiment à piles tout en l'orientant vers le bas).

④ Essuyer la surface de contact des nouvelles piles avec un chiffon sec et les insérer en plaçant la borne plus vers le haut. Des précautions doivent être prises pour ne pas inverser les polarités lors de la mise en place.

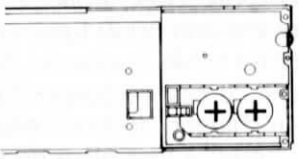
⑤ **PB-410/FX-720P:**
Remonter le couvercle de protection des piles.

FX-820P:
Remettre le couvercle du compartiment à pile en place. Visser soigneusement.

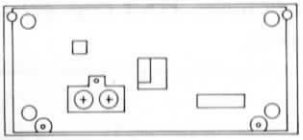
⑥ Replacer le panneau du dos. Visser soigneusement.

* Ne pas jeter les piles usagées dans le feu; ceci pourrait provoquer une dangereuse explosion.

<PB-410/FX-720P>



<FX-820P>



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 .

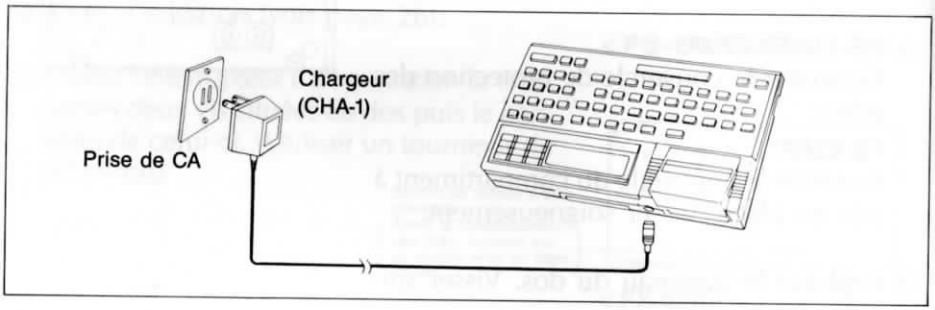
* Bien que le contenu de la mémoire ne soit pas effacé lorsque l'alimentation est coupée, les spécifications d'angle et de mode ("RAD", "WRT", "TR", "PRT", etc.) sont annulées.

1-3. POUR LES UTILISATEURS DU FX-820P

■ **Comment charger l'accumulateur de l'imprimante**

L'imprimante est alimentée par un accumulateur rechargeable au Ni-Cd incorporé. Avec un accumulateur chargé aux maximum, elle peut imprimer environ 3000 lignes en continu. Quand la puissance de l'accumulateur est faible, la vitesse d'impression diminue et l'impression devient floue. Dans ce cas, recharger l'accumulateur.

Pour charger l'accumulateur, brancher le chargeur convenable (100, 117, 220 ou 240V) à une prise de CA et le cordon au jack de l'appareil. Tant que le chargeur est branché, l'accumulateur est chargé, sauf quand l'imprimante est activée. Pour charger l'accumulateur au maximum, il faut environ 15 heures.



Vous pouvez utiliser l'appareil après avoir chargé l'accumulateur pendant 1 ou 2 heures. Toutefois, de courtes périodes de charge diminueront la durée de fonctionnement de l'accumulateur. Il est recommandé de charger l'accumulateur au maximum avant d'utiliser l'appareil.

- Pendant la charge, s'assurer que le commutateur d'alimentation de l'ordinateur est ouvert.
- L'emploi d'un chargeur autre que le chargeur Casio fourni avec l'appareil peut se traduire par un endommagement de votre appareil.
- Il est tout à fait normal que le chargeur soit chaud au toucher quand il est branché à une prise de CA. Une fois que l'accumulateur est chargé au maximum, débrancher le chargeur de la prise de CA.
- Si l'accumulateur ne tient pas la charge ou semble se décharger très rapidement, il peut être défectueux. Contacteur le magasin où on a acheté l'appareil ou le distributeur le plus proche pour en commander un autre.

■ Comment mettre le rouleau de papier en place

1) Fermer le commutateur d'alimentation.
2) Ouvrir le couvercle de l'imprimante comme illustré. (Fig. 1)
3) Tenir le rouleau de papier avec l'extrémité libre du papier en bas.
4) Insérer l'extrémité libre du papier dans la fente d'alimentation (Fig. 2) et appuyer sur la touche [FEED] jusqu'à ce que le papier sorte par l'autre côté de l'imprimante. (Fig. 3)
5) Mettre le rouleau de papier en place dans le compartiment et remonter le couvercle de l'imprimante.

Fig. 1

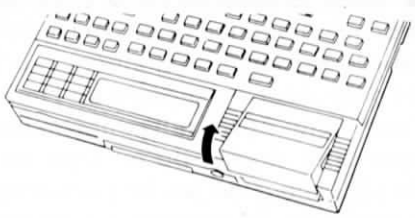


Fig. 2

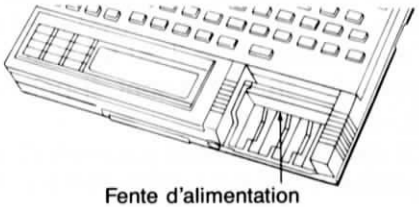
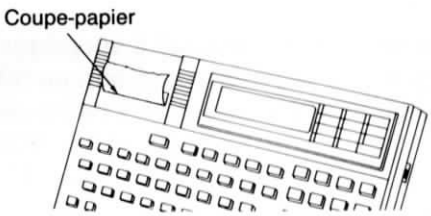


Fig. 3



ATTENTION

Les mauvais emplois suivants du papier d'impression électrothermique peuvent entraîner une impression illisible et/ou une décoloration dudit papier.

1. Stockage dans un endroit extrêmement chaud et/ou humide.
2. Exposition en plein soleil.
3. Revêtement avec de la colle chimique.
4. Stockage sur des matières plastiques.
5. Eclaboussement ou rayage.

Note:

Etant donné que cet appareil emploie un système d'impression électrothermique spécial, toujours utiliser le papier d'impression électrothermique spécifié (taille: 38 mmL x 16 mmϕ).

1-4. CARTE DE MEMOIRE VIVE (CARTE RAM)

■ **Caractéristiques de la carte RAM**

Bien qu'un ordinateur de poche ordinaire possède une mémoire incorporée pour le stockage de données ou de programmes, la mémoire interne de cet ordinateur est séparée de l'unité centrale et se présente sous la forme d'une "carte RAM" qui peut être librement insérée ou retirée. C'est très pratique pour stocker ou remettre en place des programmes.

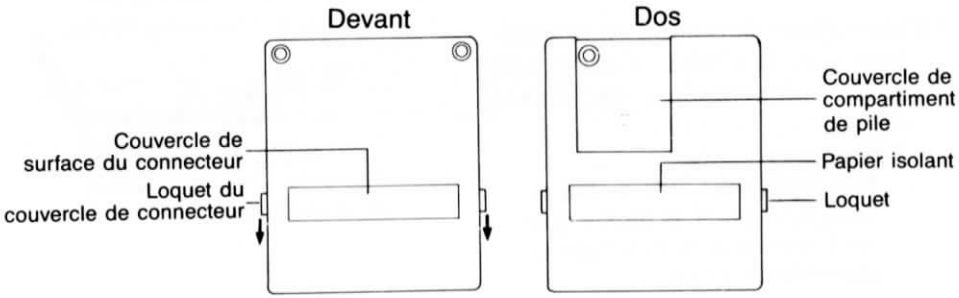
Un ordinateur de poche conventionnel utilise des cassettes pour stocker et remplacer des données ou des programmes. Cette gêne peut être éliminée en utilisant une carte RAM ce qui permet de changer rapidement et facilement de cartouche et de traiter ainsi d'autres données et d'autres programmes.

Le contenu de la carte RAM n'est pas effacé lorsque celle-ci est retirée de l'unité centrale car il est protégé par une pile incorporée. Deux différents types de carte RAM sont disponibles, la carte RC-4 (4K octets) et la carte RC-2 (2K octets).

* Cet ordinateur n'étant pas muni d'une mémoire vive incorporée, il ne peut être utilisé si une carte RAM n'est pas insérée.

■ **Précautions à prendre lors de la manipulation**

Bien que deux cartes RAM différentes, RC-4 (4K octets) et RC-2 (2K octets), soient disponibles pour cet ordinateur, leurs maniements sont identiques.



- Ne pas toucher la surface du connecteur.
  Tirer les loquets situés à gauche et à droite dans la direction de la flèche tout en les maintenant pour dégager la surface du connecteur. Si celle-ci a été touchée avec les doigts ou par un corps métallique, il arrive parfois que la carte RAM ne puisse pas être utilisée. Après avoir retiré celle-ci de l'unité centrale, placer le couvercle de protection de la surface du connecteur.
- Si une forte électricité statique est appliquée à la carte RAM, il arrive parfois que sont contenu soit modifié ou que les entrées par le clavier soient impossibles. Si cela se produit, retirer puis replacer la pile de la carte RAM. (Dans ce cas, son contenu est effacé).
- Ne pas démonter la carte RAM ni la tordre ou la courber.
- Lorsque la carte RAM est retirée de l'unité centrale, la placer dans sa boîte et la ranger dans un endroit qui ne soit ni poussiéreux ni exposé directement aux rayons du soleil.
- Une pile au lithium est placée dans la carte RAM pour protéger la mémoire. Si elle est enlevée, le contenu de la carte est effacé. Avant de changer la pile, le contenu de la carte doit être stocké sur une cassette et lorsque le changement a été effectué, recharger ce contenu dans la carte RAM (voir page 111).
- Ne pas enlever le papier isolant car il protège la surface du connecteur de l'unité centrale lorsque la carte est insérée à l'envers par erreur.
- Veiller à ce qu'une pile soit placée dans la carte RAM.
- Les cartes RAM ne doivent pas être utilisées sur d'autres appareils que les ordinateurs CASIO à carte RAM.

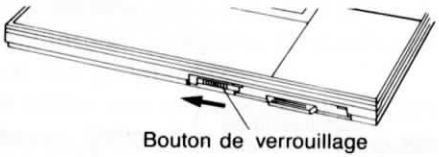
■ Mise en place de la carte RAM

①Couper l'alimentation de l'unité centrale.

②Faire glisser le bouton de verrouillage vers la droite. (Ceci coupe aussi l'alimentation de l'unité centrale.)

Remarque)

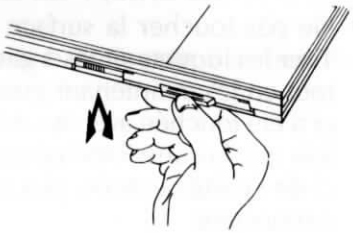
Si le support de carte est tiré de force sans que l'on ait fait glisser le bouton de verrouillage, ce dernier sera cassé.



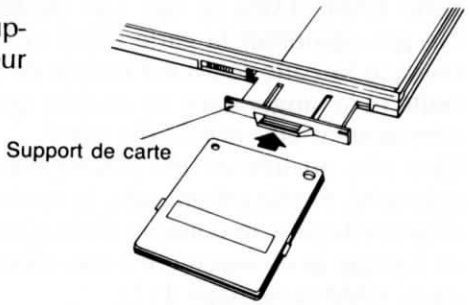
③Tirer légèrement sur le support de carte en appuyant doucement vers le bas sur la saillie.

Remarque)

Si l'on ne peut tirer le support que jusqu'à la moitié, forcer pourrait le casser.

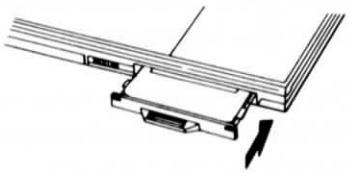


④Insérer la carte RAM dans son support avec la surface du connecteur orientée vers le haut.

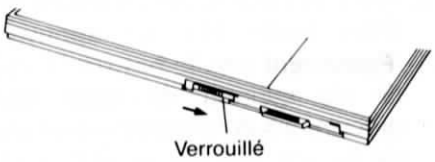


⑤ Appuyer vers le bas sur la saillie du support de carte de telle manière que la carte RAM soit horizontale et l'insérer complètement dans le support.

⑥ Insérer le support de carte en le poussant légèrement vers le haut dans la direction de la flèche jusqu'à ce qu'il émette un déclic et s'arrête complètement.



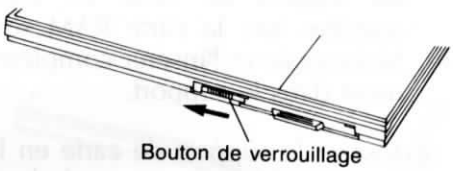
⑦ Faire glisser le bouton de verrouillage vers la droite.



Remarque) Si le bouton de verrouillage n'est pas verrouillé lorsque l'interrupteur d'alimentation est fermé, la mise sous tension ne s'effectura pas.

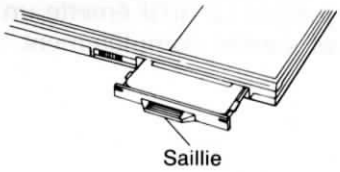
■ Dépose de la carte RAM

①Couper l'alimentation de l'unité centrale.

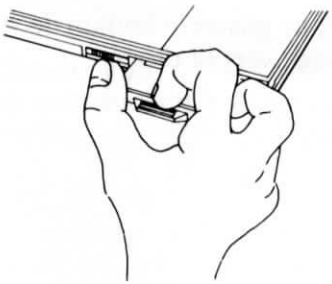


②Faire glisser le bouton de verrouillage vers la gauche.

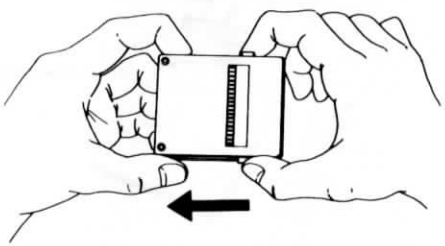
③Sortir le support de carte en le tirant légèrement tout en appuyant vers le bas sur sa saillie.



④Sortir la carte RAM en tenant les deux bords tout en appuyant légèrement vers le bas sur la saillie du support de carte. Des précautions doivent être prises afin de ne pas toucher la surface du connecteur.



⑤La surface du connecteur de la carte enlevée étant dégagée, fermer son couvercle en le faisant glisser.



Remarque) Quand une carte RAM n'est pas utilisée, la ranger dans sa boîte.

Lorsqu'une autre carte RAM est mise en place en remplacement d'une ancienne, veuillez vous référer au paragraphe "Mise en place de la carte RAM" alinéas ④ à ⑦.

■ Changement de la pile de carte RAM

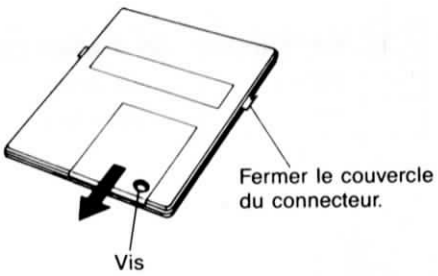
La protection de la mémoire d'une carte RAM est assurée par une pile au lithium (CR2016). Si une carte RAM est enlevée de l'ordinateur et rangée, la longévité de la pile d'une carte RC-4 est d'environ 1 an alors que celle d'une carte RC-2 est d'environ 2 ans. La longévité de la pile d'une carte RC-4 est prolongée lorsque celle-ci est utilisée dans l'ordinateur car cette pile est secondée par l'alimentation principale. Néanmoins, comme une fuite pourrait se produire si la pile est utilisée depuis plus de deux ans, cette dernière doit être changée avant ce terme.

* Une pile ayant été placée à la fabrique dans la carte RC-2 jointe, il peut arriver qu'elle soit épuisée avant d'avoir atteint la limite de longévité de deux ans.

• Changement de la pile de carte RAM

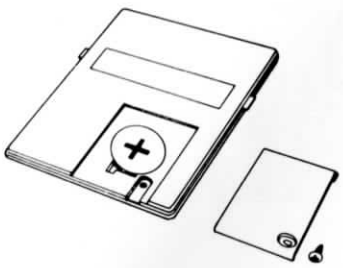
Afin déviter de toucher la surface du connecteur, changer la pile avec le couvercle de protection du connecteur fermé.

