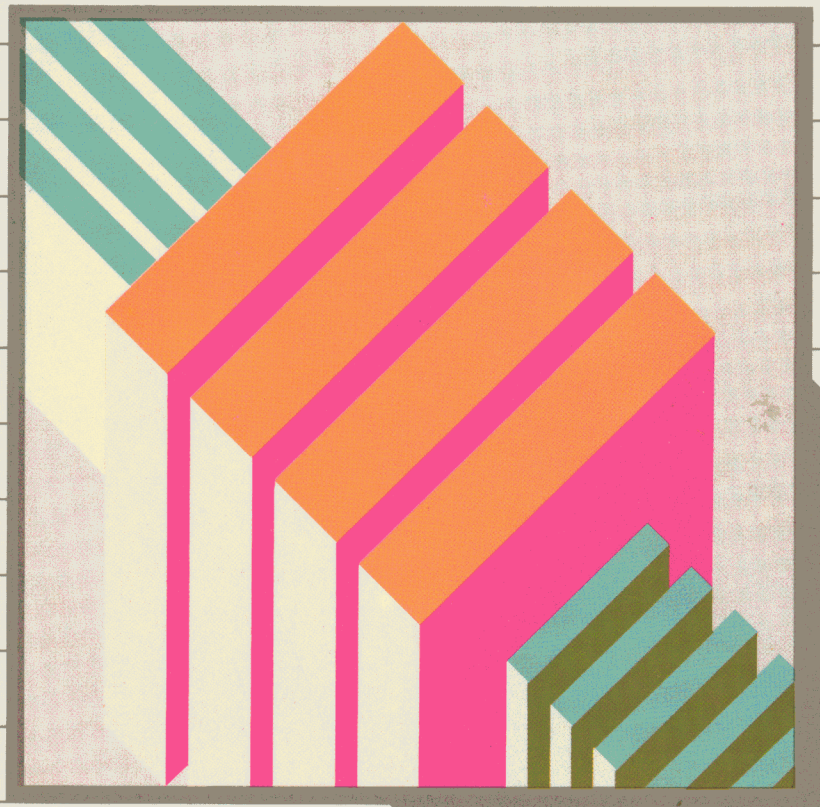
Apple *//e*

Travaux Pratiques Applesoft

Pour *//e* seulement



Avertissement

La Société Apple Computer, Inc. se réserve le droit d'apporter des améliorations au produit décrit dans le présent manuel à tout moment et sans préavis.

Limitation de garantie et responsabilités

Apple Computer, Inc. n'apporte aucune garantie, explicite ou implicite, en ce qui concerne le présent manuel ou le logiciel décrit dans ce manuel, sa qualité, ses performances, sa négociabilité, ou son aptitude à un usage particulier. Le logiciel d'Apple Computer, Inc. est vendu ou cédé en licence « en l'état ». L'intégralité du risque, quant à sa qualité et sa performance, est à la charge de l'acheteur. Si les programmes se révélaient défectueux à la suite de leur achat, l'acheteur (et non pas Apple Computer, Inc., son distributeur ou son détaillant) assumerait la totalité des frais des services nécessaires, réparation ou correction de tout dommage incident ou conséquent. En aucun cas Apple Computer, Inc. ne sera responsable de dommages directs, indirects, accidentels ou par voie de conséquences résultant d'un défaut quelconque du logiciel, même si Apple Computer Inc. a été avisé de la possibilité de tels dommages. Certains pays n'acceptent pas l'exclusion ou la limitation des garanties implicites ou des responsabilités pour des dommages accidentels ou par voie de conséquence, de telle sorte que la limitation ou exclusion ci-dessus ne pourront vous être appliquées.

Avertissement :

Cet équipement est certifié conforme aux restrictions des machines de calcul de la classe B, partie de Subpart J de la Part 15 des règles de la FCC. Les seuls périphériques (dispositifs d'entrée/sortie, ordinateurs, terminaux, imprimantes, etc.) certifiés conformes aux restrictions de la classé B peuvent être raccordés à cet ordinateur. Le fonctionnement en conjonction avec des périphériques non agréés peut être générateur d'interférences radio et télévision.

Rédigé par Joe Meyers, du Service Publications Apple PCS.

Numéro de produit Apple : A2L 2003F

Apple *IIe*

Travaux Pratiques Applesoft

traduit par Jean-Pierre Lamoitier

Parasites radio et télévision

L'équipement décrit dans ce manuel génère et utilise des signaux hautes fréquences. Si ce matériel n'est pas installé et employé convenablement, c'est-à-dire en suivant nos instructions à la lettre, il peut se produire des interférences avec la réception radio et télévision. Ces interférences ressemblent aux parasites sur une radio et aux images ''neigeuses'' à la télévision.

L'équipement a été testé ; il est conforme au matériel informatique de classe B, selon les spécifications du sous-titre J, 15e partie, des Règlementations de la Commission des Communications Fédérales (FCC). Ces règles sont conçues pour assurer une protection suffisante contre ces interférences dans une installation résidentielle. Toutefois, on ne peut garantir l'absence de telles interférences dans une installation donnée, en particulier si l'on utilise une antenne de télévision intérieure.

On peut déterminer si les interférences sont dues à l'ordinateur en l'arrêtant. Si les parasites s'arrêtent, c'est probablement qu'ils étaient dus à l'ordinateur. Si votre ordinateur provoque effectivement des interférences vis-à-vis de la réception radio/télévision, il est possible d'essayer d'y remédier, au moyen de l'une ou de plusieurs des mesures suivantes :

- Débrancher les périphériques et leurs câbles d'entrée/sortie les uns après les autres. Si les parasites s'arrêtent, c'est qu'ils proviennent soit du périphérique soit de son câble. Ces unités exigent généralement des câbles blindés. Vous pouvez vous procurer des câbles correctement blindés pour vos périphériques Apple auprès de votre concessionnaire. Si vos périphériques sont d'une marque autre que Apple, contacter le fabricant ou votre concessionnaire pour qu'il vous conseille.

- Orienter l'antenne de radio ou de télévision jusqu'à ce que les parasites s'arrêtent.

- Déplacer l'ordinateur, de part et d'autre du poste de radio ou de télévision.

- Éloigner l'ordinateur du poste de radio ou de télévision.

- Brancher l'ordinateur à une prise appartenant à un circuit différent de celui du poste de radio ou de télévision. (Ce qui revient à s'assurer que l'ordinateur et les postes de radio ou de télévision sont branchés sur des circuits commandés par des disjoncteurs ou des fusibles différents.)

- Songer à installer une antenne de télévision sur le toit, reliée par un câble coaxial à votre poste de télévision.

Si besoin est, consultez votre concessionnaire ou un technicien radio/télévision expérimenté pour qu'il vous suggère d'autres idées.

Le Guide :
''Comment identifier et Résoudre les Problèmes de Parasites Radio et Télévision'',
publié par la Communication Fédérale des Communications (FCC) pourra vous être utile.
Vous pourrez vous procurez ce guide à l'adresse suivante :
U.S. Government Printing Office, Washington, DC 20402.
Numéro de stock : 004-000-00345-4.

Applesoft Tutorial : Travaux Pratiques Applesoft

Les programmes proposés en exemple dans ce manuel sont disponibles sur la disquette « Applesoft Sampler ». Pour conserver la comptabilité avec cette disquette, les programmes ont été reproduits dans le manuel dans leur version originale.

Généralités
L'apprentissage de la programmation IX

IX Qu'est-ce que la programmation ?
X Comment préparer votre machine ?
XI Chose à savoir
XI La signification des mots
XI Les symboles
XII Où obtenir d'autres renseignements

La mise en œuvre d'Applesoft 1

4 Une première instruction
8 Questions sans réponse
8 Un mode d'affichage amusant
9 Pour faire des calculs
13 Que dire de ↵ ?
13 Un pas vers l'utilisation du graphique
19 Les messages d'erreur PLOT
20 Tracer des lignes
24 Variables et autres possibilités de l'ordinateur
26 Mode de reconnaissance d'une variable
29 Un résumé des règles concernant les variables
30 Pour gagner beaucoup de temps
30 Hiérarchie entre opérateurs
32 Des parenthèses
34 Résumé du chapitre

2 Premiers éléments de programmation 37

37 Mode programme
42 Boucles : l'instruction GOTO
45 Dialogue avec votre programme : INPUT
47 Vous voulez conserver vos programmes ?
47 Préparation
48 Sauvegarde
49 Les conditions : la recherche de la ''vérité''
51 Symboles utilisés dans les instructions conditionnelles
52 Règles d'utilisation des instructions conditionnelles
52 Boucles conditionnelles : l'instruction IF... THEN
54 Utilisation du programme Applesoft COLORLOOP
56 Compléments sur l'instruction IF... THEN
58 Remarques
59 Les boucles FOR/NEXT
62 Boucles emboîtées et enchevétrées
64 Contrôle de la présentation des résultats
70 Résumé du chapitre

3 Modification de programmes 73

73 Déplacement du curseur : mode escape
73 Règles d'utilisation du mode escape
74 Un exercice pratique
76 D'autres commandes dans le mode escape
76 Limites du mode escape
77 Pour insérer du texte dans une ligne existante
82 Pour se débarrasser des lignes d'un programme
83 Pour mettre à jour des programmes volumineux
84 Un peu d'histoire
86 Résumé des possibilités pour entrer du texte

4 Du graphique 89

89 Construction d'un jeu simple
90 Instructions multiples dans une ligne
91 Création du mouvement
92 Limite de l'écran
93 Création d'impressions visuelles
93 L'article entier
94 Programme avec intervention des utilisateurs
97 Production de bruits
100 Bruit de la balle au rebond
100 Nombres aléatoires
103 Simulation d'une paire de dés
103 Graphiques aléatoires

104 Les sous-programmes : assemblage des éléments
108 Tracés
109 Un meilleur programme de tracé de chevaux
110 Les erreurs
110 Les variables
111 Sous-programmes complémentaires
112 Un programme bien structuré
113 Graphique haute résolution
120 Résumé du chapitre

5 *Chaînes de caractères et tableaux* 123

123 Enchaînons les caractères
124 Fonctions chaîne de caractères
126 Pratique de programmation habituelle utilisant les chaînes de caractères
128 Duplication de chaînes de caractères
128 Epelons à l'envers
130 Concaténation
131 D'autres fonctions sur chaîne de caractères
134 Encore des pièges à erreurs
135 Introduction aux tableaux
139 Messages d'erreur concernant les tableaux
140 Sommaire du chapitre
141 Conclusion

A *Sommaire des instructions et commandes* 145

B *Mots réservés en Applesoft* 159

C *Messages d'erreur* 163

D *Aide* 167

167 Si votre programme (ou vous-même) êtes ''planté'' !
168 Erreurs
168 Instructions et commandes
168 Magnétophones à cassettes
169 Encore des informations utiles
169 Impression de programme Applesoft
169 La mémoire de l'Apple *II*e
170 Signification du caractère d'attente

E

Encore des programmes à essayer 171

172 Conseil pour les programmeurs novices
173 SCRAMBLER
174 Analyse des lignes du programme
177 Réglage fin
178 Programme
180 MAGIC MENU
181 Notes pour les programmeurs confirmés
181 Fonctionnement des cinq sous-programmes : démonstration
182 Le sous-programme INPUT
183 Le sous-programme GET RETURN
183 Le sous-programme Screen Formatter
184 Le sous-programme Menu Maker
186 Le sous-programme Computer Identifier
187 Autres remarques sur MAGIC MENU
188 Vitesse du programme
189 Quelques mots sur les noms de variables
190 Quelques remarques sur la logique
192 Programme
201 Le programme DISK MENU
203 Renumérotation et fusion de portions de programmes
205 Programme
211 CONVERTER
211 Programme
219 Quelques idées

Glossaire 221

Index 236

L'apprentissage de la programmation

Bonjour ! C'est le manuel qu'il vous faut si vous voulez apprendre à programmer ou vous familiariser avec la programmation. De toute façon, il est conçu pour que vous appreniez avec plaisir.

Vous apprendrez, par exemple, comment faire dire bonjour à vos amis par votre ordinateur, comment utiliser votre ordinateur comme un calculateur, et comment programmer votre ordinateur pour lui faire dessiner des chevaux sur tout l'écran.

Si vous commencez par le début, faites tous les exercices comme ils viennent et n'oubliez pas de prendre votre temps et alors, il est certain que vous apprendrez comment programmer. Le secret de la réussite est de prendre son temps et de ne rien laisser de côté.

Vous ne pouvez pas apprendre à programmer seulement par la lecture de ce livre ou d'un autre. La pratique est indispensable comme pour apprendre à faire de la bicyclette, pour faire pousser des légumes ou pour conduire une voiture. Faire des erreurs et les corriger est un facteur très important de la réussite. Pour programmer, c'est la même chose. Si vous suivez bien ce qui est indiqué dans ce manuel, vous ne devrez pas vous inquiéter si tout ne marche pas bien du premier coup. Il vaut bien mieux comprendre pourquoi tout n'a pas bien fonctionné.

Ce manuel est organisé de telle manière que vous puissiez apprendre chaque chose pas à pas. Il est également organisé pour aller à votre rythme. Les points d'arrêt sont notés dans le livre.

Qu'est-ce que la programmation ?

Un programme d'ordinateur est une suite d'instructions écrites dans ce que l'on appelle un *langage source.* C'est grâce aux programmes que les choses se passent. Ils peuvent enseigner à l'ordinateur comment afficher un message sur l'écran, comment calculer des taxes complexes, comment jouer ou comment dessiner. Les programmes, en un mot, vous permettent de faire des tas de choses avec votre ordinateur.

Le BASIC est un langage couramment utilisé par les ordinateurs individuels. Il est simple à apprendre et à utiliser. Il y a de nombreuses variétés de BASIC ; chaque type d'ordinateur individuel en utilise une version légèrement différente. C'est comme des dialectes régionaux : bien que les gens parlent différemment, ils utilisent le même vocabulaire et peuvent se comprendre.

Le BASIC Applesoft, qui est résident dans l'Appel *II*e, est une variété particulièrement puissante du BASIC. Il a des possibilité graphiques que n'ont pas beaucoup d'autres variétés de BASIC. Ce manuel apprend à mettre en œuvre le BASIC Applesoft, mais, bien qu'il soit spécifique à Applesoft, il vous permettra de comprendre d'autres versions du BASIC.

Apprendre un langage de programmation est comme apprendre un second langage : de nouveaux mots et des règles syntaxiques de liaison sont à connaître. Il y a environ 100 mots dans le langage de programmation BASIC Applesoft.

Comment préparer votre machine

Avant de commencer le chapitre 1, préparez votre machine et ses accessoires :

- Une console de visualisation et un lecteur de disquette doivent être connectés à Apple *II*e. Les câbles d'alimentation de l'ordinateur et de la console doivent être branchés dans une prise comportant une prise de terre. Si vous ne possédez pas cet accessoire reportez-vous au *Guide de l'Utilisateur Apple IIe* (*Apple IIe Owner's Manual*) pour les instructions d'installation.

- Assurez-vous que vous avez la disquette DOS 3.3 SYSTEM MASTER, la disquette APPLESOFT SAMPLER et une disquette vierge initialisée. Si vous ne savez comment préparer une nouvelle disquette à recevoir de l'information, reportez-vous au *Guide de l'Utilisateur Apple IIe.*

- Si vous utilisez un lecteur de cassette au lieu d'un lecteur de disquette, les commandes spéciales dont vous aurez besoin sont décrites dans le *Manuel de Référence du programmeur BASIC Applesoft (Applesoft BASIC Programmer's Reference Manual).*

- Si vous avez une Carte Texte 80 Colonnes, rendez-la inactive tant que vous utiliserez ce manuel : cela vous facilitera l'apprentissage d'Applesoft. Plus tard vous pourrez lire le Manuel Apple *II*e sur la *Carte Texte 80 Colonnes* pour voir comment Applesoft fonctionne avec cette carte active.

Chose à savoir

Cette présentation n'est pas un guide exhaustif d'Applesoft. Afin de simplifier les choses, ce qui n'a pas besoin d'être connu par le programmeur débutant a été passé sous silence. En d'autres termes, ne soyez pas surpris que tout ne soit pas enseigné. C'est l'objet du *Manuel de Référence du Programmeur BASIC Applesoft.*

La signification des mots

Les termes techniques qui peuvent ne pas vous être familiers sont imprimés en italique. Chaque mot en italique est défini dans le glossaire.

Il y a également plusieurs annexes qui vous aideront à comprendre la signification des mots.

- L'annexe A définit et résume les instructions Applesoft.
- L'annexe B liste les mots réservés (ne vous inquiétez pas — vous comprendrez ce qu'ils veulent dire le moment venu).
- L'annexe C commente les messages d'erreurs.

Les symboles

Il y a trois symboles particuliers utilisés dans ce manuel.

Le symbole de pause indique les endroits du texte où vous pouvez interrompre votre lecture. Ce n'est pas une obligation lorsque vous rencontrez ce symbole—C'est simplement une aide.

Le rectangle grisé est utilisé pour vous clarifier la lecture ou pour vous rappeler quelques techniques utiles. Pour le distinguer du texte principal, il est imprimé sur fond sombre.

Le symbole d'attention vous prévient qu'à cet endroit se trouve une information très importante que vous ne devez pas manquer. Il vous prévient également que vous avez quelque chose à faire pour la bonne exécution de vos programmes.

Note : les manipulations de clavier détaillées dans ce manuel sont valables pour le mode QWERTY.

Où obtenir d'autres renseignements ?

Avant de commencer l'étude de ce manuel, il vous faudrait lire le *Guide de l'Utilisateur Apple IIe* et lancer l'exécution de la disquette APPLE PRESENTS... APPLE.

Le manuel associé à cet ouvrage est le *Manuel de Référence du Programmeur Basic Applesoft.* Le manuel de référence comporte des indications détaillées et complètes sur Applesoft, commente l'architecture des programmes et donne de bons conseils de programmation. Servez-vous du manuel de référence pour perfectionner vos connaissances sur la programmation une fois que vous aurez acquis les connaissances de base dans ce manuel.

Le *Manuel DOS* contient des informations complètes concernant les commandes du système d'exploitation disque et la façon dont le système d'exploitation fonctionne. Quand vous aurez plus d'expérience dans la programmation, vous souhaiterez lire ce manuel.

La mise en œuvre d'Applesoft

4 Une première instruction
8 Questions sans réponse
8 Un mode d'affichage amusant
9 Pour faire des calculs
13 Que dire de ↵ ?
13 Un pas vers l'utilisation du graphique
19 Les messages d'erreur PLOT
20 Tracer des lignes
24 Variables et autres possibilités de l'ordinateur
26 Mode de reconnaissance d'une variable
29 Un résumé des règles concernant les variables
30 Pour gagner beaucoup de temps
30 Hiérarchie entre opérateurs
32 Des parenthèses
34 Résumé du chapitre

La mise en œuvre d'Applesoft

Maintenant au travail ! Recherchez la disquette intitulée DOS 3.3 SYSTEM MASTER. Placez-la dans le lecteur de disquette et fermez la porte du lecteur. Mettez votre ordinateur sous tension. Dès que le voyant rouge du lecteur de disquette s'éteindra, vous verrez apparaître un caractère d'attention *crochet (]) ainsi qu'un curseur* clignotant au coin gauche de votre écran.

Figure 1-1. Le curseur et le caractère d'attente.

Si vous ne les voyez pas, vérifiez votre installation. Le lecteur de disquette est-il bien connecté ? La console de visualisation est-elle branchée et mise sous tension ? Apple *IIe* est-il bien sous tension ?

Le BASIC Applesoft est résident dans Apple *IIe*, ainsi le lecteur de disquette n'est pas indispensable. Si vous utilisez un lecteur de cassette, reportez-vous au *Manuel de Référence d'Applesoft.*

Quand le curseur est apparu, enfoncez la touche ⇓ dans la position « on ». Elle se trouve dans le coin inférieur gauche du clavier. En actionnant la touche vous devez entendre un cliquetis moins important que pour les autres touches.

Les instructions du BASIC Applesoft doivent être introduites en majuscules. La touche ⇓ le facilite. Comme la touche majuscule pour les caractères d'une machine à écrire, ⇓ transforme toutes les lettres en majuscules, tout en vous laissant la possibilité de frapper des chiffres. Si vous avez besoin d'un caractère majuscule d'une touche non-alphabétique vous n'aurez à utiliser que la touche ⇑ Vous utiliserez, par exemple, la touche ⇑ pour accéder aux guillemets.

Attention

A chaque fois que vous utiliserez le BASIC Applesoft avec Apple *IIe*, assurez-vous que la touche ⇓ est activée. Si elle ne l'est pas et si vous essayez d'introduire une instruction en minuscules, votre ordinateur vous alertera par un signal sonore et par le message ?SYNTAX ERROR. Applesoft ne comprend que les instructions en majuscules.

Une première instruction

Vous êtes prêts à introduire quelques instructions dans Apple *IIe*, ainsi entrez les mots que vous voyez plus bas. Rappelez-vous que pour accéder aux guillemets vous devez appuyer sur la touche ⇧.
Tapez

```
PRINT ''HELLO''
```

et appuyez sur la touche ↵.

Caractère d'attention

PRINT ''HELLO'' est l'**instruction** Applesoft

PRINT est le **verbe** Applesoft

Hello s'affiche après avoir tapé sur la touche ↵ et le curseur se déplace sur la ligne qui suit la ligne affichée

Les lettres entre guillemets sont celles que vous voulez faire imprimer par Apple *IIe* (affichage sur l'écran).

Illustration 1-2. L'instruction PRINT. Quand vous introduisez une instruction comme celle-ci dans Apple *IIe* vous devez taper exactement ce que vous voyez dans ce manuel, y compris les espaces, les lettres majuscules, et les points de ponctuation.

Apple *IIe* devra répondre à vos instructions en affichant le mot

```
HELLO
```

sur la ligne suivante. S'il apparaît bien à la suite de vos instructions, alors bravo !

Si l'image de votre écran ne coïncide pas avec celle qui est présentée dans ce manuel, cela n'est pas grave. Recommencez. Regardez votre écran et vérifiez bien ce que vous avez tapé. Notez que chacune des remarques qui vous sont présentées plus bas correspond à un des exemples de l'illustration 1-3.

1. Avez-vous bien utilisé les lettres majuscules ? Si non, enfoncez la touche ⇩ et recommencez.

2. Le curseur clignote-t-il bien juste après le deuxième guillemet ? Si oui, tapez ↵ . (Chaque instruction doit être suivie par une frappe sur la touche ↵ pour indiquer à Apple *II*e que l'instruction est terminée).

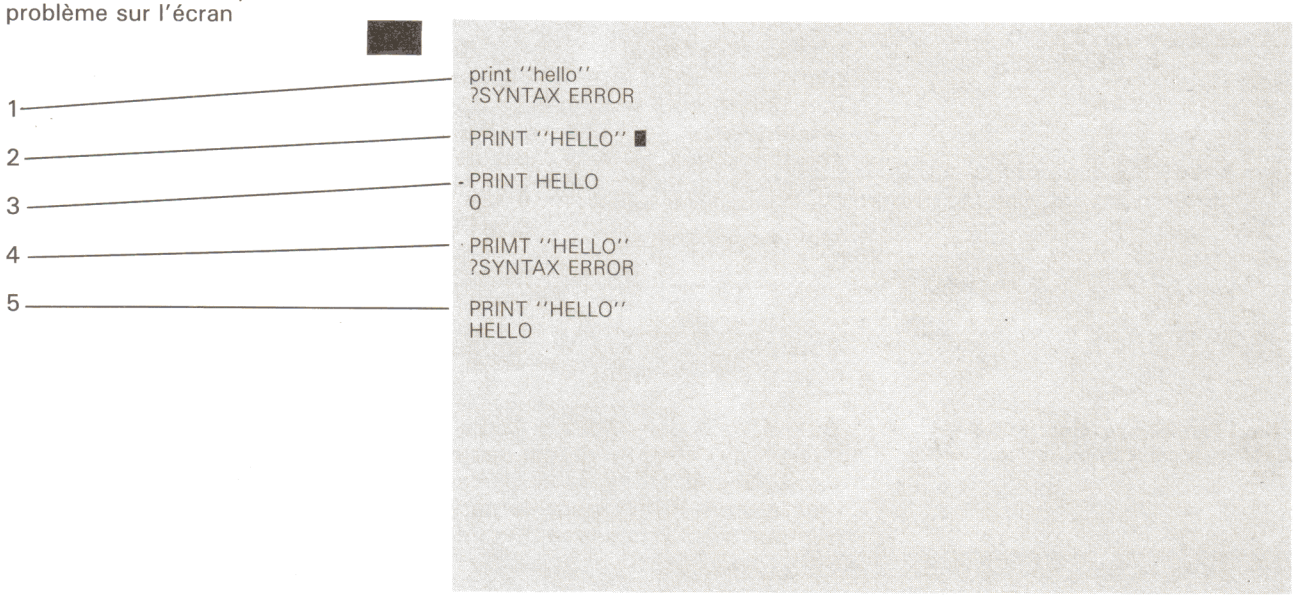
3. Y-a-t-il bien des guillemets de part et d'autre de HELLO? Si non, retapez l'instruction. Si c'est une apostrophe qui est affichée, assurez-vous que la touche ⇧ est bien enfoncée pour obtenir le guillemet.

   Si vous avez oublié le premier caractère de ponctuation, l'ordinateur affichera un zéro juste en dessous de votre instruction.

4. Le verbe PRINT est-il bien orthographié ? Si non, recommencez.

5. Maintenant tout est bon !

Illustration 1-3. Analyser un problème sur l'écran



Un **verbe** est un mot qui caractérise une instruction ou une commande particulière d'Applesoft. PRINT est un verbe.

Si un *verbe* est mal orthographié (comme PRINT) dans une instruction (PRINT ''HELLO'' est une instruction), vous recevrez le message d'erreur :

```
?SYNTAX ERROR
```

Une **instruction** est une suite de mots dans le langage de l'ordinateur.

Si vous n'avez pas détecté une faute d'orthographe avant l'apparition de ce message, il n'y a qu'une seule chose à faire, c'est de réintroduire l'instruction. Vérifiez bien que tout est correct avant d'actionner la touche ↵

Si vous détectez une erreur avant d'avoir tapé ↵ , vous pouvez la rectifier sans avoir à retaper la ligne complète. Actionnez la touche ← jusqu'à ce que le curseur se trouve sur l'erreur. Corrigez-la. Actionnez alors la touche → jusqu'à ce que le curseur soit revenu à la fin de la ligne. Tapez ↵ .

Un conseil d'ami : Une sage parole : personne n'est parfait. Cela vous prendra un bon laps de temps avant d'être complètement familiarisé avec le clavier de l'ordinateur et avant que vos doigts trouvent à chaque fois les bonnes touches. Seule la pratique vous aidera, donc ne vous découragez pas ! Dès que vous saurez utiliser les touches de déplacement du curseur, vous verrez que la correction des erreurs ne prend pas beaucoup de temps. Et, quand vous serez convaincu qu'il est nécessaire de vérifier chaque ligne avant la frappe de ↵ , que la vie est bien plus facile.

Maintenant tapez l'instruction suivante, en remplaçant les blancs par votre nom. Et n'oubliez pas de taper ↵ .

PRINT ''BONJOUR, MON NOM EST _____.''

Ça marche ? Sinon, ne vous découragez pas. L'ordinateur n'explosera pas si vous ne faites pas les choses exactement comme il faut. Il attend simplement que vous lui donniez une instruction qu'il reconnaisse. Contrairement à une personne, un ordinateur ne peut pas imaginer ce que vous voulez dire ; il ne comprend que certains mots dans un certain ordre. Vérifiez donc ce que vous tapez et recommencez.

Maintenant entrez

PRINT ''LE CIEL EST ROUGE, L'HERBE EST BLEUE''

Apple *II*e affiche exactement ce que vous lui avez demandé. Même si ce n'est pas vrai !

PRINT est l'illustration primaire Applesoft pour imprimer sur l'écran.

Quand vous demandez à Apple *II*e d'afficher quelque chose entre guillemets, vous lui demandez d'afficher sur l'écran tous les *caractères* se trouvant entre ces guillemets. Vous pouvez utiliser l'instruction PRINT pour demander à l'ordinateur d'afficher n'importe quel message que vous voulez. Essayez donc !

Si vous tapez plus de 240 caractères en une seule fois, (255 pour être exact) l'ordinateur vous répondra par un signal sonore et affichera le caractère barre oblique inverse (\) quand le nombre maximum de caractères est atteint. Vous devez alors recommencer. L'illustration 1-4 illustre ce dépassement en nombre de caractères.

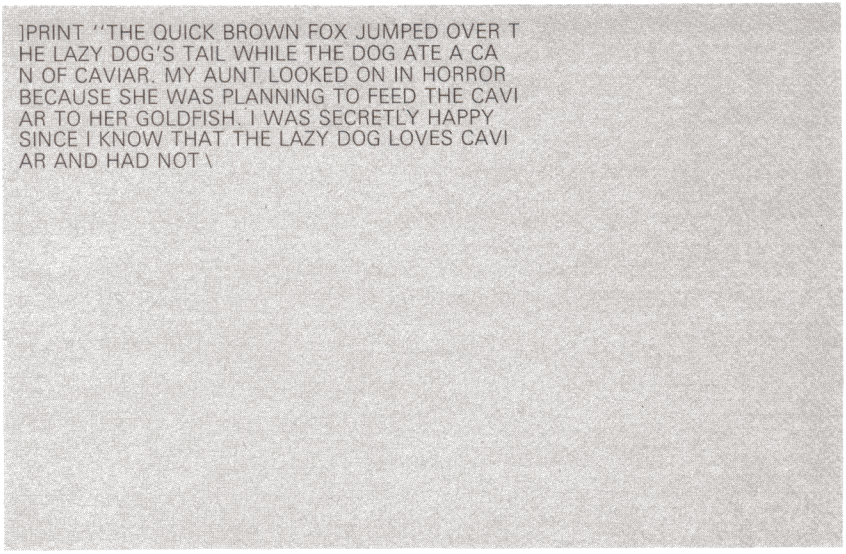


Illustration 1-4. Le nombre maximum de caractères sur l'écran. La largeur d'une ligne de l'écran est de 40 caractères. Il faut 6 lignes et demie de l'écran pour atteindre le nombre maximum de caractères d'une ligne Applesoft.

Si vous voulez que votre ordinateur affiche PRINT des caractères, vous devez utiliser les guillemets. Mais si vous entrez

```
PRINT 150
```

l'ordinateur affichera le nombre 150 sur la ligne suivante sans aucun message d'erreur concernant l'absence des guillemets. Tapez

```
PRINT ''150''
```

et il fera la même chose. Ceci n'est vrai que pour les nombres. Si vous essayez de taper

```
PRINT HELLO
```

vous obtiendrez un zéro.

Bien que l'instruction PRINT rende le même résultat dans les deux cas, il y a une différence très importante entre les caractères et le nombre. Cette différence deviendra beaucoup plus évidente lorsque vous en saurez plus sur la programmation.

Questions sans réponse

Il arrivera assez souvent que vous ayez des questions au sujet du BASIC Applesoft dont vous ne trouverez pas directement la réponse dans ce livre. Par exemple, dans l'instruction

```
PRINT ''HELLO''
```

doit-on laisser un espace après PRINT ?

Un simple essai vous apportera bien souvent la réponse. Taper

```
PRINT''HELLO''
```

et voyez ce qu'il se produit. Si vous prenez le temps de faire vous même ces essais, vous vous souviendrez bien mieux de ce que vous avez découvert que si, simplement, vous le lisez.

Si essais et erreurs ne suffisent pas, reportez-vous aux annexes de ce manuel ou au *Manuel de Référence d'Applesoft.*

Attention
Apple *IIe* consomme moins d'électricité qu'une ampoule électrique de 11 watts. S'il vous arrive de vous arrêter un court instant, laissez l'ordinateur sous tension. Vous devrez, cependant, éteindre la console de visualisation car elle consomme beaucoup plus d'électricité que l'ordinateur.

Si arrivé à un symbole de pause, vous avez éteint votre ordinateur, assurez-vous, lors du redémarrage, que la disquette DOS 3.3 SYSTEM MASTER est bien montée dans le lecteur.

Pause

Un mode d'affichage amusant

Maintenant que vous commencez à connaître l'instruction PRINT, quel autre essai aimeriez-vous faire ? Tapez

```
INVERSE
```

L'instruction INVERSE modifie le mode d'affichage vidéo de telle sorte que les caractères apparaissent en noir sur fond blanc.

et actionnez ↵ . Remarquez que le caractère d'attention apparaît différemment : il est noir sur fond blanc. Pour voir ce que fait INVERSE entrez

```
PRINT ''HELLO, HI, BONJOUR, BUENOS DIAS''
```

N'est-ce pas joli ? Essayez d'autres instructions PRINT de votre choix. Remarquez que vous n'avez pas à taper INVERSE à chaque fois. Une fois que vous avez demandé à Apple *II*e d'utiliser le noir sur fond blanc, ce sera toujours le cas jusqu'à ce que vous lui indiquiez d'arrêter. Pour cela, tapez

NORMAL modifie le mode d'affichage vidéo. Les lettres apparaissent en blanc sur un fond noir.

NORMAL

et toutes les instructions qui suivront seront en blanc sur fond noir.

Une autre instruction que vous pourrez utiliser si vous souhaitez effacer votre écran et ramener le curseur vers le coin supérieur gauche de l'écran, est

HOME ramène le curseur au coin supérieur gauche de la fenêtre de visualisation et efface l'écran.

HOME

Essayez-la. Ensuite entrez quelques instructions PRINT. Essayez HOME une nouvelle fois. Un écran vierge est à votre disposition.

Voici un autre essai. N'oubliez pas d'activer ↵ après avoir tapé une instruction. Tapez

INVERSE

HOME

PRINT ''MAINTENANT L'ECRAN EST VIERGE ET LES MOTS APPARAISSENT EN NOIR SUR FOND BLANC''

Essayez encore quelques instructions PRINT et voyez ce qu'il se passe.

Chacune de ces instructions demande à Apple *II*e d'effectuer une fonction spécifique : HOME efface l'écran et ramène le curseur au coin supérieur gauche ; INVERSE provoque l'affichage des lettres en noir sur fond blanc, et NORMAL ramène l'affichage sur l'écran en lettres blanches sur fond noir.

Pour faire des calculs

A ce point de l'apprentissage, vous pouvez utiliser Apple *II*e comme une simple calculatrice de bureau.

Essayez de faire cela avec Apple *II*e :

PRINT 3 + 4

La réponse, 7, apparaît sur la ligne suivante.

Apple *IIe* connaît les cinq différentes *opérations arithmétiques* de base :

1. L'**addition** est indiquée par le signe habituel plus (+).

2. La **soustraction** utilise le signe conventionnel moins (−). Pour faire un essai de soustraction, tapez

   PRINT 1086-99

3. La **multiplication** est indiquée par un astérisque (*). (Si un X avait été utilisé pour la multiplication il aurait pu y avoir une confusion avec la lettre X). Pour trouver le résultat de 7 fois 8 (au cas où vous ne vous souviendriez plus de la réponse), tapez seulement

   PRINT 7 * 8

   et votre mémoire sera rafraîchie.

4. La **division** est indiquée par une barre oblique (/). Pour diviser 63 par 7, tapez

   PRINT 63 / 7

   et la réponse correcte s'affichera.

   Essayez de diviser 3 par 2. Apple *IIe* vous donne la réponse sous forme décimale : 1.5.

5. L'**exponentielle** est indiquée par un accent circonflexe (^). Il est souvent fastidieux de multiplier un nombre par lui-même un certain nombre de fois. Au lieu d'entrer

   PRINT 4 * 4 * 4 * 4 * 4

   vous pouvez entrer l'expression plus compacte

   PRINT 4 ^ 5

   Pour atteindre l'accent circonflexe (flèche pointant vers le haut), enfoncez la touche ⇧ tout en appuyant sur la touche 6.

   Il n'y a rien à dire de particulier sur la forme exponentielle si ce n'est que c'est une abréviation d'une suite de multiplications. En notation classique (autre que celle de l'ordinateur), elle est désignée par 4 à la puissance 5 et on l'écrit ainsi :

   4^5

Vous pouvez combiner plusieurs de ces opérations entre elles dans la même instruction. Par exemple, vous pouvez demander à Apple *IIe* de calculer

PRINT 3 + 51 + 9 + 400

ou

PRINT 3 + 5 / 9 * 4 ^ 2 – 1

```
PRINT 3 + 51 + 9 + 400
463

PRINT 3 + 5 / 9 * 4 ^ 2 – 1
10.8888889
```

Les règles prévues pour un tel calcul sont décrites plus loin dans ce chapitre, mais vous pouvez l'essayer dès maintenant si vous le voulez.

Rappelez-vous, qu'Applesoft ne peut pas faire des opérations avec des nombres entre guillemets.

Si vous tapez

PRINT 45.340

votre ordinateur vous répondra

```
45.34
```

Sans afficher le zéro de gauche. Apple *IIe* n'affiche ni les zéros en tête, ni les zéros de gauche, c'est-à-dire, ceux qui se trouvent au début du nombre à gauche du point décimal et ceux qui se trouvent à la fin du nombre à droite du point décimal.

Les très, très petits nombres (entre environ .00000000000000000000000000000000000003 et –.00000000000000000000000000000000000003) seront convertis en zéros par Apple *IIe*. Ne nous prenez pas au mot pour cela. Essayez-le vous même. Une façon plus simple d'écrire ces nombres est
PRINT 3 * 10 ^ – 39 et PRINT – 3 * 10 ^ – 39.

Maintenant tapez

PRINT 985788.6898

Oh surprise ! Les deux derniers chiffres ont disparu et le nombre à gauche de ces chiffres est la plus juste approximation possible réalisable par l'ordinateur. Ce procédé est appelé un arrondi. Essayez de taper

```
PRINT 788.6898
```

Apple *II*e n'a pas arrondi le nombre, mais l'a affiché exactement comme vous l'avez entré. Il est fou, pensez-vous ? Ah, il y a parfois de la logique dans ce qui paraît être une folie. Les nombres ne sont arrondis que s'ils comportent plus de neuf chiffres. Tout nombre comportant moins de dix chiffres ne sera pas arrondi. Applesoft fait de son mieux, mais il ne peut travailler qu'avec neuf chiffres.