① Enlever la vis du couvercle du compartiment à pile situé au dos, puis enlever le couvercle en le tirant légèrement dans la direction de la flèche et retirer la pile usagée.



② Insérer une pile neuve après l'avoir essuyée avec un chiffon sec, la borne plus placée vers le haut et replacer le couvercle du compartiment à pile. Visser soigneusement. Des précautions doivent être prises afin d'éviter que les bornes plus et moins de la pile ne soient inversées par erreur.

* Ne pas jeter la pile usagée dans le feu; ceci pourrait provoquer une dangereuse explosion.



Conserver les piles dans un endroit situé hors de portée des enfants. Si une pile a été avalée, contacter immédiatement un médecin.

Etant donné que les programmes et les données stockés dans une carte RAM sont protégés par cette pile, veiller à la changer avant qu'elle ne soit épuisée. Il est recommandé de stocker les programmes et les données importants sur cassette avant de changer la pile.

■ Zone utilisateur et zone système

La capacité d'une carte RAM RC-2 est de 2048 octets et celle d'une carte RAM RC-4 est de 4096 octets. Cette capacité peut être divisée en trois zones qui sont:

1. Une zone système qui gère les programmes et les variables,
2. Une zone de variables fixes utilisée pour les variable de A à Z et
3. Une zone libre ou zone utilisateur utilisée pour les programmes et la banque de données.

Carte RAM	Zone système	Zone de variables fixes	Zone libre
RC-4	272 octets	208 octets	1568 octets
RC-2	272 octets	208 octets	3616 octets

1-5. AVANT D'EFFECTUER DES CALCULS

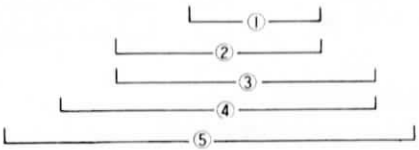
■ **Ordre de priorité des calculs**

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 facon suivante.

①Fonctions (SIN, COS, etc.)
②Puissance (↑)
③X (*), ÷ (/)
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 * SIN (17 + 13) ↑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 EXE
12345678912✱100 EXE
12345678912✱−100 EXE

1.234567891
1.2345678 E12
-1.234567 E12

Calcul effectué par programme

Pour PRINT 12345678912✱−100

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

Est automatiquement déplacé.

Disparaît de l'affichage.

N'a pas été affiché.

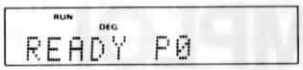
CHAPITRE 2

PASSONS A L'EMPLOI

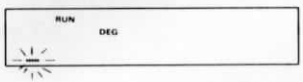
Vous devez utiliser cet ordinateur afin de vous habituer à lui, mais une mauvaise utilisation ne risque pas d'entraîner de panne. Tout d'abord, essayer une opération simple en accord avec le proverbe "il vaut mieux faire l'expérience pratique des choses qu'uniquement consulter des livres."

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.



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.

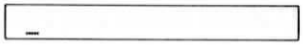


Lorsque le curseur clignote, on dit que l'ordinateur est en "état d'attente d'entrée (de données)", il attend l'entrée d'une opération ou d'une instruction. Ordinairement le curseur clignote sous la forme " _ ", mais lorsqu'on entre des caractères, il clignote également sous la forme " ▮ ". On peut entrer 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.

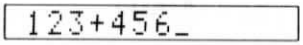
Exemple) 123+456=579



Appuyer sur la touche [AC] .

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

[1][2][3][+][4][5][6]



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

[EXE]

Une autre opération.

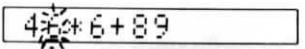
Exemple) 45×6+89=359

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

[4][6][×][6][+][8][9]

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 [←] .

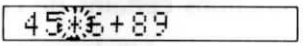
[←][←][←][←][←][←]



Le curseur et le chiffre 6 clignotent.

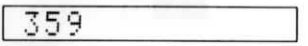
Entrer alors un [5] .

[5]



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

[EXE]



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.

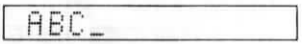
Maintenant entrons des caractères en utilisant les touches alphabétiques. L'ordre des touches alphabétiques est le même que celui d'une machine à écrire de type QWERTY. Etant donné que la plupart des ordinateurs ont un clavier de type QWERTY, se souvenir de l'emplacement des caractères même si ce n'est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

D'abord entrer des caractères.

Exemple) Les caractères utilisés sont "ABCXYZ".

Entrer ABC.

[A][B][C]

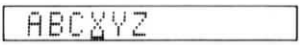


Entrer ensuite XYZ.

[X][Y][Z]

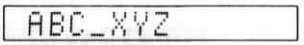
Maintenant mettre un espace entre ABC et XYZ.
Placer le curseur sous le X.

[←][←][←]



Introduire un espace.

[SHIFT][INS DEL]



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 DEL]. Répéter cette procédure si d'autres espaces doivent être insérés.

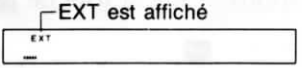
Cet ordinateur possède des caractères spéciaux en plus des caractères alphanumériques, ils servent pour les jeux ou comme symboles scientifiques. Voir page 16 les différentes sortes de caractères spéciaux.

Affichons certains de ces caractères.

Exemple) Affichage des symboles ♤♡◇♧.

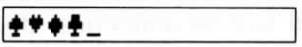
D'abord mettre en mode extension.

[AC][MODE][•]



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

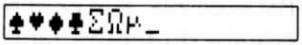
[SHIFT][♤][SHIFT][♡][SHIFT][◇][SHIFT][♧]



Exemple) Afficher les symboles Σ , Ω , μ .

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

[SHIFT] [S] [SHIFT] [A] [SHIFT] [M]



Comme l'ordinateur possède ces symboles, veuillez les utiliser. Pour retourner au mode dans lequel les lettres majuscules sont affichées, appuyer de nouveau sur [MODE] [•] 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 64 et 192.

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 [MEMO] . 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) [2][3][+][4][.][5][−][5][3][EXE] | -25.5

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

Exemple 2) 56×(−12)÷(−2.5)=268.8

Opération) 56 [×][−] 12 [÷][−] 2.5 [EXE] | 268.8

* Pour entrer un nombre negatif, appuyer sur la touche [−] avant le nombre.

Exemple 3) 7×8−4×5=36

Opération) 7 [×] 8 [−] 4 [×] 5 [EXE] | 36

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

Exemple 4) $(4.5\times10^{75})\times(-2.3\times10^{-78})=-0.01035$

Opération) 4.5 [E] 75 [×][−] 2.3 [E] [−] 78 [EXE] | -0.01035

* Appuyer sur la touche [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, $3x+5=$
$4x+6=$
$5x+7=$

Si la valeur de x dans ces calculs est 123,456, 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 DE BASE.

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		**Format**
Fonctions trigonométriques	$\sin x$	SIN x
	$\cos x$	COS x
	$\tan x$	TAN x
Fonctions trigonométriques inverses	$\sin^{-1} x$	ASN x
	$\cos^{-1} x$	ACS x
	$\tan^{-1} x$	ATN x
Racine carrée	$\sqrt{x}$	SQR x
Logarithme	$\log x$	LOG x
Logarithme népérien	$\ln x$	LN x
Fonction exponentielle	e^x	EXP x
Puissance	x^y	$x\uparrow y$
Conversion décimal/sexagésimal		DMS $ (x)*
Conversion sexagésimal/décimal		DEG (x,y,z)*
Valeur entière		INT x
Partie fractionnaire		FRAC x
Valeur absolue $\lvert x \rvert$	$\lvert x \rvert$	ABS x
Signe	nombres positifs $\rightarrow 1$ $0 \rightarrow 0$ nombres négatifs $\rightarrow -1$	SGN x
Arrondi	la valeur de x est arrondie à la position 10^y	RND (x,y)*
Nombre aléatoire		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°=0.2138079201

Opération MODE 4 → "DEG"
SIN 12.3456 EXE

```
0.2138079201
```

* A partir de maintenant, les touches alphabétiques et les touches numériques seront représentées sans cadre.
* Avec les FX-720P et FX-820P le même résultat peut être obtenu en appuyant sur FUNC SIN 12.3456 EXE .

Exemple) cos 63°52'41"=0.4402830847

Opération) COS DEG SHIFT , 63) 52) 41 SHIFT , EXE

```
0.4402830847
```

Exemple) 2·sin45°×cos65.1°=0.5954345575

Opération) 2 × SIN 45 × COS 65.1 EXE

```
0.5954345575
```

Exemple) $\sin^{-1}0.5=30°$

Opération) ASN 0.5 EXE

```
30
```

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

Opération) MODE 5 → "RAD"
COS SHIFT , SHIFT (π) ÷ 3 SHIFT , EXE

```
0.5
```

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

Opération) ACS SHIFT , SQR 2 ÷ 2 SHIFT , EXE

```
0.7853981634
```

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

Opération) MODE 6 → "GRA"
TAN − 35 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) LOG 1.23 EXE

```
0.0899051114
```

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

Opération) LN90 EXE

```
4.49980967
```

Exemple) $e^5=148.4131591$

Opération) EXP5 EXE

```
148.4131591
```

Exemple) $10^{1.23}=16.98243652$
(L' antilogarithme de 1,23 est obtenu.)

Opération) 10 SHIFT $\square^{\square}$ 1.23 EXE

```
16.98243652
```

Exemple) $5.6^{2.3}=52.58143837$

Opération) 5.6 SHIFT $\square^{\square}$ 2.3 EXE

```
52.58143837
```

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

Opération) 123 SHIFT $\square^{\square}$ SHIFT $\square^{\square}$ 1 ◢ 7 SHIFT $\square^{\square}$ EXE

```
1.988647795
```

x^y

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

Exemple) log sin40° + log cos35° = −0.278567983
Le nombre est 0,5265407845 (calcul logarithmique de sin 40° × cos 35°).

Opération) MODE 4 → " DEG "
LOG SIN 40 + LOG COS 35 EXE
10 SHIFT $\square^{\square}$ SHIFT ANS EXE

```
-0.278567983
0.5265407845
```

• Autres fonctions (√, SGN, RAN#, RND, ABS, INT, FRAC)

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

Opération) SQR 2 + SQR 5 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 < RAN# < 1)

Opération) RAN SHIFT # EXE

0.7903739076

Exemple) Arrondir la réponse de 12,3 × 4,56 à une décimale.

12.3×4.56=56.088

Opération) RND SHIFT 12.3 × 4.56) − 2 SHIFT EXE

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

56.1

Exemple) |−78.9÷5.6|=14.08928571

Opération) ABS SHIFT − 78.9 ÷ 5.6 SHIFT EXE

14.08928571

Exemple) Partie entière de 7800/96 ... 81

Opération) INT SHIFT 7800 ÷ 96 SHIFT 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 ... 0,25

Opération) FRAC SHIFT 7800 ÷ 96 SHIFT 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".

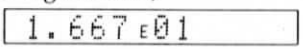
Désignation du nombre de chiffres significatifs ..... SET E n (n = 0 à 8)
Désignation du nombre de décimales ..... SET F n (n = 0 à 9)
Annulation de désignation de nombre de chiffres ..... SET N

* 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÷6=16.66666666·····

SET E 4 EXE (Désignation de 4 chiffres significatifs)

Opération) 100 ÷ 6 EXE



Exemple) 123÷7=17.57142857·····

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

123 ÷ 7 EXE

Exemple) 1÷3=0.3333333333·····

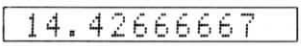
Opération) SET N EXE (Annulation de désignation)

1 ÷ 3 EXE

- **Conversion décimal ↔ sexagésimal (DEG, DMS $)**

Exemple) 14°25'36"=14.42666667

Opération) DEG SHIFT ° 14 ▷ 25 ▷ 36 SHIFT ° EXE



Exemple) 12.3456° = 12°20'44.16"

Opération) DMS [SHIFT] [,] [SHIFT] [,] 12.3456 [SHIFT] [,] [EXE]

12° 20' 44.16

Exemple) sin 63°52'41" = 0.897859012

Opération) [MODE] [4]

SIN DEG [SHIFT] [,] 63 [▷] 52 [▷] 41 [SHIFT] [,] [EXE]

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 maîriser 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.

```
10 INPUT X
```

N° de ligne Commande Opérande

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 123 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,

```
20 Y=2*X↑2+5*X+13
```

N° de ligne Instruction d'affectation

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 maitriser 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

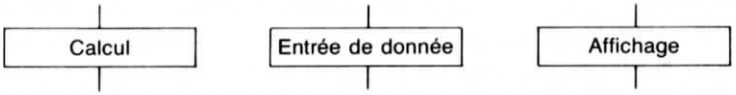
Vous ne devez pas être habitué à un organigramme! C'est un graphique qui décrit les étapes d'un travail.

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.

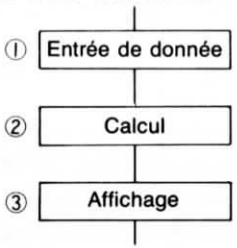
Préparons un programme en expliquant chaque élément.

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.



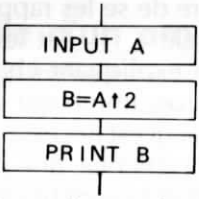
Pour mettre dans l'ordre ces trois éléments, on peut facilement deviner que le dernier est l'affichage de la réponse. Ensuite, qu'il est nécessaire de faire le calcul pour obtenir une réponse et également d'entrer une donnée pour faire le calcul.

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
soit plus proche de celle d'un programme.