Si vous entrez une instruction avec un nombre très grand comme

```
1234567890
```

Apple *II*e vous répondra sous la forme

```
1.23456789E+09
```

Si vous affichez le nombre 10 millions (qui a 10 zéros), Apple *II*e vous répondra sous la forme

```
1E+10
```

Les nombres 10000000000 et 1E+10 et les nombres 1234567890 et 1.23456789E+09 représentent la même valeur. Vraiment. Le nombre affiché par votre ordinateur est sous la forme appelée *notation scientifique*. Si vous avez besoin de tels nombres vous savez sans doute comment les lire. Reportez-vous au *Manuel de Référence d'Applesoft* pour des informations supplémentaires ou si vous êtes intéressé par cette notation.

Que dire de ↵ ?

Précédemment, vous avez actionné la touche ↵ après chaque ligne. Vous aimeriez sans doute savoir pourquoi cette touche agit lorsqu'elle est frappée. La raison en est simple : sans le ↵, l'ordinateur ne peut pas savoir quand vous avez terminé une instruction. Par exemple, si vous aviez tapé

```
PRINT 4 + 5
```

et que l'ordinateur vous eut immédiatement répondu 9, vous pourriez être surpris de n'avoir pas pu introduire toute l'instruction

```
PRINT 4 + 5 + 346
```

qui donnerait un résultat totalement différent. Puisque l'ordinateur ne peut pas, de lui-même, détecter la fin d'une instruction, vous devez la lui indiquer. Ceci s'obtient en actionnant la touche ↵.

Attention

Puisque vous avez toujours dû actionner la touche ↵ après avoir introduit une instruction, ce manuel ne vous le rappelera plus. Et, si vous avez bien réalisé tous les exercices, frapper ↵ doit vous êtes devenu habituel.

Pause

Un pas vers l'utilisation du graphique

Jusqu'à présent, nous avons joué avec des mots et des nombres. Il est temps, maintenant, de changer de sujet. Continuez la lecture et vous découvrirez quelques unes des merveilleuses possibilités d'Apple *II*e.

Pour préparer l'écran au dessin, tapez

L'instruction GR prépare la session de travail en mode graphique basse-résolution

```
GR
```

Quand vous entrez GR, abréviation de "graphique", vous remarquerez qu'un dessin s'affiche un très court instant puis, l'écran s'efface, et le curseur se déplace au bas de l'écran. (Si votre écran ne s'efface pas, c'est que vous avez sans doute oublié d'actionner ↵ . Et n'oubliez pas que l'on ne vous le mentionnera plus !).

L'instruction GR demande à l'ordinateur de construire une grille invisible de 40 colonnes verticales et de 40 lignes horizontales où vous pourrez dessiner. Cet état s'appelle le *mode graphique basse résolution.* L'instruction GR demande également à l'ordinateur de laisser assez de place dans la bas de l'écran pour quatre lignes de texte, appelée *fenêtre de texte.* GR efface aussi l'écran et sélectionne la couleur noire.

Figure 1.5. Le mode graphique basse-résolution

L'instruction GR initialise la session de travail de façon à pouvoir faire des dessins très divers et de motifs visuels amusants. Cependant, avant de pouvoir regarder sur l'écran les résultats de ce que vous avez fait, vous devez indiquer à l'ordinateur d'utiliser une couleur qui s'affichera sur le fond noir. Essayez donc cela, en tapant les deux instructions suivantes :

```
COLOR = 15
PLOT 20,20
```

Pour sélectionner la couleur dans le mode graphique basse résolution utilisez l'instruction COLOR=, suivie par un nombre entier compris entre 0 et 15. L'instruction PLOT positionne une ''brique'' à l'endroit indiqué.

Voyez-vous la brique blanche (un petit rectangle) au milieu de l'écran ? Pour placer deux autres briques lé long de la même ligne verticale, tapez

```
PLOT 20,22
PLOT 20,24
```

Note : Au cas où rien n'apparaîtrait sur l'écran, il se pourrait fort bien que vous ayez oublié de spécifier une couleur. Vous devez absolument le faire quel que soit le type d'écran connecté.

L'instruction PLOT indique à l'ordinateur où il doit placer la brique de couleur en l'associant à deux nombres sur la grille invisible. Le premier nombre dans la troisième instruction PLOT que vous avez utilisé précédemment, 20, désigne la 20e des 40 colonnes. Le numéro de la colonne est toujours en premier dans l'instruction PLOT. Le second nombre (vous avez indiqué 20, 22 et 24) désigne le numéro de la ligne.

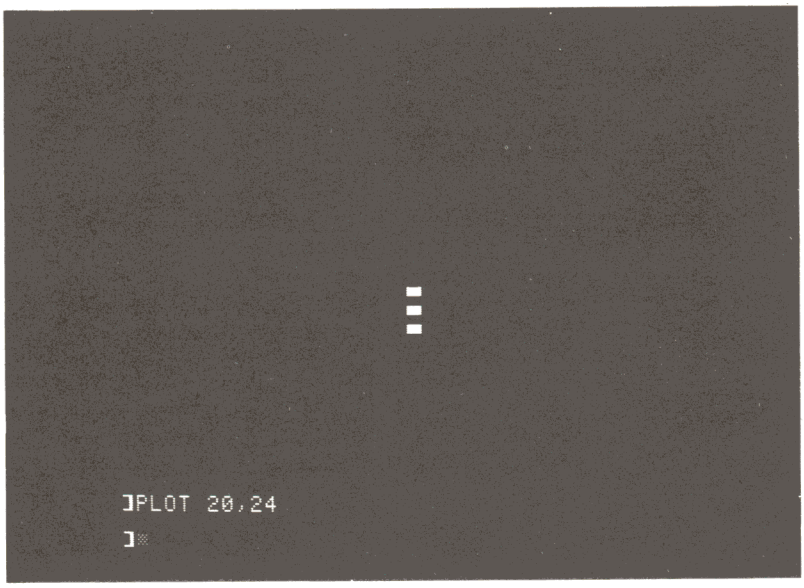


Figure 1.6. Comment utiliser l'instruction PLOT.

Maintenant essayez de placer quelques briques sur votre écran. Pouvez-vous en mettre quelques-unes dans la colonne la plus à gauche ? (Un conseil : le premier nombre après PLOT devra être 0, bien que le nombre 1 marche tout aussi bien). Que dire de la colonne à l'extrême droite ? (Réponse : le premier nombre après PLOT devra être 39).

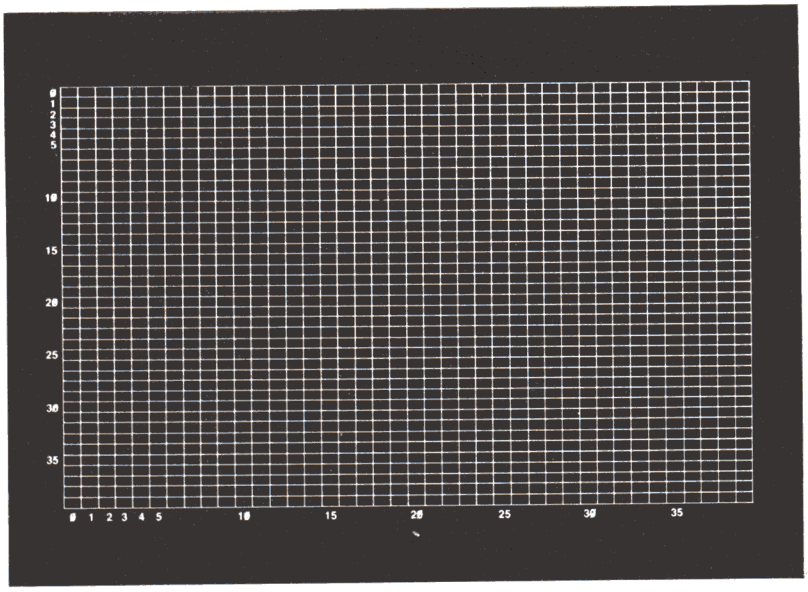


Figure 1.7. Le mode de numérotation de la grille du graphique basse résolution.

Comme vous pouvez le voir sur la figure 1.7, le mode de numérotation des colonnes débute par 0 et s'étend jusqu'à 39, de gauche à droite. La numérotation des lignes s'étend également de 0 à 39, de haut en bas de l'écran. Ceux qui parmi vous, connaissent l'algèbre, reconnaîtront les coordonnées cartésiennes dans ce mode de numérotation. Ce livre se réfère aux coordonnées pour désigner les colonnes et les lignes. Puisque la grille est déjà divisée en 40 colonnes et 40 lignes, tout ce que vous avez à faire est d'indiquer les coordonnées du point qui convient dans chaque instruction PLOT.

Vous choisissez la couleur des briques grâce à l'instruction COLOR=. Apple *II*e conservera cette couleur tant que vous ne lui aurez pas demandé d'en changer. Essayez maintenant en entrant

```
COLOR = 5
```

Remarquez qu'à ce point, l'écran n'a pas changé. Il faut indiquer à Apple *II*e où placer chaque brique. Tapez donc

```
PLOT 20,21
```

Et qu'arrive-t-il ? Une brique d'un ton différent devrait apparaître dans la colonne déjà utilisée, mais sur une autre ligne. Poursuivez par

```
PLOT 20,23
PLOT 20,0
```

Deux autres briques devraient apparaître dans la même colonne.

Précédemment, si vous avez réalisé les exercices, vous avez fait apparaître sur l'écran des briques de deux tons différents. Comment faut-il procéder ?

Il y a 16 nombres associés à COLOR= dans Applesoft. Regardez la figure 1-8 pour voir à quel nombre est associé telle couleur.

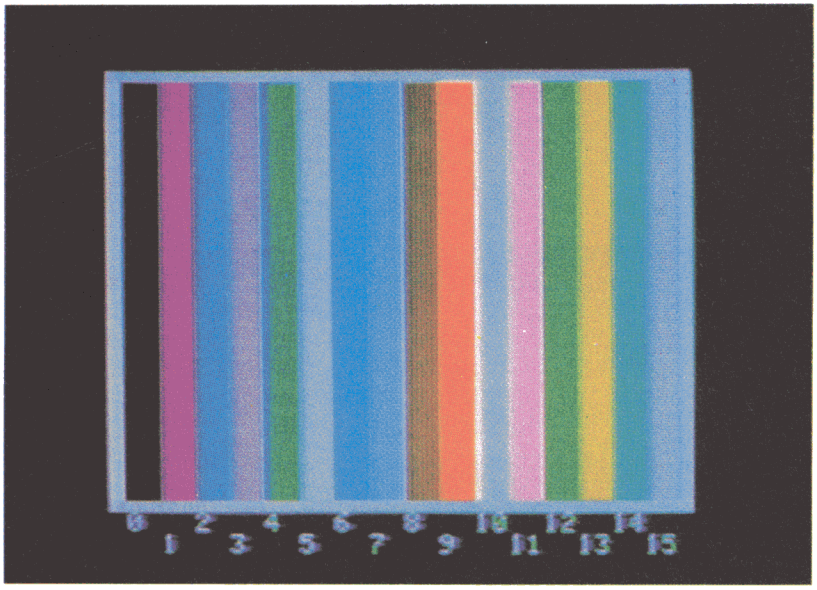
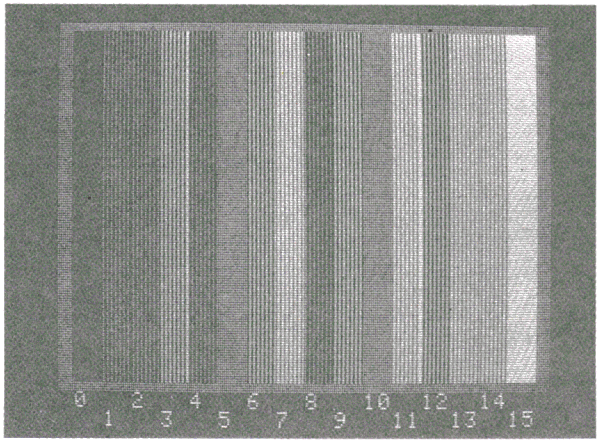
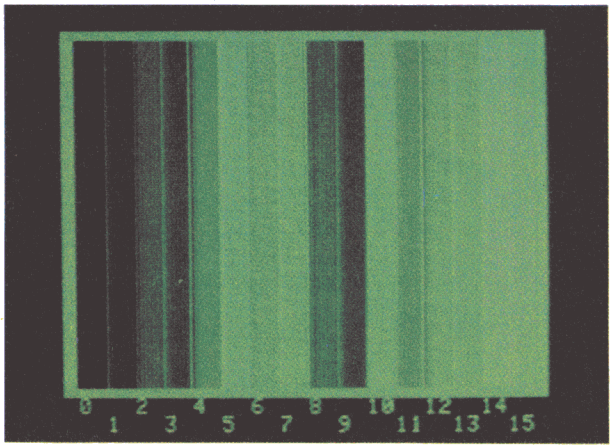


Figure 1.8. Noms des couleurs d'Applesoft et nombres associés dans COLOR

0	noir	8	brun
1	magenta	9	orange
2	bleu foncé	10	gris
3	violet	11	rose
4	gris foncé	12	vert
5	gris	13	jaune
6	bleu	14	vert eau
7	bleu clair	15	blanc

Figure 1.9. Groupes de couleurs. Pour obtenir un bon contraste sur une console noir et blanc ou au phosphore vert, utilisez un nombre de chaque groupe.





Gris foncé : 1, 2, 4, 8
Gris : 5, 10
Gris clair : 3, 6, 9, 12
Gris très clair : 7, 11, 13, 14
Blanc : 15

Si vous utilisez une console noir et blanc ou au phosphore vert, il y a cinq groupes de couleurs qui vous permettront d'obtenir un bon contraste sur l'écran. Toutes les seize couleurs peuvent être utilisées et vous devez vous sentir totalement libre de vous en servir.

Pour ceux qui possèdent un téléviseur couleur : vous pouvez connecter un téléviseur couleur à Apple IIe grâce à un modulateur de fréquence radio . Tous les exemples de ce manuel, relatifs au graphisme, peuvent être joliment réalisés en couleur. Vous n'aurez pas à vous limiter aux groupes de couleurs car les 16 couleurs se marient bien sur un écran couleur.

Des briques apparaissent sur votre écran, placées grâce à l'instruction COLOR = 15 et COLOR = 5. Essayez ensuite ces instructions pour faire apparaître de nouveaux tons :

```
COLOR = 3
PLOT 21,0
PLOT 21,20

COLOR = 1
PLOT 22,22
PLOT 22,25
PLOT 22,26
```

Pour comprendre pourquoi les couleurs de nombres 1, 2, 4 et 8 font partie du même groupe, essayez de placer des briques de ces différentes couleurs côte à côte. Sur un écran noir et blanc ou au phosphore vert, vous ne détecterez pas de grandes différences entre elles. Elles sont, bien sûr, très différentes sur un écran couleur (magenta, bleu foncé, gris foncé et brun). Poursuivez par

```
COLOR = 0
PLOT 22,27
```

Rien ne se passe, n'est-ce pas ? Jetez un coup d'œil sur la Figure 1-8 pour en comprendre la raison. Il se produira la même chose si vous entrez l'instruction GR sans indiquer une couleur particulière. Ceci, parce que la couleur est initialement à zéro ou au noir. Vous en souvenez-vous ?

Maintenant entrez quelques instructions COLOR= et PLOT. Et recommencez l'exercice jusqu'à ce que cela vous devienne familier.

Si vous voulez effacer l'écran et entrer d'autres instructions PLOT, tapez

```
GR
```

Quand vous êtes en mode graphique, HOME efface seulement la fenêtre de visualisation. GR doit être utilisé pour blanchir l'espace écran servant au graphisme.

Les messages d'erreurs PLOT

Il y a deux messages d'erreurs qui surviennent souvent lorsque vous utilisez l'instruction PLOT. Vous savez, sans doute, déjà que si vous tapez

PLAT

ou

PLOP

au lieu de

PLOT

Vous obtiendrez le message

?SYNTAX ERROR

Un message d'erreur différent s'affiche si vous utilisez un nombre plus grand ou plus petit que ceux qui sont autorisés pour définir les coordonnées dans PLOT. Tapez

PLOT 13,85

et le message apparaît

?ILLEGAL QUANTITY ERROR

Ce message signifie que vous avez essayé d'atteindre un point en dehors des limites de l'écran. Le nombre le plus grand que vous pouvez utiliser dans l'instruction PLOT est 39 car les coordonnées vont de 0 à 39.

Exception : Il y a, en fait, plusieurs façons d'accéder à un numéro de ligne supérieur à 39. Elles sont explicitées dans le *manuel de référence d'Applesoft.*

Essayer de donner une valeur négative dans l'instruction PLOT est une autre façon d'obtenir le

?ILLEGAL QUANTITY ERROR

message.

Pause

Tracer des lignes

Préparer votre écran pour de nouveaux graphismes en tapant

```
GR
COLOR = 1
```

Vous vous êtes sans doute rendu compte, à ce stade de l'étude, qu'il faut un grand nombre d'instructions PLOT pour tracer une ligne sur l'écran. Il vous faudra, par exemple, 40 instructions pour tracer une ligne horizontale barrant le milieu de l'écran ;

```
PLOT 0,20
PLOT 1,20
PLOT 2,20
```

et ainsi de suite jusqu'à

```
PLOT 39,20
```

Cependant, il y a une façon plus simple de tracer des lignes horizontales. Tapez simplement

L'instruction HLIN permet de tracer des lignes horizontales en mode graphique basse résolution.

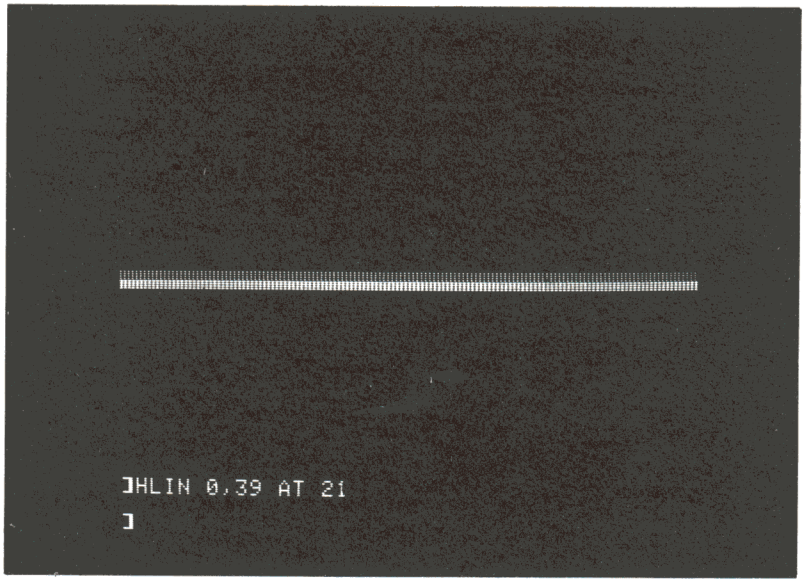
```
HLIN 0,39 AT 20
```

Et c'est fait : vous obtenez instantanément une ligne de la colonne 0 à la colonne 39 sur la ligne 20.

Pour tracer une ligne horizontale dans une autre couleur, juste en dessous de la première, entrez

```
COLOR = 7
HLIN 0,39 AT 21
```

Figure 1-10. L'instruction HLIN



Maintenant vous pouvez trouver comment tracer une troisième ligne d'un autre ton sur la ligne 22. Mais, cette fois, elle doit débuter en colonne 10 pour se terminer en colonne 30.

Pour comprendre l'instruction HLIN, examinez avec soin l'ordre des nombres qui suivent HLIN. La *syntaxe,* ou les règles d'écriture de l'instruction, nécessitent la présence de trois nombres dans un certain ordre. La troisième ligne, par exemple, s'étend de la colonne 10 à la colonne 30 sur la 22e ligne et est obtenue par l'instruction de la Figure 1-11.

Figure 1-11. La syntaxe de HLIN

HLIN 10, 30 AT 22

Du numéro de colonne 10 (à gauche) au numéro de colonne 30 (à droite) sur la ligne de numéro 22.

Pour voir comment les instructions PLOT et HLIN peuvent être utilisées conjointement, essayez ce qui suit :

```
COLOR = 8
HLIN 29,39 AT 35
HLIN 29,39 AT 37
COLOR = 10
PLOT 29,36
PLOT 39,36
```

Il existe pour les colonnes une instruction similaire à celle utilisée pour les lignes horizontales. Pour tracer un autre rectangle, encore plus grand, entrez

Dans les graphiques basse résolution, VLIN trace une ligne verticale dans la couleur indiquée par la plus récente indication COLOR =

```
COLOR = 15
VLIN 0,20 AT 0
VLIN 0,20 AT 15
COLOR = 5
HLIN 0,15 AT 0
HLIN 1,14 AT 20
```

Remarquez qu'au point où les lignes horizontales croisent les lignes verticales, c'est la nouvelle couleur qui reste affichée alors que l'ancienne disparaît. Entraînez-vous à tracer d'autre lignes verticales. Puis, pour effacer l'écran, utilisez l'instruction GR.

Maintenant, testez vos dons pour les lignes horizontales et verticales en traçant la bordure de l'écran en cinq instructions seulement. Dessinez un parcours sur l'écran. Amusez-vous avec les instructions PLOT, HLIN et VLIN jusqu'à ce que vous arriviez à placer les lignes exactement où vous le voulez.

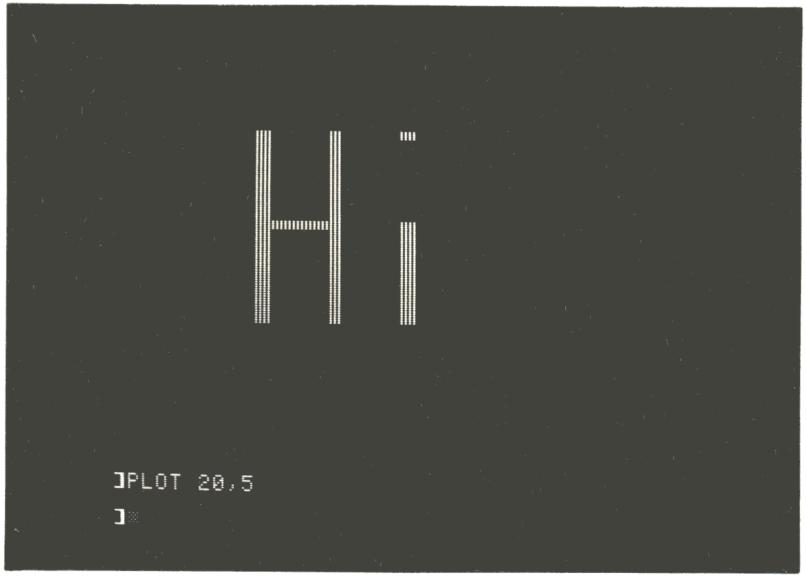
Maintenant voici un test. Regardez la Figure 1-12. Trouvez les coordonnées de la grille permettant de tracer ''Hi'' * sur votre écran avec une instruction HLIN et trois instructions VLIN. Débutez par

```
COLOR = 3
VLIN 5,25 AT 10
```

et trouvez le reste. Faites le point du ''i'' à l'aide de l'instruction PLOT. Vous pouvez, si vous le voulez, ajouter un point d'exclamation en colonne 28 (pour animer le tout).

(*) Hi est une interjection anglaise pour dire bonjour.

Figure 1-12. L'utilisation des coordonnées de la grille du mode graphique basse résolution



L'instruction TEXT initialise l'écran en mode texte (l'alternative du mode graphique) : 40 caractères par ligne et 24 lignes. S'il est utilisé pour sortir du mode graphique il vaut mieux l'utiliser conjointement avec l'intruction HOME.

Dès que vous aurez terminé vos essais en mode graphique, vous pourrez ramener l'ordinateur en mode texte afin de pouvoir affecter l'écran à vos programmes. En tapant

```
TEXT
```

vous verrez apparaître des symboles bizarres sur tout l'écran. Le terme technique qui le décrit est *garbage.* C'est l'indication qu'Applesoft passe du mode graphique au mode texte. Pour supprimer le *garbage* de votre écran, faites suivre l'instruction TEXT par une instruction HOME. Comme vous le savez déjà, HOME efface tout texte de votre écran.

Bien que ce manuel abandonne l'étude du mode graphique pour un court moment, assurez-vous, avant de continuer, qu'Apple *IIe* a bien été ramené en mode texte.

Pause

Variables et autres possibilités de l'ordinateur

Il y a, dans la mémoire principale de votre Apple *II*e, un grand nombre de zones particulières. Dans chacune de ces zones, vous pouvez ranger soit un nombre, soit le résultat d'une multiplication, soit des mots, soit encore une suite de caractères. Vous y arriverez par l'intermédiaire de symboles représentatifs des différentes adresses des zones de rangement.

C'est comparable à la mémorisation d'un nombre sur une simple calculatrice en vue d'une utilisation ultérieure. Habituellement sur une calculatrice, on y parvient en actionnant la touche ''mémoire''. Dans Apple *II*e, comme il y a bien plus de zones mémoires que dans une calculatrice, il faut donner un nom à chaque zone.

Supposons que vous vouliez sauvegarder le nombre 77. Si vous voulez baptiser la zone mémoire A, vous le demanderez à l'ordinateur en entrant

L'instruction LET sert à définir une variable.

```
LET A = 77
```

Le nombre, ou sa valeur, 77, n'est pas affiché. Il est rangé dans la zone que vous avez appelé A. Maintenant, si vous tapez

Si l'instruction LET A = 77 a disparu de votre écran, c'est que vous avez oublié de quitter le mode graphique par l'instruction TEXT. Si vous avez des signes incompréhensibles sur l'écran, c'est que vous avez oublié d'effacer l'écran avec l'instruction HOME.

```
PRINT A
```

l'ordinateur affichera la valeur de la variable A, qui est 77.

Continuez en tapant les deux instructions précédentes.

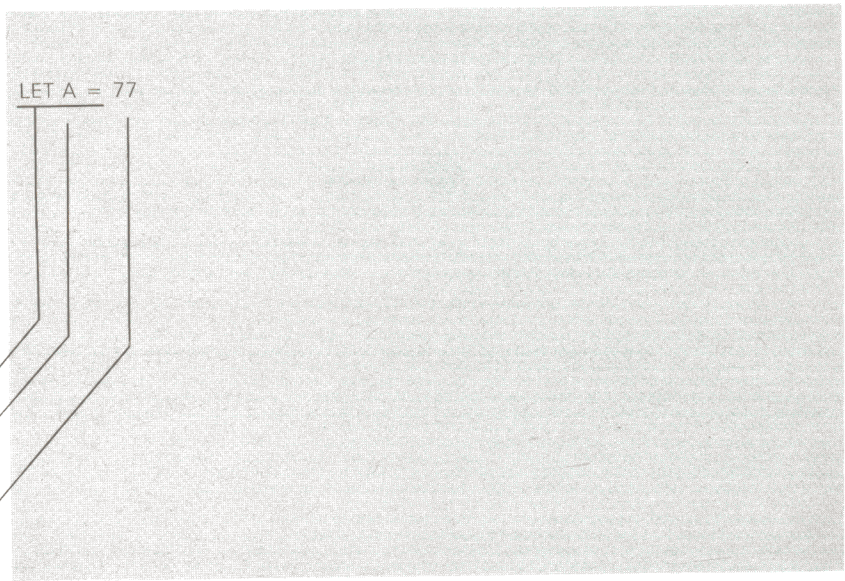


Figure 1-13. La définitition d'une variable avec LET. Une **variable** est un symbole représentatif d'une adresse mémoire. Vous devez vous l'imaginer comme une zone où une valeur, par exemple 77, serait rangée.

Une variable peut avoir presque n'importe quel nom pourvu qu'elle débute par une lettre. Par exemple

```
LET A = 77
LET ROUGE3 = 77
LET VETEMENT = 77
```

Chacune de ces variables représente une adresse différente de la mémoire. Elles mémorisent toutes la même valeur, 77. Maintenant si vous tapez

```
VETEMENT = 100
```

et si vous affichez la valeur de VETEMENT, vous obtenez 100, n'est-ce pas ? 77 a disparu et vous avez affecté une nouvelle valeur à VETEMENT.

Tant que vous mettrez une nouvelle valeur dans une variable, l'ancienne valeur sera effacée de la mémoire.

Cependant, les variables A et ROUGE3 contiennent toujours 77. Essayez de taper

```
PRINT ROUGE3
```

pour voir. La même chose est-elle vraie pour la variable A ?

Il se peut que vous ayez remarqué que dans l'instruction VETEMENT = 100, vous n'avez pas tapé LET. LET est facultatif lorsqu'il s'agit de désigner une variable. En d'autres termes, les instructions

```
LET A = 45
```

et

```
A = 45
```

donnent le même résultat. Il est parfois plus facile de reconnaître une variable, à la première lecture d'Applesoft, si elle est précédée de LET. Mais ce n'est pas une obligation.

Vous devez sans doute vous dire, en pensant à la fatigue de vos doigts, qu'il vaut mieux utiliser des noms abrégés comme noms de variables. Vous découvrirez également que certains noms ne sont pas autorisés car ils ont, en eux-mêmes, un sens particulier pour Applesoft. On les appelle les mots réservés. L'un d'entre eux est COLOR. Ainsi, un nom de variable ne devra jamais comporter cette suite de lettres dans son orthographe. Si vous tapez

```
THISCOLOR = 6
```

ou

```
COLORFUL = 9
```

Vous obtenez le message d'erreur ?SYNTAX ERROR qui vous signale que vous avez inopinément formé un nom avec un mot réservé. Ce n'est pas grave. Choisissez un autre nom. En prenant un peu de liberté avec l'orthographe des mots (en anglais dans cet exemple) vous pourrez renommer les variables de façon qu'elles gardent tout leur caractère mnémotechnique : COLOURFUL.

Une liste des mots réservés, qui ne peuvent pas être utilisés seuls ou en formation d'un nom de variable, se trouve dans l'annexe B. Le *Manuel de Référence d'Applesoft* contient d'autres commentaires sur les variables.

Mode de reconnaissance d'une variable

Applesoft distingue deux variables en comparant les deux premiers caractères formant leur nom.

Pour bien le comprendre, tapez

```
OISEAU = 11
PRINT OISEAU
```

Est-ce bon ? Poursuivez par

```
PRINT OISILLON
```

Qu'arrive-t-il ? Tapez

```
PRINT OIE
```

Tous ces noms commencent par OI. Ainsi les noms OISEAU, OISILLON et OIE se rapportent tous à la même variable.

Applesoft interprète de façon différente les instructions

PRINT CHAT

et

PRINT ''CHAT''

Introduisez-les. Que se passe-t-il ?

Dans la première instruction, le mot CHAT est le nom d'une variable alors que dans la deuxième le mot CHAT est la suite des lettres permettant d'orthographier le nom de cet animal. C'est un exemple d'utilisation des guillemets par Applesoft.

Si vous oubliez les guillemets dans l'instruction PRINT, Applesoft considèrera que le mot, ou la lettre, est une variable et tentera d'afficher la valeur de cette variable.

A ce point de l'étude, vous n'avez utilisé les variables que pour mémoriser un nombre. Vous pouvez également les utiliser pour mémoriser le résultat d'une opération. Entrez les instructions :

```
A = 4 + 5
PRINT A
```

La valeur de A est 9, n'est-ce pas ? Maintenant vous pouvez utiliser A dans une autre opération. Par exemple, tapez

PRINT A + 2

Est-ce bien la réponse escomptée ? Tapez quelques instructions supplémentaires qui utilisent la variable A.

Applesoft extrait tout ce qui se trouve à droite du signe (=), l'interprète, le traite et en range le résultat dans la variable se trouvant à gauche du signe (=).

Maintenant, supposons que PAIN ait la valeur 28 et que vous vouliez augmenter cette valeur de 5. Si vous entrez :

```
PAIN = 28
PRINT PAIN
PRINT PAIN + 5
```

vous obtiendrez 33. Mais si vous tapez encore

```
PRINT PAIN
```

Applesoft vous redonnera la valeur initiale de la variable PAIN. Un moyen d'augmenter la valeur de la variable en mémoire est de taper

```
PAIN = PAIN + 5
PRINT PAIN
```

L'instruction PAIN = PAIN + 5 peut vous paraître illogique sauf si vous vous rappelez qu'Applesoft ne rend pas obligatoire l'utilisation du verbe LET (qui veut dire ''faire'' en français). Ce que vous voulez réellement dire est ''faire que la variable PAIN soit maintenant égale à l'ancienne valeur de PAIN augmentée de 5''.

Ainsi PAIN contiendra une nouvelle valeur — tant que vous ne lui aurez pas assigné une autre valeur. Entrez les instructions qui suivent en respectant l'ordre :

```
PAIN = 2
PRINT PAIN
PAIN = PAIN + 3
PRINT PAIN
PAIN = PAIN * 6
PRINT PAIN
PAIN = PAIN / 10
PRINT PAIN
```

A l'issue de cette séquence d'instructions, vous devriez voir affichée la valeur 3. Est-ce bon ? Est-ce bien ce que vous attendiez ?

Contrairement à la règle mathématique que vous avez pu connaître à l'école, à savoir : X est toujours égal à X, l'ordinateur introduit la notion de variation de valeur d'une variable. Dans l'instruction X = X + 1, le signe égal signifie ''prendre la valeur'' ou ''être assigné à la valeur''.

Apple *II*e peut mémoriser toute nouvelle valeur de la variable (PAIN) dès qu'elle est calculée ou assignée. Ce n'est pas la peine de taper PRINT PAIN à chaque fois. Pour vous en convaincre, tapez donc

```
PAIN = 2
PAIN = PAIN + 3
PAIN = PAIN * 6
PAIN = PAIN / 10
PRINT PAIN
```

Regardez la séquence qui suit, Quel doit être le résultat final ? Essayez.

```
POMMES = 55
BANANES = 11
QUOTIENT = POMMES / BANANES
PRINT QUOTIENT
```

Cet exemple illustre le fait suivant : dès que vous avez mémorisé plusieurs variables dans Apple *II,* vous pouvez faire des opérations sur ces variables. Les pommes ne peuvent pas, jusqu'à présent être divisées par des bananes, mais Applesoft reconnaît les variables et divise leur valeur.

Cette section a expliqué le mode de rangement des noms et les résultats des opérations entre variables. Vous devez toujours avoir présent à l'esprit qu'il est également possible de ranger des mots ou des suites de caractères dans des variables. On en parlera à nouveau au cours du Chapitre 5.

Un résumé des règles concernant les variables :

1. Une variable est un symbole représentant l'adresse en mémoire.

2. L'instruction LET est utilisée pour définir une variable.

3. LET est facultatif ; LET A = 23 et A = 23 a le même effet.

4. Applesoft extrait tout ce qui est à droite du signe égal, l'interprète, le traite et range le résultat dans la variable qui est à gauche du signe égal.

5. Un nom de variable doit commencer par une lettre. C'est une bonne habitude de donner des noms significatifs à vos variables pour faciliter leur identification.

6. Applesoft ne différencie une variable d'une autre qu'en analysant les deux premiers caractères de leur nom.

7. Les verbes d'Applesoft et quelques autres mots sont réservés et ne peuvent pas être utilisés comme noms de variables.

Pause

Pour gagner beaucoup de temps

Vous vous êtes sans doute aperçu que l'instruction PRINT est souvent utilisée dans Applesoft.

Cependant, vous allez apprendre quelque chose qui va vous faire gagner beaucoup de temps : un point d'interrogation peut être utilisé à la place de PRINT. En fait ? et PRINT sont identiques pour Applesoft. Faites-en l'essai en tapant ? ''EFFICACE !''

Hiérarchie entre opérateurs

Maintenant que vous utilisez Apple *IIe* pour faire des opérations, vous devez apprendre dans quel ordre, ou *hiérarchie,* Applesoft interprète vos instructions. Le résultat du calcul suivant

```
PRINT 4 + 8 / 2
```

sera-t-il 6 ou 8 ? Cela dépend de la première opération réalisée par Applesoft. Si Applesoft ajoute 4 à 8, puis divise le résultat (12) par 2, le résultat final sera 6. Si Applesoft divise 8 par 2, puis ajoute 4, le résultat final sera 8. Regardez la figure 1-14 pour voir dans quel ordre agit Applesoft.

Figure 1-14. Exemple de priorité. La division est faite en premier. L'expression devient alors 4 + 4 et l'addition est réalisée.

```
PRINT 4 + 8 / 2
PRINT 4 + 4
```

Voici d'autres exemples :

1. Quand un signe moins est utilisé pour définir un nombre, on l'appelle un signe moins. Dans l'exemple suivant

   ```
   PRINT -3 + 2
   ```

   Applesoft commence à convertir le nombre immédiatement précédé du signe moins en nombre négatif avant d'effectuer les opérations arithmétiques. Le résultat de $-3 + 2$ est -1. Si l'addition avait été faite en premier, $-3 + 2$ aurait donné -5. Mais ce n'est pas le cas. Un autre exemple

   ```
   NOMBRE = 6
   PRINT -NOMBRE + 10
   ```

   La réponse est 4. (Notez bien, cependant, que dans *l'expression arithmétique* $5-3$, le signe $-$ représente la soustraction et non un nombre négatif).

2. Après avoir traité tous les nombres négatifs, Applesoft exécute les exponentielles. L'instruction

   ```
   PRINT 4 + 3 ^ 2
   ```

   est évaluée en multipliant d'abord 3 par lui-même ($3 * 3 = 9$) et en ajoutant 4 au résultat obtenu ce qui donne 13 comme résultat final. Quand il y a plusieurs formes exponentielles dans l'expression, celles-ci sont exécutées de la gauche vers la droite, ainsi

   ```
   PRINT 2 ^ 3 ^ 2
   ```

   est évalué en multipliant 2 par lui-même trois fois ($2 * 2 * 2$), ce qui donne 8, puis en multipliant 8 par lui-même ($8 * 8$). La réponse est 64.

3. Une fois les formes exponentielles traitées, toutes les multiplications et les divisions sont faites, de la gauche vers la droite. Les opérateurs de même priorité sont toujours traités de la gauche vers la droite. La multiplication (*) et la division (/) ont la même priorité.

4. Addition et soustraction sont faites en suivant, avec la même priorité.

Ne retenez pas par cœur la hiérarchie entre les opérateurs. Vous pourrez toujours vous reporter à ces pages lorsque vous en aurez besoin.

Premier :	–	Unaire, ou signe moins utilisé pour définir un nombre négatif
Second :	^	Les formes exponentielles de gauche à droite
Troisième :	* /	Multiplication et division de gauche à droite
Quatrième :	+ –	Addition et soustraction de gauche à droite

Figure 1-15. Ordre de traitement des opérateurs par Applesoft pour évaluer les expressions arithmétique

Quelques expressions arithmétiques à évaluer sont présentées ci-après. Evaluez-les d'abord vous-mêmes puis avec Apple *II*e. Si votre réponse diffère de celle de l'ordinateur, essayez de comprendre pourquoi.

N'oubliez pas de faire précéder chaque expression de ? ou de PRINT. Faites les exemples un par un, en comparant votre réponse à celle du calculateur, avant de poursuivre. (Sauf si vous avez déjà compris la façon dont le calculateur procède).

```
PRINT 4 + 6 – 2 + 1
PRINT 5 – 4 / 2
PRINT 20 / 2 * 5
PRINT 6 * –2 + 6 / 3 + 8
PRINT 2 ^ 2 ^ 3 + 2 ^ 3
PRINT 8 * 2 / 2 + 3 * 2 ^ 2 * 1
```

Les réponses ne sont pas données dans ce livre. Apple *II*e vous donnera la bonne réponse. Si cela vous a plu, essayez d'autres expressions de votre choix. Si non, tenez bon !

Des parenthèses

Imaginons que vous vouliez diviser 12 par le résultat de 4 + 2. Si vous tapez

```
PRINT 12 / 4 + 2
```

vous obtiendrez 5 comme réponse. Mais ce n'est pas cela que vous souhaitez. Pour que vos désirs deviennent réalité, utilisez les parenthèses pour modifier la hiérarchie de calcul. Tapez

```
PRINT 12 / (4 + 2)
```

La règle suivie par Applesoft est simple ; il traite d'abord ce qui est entre parenthèse. Si des parenthèses sont incluses entre d'autre parenthèses, Applesoft évalue d'abord ce qui se trouve entre les plus proches. Voici un exemple :

```
PRINT 12 / (3 + (1 + 2) ^ 2)
```

Dans l'expression :

```
12 / (3 + 3 ^ 2)
```

Applesoft utilise alors les règles de priorité entre opérateurs pour évaluer : 3 + 3 ^ 2 soit 3 + 9, ou 12 et enfin 12 / 12 soit 1.

Dans le cas où il y a plus d'un couple de parenthèses, par exemple (9 + 4) * (1 + 2), Applesoft évalue les expressions entre chaque couple de parenthèses, commençant à gauche et continuant sur la droite. L'expression devient 13 * 3, soit 39.

Voici quelques autres expressions à évaluer. N'oubliez pas que vous pouvez taper un point d'interrogation (?) à la place de PRINT. Une fois n'est pas coutume : la plupart de ces règles sur la hiérarchie des opérateurs et sur les parenthèses sont respectées par l'ensemble des constructeurs du monde entier et pas seulement par Apple.

```
PRINT (44 / 2) + 2
PRINT 100 / (200 / (1 * (9 - 5)))
PRINT 32 / (1 + (7 / 3) + (5 / 4))
```

Pause

Résumé du chapitre

Voici tout ce que vous avez appris au cours de ce chapitre. Impressionnant, n'est-ce pas ? Chaque groupe reprend l'ensemble des termes de ce chapitre et dans l'ordre de leur apparition. La définition des instructions se trouve dans l'Annexe A. La définition des termes se trouve dans le Glossaire.

Instructions Applesoft

PRINT
INVERSE
NORMAL
HOME
GR
COLOR=
PLOT
HLIN
VLIN
TEXT
LET

Opérations Arithmétiques

addition
soustraction
multiplication
division
exponentiation

Termes Employés

caractères d'attention
curseur
verbe
instruction
caractère
nombre maximum de caractères
opérations arithmétiques
notation scientifique
mode graphique basse résolution
fenêtre de texte
modulateur de fréquence radio
syntaxe
mémoire principale
valeur
variable
mot réservé
hiérarchie
monadique
expression arithmétique
opérateur

Touches

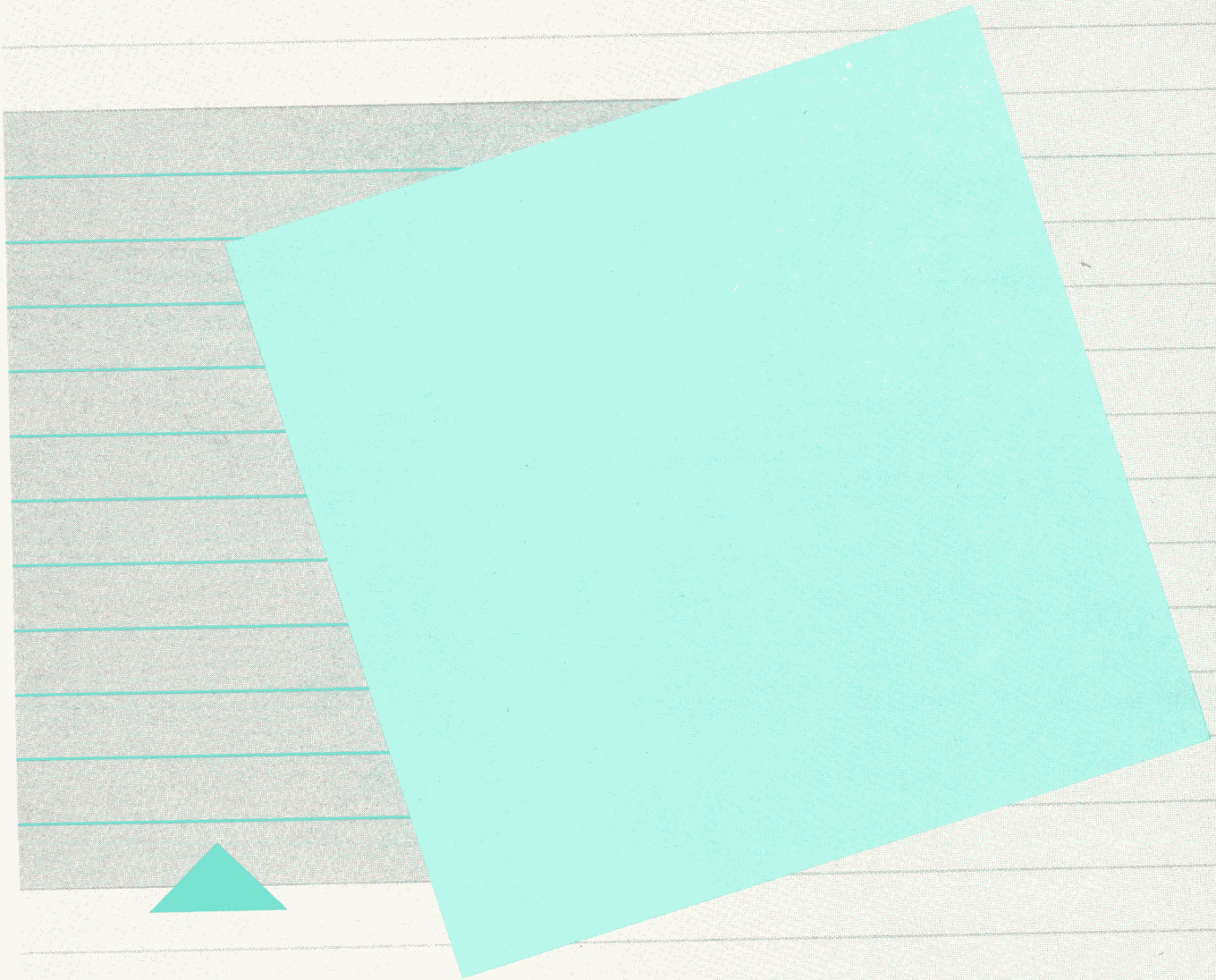
⇓
⇑
↵
←
→

Messages d'Erreur

?SYNTAX ERROR
?ILLEGAL QUANTITY ERROR

Premiers éléments de programmation

37 Mode programme
42 Boucles : l'instruction GOTO
45 Dialogue avec votre programme : INPUT
47 Vous voulez conserver vos programmes ?
47 Préparation
48 Sauvegarde
49 Les conditions : la recherche de la ''vérité''
51 Symboles utilisés dans les instructions conditionnelles
52 Règles d'utilisation des instructions conditionnelles
52 Boucles conditionnelles : l'instruction IF... THEN
54 Utilisation du programme Applesoft COLORLOOP
56 Compléments sur l'instruction IF... THEN
58 Remarques
59 Les boucles FOR/NEXT
62 Boucles emboîtées et enchevétrées
64 Contrôle de la présentation des résultats
70 Résumé du chapitre



Premiers éléments de programmation

Lorsqu'un ordinateur fonctionne d'après les instructions que vous lui avez données, on dit qu'il exécute les instructions. Jusqu'à présent vous tapiez

```
PRINT 3 + 4
```

et, lorsque vous pressiez ↵ , l'ordinateur faisait ce que vous lui demandiez immédiatement. Ce mode d'exécution est appelé *exécution immédiate.* La plupart des instructions de l'Applesoft peuvent être essayées en exécution immédiate.

Mais comment faire maintenant si vous voulez impressionner vos amis avec la magie de l'ordinateur ? Ou si vous voulez écrire quelques instructions maintenant, puis continuer plus tard ? Pour cela vous devez pouvoir mémoriser les instructions pour les exécuter plus tard. Ce mode de travail est appelé l'*exécution différée.* (ou mode programme).

Mode programme

Tout d'abord assurez-vous que la mémoire de l'ordinateur est vide, tapez :

```
NEW
```

La commande NEW efface la mémoire centrale de l'Apple *IIe* et doit être utilisée chaque fois que vous commencez un nouveau programme

Ensuite tapez la ligne suivante :

```
100 PRINT ''MY FAVORITE FOOD IS ARTICHOKES''
```

Si vous pressez sur ↵ , rien n'apparaît sur l'écran. Si vous pressez ↵ à nouveau, le curseur descend d'une ligne sur l'écran, mais votre instruction n'est toujours pas exécutée.

```
]NEW
]100 PRINT "MY FAVORITE FOOD IS ARTICHOKES"
]
]
]
```

La raison de ce fonctionnement est simple : lorsque vous avez tapé 100, vous avez donné un *numéro de ligne* à l'instruction et indiqué à l'ordinateur que vous vouliez une exécution différée. L'Apple *II*e garde la ligne 100 dans sa *mémoire* jusqu'à ce que vous lui demandiez le contraire (par exemple si vous coupez le courant de votre Apple *II*e, la ligne 100 sera perdue).

Bien sûr, vous ne voudrez sans doute pas retarder indéfiniment l'exécution de cette instruction. Vous vous demandez aussi certainement comment vous pouvez vous assurer que votre ligne 100 est bien là. Pour afficher sur votre écran le contenu de la mémoire de l'Apple *II*e, tapez :

La commande LIST affiche les lignes de programme qui sont dans la mémoire centrale de l'Apple *II*e.

LIST

et, sauf si vous avez fait une faute de frappe,

```
LIST
100 PRINT "MY FAVORITE FOOD IS ARTICHOKES"
```

apparaît sur l'écran.

LIST commande à l'ordinateur d'afficher toutes les lignes numérotées qu'il a en mémoire. Pour indiquer à l'ordinateur d'exécuter la ligne 100, tapez :

La commande RUN demande à l'ordinateur d'exécuter les instructions qui sont en mémoire.

RUN

et la chaîne

```
MY FAVORITE FOOD IS ARTICHOKES
```

apparaît sur l'écran.

Notez soigneusement les différences entre la ligne affichée par la commande LIST et le texte affiché lors de l'exécution. Vous remarquerez que le numéro de ligne 100 a disparu ainsi que PRINT. Comme vous vous en souvenez, l'instruction PRINT est utilisée pour afficher des informations sur l'écran. Lorsque vous faites exécuter la ligne 100, seul le texte compris entre les guillemets est affiché. Vous pouvez afficher et exécuter la ligne 100 autant de fois que vous le désirez.

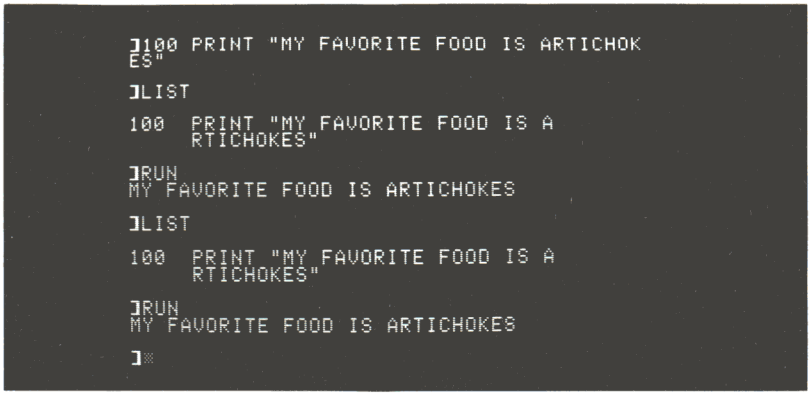


Illustration 2-1. Utilisation de LIST et RUN en exécution différée.

Vous avez besoin d'un certain temps pour bien comprendre la différence entre ce que vous voyez sur l'écran, ce qui est dans la mémoire centrale et ce qui est conservé sur un disque. Au fur et à mesure que vous travaillerez avec votre Apple *IIe* et ce manuel, cette différence deviendra de plus en plus claire. Le *Guide de l'Utilisateur Apple IIe* contient également des informations complémentaires sur ce sujet.

Pour voir ce qui se passe lorsque vous entrez une autre instruction avec le numéro de ligne 100, tapez :

```
100 PRINT ''THE SUM OF 3 + 4 IS''
110 PRINT 3 + 4
```

L'ancienne ligne 100 a été remplacée et n'est plus dans la mémoire. Pour le vérifier tapez LIST. Puis tapez :

```
RUN
```

Qu'obtenez vous ? Et maintenant si vous tapez :

```
NEW
```

puis

```
LIST
```

La mémoire de l'ordinateur a été effacée, les deux lignes 100 et 110 sont perdues. Si vous tapez :

```
RUN
```

rien n'est exécuté puisqu'il n'y a rien dans la mémoire. Vous devez donner à l'ordinateur de nouvelles instructions, tapez :

```
2 PRINT ''P''
1 PRINT ''A''
4 PRINT ''E''
3 PRINT ''L''
```

Affichez maintenant ces instructions. Remarquez que l'ordinateur mémorise les instructions dans l'ordre croissant des numéros de ligne : il a donc réordonné les instructions que vous lui avez données et affiche :

```
1 PRINT ''A''
2 PRINT ''P''
3 PRINT ''L''
4 PRINT ''E''
```

Lorsque vous faites exécuter ces instructions vous observez que l'ordinateur exécute aussi les instructions dans l'ordre croissant des numéros de ligne. Cela n'est pas suffisant : il manque un P. Pour l'ajouter, afin que l'ordinateur affiche :

```
A
P
P
L
E
```

vous devez retaper les lignes 3 et 4 en les numérotant 4 et 5 et vous devez ajouter une nouvelle ligne 3. Pour faire ces corrections, tapez :

```
3 PRINT ''P''
4 PRINT ''L''
5 PRINT ''E''
```

Pour voir ce qui s'est passé, utilisez la commande LIST puis la commande RUN.

Félicitations ! vous venez d'écrire un *programme*. Assembler des suites d'instructions précédées par des numéros de ligne est le B.A BA de la programmation. Un programme est une séquence (ce sont les numéros de ligne qui assurent cet ordre) mémorisée d'instructions (comme PRINT) qui font faire, à un ordinateur, une fonction donnée (dans ce cas : afficher APPLE sur l'écran).

Lorsqu'un P a été oublié dans APLE, il a été pénible de retaper les lignes 3 et 4. Heureusement, il existe une méthode plus facile qui consiste à programmer en laissant des intervalles entre les numéros de ligne et avant le premier numéro. Si, par exemple, les numéros avaient été 10, 20, 30, 40, vous auriez pu rajouter le P manquant avec l'instruction 15.

Un des avantages du mode programme est de vous permettre d'ajouter ou de modifier des instructions sans avoir à retaper tout le programme chaque fois. Par exemple, tapez :

```
NEW
```

pour éliminer les anciennes instructions, puis :

```
100 PRINT ''C''
110 PRINT ''T''
```

Lorsque vous faites exécuter ce programme, il n'affiche pas tout à fait CAT verticalement mais vous pouvez le modifier simplement en tapant :

```
105 PRINT ''A''
```

Affichez et exécutez à nouveau ce programme.

Voilà maintenant une plus longue liste d'instructions à essayer :

```
NEW
10 HOME
20 PRINT ''MY APPLE GIVES MESSAGES :''
30 PRINT
40 PRINT ''HI, THERE, PROGRAMMER !''
50 PRINT
60 PRINT ''MY APPLE DOES COMPUTATIONS :''
70 PRINT
'80 PRINT 60/12
90 PRINT 4 ^ 5
```

Le mode programme présente l'avantage de permettre de stocker plusieurs instructions à la fois, à condition qu'elles aient toutes un numéro de ligne différent.

Vous voulez certainement voir tout de suite le résultat de ces instructions, tapez :

```
RUN
```

et regardez le résultat qui apparaît :

```
MY APPLE GIVES MESSAGES :
HI, THERE, PROGRAMMER !
MY APPLE DOES COMPUTATION :
5
1024
```

Est-ce que votre écran ressemble à cela ? Sinon, affichez votre programme pour voir ses erreurs (Remarquez que vous pouvez afficher votre programme avant ou après l'exécution). Voici une liste de points à vérifier lorsque vous affichez un programme :

- Est-ce que toutes les lignes ont des numéros différents ?
- Est-ce que vous avez orthographié PRINT correctement ?
- Est-ce que vous avez pensé à tous les guillemets dans les instructions PRINT (les lignes 30, 50 et 70 introduisent simplement des lignes blanches entre les textes affichés comme le ferait le retour chariot d'une machine à écrire).

Si vous devez modifier une ligne, retapez-la simplement. Ensuite, affichez à nouveau le programme pour vérifier que tout est correct. (Dans le prochain Chapitre, vous apprendrez des méthodes plus rapides pour corriger des lignes de programme).

Dans cette section, vous avez appris plusieurs commandes qui vous permettent de travailler avec des programmes complets. Elles sont résumées dans l'illustration 2-2.

Illustration 2-2. Les fonctions des commandes NEW, LIST et RUN

NEW	Efface le programme de la mémoire de l'ordinateur. Après avoir utilisé NEW, vous devez introduire de nouvelles instructions soit à partir du clavier soit à partir d'un disque.
LIST	Affiche sur l'écran le programme qui est dans la mémoire.
RUN	Fait exécuter par l'ordinateur le programme qui est dans la mémoire en commençant par l'instruction qui a le plus petit numéro de ligne (il est aussi possible de commencer l'exécution avec n'importe quel autre numéro de ligne).

Pause

Boucles : l'instruction GOTO

Supposons que vous vouliez afficher tous les nombres de 1 à 200, en affichant un nombre par ligne. Une façon évidente de le faire est de taper :

```
100 PRINT 1
110 PRINT 2
120 PRINT 3
```

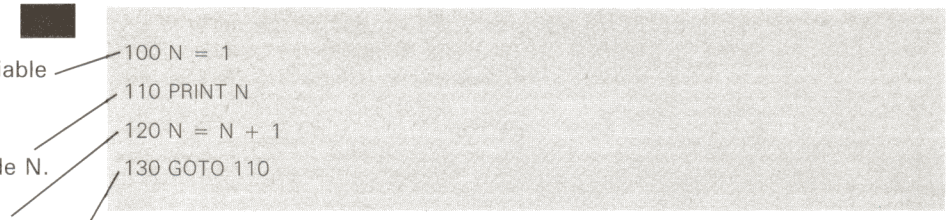
et ainsi de suite. Mais cette méthode nécessiterait 200 instructions et donc une longue frappe attentive ! Heureusement, il y a une solution plus rapide. Vous pouvez afficher les entiers positifs avec 5 instructions seulement.

```
NEW
100 N = 1
110 PRINT N
120 N = N + 1
130 GOTO 110
```

Avant de faire exécuter ce programme, étudiez la figure 2-3 pour comprendre comment le programme fonctionne.

L'exécution de l'instruction GOTO, provoque le branchement à la ligne indiquée. Cette instruction est utilisée pour créer des boucles dans un programme.

Illustration 2-3. L'instruction GOTO



La ligne 100 affecte 1 à la variable N. Voir *''Variables et autres facilités''*, Chapitre 1.

La ligne 110 affiche la valeur de N.

La ligne 120 augmente de 1 la valeur de N.

La ligne 130 provoque le retour du programme à la ligne 110. La ligne 110 affiche N, la ligne 120 incrémente la valeur de N et la ligne 130 renvoie à la ligne 110 et ainsi de suite. Chaque fois une nouvelle valeur de N est affichée.

Maintenant faites exécuter ce programme. Vous allez apprendre bientôt comment arrêter l'exécution. Pour le moment admirez la puissance de ce programme. Si vous avez tapé RUN ainsi qu'on vous l'a indiqué plus haut, l'Apple *IIe* a déjà exécuté l'instruction PRINT plusieurs centaines de fois.

Pour arrêter le programme, appuyez sur la touche Ctrl, puis sur la touche C en maintenant la touche Ctrl appuyée.

La commande Ctrl - C provoque l'arrêt de l'exécution et vous indique où l'exécution a été interrompue en affichant un numéro de ligne, par exemple :

```
BREAK IN 110
```

Essayez. Vous pouvez remarquer qu'il s'agit d'une exception à la règle disant qu'il faut presser ↵ après chaque instruction. Il n'est pas en général nécessaire d'appuyer sur ↵ , lorsqu'un programme est interrompu par Ctrl - C.

Utilisez Ctrl - C pour arrêter l'exécution du programme.

Lorsque vous interrompez l'exécution d'un programme par Ctrl - C, vous pouvez généralement reprendre son exécution en tapant :

```
CONT
```

La commande CONT permet de reprendre ou de continuer l'exécution d'un programme interrompu par un Ctrl - C, un STOP ou un END.

qui signifie ''continue''. Faites-le maintenant.

En regardant l'écran vous remarquerez que les nombres défilent puis disparaissent. Chaque nouveau nombre est affiché en bas de l'écran et tous les autres sont déplacés d'une ligne vers le haut de l'écran. Ce mécanisme s'appelle le *scrolling (défilement)*. Vous avez pu l'observer depuis que vous travaillez sur votre Apple *II*e mais beaucoup plus lentement.

Arrêtez à nouveau le programme par Ctrl - C. Entraînez vous à utiliser CONT et Ctrl - C, ils vous seront très utiles.

Si vous voulez redémarrer le programme depuis le début, tapez RUN (à la place de CONT). Si vous êtes prêts pour un autre exemple d'utilisation de l'instruction GOTO, tapez :

```
NEW
100 PRINT ''RAIN''
110 PRINT ''IS''
120 PRINT ''FALLING''
130 PRINT ''DOWN''
140 PRINT ''DOWN''
150 PRINT ''DOWN''
160 GOTO 100
RUN
```

Ctrl - C permet d'interrompre l'exécution du programme et CONT de la reprendre.

Précédemment, on vous a appris que la commande RUN lance l'exécution du programme par l'instruction du plus petit numéro de ligne. Mais si vous voulez commencer l'exécution du programme par une autre ligne, par exemple la ligne 130, tapez :

```
RUN 130
```

et vous obtiendrez :

```
DOWN
DOWN
DOWN
```

sur votre écran avant que le programme ne revienne à la ligne 100 et affiche le début du texte.

De même vous pouvez préciser des numéros de ligne dans la commande LIST. Si vous tapez :

```
LIST 130
```

l'Apple *IIe* n'affichera que la ligne 130 (si elle existe). Si vous tapez :

```
LIST 130, 150
```

ou

```
LIST 130-150
```

l'ordinateur affichera toutes les lignes de programme entre 130 et 150 comprises. Essayez. Par contre, vous ne pouvez pas indiquer une fourchette de lignes pour la commande RUN.

L'Annexe A contient une brève description de chaque commande de l'Apple *IIe*. Lorsque vous avez besoin d'une information rapide (par exemple, si vous voulez savoir la différence entre les commandes RUN et LIST) cette annexe sera très utile.

Pause

Dialogue avec votre programme : INPUT

L'instruction INPUT permet à l'utilisateur de dialoguer avec le programme en cours d'exécution.

Les programmes que vous avez écrits jusqu'ici peuvent déjà impressionner vos amis les moins expérimentés. Mais comment faire un programme qui pose des questions ? Pour cela vous pouvez utiliser l'instruction INPUT.

Imaginons par exemple que vous vouliez écrire un programme qui demande l'âge d'une personne et utilise le nombre qu'elle a entré pour afficher un message. Vous connaissez déjà l'instruction PRINT pour afficher un message. La première chose à faire est donc d'inclure dans le programme le message que vous voulez utiliser. Vous savez également comment associer un nom de variable à une valeur. L'âge est une variable : utilisez ce que vous avez appris. Ce que vous ne savez pas encore, c'est comment utiliser l'instruction INPUT.

Tapez ce programme puis affichez-le pour vérifier que vous n'avez pas fait de faute de frappe. Avant d'exécuter le programme, étudiez soigneusement l'illustration 2-4. L'ordre et la syntaxe de chaque instruction sont très importants.

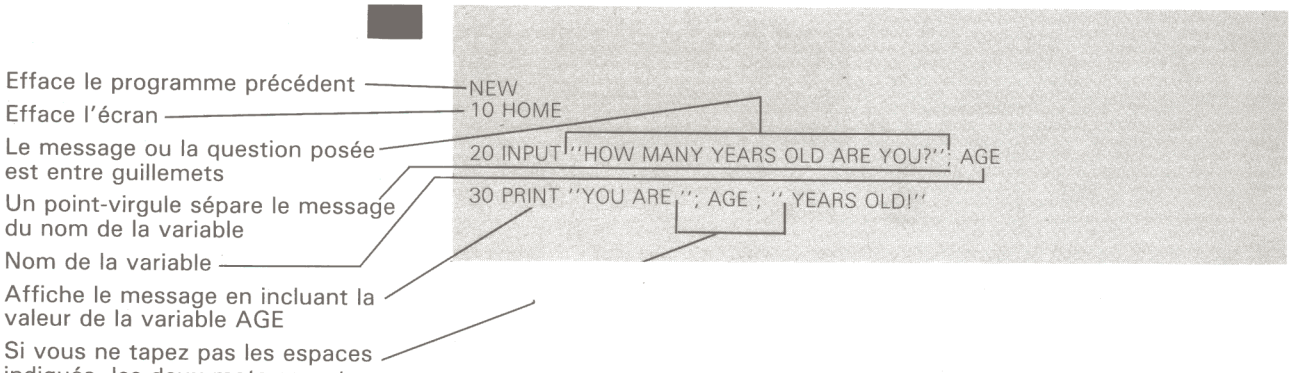
```
NEW
10 HOME
20 INPUT ''HOW MANY YEARS OLD ARE YOU ?'' ; AGE
30 PRINT ''YOU ARE'' ; AGE ; ''YEARS OLD !''
```

Faites exécuter ce programme. Lorsque vous voyez l'écran vide et la question

```
HOW MANY YEARS OLD ARE YOU ?
```

tapez une réponse (votre âge). Que se passe-t-il ? Essayez à nouveau en prétendant que vous avez un autre âge.

Illustration 2-4. Utilisation de l'instruction INPUT



Lorsque vous utilisez l'instruction INPUT dans sa forme la plus simple, elle affiche un point d'interrogation et attend que vous tapiez quelque chose. Ce que vous tapez est conservé dans la variable.

Lorsque vous utilisez l'instruction INPUT, vous devez définir :

- quelle question ou quel message, vous voulez afficher pour l'utilisateur du programme. INPUT, comme PRINT, affiche exactement ce que vous écrivez, il est donc important de poser une question judicieuse. Le message doit être entre guillemets et séparé du nom de la variable par un point-virgule.
- que voulez-vous que tape l'utilisateur ? S'il doit entrer un nombre, vous devez définir un nom de variable.

L'utilisateur peut également taper un mot ou un groupe de caractères. Un programme illustrera cela plus loin dans ce chapitre et vous en apprendrez plus sur l'instruction INPUT dans les Chapitres 4 et 5.

Voici maintenant un exemple de calcul que vous pouvez ajouter à votre programme INPUT. Ne modifiez pas les lignes 10 et 20 ; ajoutez la ligne :

```
25 AGE = AGE * 365
```

et modifiez comme suit la ligne 30 :

```
30 PRINT ''YOU ARE ABOUT''; AGE ; ''DAYS OLD !''
```

Utilisez la commande LIST pour vérifier les nouvelles lignes puis exécutez le programme. Vous remarquerez que ce programme ne donne pas exactement l'âge d'une personne en jours — sauf si c'est son anniversaire — il donne une estimation ; c'est pourquoi vous avez ajouté ABOUT dans la ligne 30. Chaque fois que vous voulez voir le résultat du programme, tapez RUN pour recommencer l'exécution.

Vous voulez conserver vos programmes ?

Jusqu'à présent vous avez utilisé le mode programme en tapant des programmes avec des numéros de ligne, mais chaque fois vous tapiez NEW et votre programme précédent était effacé de la mémoire de l'ordinateur. Cette section explique comment conserver vos programmes en utilisant le système d'exploitation disque de votre ordinateur.

Préparation

Pour sauvegarder un programme, vous avez besoin de deux choses : tout d'abord d'un programme à sauvegarder et ensuite d'un disque *initialisé* ou d'une cassette magnétique sur laquelle écrire le programme. Bien sûr vous pouvez aussi conserver des programmes en les recopiant à la main sur du papier mais ce n'est pas aussi amusant que d'utiliser l'Apple *II*e.

L'**initialisation** ou le **formatage** est un processus utilisé pour préparer un disque pour recevoir des informations. Ce processus est décrit dans le *Guide de l'Utilisateur Apple IIe.*

- Vous devriez avoir initialisé un disque avant de commencer ce manuel. Si vous l'avez fait, retrouvez-le et passez directement à la section suivante ''Sauvegarde''.
- Si vous n'êtes pas sûr qu'un disque est initialisé vérifiez-le en tapant :

  ```
  CATALOG
  ```

  et appuyez sur ↵. Votre écran devrait afficher quelque chose comme :

CATALOG est une commande de DOS qui permet d'afficher la liste des fichiers contenue dans le disque placé dans un lecteur donné.

```
DISK VOLUME 254
 A 002 HELLO
```

Si ce message apparaît sur l'écran, le disque est initialisé et vous pouvez passer directement à la section "Sauvegarde". Sinon référez vous au *Guide de l'Utisateur Apple IIe* qui explique comment initialiser un disque.

- Si vous avez deux lecteurs de disque connectés à votre Apple *IIe* vous devez savoir quel lecteur utiliser. Le DOS utilise automatiquement le disque 1 pour démarrer le système et continue à l'utiliser jusqu'à ce qu'on lui indique le contraire. Lorsque vous tapez, D2 dans une commande DOS — par exemple CATALOG, D2 — le DOS utilise le lecteur 2 (c'est ce que signifie D2) jusqu'à ce que vous lui demandiez le contraire en indiquant, D1 dans une autre commande DOS.

  Normalement avec deux lecteurs de disques, vous avez un disque système (par exemple DOS 3.3 SYSTEM MASTER) sur le lecteur 1 et un autre disque (comme le disque initialisé que vous voulez utiliser) sur le lecteur 2.

  Pour plus d'information concernant l'utilisation d'un Apple *IIe* avec deux lecteurs de disques, veuillez vous référer au *Guide de l'Utilisateur Apple IIe.*

- Si vous utilisez un magnétophone à cassettes à la place d'un lecteur de disque, consultez le *Guide de l'Utilisateur Apple IIe* pour savoir comment conserver des informations sur une cassette et au *Manuel Applesoft* de *Référence* pour ce qui est de l'utilisation des commandes de cassettes.

Sauvegarde

Si vous avez suivi le manuel, le programme de calcul de l'âge est toujours en mémoire. Pour créer une copie permanente de ce programme sur le disque, tapez la commande SAVE suivie du nom que vous voulez donner au programme. Pour voir le déroulement de ceci, assurez-vous que vous avez une disquette initialisée et tapez :

La commande SAVE suivie d'un nom de fichier est une commande de DOS qui permet de copier sur le disque le programme actuellement en mémoire. La commande SAVE utilisée sans nom de fichier recopie le programme actuellement en mémoire sur cassette.

```
SAVE AGE
```

Votre lecteur de disque va faire un bref bruit et vous verrez la lampe rouge s'allumer. Lorsque cette lampe s'éteint, tapez :

```
CATALOG
```

et vous obtiendrez la liste des programmes sur le disque ; il devrait y en avoir 2, ou davantage si d'autres programmes ont été déjà sauvegardés.

Remarque : si vous avez éteint votre système, le programme de calcul de l'âge n'est plus dans la mémoire. Pour le sauvegarder sur le disque, retapez-le puis passez la commande SAVE AGE.

Si vous utilisez un système avec deux lecteurs de disques et voulez sauvegarder votre programme sur le deuxième lecteur tapez, D2 après le nom du programme.

Lorsque vous avez sauvegardé un programme sur un disque vous pouvez le recharger dans la mémoire de l'ordinateur en tapant :

LOAD AGE

La commande LOAD, utilisée avec un nom de programme, recherche ce programme sur le disque indiqué ou sur le disque utilisé par défaut. Lorsque LOAD est utilisée sans nom de programme, le programme est lu sur le lecteur de cassette.

Lorsque vous tapez la commande LOAD, vous entendez le lecteur de disque rechercher le programme. Dès que la lampe rouge s'éteint, affichez ou faites exécuter le programme que vous avez chargé.

Si rien ne se passe : retapez le programme et essayez de le sauvegarder à nouveau. Si, encore une fois, rien ne se passe, vérifiez :

- que vous avez bien initialisé le disque.
- que le programme HELLO est bien affiché lorsque vous exécutez la commande CATALOG.
- que vous n'êtes pas en train d'essayer de sauvegarder le programme sur le disque maître DOS 3.3 (ce disque est *protégé en écriture,* c'est-à-dire que vous ne pouvez pas y sauvegarder de programmes).
- que si vous voulez utiliser le deuxième lecteur de disque, vous n'avez pas oublié de taper, D2 dans votre commande.

Pause

Les conditions : la recherche de la ''vérité''

Beaucoup de choses sont conditionnelles dans la vie courante : un événement doit arriver avant un autre. Par exemple, vous devez avoir un certain âge avant de pouvoir voter ou avant de prendre votre retraite. Le gouvernement peut savoir si telle condition est remplie et prendre une décision si elle est remplie et une autre dans le cas contraire ; il en est de même pour l'Applesoft.

Lorsque vous utilisez une instruction conditionnelle de l'Applesoft, il y a toujours deux réponses possibles : vrai ou faux. La condition faux est représentée par (0) et la condition vrai par (1). Si, par exemple, vous tapez :

```
PRINT 17 > = 18
```

l'Apple *II*e affichera 0 sur la ligne suivante.

```
PRINT 17 > = 18
0
```

L'Applesoft a interprété l'instruction de la manière suivante : 17 est-il plus grand que 18, non ; donc afficher 0 pour indiquer que la condition est fausse. Si au contraire vous tapez :

```
PRINT 75 > = 18
```

l'ordinateur affichera 1 puisque 75 est bien supérieur à 18. Essayez d'autres exemples de l'utilisation de la condition ''supérieur ou égal à'' :

```
PRINT 18 > = 18
PRINT  3 > = 18
PRINT 27 > = 18
```

Les instructions conditionnelles, combinées avec les autres instructions présentées dans ce chapitre, permettent à l'Applesoft de faire des choix. Si une condition est vérifiée, un certain ensemble d'instructions sera exécuté sinon un autre ensemble d'instructions le sera.

La méthode utilisée par l'Applesoft pour évaluer les conditions est basée sur le système *binaire* composé de 0 et de 1. Binaire est un mot très utilisé dans le vocabulaire informatique parce que les ordinateurs stockent et manipulent l'information sous forme binaire. Autrement dit, les informations que vous entrez dans un ordinateur sont traduites en suites de 0 et de 1 ; on appelle encore cela le *langage machine.*

Symboles utilisés dans les instructions conditionnelles

L'Applesoft utilise six symboles pour définir les relations entre des valeurs. La Figure 2-5 donne la liste de ces symboles ainsi que quelques exemples d'utilisation.

Figure 2-5. Symboles utilisés par l'Applesoft dans les instructions conditionnelles.

Symbole	Signification	Exemples Vrai(1)	Exemples Faux(0)
=	Egal	3=3	3=1
>	Plus grand que (1)	78>55	78>124
<	Plus petit que (1)	10<20	10<9
>=	plus grand ou égal (1)	4>=4	4>=25
<=	plus petit ou égal (1)	32<=33	32<=30
<>	différent de	32<>33	32<>32

Remarquez que les symboles utilisés pour ''plus grand ou égal'' et ''plus petit ou égal'' sont formés de < ou > suivis du signe =. Le symbole pour ''différent de'' est formé de < suivi de >.

Parmi les instructions suivantes, essayez de trouver celles qui sont vraies et celles qui sont fausses, puis essayez-les. (Souvenez-vous que vous pouvez remplacer PRINT par un point d'interrogation).

```
PRINT 5<>5
PRINT 8<=8
PRINT -8 <-7
PRINT -2>=-5
PRINT 9<>-9
?RINT (45*6)<>(-45+6)
```

Règles d'utilisation des instructions conditionnelles

- Les instructions conditionnelles peuvent contenir des variables (par exemple AGE), des nombres (comme 5) ou des expressions (comme 45*6).
- Lorsque l'ordinateur évalue la valeur d'une condition, le résultat est toujours soit 0 soit 1.
- Dans les instructions conditionnelles, tous les nombres différents de 0 sont considérés comme ''vrais''.
- L'Applesoft évalue les conditions utilisées dans une instruction après avoir évalué les expressions de cette instruction.

L'Applesoft respecte un ordre précis pour l'évaluation d'une instruction contenant des opérations arithmétiques et des conditions.

- Premièrement, les opérations entre parenthèses sont évaluées.
- Deuxièmement, les signes moins utilisés pour désigner des valeurs négatives sont traités.
- Troisièmement, les élévations à la puissance sont évaluées.
- Quatrièmement, les multiplications et les divisions sont calculées de gauche à droite dans une expression.
- Cinquièmement, les additions et les soustractions sont calculées ; elles sont traitées de gauche à droite lorsqu'elles sont sur une même ligne.
- Sixièmement, les symboles utilisés dans les conditions sont évalués ; les six symboles ont la même priorité et sont évalués de gauche à droite.

Pause

Boucles conditionnelles : l'instruction *IF... THEN*

La première fois que vous avez essayé les boucles, en utilisant l'instruction GOTO, vous avez dû interrompre l'exécution de votre programme en tapant Ctrl - C. Une autre façon de limiter l'exécution d'un programme est d'utiliser l'instruction IF... THEN.