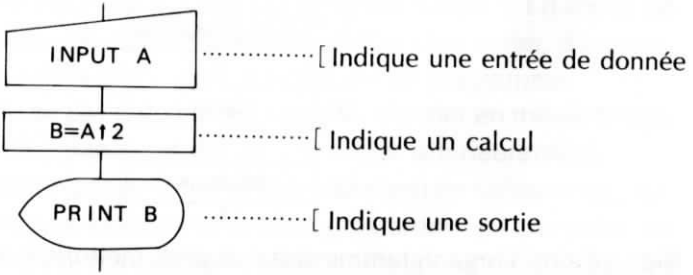
Le premier élément est une instruction d'entrée de donnée. Comme en utilisant une commande INPUT, une donnée est entrée dans une variable, déterminer cette variable. Si la variable A est utilisée, le contenu de (1) est "INPUT A".
Dans le calcul effectué dans le second élément, la valeur entrée dans la 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=A↑2
30 PRINT B
```

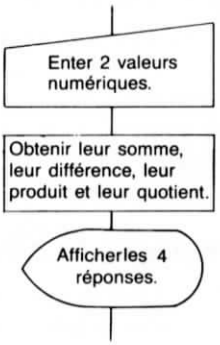
On peut facilement et rapidement écrire un programme en construisant un organigramme après avoir préparé chaque élément comme mentionné ci-dessus.
Un véritable organigramme comporte des symboles formels qui ont des significations spéciales. Faisons l'organigramme de notre exemple en utilisant ces symboles 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.

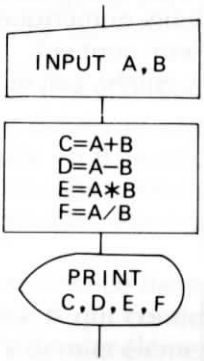


Utiliser une instruction INPUT (instruction qui permet d'entrer des données dans une variable à partir du clavier) pour enter 2 valeurs numériques. Les éléments qui doivent être notés sont les variables dans lesquelles sont affectées les données entrées et les réponses. L'utilisation des variables est 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.

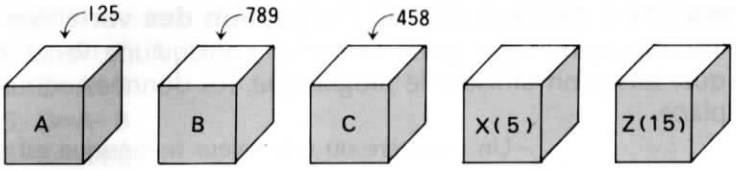
```
70 END
```

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.

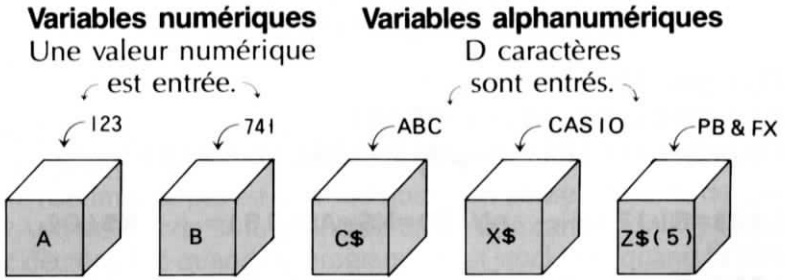
REMARQUE

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).



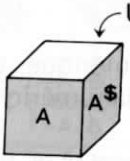
Il y a également deux types de variables, les numériques dans lesquelles sont entrées des valeurs numériques et les alphanumériques qui servent à entrer des chaînes de caractères.
Les variables que nous avons utilisées précédemment et où nous avons entré des valeurs numériques pour effectuer des calculs 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.



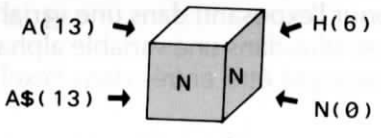
Etant donné que les éléments entrés dans ces deux types de variables sont différents, les caractères tels que "ABC" ne peuvent être entrés dans une variable numérique, de même des valeurs numériques ne peuvent être affectées pour des calculs à des variables alphanumériques.
L'utilisation de ces variables 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 1ère variable, la 2e, 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 eviter 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 dit également insérer pour stocker un programme, le mode WRT est appelé mode "insertion". Appuyer sur [MODE] [1] pour passer en mode WRT.

Cet affichage montre un état dans lequel aucun programme n'est stocké. Le nombre à 4 chiffres ci-dessus indique le nombre de pas restants. Le nombre maximal de pas est de 1568 quand on utilise une carte de mémoire vive (RAM) RC-2 et de 3616 quand on utilise une carte RAM RC-4. Ce nombre est décrémenté quand un programme est stocké ou quand la mémoire est étendue (voir page 95). 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 [EXE]
```

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

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

└Cette opération peut aussi être exécutée en appuyant sur les touches alphabétiques I N P U T .

20 C = A + B [EXE]

30 D = A − B [EXE]

40 E = A × B [EXE]

50 F = A ÷ B [EXE]

60 [SHIFT] [PRINT] C , D , E , F [EXE]

70 E N D [EXE]

Après l'appui sur la touche [EXE] , 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 [EXE] .**

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 [EXE] .

, B [EXE]

```
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 [EXE] .

```
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, ₂

Ouvrir un espace d'un caractère.

SHIFT INS

60 PRINT C, _

Insérer "D".

D

0 PRINT C, D₂

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

EXE

60 PRINT C, D

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

Vu qu'une ligne est stockée comme partie d'un programme une fois qu'on a appuyé sur la touche EXE , 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.

SHIFT LIST 50 EXE

50 F=A/N_

L' appui sur L I S T 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 EXE .

EXE

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 EXE , 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 d'une fonction partage de mémoire qui divise la mémoire en 10 zones de programme, P0 à P9, dans lesquelles des programmes indépendants peuvent être stockés. L'emploi de toutes ces zones de programme est identique. Par exemple, s'il n'y avait pas cette fonction partage et si 3 programmes devaient Oetre utilisés très souvent, il faudrait les charger chaque fois à partir d'une cassette, ou la carte RAM devrait être changée. 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 (1568 pas quand on emploie une carte RAM RC-2, et 3616 pas quand on emploie une carte RAM RC-4).

Une zone de programme peut être spécifiée en appuyant sur une touche numérique, de [0] à [9], après avoir appuyé sur la touche [SHIFT]. 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 [MODE] [0].

Exemple) [MODE] [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 MODE 0 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 SHIFT . Si un programme est stocké, il démarrera.

Exemple) SHIFT PO

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

Exemple) SHIFT RUN (de la même manière que R U N) 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

SHIFT PO

Affichage

```
?
```

Entrer 2 données.

Exemple)

45 EXE
36 EXE

```
?
81                STOP
```

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                 STOP
1620              STOP
1.25              STOP
```

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

[SHIFT] [RUN] 20 [EXE]

(ou RUN 20 [EXE] .)

```
 81
```

[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 [SHIFT] [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.

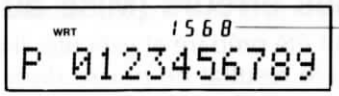
REMARQUE

COMMENT COMPTER LE NOMBRE DE PAS

Cet ordinateur a une capacité mémoire de 1568 pas avec une carte RAM RC-2 et de 3616 pas avec une carte RAM RC-4.
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 [MODE] [1] , le nombre de pas restants est affiché.

Exemple)

MODE 1



Nombre de pas restants

On compte le nombre de pas de la manière suivante.

- Numéro de ligne ..... 2 pas par numéro de 1 à 9999.
- Commande..... 1 pas
- Function..... 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
␣  ──────  ␣ 1
2     1    1

10 B=SIN A EXE ..... 7 pas
␣  ␣␣ 1  ␣ 1
2  1 1   1

100 PRINT "B=";B EXE .....10 pas
␣   ──────  ──── ␣␣ 1
2      1     4   1 1
                        Total: 22 pas
```

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

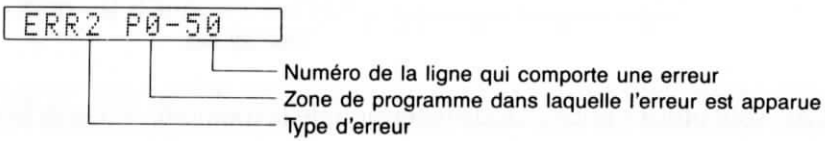
Exemple) Etat initial..... 26 mémoires 1568 pas
DEFM 10 EXE ..... 36 mémoires 1488 pas

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 192 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

MODE 1
SHIFT LIST 20 EXE
← ← DEL
EXE
AC MODE ⊡

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

SHIFT PO
45 MODE
12 EXE

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
MODE 1
SHIFT LIST 20 EXE

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

←
SHIFT INS
+ EXE
AC MODE ⊡

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 98), 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 FA-3 ou la mini-imprimante FP-12S sont bien branchés. Contrôler si l'accumulateur de la FP-12S est chargé et si le papier peut correctement se dérouler. Egalement régler le volume ou la tonalité du magnétophone à cassette connecté à la FA-3, 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 123.

(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 MODE 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	Affichage
MODE 1	P _ 123456789
SHIFT P1	P _ 123456789
NEW EXE	P _ 123456789
└ Commande d'effacement du programme.	

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

Appuyer sur AC MODE ⊡ pour spécifier le mode RUN.

	Opération	Affichage
Exemple)	SHIFT P1	?
	12 EXE	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.

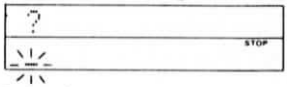
Opération	Affichage
MODE 1	P _ 123456789
55 STOP EXE	55 STOP
AC MODE ⊡	READY P1

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 **Affichage**

SHIFT P1

12 EXE

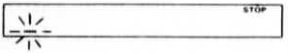


Contrôler la valeur de la variable de contrôle de boucle C.

C EXE

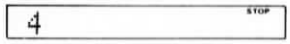
Continuer l'exécution

EXE



Contrôler de nouveau la valeur de la variable C.

C EXE



Bien que la valeur de la variable C doit être incrémentée de 11 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'incrémentation 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.

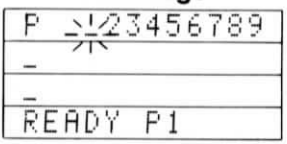
Opération **Affichage**

MODE 1

50 EXE

55 EXE

AC MODE 0



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 EXE pour passer à l'instruction suivante.
Essayer cela avec le précédent exemple. Chaque fois qu'on appuie sur EXE, la zone de programme et le numéro de ligne sont affichés.
Appuyer sur MODE 3 pour quitter le mode analyse, "TR" disparaît.

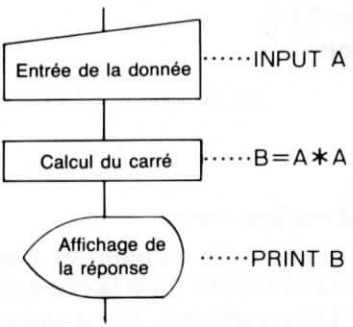
Etant donné que la correction des erreurs peut être effectuée comme mentionné ci-dessus, quand une erreur est affichée ou quand le résultat ne peut être obtenu, ne pas se décourager mais tenter la correction.

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	Affichage
RUN EXE	?
15 EXE	225
RUN EXE	?
45 EXE	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
```

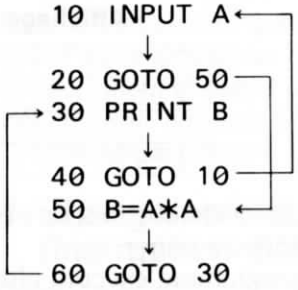
Cela signifie que le déroulement du programme se poursuivra à la ligne 10 après la ligne 40.
Exécuter le programme modifié par ce branchement.

Opération	Affichage
RUN EXE	?
15 EXE	225
EXE	?
43 EXE	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 "CASIO" → CASIO

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 EXE . 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-
                     puie sur la touche EXE .
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.
```

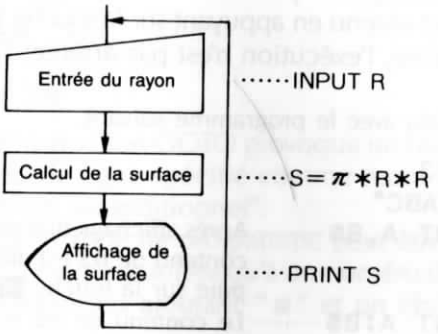
Ce programme est exécuté de la manière suivante.

Opération	Affichage	
RUN EXE	123	PRINT A,B$
EXE	ABC	
EXE	123ABC	PRINT A;B$
EXE	ABC 123	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 [SHIFT] [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=π∗R∗R
30 PRINT S
40 GOTO 10
```

ou

```
10 INPUT R
20 S=π∗R↑2
30 PRINT S
40 GOTO 10
```

"↑2" 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.

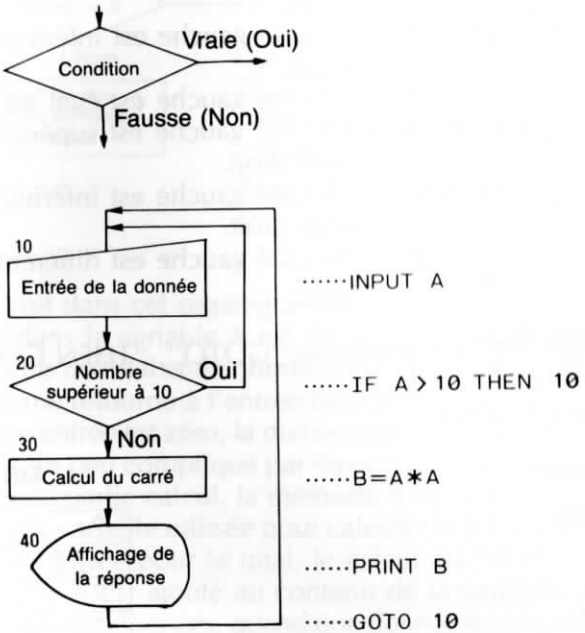
IF condition THEN { Numéro de ligne ou de zone de programme / Instruction }

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 demander 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 ≧ côté droit ... Le côté gauche est supérieur ou égal au côté droit.
Côté gauche ≦ côté droit ... Le côté gauche est inférieur ou égal au côté droit.
Côté gauche ≠ côté droit ... Le côté gauche est différent du côté droit.

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

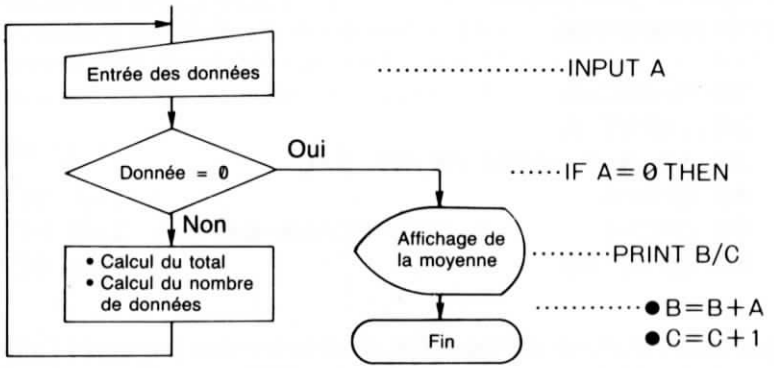
Opération	Affichage
RUN EXE	?
5 EXE	25
EXE	?
12 EXE	?
9 EXE	81

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

Exemple 2)

```
50 C=A+B
60 D=A-B    ⇒ 50 C=A+B:D=A-B:E=A*B:F=A/B
70 E=A*B
80 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 celà des précautions doivent être prises.

Exemple 3)

```
IF A<0 THEN A=10:B=20:·····
```

Exécutée que lorsque la condition est vraie.

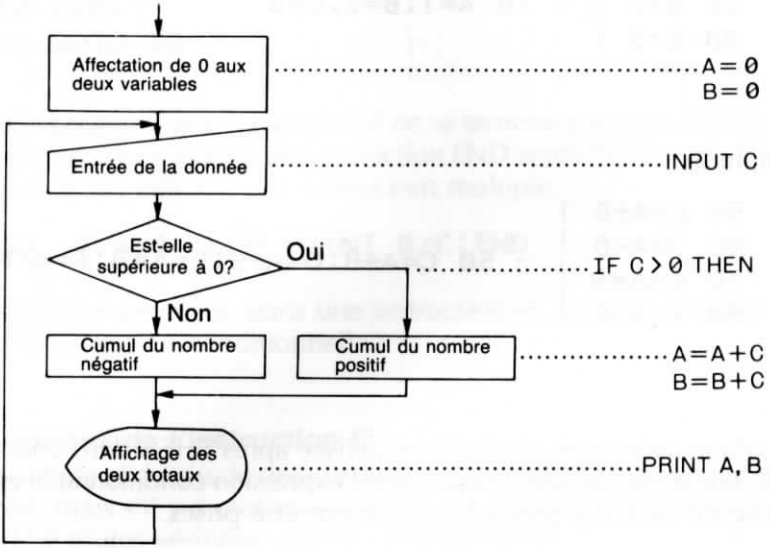
Une instruction multiple est pratique pour ordonner un programme. Néanmoins, comme une ligne n'est pas facile à lire si elle est trop longue, faire en sorte qu'elle soit d'une longueur appropriée et écrire les éléments restants sur la ligne suivante.

[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
```

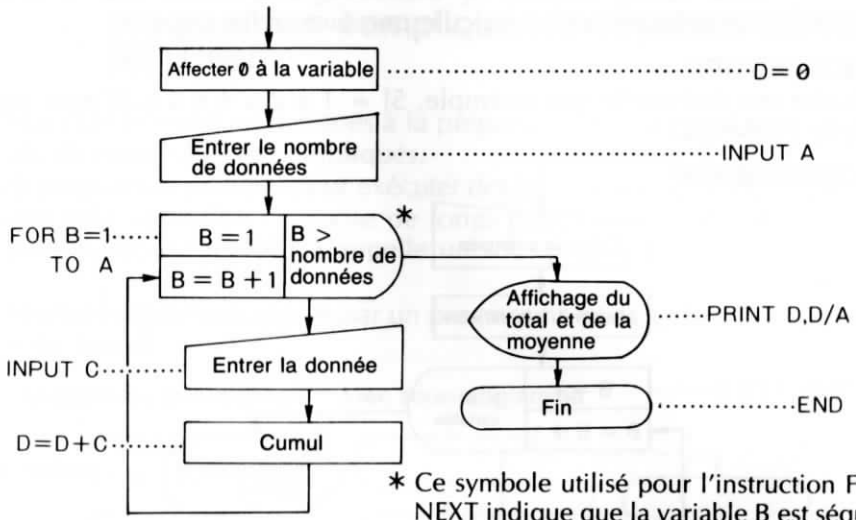
Ce programme est exécuté de la manière suivante.

Opération	Affichage
RUN [EXE]	?
3 [EXE]	1
[EXE]	4
[EXE]	7
[EXE]	10
[EXE]	?
0.8 [EXE]	1
[EXE]	1.8
[EXE]	2.6
[EXE]	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.

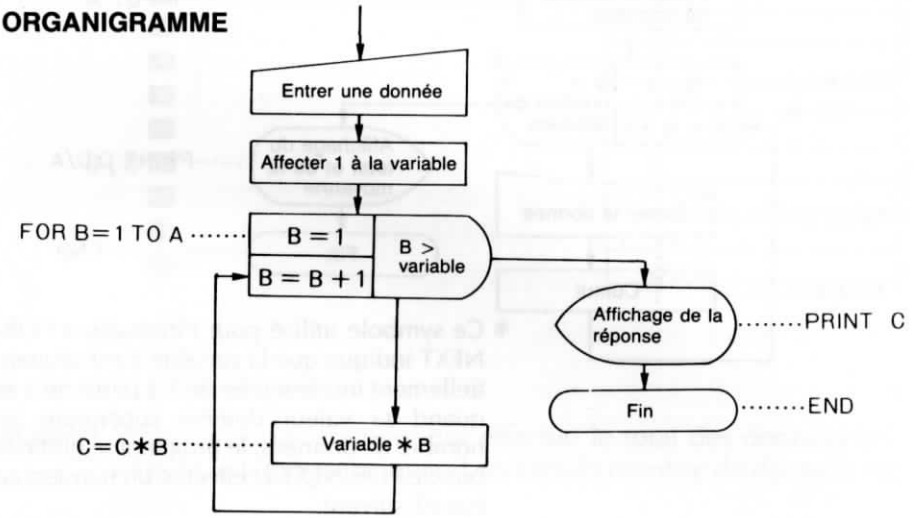
[EXERCICE]

Préparons un programme qui calcule une factorielle.

< SUGGESTION >

Calculer une factorielle (par exemple, 5! = 1 x 2 x 3 x 4 x 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
```

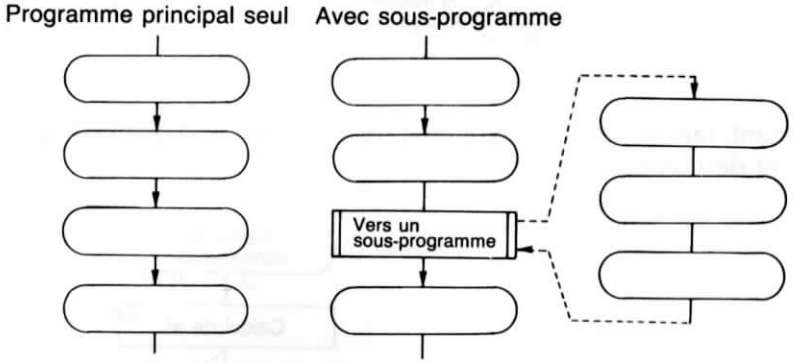
A la ligne 20, la valeur 1 est affectée à la variable C qui calcule la factorielle car on obtiendrait un résultat faux si la variable C n'était pas initialisée à 1.

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 x 2 x 3 x 4 x ... 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.



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

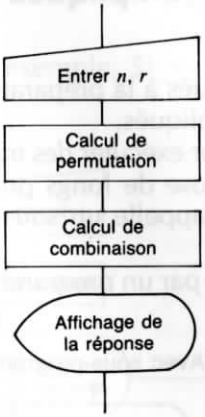
Exemple) Préparer un programme qui calcule permutations et combinaisons.

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

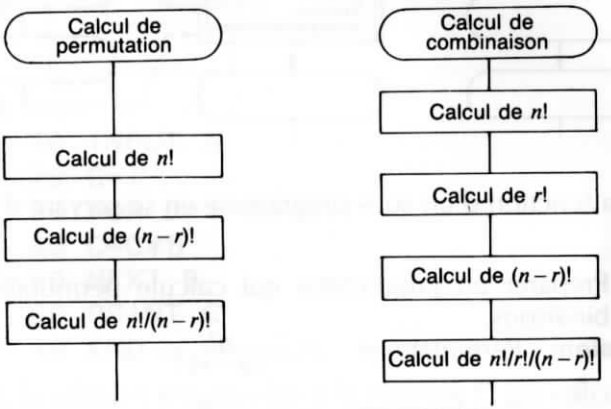
Combinaisons $_nC_r=\frac{n!}{r!(n-r)!}$

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
```

calcul de $n!$

calcul de $(n-r)!$

calcul de $\frac{n!}{(n-r)!}$

calcul de $r!$

calcul de $\frac{n!}{r!\,(n-r)!}$

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 nésessaire 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                  } Sous-programme
180 NEXT B                 │
190 RETURN                 ┘
```

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, celà a une fonction importante quand un programme est plus compliqué ou plus long.

[EXERCICE]

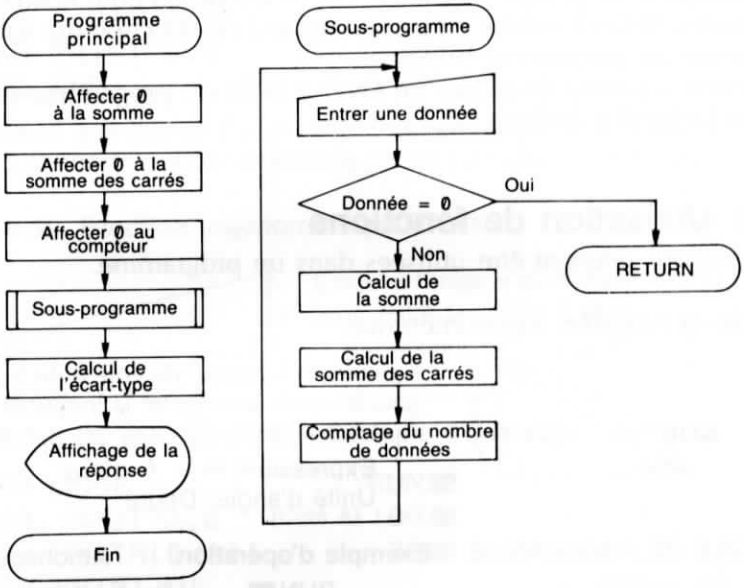
Préparer un programme qui calcule un écart-type. L'entrée des données, le calcul de la somme, le calcul de la somme des carrés et le comptage du nombre de données sont effectués dans un sous-programme.

Expression: $\sigma n = \sqrt{\dfrac{\Sigma x^2 - (\Sigma x)^2/n}{n}}$

Σx : Somme
Σx^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
```

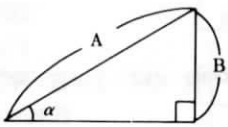
Sous-programme

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·sin α
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

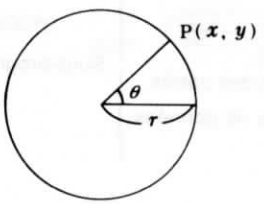
RUN EXE
(Côté A) 15 EXE
(Angle α) 30 EXE

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

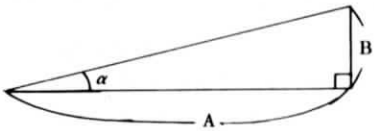
EXEMPLE DE PROGRAMME **Exemple d'opération** **Affichage**

```
10 MODE 5                    RUN EXE
20 INPUT R,T    (Rayon)        5 EXE
30 X=R*COS T    (Angle θ) π /3 EXE
40 Y=R*SIN T                   EXE
50 PRINT X,Y
60 END
```

```
?
?
 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 **Exemple d'opération** **Affichage**

```
10 MODE 4                    RUN EXE
20 INPUT A,B    (Côté A)   100 EXE
30 D=ATN(B/A)                20 EXE
40 PRINT D
50 END
```

```
?
?
 11.30993247
```

Exemple 4) Conversion décimal ↔ sexagésimal

Préparer un programme qui effectue un calcul de temps.

EXEMPLE DE PROGRAMME **Exemple d'opération** **Affichage**

```
10 T=0                              RUN EXE
20 INPUT D,E,G    (Heures)            1 EXE
30 S=SGN D        (Minutes)          25 EXE
40 D=ABS D        (Secondes)         36 EXE
50 T=T+S*DEG(D,E,G)                  EXE
60 PRINT DMS$(T)  (Heures)            2 EXE
70 GOTO 20        (Minutes)          15 EXE
                  (Secondes)          5 EXE
```

```
?
?
?
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
```

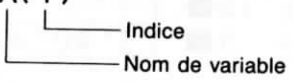
Un nombre aléatoire est généré à la ligne 10. Multiplier le nombre aléatoire généré par 99 puis ajouter 1 au résultat pour obtenir un nombre de 1 à 99 (voir page 160).

Quand le nombre aléatoire
est compris entre 5 et 9 → `INT(RAN#*5)+5`
Quand le nombre aléatoire
est compris entre 10 et 20 → `INT(RAN#*11)+10`

3-3-6 Utilisation d'un tableau

Différente des rariables 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)`



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 50:

```
10 A=0
20 FOR B=1 TO 50
30 IF P(B)>A THEN A=P(B)
40 NEXT B
50 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 196.

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 9e 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 100 éléments est nécessaire,

```
10 DEFM 76
20 B=0
30 FOR A=0 TO 99
40 INPUT C(A)
50 B=B+C(A)
60 NEXT A
```

..... Augmenter le nombre de variables de 76 pour en avoir 102.

..... Un tableau de C(0) à C(99) peut être utilisé.

Quand de nombreuses données sont gérées par un tableau, des précautions dóivent 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 99 dans les éléments d'un tableau allant de G(Ø) à G(99) et les ordonner de manière décroissante pour affichage.

EXEMPLE DE PROGRAMME

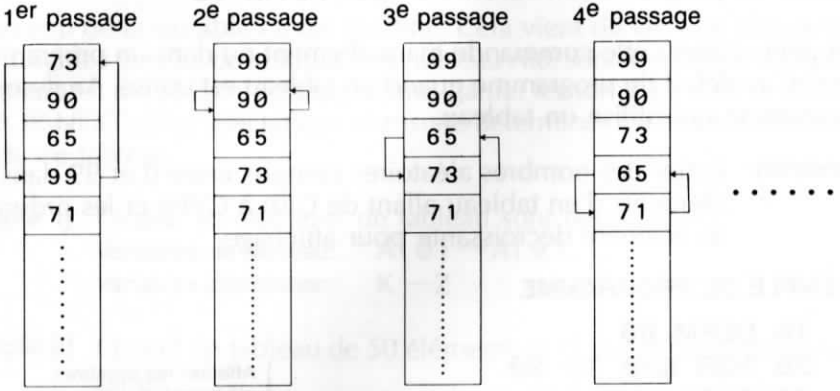
```
 10 DEFM 80
 20 FOR B=0 TO 99
 30 G(B)=INT(RAN#*100)
 40 NEXT B
 50 BEEP 1
 60 FOR B=0 TO 99
 70 A=-1
 80 FOR C=B TO 99
 90 IF G(C)>A THEN A=G(C):D=C
100 NEXT C
110 E=G(B):G(B)=G(D):G(D)=E
120 NEXT B
130 BEEP 0
140 FOR B=0 TO 99
150 PRINT G(B)
160 NEXT B
170 END
```

Affecter les nombres aléatoires compris entre 0 et 99 dans les éléments G(Ø) à G(99).

Ordonner les éléments de manière décroissante.

Les afficher séquentiellement.

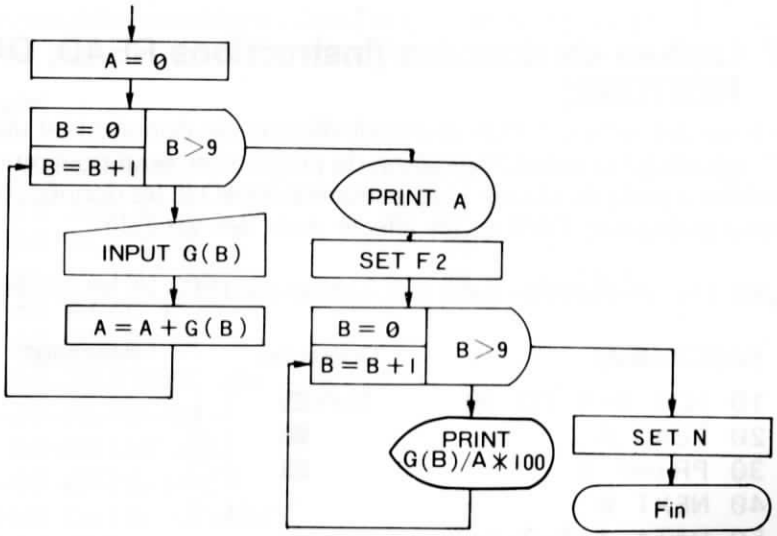
Ce programme est constitué de trois parties. Dans la première, des nombres aléatoires dont les valeurs sont comprises entre 0 et 99, sont générés et affectés dans les éléments d'un tableau allant de G(Ø) à G(99). Dans la deuxième partie, les éléments sont ordonnés de manière décroissante. Les instructions 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 Ø". 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 100 éléments de G(Ø) à G(99) est nécessaire, une extension mémoire de 80 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(Ø) 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 EXE	1
20 READ A	EXE	2
30 PRINT A	EXE	3
40 NEXT B	⋮	⋮
50 DATA 1,2,3,4,5, 6,7,8,9,10	⋮	⋮
60 END	EXE	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 EXE	CASIOPB&FX
20 PRINT A$;B$		
30 END		
40 DATA CASIO,PB & FX		

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

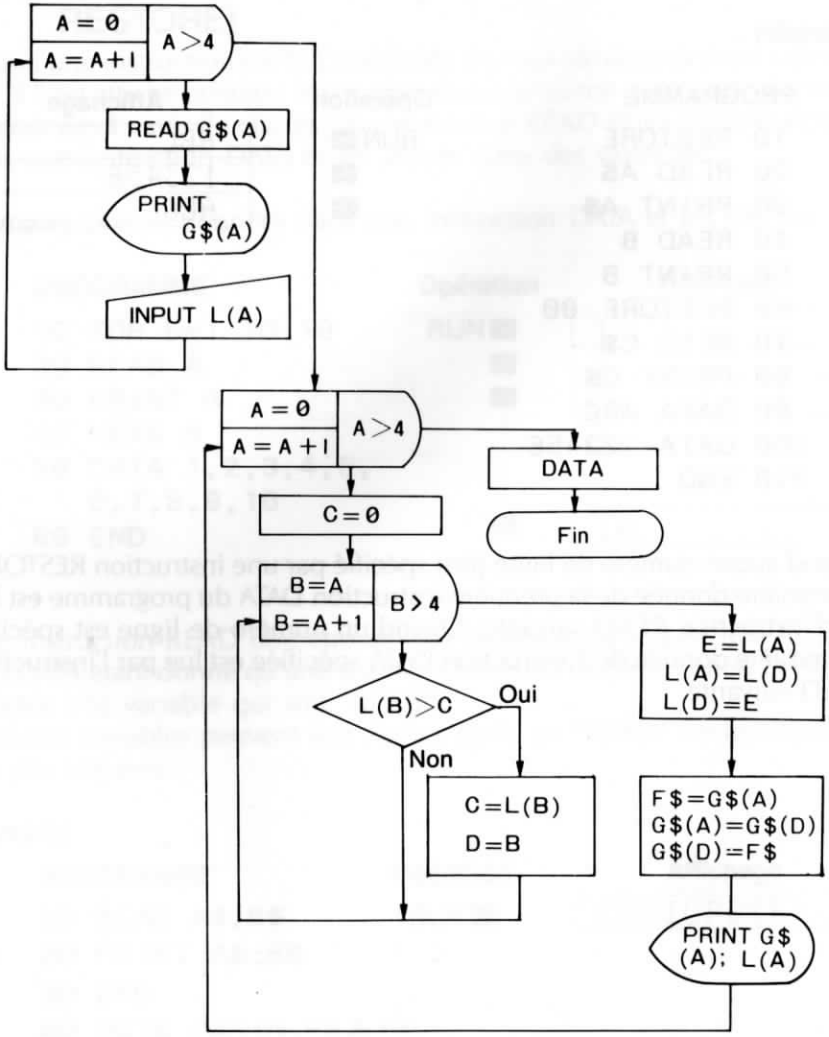
```
 10 RESTORE
 20 READ A$
 30 PRINT A$
 40 READ B
 50 PRINT B
 60 RESTORE 90
 70 READ C$
 80 PRINT C$
 90 DATA ABC
100 DATA 123456
110 END
```

Opération	Affichage
RUN [EXE]	ABC
[EXE]	123456
[EXE]	ABC

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

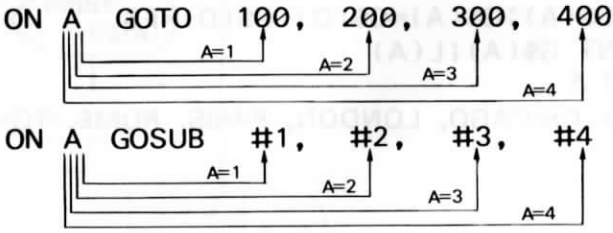
A : Utilisée dans une instruction FOR-NEXT
B : Utilisée dans une instruction FOR-NEXT
C : Valeur maximale
D : Nombre de valeur maximale
E : Utilisée pour le tri
F$: Utilisée pour le tri
G$(0)—G$(4): Nom de villes
L(0)—L(4): Données

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.

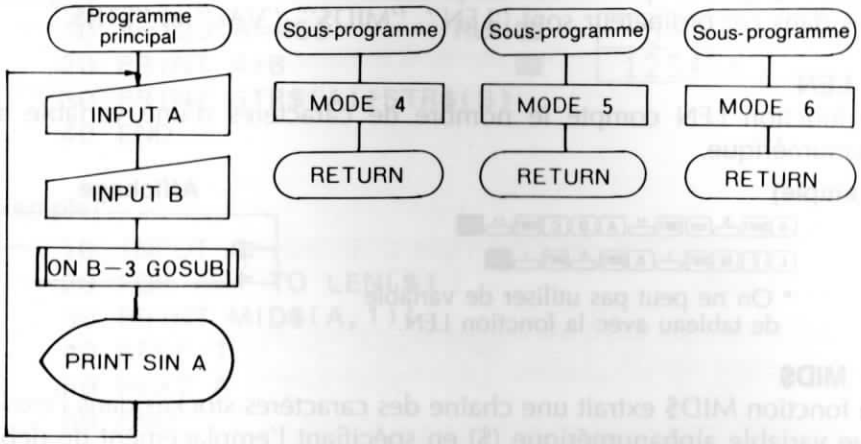
Exemple)