Imaginons par exemple que vous vouliez compter jusqu'à 20 puis vous arrêter. Vous avez besoin d'une instruction conditionnelle qui limitera la valeur de N à 20 au plus. Cela s'écrit N<=20.

L'instruction IF... THEN permet de créer des boucles conditionnelles.

Le programme doit compter tant que N est inférieur ou égal à 20 et sinon s'arrêter (c'est-à-dire si N > 20). Voici le programme, il utilise l'instruction IF... THEN.

```
NEW
200  N=1
210  PRINT N
220  N=N+1
230  IF N<=20 THEN GOTO 210
```

Tant que N sera inférieur ou égal à 20 le programme retournera à la ligne 210. Lorsque cette condition ne sera plus remplie, le programme ne retournera pas en 210 et l'exécution s'arrêtera.

Les boucles, dans un programme d'ordinateur, ont un début (ici la ligne 210) et une fin (ici la ligne 230) ; elles ont également une limite (ici 20).

D'une manière générale, l'instruction IF fonctionne de la manière suivante : l'expression qui suit le mot IF est évaluée, si le résultat est 0 (c'est-à-dire si la condition est fausse) le reste de la ligne de programme est ignoré et l'ordinateur continue avec la ligne de programme suivante ; si le résultat de l'expression n'est pas nul (c'est-à-dire si l'expression est vraie), l'instruction suivant le mot THEN est exécutée.

Pour rechercher le programme AGE dans la mémoire, tapez :

```
LOAD AGE
```

puis tapez LIST. Vous devriez obtenir sur votre écran :

```
10  HOME
20  INPUT "HOW MANY YEARS OLD ARE YOU ?"; AGE
25  AGE = AGE * 365
30  PRINT "YOU ARE ABOUT"; AGE ; "DAYS OLD !"
```

Pour expérimenter un peu les instructions IF... THEN, modifiez les lignes 25 et 30 de la manière suivante :

```
25  IF AGE  > 18 THEN GOTO 40
30  PRINT ''YOU WILL BE ABLE TO VOTE IN'';
    18 - AGE ; ''YEARS.''
```

et ajoutez la ligne 40 :

```
40  PRINT ''YOU ARE ELIGIBLE TO VOTE. HAVE YOU REGISTERED ?''
```

La Figure 2-6 montre les deux branches que peut prendre l'exécution du programme. Après avoir étudié ce schéma, faites exécuter votre programme.

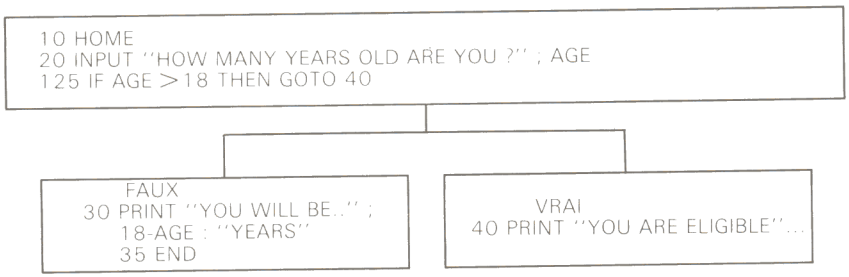


Figure 2-6. Boucle conditionnelle.

Si vous ne voulez pas effacer votre ancien programme AGE, vous devez sauvegarder cette nouvelle version sous un nouveau nom. Il est intéressant de toujours utiliser des noms de programme significatifs : vous pouvez par exemple appeler ce nouveau programme VOTE-AGE.

Utilisation du Programme Applesoft COLORLOOP

Le programme COLORLOOP est un programme de démonstration graphique qui utilise les boucles conditionnelles. Il figure sur le disque APPLESOFT SAMPLER, vous n'aurez donc pas à le retaper.

Pour charger ce programme du disque dans la mémoire de l'ordinateur, vous utiliserez les commandes que vous avez déjà utilisées pour charger vos propres programmes. Vérifiez simplement que vous avez bien le bon disque dans le lecteur.

Enlevez le disque que vous utilisiez pour sauvegarder vos programmes du lecteur et introduisez le disque marqué APPLESOFT SAMPLER. Puis tapez :

RUN COLORLOOP

vous devez entendre le lecteur chercher le programme sur le disque. Quand le programme est chargé dans la mémoire, il démarre automatiquement. Si COLORLOOP ne démarre pas :

- tapez CATALOG pour vérifier que le disque APPLESOFT SAMPLER est dans le lecteur et que le programme COLORLOOP y figure.
- tapez à nouveau la commande RUN en vérifiant que vous ne faites pas de faute de frappe et que vous avez bien spécifié le bon lecteur de disque (si le disque APPLESOFT SAMPLER est sur le lecteur 2 tapez, D2 après le nom du programme).

Lorsque l'exécution est terminée, examinez le programme en tapant : TEXT, HOME puis LIST. Vous devriez voir ceci sur votre écran.

```
400  GR
410  ROW = 1
420  COLOR = ROW
430  HLIN 0,39 AT ROW
440  ROW = ROW + 1
450  IF ROW < 16 THEN GOTO 420
```

Si vous n'avez pas de lecteur de disque, tous les programmes du disque APPLESOFT SAMPLER figurent dans ce manuel ; lorsqu'on vous demande de charger l'un de ces programmes à partir du disque trouvez-en le texte et tapez-le.

Dans ce programme ROW est le nom d'une variable. L'instruction de la ligne 420 affecte la valeur de la variable ROW à COLOR. Chaque fois que la valeur de la variable ROW est modifiée, la valeur de COLOR est modifiée également.

Observer l'exécution du programme : il illustre par un graphique ce que vous avez utilisé dans le programme AGE et dans le programme de comptage. Tant que la valeur de ROW est inférieure à 16, le programme retourne à la ligne 420 et exécute une nouvelle fois la boucle d'instructions ; quand ROW dépasse 16 le programme s'arrête. Vous commencez à comprendre comment l'on peut combiner les instructions de l'APPLESOFT de manière intéressante.

Compléments sur l'instruction IF... THEN

L'instruction IF est une instruction très puissante, vous l'utiliserez dans la plupart des programmes que vous écrirez. Le programme que nous allons étudier maintenant démontre ce que l'on peut faire en introduisant l'instruction IF dans un programme simple. Ce programme n'est pas sur le disque APPLESOFT SAMPLER car vous aurez à le modifier plusieurs fois. Les différentes modifications que vous ferez à ce programme vous montreront comment les diverses instructions que vous avez apprises peuvent être combinées dans un programme. Souvenez-vous que vous devez taper :

```
TEXT
```

si vous voulez voir toutes les lignes du programme pendant que vous tapez.

```
NEW
200  GR
210  COLOR=9
220  PLOT 0,0
230  PLOT 0,39
240  PLOT 39,39
250  PLOT 39,0
```

Affichez votre programme pour le vérifier ; puis faites-le exécuter. Modifiez maintenant la ligne 210 pour changer la couleur affichée :

```
210  COLOR = 15
```

Faites exécuter le programme encore une fois. Essayez d'afficher votre programme à nouveau : vous ne voyez que les lignes 240 et 250. Le reste du programme a disparu de la petite fenêtre de texte située en bas de votre écran. Ce fonctionnement durera tant que vous ne taperez pas :

TEXT
pour sortir du mode graphique, puis

HOME
pour effacer l'écran avant de passer la commande LIST.

Maintenant remplacez les lignes 220 à 250 avec les instructions suivantes et réaffichez votre programme (tant que vous êtes toujours en mode texte).

```
220 COLUMN=0
230 VLIN 0,39 AT COLUMN
240 COLUMN=COLUMN+1
250 IF COLUMN<40 THEN GOTO 230
```

Ce programme remplit tout l'écran avec la même couleur. Faites-le exécuter.

Puisque vous n'avez pas tapé NEW, l'ancienne ligne 200 est toujours là et sera exécutée dans votre programme ; elle restera tant que vous ne taperez pas NEW, que vous ne couperez pas votre ordinateur ou que vous ne taperez pas une nouvelle ligne 200.

Figure 2-7. Le programme VLIN-LOOP

Initialise le mode graphique.

Positionne la couleur.

Nomme la variable COLUMN car le programme va travailler sur les colonnes du mode graphique basse résolution. Le programme commencera à la colonne 0 (première colonne de l'écran).

Trace une ligne verticale dans la colonne 0.

Ajoute 1 à la valeur de la variable COLUMN.

Si la valeur de COLUMN est inférieure à 40 le programme retourne à la ligne 230 et trace une ligne dans la nouvelle colonne. Lorsque la valeur de COLUMN atteint 40 le programme s'arrête (en effet la dernière colonne de l'écran est 39).

```
200 GR
210 COLOR=15
220 COLUMN=0
230 VLIN 0,39 AT COLUMN
240 COLUMN=COLUMN + 1
250 IF COLUMN < 40 THEN GOTO 230
```

Pour éviter de taper RUN chaque fois que vous voulez colorier l'écran, ajoutez la ligne

```
260 GOTO 210
```

Que se passe-t-il quand vous faites exécuter le programme ? Essayez de taper quelque chose... ce que vous tapez n'apparaît pas parce que le programme s'exécute en permanence : chaque fois qu'il arrive à l'instruction 260, il retourne en 210. Pour arrêter ce programme, tapez Ctrl - C. Si vous voulez le relancer, tapez CONT.

Arrêtez votre programme à nouveau avec Ctrl - C. Tapez TEXT, HOME et LIST. La ligne 260 crée une boucle infinie avec l'instruction GOTO car il n'y a rien dans le programme pour arrêter cette boucle. Si vous voulez sauvegarder ce programme sans la boucle infinie, enlevez la ligne 260.

Remarque sur les numéros de ligne : les numéros de ligne peuvent commencer par 10, 100, 1000 ou n'importe quel nombre, mais ce qui est important est de toujours laisser un intervalle entre deux numéros successifs afin de pouvoir ajouter facilement des instructions, si nécessaire. Toutefois dans les exercices de ce manuel, gardez les numéros indiqués sinon il vous sera difficile de suivre les différentes modifications du programme.

Remarques

Comme vous avez pu le voir dans l'exemple précédent, ce qui se passe dans une ligne donnée peut être assez compliqué. L'Applesoft comporte une instruction prévue pour permettre de décrire ce qui va se passer : REM, qui est une abréviation de remarque.

L'ordinateur ignore les instructions REM ; elles ne sont utilisées que par les humains. Par exemple, voyez comme il est facile de suivre le déroulement du programme VLIN LOOP lorsqu'il est abondamment commenté comme dans la Figure 2-8.

Figure 2-8. Utilisation de l'instruction REM

Programme commenté	Programme non commenté
195 REM SET GRAPHICS MODE	
200 GR	200 GR
205 REM SET COLOR	
210 COLOR = 15	210 COLOR = 15
215 REM START AT COLUMN 0	
220 COLUMN = 0	220 COLUMN = 0
225 REM DRAW VERTICAL LINE AT COLUMN	
	230 VLIN 0,39 AT COLUMN
235 REM PROCEED TO NEXT COLUMN BY ADDING 1	
240 COLUMN = COLUMN + 1	240 COLUMN = COLUMN + 1
245 REM LOOP MAKES IT EXECUTE OVER AND OVER UNTIL COLUMN 39 IS REACHED	
250 IF COLUMN < 40 THEN GOTO 230	250 IF COLUMN < 40 THEN GOTO 230

Si vous regardez attentivement, vous pouvez vérifier que le programme contient toujours les mêmes instructions ; les instructions REM expliquent simplement ce que la ligne (ou les lignes) suivante va faire.

Les instructions REM sont facultatives. L'Applesoft ne les exécute pas, elles ne sont dans le programme qu'à titre d'information. Vous pouvez les utiliser comme vous l'entendez. Plus vos programmes seront longs, plus les instructions REM seront utiles. Elles aident à comprendre ce que fait le programme.

Les boucles FOR/NEXT

Les instructions FOR/NEXT définissent une boucle d'instructions qui seront exécutées un nombre de fois donné, spécifié par le champ TO de l'instruction FOR.

Les instructions FOR/NEXT créent une boucle d'instructions qui seront exécutées un nombre de fois donné. Ce nombre est défini par une variable. Les instructions FOR/NEXT fournissent une méthode plus efficace pour écrire des boucles dans lesquelles vous utilisez une variable qui est régulièrement augmentée. Tapez TEXT pour revenir en mode texte, puis HOME pour effacer l'écran et enfin NEW pour effacer la mémoire. Ensuite essayez ce programme :

```
NEW
100 FOR NUMBER = 0 TO 12
110 PRINT NUMBER
120 NEXT NUMBER
130 PRINT ''PROGRAM IS FINISHED WHEN LAST NUMBER IN
    THE FOR STATEMENT IS DISPLAYED''
```

Faites exécuter ce programme. Vous pouvez voir que son résultat est le même que celui du programme de comptage que vous aviez écrit avec IF... THEN. La Figure 2-9 explique comment ce programme fonctionne.

```
100 FOR NUMBER = 0 TO 12

110 PRINT NUMBER

120 NEXT NUMBER

130 PRINT "PROGRAM IS FINISHED WHEN LAST NUMBER IN THE FOR
    STATEMENT IS DISPLAYED"
```

Définition de la plage de variation de la variable NUMBER : 0 à 12.

La valeur de NUMBER est affichée.

Cette ligne définit la limite de la boucle. La variable est augmentée de 1 puis comparée à la limite spécifiée dans la ligne 100 : 12. Si cette limite est dépassée, le programme exécute l'instruction 130 et s'arrête.

Figure 2-9. Instruction FOR/NEXT. Une explication de chaque ligne pourrait être introduite dans le programme en utilisant l'instruction REM.

Dans une instruction FOR/NEXT, la variable est utilisée comme compteur : elle indique combien de fois la boucle va être exécutée. Dans une boucle FOR/NEXT, si la variable n'a pas atteint sa limite supérieure, l'exécution continue par l'instruction suivant FOR/NEXT. Si la variable a atteint sa limite supérieure, le programme sort de la boucle et exécute l'instruction suivant le NEXT. Lors de l'exécution du programme précédent le texte de la ligne 130 n'est affiché que lorsque la valeur de la variable excède 12.

L'avantage le plus important des instructions FOR/NEXT est de vous éviter d'avoir à vous préoccuper du compteur lors de l'écriture du programme. Si vous voulez tracer une série de lignes horizontales sur l'écran en utilisant les 16 couleurs, le programme suivant peut remplacer le programme COLORLOOP présenté plus haut :

```
NEW
3000 GR
3005 REM SETS VARIABLE RANGE TO 0-15
3010 FOR ROW=0 TO 15
3015 REM MAKES VALUE OF COLOR THE SAME AS VALUE OF
     VARIABLE
3020 COLOR=ROW
3025 REM DRAWS HLIN AT EACH ROW WITHIN RANGE
3030 HLIN 0,39 AT ROW
3035 REM WHEN LIMIT OF RANGE IS REACHED PROGRAM ENDS
3040 NEXT ROW
```

Pour sortir du mode graphique tapez TEXT et HOME **pour effacer l'écran. Regardez maintenant les deux programmes suivants, ils illustrent deux manières d'afficher les nombres pairs entre 0 et 12.** Le premier utilise l'instruction IF... THEN.

```
NEW
100 X = 0
110 PRINT X
120 X = X + 2
130 IF X <= 12 THEN GOTO 110
```

A la ligne 120, la valeur de X est augmentée de deux. Dans cette boucle la valeur de X croît de 2 chaque fois.

Le deuxième exemple montre comment l'on peut faire en utilisant cette fois la commande STEP dans l'instruction FOR.

STEP ne doit être utilisé que dans une instruction FOR/NEXT et est facultatif.

```
210 FOR X = 0 TO 12 STEP 2
210 PRINT X
220 NEXT X
```

Si vous exécutez votre programme, vous vous demandez peut-être pourquoi vous voyez deux suites de nombres pairs sur l'écran : tapez LIST et essayez de deviner pourquoi.

Réponse : les deux suites de nombre sont apparues sur l'écran parce que les lignes 100 à 130 ont été exécutées en même temps que les lignes 200 à 220 lorsque vous avez tapé RUN. Si vous ne voulez voir que le résultat du deuxième programme, tapez RUN 200. Pour supprimer un programme de la mémoire, vous pouvez, soit supprimer chaque instruction une à une, soit taper NEW.

La valeur précisée après STEP peut être un nombre quelconque traitable par l'Applesoft, c'est-à-dire entre −32768 et +32767. Un programme peut faire varier le compteur par pas de 5, 50 ou 500. Un programme peut également diminuer le compteur :

```
200 FOR X = 39 TO 15 STEP −3
```

Entrez cette ligne et faites-la exécuter en tapant :

```
RUN 200
```

Notez que les lignes 210 et 220 ont également été exécutées, car elles ont des numéros de ligne plus grands que celui indiqué dans l'instruction RUN.

Vous pouvez maintenant essayer d'écrire vos propres instructions FOR/NEXT. Plusieurs programmes utilisés dans la suite de ce manuel utiliseront également l'instruction FOR/NEXT.

Boucles emboîtées et enchevêtrées

L'instruction FOR/NEXT est très pratique mais présente quelques limitations. Par exemple les boucles FOR/NEXT peuvent être imbriquées, c'est-à-dire qu'une boucle peut être comprise dans une autre, mais elles ne peuvent pas s'enchevêtrer. Les figures 2-10 et 2-11 donnent des exemples de boucles emboîtées et enchevêtrées.

Figure 2-10. Boucles emboîtées. Le programme HUE

```
NEW
300 GR
310 FOR HUE = 1 TO 15
320 COLOR = HUE
330 FOR ROW = 0 TO 39
340 HLIN 0,39 AT ROW
350 NEXT ROW
360 COLOR = HUE - 1
370 FOR COLUMN = 0 TO 39
380 VLN 0,39 AT COLUMN
390 NEXT COLUMN
400 NEXT HUE
```

Le programme présent dans la figure 2-10 est un exemple d'emboîtement à deux niveaux, il est sur le disque APPLE SAMPLER sous le nom HUE. Faites-le exécuter et essayez de comprendre son fonctionnement avant de continuer.

Attention
Lorsque vous écrivez un programme utilisant des instructions FOR, rappelez-vous qu'*à chaque* FOR **doit être associé un** NEXT.

Le programme de la Figure 2-11 illustre ce qu'on appelle des boucles enchevêtrées.

Figure 2-11. Boucles enchevêtrées.

```
NEW
500 FOR N = 10 TO 20
510 PRINT N
520 FOR J = 30 TO 40
530 PRINT J
540 NEXT N
550 NEXT J
```

Ce programme ne fonctionne pas. Tapez TEXT, HOME puis RUN. C'est un exemple de ce qui se passe lorsque les boucles ne sont pas correctement emboîtées. Les boucles sont enchevêtrées et les nombres affichés ne sont pas bons, finalement un message d'erreur est affiché. Essayez de corriger ce programme en décroisant les boucles.

Le programme de dessin de tissu écossais présenté dans la figure 2-12 est un programme avec trois niveaux de boucles emboîtées. Ce programme figure sur le disque APPLESOFT SAMPLER et s'appelle QUILT. Ce programme n'utilise pas le nom COLOR comme nom de variable parce que vous vous souvenez que COLOR est un mot réservé de l'Applesoft.

Figure 2-12. Emboîtement à trois niveaux. Le programme QUILT.

```
NEW
400 GR
410 HUE = 0
420 FOR COLUMN = 0 TO 35 STEP 5
430 FOR LINE 3 0 TO 30 STEP 10
440 HUE = HUE + 1
450 IF HUE > 15 THEN HUE = 0
460 COLOR = HUE
470 FOR ROW = LINE TO LINE + 9
480 HLIN COLUMN, COLUMN + 4 AT ROW
490 NEXT ROW
500 NEXT LINE
510 NEXT COLUMN
```

Tentons maintenant une petite expérience. Tapez TEXT puis supprimez la ligne 400 en tapant 400 puis ↵ . Faites exécuter ce programme.

Lorsque vous supprimez la ligne 400, le programme s'exécute en mode texte ; si vous remettez la ligne 400, il s'exécute à nouveau en mode graphique. Vous pouvez faire cela avec n'importe quel programme utilisant le graphique.

Pause

Contrôle de la présentation des résultats

La virgule et le point-virgule sont utilisés par l'Applesoft pour produire différentes présentations des textes affichés par les programmes. Il y a également trois instructions APPLESOFT pour contrôler cet affichage. Les exemples présentés dans cette section vous permettront de vous familiariser avec ces différentes possibilités.

Tout d'abord entrez ce programme et regardez ce qu'il fait lorsque vous le faites exécuter :

```
NEW
100  PRINT ''MELLOW''
110  GOTO 100
```

Arrêtez maintenant ce programme par Ctrl - C et ajoutez une virgule à la fin de la ligne 100 :

```
100  PRINT ''MELLOW'',
```

Faites exécuter ce programme à nouveau. Le programme affiche maintenant le mot en colonne. Utilisez encore Ctrl - C pour interrompre le programme et remplacez la virgule par un point-virgule sur la ligne 100 :

```
100  PRINT ''MELLOW'';
```

Si vous faites exécuter ce programme, vous verrez que cette fois les mots MELLOW sont tous accolés. Le programme affichera MELLOW, tant que vous n'arrêterez pas par un Ctrl - C.

L'utilisation du point-virgule produit des affichages sans espace entre les mots ou les chiffres et l'utilisation de la virgule permet de produire des résultats par colonnes.

Modifiez votre programme en ajoutant l'instruction :

```
90 V = 99
```

et modifiez la ligne 100 de la manière suivante :

```
100  PRINT V
```

Faites exécuter ce programme. Recommencez en ajoutant une virgule à la ligne 100 :

```
100  PRINT V,
```

Puis remplacez la virgule par un point-virgule

```
100 PRINT V ;
```

Vous remarquez que la virgule et le point-virgule peuvent être utilisés également avec les nombres. La possibilité d'afficher des valeurs numériques non séparées par des espaces peut parfois être très utile.

Les virgules et les point-virgules peuvent être utilisés au milieu d'une instruction PRINT. Effacez votre précédent programme en tapant NEW puis entrez le programme :

```
100 BALLS = 3
110 STRIKES = 2
120 PRINT BALLS, STRIKES
RUN
```

Votre écran devrait afficher :

Vous pouvez rendre votre affichage plus clair en incluant un message dans l'instruction PRINT. Par exemple modifiez la ligne 120 de la manière suivante :

```
120   PRINT ''THE BALLS AND STRIKES ARE '';BALLS,STRIKES
```

et le programme affichera :

sauf si vous avez oublié l'espace après ARE. Une autre présentation peut être :

```
120   PRINT ''THE BALLS ARE  '';BALLS;'' AND THE STRIKES
      ARE '';STRIKES
```

La façon élégante de présenter ce résultat est peut être :

```
120   PRINT ''BALLS  '';BALLS;''  STRIKES  '';STRIKES
```

qui produit un résultat analogue à un tableau d'affichage :

Imaginons maintenant que vous vouliez afficher le mot HERE à partir de la dixième colonne de l'écran (l'écran a 40 colonnes). Vous pouvez utiliser l'instruction :

```
120 PRINT''         HERE''
```

en faisant très attention à bien taper neuf espaces avant le mot HERE. Vous pouvez également utiliser la fonction TAB qui permet d'introduire dans votre programme des tabulations analogues aux tabulations d'une machine à écrire. Essayez l'instruction :

La fonction TAB doit être utilisée avec l'instruction PRINT et permet de positionner le curseur à une colonne donnée sur l'écran.

```
120 PRINT TAB(10) ; ''HERE''
```

Pour voir TAB en action.

TAB doit être utilisé dans l'instruction PRINT. Si vous essayez une instruction :

```
105 TAB (5)
```

vous obtiendrez un message d'erreur. TAB doit être suivi d'un *paramètre* qui est un nombre ou une variable entre parenthèses. Lorsque TAB est ainsi utilisé avec une variable, il produit un nombre d'espaces égal à la valeur de cette variable. Par exemple :

```
NEW
200 FOR N = 1 TO 24
210 PRINT TAB(N) ; ''X''
220 NEXT N
```

Il existe deux autres instructions qui permettent de positionner le curseur sur l'écran.

VTAB déplace le curseur verticalement vers une ligne donnée de l'écran (de 1 à 24).

VTAB permet de déplacer le curseur verticalement et de le positionner sur l'une des 24 lignes de l'écran. La ligne supérieure est la ligne 1 et la ligne inférieure est la ligne 24. Contrairement à TAB, VTAB n'est pas utilisé dans l'instruction PRINT.

HTAB déplace le curseur horizontalement vers une colonne donnée de l'écran (de 1 à 40).

Vous pouvez également utiliser HTAB pour la tabulation horizontale. Il fonctionne comme TAB mais il permet de déplacer l'affichage aussi bien à gauche qu'à droite de la position d'affichage courante. HTAB ne doit pas être utilisé avec PRINT. Les colonnes de l'écran sont numérotées de 1 à 40, 1 étant la colonne la plus à gauche de l'écran.

Voici un programme qui montre comment HTAB déplace le curseur sur l'écran :

```
10 PRINT ''RUN''
20 HTAB 5
30 PRINT ''FASTER,''
40 HTAB 34
50 PRINT ''FASTER,''
60 HTAB 81
70 PRINT ''OR ELSE !''
```

Le programme SPACES du disque APPLESOFT SAMPLER illustre l'utilisation simultanée de HTAB et VTAB. Chargez-le dans la mémoire ou entrez-le à votre clavier :

```
NEW
590 HOME
600 FOR X = 1 TO 24
610 FOR Y = 1 TO X
620 HTAB X
630 VTAB Y
640 PRINT ''APPLE''
650 NEXT Y
660 NEXT X
670 GOTO 600
```

Avant de demander l'exécution de ce programme, essayez (et ce n'est pas facile !) de trouver ce qu'il fait.

Bien que TAB, HTAB et VTAB fonctionnent à peu près comme les coordonnées des instructions PLOT, elles présentent quelques différences.

1. Les 40 colonnes de l'écran sont numérotées de 1 à 40 pour les instructions TAB, VTAB et HTAB comme sur une machine à écrire alors que les coordonnées sont numérotées de 0 à 39 pour l'affichage graphique.

2. Le paramètre de VTAB peut varier de 1 à 24 seulement ; en effet, les caractères sont plus gros que les points utilisés dans le graphique à faible résolution et il n'y a donc que 24 lignes de texte.

3. La plus grande valeur qui peut être utilisée avec HTAB et VTAB est 255.

4. Si l'on utilise 0 ou un nombre trop grand (ou trop petit) avec TAB, VTAB ou HTAB on obtient un message d'erreur : ?ILLEGAL QUANTITY ERROR.

5. Si l'on utilise une valeur plus grande de 40 avec TAB ou HTAB on revient au début de la ligne suivante on appelle cela le *bouclage.*

Pour voir ce fonctionnement tapez :

```
NEW
300 FOR K = 1 TO 255
310 PRINT TAB(K) ; K
320 NEXT K
```

Normalement, on n'utilise pas TAB de cette manière ; il serait plus normal d'écrire cela en remplaçant les lignes 310 et 320 par :

```
310 HTAB K
320 PRINT K
```

et en ajoutant une ligne 330 :

```
330 NEXT K
```

Que se passe-t-il maintenant si vous remplacez HTAB par VTAB ? Comment pouvez-vous corriger ce programme pour respecter les limites de l'écran ?

Vous serez certainement de plus en plus à l'aise avec ce système. L'écran est toujours divisé en 24 lignes horizontales et 40 colonnes verticales. Les numéros de colonne varient suivant le mode de 1 à 40 en mode texte et de 0 à 39 en mode graphique. Les numéros de lignes changent parce qu'un caractère occupe plus de place que le point utilisé en mode graphique.

Voici un programme qui combine la plupart des instructions que vous avez utilisées dans ce Chapitre. Ce programme comporte quelques nouveautés pour vous : la variable N $ (vous en apprendrez plus sur ce type de variable dans le chapitre 5) et la ligne 70.

```
NEW
10 HOME
20 INPUT ''WHAT IS YOUR FIRST NAME?''; N$
30 PRINT
40 FOR A = 1 TO 5
50 PRINT TAB (A); ''HI, ''; N$; '', HOW ARE YOU ?''
60 NEXT A
70 END
```

L'instruction END termine l'exécution d'un programme et retourne le contrôle à l'utilisateur.

Affichez ce programme pour vérifier votre frappe puis faites-le exécuter. S'il vous plaît et que vous vouliez le conserver sur le disque, tapez :

```
SAVE WELCOME
```

puis CATALOG pour vérifier que le programme est bien sur le disque.

Résumé du chapitre

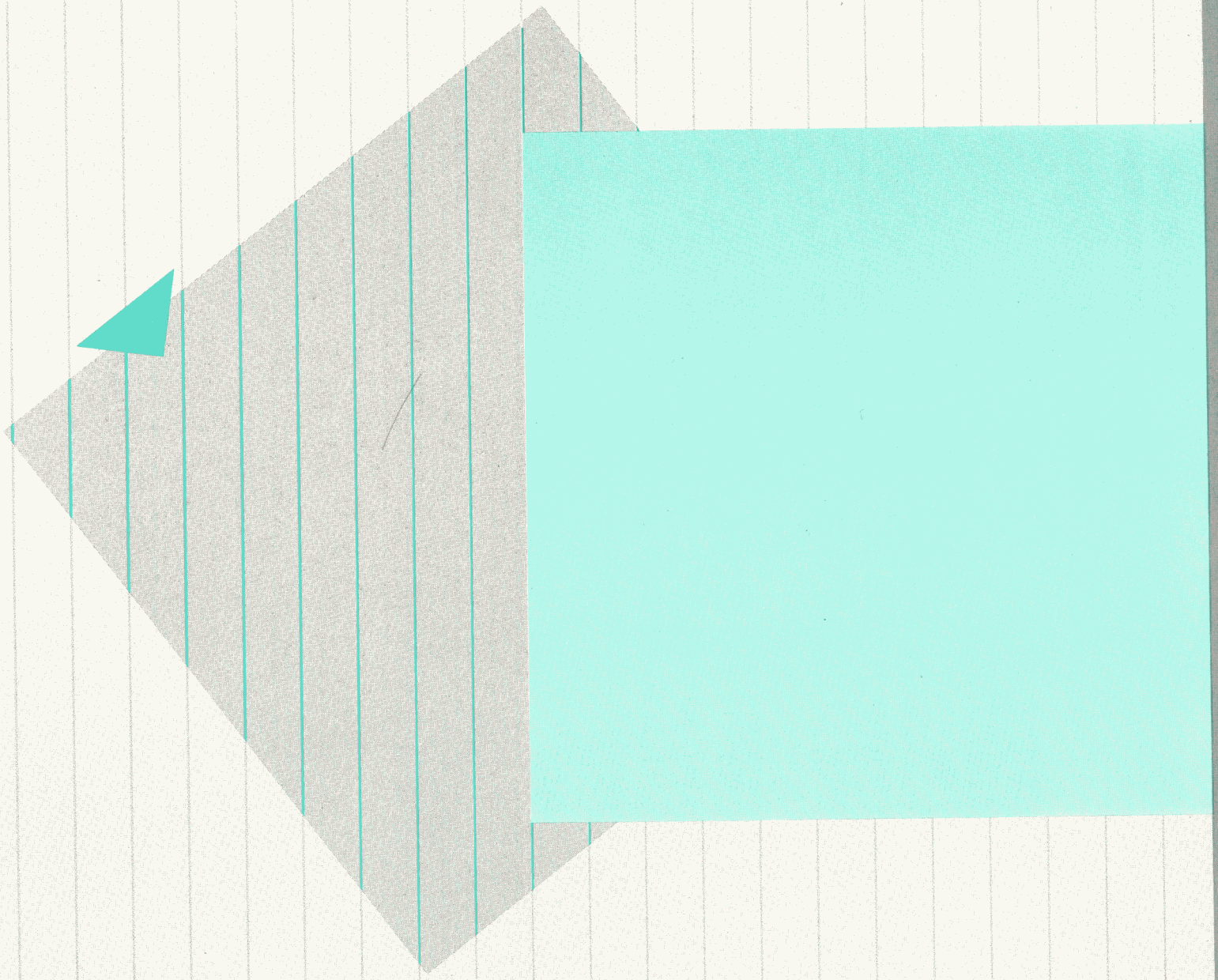
Instructions	Commandes DOS	Vocabulaire
NEW	RUN	exécuter (execute)
LIST	CATALOG	exécution immédiate (immediate execution)
GOTO	SAVE	mode programme (deferred execution)
CONT	LOAD	numéro de ligne (line number)
INPUT		programme (program)
IF... THEN		défilement (scroll)
REM		initialiser (initialize)
FOR/NEXT		par défaut (default)
STEP		protégé en écriture (write protected)
PRINT		binaire (binary)
TAB		langage machine (machine language)
VTAB		emboîté (nested)
HTAB		paramètre (argument)
END		bouclage (wrap around)

Touchés	Message d'erreur
Ctrl - C	BREAK IN 110

Chapitre 3

Modification de programmes

73 Déplacement du curseur : mode escape
73 Règles d'utilisation du mode escape
74 Un exercice pratique
76 D'autres commandes dans le mode escape
76 Limites du mode escape
77 Pour insérer du texte dans un ligne existante
82 Pour se débarrasser des lignes d'un programme
83 Pour mettre à jour des programmes volumineux
84 Un peu d'histoire
86 Résumé des possibilités pour entrer du texte



Modification de programmes

Jusqu'à présent vous avez appris plusieurs méthodes simples pour corriger les erreurs dans les programmes. Maintenant vous commencez à connaître la programmation et à écrire des programmes plus longs. Il est donc temps de vous donner quelques outils supplémentaires pour changer et éditer votre travail.

Simplement pour rafraîchir votre mémoire : vous avez appris que si vous faîtes une faute de frappe avant d'appuyer sur [↵], vous pouvez utiliser les touches [→] et [←] pour corriger l'erreur. [←] vous permet de reculer le curseur jusqu'à l'erreur et [→] de copier le reste de la ligne après que vous ayez apporté la correction.

Vous savez également que lorsque vous remarquez une erreur après avoir appuyé sur [↵] vous pouvez à nouveau retaper la ligne.

Déplacement du curseur : mode Escape

Une autre méthode est d'utiliser la touche [Esc] en combinaison avec les 4 touches de direction. Appelée *mode escape,* cette méthode vous permet de déplacer le curseur sur l'écran sans rien modifier à l'exception de la position du curseur.

Règles d'utilisation du mode Escape

- Pour entrer dans le mode escape appuyez sur la touche [Esc]. Pour quitter ce mode appuyez sur la touche [Espace] . Quand vous avez quitté le mode escape pour y revenir il suffit d'appuyer sur la touche [Esc].

- En utilisant le mode escape pour corriger ou modifier quelque chose, vous devez positionner le curseur sur le premier chiffre du numéro de la ligne que vous désirez corriger.

- Ce mode vous permet de corriger seulement une ligne de programme à chaque fois.

- Après avoir modifié une partie de la ligne, il faut recopier le reste de la ligne en actionnant la touche →.

- Les touches → et ← n'ont pas le même effet en mode escape et en mode normal. En mode escape, elles déplacent le curseur sans modifier les caractères sur l'écran. Les touches spéciales de déplacement du curseur utilisées dans le mode normal (donc sans actionner préalablement la touche Esc) effacent ←, et recopient →, les caractères qu'elles rencontrent.

Un exercice pratique

Il y a plusieurs erreurs dans les lignes qui suivent. Si vous suivez les instructions pas à pas, vous aurez une idée de la façon dont fonctionne le mode escape.

Ce que vous faites...	**Ce qui se passe...**
Tapez exactement ce qui suit : NEW 10 PRUNT ''THE MOCKINGBIRD'' 20 PRINT TAB (5) ; ''SINGS'' 30 PRINT RAB (3) : ''IN SPRING''	toutes les erreurs que vous tapez apparaissent sur l'écran
Tapez LIST 10	10 P RUN T''THE MOCKINGBIRD''
Enfoncez une fois et relachez la touche Esc	Cela vous place dans le mode ''escape''.
Actionnez deux fois la touche ↑	Le curseur se déplace à la ligne 10.

```
]LIST 10
10 P RUN T"THE MOCKINGBIRD"

]
```

Actionnez une fois la touche ←. Le curseur doit se placer sur le premier caractère.

Remarquez le déplacement du curseur, et pourtant, rien n'est modifié sur l'écran quand vous êtes dans le mode escape. (Vous ne pouvez pas le voir, mais rien non plus n'est modifié dans la mémoire de l'ordinateur).

Actionnez la touche Espace

Cela vous fait quitter le mode escape.

Actionnez six fois la touche →

La touche → recopie les caractères si vous n'êtes pas dans le mode escape (ainsi vous n'avez pas à les retaper).

Remplacez le U **par un** I.

Copiez les caractères restant de la ligne en utilisant la touche →. Lorsque vous aurez atteint la fin de la ligne, actionnez la touche ↵.

Tapez LIST 10 pour faire apparaître la ligne corrigée.

```
]LIST 10
10  PRINT "THE MOCKINGBIRD"
]
```

Si maintenant vous lanciez l'exécution du programme, le message ?SYNTAX ERROR IN 30 apparaîtra sur l'écran car il y a deux erreurs dans la ligne 30 que vous n'avez pas encore corrigées. RAB devrait être TAB, et les deux points devraient être un point virgule. Pour corriger la ligne 30, tapez d'abord LIST, puis placez-vous dans le mode escape en actionnant la touche Esc et utilisez les touches spéciales de déplacement du curseur pour positionner le curseur au début de la ligne 30.

Suivez l'enchaînement des opérations que vous avez utilisé pour la ligne 10 : quittez le mode escape, utilisez la touche ← pour placer le curseur sur les caractères erronés, corrigez-les et recopiez le reste de la ligne. Listez à nouveau pour vous assurez que toutes les erreurs ont bien été corrigées, puis lancez l'exécution du programme.

Il peut vous paraître un peu compliqué, tout au moins dans un premier temps, de vous servir du mode escape, mais avec un peu de maîtrise, cela deviendra beaucoup plus facile. C'est sûr !

D'autres commandes dans le mode Escape

Trois autres commandes sont utiles si vous devez introduire beaucoup de texte. Elles sont toutes actives dans le mode escape.

- Esc @ (enfoncez la touche Esc, laissez-la revenir puis servez-vous de la touche @ pour produire le caractère @ efface entièrement l'écran comme le fait l'instruction HOME.
- Esc F efface toute la partie de l'écran comprise entre le curseur et le bas de l'écran.
- Esc E efface la ligne à partir de la position du curseur.