```
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 149). 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] [$] [=] [SHIFT] ["] [A] [B] [C] [SHIFT] ["] [EXE]
[L] [E] [N] [SHIFT] [(] [A] [SHIFT] [$] [SHIFT] [)] [EXE]

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

Affichage

```
_
3
```

• 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)

```
10 $="CASIO PB&FX"
20 PRINT $
30 PRINT MID$(1,5)
40 PRINT MID$(7,5)
50 END
```

Opération

RUN [EXE]
[EXE]
[EXE]

Affichage

```
CASIO PB&FX
CASIO
PB&FX
```

• VAL

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

Exemple)

```
10 A$="123":B$="456"
20 PRINT A$+B$
30 PRINT VAL(A$)
   +VAL(B$)
40 END
```

Opération

RUN [EXE]
[EXE]

Affichage

```
123456
  579
```

* 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)

```
10 A=123:B=456
20 PRINT A+B
30 PRINT STR$(A)+STR$(B)
40 END
```

Opération

RUN EXE
EXE

Affichage

```
 579
123456
```

Exemple)

```
10 INPUT $
20 FOR A=1 TO LEN($)
30 PRINT MID$(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 $="END" THEN 100
50 B=B+VAL($)
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.
A 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 [SHIFT] [P1] est utilisée, étant donné que l'entrée d'un caractère est effectuée si on continue d'appuyer sur la touche numérique [1] , 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$
   :
   :
```

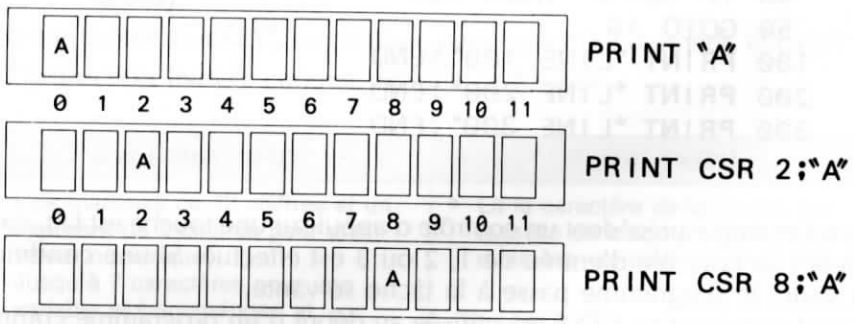
Attend que la touche enfoncée soit relâchée.

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 [2] 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	Opération
01↑3456789	Appuyer sur [2].
0123456↑89	Appuyer sur [8].
0123↑56789	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

Cet ordinateur a des équipements optionnels: une interface pour magnétophone à cassette (FA-3) et une imprimante à caractères pour les PB-410 et FX-720P (FP-12S).
Le FA-3 permet de stocker des programmes et données de l'ordinateur à une cassette, ou de les charger d'une cassette à l'ordinateur.
La FP-12S peut imprimer des programmes, des données et des résultats arithmétiques.
La fonction de chaque appareil est expliquée ci-dessous.

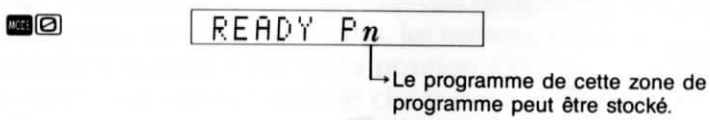
3-4-1 Stockage d'un programme ou de données.

Le FA-3 est nécessaire pour stocker un programme ou des données sur cassette. Pour brancher le FA-3 à l'ordinateur et à un magnétophone à cassette, voir le mode d'emploi fourni avec le FA-3.

■ Stockage et chargement de programme

Etant donné que les programmes sont mémorisés dans l'ordinateur, il arrive parfois qu'un programme suivant ne puisse pas être mémorisé car il dépasse la capacité mémoire. Dans ce cas, si l'on efface le précédent programme, il ne pourra pas être réutilisé. De même, lorsque l'on change la pile de la carte RAM, programme et données sont effacés. Dans ces cas, le FA-3 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



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 "CASIO" [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 "X".
* 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 au FA-3.
- "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 du FA-3.

■ 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 ou lorsque la pile de la carte RAM est changée, 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" EXE

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" EXE

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 "CASIO" comme nom de fichier.

```
PUT "CASIO" 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 "A$, Z$".
"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 "PB" dans les variables G(0) à G(99).

```
GET "PB" G(0),G(99)
```

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 à caractères.
Se mettre en mode impression ("PRT" est affiché) pour effectuer cette dernière.
On spécifie le mode impression en appuyant sur MODE 7 et on l'annule en appuyant sur MODE 8 .
Pour imprimer le contenu d'un programme, exécuter la commande LIST après avoir appuyé sur MODE 7 .

Exemple)
Entrer le programme suivant.

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

Ensuite, effectuer l'impression.

MODE 7 LIST EXE

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 EXE

Annuler le mode impression en appuyant sur MODE 8 après avoir effectué l'impression.
Pour imprimer un résultat arithmétique, spécifier le mode impression en appuyant sur MODE 7 ou en écrivant "MODE 7" dans le programme. Lorsque l'on a appuyé sur MODE 7 , 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 M O D E au lieu de la touche MODE .

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 PB-100

Les programmes préparés pour les PB-100 et PB-300 peuvent être utilisés par cet ordinateur. Cet ordinateur possède plus de commandes que les PB-100 et PB-300; son utilisation est plus pratique.
Le langage BASIC utilisé par cet ordinateur est presque le même que celui utilisé par les PB-100 et PB-300.

■ 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	PB-100/PB-300
NEW (NEW ALL)	CLEAR (CLEAR ALL)
CLEAR	VAC
IF-THEN	IF-;
SAVE ALL	SAVE A
LOAD ALL	LOAD A
VERIFY	VER
DEFM (Peut être écrite dans un programme)	DEFM (Ne peut être exécutée que manuellement.)

- **Fonctions modifiées**

Cet ordinateur	PB-100/PB-300
KEY$	KEY
MID$	MID

Malgré ces différents points, un programme préparé pour les PB-100/PB-300 peut être utilisé fondamentalement par cet ordinateur.
Néanmoins, il est préférable de réécrire les programmes pour vcet ordinateur afin qu'ils puissent être utilisés plus facilement ou revus plus tard sans difficultés.

Exemple 1)

Programme de PB-100

```
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 le PB-100/PB-300.

```
5 DEFM 20
```

Exemple 2)

Programme de PB-100

```
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 ordinateurs de poche de la gamme CASIO 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 → FX-710P

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

Cet ordinateur → PB-100, PB-300, FX-700P, FX-802P

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 d'autres ordinateurs CASIO, les commandes READ#, WRITE# et RESTORE# ne doivent pas être utilisées.
  KEY$ et MID$ doivent être changées en KEY et MID pour les PB-100, PB-300, FX-700P et FX-802P.
- Quand un programme préparé par d'autres ordinateurs CASIO 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.

- $\left\{\begin{matrix}\times\times\times\times\\ \bigcirc\bigcirc\bigcirc\bigcirc\end{matrix}\right\}$ ..... Un des éléments entre { } doit être sélectionné.
- $\left[\bigcirc\bigcirc\bigcirc\bigcirc\right]$ ..... L'élément entre [ ] peut être omis.
- $\bigcirc\bigcirc\bigcirc\bigcirc^{*}$ ..... 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.
- Ⓟ ..... Peut être uniquement exécuté dans un programme.
- Ⓜ ..... Peut être uniquement exécuté manuellement.
- Ⓐ ..... Peut être exécuté et manuellement et dans un programme.
- Ⓕ ..... 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". Vue 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 MODE 1 NEW EXE

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 PØ 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), 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 PØ 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. Egalement, 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 á prendre

Si un mot de passe a été oublié après qu'il ait été spécifié, appuyer sur le bouton ALL RESET situé au dos de l'ordinateur pour effacer tous les programmes et la mémoire.

Exemple PASS "CASIO" EXE

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 programe.
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 [EXE]
SAVE "CASIO"[EXE]
SAVE ALL "PB"[EXE]
```

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 abrégrée sous la forme LOAD A.

Relations entre SAVE et LOAD

	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 noms de fichier sont identiques.

○···peut être chargé.
×···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 [EXE]
VERIFY "PROG1" [EXE]
```

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 140), 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 } Ⓟ

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$="CASIO"
50 B$="PB & FX"
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=π*R↑2:REM AREA
30 PRINT S
40 END
```

INPUT ["Instruction de message";] (Chaîne de caractères) Nom de variable [, ["Instruction de message";] (Chaîne de caractères) Nom de variable]* Ⓟ

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 EXE 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 AC . 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 EXE pendant l'attente d'entrée de données, celà 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]]* Ⓟ

Fonction

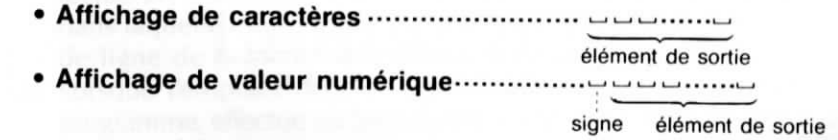
Affiche un élément de sortie.

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.



(4) Lorsqu'un élément de sortie est une expression numérique et lorsque la mantisse comporte plus de 10 chiffres, le 11e 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 EXE. 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 7).
(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
```

CSR Spécification de l'emplacement de sortie Ⓕ

Expression numérique

Fonction

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

Paramètre

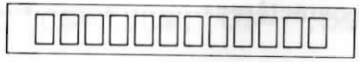
Spécification de l'emplacement de sortie: Expression numérique. Les chiffres après la virgule sont supprimés.

$$0 \leqq \text{spécification} < 12$$

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 CSR I;"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 EXE.

GOTO

{ Numéro de ligne de branchement / Numéro de ligne
numéro de zone de programme / Caractère de 0 à 9 }

Ⓟ

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 Condition de branchement / Expression numérique GOTO [Emplacement de branchement] [, [Emplacement de branchement]]* Ⓟ

★ Emplacement de branchement { Numéro de ligne de branchement / # 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 $\underset{A=1}{100}$, $\underset{A=2}{200}$, $\underset{A=3}{300}$, ...

(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, celà provoque les branchements respectifs de 100 à 300, dans le cas contraire "OTHER" est affiché.

IF Condition de branchement / Expression de comparaison **THEN** { Instruction [: instruction]* / Emplacement de branchement }

★Emplacement de branchement { Numéro de ligne de branchement / # 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;"↑";
30 K$=KEY$
40 IF K$="4"THEN N=N-1:IF N<0THEN N=0
50 IF K$="6"THEN N=N+1:IF N>11THEN N=11
60 PRINT
70 GOTO 20
```

- "↑" est décalé vers la gauche quand on appuie sur la touche 4 et vers la droite quand on appuie sur la touche 6.

FOR Nom de variable de contrôle = Valeur initiale (Expression numérique) **TO** Valeur finale (Expression numérique) Ⓟ
[**STEP** Incrément (Expression numérique)] **NEXT** Nom de variable de contrôle

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'emboitement 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'emboitement 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 { Numéro de ligne de branchement / Numéro de ligne ; # numéro de zone de programme / Un caractère de 0 à 9 } (P)

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'emboitement 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 1000, 2000, 3000·····
(B=1) (B=2) (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 ABC, DEF, "GHI, JKL", ·····
(1ère) (2e) (3e)

(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 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 necessaires 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 [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 ≦ Numéro de ligne ≦ 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 celà, 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(100) ..... Contenu des variables ou éléments de tableau de A à A (100).

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(100) ..... Lit et affecte dans les variables ou les éléments de tableau de A à A (100).

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 [{ 0 / 1 }]

Ⓐ

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

Ⓐ

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 ≦ Taille de l'extension mémoire < 453
(Quand la carte RAM RC-4 est utilisée).

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

Ⓟ

Fonction

Définit l'état de l'ordinateur.

Paramètre

Expression numérique: Les décimales ne sont pas prises en compte.

4 ≦ 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
```

SET { Fn / En / N }

Ⓐ

★ *n* est un entier compris entre 0 et 9.

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
```

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
```

MID$ (Emplacement [, Nombre de caractères]) Ⓕ

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 ≦ Emplacement < 101

Nombre de caractères: Expression numérique.
Les décimales ne sont pas prises en compte.

1 ≦ 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) Ⓕ

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 +, −, •, E ou E⁻, 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
```

STR$ (Expression numérique) Ⓕ

Fonction

Convertit la valeur d'une expression numérique en chaîne de caractères.

Paramètre

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.

Exemple

```
10 PRINT STR$(123)
20 PRINT STR$(45+78)
30 A=963
40 PRINT STR$(A)
50 END
```

FONCTIONS NUMERIQUES

SIN Argument / Expression numérique **COS** Argument / Expression numérique Ⓕ

TAN Argument / Expression numérique

Fonction

Pour un argument, donne la valeur d'une fonction trigonométrique.

Paramètre

Argument: Expression numérique

$-1440° <$ argument $< 1440°$ (degrés)

$-8\ \pi <$ argument $< 8\ \pi$ (radians)

$-1600 <$ argument < 1600 (grades)

Toutefois, pour TAN, "| Argument | = $(2n-1) * 1$ angle droit" est exclu.

1 angle droit = $90° = \frac{\pi}{2}$ radians = 100 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 MODE ou la commande MODE).

ASN Argument / Expression numérique ACS 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, $-1 \leqq$ argument $\leqq 1$

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.

$-90° \leqq$ **ASN X** $\leqq 90°$
$0° \leqq$ **ACS X** $\leqq 180°$
$-90° <$ **ATN X** $< 90°$

LOG Argument / Expression numérique LN Argument / Expression numérique Ⓕ

Fonction

Donne la valeur d'une fonction logarithmique.

Paramètre

Argument: Expression numérique.

0 < 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

$-227 \leqq$ argument $\leqq 230$

Explication

Donne la valeur d'une fonction exponentielle.

EXP e^x

SQR

Argument
Expression numérique

Ⓕ

Fonction

Donne la racine carrée d'un argument.

Paramètre

Argument: Expression numérique.

$0 \leqq$ argument

Explication

Donne la racine carrée d'un argument.

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) Ⓕ

(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) Ⓕ

(Argument: Expression numérique ; Position: 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 < 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 < nombre aléatoire < 1

(2) Le nombre aléatoire possède 10 chiffres.

Exemple

Fournit un chiffre aléatoire compris entre 0 et 9.

INT (RAN# * 10)

Fournit un chiffre aléatoire compris entre 1 et 5.

INT (RAN# * 5) + 1

Fournit un nombre aléatoire de deux chiffres compris entre 10 et 99.

INT (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.

| DEG (degrés, minutes, secondes) | < 10^{100}

Explication

Convertit le sexagésimal exprimé en degrés, minutes et secondes en décimal.

Exemple

```
DEG( 12,34,56) EXE
```

```
12.58222222
```

```
10 INPUT A,B,C
20 PRINT DEG(A,B,C)
30 END
```

DMS$ Argument (Expression numérique) (F)

Fonction

Convertit le décimal en sexagésimal.

Paramètre

Argument: Expression numérique.

| **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) EXE
```

```
45°40'40.8
```

```
10 INPUT A
20 $=DMS$( A)
30 PRINT$
40 END
```

COMMANDES DE BANQUE DE DONNEES

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
MODE 0

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. 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 # "CASIO" 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 # "CASIO" EXE

READ# Nom de variable [, Nom de variable]* Ⓟ

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
Nº1  A,X,Y
Nº2  B,Z
Nº3  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 165).
(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〉

```
Nº1  1,2,3
Nº2  4,5,6
Nº3  7,8,9
Nº4  10,
```

〈Programme〉

```
10 A=0
20 READ#$
30 IF $=" " THEN 60
40 A=A+VAL($)
50 GOTO 20
60 PRINT "Σx=";A
70 END
```

• Lit des données numériques pour obtenir une somme.

RESTORE# ["Chaîne de caractères recherchée" / Expression alphanumérique [, [{ 0 / 1 }]] Ⓟ

[, { Numéro de ligne / # numéro de zone de programme }]]]

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 < numéro de ligne < 10000

Numéro de zone de programme: Expression numérique.

0 ≦ 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

<Programma>

```
 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 EXE

EXE

EXE

Affichage

```
ABC
123
ERR4 P0-230
```

Manque de données par suite de l'effacement.

CHAPITRE 5

BIBLIOTHEQUE DE PROGRAMMES

1. Calcul de statistiques
2. Totalisation
3. Jeu de course de voiture
4. Jeu de bombardement
5. Jeu de sports athlétiques

1. CALCUL DE STATISTIQUES

Ce programme peut être utilisé à la fois pour un calcul d'écart-type avec une variable et pour une analyse de régression avec deux variables appariées. Son utilisation est très simple car il suffit d'entrer les données pour obtenir une réponse. On peut entrer autant de données que l'on désire.

Les expressions de calcul sont les suivantes:
n: Nombre de données
Σx: Somme des données x
Σy: Somme des données y
Σx^2: Somme des carrés des données x
Σy^2: Somme des carrés des données y
Σxy: Somme des produits des données

Moyenne des données x ($\bar{x}$) : $\dfrac{\Sigma x}{n}$

Moyenne des données y ($\bar{y}$) : $\dfrac{\Sigma y}{n}$

Ecart-type des données x ($x\sigma_{n-1}$) : $\sqrt{\dfrac{n\Sigma x^2-(\Sigma x)^2}{n(n-1)}}$ [Lorsque l'on utilise des données de population finie]

Ecart-type des données x ($x\sigma_n$) : $\sqrt{\dfrac{n\Sigma x^2-(\Sigma x)^2}{n^2}}$ [Lorsque l'on utilise des données d'échantillon de population]

Ecart-type des données y ($y\sigma_{n-1}$) : $\sqrt{\dfrac{n\Sigma y^2-(\Sigma y)^2}{n(n-1)}}$

Ecart-type des données y ($y\sigma_n$) : $\sqrt{\dfrac{n\Sigma y^2-(\Sigma y)^2}{n^2}}$

Terme constant de régression linéaire (A): $\dfrac{\Sigma y-\mathrm{LRB}\cdot\Sigma x}{n}$

Coefficient de régression linéaire (B) : $\dfrac{n\cdot\Sigma xy-\Sigma x\cdot\Sigma y}{n\cdot\Sigma x^2-(\Sigma x)^2}$

Coefficient de corrélation (r) : $\dfrac{n\cdot\Sigma xy-\Sigma x\cdot\Sigma y}{\sqrt{\{n\Sigma x^2-(\Sigma x)^2\}\{n\Sigma y^2-(\Sigma y)^2\}}}$

Valeur estimée de x ($\hat{x}$) : $\dfrac{y_n-\mathrm{LRA}}{\mathrm{LRB}}$

Valeur estimée de y ($\hat{y}$) : $\mathrm{LRA}+x_n\cdot\mathrm{LRB}$

• **Liste du programme**

```
10 PRINT "START ?(
   Y/N)";
20 $= KEY$
30 IF $="N" THEN 1
   00
40 IF $<>"Y" THEN 2
   0
50 PRINT : BEEP 0
60 CLEAR
70 PRINT "DATA 1 O
   R 2?";
80 A$= KEY$
90 IF A$<>"1" THEN
   IF A$<>"2" THEN
   80
100 B=B+1
110 PRINT : BEEP 1
120 PRINT "X DATA";
   B;
130 INPUT X$: BEEP
   1
140 IF X$="E" THEN
   B=B-1: BEEP 0:
   GOTO 240
150 C=C+ VAL(X$)
160 D=D+ VAL(X$)↑2
170 IF A$="1" THEN
   100
180 PRINT "Y DATA";
   B;
190 INPUT Y
200 H=H+Y
210 I=I+Y*Y
220 M=M+ VAL(X$)*Y
230 GOTO 100
240 E=C/B
250 F= SQR((B*D-C*C
   )/(B*(B-1)))
260 G= SQR((B*D-C*C
   )/(B*B))
270 IF A$="1" THEN
   340
280 J=H/B
290 K= SQR((B*I-H*H
   )/(B*(B-1)))
300 L= SQR((B*I-H*H
   )/(B*B))
310 O=(B*M-C*H)/(B*
   D-C*C)
320 N=(H-O*C)/B
330 P=(B*M-C*H)/ SQ
   R((B*D-C*C)*(B*
   I-H*H))
340 INPUT "INPUT(0-
   17)",Z
350 IF Z=0 THEN PRI
   NT "END":: END
360 ON Z-15 GOTO 45
   0,480
370 RESTORE
380 FOR W=1 TO Z
390 READ V$
400 NEXT W
410 PRINT V$;"=";A(
   Z)
420 DATA N,SUMX,SUM
   X2,MEANX,SDX,SD
   XN,SUMY
430 DATA SUMY2,MEAN
   Y,SDY,SDYN,SUMX
   Y,LRA,LRB,COR
440 GOTO 340
450 INPUT "Y DATA",
   Y
460 PRINT "EOX=";(Y
   -N)/O
470 GOTO 340
480 INPUT "X DATA",
   X
490 PRINT "EOY=";N+
   X*O
500 GOTO 340
```

Total 728 pas

- **Contenus des variables**

A		Décision d'utiliser une variable ou deux variables appariées	L	A(11)	Ecart-type des données y ($y\sigma_n$)
B	A(1)	Nombre de données	M	A(12)	Somme des produits des données
C	A(2)	Somme des données x	N	A(13)	Terme constant de régression linéaire
D	A(3)	Somme des carrés des données x	O	A(14)	Coefficient de régression linéaire
E	A(4)	Moyenne des données x	P	A(15)	Coefficient de corrélation
F	A(5)	Ecart-type des données x ($x\sigma_{n-1}$)	V$		Nom de la sortie
G	A(6)	Ecart-type des données x ($x\sigma_n$)	W		Utilisée dans une boucle
H	A(7)	Somme des données y	X		Pour entrée de donnée x
I	A(8)	Somme des carrés des données y	Y		Pour entrée de donnée y
J	A(9)	Moyenne des données y	Z		Utilisée pour la sélection de sortie
K	A(10)	Ecart-type des données y ($y\sigma_{n-1}$)	$		Utilisée pour la fonction KEY$

Utilisons ce programme.
Les données suivantes sont utilisées comme exemple.

	1	2	3	4	5
x (Températures)	10	15	20	25	30
y (Barre d'acier)	1003	1005	1010	1011	1014

Opération **Affichage**

RUN [EXE]

```
START ?(Y/N)
```

Si l'on veut entrer une donnée, appuyer sur la touche [Y].

[Y]

```
DATA 1 OR 2?
```

Le programme demande si l'on veut utiliser une variable ou deux variables appariées. Notre exemple nécessite l'emploi de deux variables, appuyer donc sur la touche [2].

[2]

```
X DATA 1?
```

Après cela, entrer séquentiellement les données x et y.

10 EXE
1003 EXE
15 EXE
1005 EXE
⋮
30 EXE
1014 EXE

```
Y DATA 1?
X DATA 2?
Y DATA 2?
X DATA 3?
⋮
Y DATA 5?
X DATA 6?
```

Après avoir terminé l'entrée des données, entrer [E] comme signe de terminaison.

[E] EXE

```
INPUT(0-17)?
```

Ensuite, pour choisir l'affichage d'une réponse, entrer le numéro du code (0 à 17) correspondant. La liste de ces codes est donnée après cet exemple. Tout d'abord, obtenons la moyenne de ces deux suites de données.

($\bar{x}$) 4 EXE
EXE
($\bar{y}$) 9 EXE
EXE

```
MEANX= 20
INPUT(0-17)?
MEANY= 1008.6
INPUT(0-17)?
```

* Les contenus des lignes pointillées situées à gauche de l'affichage avancent séquentiellement et disparaissent.

Puis, obtenons le terme constant de régression linéaire, le coefficient de régression linéaire et le coefficient de corrélation linéaire

(A) 13 EXE
EXE
(B) 14 EXE
EXE
(r) 15 EXE
EXE

```
LRA= 997.4
INPUT(0-17)?
LRB= 0.56
INPUT(0-17)?
COR= 0.9826073689
INPUT(0-17)?
```

Ensuite, obtenons la valeur estimée de x ($\hat{x}$) quand celle de y est 1000 et la valeur estimée de y ($\hat{y}$) quand celle de x est 18.

16 EXE
(y_n) 1000 EXE
EXE
17 EXE
18 EXE
EXE

```
Y DATA?
EOX= 4.642857143
INPUT(0-17)?
X DATA?
EOY= 1007.48
INPUT(0-17)?
```

Pour terminer le programme, entrer 0.

0 EXT

```
END
```

Pour entrer des données sans interruption, appuyer sur la touche [N] , après le démarrage du programme.

RUN EXE
[N]

```
START ?(Y/N)
X DATA 6?
```

Lorsqu'une variable doit être utilisée, procéder de la manière suivante.

RUN EXE
[Y]
[1]
10 EXE
15 EXE

```
START ?(Y/N)
DATA 1 OR 2?
X DATA 1?
X DATA 2?
X DATA 3?
```

Table des numéros de codes

Code		Code	
1	Nombre de données (n)	10	Ecart-type des données y ($y\sigma_{n-1}$)
2	Somme des données x (Σx)	11	Ecart-type des données y ($y\sigma_n$)
3	Somme des carrés des données x (Σx^2)	12	Somme des produits des données (Σxy)
4	Moyenne des données x ($\bar{x}$)	13	Terme constant de régression linéaire (A)
5	Ecart-type des données x ($x\sigma_{n-1}$)	14	Coefficient de régression linéaire (B)
6	Ecart-type des données x ($x\sigma_n$)	15	Coefficient de corrélation (r)
7	Somme des données y (Σy)	16	Valeur estimée de x ($\hat{x}$)
8	Somme des carrés des données y (Σy^2)	17	Valeur estimée de y ($\hat{y}$)
9	Moyenne des données y ($\bar{y}$)		

★ Point ★

Dans ce programme, les variables sont utilisées de deux manières différentes: utilisation ordinaire des variables de B à P et comme éléments de tableau A (1) à A (15).
La variable B utilisant le même élément de mémoire que la variable de tableau A (1), le contenu est le même bien que les noms soient différents. Une expression de calcul différente étant utilisée jusqu'à la ligne 330, les variables sont traitées comme des variables dont le nom n'est composé que d'un caractère comme B, C, D ....
Le programme peut être raccourci et simplifié en entrant le numéro de code aux lignes 340 et suivantes; un tableau (A(1)—A(15)) est utilisé.

2. TOTALISATION

Ce programme se compose de six programmes indépendants. Le "Programme d'Entrée des Données" entré dans la zone P0 est utilisé pour spécifier les lignes et les colonnes d'une table dans laquelle des données sont entrées.
Le "Programme d'Affichage (Impression)" entré dans la zone P1 est utilisé pour séquentiellement afficher ou imprimer les données de la table.
Le "Programme de Mise en forme des Données" entré dans la zone P2 est utilisé pour corriger les données stockées.
Le "Programme de Calculs" entré dans la zone P3 est utilisé pour calculer les totaux verticaux, les totaux horizontaux et le total général.
Le "Programme de Sauvegarde des Données" entré dans la zone P4 est utilisé pour sauvegarder les données sur une cassette.
Le "Programme de Chargement des Données" entré dans la zone P5 est utilisé pour charger les données à partir d'une cassette dans les variables.
Exécutons ce programme avec les données suivantes.

	1	2	3	4	5	6
1	376	159	248	767	311	351
2	320	85	287	833	291	541
3	480	41	166	750	426	367
4	518	269	343	565	221	268
5	536	158	426	495	235	492

Opération	Affichage	
SHIFT PO	New[Y/N]?	Quand une nouvelle table est préparée, appuyer sur Y.
Y	VERTICAL?	Entrer le nombre de colonnes.
5 EXE	HORIZONTAL?	Entrer le nombre de lignes.
6 EXE	(1, 1)?	Entrer séquentiellement les données.
376 EXE	(1, 2)?	
159 EXE	(1, 3)?	
248 EXE	(1, 4)?	
⋮	⋮	
235 EXE	(5, 6)?	
492 EXE	END	Fin d'entrée des données.

Ensuite, contrôlons les données entrées.