Limites du mode Escape

Maintenant que vous avez utilisé avec succès le mode escape pour corriger quelques lignes d'un programme, vous pouvez prendre un peu de repos. Placez vous dans le mode escape et jouez avec les quatre touches spéciales de déplacement du curseur : déplacez le curseur vers le haut de l'écran, vers le bas, remontez au milieu, vers le côté droit et tout autour.

Que se passe-t-il lorsque le curseur atteint le côté droit de l'écran ? (Réponse : il vient sur le côté gauche. Vous avez déjà vu cela quand vous avez utilisé TAB et HTAB). Que se passe-t-il si vous actionnez accidentellement d'autres touches alors que vous êtes dans le mode escape ? Essayez donc. Découvrir dès maintenant ce qui se passe vous évitera plus tard quelques tracas. Lorsque vous en aurez assez de déplacer le curseur tout autour de l'écran, actionnez la touche Espace pour quitter le mode escape.

Conseil utile : Si vous ne placez pas le curseur sur le premier caractère du numéro de la ligne que vous voulez corriger, vous perdrez quelques caractères de la ligne. Si vous ne le croyez pas, essayez-le donc !

Si vous avez d'abord corrigé la ligne 10, puis listé et corrigé la ligne 30, c'est qu'il est difficile de corriger plus d'une ligne à la fois en mode escape.

Pour insérer du texte dans une ligne existante

Vous remarquerez, parfois, qu'ajouter quelque chose à une ligne d'un programme améliore un peu ce dernier. Supposons, par exemple, que vous vouliez insérer l'instruction TAB (10) après PRINT dans la ligne

90 PRINT ''THIS IS A SHORT PROGRAM''

Suivez pas à pas les instructions qui suivent pour comprendre comment utiliser l'insertion de texte

Ce que vous faites...	**Ce qui se passe...**
Tapez NEW 90 PRINT ''THIS IS A SHORT PROGRAM''	
Utilisez RUN **pour visualiser l'aspect de la ligne**	

```
]RUN
THIS IS A SHORT PROGRAM

]
```

Tapez LIST 90

```
]LIST 90
90  PRINT "THIS IS A SHORT PROGRA
    M"

]
```

Actionnez la touche [Esc]	Cela vous place dans le mode escape.
Actionnez la touche [↑] **trois fois et une fois la touche** [←]	Cela positionne le curseur sur le premier caractère du numéro de la ligne. Le curseur doit être sur le premier caractère pour que les corrections ou les insertions se passent normalement.
Actionnez la touche [Espace]	Cela fait quitter le mode escape.
Actionnez la touche [→] **dix fois.**	Cela positionne le curseur à l'endroit de la ligne où vous voulez insérer quelque chose.

```
]LIST 90
90  PRINT "THIS IS A SHORT PROGRA
    M"

]
```

Actionnez [Esc].	Cela vous ramène dans le mode escape.
Actionnez une fois la touche [↑]	Cela positionne le curseur juste au-dessus de la ligne, ce qui vous permet d'ajouter du texte à la ligne.

```
]LIST 90
90  PRINT "THIS IS A SHORT PROGRA
    M"

]
```

Actionnez une fois la touche [Espace]	Cela vous fait quitter le mode escape.

Tapez TAB (10);

C'est le texte que vous ajoutez à la ligne.

```
]LIST 90
                TAB (10);
90  PRINT "THIS IS A SHORT PROGRA
    M"

]
```

Actionnez Esc

Cela vous ramène dans le mode escape.

Actionnez une fois la touche ↓.

```
]LIST 90
                TAB (10);
90  PRINT "THIS IS A SHORT PROGRA
    M"

]
```

Actionnez huit fois la touche ←.

Cela ramène le curseur au point où vous avez commencé l'insertion. Cela vous permet, après avoir quitté le mode escape, de recopier le reste de la ligne avec la touche →.

```
]LIST 90
                TAB (10);
90  PRINT "THIS IS A SHORT PROGRA
    M"

]
```

Actionnez la touche Espace

Cela vous ramène dans le mode escape.

Actionnez 23 fois la touche → jusqu'à ce que le curseur soit sur l'espace suivant la lettre A **de** PROGRAM.

```
]LIST 90
                TAB (10);
90  PRINT "THIS IS A SHORT PROGRA
    M"

]
```

Actionnez la touche Esc

Cette action, comme les deux qui suivent, permet de déplacer le curseur vers la droite sans recopier les espaces entre PROGRA et M. Reportez-vous à **conseil utile** (qui suit) pour plus de détails.

Actionnez la touche → jusqu'à ce que le curseur soit sur le M.

```
]LIST 90
          TAB (10);
90  PRINT "THIS IS A SHORT PROGRA
    M"

]
```

Actionnez la touche Espace

Cela vous fait quitter le mode escape.

Recopiez le reste de l'instruction grâce à la touche → et actionnez ↵. Listez (par LIST) la ligne.

```
]LIST 90
          TAB (10);
90  PRINT "THIS IS A SHORT PROGRA
    M"

]LIST 90

90  PRINT  TAB( 10);"THIS IS A SH
    ORT PROGRAM"

]
```

Exécutez le programme (par RUN) pour apprécier la différence avec l'ajout de TAB (10).

```
]RUN
          THIS IS A SHORT PROGRAM

]
```

Maintenant essayez d'ajouter VERY devant SHORT dans la même ligne. Lorsque vous aurez terminé, l'instruction devra être semblable à la liste suivante

```
]LIST
90  PRINT  TAB( 10);"THIS IS A VE
    RY SHORT PROGRAM"

]
```

Conseil utile : Applesoft utilise une grille de 40 colonnes de large. C'est à dire qu'une ligne de 40 caractères peut être affichée sur une seule ligne. Toute ligne comportant plus de 40 caractères est automatiquement coupée et terminée sur la ligne suivante. Lorsque vous modifiez une telle ligne, vous devez utiliser le mode escape pour que la ligne corrigée ne comporte pas les espaces à droite de la ligne automatiquement scindée. C'est ce que vous avez fait lorsque vous êtes revenu au mode escape entre le A et le M de PROGRAM. Lorsque le curseur est placé sur le premier caractère de la deuxième ligne, le mode escape abandonné, utilisez la touche → pour achever la recopie.

Une autre instruction permet aussi de diminuer la largeur de l'écran et de supprimer l'ajout de ces espaces ; c'est POKE 33,33. Vous n'avez pas besoin de la connaître pour l'instant, mais vous pourrez l'étudier plus tard et dans le détail dans le *Manuel de référence d'Applesoft* (Applesoft Reference Manual).

Pour se débarrasser des lignes d'un programme

Naturellement, il peut arriver que vous décidiez qu'une ligne d'un programme ne vous convienne pas et que vous vouliez vous en débarrasser. Il y a plusieurs façons de supprimer des lignes d'un programme.

Si vous avez juste terminé de frapper la ligne et décidé qu'elle ne vous convenait plus avant d'avoir actionné [↵], vous pouvez utiliser le caractère de contrôle : [Ctrl] - X. Essayez-le en tapant la ligne plus bas. Avant de taper [↵], enfoncez la touche [Ctrl] tout en tapant sur la touche marquée X. Si vous ne faites pas attention et que vous tapez [↵] avant d'avoir utilisé [Ctrl] - X, vous n'aurez qu'à taper NEW et à recommencer.

[Ctrl] - X détruit une ligne seulement avant d'avoir actionné la touche [↵]

```
10 PRINT ''WHY AM I DOING THIS ?''
```

```
10 PRINT ''WHY AM I DOING THIS ?''\
```

Comme vous pouvez le voir dans l'illustration et comme cela apparaît sur l'écran (vous êtes bien en train de faire les exercices, n'est-ce pas ?), un slash (\) apparaît à la fin de la ligne au moment où vous utilisez [Ctrl] - X. Si vous essayez de lister la ligne, rien n'apparaît. Vous avez sûrement supprimé cette ligne !

Maintenant tapez

```
20 PRINT ''AN ELEPHANT JUST WALKED IN''
```

et actionnez [↵]. Si vous décidez de modifier ou de supprimer cette ligne maintenant, vous avez le choix entre plusieurs possibilités. La première est de remplacer la ligne en retapant sur celle-ci une autre instruction, comme par exemple

```
20 PRINT ''THIS IS VERY SILLY''
```

Cependant si vous voulez éliminer complètement la ligne, vous pouvez taper

```
20
```

et elle disparaîtra. Si vous en doutez, tapez LIST 20.

La deuxième façon de procéder est de taper l'instruction

```
DEL 20,20
```

L'instruction DEL retire ou supprime, de vos programmes, les lignes spécifiées.

qui demande à l'ordinateur de détruire une ou plusieurs lignes. Comme il est, bien évidemment, plus simple de taper 20 que de taper DEL 20,20, vous devez sans doute vous demander pourquoi l'instruction DEL est si remarquable. Cela ne vous sera pas évident tant que vous ne voudrez pas supprimer plusieurs lignes de votre programme. Au lieu de taper séparément chaque numéro de ligne, vous pouvez taper

```
DEL 10,90
```

Ce qui supprimera toutes les instruction comportant un numéro de ligne entre 10 et 90, bornes incluses.

La commande DEL d'Applesoft ressemble à la commande DELETE utilisée par le ''gestionnaire du disque'' (Disk Operating System DOS) mais ce n'est pas la même chose.

DELETE est toujours employée pour effacer du disque des programmes complets et est toujours suivie par le nom du programme que vous voulez retirer du disque.

DEL, comme vous venez de l'apprendre, est suivie de deux numéros de lignes séparés par une virgule. Si vous confondez les deux commandes et utilisez la mauvaise, ne vous inquiétez pas. Apple *II*e vous le rappellera par un message d'erreur.

La touche Del du clavier d'Apple *II*e est différente des commandes DEL ou DELETE. Reportez-vous au *Guide de l'Utilisateur Apple II*e et au *Manuel de Référence d'Apple II*e pour plus de détails.

DELETE est une commande DOS qui retire du disque le programme dont le nom a été spécifié.

Pour mettre à jour des programmes volumineux

Tout l'intérêt d'utiliser le mode escape pour faire des modifications vous deviendra évident quand vous travaillerez sur des lignes très longues ou des programmes très longs. Plus vous écrirez vos programmes et plus vous utiliserez le mode escape.

Le mode escape présente plusieurs avantages :

- Il est plus rapide de l'utiliser pour mettre à jour une ligne particulièrement longue d'un programme (par exemple, 230 caractères).
- Il est plus facile de fusionner deux programmes qui ont un grand nombre de lignes identiques.
- Vous pouvez lister une ligne qui apparaît plus d'une fois sous la même forme, en faisant les changements nécessaires, et en la recopiant pour une reproduction plus simple.

Cependant, ces utilisations du mode escape sont un peu prématurées. Un sujet plus simple pour le moment est l'utilisation de Ctrl - S.

Ctrl - S arrête et relance la liste du programme.

Quand un programme comportant beaucoup de lignes est listé, les lignes défilent trop vite pour la plupart d'entre nous, ce qui nous empêche de les lire. Une façon de résoudre cette difficulté est de lister le programme par morceaux, en tapant, par exemple

LIST 10,100

et puis

LIST 110,200

et puis

LIST 210,300

et ainsi de suite jusqu'à ce que vous ayez examiné toutes les lignes du programme.

Une autre méthode, plus simple, est de maîtriser le défilement des lignes en utilisant Ctrl- S pour arrêter le défilement et le relancer lorsque vous avez examiné les lignes sur l'écran. Cela prend un moment avant d'arriver à actionner assez vite Ctrl et S, mais cela en vaut la peine.

Un peu d'histoire

Huit autres touches, à côté des quatre touches de déplacement du curseur, peuvent servir pour entrer du texte. Si vous connaissez bien les modèles plus anciens d'Apple, vous devez savoir à quoi servent ces touches. De toute façon, un peu d'histoire est utile.

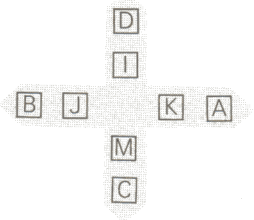
Dans la première version d'Apple II, il n'y avait, sur le clavier, ni les touches ↑ ni ↓ et le mode escape n'existait pas. L'entrée du texte se faisait en alternant l'usage de la touche Esc avec les touches A, B, C, et D : A déplaçait le curseur vers la droite, B le déplaçait vers la gauche, C le déplaçait vers le bas et D le déplaçait vers le haut. Si, par exemple, vous vouliez déplacer cinq fois le curseur vers le haut et placer quatre espaces à sa gauche, vous deviez taper, dans l'ordre

Esc D Esc D Esc D Esc D Esc D
Esc B Esc B Esc B Esc B

Pas simple ! Vous imaginez qu'il était facile de se tromper.

La façon de faire fut changée par la suite : une seule frappe sur la touche Esc devint suffisante pour se placer dans le mode escape. Cependant, le clavier n'avait pas encore de touche de déplacement du curseur. Les quatre touches I, J, K et M furent utilisées pour déplacer le curseur. En regardant le clavier, vous vous apercevrez que ces touches sont disposées en forme de croix. Leur action est en rapport avec leur position : I déplace le curseur vers le haut, J le déplace vers la gauche, M le déplace vers le bas et K le déplace vers la droite.

Figure 3-1. Touches de direction.



Bien qu'il fut ajouté des touches de déplacement du curseur, I, J, K et M fonctionnent toujours dans le mode escape. Vous pouvez toujours vous servir des touches A, B, C, et D, si vous faites précéder chaque frappe d'une frappe sur la touche Esc. Si vous avez appris l'une des anciennes méthodes, vous pouvez toujours pratiquer de la même façon et alors ne vous forcez pas à utiliser les touches spéciales de déplacement du curseur. Il se peut également qu'une des anciennes méthodes vous convienne mieux. Apple *IIe* a été conçu de telle sorte que les trois touches déplacent le curseur dans la même direction. Figure 3-1 visualise ces directions.

Résumé des possibilités pour entrer du texte

En mode escape

Pour se placer dans le mode escape :	Actionnez la touche [Esc]
Pour quitter le mode escape :	Actionnez la touche [Espace]
Pour déplacer le curseur vers le haut :	Utilisez [↑], I, i, D ou d
Pour déplacer le curseur vers le bas :	Utilisez [↓], M, m, C ou c
Pour déplacer le curseur vers la gauche :	Utilisez [←], J, j, B ou b
Pour déplacer le curseur vers la droite :	Utilisez [→], K, k, A ou a
Pour effacer la ligne à partir de la position du curseur :	Actionnez [Esc] - E
Pour effacer l'écran à partir de la position du curseur :	Actionnez [Esc] - F
Pour effacer tout l'écran :	Actionnez [Esc] - @

Hors du mode escape

Pour annuler un caractère :	Actionnez la touche [←]
Pour recopier ou retaper un caractère :	Actionnez la touche [→]
Pour annuler une ligne avec d'avoir actionné	Actionnez [Ctrl] - X
Pour arrêter la liste d'un programme :	Actionnez [Ctrl] - S
Pour relancer la liste d'un programme :	Actionnez [Ctrl] - S
Pour supprimer une ou plusieurs lignes :	Utilisez l'instruction DEL
Pour supprimer du disque tout un programme :	Utilisez la commande DELETE

Du graphique

89 Construction d'un jeu simple
90 Instructions multiples dans une ligne
91 Création du mouvement
92 Limite de l'écran
93 Création d'impressions visuelles
93 L'article entier
94 Programme avec intervention des utilisateurs
97 Production de bruits
100 Bruit de la balle au rebond
100 Nombres aléatoires
103 Simulation d'une paire de dés
103 Graphiques aléatoires
104 Les sous-programmes : assemblage des éléments
108 Trace
109 Un meilleur programme de tracé de chevaux
110 Les erreurs
110 Les variables
111 Sous-programmes complémentaires
112 Un programme bien structuré
113 Graphique haute résolution
120 Résumé du chapitre

Du Graphique

Maintenant que vous disposez de quelques outils supplémentaires de mise à jour, vous êtes probablement pressé de vous attaquer de plus près à la programmation. Ce chapitre vous fera faire un grand bond en avant en vous guidant dans la construction et les modifications de programmes plus longs et plus compliqués. Vous verrez comment des programmes sont construits morceau par morceau et comment les instructions travaillent en tant qu'éléments de la construction.

Ne croyez pas être capable de faire tout ceci par vous-même correctement ! Apprendre à programmer ne va pas de soi ; plus vous travaillerez les programmes de ce manuel, mieux vous saurez vous y prendre pour votre propre compte.

Construction d'un jeu simple

Ce paragraphe présente un certain nombre d'éléments ou parties de programme utilisés de façon typique dans les jeux vidéo. Il montre une méthode pour utiliser la couleur afin de créer des impressions visuelles sur l'écran.

Voir Chapitre 2, « Utilisation de l'APPLESOFT SAMPLER COLORLOOP » pour plus d'information.

Avant d'aller plus loin, chargez le programme COLORBOUNCE d'APPLESOFT SAMPLER en tapant

```
LOAD COLORBOUNCE
```

Puis, pour travailler COLORBOUNCE, exécutez le en tapant RUN.

Si vous n'utilisez pas d'unité de disque, voyez « l'article entier » de ce chapitre. Tapez le programme, testez-le et exécutez-le.

Pour arrêter le programme, appuyez sur Ctrl - C. Maintenant lisez les quelques paragraphes suivants pour découvrir le fonctionnement de ce programme. Chaque paragraphe présente un élément différent du programme.

Instructions Multiples dans une Ligne

Jusqu'à maintenant la plupart des lignes de programmes sur lesquelles vous avez travaillé ne renfermaient qu'une instruction par ligne. Avec cette méthode vous pouvez commencer un programme par (souvenez-vous qu'il vous suffit de lire ce paragraphe sans utiliser votre Appel *II*e).

```
400 REM MODE GRAPHIQUE
420 GR
```

Cependant, l'utilisation du deux points (:) entre des instructions vous permet d'associer plusieurs instructions sur un seul numéro de ligne dans un programme. Ainsi :

```
400 GR : REM MODE GRAPHIQUE
```

Cela rend parfois les lignes de programmes plus faciles à comprendre si les commentaires sont sur la ligne concernée, comme dans la ligne 400. D'autres fois, il est meilleur que la remarque soit sur une ligne différente. COLORBOUNCE utilise les deux méthodes.

Le deux points peut être utilisé pour deux instructions quelconques. Voici d'autres exemples :

```
390 HOME : PRINT ''THAT WASN'T BETWEEN 1 AND 15 !'':
    PRINT
720 COLOR = 0 : PLOT X, Y
760 X = NX : Y = NY
```

La ligne 390 associe trois instructions dont la signification est la suivante : HOME vide l'écran, le premier PRINT affiche un message pour l'utilisateur, et le second PRINT saute une ligne après le message. A la ligne 720, la couleur noire est retenue chaque fois que X et Y sont représentés ; le paragraphe « Création d'impression visuelles » donne davantage d'explications sur ce sujet. La ligne 760 donne à la variable X la valeur de la variable NX, et à la variable Y la valeur de la variable NY.

Une ligne peut contenir plusieurs instructions séparées par le caractère « deux points », les commentaires éventuels doivent être placés en fin de ligne.

Création du mouvement

La balle bondissante, qui en réalité et un pavé visualisé se déplaçant sur la grille, est créée au moyen de six variables dans COLORBOUNCE :

X représente le point de départ du mouvement latéral
Y représente le point de départ du mouvement vertical
XV représente la vitesse de X
YV représente la vitesse de Y
NX représente la position courante de X (NX = X+XV)
NY représente la position courante de Y (NY = Y+YV)

Les lignes de programme qui introduisent ces variables dans COLORBOUNCE s'identifient presque aux descriptions que vous venez de lire. L'utilisation des instructions REM en clair est d'une aide certaine.

```
440 X = 0 : REM SET STARTING POSITION OF BACK-AND-
    FORTH VARIABLE
460 Y = 5 : REM SET STARTING POSITION OF UP-AND-DOWN
    VARIABLE
480 XV = 2 : REM SET X VELOCITY
500 YV = 1 : REM SET Y VELOCITY
520 REM CALCULATE NEW POSITION
540 NX = X + XV : NY = Y + YV
```

Conseil utile : Si vous trouvez que plusieurs instructions par ligne de programme sont une source de confusion, il est parfaitement autorisé de développer. Dans cette partie de programme, vous aurez par exemple 11 lignes au lieu de 6.

Une des raisons principales de l'association des instructions est l'économie de place pour les longs programmes : plus il y a de lignes dans un programme, plus il y a besoin de mémoire. Une autre raison est que l'exécution d'un programme est plus rapide s'il y a plusieurs instructions par ligne. Une troisième raison est que certaines instructions, comme IF...THEN **contrôlent** toutes les instructions de la même ligne ; ainsi le programme travaille **en fonction** de la position des instructions. Quand vous commencerez à écrire pour vous des programmes plus longs, vous trouverez des « recettes » utiles pour économiser place et temps dans le *Manuel de Référence Applesoft.*

Limites de l'écran

Pour commencer à représenter des points sur la grille graphique vous devez connaître deux éléments essentiels d'information : la grille comporte 40 points horizontaux et 40 points verticaux ; le système de numérotage de la grille va de 0 à 39 dans les deux directions. Pour conserver une balle bondissante à l'intérieur de ces limites, vous avez deux choses à faire :

1. Initialiser ces limites dans le programme pour que la balle ne semble pas sortir de l'écran.

2. Changer la direction de la balle quand elle atteint le bord de l'écran.

Les lignes suivantes de COLORBOUNCE font ces deux choses. Examinez-les attentivement de façon à comprendre la méthode de travail.

```
560 REM IF BALL EXCEEDS SCREEN EDGE, THEN BOUNCE
580 IF NX > 39 THEN NX = 39 : XV = -XV
600 IF NX < 0 THEN NX = 0 : XV = -XV
620 IF NY > 39 THEN NY = 39 : YV = -YV
640 IF NY < 0 THEN NY = 0 : YV = -YV
```

Il n'est pas facile de représenter ceci. Une chose à se rappeler est le fait que les variables ne sont pas statiques : si sur une ligne on a XV = - XV, le contenu de la variable prend réellement une valeur négative.

En d'autres termes, quand la table atteint la limite de l'écran, en 39, la ligne 580 change XV en la valeur négative de XV. Puis, quand la ligne 540 est atteinte dans la boucle suivante, NX passe à 38. Dans la boucle suivante, NX passe à 37 et ainsi de suite, jusqu'à ce que la balle atteigne l'autre bord de l'écran ; à partir de ce point 1 est ajouté à la valeur de NX à chaque boucle. Voilà comment on change de direction dans le déplacement latéral.

Création d'impressions visuelles

En alternant entre le noir et une couleur pour la balle à chaque fois qu'une nouvelle position est représentée en X et Y, vous pouvez effectuer un effacement apparent, donnant l'impression visuelle d'un mouvement de la balle. Ceci est réalisé dans les lignes suivantes de COLORBOUNCE :

```
660  REM PLOT NEW POSITION
680  COLOR = 7 : PLOT NX, NY
700  REM ERASE OLD POSITION
720  COLOR = 0 : PLOT X, Y
740  REM SAVE CURRENT POSITION
760  X = NX : Y = NY
780  GOTO 540
```

L'instruction PLOT NX, NY de la ligne 680 fait apparaître un « pavé » (autrement dit une balle) dans la couleur indiquée à la position nouvelle NX et NY. A la ligne 720 le tracé sur X et Y est en réalité un tracé sur les **anciennes** coordonnées NX et NY qui ont été sauvegardées après la mise en place du « pavé » précédent (ligne 760). L'instruction COLOR = 0 de la ligne 720 met en place la couleur noire, donnant ainsi l'impression que le pavé a été effacé. Ceci est dû au fait que le tracé s'effectue rapidement. Naturellement, la boucle suivante replace la couleur courante et effectue le tracé en NX et NY avec cette couleur (ligne 680). L'alternance du noir et de la couleur à chaque boucle améliore l'apparence du mouvement.

Si vous ne comprenez pas l'ordre de ces instructions, essayez de les déplacer et d'exécuter le programme pour voir ce que cela entraîne.

L'article entier

Voici un listing de COLORBOUNCE. C'est exactement le même programme que celui d'APPLESOFT SAMPLER. Listez ce programme pour comparer les versions. Vous pouvez soit le lister par parties, par exemple avec :

```
LIST 400,580
LIST 600,780
```

soit le lister et utiliser Ctrl - S pour stopper et faire repartir le listing. Allez, essayez le !

```
400 GR : REM SET COLOR GRAPHICS AREA
420 HOME : REM CLEAR TEXT AREA
440 X = 0 : REM SET STARTING POSITION OF BACK-AND-
    FORTH VARIABLE
460 Y = 5 : REM SET STARTING POSITION OF UP-AND-DOWN
    VARIABLE
480 XV = 2 : REM SET X VELOCITY
500 YV = 1 : REM SET Y VELOCITY
520 REM CALCULATE NEW POSITION
540 NX = X + XV : NY = Y + YV
560 REM IF BALL EXCEEDS SCREEN EDGE, THEN BOUNCE
580 IF NX > 39 THEN NX = 39 : XV = -XV
600 IF NX < 0 THEN NX = 0 : XV = -XV
620 IF NY > 39 THEN NY = 39 : YV = -YV
640 IF NY < 0 THEN NY = 0 : YV = -YV
660 REM PLOT NEW POSITION
680 COLOR = 7 : PLOT NX, NY
700 REM ERASE OLD POSITION
720 COLOR = 0 : PLOT X, Y
740 REM SAVE CURRENT POSITION
760 X = NX : Y = NY
780 GOTO 540
```

En résumé, voici comment COLORBOUNCE fonctionne : le programme affiche un pavé, en efface un, affiche un autre pavé, une colonne au-delà, efface le précédent et ainsi de suite. Quand le bord de l'écran est atteint (lignes 580-640 quand NX et NY sont supérieurs à 39 et quand NX et NY sont inférieurs à 0, le programme inverse la direction du tracé.

Programme avec intervention des utilisateurs

Supposons que vous décidiez de modifier COLORBOUNCE de façon à introduire une nouvelle couleur de balle à chaque exécution du programme. Une solution serait de changer la ligne 680, d'exécuter le programme, puis de changer le ligne 680 à nouveau. Mais ce serait beaucoup de peine et ce serait difficile à expliquer si vous voulez le montrer à un ami.

Une meilleure méthode est d'utiliser l'instruction INPUT pour permettre à l'utilisateur d'intervenir dans le programme. Vous avez vu dans le Chapitre 2 : qu'une instruction INPUT contient un nombre de variables et demande à la personne d'utiliser le programme pour entrer quelque chose à partir du clavier.

Vous devrez faire très attention à la manière de formuler l'instruction, car COLOR est un mot réservé. Si par exemple vous ajoutez la ligne

```
350  INPUT COLOR
```

le programme deviendrait mauvais : COLOR ne peut pas être utilisé, comme nom de variable. Vous pouvez, cependant, écrire la ligne 350 comme suit :

```
350  INPUT ''COLOR? ''; HUE
```

et changer la ligne 680 pour définir COLOR = au moyen de la variable HUE :

```
680  COLOR = HUE : PLOT NX, NY
```

Si vous ne l'avez pas fait, ajoutez ces lignes à COLORBOUNCE. Puis exécutez-le pour voir comment fonctionnent ces lignes. (Quand le programme exécute la ligne 350, le mot COLOR, suivi d'un point d'interrogation (?) apparaît sur l'écran. Le curseur restera clignotant jusqu'à ce que quelqu'un tape un nombre et appuie sur ↵).

Vous savez quoi faire quand apparaît le point d'interrogation, mais votre ami peut ne pas le savoir. Ce serait donc une bonne idée que ce soit le programme qui dise à votre ami, (l'utilisateur), ce qui est attendu. L'addition de quelques instructions PRINT et le passage en mode TEXT en début de programme feront l'affaire :

```
280  REM SET TEXT MODE
300  TEXT : HOME
310  PRINT ''TO SELECT A COLOR FOR''
320  PRINT ''THE BOUCING BALL, FIRST TYPE''
330  PRINT ''IN ANY NUMBER FROM 1 TO 15.''
340  ''THEN PRESS THE KEY LABELED RETURN.'' : PRINT
```

De même un message plus spécifique accompagnant l'instruction INPUT pourrait être une aide :

```
350  INPUT ''WHAT COLOR WOULD YOU LIKE THE BALL TO BE
     (1-15)? ''; HUE
```

Pour l'écriture de l'instruction INPUT :

- Le message doit être placé entre guillemets dans une instruction INPUT.
- Si l'instruction INPUT comporte un message, l'ordinateur n'ajoute pas de point d'interrogation. Si vous voulez qu'un point d'interrogation apparaisse, vous devez l'inclure dans le message accompagnant INPUT.
- La mise en place d'un espace à la suite du point d'interrogation à l'intérieur des guillemets sépare la réponse de la question.
- Il doit y avoir un point virgule entre le message et le nom de variable.

L'addition des nouvelles lignes 280 à 350 à COLORBOUNCE rendra le programme beaucoup plus agréable pour les utilisateurs. Il y a cependant encore des pièges potentiels pour les gens qui n'ont pas l'habitude des ordinateurs. Qu'arrive-t-il si on fait une faute et on tape ↵ ? Des messages d'erreur immédiats et des sonneries !

Si on entre un nombre trop grand ou trop petit, soit le programme laissera la balle se déplacer vers le côté droit et s'arrêtera, soit le message

```
?ILLEGAL QUANTITY ERROR IN 680
```

apparaîtra sur l'écran et le programme s'arrêtera ensuite. Dans la plupart des cas, vos amis ne sauront pas comment relancer l'ordinateur — et ils ne devront pas avoir à le faire. Vous devrez donc faire tester par le programme le nombre tapé par l'utilisateur avant de reprendre l'exécution. Ainsi les lignes suivantes le feront :

```
370 REM IS HUE OF BALL IN RANGE?
380 IF HUE > 0 AND HUE < 16 THEN GOTO 400
390 HOME : PRINT ''THAT WASN'T BETWEEN 1 AND 15!'':
    PRINT
395 GOTO 310
```

Si le caractère entré n'est pas un nombre,

```
]?REENTER
]?
```

apparaîtra sur l'écran. Vous apprendrez à **modifier le programme** pour éviter cela au Chapitre 5.

C'est une bonne pratique de la programmation qui permet de faire un programme aussi confortable que possible. Vous avez atteint le stade où vous avez à écrire des messages d'erreur qui seront lus par d'autres personnes. Il est parfaitement normal qu'un programmeur comme vous fasse son affaire du jargon du type ?SYNTAX ERROR, mais un utilisateur innocent ne doit pas le subir.

Chaque fois que vous utilisez une instruction INPUT, vous devez faire tester par le programme l'entrée de l'utilisateur de façon à éviter tout abandon. La conduite d'un utilisateur non préparé (et vous devez prévoir que les utilisateurs ne sont pas des programmeurs) est un art en soi. Il est toujours nécessaire d'utiliser des instructions INPUT conçues de façon spécifique et de tester soigneusement la réponse de l'utilisateur.

Maintenant que vous avez apporté quelques modifications à COLORBOUNCE, listez-le à nouveau. Vérifiez que vous avez ajouté la totalité des nouvelles lignes (280-395), que vous avez changé la ligne 680, et que toutes les lignes sont tapées correctement. Exécutez la nouvelle version. Puis sauvegardez-là sous le nom de NEW COLORBOUNCE.

Chaque fois que vous sauvegardez une version différente d'un programme, vous devez donner à la version un nouveau nom. Sinon la vieille version serait écrasée et perdue.

Pause

Production de bruits

Des clics, des tics, des tocs, ainsi que des bourdonnements variés sont facilement produits sur l'Apple *II*e. Vous pouvez produire des sons sur votre ordinateur si vous le frappez légèrement, si vous l'égratignez avec vos doigts ou si vous le laissez tomber ; vous pouvez également en produire en les programmant.

Pour construire un programme producteur-de-son sur l'Apple *II*e, vous avez besoin de cette formule :

```
NEW
150 SOUND = PEEK (-16336)
```

Il n'y a pas d'explication simple à cette formule. Le nombre -16336, fait référence à l'adresse mémoire du producteur de son de l'Apple *II*e et qui est fixée par les circuits électroniques de l'ordinateur.

La fonction PEEK fournit le contenu sous forme décimale de l'octet dont l'adresse mémoire est spécifiée par l'argument.

PEEK est une *fonction*. Dans Applesoft une fonction reçoit un ou plusieurs nombres, effectue certaines opérations sur eux et produit une valeur unique. Applesoft dispose d'un nombre défini de fonctions arithmétiques préétablies (comme SQR, qui détermine la racine carrée d'un nombre) ; certaines autres fonctions peuvent en être dérivées. Le *Manuel de Référence Applesoft* présente toutes les fonctions d'Applesoft.

Le nombre que la fonction utilise (– 16336 en ce cas) est appelé paramètre et il est toujours placé entre des parenthèses à la suite du nom de fonction. Le nombre que la fonction trouve est dit être renvoyé au programme. PEEK renvoie la valeur numérique du contenu de n'importe quel octet en mémoire. Vous fournissez l'adresse de cet octet quand vous placez le nombre entre les parenthèses ; en ce cas l'adresse est – 16336. L'Apple *IIe* a des adresses mémoire qui vont de 0 à 65535.

L'échelle des adresses mémoire est déterminée par la taille mémoire de la machine, L'Apple *IIe* a *64 Kilo octets* (64K) de mémoire, ce qui équivaut à 65535 octets car 1K représente 2 à la puissance 10 soit 1.024.

Pour la plupart des adresses PEEK renvoie seulement une valeur numérique, mais à certaines adresses, telles que – 16336, il peut déclencher quelque chose. En ce cas, il faut faire un clic au générateur de bruit de l'Apple *IIe*. Chaque fois que le programme exécute cette instruction, vous entendrez un clic à peine audible. Exécutez le programme et écoutez tout près votre ordinateur.

Maintenant ajoutez la ligne :

```
160 GOTO 150
```

et exécutez le programme. Il n'y a pas de problème pour entendre cela !

Pour que votre programme provoque un bourdonnement pendant une période limitée de temps, ajoutez ces instructions :

```
140 FOR BUZZ = 1 TO 100
160 NEXT BUZZ
```

Un ton est généré par une succession rapide de clics. Les programmes qui utilisent PEEK (– 16336) plus d'une fois généreront certaines sortes de bruits. Comme – 16336 est ennuyeux à taper, il y a une instruction qui vous permettra de substituer la variable S au nombre. Ajoutez ceci à votre programme de production de son :

```
100 S = – 16336
```

Pour produire un beau clic bien sonore, changez la ligne 150 en :

```
150 SOUND = PEEK(S) - PEEK(S) + PEEK(S) - PEEK(S) +
    PEEK(S) - PEEK(S)
```

Les variations dans le nombre de PEEK de l'instruction entraînent des variations dans les qualités du son. Faites quelques essais. Voyez ce qui se produit quand vous n'utilisez que des -PEEK(S)s, le changement du son si vous ne mettez que des +PEEK(S)s.

Pour obtenir un bourdonnement, placez une de vos associations dans une boucle. En général, plus la boucle tourne rapidement, plus la tonalité est haute.

Applesoft effectue les opérations arithmétiques à différentes vitesses : la soustraction est plus lente que l'addition et la division est la plus lente. Ainsi la boucle la plus rapide de PEEK est composée en totalité de +PEEK(S)s. Une boucle plus lente sera composée de -PEEK(S)s. La boucle la plus lente utilisera la division : PEEK(S)/PEEK(S)/PEEK(S) par exemple. On trouvera une information plus complète sur PEEK dans le *Manuel de Référence Applesoft.*

Pour obtenir des tonalités plus élevées sur Apple *II*e, essayez ces lignes

```
NEW
40  FOR PAUSE = 1 TO 2500 : NEXT PAUSE
50  S = PEEK(-16336) : GOTO 50
```

Souvenez-vous que Ctrl - C arrête l'exécution du programme. Comme vous pouvez le supposer, le ligne 40 provoque une pause avant l'exécution de la ligne 50. L'instruction GOTO de la ligne 50 entraîne un bruit continu après la pause.

Afin de faire bon usage de ces méthodes de production sonore, chargez votre plus récente version de COLORBOUNCE sur l'ordinateur. Listez-la et étudiez les lignes. Puis essayez d'ajouter un bruit chaque fois que la balle rebondit sur un mur.

Une solution possible est donnée dans le prochain paragraphe, mais essayez d'en réaliser une vous-même.

Note : Un rebondissement se produit chaque fois que XV et YV change de valeur (de signe) et de direction.

Bruit de la balle au rebond

Voici une façon de rendre audible le rebondissement. Ajoutez ces lignes à COLORBOUNCE :

```
240 REM SET S TO ADRESS OF SPEAKER
260 S = -16336
580 IF NX > 39 THEN NX = 39 : XV = -XV : FOR B = 1 TO 5 :
    BOUNCE = PEEK(S) + PEEK(S) + PEEK(S) : NEXT B
600 IF NX < 0 THEN NX = 0 : XV = -XV : FOR B = 1 TO 5 :
    BOUNCE = PEEK(S) + PEEK(S) + PEEK(S) : NEXT B
620 IF NY > 39 THEN NY = 39 : YV = -YV : FOR B = 1 TO 5 :
    BOUNCE = PEEK(S) + PEEK(S) + PEEK(S) : NEXT B
640 IF NY < 0 THEN NY = 0 : YV = -YV : FOR B = 1 TO 5 :
    BOUNCE = PEEK(S) + PEEK(S) + PEEK(S) : NEXT B
```

Vous pouvez voir l'avantage d'avoir plusieurs instructions sur une seule ligne dans un programme comme celui-ci. Les sons sont en liaisons directes avec les valeurs de XV et YV en sorte que l'on voit (avec une exécution plus efficace) l'addition à des lignes déjà existantes.

Maintenant essayez vos propres sons. Pourquoi ne pas produire un son différent pour le rebondissement sur chaque mur ? Quand vous aurez trouvé la combinaison de sons qui vous plaît, n'oubliez pas de sauvegarder une nouvelle version de COLORBOUNCE comprenant les lignes de fabrication de bruits. Le reste de ce chapitre sera consacré à l'introduction de nouveaux programmes et concepts ; nous reviendrons à COLORBOUNCE dans le Chapitre 5.

Pause

Nombres aléatoires

Voici une autre fonction Applesoft. Essayez-la (souvenez vous que vous pouvez l'arrêter quand vous voulez avec Ctrl - C) :

```
NEW
100 PRINT RND(1)
110 GOTO 100
RUN
```

La fonction numérique RND renvoie un nombre réel aléatoire compris entre 0 et 1.

RND est utilisé à la ligne 100 pour ''Random''. La fonction RND fournit des nombres aléatoires. Chaque fois que vous l'exécuterez, vous verrez un échantillon différent de nombres. Essayez le !

Les nombres produits par ce programme sont des nombres décimaux aléatoires compris entre zéro et un. N'importe quel argument (le nombre entre parenthèses) plus grand que zéro renverra des nombres décimaux aléatoires compris entre zéro et un. Essayez, par exemple, de changer la ligne 100 en

```
100 PRINT RND (6)
```

Bien que vous n'ayiez pas à le faire dans ce guide vous pouvez produire différents effets en utilisant un argument nul ou négatif. Voyez le *Manuel de Référence Applesoft* pour une information plus complète.

Les nombres décimaux aléatoires compris entre un et zéro peuvent être malcommodes. Souvent des entiers (des nombres comme trois, six, et dix) sont plus faciles à utiliser. Pour obtenir des entiers aléatoires entre zéro et neuf, vous n'avez qu'à modifier le programme. Tapez

```
NEW
 90 REM ASSIGNS RND NUMBER TO X
100 X = RND(1)
110 REM MULTIPLIES X BY 10
120 X = X * 10
130 REM CHOPS OFF THE FRACTION
140 X = INT(X)
150 PRINT X
160 GOTO 100
```

La fonction INT renvoie le plus grand entier inférieur ou égal à l'argument donné.

La ligne 140 introduit INT, la fonction INTEGER. La fonction X = INT(X) renvoie l'entier le plus grand inférieur ou égal à X. Par exemple, si la valeur de X est 3.6574, INT(X) est égal à 3 ; si la valeur de X est −45.12345, INT(X) est égal à −46. Les parenthèses après INT peuvent contenir une expression numérique ou une variable numérique.

Maintenant exécutez le programme. Fait-il ce que vous attendez ?

Pour modifier le programme de façon à lui faire générer des nombres de un à dix (au lieu de zéro à neuf) ajoutez cette ligne à votre programme :

```
145 X = X + 1
```

Ce programme peut sembler un peu compliqué au premier abord. Pour voir ce qui se passe pas à pas, ajoutez des instructions PRINT. Le programme suivant indique comment. Cependant vous n'avez pas besoin de les taper. Le programme RANDOM du disque APPLESOFT SAMPLER est le même que celui qui est présenté.

```
 90 REM ASSIGNS RND NUMBER TO X
100 X = RND(1) : PRINT ''X = RND(1)'', X
105 PRINT
110 REM MULTIPLIES X BY 10
120 X = X * 10 : PRINT ''X = X * 10'',X
125 PRINT
130 REM CHOPS OFF THE FRACTION
140 X = INT(X) : PRINT ''X = INT(X)'', X
145 PRINT
150 REM ADDS 1 TO THE VALUE OF X
160 X = X + 1 : PRINT ''X = X + 1'', X
170 PRINT : PRINT
180 FOR PAUSE = 1 TO 2000 : NEXT PAUSE
190 GOTO 100
```

Chargez et exécutez ce programme pour voir ce qui se produit. Pour modifier la façon dont les nombres sont affichés, vous pouvez enlever la virgule (,) et le X de la fin de la ligne 160, changez la ligne 170 et ajoutez la ligne 175 comme ceci :

```
160 X = X + 1 : PRINT ''X = X + 1''
170 PRINT X
175 PRINT
```

De façon assez étonnante, ce programme entier peut être condensé en seulement une ligne :

```
100 PRINT INT (10 * RND(1)+1 : GOTO 100
```

Etudiez cette ligne pour voir comment ces différents éléments correspondent chacun à des lignes de RANDOM.

Simulation d'une paire de dés

Vous pouvez utiliser maintenant ce que vous avez appris sur les nombres aléatoires pour faire un programme représentant une paire de dés :

```
NEW
100  PRINT ''WHITE DICE'',
110  PRINT INT(6 * RND(1) ) + 1
120  PRINT ''RED DICE''
130  PRINT INT(6 * RND(1) ) + 1
```

Ce programme produit des entiers aléatoires compris entre un et six pour chaque dé. Chaque fois que vous exécutez le programme, vous simulez un coup de dés. Que pouvez vous ajouter au programme pour ne pas avoir à l'exécuter sans cesse ? Pouvez vous écrire un programme qui utilise ces ''dés'' pour réaliser un jeu ? Essayez.

Maintenant essayez d'écrire un programme d'une ligne qui produit des nombres aléatoires compris entre 1 et 50, puis entre 0 et 25. Définissez vos propres limites. N'oubliez pas d'ajouter un (1) au nombre aléatoire si vous ne voulez pas générer de zéros.

Conseil utile : Le nombre multipliant RND (comme dans 50 * RND(1)) représente l'échelle des nombres produits. Le nombre ajouté à RND est le plus petit nombre produit : si 1 est ajouté, le plus petit nombre sera 1 ; si 3 est ajouté, le plus petit nombre sera 3 ; si rien n'est ajouté ; le plus petit nombre sera 0.

Graphiques aléatoires

Voici une association multicolore de la fonction RND et de quelques instructions graphiques :

```
NEW
200  GR
210  REM CHOOSE A RANDOM COLOR
220  COLOR = INT (16 * RND(1) )
230  REM CHOOSE A RANDOM POINT
240  X = INT (40 * RND(1) )
250  Y = INT (40 * RND(1) )
260  REM PLOT THE RANDOM POINT
270  PLOT X, Y
280  REM DO IT AGAIN
290  GOTO 220
```

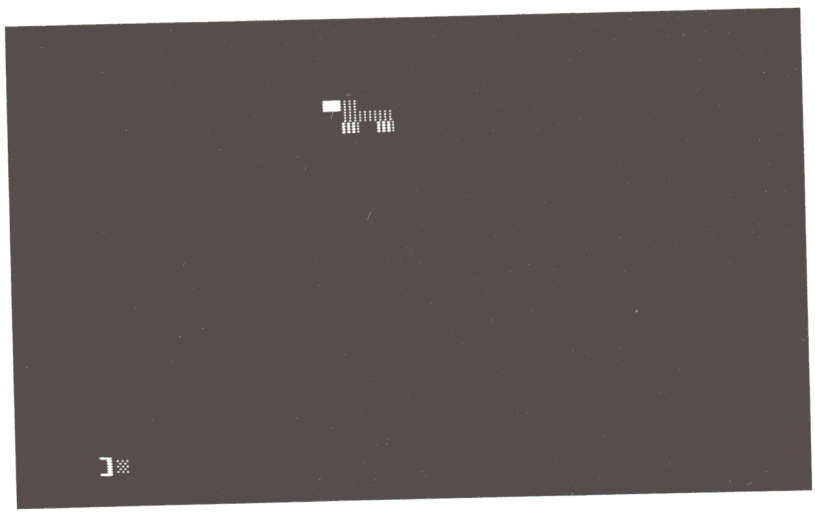
Essayez d'utiliser RND dans d'autres programmes. Pouvez-vous écrire un programme qui trace des lignes aléatoires dans des couleurs aléatoires sur un écran ? Comment un programme peut-il ombrer un écran graphique avec des couleurs aléatoires ?

La fonction RND est utilisée dans beaucoup de jeux. Un exemple d'utilisation un peu plus longue se trouve dans SCRAMBLER de l'APPLESOFT SAMPLER. Ce programme est représenté dans l'Annexe E, ''Autres Programmes de Jeu''.

Les sous-programmes : assemblage des éléments

Ce paragraphe suit pour un programme la démarche de la pensée pas à pas et assemble les éléments. Commençons par une idée : vous voulez dessiner un cheval qui sera un élément de jeu que vous allez construire. Il y a un programme qui dessine un cheval bleu avec des membres oranges et une tête blanche :

```
NEW
1000  REM DRAW BLUE HORSE WITH WHITE FACE AND ORANGE
      FEET
1010  GR
1020  COLOR = 2 : REM DARK BLUE
1030  PLOT 15,15
1040  HLIN 15,17 AT 16
1050  COLOR = 9 : REM ORANGE
1060  PLOT 15,17
1070  PLOT 17,17
1080  COLOR = 15 : REM WHITE
1090  PLOT 14,15
```



4-1. Un Cheval bleu

Il n'y a rien de mauvais dans ce programme : il dessine un cheval bleu avec des membres oranges et une tête blanche. Mais supposons que vous vouliez dessiner un autre cheval quelque part ailleurs sur l'écran. Vous pouvez réécrire le programme en donnant de nouvelles valeurs à X et Y, mais, c'est ennuyeux. Il devrait y avoir une méthode pour utiliser le même programme pour placer n'importe où sur l'écran une figure sans avoir à la réécrire à chaque fois.

Vous avez appris avec COLORBOUNCE que des variables pouvaient être utilisées pour changer la vitesse et la direction du déplacement des points tracés sur l'écran graphique. Une méthode analogue peut être utilisée pour changer le cheval de place sur l'écran.

Vous pouvez déplacer un point ayant pour coordonnées (A,B) vers la droite en ajoutant quelque chose à la première coordonnée, A. Si A = 4 et B = 17, vous pouvez déplacer le point (A,B) de 10 colonnes à droite en ajoutant 10 à la première coordonnée, le transformant en un point (14,17).

De même, un point se déplace à gauche et si vous retranchez quelque chose de la première coordonnée (ou lui ajoutez une valeur négative). Un simple essai montre qu'en ajoutant ou retranchant quelque chose de la seconde coordonnée, on déplace le point vers le bas et vers le haut, respectivement.

C'est la base essentielle de tout programme d'animation.

A partir de là, vous pouvez réécrire le programme du cheval pour positionner le cheval presque n'importe où sur l'écran. Pourquoi presque n'importe où ? Parce que si le centre du cheval est tracé au bord de l'écran, une partie du cheval sortira de l'écran. Ceci pourrait vous amener un message d'erreur ?ILLEGAL QUANTITY ERROR IN 1030. Le numéro se trouvant à la fin du message d'erreur identifie la ligne sur laquelle se trouve l'erreur : ceci facilite la correction par le programmeur. Voici un élément du programme amélioré :

```
NEW
1000  REM PUT A HORSE ANYWHERE ON THE SCREEN
1010  COLOR = 2 : REM DARK BLUE BODY
1020  PLOT X, Y - 1 : REM CENTER OF HORSE
1030  HLIN X, X + 2 AT Y : REM REST OF BODY
1040  COLOR = 9 : REM ORANGE FEET
1050  PLOT X, Y + 1 : REM FRONT FOOT
1060  PLOT X + 2, Y + 1 : REM REAR FOOT
1070  COLOR = 15 : REM WHITE HEAD
1080  PLOT X - 1, Y - 1
```

Notez que l'instruction GR a été abandonnée. Cet élément de programme est supposé placer sur l'écran plusieurs chevaux. Une instruction GR en 1005 viderait l'écran avant que chaque nouveau cheval soit dessiné.

Ce programme ne peut pas être exécuté sous cette forme. Vous devez initialiser le mode « graphique » et choisir X et Y :

```
20 GR
30 REM FIRST HORSE CENTER
40 X = 12
50 Y = 35
```

Quand vous exécutez ceci, vous obtenez un cheval à l'emplacement désiré avant la fin du programme. Mais vous voulez encore placer deux chevaux sur l'écran. Et si vous pouviez écrire ce que vous lisez Figure 4-2 :

Figure 4-2. Un programme « Et si ».

```
60 DO THE PORTION OF THE PROGRAM AT LINE 1000
   AND THEN COME BACK TO LINE 70
70 REM SECOND HORSE CENTER
80 X = 33
90 Y = 2
100 DO THE PORTION OF THE PROGRAM AT LINE 1000
    AGAIN AND THEN END
```

Ne serait-ce pas beau et facile ? Le problème est que l'ordinateur ne peut pas lire les instructions étranges des lignes 60 et 100. Il peut cependant lire l'instruction

```
GOSUB 1000
```

GOSUB provoque un branchement du programme au numéro de ligne indiqué. Le sous-programme commence à ce numéro de ligne et se termine à une instruction RETURN, qui entraîne en retour le branchement du programme à l'instruction suivant immédiatement GOSUB.

Un programme tel que celui qui commence à la ligne 1000 est appelé un *sous-programme*. GOSUB 1000 dit à l'ordinateur d'aller au sous-programme commençant à la ligne 1000 et cette instruction est exécutée. Il dit aussi à l'ordinateur de revenir à la ligne qui suit l'instruction GOSUB quand le sous-programme sera terminé. L'ordinateur sait que le sous-programme est terminé quand il rencontre l'instruction RETURN. Pour faire un sous-programme complet dans votre programme de tracé de chevaux, ajoutez la ligne

```
1090 RETURN
```

La partie du programme allant de la ligne 1000 à la ligne 1090 est appelée un sous-programme. Un sous-programme est un élément de programme qui est utilisé maintes et maintes fois par le programme principal (lignes 20 à 100).

En associant toutes ces lignes, vous avez le programme « et si vous pouviez seulement » :

```
  20 GR
  30 REM CHOOSE CENTER OF THE FIRST HORSE
  40 X = 12
  50 Y = 35
  60 GOSUB 1000
  70 REM CHOOSE CENTER OF THE SECOND HORSE
  80 X = 33
  90 Y = 2
 100 GOSUB 1000
1000 REM PUT A HORSE ANYWHERE ON THE SCREEN
1010 COLOR = 2 : REM DARK BLUE BODY
1020 PLOT X, Y - 1 : REM CENTER OF HORSE
1030 HLIN X, X + 2 AT Y : REM REST OF BODY
1040 COLOR = 9 : REM ORANGE FEET
1050 PLOT X, Y + 1 : REM FRONT FOOT
1060 PLOT X + 2, Y + 1 : REM REAR FOOT
1070 COLOR = 15 : REM WHITE HEAD
1080 PLOT X - 1, Y - 1
1090 RETURN
```

Si vous exécutez ce programme, vous obtiendrez le message d'erreur suivant :

```
?RETURN WITHOUT GOSUB ERROR IN 1090
```

Vous découvrirez bientôt comment arrêter le programme pour éviter cette erreur. Retenez la leçon ! Cependant, en dépit du message d'erreur, le programme tourne correctement. Vous avez bien créé un *programme* pour dessiner un cheval. Maintenant utilisez l'instruction

```
GOSUB 1000
```

pour dessiner un de ces chevaux spéciaux à un quelconque emplacement (X,Y) que vous aurez choisi.

Trace

L'instruction de mise au point de programme TRACE donne le chemin suivi par le programme pensant son exécution.

Pour suivre le déroulement du programme, ou le chemin suivi dans l'exécution, vous pouvez invoquer une instruction appelée TRACE. Cette instruction vous montrera pourquoi l'Apple *II*e a donné un message d'erreur quand le programme de tracé de chevaux a été exécuté. Ajoutez cette ligne :

```
10 TRACE
```

et pour le moment, effacez la ligne 20 avec

```
DEL 20,20
```

Pour quitter le mode graphique et vider l'écran, tapez TEXT et HOME. **Puis exécutez le programme.**

Figure 4-3. Utilisation de TRACE. TRACE affiche le numéro de ligne de chaque instruction à son exécution.

```
#10 #30 #40 #50 #60 #1000 #1010 #1020 #1
030 #1040 #1050 #1060 #1070 #1080 #1090
#70 #80 #90 #100 #1000 #1010 #1020 #1030
#1040 #1050 #1060 #1070 #1080 #1090 #10
00 #1010 #1020 #1030 #1040 #1050 #1060 #
1070 #1080 #1090
?RETURN WITHOUT GOSUB ERROR IN 1090
```

TRACE montre que le programme commence à la ligne 10, le plus petit numéro de ligne, et continue jusquà atteindre la ligne 60, qui l'envoie au sous-programme. Il exécute le sous-programme, revient au programme principal à la ligne 1090, puis exécute le sous-programme à nouveau. Quand il atteint à nouveau la ligne 1090, ne trouvant pas de numéro de ligne plus petit, il va à la ligne 1000 et exécute le sous-programme à nouveau. C'est là que se produit un problème. Comprenez-vous le message d'erreur maintenant ?

Pour remédier au problème de répétition sans marque de fin, ajoutez cette nouvelle ligne au programme :

L'instruction END entraîne l'arrêt de l'exécution du programme et donne la main à l'utilisateur.

```
110 END
```

Quand le programme atteint la ligne 110, il ne fera que ce que la ligne dit : end. Exécutez le programme une fois encore. Aucun message d'erreur cette fois-ci. Comme l'instruction RETURN provoque le branchement du programme à l'instruction suivant immédiatement le dernier GOSUB exécuté, la ligne 110 est atteinte et le programme s'arrête.

Comme vous voyez, TRACE est extrêmement commode quand vous avez des problèmes avec un programme. En ajoutant TRACE, vous pouvez savoir où se trouve le problème.

L'instruction NOTRACE met TRACE hors service.

Si vous voulez ne traiter par TRACE qu'une partie du programme, utilisez l'instruction NOTRACE. Ajoutez cette ligne :

```
65 NOTRACE
```

et la trace du programme ne sera suivie que jusqu'à la ligne 65.

TRACE peut aussi être lancée quand vous travaillez sans numéro de ligne en exécution immédiate avec TRACE suivi de RUN. Ce n'est pas une bonne idée de l'essayer maintenant, à moins de sauvegarder la version courante du premier programme de tracé de chevaux.

Une fois que vous aurez lancé l'instruction TRACE, que l'exécution soit immédiate ou différée, la trace du programme sera effectuée chaque fois que vous l'exécuterez. Pour arrêter TRACE vous devez exécuter une instruction NOTRACE et enlever l'instruction TRACE.

Maintenant que vous avez terminé le suivi de la trace du programme de tracé de chevaux, exécutez l'instruction NOTRACE, enlevez les lignes 10 et 65 et replacez la ligne 20. De cette façon vous pourrez ultérieurement exécuter le programme sans avoir à exécuter sa trace à chaque fois.

Un meilleur programme de tracé de chevaux

Ce paragraphe introduit plusieurs compléments et modifications au programme de tracé de chevaux. Les modifications sont basées sur des techniques qui seront utiles pour tous vos programmes futurs.

Les erreurs

Il est important de prévoir les erreurs possibles quand on écrit des programmes afin d'éviter des problèmes. Un des problèmes du sous-programme de tracé de cheval est le suivant : certaines valeurs de X et Y placeront le cheval en dehors des limites de l'écran. On peut y remédier en ajoutant un ensemble d'instructions du type suivant :

```
1000 REM PUT A HORSE ANYWHERE WITHIN SCREEN
1012 IF X < 1 THEN X = 1
1014 IF X > 37 THEN X = 37
1016 IF Y < 1 THEN Y = 1
1018 IF Y > 38 THEN Y = 38
```

Le format de ces lignes devrait vous être familier : il est semblable à celui utilisé dans COLORBOUNCE. Pour garder les chevaux à l'intérieur des limites de l'écran, les valeurs de X et Y sont comparées aux valeurs de la grille graphique. (Pourquoi la valeur maximum de Y devrait-elle être de 38, alors que X doit être limitée à 37 ?).

S'il y a une tentative pour placer un cheval hors de l'écran, le cheval sera déplacé vers le bord le plus proche. Il y a d'autres stratégies de programmation, comme de délivrer un message d'erreur et d'arrêter le programme. Cependant la limitation par le créneau a l'avantage de ne pas arrêter le programme.

Notez que la ligne 1000 est légèrement différente de l'ancienne ligne 1000 ; le reste des lignes est à ajouter au sous-programme.

Les Variables

Comme vous vous en êtes déjà rendu compte, l'utilisation de variables, au lieu des affectations de numéros spécifiques aux couleurs, rend un programme plus souple. Si, par exemple, vous désirez ajouter un second joueur au jeu et donner à ce joueur un cheval d'une couleur différente, vous pourrez remplacer la ligne 1010 COLOR = 2 par

```
1010 COLOR = BODY
```

De même, vous pourrez écrire :

```
1040 COLOR = FEET
1070 COLOR = FACE
```

Alors le programme principal deviendrait :

```
 20  GR
 30  REM FIRST PLAYER'S HORSE COLOR
 40  BODY = 2 : REM DARK BLUE
 50  FEET = 9 : REM ORANGE
 60  FACE = 15 : REM WHITE
 70  REM FIRST PLAYER'S HORSE CENTER
 80  X = 15
 90  Y = 30
100  GOSUB 1000
```

et ainsi de suite. Maintenant vous ajoutez les lignes pour la couleur du cheval du second joueur. N'oubliez pas d'inclure une instruction END. Exécutez le programme pour voir les deux chevaux s'afficher. Changez les lignes de votre programme jusqu'à obtenir les associations de couleurs que vous aimez.

C'est une bonne occasion d'utiliser quelques unes des finesses de l'éditeur que vous avez appris dans le Chapitre 3, « Modification de programmes ». Un des avantages de l'utilisation du mode escape est que vous pouvez lister une ligne qui apparaît sous une forme semblable plus d'une fois dans le programme, faire les modifications prévues, et la copier en vue d'une reproduction facile. Essayez le maintenant — cela vous économisera beaucoup de frappe.

Sous-programmes complémentaires

Pour affiner encore davantage le programme de tracé de chevaux, utilisez des sous-programmes qui affectent les couleurs pour le cheval de chaque joueur et qui, à leur tour, iront chacun au sous-programme de tracé de cheval (ou l'appelleront). Voici deux exemples :

```
2000  REM DRAWS BLUE HORSE WITH ORANGE FEET AND WHITE
      FACE
2010  BODY = 2 : REM DARK BLUE
2020  FEET = 9 : REM ORANGE
2030  FACE = 15 : REM WHITE
2040  GOSUB 1000
2050  RETURN

2500  REM DRAWS ORANGE HORSE WITH PINK FEET AND GREEN
      FACE
2510  BODY = 9 : REM ORANGE
2520  FEET = 11 : REM PINK
2530  FACE = 4 : REM GREEN
2540  GOSUB 1000
2550  RETURN
```

Quand vous ajoutez les sous-programmes 2000 et 2500 vous devez aussi changer le programme principal. Pour placer un cheval bleu avec une tête blanche et des pattes orange en (10,11), changez comme suit les lignes 30 à 60 :

```
30  REM FIRST PLAYER'S HORSE
40  X = 10
50  Y = 11
60  GOSUB 2000
```

Pour placer un cheval orange en (19,2), remplacez les lignes 70-100 par :

```
70  REM SECOND PLAYER'S HORSE
80  X = 19
90  Y = 2
100 GOSUB 2500
```

Les sous-programmes du tracé de chevaux des lignes 2000 et 2500 appellent un autre sous-programme qui commence à la ligne 1000. Les choses deviennent tout à fait efficaces à ce stade. Les trois sous-programmes rendent très facile la mise en place d'un attrayant affichage de chevaux.

Mais d'abord, un autre sous-programme commode qui remplace les anciennes lignes 1012-1018 :

```
3000 REM CHOOSES A RANDOM PAIR OF COORDINATES
3010 X = INT (RND(1) * 37)+ 1
3020 Y = INT (RND(1) * 38) + 1
3030 RETURN
```

Un programme bien structuré

Voici le programme entier. Il associe le programme principal aux trois sous-programmes. Vous pouvez modifier le programme que vous avez tapé pour le mettre en accord avec ce que vous voyez ou chargez HORSES à partir de l'APPLESOFT SAMPLER.

```
 10  REM SET GRAPHICS MODE
 20  GR
 30  REM CHOOSE A RANDOM POINT
 40  GOSUB 3000
 50  REM PUT A BLUE HORSE THERE
 60  GOSUB 2000
 70  REM CHOOSE ANOTHER RANDOM POINT
 80  GOSUB 3000
 90  REM PUT AN ORANGE HORSE THERE
100  GOSUB 2500
110  REM DO IT ALL AGAIN
120  GOTO 30
```

Voici comment se présente un programme principal si vous écrivez des programmes bien structurés : principalement des instructions REM et GOSUB. Le travail doit être fait dans des sous-programmes relativement courts, chacun étant facile à écrire et un tout en soi.

L'Annexe E donne plus d'exemples de sous-programmes et indique comment les structurer pour avoir le maximum d'efficacité dans le programme.

Si les tâches de ce programme ne sont pas encore tout à fait claires, utilisez TRACE pour suivre le déroulement du programme. Tapez TEXT et HOME. Puis ajoutez

```
  5  TRACE
130  NOTRACE
```

ensuite, RUN. Comparez ce que vous voyez sur l'écran avec le listing du programme jusqu'à ce que vous puissiez suivre les sous-programmes.

Pause

Graphique haute résolution

Jusqu'ici vous avez utilisé des graphiques basse résolution. Dans ce paragraphe sera introduit une autre espèce de graphique appelé *graphique haute résolution.*

Le graphique haute résolution vous permet de dessiner avec beaucoup plus de détails que ce que l'on peut avoir avec la grille basse résolution. L'écran du graphique haute résolution a une définition de 280 sur 160 points pour le tracé. La coordonnée horizontale part de 0 à gauche de l'écran et va jusqu'à 279 à droite. De même la coordonnée verticale va de 0 au sommet de l'écran jusqu'à 159 en bas.

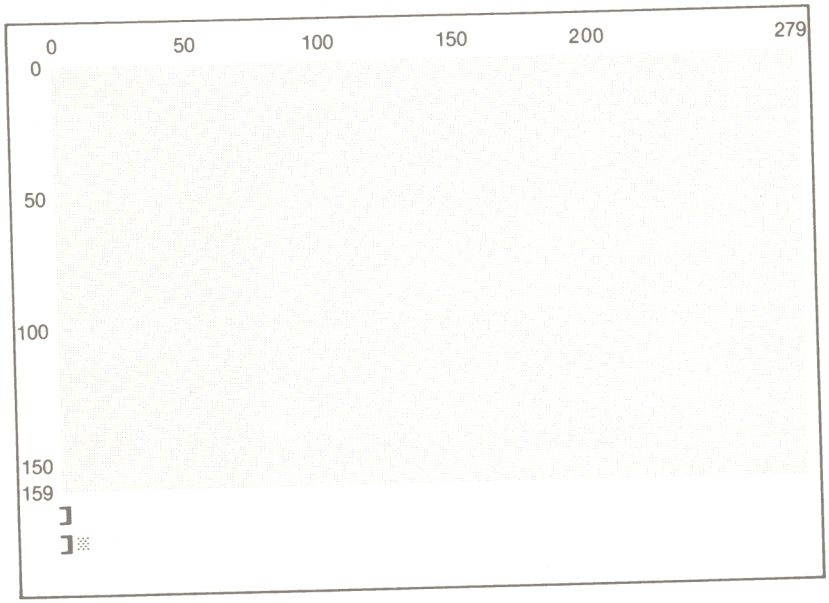


Figure 4-4. Coordonnées d'un écran Graphique haute résolution. L'échelle des coordonnées horizontales va de 0 à 279 ; l'échelle des coordonnées verticales va de 0 à 159.

Les commandes du graphique haute résolution sont souvent les mêmes que celles correspondantes du graphique basse résolution à part l'addition d'un H (pour « high » résolution). Votre familiarisation avec le graphique basse résolution vous aidera dans ce paragraphe.

Tapez

```
HGR
```

Le graphique haute résolution, initialisé par HGR utilise une grille écran avec une définition de 280 sur 168 points, et laisse quatre lignes pour le texte en bas de l'écran. HGR vide l'écran et le met en noir.

pour obtenir le graphique haute résolution. Cette instruction vide l'écran, le met en noir et laisse quatre lignes en bas pour le texte. Comme le graphique basse résolution, le graphique haute résolution vous permet d'utiliser des coordonnées verticales que vous mettrez dans la zone texte (191 est le maximum) mais ces points ne sont pas apparents sur l'écran à moins que vous fassiez certaines manœuvres subtiles. Voyez le *Manuel de Référence Applesoft* pour plus d'information.

Conseil Utile : Si le curseur n'est pas visible, tapez ↵ jusqu'à ce que le curseur apparaisse près du bas de l'écran.

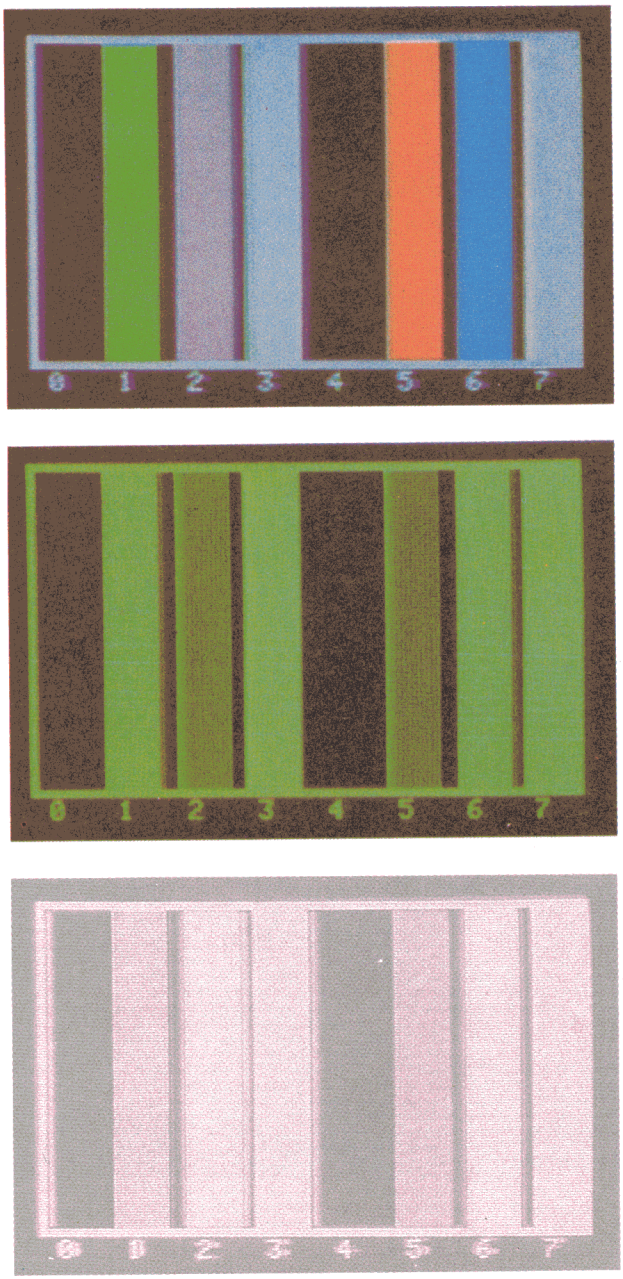
HCOLOR= place la couleur du graphique haute résolution au numéro spécifié.

La couleur des points que vous tracez avec le graphique haute résolution est déterminée par l'instruction HCOLOR=. Comme pour le graphique basse résolution, l'Apple *II*e utilisera la couleur assignée jusqu'à ce que vous en changiez.

Le graphique haute résolution est vraiment étonnant, mais vous devrez faire quelques sacrifices pour l'utiliser. Il y a peu de numéros attribués à HCOLOR= et les couleurs varient en fonction de leurs positions sur l'écran. Voyez la figure 4-5 pour savoir quel numéro va avec quelle couleur.

Figure 4-5. Graphique haute résolution

0	noir1	4	noir2
1	vert	5	orange
2	violet	6	bleu
3	blanc1	7	blanc2



Pour essayer le graphique haute résolution une fois que vous avez lancé l'instruction HGR tapez

```
HCOLOR = 2
HPLOT 130,100
HPLOT 50,50
```

et ainsi de suite. Remarquez comme les points tracés en haute résolution apparaissent beaucoup plus petits. Ce mode rend possible la création d'images avec beaucoup plus de détails que ce qu'on pouvait obtenir en basse résolution.

Figure 4-6. La bonne résolution de l'écran en mode HGR.

L'instruction HPLOT trace des points et des lignes en graphique haute résolution en utilisant la valeur la plus récemment spécifiée de HCOLOR

Le tracé des lignes est même plus facile avec le graphique haute résolution qu'avec le graphique basse résolution. Vous utilisez simplement HPLOT à partir d'un point de l'écran jusqu'à un autre point. Pour tracer une ligne le long du bord haut de l'écran, tapez

```
HCOLOR = 1
HPLOT 0,0 TO 279,0
```

Si vous désirez tracer une ligne allant du coin de coordonnée 279,0 au coin bas de l'écran, tout ce que vous avez à faire est de taper

```
HPLOT TO 279,159
```

et une ligne apparaît le long du bord droit de l'écran. Quand vous utilisez cette dernière instruction, la nouvelle ligne prend son point de départ du dernier point tracé et utilise la couleur de ce point (même si vous avez lancé une nouvelle commande HCOLOR= depuis que ce point a été tracé). Pour le voir vous même tapez

```
HCOLOR = 4
HPLOT TO 0,159
```

La couleur 4 est le noir ; ainsi vous pourriez penser que la ligne n'apparaîtra pas. En fait elle apparaît car elle prend la couleur du point de départ qui est le vert (numéro 1).

Vous pouvez aussi associer plusieurs lignes dans une instruction HPLOT. Videz l'écran avec HGR et essayez ceci sur l'Apple *II*e :

```
HCOLOR = 3
HPLOT 0,0 TO 279,0 TO 279,159 TO 0,159 TO 0,0
```

Il y aura une ligne bordant l'écran tout autour comme le montre la figure 4-7. S'il n'y a pas une ligne continue autour de l'écran, vérifiez ce que vous avez tapé.

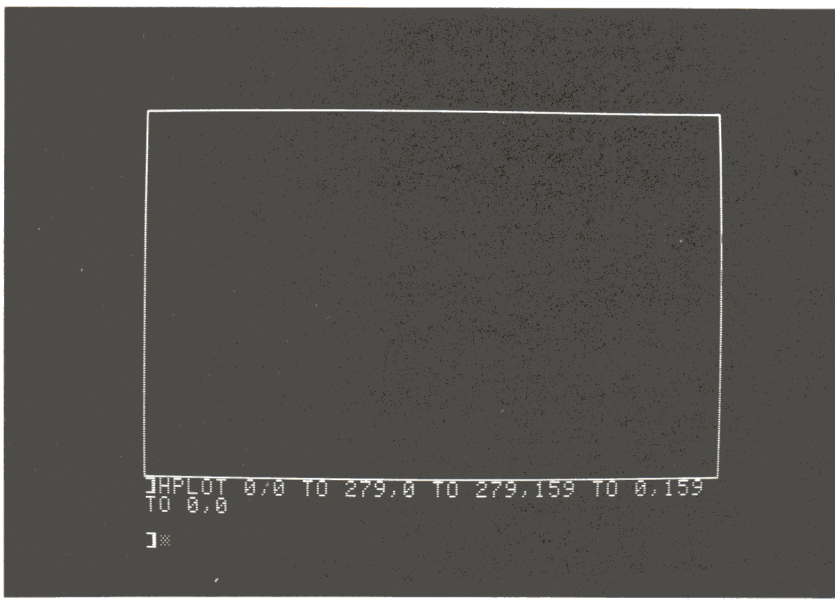


Figure 4-7. Une ligne tout autour et au bord de l'écran haute résolution.

Si la ligne du haut n'est pas visible sur votre écran, votre télévision ou votre moniteur vidéo ont besoin d'un réglage. Essayez de le régler. Si cela ne va pas, remplacez HPLOT 0,0 TO 279,0 par HPLOT 0,2 TO 279,2 et ainsi de suite.

Vous voyez comme il est facile de tracer des lignes droites en haute résolution. Eh bien, les lignes diagonales sont tout aussi faciles. Pour tracer une ligne depuis le coin haut et gauche de l'écran jusqu'au coin bas et droite, tapez :

```
HPLOT 0,0 TO 279,159
```

Tracez des lignes en haute résolution de longueurs et couleurs variées. Vous découvrirez que certaines couleurs ne donnent pas toujours des lignes verticales sur les moniteurs en noir et blanc. Les couleurs 2 et 6 seules tracent des lignes verticales si on commence par des nombres pairs, tandis que 1 et 5 seulement tracent des lignes en commençant par des nombres impairs. Par exemple si HCOLOR = 1, HPLOT 279,0 TO 279,159 trace une ligne verticale, HPLOT 2,159 TO 0,0 ne le fera pas. Ceci est dû au mode d'interprétation des couleurs en mode haute résolution par les moniteurs en noir et blanc.

Pour voir un exemple de ce que vous pouvez faire avec le graphique haute résolution, chargez le programme MOIRE d'APPLESOFT SAMPLER. Exécutez-le ; arrêtez le programme avec Ctrl - C. Puis tapez TEXT, HOME, et LIST pour voir les lignes du programme.

Figure 4-8. MOIRE. Listing du programme. Les numéros 1 et 2 correspondent à ceux du texte.

```
NEW
 80 REM MOIRE PROGRAM
 90 HOME
100 VTAB 24 : REM MOVE CURSOR TO BOTTOM LINE
120 HGR : REM SET HIGH-RESOLUTION GRAPHICS
140 A = RND(1) * 279 : REM PICK AN X FOR
    ''CENTER''
160 B = RND(1) * 159 : REM PICK A Y FOR ''CENTER''
180 N = INT (RND(1) * 4) + 2 : REM PICK A
    STEP SIZE
200 HTAB 15 : PRINT ''STEPPING BY''; N;
220 FOR X = 0 TO 278 STEP N : REM STEP THRU A VALUES
240 FOR S = 0 TO 1 : REM 2 LINES, FROM X AND
    X + 1
260 HCOLOR = 7 * S : REM FIRST LINE BLACK, NEXT
    WHITE
280 REM DRAW LINE THROUGH ''CENTER'' TO OPPOSITE SIDE
300 HPLOT X + S,0 TO A,B TO 279 - X - S,159
320 NEXT S,X
340 FOR Y = 0 TO 158 STEP N : REM STEP THRU B
    VALUES
360 FOR S = 0 TO 1 : REM 2 LINES, FROM Y AND
    Y + 1
380 HCOLOR = 7 * S : REM FIRST LINE BLACK, NEXT
    WHITE
400 REM DRAW LINE THROUGH ''CENTER'' TO OPPOSITE SIDE
420 HPLOT 279, Y + S TO A,B TO 0,159 - Y - S
440 NEXT S,Y
460 FOR PAUSE = 1 TO 1500 : NEXT PAUSE : REM DELAY
480 GOTO 120 : REM DRAW A NEW PATTERN
```

1 — 280 REM DRAW LINE THROUGH ''CENTER'' TO OPPOSITE SIDE

2 — 320 NEXT S,X

1 — 400 REM DRAW LINE THROUGH ''CENTER'' TO OPPOSITE SIDE

2 — 440 NEXT S,Y

1. Le centre de l'écran haute résolution est à l'intersection de quatre points traçables, et il ne peut pas être visualisé avec précision. Ainsi « CENTER » des lignes 280 et 400 est approché.