Opération	Affichage	
SHIFT P1	Printer[Y/N]?	Pour sortir sur l'imprimante, appuyer sur Y.
N	(1, 1) 376	Chaque fois que l'on appuie sur EXE, une donnée est affichée.
EXE	(1, 2) 159	
EXE	(1, 3) 248	
⋮	⋮	
EXE	(5, 6) 492	
EXE	END	

Le programme de mise en forme entré dans la zone P2 est utilisé lorsque les données entrées sont incorrectes ou lorsqu'une partie des données doit être modifiée.
Par exemple, "450" a été entré par erreur comme donnée située à l'intersection de la ligne 4 et de la colonne 3.

Opération	Affichage	
SHIFT P2	VERTICAL?	Spécifie la colonne.
3 EXE	HORIZONTAL?	Spécifie la ligne.
4 EXE	(3, 4) 450?	La donnée située à l'intersection de la ligne 4 et de la colonne 3 est affichée. Si une correction est nécessaire, entrer la valeur numérique correcte.
750 EXE	(3, 5) 462?	

Pour contrôler la donnée suivante, appuyer sur + EXE et pour contrôler la donnée précédente, appuyer sur − EXE .

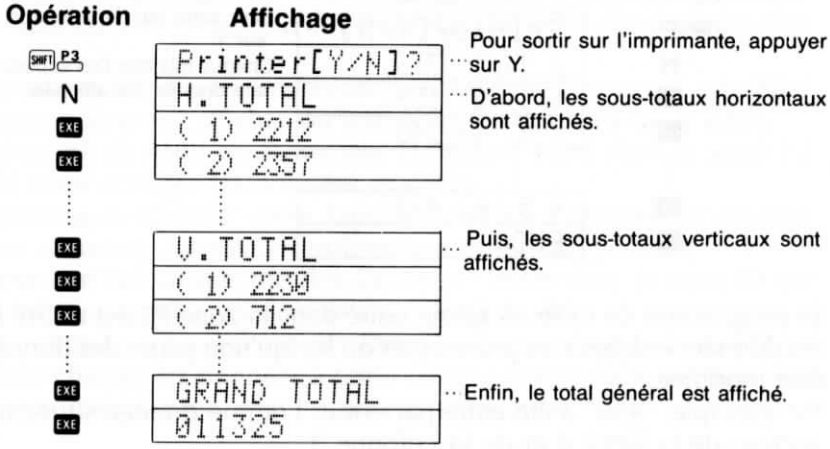
+ EXE	(3, 6) 367?
+ EXE	(4, 1) 518?

Lorsque la correction est effectuée, appuyer sur = EXE pour retourner à l'affichage de "VERTICAL?" qui permet de spécifier une colonne et une ligne. Si l'on appuie sur = EXE lorsque "VERTICAL?" est affiché, ce programme est terminé.

= EXE	VERTICAL?
= EXE	END

Si une valeur numérique est entrée lorsqu'une donnée est affichée, la nouvelle valeur remplace la précédente.

Le programme entré dans la zone P3 est utilisé pour obtenir les sous-totaux verticaux et horizontaux et le total général.



Les programmes P4 et P5 nécessitent l'interface cassette (FA-3).
L'exécution du programme de la zone P4 sauvegarde les données sur cassette. Brancher l'unité centrale et un magnétophone à la FA-3 et insérer les fiches dans les jacks de microphone et de commande à distance.
Le programme de la zone P5 permet de charger des données. Brancher l'ordinateur et un magnétophone à la FA-3 et insérer les fiches dans les jacks d'écouteur et de commande à distance.
Pour effectuer le stockage, mettre en place une cassette neuve et une cassette sur laquelle des données ont été sauvegardées pour effectuer le chargement.

★ Point ★

Ce programme utilisant un total de 1107 pas, le nombre de données (lignes × colonnes) ne peut être que 57 au maximum lorsque l'on utilise une carte RC-2 et 313 au maximum lorsque l'on utilise une carte RC-4. Lorsque l'on doit manipuler plus de données, modifier "57" à la ligne 80 de la zone P0 en fonction du nombre de pas restants.
Le programme de calculs entré dans la zone P3 est utilisé pour obtenir les sous-totaux verticaux et horizontaux et le total général. Si d'autre calculs doivent être effectués, modifier ce programme.

P0

```
 10 PRINT "New [Y/N
    ]?";
 20 K$= KEY$: IF K$
    ="Y" THEN PRINT
     : GOTO 50
 30 IF K$="" THEN 2
    0
 40 PRINT : GOTO 21
    0
 50 CLEAR
 60 INPUT "VERTICAL
    ",Y
 70 INPUT "HORIZONT
    AL",X
 80 IF Y*X>57 THEN
    60
 90 DEFM X*Y
100 FOR I=1 TO Y
110 FOR J=1 TO X
120 PRINT "(";I;","
    ;J;")";: INPUT
    $
130 IF $)"*" THEN I
    F $("π" THEN 18
    0
140 IF $*"=" THEN 1
    10
150 IF J-1)0 THEN J
    0J-1: GOTO 120
160 IF I-1(1 THEN 1
    20
170 I=I-1:J=X: GOTO
     120
180 Z((I-1)*X+J)= V
    AL($)
190 NEXT J
200 NEXT I
210 PRINT "END"
```

281 pas

P1

```
 10 PRINT "Printer[
    Y/N]";
 20 K$= KEY$: IF K$
    ="" THEN 20
 30 PRINT
 40 IF K$="Y" THEN
    MODE 7: PRINT "
    DATA"
 50 FOR I=1 TO Y
 60 FOR J=1 TO X
 70 PRINT "(";I;","
    ;J;")";Z((I-1)*
    X+J)
 80 NEXT J
 90 IF K$="Y" THEN
    PRINT " "
100 NEXT I
110 MODE 8
120 PRINT "END"
```

151 pas

P2

```
 10 INPUT "VERTICAL
    ",$
 20 IF $="=" THEN P
    RINT "END";: EN
    D
 30 IF $)"*" THEN I
    F $("π" THEN 50
 40 GOTO 10
 50 INPUT "HORIZONT
    AL",P
 60 O= VAL($)
 70 PRINT "(";O;","
    ;P;")";Z((O-1)*
    X+P);: INPUT $
 80 IF $="=" THEN 1
    0
 90 IF $="+" THEN 1
    40
100 IF $="-" THEN 1
    60
110 IF $)"*" THEN I
    F $("π" THEN 13
    0
120 GOTO 70
130 Z((O-1)*X+P)= V
    AL($)
140 IF P+1)X THEN O
    =O+1:P=0: IF O)
    Y THEN O=1:P=1:
     GOTO 70
150 P=P+1: GOTO 70
160 IF P-1(1 THEN O
    =O-1:P=X+1: IF
    O(1 THEN O=Y:P=
    X: GOTO 70
170 P=P-1: GOTO 70
```

273 pas

P3

```
 10 PRINT "Printer[
    Y/N]":
 20 K$= KEY$: IF K$
    ="" THEN 20
 30 IF K$="Y" THEN
    MODE 7
 40 PRINT
 50 PRINT "H. TOTAL
    "
 60 FOR I=1 TO Y
 70 A=0
 80 FOR J=1 TO X
 90 A=A+Z((I-1)*X+J
    )
100 NEXT J
110 PRINT "(";I;")"
    :A
120 NEXT I
130 PRINT "V. TOTAL
    "
140 B=0
150 FOR J=1 TO Y
160 A=0
170 FOR I=1 TO X
180 A=A+Z((I-1)*X+J
    )
190 NEXT I
200 PRINT "(";J;")"
    ;A
210 B=B+A
220 NEXT J
230 PRINT "GRAND TO
    TAL"
240 PRINT B
250 MODE 8
```

289 pas

P4

```
 10 PRINT "DATA PUT
    ";
 20 PUT "DATA"X,Y
 30 PUT Z(1),Z(X*Y)
 40 PRINT "→END"
```

53 pas

P5

```
 10 PRINT "DATA GET
    ";
 20 GET "DATA"X,Y
 30 DEFM X*Y
 40 GET Z(1),Z(X*Y)
 50 PRINT "→END"
```

60 pas

Total 1107 pas

3. JEU DE COURSE DE VOITURE

Il s'agit d'une course dans laquelle une longue distance est couverte sur un parcours difficile en tournant un volant vers la gauche ou vers la droite afin d'éviter de heurter les barrières.

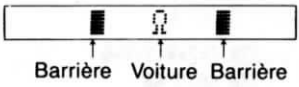
■ Liste du programme

```
 10 PRINT " CAR RAC
    E !";
 20 BEEP 0: GOSUB 5
    00
 30 PRINT "HI-SCO:"
    ;S;"km";
 40 GOSUB 500
 50 X=6:Y=3:Z=9:T=0
    :C=0
 60 PRINT
 70 PRINT CSRY;"■";
     CSRX;"Q"; CSRZ
    ;"■";
 80 IF X= INTY THEN
     GOSUB 600
 90 IF X= INTZ THEN
     GOSUB 600
100 T=T+1
110 $= KEY$
120 IF $="4" THEN X
    =X-1
130 IF $="6" THEN X
    =X+1
140 BEEP 0
150 R= RAN#*.9
160 IF RAN#>.5 THEN
     R=-R
170 IF Z+R>12 THEN
    R=0
180 Q= RAN#*.8
190 IF RAN#>.5 THEN
     Q=-Q
200 IF Y+Q<0 THEN Q
    =0
210 IF Z-Y<3 THEN 2
    30
220 Z=Z+R:Y=Y+Q
230 GOTO 60
500 REM TIME
510 FOR U=1 TO 100:
     NEXT U
520 PRINT
530 BEEP 0
540 RETURN
600 REM CRASH
610 FOR I=1 TO 10
620 PRINT CSRX;"*";
630 BEEP 1
640 PRINT CSRX;"Q";
650 NEXT I
660 PRINT CSR0:"<<C
    RASH !!>>";
670 GOSUB 500
680 PRINT "SCORE:";
    T*3;"km";
690 GOSUB 500
700 X=6:Y=3:Z=9
710 C=C+1
720 IF C<3 THEN RET
    URN
730 T=T*3
740 IF T>S THEN S=T
750 PRINT
760 $="GAME OVER !!
    "
770 FOR I=1 TO 12
780 PRINT MID$(I,1)
    ;: BEEP 1
790 NEXT I
800 END
```

Total 540 pas

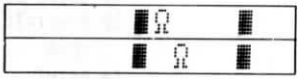
■ Explication du jeu

Seules les touches 4 et 6 sont utilisées. Appuyer sur ces touches pour respectivement déplacer la voiture vers la gauche ou vers la droite.

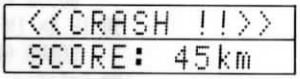


Les barrières de gauche et de droite se déplacent pour rendre le parcours plus large ou plus étroit. Utiliser adroitement les touches afin que la voiture ne heurte pas une des barrières.

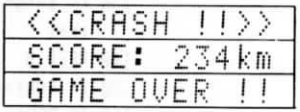
La voiture est près de la barrière de gauche.
6



Lorsque la voiture touche une barrière, elle explose et la distance parcourue est affichée.



La voiture peut avoir deux accidents, mais au troisième, le jeu est terminé.



4. JEU DE BOMBARDEMENT

Dans ce jeu, un sous-marin ennemi est détruit par un destroyer dont la navigation est adroitement commandée. Le sonar du destroyer est défectueux et ne répond que lorsque le sous-marin est directement situé sous le destroyer. En outre, la profondeur est inconnue et le destroyer a un minimum de carburant. Dans cette situation, le destroyer doit tirer en évitant les torpilles ennemies.

■ Liste du programme

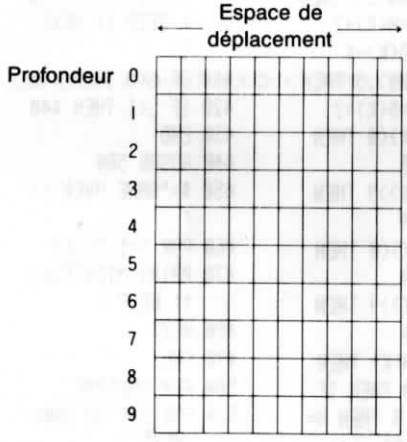
```
10 PRINT " (SUBMAR
   INE)";
20 BEEP : GOSUB 50
   0
30 PRINT "HI-SCO:"
   ;T;
40 BEEP : GOSUB 50
   0
50 X=4:S=100:R=0:N
   =0:L=3
60 FOR I=0 TO 2
70 A(I)= INT( RAN#
   *10):D(I)= INT(
    RAN#*10)
80 NEXT I
90 FOR K=0 TO 2
100 PRINT
110 S=S-1
120 IF S<20 THEN BE
   EP 1
130 IF S<0 THEN 370
140 PRINT "■■■■■■■■
   ■■"; CSRX:"◆";
150 IF A(K)=X THEN
   PRINT CSR11;"*"
   ;
160 $= KEY$: IF $="
   " THEN 200
170 IF $="Z" THEN.X
   =X-1: IF X<0 TH
   EN X=0: GOTO 20
   0
180 IF $="X" THEN X
   =X+1: IF X>9 TH
   EN X=9: GOTO 20
   0
190 IF $≥"0" THEN I
   F $≤"9" THEN GO
   SUB 600
200 IF A(K)<0 THEN
   360
210 IF RAN#<.8 THEN
    300
220 A(K)=A(K)-1
230 IF RAN#>.5 THEN
    A(K)=A(K)+2
240 D(K)=D(K)-1
250 IF RAN#>.5 THEN
    D(K)=D(K)+2
260 IF A(K)<0 THEN
   A(K)=0
270 IF A(K)>9 THEN
   A(K)=9
280 IF D(K)<0 THEN
   D(K)=0
290 IF D(K)>9 THEN
   D(K)=9
300 IF X=A(K) THEN
   IF N=0 THEN IF
   RAN#>.8 THEN N=
   1:M=D(K)
310 IF N=0 THEN 350
320 PRINT CSR11;"↑"
   ;: BEEP
330 IF M<0 THEN N=0
   : IF X=A(K) THE
   N GOSUB 900
340 M=M-1
350 GOTO 100
360 NEXT K
370 PRINT
380 IF S>0 THEN R=R
   +S
390 PRINT "SCORE:";
   R;
400 IF T<R THEN T=R
   : FOR I=1 TO 10
   : BEEP 1: NEXT
   I
410 IF S<0 THEN 440
420 IF L<1 THEN 440
430 END
440 GOSUB 500
450 $="GAME OVER !!
   "
460 FOR I=1 TO 12
470 PRINT MID$(I,1)
   ;: BEEP 1
480 NEXT I
490 END
500 REM SUBTIME
510 FOR U=1 TO 100:
    NEXT U
520 PRINT
530 RETURN
600 REM FIRE
```

```
610 BEEP
620 IF A(K)=X THEN
    IF D(K)= VAL($)
     THEN 650
630 IF A(K)=X THEN
    IF ABS(D(K)- VA
    L($))<2 THEN 71
    0
640 RETURN
650 FOR I=1 TO 10
660 PRINT CSR11;"*"
    ;: BEEP 1
670 PRINT CSR11;"+"
    ;: BEEP 0
680 NEXT I
690 A(K)=-1:R=R+ IN
    T( RAN#*5+1)*10
    0:S=S+50
700 RETURN
710 FOR I=1 TO 5
720 PRINT CSR11;"@"
    ;: BEEP 0
730 NEXT I
740 RETURN
900 REM DEAD
910 FOR I=1 TO 10
920 PRINT CSRX;"x";
    : BEEP 1: PRINT
     CSRX;"@";
930 NEXT I
940 L=L-1
950 IF L<1 THEN PRI
    NT : GOTO 380
960 RETURN
```

Total 999 pas

■ Explication du jeu

La zone marine est la suivante.

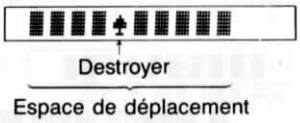


Appuyer sur les touches [Z] et [X] pour respectivement déplacer le destroyer vers la gauche ou vers la droite.
Pour utiliser une charge anti-sous-marine correspondant à la profondeur, celle-ci doit être spécifiée en appuyant sur une touche de [0] à [9] .

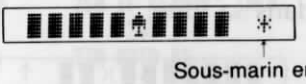
Il y a trois destroyers et trois sous-marins ennemis. Lorsque les trois sous-marins ont été détruits, le jeu est terminé et le score est affiché.
Le jeu se termine également quand les trois destroyers ont été coulés les premiers ou quand le carburant est épuisé.
Au début du jeu, le titre et le meilleur score sont affichés.

RUN EXE
(ou SHIFT PO)

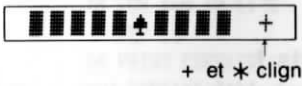
Tout d'abord, un destroyer et l'espace de déplacement sont affichés.



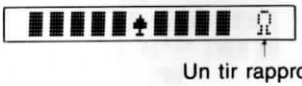
Lorsque l'on déplace le destroyer vers la gauche ou vers la droite en utilisant les touches Z ou X , le sonar signale la présence du sous-marin ennemi.



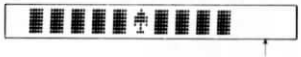
Le sous-marin est situé juste au-dessous du destroyer à une profondeur inconnue.
Lancer une charge anti-sous-marine en appuyant sur une touche de 0 à 9 pour indiquer la profondeur présumée.
Le lancement d'une charge anti-sous-marine est accompagné de sons. Lorsque la charge touche le sous-marin, celui-ci explose.



Lorsque la charge a manqué le sous-marin mais est passée près de lui (à une profondeur de ±1), la réponse du sonar change.

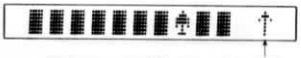


Le sous-marin s'échappe en se déplaçant vers la gauche ou vers la droite ou en changeant de profondeur. Essayer de ne pas le perdre. Lorsque le sous-marin n'est plus situé au-dessous du destroyer, la réponse du sonar disparaît.



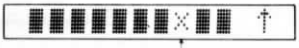
La réponse disparaît

Il arrive parfois que le sous-marin ne s'échappe pas seulement mais lance une torpille sur le destroyer.



Une torpille est lancée. (†et ✳ clignotent)

Lorsqu'une torpille a été lancée sur le destroyer, s'échapper à toute vitesse. Toutefois, la torpille ennemie est dotée d'une grande efficacité et continue de suivre le destroyer dans sa fuite.



Une torpille touche le destroyer
(♠ et × clignotent)

Lorsque le niveau de carburant devient bas, une alarme retentit sous la forme d'un bip-bip permanent. Etant donné que l'on ne peut pas refaire la plein de carburant, lorsque celui-ci est épuisé, le jeu prend fin. Toutefois, lorsqu'un sous-marin ennemi est touché, un peu de carburant est transféré de ce sous-marin au destroyer.

[Comptage des points]

Quand un sous-marin est détruit, de 100 à 500 points sont acquis.
Un bonus est donné pour le carburant restant à la fin du jeu.

5. JEU DE SPORTS ATHLETIQUES

Ce jeu se compose de trois épreuves comme indiqué ci-dessous.

1. (P0) : 100 mètres plat
2. (P1) : Saut en longueur
3. (P2) : Course de haies

Chaque programme est chargé dans une zone de programme indépendante. Le jeu commence par le 100 mètres plat. Si un résultat satisfaisant est obtenu, on passe à la deuxième épreuve. Lorsque la course de haies est terminée (dernière épreuve), le score total est affiché.

■ Liste du programme

```
P0
 10 X=0:Y=0:W=0:Z=0
    :K=0:B=100
 20 PRINT "HI-SCO";
    Q:"s";
 30 V=B: GOSUB #9:
    BEEP 1
 40 IF KEY$<>"" THEN
     GOSUB #7: GOTO
     30
 50 PRINT CSRX;"Q";
     CSR11;"!";
 60 FOR I=1 TO 5
 70 IF KEY$<>"" THEN
     W=W+.2
 80 Z=Z+1
 90 NEXT I
100 PRINT CSRX;" ";
110 FOR I=1 TO 5
120 IF KEY$<>"" THEN
     W=W-.2
130 NEXT I
140 X=X+W:W=0
150 IF INTX<>Y THEN
    BEEP :Y= INTX
160 IF X<0 THEN X=0
170 IF X<11 THEN 50
180 PRINT : BEEP 1
190 D= RND(Z/12,-3)
200 PRINT "TIME:";D
    ;"s";
210 IF Q=0 THEN Q=D
220 IF D<Q THEN Q=D
    : GOSUB #6
230 GOSUB #9
240 IF D>15 THEN #8
250 PRINT "NEXT GAM
    E ?";
260 IF KEY$="" THEN
     260
270 GOTO #1
```

343 pas

```
P1
 10 MODE 4:K=0
 20 PRINT
 30 PRINT "HI-SCO";
    P;"m";
 40 V=B: GOSUB #9
 50 W=0
 60 BEEP 1
 70 FOR X=0 TO 11 S
    TEP .5
 80 PRINT CSRX;"Q";
     CSR11;"_";
 90 $= KEY$
100 IF $<"Z" THEN I
    F $>"A" THEN W=
    W+1
110 IF $>"0" THEN I
    F $<"9" THEN GO
    TO 200
120 V=20-W
130 GOSUB #9: BEEP
140 NEXT X
150 FOR M=1 TO 5
160 $= KEY$
170 IF $>"0" THEN I
    F $<"9" THEN 20
    0
180 NEXT M
190 GOSUB #7: GOTO
    40
200 BEEP 1
210 FOR J=1 TO 80
220 IF KEY$="" THEN
     240
230 NEXT J
240 BEEP
250 R=W/2*(6-M)* CO
    S ABS(45-J)/6
260 FOR X=0 TO R ST
    EP .5
270 PRINT CSRX;"0";
280 V=10: GOSUB #9
290 BEEP 1
300 NEXT X
310 BEEP
320 PRINT,CSRR;"R";
330 V=B: GOSUB #9
340 E= RND(R,-3)
350 IF E<2 THEN GOS
    UB #7: GOTO 40
```

```
360 PRINT "SCORE:";
    E;"m";
370 IF P<E THEN P=E
    : GOSUB #6
380 V=8: GOSUB #9
390 IF E<7 THEN #8
400 PRINT "NEXT GAM
    E ?";
410 IF KEY$="" THEN
     410
420 GOTO #2
```

456 pas

```
P2
 10 $="    1   1
    1   1   1":X=0:
    K=0:Y=2:W=0:Z=0
 20 PRINT : PRINT "
    HI-SCO";0;"s";
 30 V=8: GOSUB #9
 40 BEEP 1
 50 IF KEY$<>"" THEN
     GOSUB #7: GOTO
     30
 60 PRINT CSR0; MID
    $(Y,11);
 70 PRINT CSRX;"Q";
 80 Z=Z+1
 90 A$= KEY$
100 IF A$<="9" THEN
    IF A$>="0" THEN
    H=1: BEEP 1: PR
    INT CSR0;"µ";
110 IF Y=1 THEN Y=Y
    +1: IF H<>1 THEN
     Z=Z+3: GOSUB #
    7
120 IF A$<="Z" THEN
    IF A$>="A" THEN
    Y=Y+1:W=W+1: BE
    EP
130 IF Y>4 THEN Y=1
140 H=0
150 IF W<30 THEN 60
160 PRINT CSR0;"1
     1   1   !";
170 PRINT CSRX:"Q";
180 Z=Z+1
190 A$= KEY$
200 IF A$<="9" THEN
    IF A$>="0" THEN
    H=1: BEEP 1: PR
    INT CSRX;"µ";
210 IF FRAC(X/4)=0
    THEN X=X+1: IF
    H<>1 THEN Z=Z+3:
     GOSUB #7
220 IF A$<="Z" THEN
    IF A$>="A" THEN
    X=X+1: BEEP
230 H=0
240 IF X<12 THEN 16
    0
250 BEEP : PRINT
260 F= RND(Z/1.1,-2
    )
270 PRINT "TIME:";F
    ;"s";
280 IF O=0 THEN O=F
290 IF F<O THEN O=F
    : GOSUB #6
300 GOSUB #9
310 IF F>60 THEN #8
320 R= INT( COS(D*3
    )*200+ SIN(E*3)
    *200+ COS(F*3)*
    200)
330 PRINT "TOTAL SC
    ORE";R;" points
     ";
340 IF T=0 THEN T=R
350 IF T<R THEN T=R
    : GOSUB #6
```

603 pas

```
P6
 10 FOR I=1 TO 10:
    BEEP 1: BEEP :
    NEXT I
 20 RETURN
```

22 pas

```
P7
 10 PRINT : PRINT "
    FOUL !!":: BEEP
     : BEEP
 20 IF K=0 THEN K=1
    : RETURN
 30 GOTO #8
```

39 pas

```
P8
 10 PRINT
 20 $="GAME OVER !!
    "
 30 FOR I=1 TO 12
 40 PRINT MID$(I,1)
    ;: BEEP
 50 NEXT I
```

20 pas

```
P9
 10 FOR U=1 TO V: N
    EXT U
 20 PRINT
 30 RETURN
```

50 pas

Total 1533 pas

P0 : 100 mètres plat
P1 : Saut en longueur
P2 : Course de haies
P6 : Bip-bip
P7 : Traitement des faux-départs
P8 : Traitement de fin de jeu
P9 : Arrêt après un certain laps de temps.

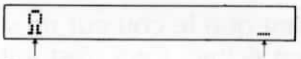
■ Explication du jeu

Lancer le jeu en appuyant sur RUN [EXE] ou sur [SHIFT] [0].

100 mètres plat

Le coureur clignote. Appuyer sur une des touches quand il est affiché pour le faire partir vers la droite. Si l'on appuie sur une touche lorsque le coureur n'est pas affiché, il recule vers la gauche.
Si le temps passé ("TIME") est inférieur à 15 secondes, le saut en longueur commence.
Si l'on appuie sur une touche avant que le coureur ne soit affiché, un faux-départ ("FOUL!!" est affiché) se produit et l'on doit effectuer un nouveau départ. Un seul faux-départ est autorisé. Au deuxième, le jeu prend fin.

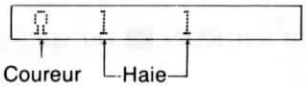
Saut en longueur



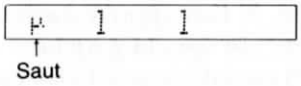
La vitesse de l'élan du sauteur est augmentée en appuyant sur les touches de [A] à [Z]. Bien que le sauteur avance lorsque l'on appuie sur aucune touche, le saut sera nul. Lorsque le sauteur a attteint la ligne d'appel, appuyer sur une des touches de [0] à [9]. L'angle de saut est calculé en fonction du laps de temps pendant lequel on a appuyé sur une de ces touches. Son enfoncement ne doit donc être ni trop court ni trop long.

Lorsqu'un saut n'a pas été effectué même si la ligne d'appel a été atteinte ou lorsque le saut a échoué, c'est un échec ("FOUL!!" est affiché). Ceci n'est autorisé qu'une seule fois.
Si la longueur du saut est inférieure à sept mètres, le sauteur est disqualifié et l'on ne peut pas passer à la course de haies.

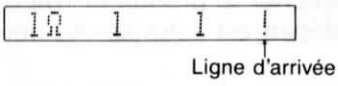
Course de haies



Pour faire courir le coureur, maintenir une touche alphabétique enfoncée. Lorsque le coureur atteint une haie, appuyer sur une touche numérique au bon moment pour le faire sauter.



Après avoir sauté plusieurs haies, on peut voir la ligne d'arrivée.



Si le départ est donné avant que le coureur ne soit affiché ou lorsqu'une haie est renversée, c'est un échec. Ceci n'est autorisé qu'une seule fois. Si le temps passé dépasse 60 secondes, le coureur est disqualifié et le score total n'est pas affiché.

• Touches utilisées

Les touches autres que [MODE], [←], [→], [SHIFT], [FUNC], [AC], [DEL], [STOP], [EXE] et [MEMO] peuvent être utilisées pour le 100 mètres plat.
Lorsque le coureur de haies ou le sauteur en longueur courent, n'importe quelle touche alphabétique de [A] à [Z] peut être utilisée alors que pour les faire sauter on doit utiliser une touche numérique de [0] à [9] .

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.	• Etendre 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.

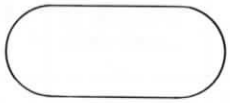
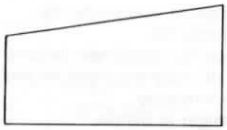
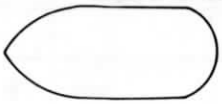
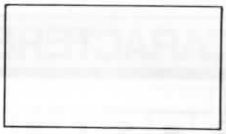
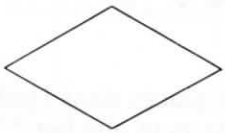
Code erreur	Signification	Cause	Mesure corrective
8	Erreur de mot de passe	• Lorsqu'un mot de passe est spécifié, a) un autre mot de passe est spécifié b) une commande telle que LIST ou NEW qui ne peut pas être utilisée, est exécutée.	• Supprimer le mot de passe.
9	Erreur d'option	• Les commandes SAVE ou PUT sont exécutés sans qu'un magnétophone soit brnaché. • 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. • Débourrer le papier.

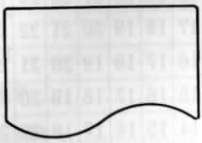
6-2. TABLEAU DES CODES DE CARACTERES

	Espace	+	−	*	/	↑	!	"	#	$	>	≥	=	≤	<	≠
Nombres	0	1	2	3	4	5	6	7	8	9	.	π	)	(	Ē	E
Majuscules	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						
Minuscules	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 .)

6-3. SYMBOLES D'ORGANIGRAMME

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'equivalence entre les variables.

Exemple) H(0)～H(9)→H～Q

INDEX DES COMMANDES/FONCTIONS

CARACTERISTIQUES

- **Type**
  PB-410/FX-720P/FX-820P
- **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 ↔ 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, DEFM, 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**
  Système de stockage utilisant une carte RAM
- **Langage de programmation**
  BASIC
- **Capacité de carte RAM**
  RC-2: 2K octets
  RC-4: 4K octets
  (incluant 272 octets de zone système et 208 octets de zone de variables fixes)
- **Capacité en programmes**
  Maximum de 10 programmes (dans les zones P0 à P9)
- **Nombre de variables**
  Minimum de 26 variables et l'exclusive variable alphanumérique ($)
- **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.

- **Eléments d'affichage**
  Affichage de 12 digits par matrice par points (cristaux liquides)
- **Composants principaux**
  C-MOS VLSI et autres
- **Alimentation**
  Unité centrale: 2 piles au lithium (CR2032)
  Carte RAM: 1 pile au lithium (CR2016)
  Imprimante à caractères intégrée (uniquement sur le FX-820P): accumulateur au Ni-Cd intégré
- **Consommation**
  Unité centrale: maximum 0,03 W
  Imprimante à caractères intégrée (uniquement sur le FX-820P): maximum 4 W
- **Longévité des piles (utilisation continue)**
  Unité centrale seulement (PB-410/FX-720P): environ 140 heures
  (FX-820P): environ 90 heures
  Avec des appareils optionnels branchés (PB-410/FX-720P): environ 70 heures
  (FX-820P): environ 80 heures
  Carte RAM (quand elle est stockée à part de l'unité centrale)
  RC-2: environ 2 ans
  RC-4: environ 1 an
  Imprimante à caractères intégrée (uniquement sur le FX-820P): impression d'environ 3000 lignes de "5555555555" en continu avec un accumulateur chargé au maximum.
- **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 et poids**
  PB-410/FX-720P — 14,3 mmH x 165 mmL x 82 mmP, 177 g (piles et carte RAM comprises).
  FX-820P — 26 mmH x 173 mmL x 95 mmP, 335 g (piles et carte RAM comprises).
  Carte RAM — 3,8 mmH x 60mmL x 50mmP, 17 g (pile comprise).

CASIO®

104B SA Imprimé au Japon