2. Une instruction NEXT peut être relative à plus d'une instruction FOR, comme vous le voyez dans les lignes 320 et 440. Veuillez bien placer les variables du NEXT dans le bon ordre pour éviter de croiser les boucles.

Pouvez vous réfléchir aux façons de modifier le programme ? Par exemple, essayez de changer la valeur de HCOLOR= de façon aléatoire. Essayez de tracer des lignes oranges puis bleues ou seulement des lignes bleues.

Le mode haute résolution contient bien d'autres possibilités que celles présentées ici. Quand vous vous sentirez à l'aise pour utiliser les instructions graphiques haute résolution présentées dans ce paragraphe, référez vous au *Manuel de Référence Applesoft* pour avoir plus d'information sur le graphique haute résolution.

Résumé du chapitre

Instructions

PEEK
RND
INT
GOSUB
TRACE
NOTRACE
END
HGR
HCOLOR=
HPLOT

Termes

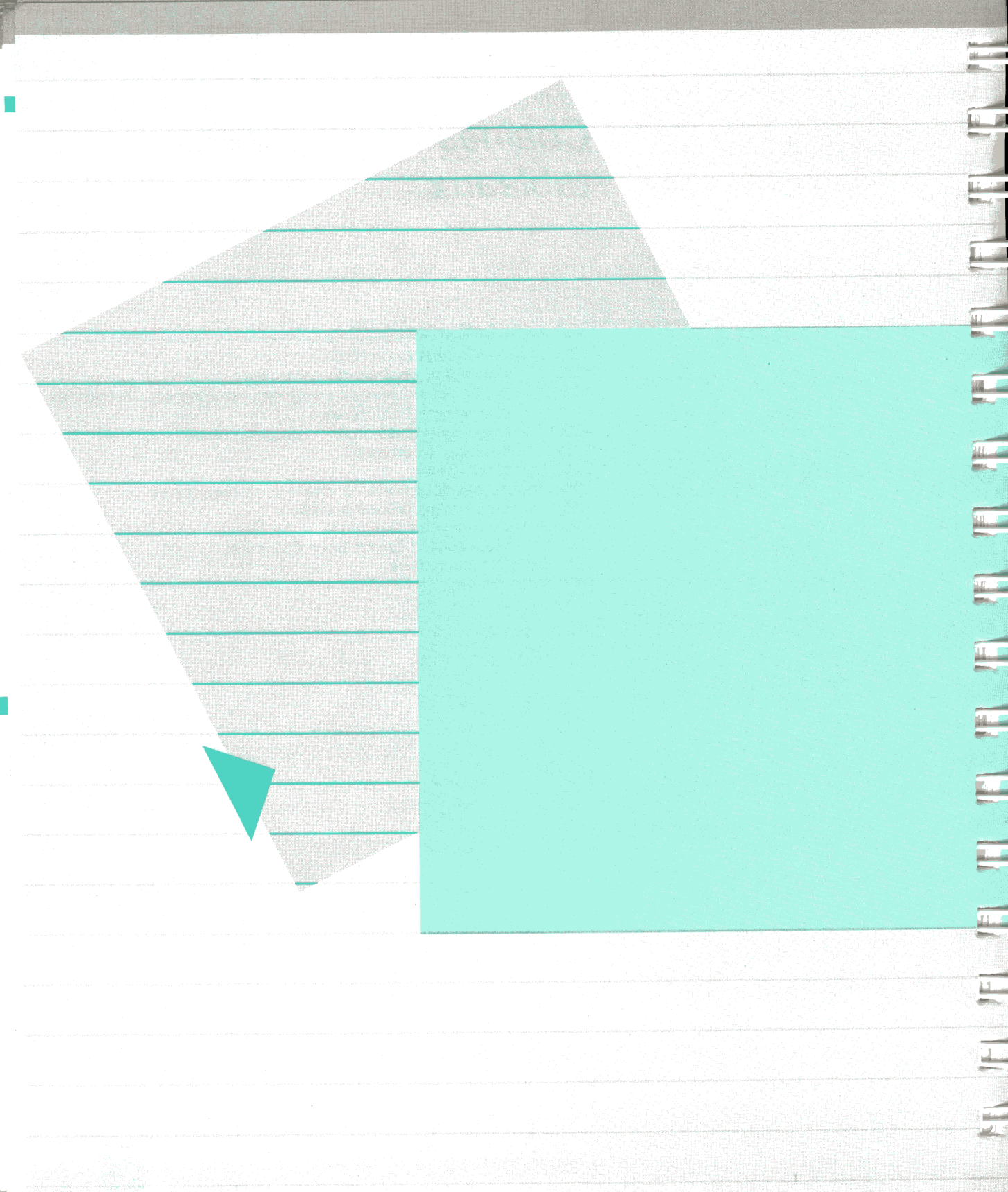
adresse
fonction
octet
kilo octet
mémoire
sous-programme
programme
graphique haute résolution

Messages d'erreur

?REENTER
?RETURN WITHOUT GOSUB
ERROR

Chaînes de caractères et tableaux

123 Enchaînons les caractères
124 Fonctions chaînes de caractères
126 Pratiques courantes en programmation en utilisant les chaînes de caractères
128 Duplications de chaînes de caractères
128 Epelons à l'envers
130 Concaténation
131 Encore des fonctions de chaînes de caractères
134 Encore des pièges à erreurs
135 Introduction aux tableaux
139 Messages d'Erreur pour Tableaux
140 Chapitre sommaire
141 Conclusion



Chaînes de caractères et tableaux

Les ordinateurs peuvent manipuler des lettres et des symboles aussi bien que des graphiques et des nombres. Dans ce chapitre vous apprendrez comment l'Apple *II*e manipule des *chaînes* entières de caractères. Vous en saurez également plus sur la façon dont il stocke des caractères et des nombres dans ce qui est appelé des *tableaux*.

Enchaînons les caractères

Une chaîne de caractères, ou suite de caractères, est une variable. Les noms des *variables chaîne* de caractères suivent les mêmes règles que les noms des variables numériques sauf qu'ils sont tous terminés par le caractère dollar ($). Voici quelques exemples de noms de variables chaîne de caractères :

A$
MYNAME$
SENTENCES$

Vous avez déjà utilisé une variable chaîne de caractères sans le savoir. Dans le programme WELCOME au Chapitre 2, N$ était utilisé comme variable chaîne pour stocker votre prénom. Regardez le programme maintenant. N$ est un bon exemple de ce que font les variables chaîne : retenir chaque fois qu'un caractère (lettre) est entré au clavier et utiliser les mêmes caractères dans un message pour l'utilisateur. Les variables chaîne peuvent encore faire autre chose, continuez à lire.

Les variables numériques (comme A) sont différentes des variables chaîne (comme A$) : A ne peut contenir qu'un nombre (l'âge de quelqu'un, par exemple) alors que A$ peut contenir une chaîne de caractères (le nom de quelqu'un par exemple). On peut utiliser les deux dans le même programme.

Les chaînes de caractères peuvent prendre toutes sortes de noms, aussi longs que l'on veut pourvu qu'ils se terminent par un signe dollar. Si vous voulez une variable chaîne contenant les lettres HARRY S TRUMAN, vous pouvez utiliser la variable

```
A$ = ''HARRY S TRUMAN''
```

ou

```
NAME$ = ''HARRY S TRUMAN''
```

Les caractères que vous mettez dans la variable chaîne doivent être encadrés par des guillemets. L'instruction

```
PRINT NAME$
```

imprimera le contenu de la variable NAME$: dans ce cas le nom du 33e Président des Etats-Unis.

Fonctions chaînes de caractères

Il existe de nombreuses instructions qui manipulent des chaînes. Supposons que vous vouliez connaître la longueur d'une chaîne de caractères (c'est-à-dire le nombre de caractères qu'elle contient), vous pouvez utiliser la fonction longueur, LEN, en tapant

La fonction chaîne LEN renvoie le nombre de caractères contenus dans la chaîne (spécifié entre parenthèses), ce nombre étant compris entre 0 et 255.

```
PRINT LEN (''HARRY S TRUMAN'')
```

ou alors vous pouvez taper l'instruction équivalente

```
PRINT LEN (NAME$)
```

et Applesoft affichera la longueur de la chaîne. Comme vous pouvez le voir (vérifiez en comptant vous-même) la longueur de la chaine NAME$ est 14. Souvenez-vous que le micro-ordinateur compte les espaces et la ponctuation comme des caractères.

Le nombre de caractères varie de 0 à 255. Plus de 255 caractères par chaîne produisent l'arrêt et l'un des messages d'erreur ?STRING TOO LONG ERROR ou ?SYNTAX ERROR. Une chaîne sans caractère est une *chaîne vide.* L'ordre RUN provoque la réinitialisation des variables à zéro, et remplace les chaînes par des chaînes vides. Chaque chaîne doit donc être définie en début de programme.

Dans certaines occasions peut-être, vous voudrez afficher uniquement une partie de la chaîne de caractères contenue dans NAME$. Pour faire cela, vous disposez de trois fonctions de chaînes de caractères à votre disposition : LEFT$, RIGHT$, et MID$.

Si par exemple, vous voulez imprimer les cinq premières lettres de NAME$, tapez

La fonction de chaîne de caractères LEFT$ renvoie le nombre spécifié de caractères les plus à gauche de la chaîne.

```
PRINT LEFT$ (NAME$,5)
```

et

```
HARRY
```

sera affiché sur l'écran. Les raisons pour lesquelles vous voulez faire cela deviendront bientôt claires. Si vous tapez

La fonction de chaîne de caractères RIGHT$ renvoie le nombre spécifié de caractères les plus à droite de la chaîne.

```
PRINT RIGHT$ (NAME$,5)
```

```
RUMAN
```

apparaîtra.

Voici un court programme qui utilise les fonctions LEN et LEFT$.

```
NEW
 90  NAME$ = ''HARRY S TRUMAN''
100  FOR N = 1 TO LEN(NAME$)
110  PRINT LEFT$(NAME$,N)
120  NEXT N
```

Exécutez ce programme.

Maintenant, écrivez un autre programme en substituant RIGHT$ par LEFT$. La fonction RIGHT$ fonctionne comme LEFT$ à ceci près qu'elle donne les caractères les plus à droite de la chaîne. Que se passe-t-il lorsque vous l'exécutez ?

La fonction de chaîne de caractères MID$ renvoie le corps de chaîne spécifié entre parenthèses.

Pour extraire des caractères situés à l'intérieur de la chaîne, vous disposez de la fonction MID$. Tapez

```
PRINT MID$(NAMES$,7)
```

et l'Apple *II*e répondra avec

parce que S est le septième caractère dans la chaîne. Pour comprendre comment la fonction MID$ opère sur le programme, éditez la ligne 110 pour lire

```
110  PRINT MID$ (NAME$,N)
```

Avez-vous obtenu ce que vous attendiez en lançant le programme ?

Supposez que vous ne vouliez extraire que Y, S et T de la chaîne appelée NAME$. Pour cela, il est nécessaire d'ajouter un autre argument à la fonction MID$.

```
PRINT MID$ (NAME$,5,5)
```

Le second argument qui vaut 5 indique la position du premier caractère à extraire. Le troisième argument indique le nombre de caractères à extraire. Applesoft interprète cette instruction comme : « trouvez le cinquième caractère dans NAME$ et imprimez cinq caractères en commençant au cinquième et en vous déplaçant vers la droite ».

Le premier argument d'une instruction MID$ spécifie le caractère dans la chaîne (pas forcément une lettre parce que la ponctuation et les espaces sont comptés comme des caractères) où MID$ commencera. Si un seul argument est donné, MID$ renverra les caractères à partir du premier nommé jusqu'à la fin de la chaîne. Le second argument, qui est facultatif, indique le nombre de caractères à extraire.

Maintenant changez encore la ligne 110, comme suit, et exécutez à nouveau.

```
110  PRINT MID$ (NAME$,N,6)
```

N'allez pas plus loin dans ce livre si vous n'avez pas essayé les fonctions LEFT$, RIGHT$, et MID$.

Pratique de programmation habituelle utilisant les chaînes de caractères

Le programme ALPHABET sur le disque d'exemple Applesoft illustre quelques pratiques usuelles de programmation utilisant les chaînes de programmation. Chargez-le et listez-le maintenant.

La meilleure façon pour comprendre le fonctionnement de ce programme est de l'étudier soigneusement. Regardez bien ce que représente chaque variable et comment elles sont utilisées. Voyez ensuite les techniques de programmation montrées dans la figure 5-1.

Figure 5-1. Le programme ALPHABET : pratique courante de programmation. Les numéros dans l'illustration correspondent à ceux du texte.

```
NEW
190 REM THE ALPHABET PROGRAM
200 A$ = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
210 PRINT
220 PRINT "TYPE A NUMBER, FROM 1 THROUGH ";LEN
    (A$);"","
230 PRINT "AND I WILL TELL YOU WHICH LETTER HAS
    THAT POSITION IN THE ALPHABET. ";
240 INPUT P
250 IF P > LEN(A$) OR P < 1 THEN GOTO 210
260 PRINT
270 PRINT MID$ (A$,P,1);" IS LETTER NUMBER ";P;
    " IN THE ALPHABET."
280 PRINT : PRINT
290 PRINT "TYPE A LETTER, AND I WILL TELL YOU "
300 INPUT "WHERE IT IS IN THE ALPHABET. ";X$
320 FOR N = 1 TO LEN(A$)
330 IF MID$ (A$,N,1) = X$ THEN GOTO 380
340 NEXT N
350 PRINT
360 PRINT "THAT IS NOT A LETTER OF THE ALPHABET.":
    PRINT
370 GOTO 290
380 PRINT
390 PRINT X$; " IS LETTER NUMBER ";N;" IN THE
    ALPHABET."
400 PRINT
410 GOTO 210
```

1.LEN(A$) détermine la longueur de la chaîne de caractères, c'est-à-dire le A$ nombre qu'elle contient. Quand A$ change, LEN(A$) également. Ceci donne davantage de souplesse au programme et vous permet de définir un alphabet différent sans avoir à changer la suite du programme.

2.L'instruction MID$ à la ligne 270 est interprétée comme « commencer au premier caractère dans A$, aller au P (nombre entré par l'utilisateur) et afficher un caractère ». C'est de cette façon que le programme identifie la lettre de l'alphabet qui a la position spécifiée par P.

3.Remarquez l'utilité des espaces vides dans les instructions PRINT. Qu'arriverait-il à l'affichage sans ces espacements ? (Si vous n'en êtes pas sûr, essayez d'en changer quelques-uns et regardez ce qui se passe quand vous exécutez le programme).

4.Le programme utilise une boucle pour trouver la position d'un caractère dans une chaîne. Cette méthode d'utilisation d'une boucle pour analyser une chaîne de caractères, pas à pas, est communément utilisée.

Exécutez ALPHABET maintenant. Puis essayez de modifier la ligne 200, et exécutez-le de nouveau. Vous êtes libre de modifier ce programme comme il vous plaira.

Duplication de chaînes de caractères

Vous pouvez dupliquer une chaîne de caractères en utilisant un ordre d'affectation comme

```
X$ = A$
```

Cette instruction recopie le contenu de A$ dans X$. Cependant vous ne pouvez pas utiliser une notation partielle de chaîne dans un ordre d'affectation. Par exemple, l'instruction

```
MID$ (X$,3,3) = ''XYZ''
```

n'est pas permise, mais l'instruction

```
X$ = MID$ (A$,24,3)
```

est permise. L'argument de gauche dans une instruction d'affectation doit être une variable.

Epelons à l'envers

Vous plairait-il que l'Apple *II*e épelle votre nom à l'envers ? Hé bien, c'est possible, grâce, par exemple, au programme suivant

```
NEW
100 REM PROGRAM TO SPELL YOUR NAME BACKWARD
110 INPUT ''TYPE YOUR NAME AND I WILL SHOW IT
    TO YOU SPELLED BACKWARD. ''; N$
120 REM REVERSE ORDER OF LETTERS
130 FOR T = LEN(N$) TO 1 STEP -1
140 R$ = R$ + (MID$ (N$,T,1))
150 NEXT T
160 PRINT : PRINT ''YOUR NAME SPELLED BACKWARD
    IS''; R$
170 PRINT : PRINT
180 GOTO 110
```

Exécutez ce programme, en essayant des noms différents. Après plusieurs exécutions vous vous apercevrez que parfois c'est faux. La ligne 140 est la clé du problème. Si, par exemple, vous entrez SALLY, la variable N$ devient SALLY et la variable R$ devient YLLAS. Essayez. Quand le programme retourne à la ligne 110 et vous demande votre nom à nouveau, tapez JOE. Cela va initialiser N$ avec JOE mais l'ancienne variable R$ est toujours en mémoire et par conséquent la ligne 140 met dans R$, l'ancien R$ plus N$ épelé à l'envers. A la place de EOJ (JOE à l'envers) vous verrez s'afficher YLLASEOJ.

Que faut-il faire pour annuler les variables chaînes comme R$ pour les réinitialiser à chaque GOTO. Cela peut heureusement se faire avec Applesoft. Nous pouvons utiliser la chaîne vide pour vider le contenu de R$. Ajoutez cette ligne au programme :

```
175 R$ = ""
```

Maintenant exécutez le programme à nouveau. En utilisant la chaîne vide à la fin du programme, le contenu de R$ peut être annulé après que le nom ait été épelé à l'envers. Ainsi, quand le programme revient à la ligne 110 et recommence, R$ prendra la valeur du nouveau nom au lieu de le rajouter à l'ancien et d'affiche tous les anciens noms qu'on lui avait donné.

CLEAR met toutes les variables, y compris les chaînes et les tableaux, à zéro. Il est très ingénieux de l'utiliser dans un programme.

L'instruction CLEAR d'Applesoft, réinitialise toutes les variables de taille et de forme différentes, mais ne devrait être utilisée qu'en mode d'exécution immédiat.

```
N = 254
PRINT N
```

Maintenant tapez

```
CLEAR
```

puis

```
PRINT N
```

Votre micro-ordinateur vous donne-t-il 0 comme valeur de N ?

Concaténation

Il est possible de rajouter une seconde chaîne de caractères à la fin d'une chaîne existante en utilisant le signe plus (+). Ce procédé est appelé *concaténation*. Essayez ce qui suit :

```
C$ = ''GOOD MORNING''
D$ = C$ + '', '' + ''BILL''
```

L'Apple *IIe* répondra

```
GOOD MORNING, BILL
```

La concaténation est spécialement utile si vous souhaitez prendre une chaîne de caractères séparée et la rattacher ensuite après l'avoir modifiée. Par exemple pour créer une nouvelle chaîne contenant les mêmes caractères que dans D$, mais dans un ordre différent, tapez

```
E$ = RIGHT$ (D$,4) + MID$ (D$,13,2) + LEFT$ (D$,12)
PRINT E$
```

et

```
BILL, GOOD MORNING
```

apparaîtra sur votre écran. Maintenant voilà un programme qui montre une bonne utilisation de la concaténation :

```
NEW
100 INPUT ''GIVE ME ABOUT HALF OF A SENTENCE. ''; HALF$
105 PRINT
110 INPUT ''NOW GIVE ME THE SECOND HALF OF THE
    SENTENCE. ''; OTHERHALF$
120 WHOLE$ = HALF$ + '' '' + OTHERHALF$
130 PRINT
140 PRINT WHOLE$
150 PRINT : PRINT : PRINT : GOTO 100
```

Voilà comment vous faites une concaténation. Essayez de changer le blanc de la ligne 120. Qu'arrive-t-il alors aux deux demi-phrases ?

L'annexe E présente une version plus complexe de mélange de phrases et donne davantage d'indications avec l'affichage du programme.

Encore des fonctions de chaînes de caractères

Les chaînes de caractères peuvent contenir presque tous les types de caractères, y compris des nombres. Cependant, les caractères entre guillemets dans une chaîne ne peuvent être interprétés numériquement, même si ce sont des nombres. Tapez

```
C$ = "123"
PRINT C$ + 7
```

La fonction VAL essaie d'interpréter une chaîne, jusqu'au premier caractère non numérique, comme un nombre entier ou réel et renvoie la valeur de ce nombre.

L'Apple *IIe* répondra ?TYPE MISMATCH ERROR. Cependant, la fonction VAL (abréviation de value) peut résoudre ce problème. La fonction VAL renvoie la valeur du contenu d'une chaîne par opposition à son contenu actuel. Tapez

```
PRINT C$
```

puis tapez

```
PRINT VAL(C$)
```

Les deux instructions produisent apparemment le même résultat, cependant ne vous y fiez pas. Vous avez déjà vu que si vous tapez

```
PRINT C$ + 5
```

votre ordinateur répondra par ?TYPE MISMATCH ERROR. Essayez de taper

```
PRINT VAL(C$) + 5
```

et

```
PRINT VAL(C$) + 5
 1128
```

apparaîtra sur l'écran. Notez que le nom de la variable chaîne qui est l'argument de la fonction VAL doit être entre parenthèses.

Que se passe-t-il si vous voulez mettre la valeur de C$ – 21 dans une variable numérique ? Tapez seulement

```
Q = VAL(C$) – 21
```

Maintenant tapez

```
PRINT Q
```

et voyez ce que vous obtenez. Le contenu de Q est-il ce que vous attendiez ? Vous pouvez même utiliser la fonction VAL pour additionner les valeurs numériques de deux chaînes de caractères. Pour essayer cela, créez une nouvelle chaîne :

```
K$ = ''12''
```

puis tapez

```
P = VAL(C$) + VAL(K$)
PRINT P
```

Essayez la fonction VAL avec différentes chaînes, même avec des chaînes qui commencent ou finissent avec des lettres.

La fonction de chaîne de caractères STR$ renvoie une chaîne qui représente la valeur de l'argument.

Il est parfois nécessaire de changer un nombre en une chaîne. La fonction STR$, qui agit de la même façon que VAL mais à l'inverse, peut être utilisée pour faire ce changement. Supposons que vous vouliez transformer la variable numérique P en une variable chaîne, tapez

```
P$ = STR$(P)
PRINT P$
```

pour voir comment agit la fonction STR$.

STR$ est spécialement utile pour *formater* du texte à l'écran. Vous pourriez utiliser STR$, par exemple, pour aligner toutes les virgules décimales dans une série de nombres. Votre programme convertirait chaque nombre en variable chaîne et utiliserait les fonctions LEN, MID$ et HTAB pour aligner chaque entrée.

Le programme DECIMAL qui se trouve sur la disquette d'exemples Applesoft fait justement cela. Exécutez-le et regardez votre listing et la figure 5-2 pour voir comment il travaille. Ce programme contient une quantité de remarques pour vous aider à comprendre les fonctions des diverses instructions.

Figure 5-2. Le programme DECIMAL : Utilisation de STR$. Les chiffres de l'illustration correspondent à l'explication dans le texte.

```
  5 REM THE DECIMAL PROGRAM
 10 GOTO 1000
100 REM * PRINT ALIGNED NUMBERS *
110 REM P$ MUST CONTAIN NUMBER
120 REM VTAB MUST BE PRE-DONE
130 REM FIND DECIMAL POINT :
135 LET DP = LEN (P$) + 1 : REM SET ALIGNMENT
    BASED ON NO DP
140 FOR CHAR = 1 TO LEN (P$)
150 IF MID$ (P$, CHAR, 1) = "." THEN DP = CHAR:
    REM IF DP FOUND, USE IT
160 NEXT CHAR
165 REM LINE UP DP AND PRINT:
170 CALL -868: REM CLEAR LINE
180 HTAB 15 - DP: PRINT P$
190 RETURN
300 REM * INPUT NUMBER *
310 VTAB 21 : HTAB 1
320 PRINT "WHAT DECIMAL NUMBER"
330 PRINT "WOULD YOU LIKE TO ADD? ";
332 CALL -958: REM CLEAR FROM HERE TO BOTTOM
    OF SCREEN
335 INPUT "": N$: REM "" KILLS QUESTION MARK
340 N = VAL (N$)
350 IF N > 999999 OR N < .01 THEN GOTO 310:
    REM POSITIVE NUMBERS ONLY
360 N$ = STR$ (N)
370 RETURN
1000 REM * MAIN ROUTINE *
1010 TEXT : HOME
1030 GOSUB 300 : REM GET NUMBER
1100 REM MOVE DOWN AND PRINT N$
1110 ROW = ROW + 1: VTAB ROW
1120 P$ = N$ : GOSUB 100
1200 REM ADD NEW NUMBER TO SUM:
1210 SUM = SUM + N
1300 REM PRINT TOTAL:
1310 P$ = STR$ (SUM)
1320 PRINT "TOTAL: ";: GOSUB 100
1400 REM REPEAT UNTIL FULL:
1410 IF ROW < 19 THEN 1030
1420 CALL -958: END
```

1. Les astériques sont utilisés dans les instructions REM pour une identification visuelle. Ils sont généralement utilisés pour marquer le début des sous-programmes et des programmes principaux comme aux lignes 100 et 1000.

2. CALL provoque l'exécution de sous-programmes en langage machine qui font des merveilles sur l'écran : CALL-868 efface la ligne courante ; CALL-958 efface jusqu'au bas de l'écran. Il est judicieux d'utiliser ces instructions CALL en relation avec les instructions INPUT. Pour davantage d'informations, voir l'Annexe E et le *Manuel de Référence Applesoft.*

3. La chaîne nulle (‘‘’’) à la ligne 335 empêche l’affichage d’un point d’interrogation indésirable après l’instruction INPUT.

4. Remarquez que la séquence principale de ce programme commence à la ligne 1000 ; à la ligne 1030, GOSUB 300 provoque le branchement vers l’arrière à l’instruction INPUT. A la ligne 370, le programme revient à la ligne 1100.

5. L’écran n’affichera que 19 lignes de nombres sans défiler, donc la variable ROW est conditionnelle à la ligne 1410.

Les lignes 335 à 360 constituent les premières instructions d’une séquence de protection contre les erreurs. Essayez d’entrez toutes sortes de nombres, sous toutes les formes possibles, pour voir ce qui arrête le programme. Imaginez ensuite différentes façons de corriger ces erreurs de façon à ce qu’elles n’arrêtent plus le programme.

Encore des pièges à erreurs

Comme vous le voyez il y a de nombreuses chausse-trappes qu’un programmeur doit prévoir et dont il doit se garder en écrivant des programmes. Il n’est jamais agréable à l’utilisateur de trouver lui-même des écueils au milieu d’un programme sans savoir comment s’en débarrasser, pas plus qu’au programmeur de voir cela arriver.

Maintenant que vous savez comment VAL et STR$ peuvent être utilisées si le caractère entré n’est pas un nombre positif (ligne 335-360 dans DECIMAL), il serait intéressant de voir comment rajouter ces fonctions dans un programme sur lequel vous avez déjà travaillé. Chargez votre plus récente version de COLORBOUNCE et tapez

```
LIST 350-400
```

Vous devriez voir ces lignes sur votre écran :

```
350 INPUT "WHAT COLOR WOULD YOU LIKE THE BALL TO BE
    (1-15)? "; HUE
370 REM IS HUE OF BALL IN RANGE?
380 IF HUE > 0 AND HUE < 16 THEN GOTO 400
390 HOME : PRINT "THAT WASN'T BETWEEN 1 AND
    15!": PRINT
395 GOTO 310
400 GR : REM SET COLOR GRAPHICS AREA
```

La ligne 380 vérifie que le nombre entier est dans les limites correctes et les lignes 390 et 395 donnent à l'utilisateur un message et une autre chance d'entrer un nombre à l'intérieur des limites. Cependant, le programme ne couvre pas la possibilité d'entrer par l'utilisateur un UN au lieu de 1. Comme vous pourrez voir dans « Interaction entre le programme et les utilisateurs », au Chapitre 4, une telle entrée sera sanctionnée par le message d'erreur ?REENTER mais sans l'indication de ce qui est faux.

Maintenant vous pouvez arrêter le programme pour faire attention une telle possibilité. Suivez ces étapes :

1. Changer la variable HUE à la ligne 350 en une variable chaîn HUE$, de cette façon, s'il arrive à un utilisateur d'épeler un nombre, Applesoft sera capable d'en faire quelque chose. (Souvenez-vous qu'une variable numérique ne peut pas contenir de caractères.)

2. Ajoutez ces lignes au programme :

```
352 REM IS ENTRY A NUMBER?
354 HUE = VAL(HUE$) : REM VALUE OF HUE$ BECOMES
    NUMERIC VARIABLE
356 IF HUE > 999999 OR HUE <.01 THEN GOTO 300 : REM
    INPUT MESSAGE WILL REAPPEAR
358 HUE$ = STR$(HUE) : REM HUE CHANGED BACK INTO
    STRING HUE$
```

Maintenant exécutez la nouvelle version et regardez ce que cela donne lorsque vous épelez un nombre en entrée. Le programme COLORBOUNCESOUND de l'APPLESOFT SAMPLER contient ces modifications et toutes les autres que vous avez faites sur COLORBOUNCE.

Introduction aux tableaux

Un tableau est un type de variable qui est utilisé pour représenter des listes de valeurs liées ensemble par un quelconque arrangeme logique. Vous pouvez imaginer un tableau comme étant une table de nombres d'où vous pouvez choisir des individus ou des éléments (voir figure 5-3). La puissance de programmation donnée par les tableaux compense largement le temps passé à se familiariser avec eux.

Le nom d'un tableau peut être n'importe quel nom légal de variabl Un élément de tableau est composé du nom de tableau suivi de parenthèses, à l'intérieur desquelles se trouvent les indices.

Quand l'instruction DIM est exécutée, elle réserve de la place, en mémoire pour le nombre d'éléments du tableau déclaré.

Pour créer un tableau vous devez d'abord dire à l'ordinateur le nombre d'éléments que vous voulez traiter avec le tableau. Pour cela utilisez l'instruction DIM (DIM signifie « dimension ». Les éléments dans un tableau sont numérotés à partir de zéro, par conséquent pour dimensionner un tableau appelé A qui aura au maximum 16 éléments, tapez

```
DIM A(15)
```

Cette instruction crée une variable tableau à une dimension avec de la place pour 16 éléments. Les éléments se comportent exactement comme les variables que vous avez commencé à connaître et à aimer. Ce sont

```
A(0)
A(1)
A(2)
```

jusqu'à

```
A(15)
```

Les éléments d'un tableau peuvent être utilisés exactement comme les autres variables utilisées. Par exemple, vous pouvez prendre deux éléments d'un tableau A et écrire

```
A(9) = 45 + A(12)
```

Dans un élément de tableau, les nombres (ou expressions entre parenthèses) sont appelés *indices*. Les indices peuvent être une expression numérique ou être représentés par une variable.

Le programme suivant illustre l'utilisation de variables en indices et affiche le contenu de chaque élément du tableau tapez :

```
 90 REM DIMENSION ARRAY CALLED DAYS
100 REM HOLDS 7 NUMBERS
110 DIM DAYS(6)
120 REM FILL THE ARRAY
130 FOR NUM = 0 TO 6
140 DAYS(NUM) = NUM + 1
150 NEXT NUM
160 REM DISPLAY THE ARRAY ELEMENTS
170 FOR I = 0 TO 6
180 PRINT ''DAYS('';I;'') = '';DAYS(I)
190 NEXT I
```

Si on fait appel à un élément de tableau avant que ce dernier n'ait été dimensionné, l'Applesoft réserve automatiquement de la place en mémoire pour 11 éléments (indices 0 à 10). Mais quoiqu'il en soit, c'est une bonne méthode de programmation que de dimensionner tous les tableaux.

Les tableaux peuvent avoir une dimension, par exemple A(15) et DAYS(6), ou alors plusieurs. L'instruction

```
DIM TWO (3,7)
```

définit un tableau à 2 dimensions. Le premier indice peut prendre 4 valeurs (0,1,2,3) et le second en prendre 8 (0,1,2,3,4,5,6,7). Le tableau TWO contient de la place réservée pour un ensemble de 32 éléments différents (4 fois 8). La figure 5-3 illustre le concept de dimensions.

Array DAY (6)

0	1	2	3	4	5	6

Array TWO (3,7)

0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7
1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7
2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7
3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7

Figure 5-3. Dimensions de tableaux. Le tableau DAYS(6) a une dimension et sept éléments. Le tableau TWO (3,7) a deux dimensions et 32 éléments.

Supposons que vous vouliez écrire un programme qui brasse les chiffres de 1 à 8. Pour ce faire vous avez besoin d'utiliser des tables de données. Les tableaux sont précisément conçus pour traiter aisément ce type de problème. Le suivant illustre comment manipuler les « éléments » de tableaux.

```
NEW
200 REM DIMENSION THE ARRAY
210 DIM GLASS(8)
220 REM FILL THE ARRAY
230 FOR I = 1 TO 8
240 GLASS(I) = I
250 NEXT I
260 REM SCRAMBLE THE ARRAY AND CHOOSE EACH ELEMENT
270 FOR WINE = 1 TO 8
280 REM CHOOSE SOME OTHER ELEMENT
290 MILK = INT (RND(1) * 8) + 1
300 REM WAS MILK DIFFERENT FROM WINE?
310 REM IF NOT, TRY AGAIN
320 IF MILK = WINE THEN GOTO 280
330 REM INTERCHANGE GLASS(WINE) AND GLASS (MILK)
340 TEMP = GLASS(WINE) : GLASS(WINE) = GLASS(MILK) :
    GLASS(MILK) = TEMP
350 NEXT WINE
360 REM PRINT CONTENTS OF ARRAY
370 FOR C = 1 TO 8
380 PRINT GLASS(C)
390 NEXT C
```

Exécutez le programme. Comprenez-vous comment il fonctionne ? Le programme impose à chaque élément du tableau de changer de place, ainsi le programme remplit d'abord le tableau avec des nombres et ensuite brasse le contenu des éléments du tableau. Remarquez que vous ne devez pas nécessairement commencer à remplir le tableau à zéro.

Voici une description de ce que les quelques lignes les plus compliquées du programme font. Les lignes 230 à 250 remplissent le tableau et affectent à chaque élément du tableau un nombre qui correspond au rang de l'élément dans le tableau (GLASS(1) = 1,GLASS(2) = 2, etc.). La ligne 270 affecte à la variable WINE les nombres de 1 à 8. La ligne 290 donne à la variable MILK la valeur aléatoire d'un entier de 1 à 8. Ensuite la ligne 320 permet de s'assurer que la valeur de la variable WINE n'est à aucun instant égale à celle de la variable MILK. Le contenu des variables GLASS(WINE) et GLASS(MILK) est interchangé à la ligne 340. Finalement le tableau est affiché au moyen des lignes 370 à 390.

Messages d'erreur concernant les tableaux

Voici quelques messages d'erreur que vous pourriez provoquer en programmant des tableaux.

- ?REDIM'D ARRAY

  Ce message d'erreur apparaît lorsqu'un tableau est dimensionné plus d'une fois au cours d'un même programme. Par exemple les lignes

  ```
  10 A(9) = 15
  20 DIM A(50)
  ```

  provoqueront ce type de message d'erreur. Souvent, cependant ce message apparaît lorsqu'il manque un dimensionnement et plus tard une instruction de dimensionnement est ajoutée au programme.

- ?BAD SUBSCRIPT ERROR

  Si on essaie d'utiliser un élément d'un tableau dont l'indice est en dehors de la dimension, ce message apparaît. Par exemple si A a été dimensionné à 25 par l'instruction DIM A(25), la référence à l'élément A(52) ou n'importe quel autre élément dont l'indice est inférieur à 0 ou supérieur à 25 provoquera le message ?BAD SUBSCRIPT ERROR.

- ?ILLEGAL QUANTITY ERROR

  Vous recevrez ce message si vous essayez d'utiliser un nombre négatif comme indice d'un tableau.

Plusieurs programmes, dans l'Annexe E, utilisent les tableaux. En continuant maintenant par ces programmes vous aurez une meilleure vision des applications possibles des tableaux.

Sommaire du chapitre

Instructions

LEN
LEFT$
MID$
RIGHT$
CLEAR
VAL
STR$
DIM

Termes

chaîne de caractères
tableau
variable chaîne
variable numérique
chaîne vide
concaténation
format
élément
indice
programmes d'application

Messages d'Erreur

?STRING TOO LONG ERROR
?TYPE MISMATCH ERROR
?REDIM'D ARRAY
?BAD SUBSCRIPT ERROR

Conclusion

Maintenant que vous avez été familiarisé avec quelques outils de programmation de l'Applesoft, vous avez plusieurs options.

Si vous pensez que vous en avez suffisamment appris sur la programmation jusqu'à maintenant, vous allez probablement vouloi explorer les programmes d'application : ceux écrits par d'autres personnes et que l'on peut acheter, et qui vous permettent d'utiliser l'Apple *II*e pour toutes sortes de tâches pratiques. Le *Guide de l'Utilisateur Apple II*e comporte un chapitre relatif aux programmes d'application.

Si vous êtes prêt et avide d'améliorer vos connaissances en matière de programmation, voilà ci-dessous quelques suggestions.

- Si vous parcourez ce manuel une nouvelle fois en écrivant vos propres programmes avec les instructions qui vous ont été présentées, vous consoliderez de beaucoup vos connaissances.

- L'Annexe E : « Encore des programmes pour pratiquer » présente quatre nouveaux programmes. Ils apportent des exemples de quelques-unes des applications pratiques que vous pouvez faire avec l'Applesoft et qui sont destinées à vous aider à acquérir une plus grande habileté et compréhension en programmation. L'expérimentation et l'étude des programmes de l'annexe vous aideront à acquérir une bonne pratique de la programmation et vous donneront un grand nombre d'idées pour écrire vos propres programmes.

- Utilisez le *Manuel de Référence Applesoft* pour apprendre à utiliser les instructions Applesoft selon vos besoins. Vous pouvez également améliorer vos talents de programmeur en essayant les exemples de ce manuel.

Une des joies de posséder un ordinateur est dans le développemen de vos propres idées dans des programmes que vous et d'autres peuvent utiliser. Ce livre a présenté des possibilités du BASIC Applesoft. Applesoft a de nombreuses autres possibilités, que vous découvrirez quand vous aurez dominé celles qui vous ont été exposées ici.

A *Sommaire des instructions et commandes* 145

B *Mots réservés en Applesoft* 159

C *Messages d'erreur* 163

D *Aide* 167

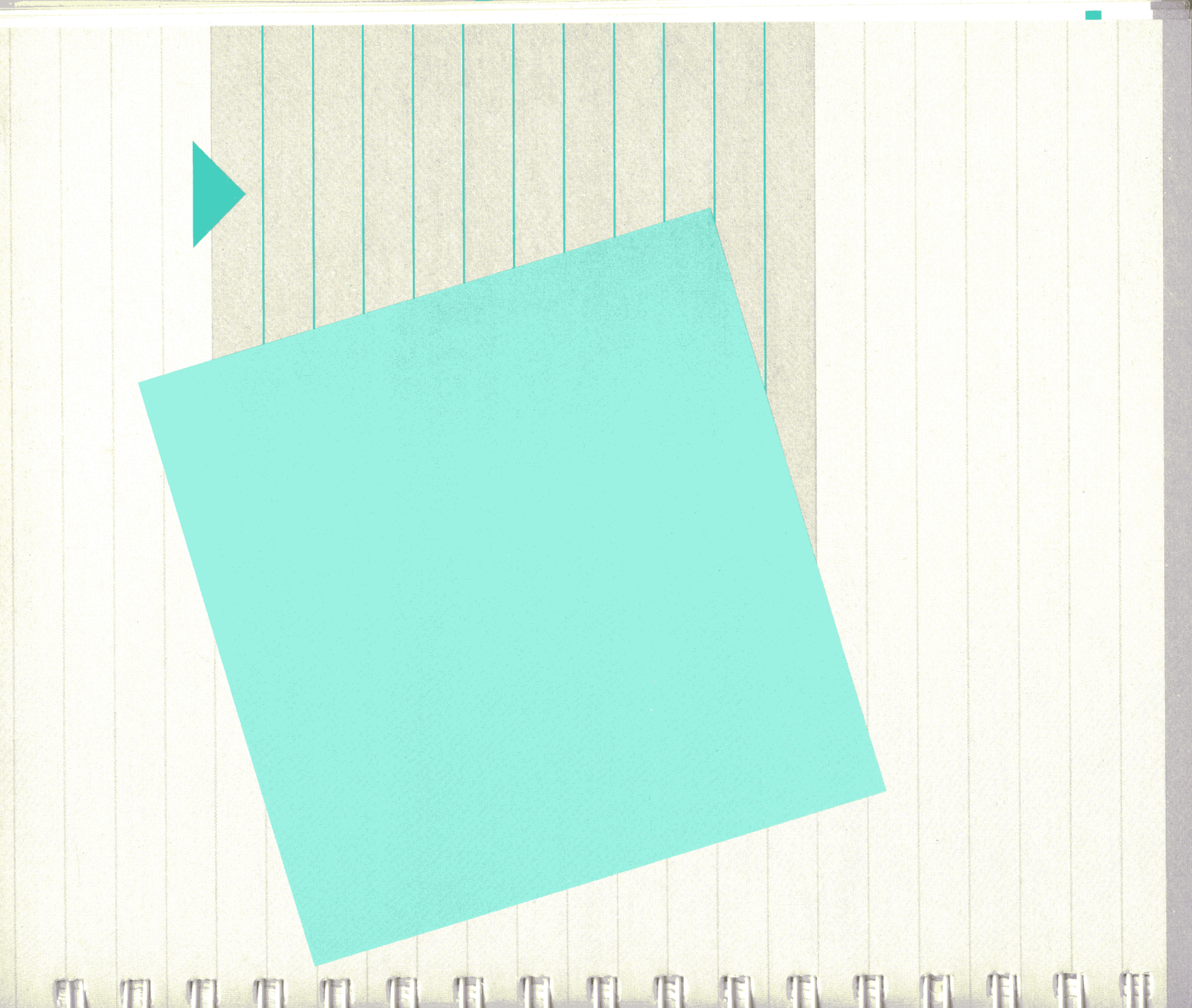
167 Si votre programme (ou vous-même) êtes ''planté'' !
168 Erreurs
168 Instructions et commandes
168 Magnétophones à cassettes
169 Encore des informations utiles
169 Impression de programmes Applesoft
169 La mémoire de l'Apple *II*e
170 Signification du caractère d'attente

Annexes

E

Encore des programmes à essayer 171

172 Conseil pour les programmeurs novices
173 SCRAMBLER
174 Analyse des lignes du programme
177 Réglage fin
178 Programme
180 MAGIC MENU
181 Notes pour les programmeurs confirmés
181 Fonctionnement des cinq sous-programmes : démonstration
182 Le sous-programme INPUT
183 Le sous-programme GET RETURN
183 Le sous-programme Screen Formatter
184 Le sous-programme Menu Maker
186 Le sous-programme Computer Identifier
187 Autres remarques sur MAGIC MENU
188 Vitesse du programme
189 Quelques mots sur les noms de variables
190 Quelques remarques sur la logique
192 Programme
201 Le programme DISK MENU
203 Renumérotation et fusion de portions de programmes
205 Programme
211 CONVERTER
211 Programme
219 Quelques idées



Sommaire des instructions et commandes

Cette annexe est un sommaire des instructions du BASIC de l'Applesoft et des commandes du DOS utilisées dans ce manuel. Le chiffre et/ou la lettre entre crochets, à la fin de chaque description, fait référence au chapitre et/ou à l'annexe où vous pouvez trouver de plus amples explications concernant l'instruction.

Vous avez appris à peu près la moitié des instructions du BASIC de l'Applesoft utilisables sur l'Apple *II*e. Le *Manuel de Référence du BASIC Applesoft* représente un aide mémoire et vous apporte un commentaire clair des instructions de l'Applesoft.

Vous pouvez trouver dans cette annexe quelques instructions qui n'ont pas été vues dans la partie principale du manuel. Elles sont utilisées dans l'Annexe E, *« Encore des programmes pour pratiquer »*.

ASC

La fonction ASC renvoie le code décimal ASCII du premier caractère de son argument. [E]

CALL

CALL provoque l'exécution du sous-programme en langage machine à partir de l'emplacement mémoire précisé par l'adresse décimale de son argument. Par exemple, CALL-868 effacera la ligne courante à partir du curseur jusqu'à la marge de droite. Il est possible d'obtenir le même résultat en utilisant une séquence Esc ou Ctrl . [5,E]

CATALOG

Cette commande du système d'exploitation disque (DOS) affiche la liste de tous les enregistrements d'un disque précisé. Par exemple CATALOG, D2 donne l'ordre au système d'exploitation d'afficher tous les enregistrements contenus dans n'importe quel disque se trouvant dans le second lecteur de disque. Le lecteur de disque 1 (D1) est pris par défaut à moins qu'un autre lecteur soit spécifié comme dans cet exemple.

Le type de fichier et le nombre de secteurs du disque occupés par ce fichier sont précisés sur la gauche du titre du fichier dans le listage provoqué par la commande CATALOG. La nature des fichiers est représentée par des lettres :

I signifie que le fichier est un programme en BASIC Entier.

A signifie que le fichier est un programme écrit en BASIC Applesoft. (Tous les programmes que vous avez sauvés sur disque lors des applications de ce manuel sont de ce type).

T signifie que le fichier contient du texte et qu'il a été crée par la commande WRITE.

B signifie que le fichier a été sauvé sous forme binaire, ou langage machine.

[2]

CHR$

Cette fonction renvoie le caractère ASCII qui correspond à la valeur de son argument, lequel doit être compris entre 0 et 255. Par exemple CHR$(65) renvoie la lettre A. [E]

CLEAR

Cette instruction de l'Applesoft annule toutes les variables, y compris les tableaux et les chaînes de caractères. Elle est utilisée en mode immédiat. A cause du fait qu'elle annule toutes les variables, il est hasardeux de l'utiliser dans un programme. [5]

COLOR=

La couleur pour placer les points dans les graphiques basse résolution se détermine par l'instruction COLOR= suivie par un entier allant de 0 à 15 comme par exemple COLOR = 5. La couleur est remise à zéro par l'instruction GR ; c'est pourquoi GR doit toujours être suivie de l'instruction COLOR= pour que quelque chose apparaisse à l'écran. Les noms des couleurs et leur numéros associés sont :

0	noir	4	vert foncé	8	marron	12	vert
1	magenta	5	gris	9	orange	13	jaune
2	bleu foncé	6	bleu moyen	10	gris	14	vert eau
3	violet	7	bleu clair	11	rose	15	blanc

Sur un écran noir et blanc ou vert, les 16 couleurs apparaissent sous quatre niveaux de gris comme suit :

Gris foncé : 1, 2, 4, 8
Gris moyen : 5, 10
Gris clair : 3, 6, 9, 12
Gris pale : 7, 11, 13, 14
Blanc : 15

[1]

CONT

Cette instruction provoque la reprise de l'exécution du programme ou la continuation après qu'un Ctrl - C, un STOP ou un END aient été utilisés pour arrêter l'exécution. La reprise se fait à l'instruction suivante (comme pour GOSUB) et non au numéro de ligne suivant. Les variables ne sont pas remises à zéro.

Si vous modifiez, ajoutez ou effacez une ligne programme, si vous obtenez un message d'erreur après l'arrêt de l'exécution, CONT n'a plus d'effet.

Si le programme n'a pas été arrêté, CONT n'a pas d'effet. [2]

DATA

L'instruction DATA permet la création d'une liste d'éléments qui peuvent être utilisés par l'instruction READ. Ces éléments peuvent être des constantes, des chaînes de caractères, des réels, des entiers, ou une combinaison de l'ensemble comme le montre l'exemple suivant

DATA SMITH, ''TRUMAN'', 3.17, −6

[E]

DEL

Cette instruction supprime ou efface dans les limites spécifiées du programme, comme par exemple DEL 23,56. Toute autre syntaxe entraînerait le message d'erreur ?SYNTAX ERROR. Pour effacer une seule ligne, par exemple la ligne 35, utilisez soit la forme DEL 35,35, soit tapez le numéro de la ligne et faites un retour chariot. [3]

DELETE

Cette commande du système d'exploitation disque supprime le programme du disque dont le nom a été spécifié. Comme d'autres commandes de système d'exploitation, elle peut être suivie d'un numéro de lecteur de disque si, par exemple, le disque contenant le programme n'est pas sur le drive pris par défaut. DELETE BOUNCE, D2 supprimera le programme BOUNCE du disque du second lecteur, à moins que ce disque ne soit protégé en écriture ou alors que le fichier soit verrouillé. [3]

Note : La touche Del sur le clavier n'est pas la même que la commande DELETE. Voyez le *Manuel de l'Utilisateur Apple IIe* et le *Manuel de Référence de l'Apple IIe*, pour plus amples informations.

DIM

Quand l'instruction DIM est exécutée, elle réserve de l'emplacement mémoire pour un tableau contenant le nombre spécifié d'éléments. Les éléments dans un tableau, appelés indices, sont numérotés à partir de zéro. DIM A (50) dimensionnera ou réservera de la place mémoire, pour un tableau contenant jusqu'à 51 éléments. DIM N$(25) réservera de la place pour 26 chaînes de caractères, de n'importe quelle longueur, du tableau N$.

Si un élément de tableau est utilisé dans un programme avant que celui-ci n'ait été dimensionné, l'Applesoft réservera automatiquement de la place pour 11 éléments (indice 0 à 10). Lorsqu'un RUN, ou un CLEAR, est effectué tous les éléments de tableau sont mis à zéro. [5]

END

L'instruction END arrête le programme en cours et rend la main à l'utilisateur. Aucun message n'apparaît. [4]

FOR

Une instruction FOR, utilisée de pair avec une instruction NEXT, met en œuvre une boucle de programme. La boucle est effectuée le nombre de fois spécifié après le TO de l'instruction. L'utilisation de STEP est optionnelle.

Dans l'instruction FOR W = 1 TO 20... NEXT W, la variable W compte le nombre de fois que l'on doit effectuer les instructions. Les instructions à l'intérieur de la boucle seront exécutées 20 fois, de W égal 1, 2, 3 jusqu'à 20. La boucle finit d'être exécutée avec W = 21 et l'instruction suivant le NEXT W est ensuite exécutée.

L'instruction FOR Q = 2 TO −3 STEP −2... NEXT Q illustre comment utiliser STEP pour incrémenter par autre chose que par 1.

Le contrôle se fait en fin de boucle ; ainsi dans l'exemple FOR Z = 5 TO 4 STEP 3... NEXT Z, les instructions internes à la boucle sont exécutées une fois.

Le message ?NEXT WITHOUT FOR ERROR apparaît si vous essayez d'exécuter un programme avec des boucles croisées. [2]

GOSUB

GOSUB provoque le branchement du progamme à la ligne dont le numéro a été précisé. Le sous-programme commençant à ce numéro de ligne devra s'arrêter par une instruction RETURN, laquelle forcera le retour de l'exécution du programme à l'instruction suivant immédiatement le GOSUB. Par exemple GOSUB 500 provoquera le branchement à l'instruction 500 et continuera jusqu'à la rencontre de RETURN. [4]

GOTO

L'instruction GOTO permet au programme de se brancher à la ligne précisée. Elle est utilisée pour la création de boucles et pour lancer un programme en évitant de réinitialiser les variables. [2]

GR

Cette instruction permet la mise en place du graphique basse résolution. Dans ce mode, l'écran possède une grille invisible de 40 colonnes et 40 lignes numérotées de 0 à 39 et 4 lignes d'espace supplémentaire en bas pour y mettre des commentaires. GR efface l'écran et établit COLOR= à zéro, c'est-à-dire écran noir. [1]

HCOLOR=

Le choix des couleurs pour le graphique haute résolution est effectué par l'instruction HCOLOR=. Les noms des couleurs et leurs numéros associés sont :

0	noir1	4	noir2
1	vert	5	orange
2	violet	6	bleu
3	blanc1	7	blanc2

[4]

HGR

Le mode graphique haute résolution, établi par HGR, crée une grille d'écran de 280 × 160 points et laisse 4 lignes en base pour des messages éventuels. L'écran est effacé en noir et la première page de la mémoire est affichée. Ni HCOLOR= ni la mémoire écran du texte ne sont affectés lorsque HGR est exécuté. Le curseur peut ne pas être visible jusqu'à ce qu'il soit déplacé sur les 4 lignes inférieures de l'écran (fenêtre de texte) en appuyant sur ↵. [4]

HLIN

L'instruction HLIN est utilisée pour tracer des lignes horizontales en graphique basse résolution. Elle utilise la couleur la plus récemment spécifiée. La syntaxe de HLIN est

```
HLIN 10, 30 AT 20
```

Cet exemple trace une ligne horizontale de la colonne 10 jusqu'à la colonne 30 à la ligne 20. [1]

HOME

L'utilisateur de HOME, lorsque vous êtes en mode texte, va effacer tout le texte et placer le curseur au coin supérieur gauche de l'écran. L'utilisation de HOME dans un des deux modes graphiques n'efface que les 4 lignes utilisables pour le texte, sur le bas de l'écran. (Pour effacer les dessins sur l'écran utiliser GR ou HGR). [1]

HPLOT

L'instruction HPLOT sert à positionner des points et des lignes en mode graphique haute résolution, en utilisant la dernière valeur spécifiée pour HCOLOR=. Il y a 3 syntaxes différentes ; chacune a un effet différent :

```
HPLOT 10, 20
```

Positionne un point à haute résolution à la 10e colonne et à la 20e ligne de la grille (invisible) d'écran.

```
HPLOT TO 70, 80
```

Trace une ligne depuis le dernier point tracé jusqu'à la colonne 70, ligne 80, en utilisant la couleur du dernier point affiché (pas nécessairement la dernière couleur spécifiée par HCOLOR=). Si aucun point n'a été positionné antérieurement aucune ligne n'est tracée.

HPLOT 30, 40 TO 50, 60

Positionne une ligne en haute résolution du point colonne 30, ligne 40, au point colonne 50, ligne 60.

La ligne positionnée peut être étendue dans la même instruction presque indéfiniment. L'unique instruction

HPLOT 0,0 TO 279,0 TO 279,159 TO 0,159 TO 0,0

dessine un liseré rectangulaire autour des 4 côtés de l'écran à haute résolution. (Si ça ne marche pas, votre écran a besoin d'être réglé).

HPLOT doit être précédé par HGR pour éviter d'effacer la mémoire de l'Apple *II*e y compris votre programme et vos variables. [4]

HTAB

L'instruction HTAB déplace le curseur vers la gauche ou vers la droite de la colonne spécifiée (1 à 40) sur l'écran. Les déplacements de HTAB sont tous relatifs à la marge gauche de la fenêtre de texte, mais indépendants de la longueur de la ligne. Une ligne a 255 caractères (c'est le nombre de caractères limite de chaque ligne). Comme l'écran est limité à 40 caractères par ligne, HTAB fait progresser le curseur à la ligne suivante pour les emplacements 41 à 80, la ligne suivante vers le bas pour les positions 81 à 120, ainsi de suite. [2]

IF

L'ordre IF... THEN crée une boucle conditionnelle de programme. Il est très utilisé pour limiter l'étendue d'une variable de programme. Quand la condition suivant le IF est évaluée comme fausse (0), tout le reste de cette ligne de programme est ignoré et l'ordinateur va à la ligne suivante. Quand la condition est vraie (1), l'instruction suivant le mot THEN est exécutée.

IF N < = 5 THEN GOTO 100

Dans cet exemple la variable N sera comparée à la condition IF N< =5 et évaluée. Chaque fois que la condition est vraie (quand N est égale à 1, 2, 3, 4 par exemple), le reste de l'instruction sera exécuté et le programme se branchera à la ligne 100 (spécifié par GOTO). Si la condition est fausse quand N = 6 ou 10, par exemple), le GOTO est ignoré.

Les expressions sont évaluées selon leur rang alphabétique.

Un THEN sans le IF correspondant, ou un IF sans le THEN correspondant, causera un ?SYNTAX ERROR. [2]

INPUT

L'ordre INPUT vous permet une interaction avec un utilisateur de programme à partir de l'intérieur du programme. Une instruction INPUT doit nommer la variable qui doit être entrée par l'utilisateur et doit contenir une question ou une instruction. Dans l'exemple

```
INPUT A
```

un point d'interrogation s'affiche sur l'écran quand l'instruction est exécutée et le programme attend que l'utilisateur entre un nombre qui sera affecté à la variable numérique A. Dans l'exemple

```
INPUT ''TYPE AGE THEN A COMMA THEN YOUR NAME''; B, C$
```

la chaîne de caractères s'affiche exactement comme ci-dessus (si le programmeur veut inclure un point d'interrogation, il faudrait l'inclure à l'intérieur des guillemets). Le programme attend que l'utilisateur tape un nombre qui est affecté à la variable B, puis une virgule, puis un nom qui est affecté à la variable chaîne C$. Des entrées multiples doivent être séparées par des virgules ou des pressions de la clé RETURN. INPUT ne peut pas être utilisé en exécution immédiate. [2]

INT

La fonction INT renvoie un entier inférieur ou égal à l'argument donné. Dans l'exemple INT(NUM), si NUM est 2,389, 2 sera renvoyé ; si NUM est −45,12345, −46 sera renvoyé. [4]

INVERSE

L'instruction INVERSE positionne le mode vidéo de façon à afficher des lettres noires sur fond blanc. Utilisez NORMAL pour retourner en lettres blanches sur fond noir. [1]

LEFT$

La fonction de chaîne LEFT$ renvoie le nombre spécifié de caractères les plus à gauche de la chaîne. Si vous tapez PRINT LEFT$ (''APPLESOFT'',5), les cinq caractères les plus à gauche, APPLE, seront renvoyés. [5]

LEN

La fonction de chaîne LEN renvoie le nombre de caractères dans une chaîne (spécifiée entre parenthèses) entre 0 et 255. Dans l'exemple LEN (''AN APPLE A DAY''), 14 sera renvoyé. Dans l'exemple LEN(A$), le nombre renvoyé sera le compte des caractères dans la chaîne A$. [5]

LET

L'instruction LET est utilisée pour définir une variable. On assigne au nom de variable à gauche du signe égal, qui est utilisé en relation avec LET, la valeur de la chaîne ou de l'expression à droite du signe égal. LET est optionnel, les instructions LET A = 23 et A = 23 donnent le même résultat. [1]

LIST

L'instruction LIST affiche les lignes de programme qui sont dans la mémoire de l'Apple *IIe*.

LIST	Affiche tout le programme.
LIST 150	Affiche uniquement la ligne 150.
LIST –200	Affiche depuis le début du programme jusqu'à la ligne 200.
LIST 200–	Affiche depuis la ligne 200 jusqu'à la fin du programme
LIST 200,3000 ou LIST 200-3000	Liste des lignes du programme de 200 à 3000

Ctrl - S est utillisé pour arrêter et pour reprendre un listage. Un listage est arrêté par Ctrl - C, et la commande CONT ne peut pas être utilisée. [2]

LOAD

La commande LOAD, lorsqu'elle est suivie par un nom de fichier, a pour but de trouver le fichier programme, nommé sur le disque, logé dans le lecteur assigné ou pris par défaut. Si le programme est trouvé, il est transféré dans la mémoire de l'ordinateur. LOAD efface tout programme dans l'ordinateur avant de placer le nouveau programme en mémoire.

LOAD est une instruction utilisée également avec les enregistreurs de cassettes. Quand il n'est pas suivi par un nom de fichier, LOAD lit un programme Applesoft à partir d'une cassette magnétique dans la mémoire de l'ordinateur. Aucun caractère d'avis n'est donné : l'utilisateur doit rembobiner la bande magnétique et appuyer sur le bouton ''play'' sur le magnétophone pour charger. On entend un bip quand l'information est trouvée sur la bande que l'on charge. Quand le chargement a été réussi, un second bip est émis et le caractère Applesoft d'avis (]) est affiché. Le fait de simplement appuyer sur la touche [Reset] peut interrompre la commande. Regardez le *Manuel de Référence Applesoft* pour une liste de toutes les instructions utilisées avec les magnétophones à cassettes. [2]

MID$

La fonction de chaîne MID$ renvoie une sous-chaîne spécifiée entre parenthèses. Si vous tapez PRINT MID$(''AN APPLE A DAY'',4) les caractères depuis le quatrième jusqu'au dernier de la chaîne seront renvoyés : APPLE A DAY. Dans l'exemple MID$(''AN APPLE A DAY'',4,9), les neuf caractères à partir du quatrième caractère de la chaîne seront renvoyés : APPLE A D. [5]

NEW

L'instruction NEW efface le programme courant de la mémoire et positionne toutes les variables à zéro. [2]

NEXT

NEXT est utilisé à l'intérieur d'une boucle FOR/NEXT. Voyez l'instruction FOR pour explication. [2]

NORMAL

NORMAL positionne le mode vidéo dans les habituelles lettres blanches sur fond noir. [1]

NOTRACE

L'instruction NOTRACE annule l'instruction TRACE. Voir TRACE. [4]

ONERR GOTO

ONERR GOTO est utilisé pour s'affranchir d'un message d'erreur qui stoppe l'exécution quand une erreur apparaît. A l'exécution, ONERR GOTO positionne un drapeau qui provoque un saut inconditionnel au numéro de ligne indiqué si une erreur est rencontrée dans la suite du programme. [E]

PEEK

La fonction PEEK renvoie le contenu, sous forme décimale, de l'octet mémoire dont l'adresse est spécifiée. Dans l'exemple PEEK (−16336), l'adresse, ou argument, est −16336. Cette adresse particulière correspond à l'adresse mémoire du haut parleur de l'Apple *IIe*. Quand la fonction est exécutée, le haut parleur produit simplement un clic audible. [4,E]

PLOT

En mode graphique basse résolution, l'instruction PLOT, positionne un pavé à l'endroit indiqué. Dans l'exemple PLOT 10, 20, un pavé sera placé à la colonne 10, ligne 20. La couleur du pavé est déterminée par la plus récente valeur de COLOR=, la couleur noire est prise par défaut. [1]

POKE

POKE stocke l'équivalent binaire du second argument (3 dans l'exemple qui suit) dans l'emplacement mémoire dont l'adresse décimale est donnée par le premier argument (34 dans l'exemple). Les instructions POKE sont très utiles pour réaliser certaines choses, comme commuter la fenêtre texte/graphique et pour contrôler la taille et le défilement de la fenêtre de texte. Dans l'exemple POKE 34,3, la marge haute de l'affichage est positionnée 3 lignes plus bas que le haut et le texte est décalé jusqu'à cette marge. Un autre exemple est POKE 33,33 qui rétrécit la largeur de la fenêtre de texte. [E]

PRINT

C'est la première des instructions Applesoft utilisées pour afficher des informations sur l'écran. L'instruction PRINT peut afficher un nombre, comme dans PRINT 150 ; elle peut aussi afficher le contenu d'une variable, comme dans PRINT N ; elle peut afficher un groupe de caractères contenu entre guillemets comme dans PRINT ''HELLO THERE'' et afficher une ligne vide, comme dans PRINT, ce qui provoque un saut de ligne et un retour chariot.

Pour afficher une liste de termes sans séparation entre eux, utilisez des points virgules dans l'instruction PRINT. Pour afficher une liste de termes dans des champs séparés, utilisez des virgules. [1]

PR#

PR# est une commande du système d'exploitation disque (DOS) qui envoie des informations sur un point d'entrée sortie spécifié, 1 à 7. Pour envoyer des informations au point 1, où une imprimante est habituellement connectée, vous tapez PR#1. PR#0 renvoie sur l'écran d'affichage, bien que ce ne soit pas la meilleure méthode à utiliser avec certaines cartes périphériques. [E]

READ

Quand un programme exécute une instruction READ, il cherche une instruction DATA. Il utilise le premier élément de l'instruction DATA comme une variable dans l'instruction READ. Les éléments successifs de DATA sont successivement assignés aux variables du READ, chaque fois que READ est exécuté. Dans l'exemple READ A, B, C$, les deux premiers éléments de l'instruction DATA doivent être des nombres et le troisième élément doit être une chaîne de caractères. Les éléments seront donc successivement affectés, respectivement, aux variables A, B, et C$. [E]

REM

REM est une instruction qui vous permet de mettre une remarque, ou un commentaire, dans un programme. Les instructions REM ne sont pas affichées à l'écran ou exécutées par un programme. Elles sont utilisées par les programmeurs pour commenter les lignes de programmes. [2]

RETURN

L'instruction RETURN est utilisée à la fin des sous-programmes. Elle provoque le branchement du programme à l'instruction qui suit immédiatement le dernier GOSUB rencontré. [4]

RIGHT$

La fonction de chaîne de caractères RIGHT$ renvoie le nombre spécifié de caractères les plus à droite dans la chaîne. Si vous tapez PRINT RIGHT$ (''SCRAPPLE'',5), APPLE (les 5 caractères les plus à droite) sera renvoyé. [5]

RND

La fonction arithmétique RND renvoie un nombre réel aléatoire plus grand ou égal à zéro et plus petit que 1. Chaque fois que RND est utilisé avec un argument positif quelconque, un nouveau nombre compris entre 0 et 1 est généré, à moins qu'il ne fasse partie d'une suite de nombres aléatoires générés par un argument négatif. RND(0) renvoie le dernier nombre aléatoire généré. [4,E]

RUN

L'instruction RUN met à zéro toutes les variables et commence l'exécution du numéro de ligne indiqué.

RUN	Exécute le programme en entier, en commençant par le plus petit numéro de ligne.
RUN 130	Commence l'exécution à la ligne 130 (ou à n'importe quelle ligne spécifiée) et continue jusqu'à la fin du programme.

RUN suivi d'un nom de fichier est une commande du système d'exploitation disque (DOS). Elle charge le fichier nommé à partir du lecteur de disque spécifié ou pris par défaut et ensuite exécute le programme qui a été chargé. [2]

SAVE

SAVE, suivi par un nom de fichier est une commande du système d'exploitation disque (DOS) qui sauve sur disque le programme courant en mémoire. Si on donne l'ordre SAVE AGE, et qu'aucun fichier appelé AGE n'est trouvé sur le disque dans le lecteur spécifié ou pris par défaut, un fichier est créé sur ce disque et le programme courant en mémoire est sauvé sous ce nom de fichier. Si le disque contient déjà un fichier dans le même langage de programmation avec le nom demandé, les enregistrements originaux sont perdus dans le fichier et le programme courant est sauvé à leur place. Aucun avertissement n'est donné.

SAVE, utilisé sans nom de fichier, stocke le programme courant en mémoire sur une cassette magnétique. Aucun signal d'avertissement n'est donné, l'utilisateur doit enfoncer simultanément les touches d'enregistrement et de lecture sur le magnétophone avant que l'ordre SAVE soit exécuté. L'ordre SAVE ne vérifie pas si les bonnes touches sont enfoncées ; un bip sonore signale le début et la fin d'un enregistrement. Voyez le *Manuel de Référence de l'Applesoft* pour avoir la liste de toutes les instructions pour magnétophones à cassettes. [2]

STR$

La fonction STR$ renvoie une chaîne de caractères qui représente la valeur de l'argument. Dans l'exemple STR$(12.45), 12.45 est renvoyé. [5]

TAB

La fonction d'affichage TAB, qui doit être utilisée dans une instruction PRINT, déplace le curseur sur des tabulateurs sur l'écran. Ses arguments doivent être compris entre 0 et 255 et entre parenthèses.

Si l'argument est plus grand que la valeur courante de la position du curseur, TAB déplace le curseur à la position spécifiée pour imprimer, en comptant à partir du bord gauche du début de ligne où le curseur est positionné. Si l'argument est plus petit que la valeur courante du curseur, ce dernier ne bouge pas. [2]

TEXT

L'instruction TEXT positionne l'écran dans le mode non graphique habituel pour le texte, avec 40 caractères par lignes et 24 lignes. Quand elle est utilisée pour quitter le mode graphique, il est préférable de l'utiliser avec l'instruction HOME. Ce qui a pour effet de positionner la fenêtre texte sur tout l'écran. [1]

TRACE

L'instruction de mise au point TRACE provoque l'affichage à l'écran du numéro de chaque ligne pendant son exécution. TRACE n'est pas désactivé par RUN, CLEAR, NEW, DEL ou Reset , seul NOTRACE le désactive. [4]

VAL

La fonction VAL tente d'interpréter une chaîne de caractères, jusqu'au premier caractère non numérique, comme un nombre réel ou entier et renvoie la valeur de ce nombre. S'il n'y a aucun nombre avant le premier caractère non numérique, elle renvoie zéro. [5]

VLIN

En mode graphique basse résolution, VLIN dessine une ligne verticale dont la couleur est celle définie par la plus récente instruction COLOR= . La ligne est dessinée dans la colonne spécifiée par le troisième argument. Dans l'exemple VLIN 10, 20 AT 30, la ligne est dessinée de la ligne 10 à la ligne 20 sur la colonne 30. [1]

VTAB

VTAB déplace le curseur vers la ligne spécifiée, verticalement, sur l'écran. La ligne du haut est le numéro 1 et celle du bas est le numéro 24. VTAB déplace le curseur verticalement vers le haut ou vers le bas, mais pas vers la gauche ou la droite. [2]

Mots réservés en Applesoft

La liste qui suit contient tous les mots réservés en Applesoft BASIC. Ils sont réservés pour être utilisés comme dés dans les instructions en Applesoft. Quand Applesoft les trouve dans un programme, il essaie de l'exécuter et ne comprend que leur sens dans le langage. Dans la plupart des cas ces mots réservés ne peuvent pas être utilisés comme noms de variables. Les commentaires à la fin de la liste notent les exceptions. Voyez le *Manuel de Référence du Programme en BASIC Applesoft* pour une explication des instructions non abordées dans ce manuel.

&	GET	NEW	SAVE
	GOSUB	NEXT	SCALE=
ABS	GOTO	NORMAL	SCRN(
AND	GR	NOT	SGN
ASC		NOTRACE	SHLOAD
AT	HCOLOR=		SIN
ATN	HGR	ON	SPC(
	HGR2	ONERR	SPEED=
CALL	HIMEM:	OR	SQR
CHR$	HLIN		STEP
CLEAR	HOME	PDL	STOP
COLOR=	HPLOT	PEEK	STORE
CONT	HTAB	PLOT	STR$
COS		POKE	
	IF	POP	TAB(
DATA	IN£	POS	TAN
DEF	INPUT	PRINT	TEXT
DEL	INT	PR£	THEN
DIM	INVERSE		TO
DRAW		READ	TRACE
	LEFT$	RECALL	
END	LEN	REM	USR
EXP	LET	RESTORE	
	LIST	RESUME	VAL
FLASH	LOAD	RETURN	VLIN
FN	LOG	RIGHT$	VTAB
FOR	LOMEM:	RND	
FRE		ROT=	WAIT
	MID$	RUN	
			XPLOT
			XDRAW

L'Applesoft ''transforme'' en jetons ces mots réservés : chaque mot n'occupe qu'un octet d'espace mémoire. Habituellement un caractère seul occupe un octet.

- Le ''et commercial'' (&) est destiné à l'usage interne du micro ordinateur ; ce n'est pas une instruction propre à l'Applesoft. Ce symbole, lorsqu'il est exécuté comme une instruction, provoque un saut inconditionnel à l'adresse $3F5.

- XPLOT est un mot réservé qui ne correspond pas à une instruction courante en Applesoft.

Certains mots réservés sont reconnus par l'Applesoft seulement dans certains contextes :

- COLOR=, HCOLOR=, SCALE=, SPEED= et ROT= sont interprétés comme des mots réservés seulement si le caractère qui suit est le signe égal (=). Dans le cas de COLOR= et de HCOLOR=, c'est de peu d'intérêt car le mot réservé OR empêche de toute façon leur utilisation comme noms de variables.

  Quand vous essayez d'exécuter une instruction comme 10 COLORFUL=5 vous obtenez ?SYNTAX ERROR. Si vous essayez de lister la même instruction, elle est découpée en

```
10 COL OR FUL = 5
```

- SCRN, SPC, et TAB sont reconnus comme des mots réservés seulement si le caractère non blanc qui suit est une parenthèse ouvrante <(>.

- HIMEM doit être suivi de deux points (:) pour être reconnu comme un mot réservé.

- LOMEM nécessite aussi deux points (:).pour être reconnu.

- ATN est reconnu comme un mot réservé seulement s'il n'y a pas d'espace entre le T et le N, le mot réservé AT est reconnu à la place de ATN.

- TO est interprété comme un mot réservé à moins qu'il ne soit précédé par un A avec un blanc entre le T et le O. Si un blanc est entre le T et le O, le mot réservé AT est reconnu à la place deTO.

Quelquefois les parenthèses peuvent être utilisées pour entourer les mots réservés :

100 FOR A = LOFT OR CAT TO 15 se liste ainsi


```
100 FOR A = LOF TO RC AT TO 15
```

Alors que, 100 FOR A = (LOFT)OR(CAT) TO 15 se liste


```
100 FOR A = (LOFT) OR (C AT) TO 15
```

Messages d'erreur

Cette annexe présente tous les messages d'erreur du BASIC de l'Applesoft. Le *Manuel de Référence BASIC d'Applesoft* contient davantage d'informations relatives à la récupération des erreurs et à la mise au point de programmes.

Quand une erreur apparaît, le BASIC retourne en mode commande indiqué par le caractère d'attente]. Les valeurs des variables dans le programme restent intactes mais le programme ne peut continuer même par la commande CONT. Tous les compteurs de boucles FOR et d'appels de GOSUB sont remis à zéro.

Le format des messages d'erreur est le suivant :

- Errors in immediate execution statements display as ?XX ERROR.

- Errors in deferred execution statements display as ?XX ERROR IN YY.

Dans les deux formats XX est le nom de l'erreur spécifique. En mode programme YY est le numéro de ligne de l'instruction où l'erreur est apparue. Dans ce mode les erreurs ne sont pas détectées tant que l'instruction n'est pas exécutée.

Tous les messages d'erreur sont précédés par un point d'interrogation (?). Si un message d'erreur est affiché sur votre écran sans point d'interrogation il s'agit d'un message ne correspondant pas à une erreur (tel que BREAK IN 110) ou un message du DOS ; voir le *Manuel DOS* pour des informations complémentaires. Un message d'erreur est précédé par trois astériques (***) comme dans ***SYNTAX ERR, qui est un message d'erreur du BASIC entier.

Messages d'erreurs et leurs explications.

?BAD SUBSCRIPT ERROR

Une tentative de référence à un élément de tableau en dehors de la plage des variations des indices a été faite. Cette erreur peut apparaître si un mauvais nombre de dimensions a été utilisé dans une référence ou si un indice est supérieur à la taille limite ; par exemple, LET A(11) = Z quand A a été dimensionné par DIM A(2).

?CAN'T CONTINUE ERROR

Tentative de poursuivre l'exécution d'un programme :

- le programme n'existe pas
- une errreur est apparue
- une ligne a été changée, supprimée, ou ajoutée au programme.

?DIVISION BY ZERO ERROR

Une division par zéro produit ce message.

?FORMULA TOO COMPLEX ERROR

Plus de deux instructions IF...THEN ont été exécutées.

?ILLEGAL DIRECT ERROR

Vous ne pouvez pas utiliser les instructions INPUT, DEF FN, GET, ou DATA en mode immédiat.

?ILLEGAL QUANTITY ERROR

Le paramètre passé à une fonction standard de type numérique ou chaîne, est en dehors des limites :

- un indice de tableau négatif (exemple LET A(−1) = 0)
- utilisation de LOG avec un paramètre négatif ou nul
- utilisation de SQR avec paramètre négatif
- A ^ B avec A négatif et B en entier
- utilisation de MID$, LEFT$, RIGHT$, WAIT, PEEK, POKE, TAB, SPC, ON...GOTO, ou d'une fonction graphique avec un paramètre incorrect.

?NEXT WITHOUT FOR ERROR

L'instruction NEXT rencontrée contient une variable ne correspondant pas à celle de la dernière instruction FOR exécutée, ou une instruction NEXT rencontrée sans aucune instruction FOR préalablement exécutée. Les trois causes les plus fréquentes d'apparition de cette erreur sont : d'oublier de frapper l'Instruction FOR ou NEXT appropriée, de taper un mauvais nom de variable après l'instruction NEXT, ou de provoquer un branchement à l'intérieur de la portée d'une boucle FOR.

?OUT OF DATA ERROR

Une instruction READ a été exécutée mais toutes les instructions DATA du programme ont déjà été lues. Le programme essaie de lire plus de données que le programme n'en contient.

?OUT OF MEMORY ERROR

Les erreurs suivantes peuvent produire ce message :

- programme trop grand
- trop de variables
- plus de 10 niveaux de boucles FOR emboîtés
- plus de 24 appels consécutifs d'instructions GOSUB
- expressions trop complexes
- plus de 36 niveaux de parenthèses
- tentative de donner à LOMEM une valeur trop élevée ou une valeur inférieure à la valeur actuelle
- tentative de donner à HIMEM une valeur trop basse.

?OVERFLOW ERROR

Le résultat d'un calcul est trop grand pour être représenté dans le format de l'Applesoft. Si un sous-passement apparaît, la valeur prise comme résultat est zéro et l'exécution se poursuit sans qu'un message d'erreur ne soit imprimé.

?REDIM'D ARRAY ERROR

Après qu'un tableau ait été dimensionné, une autre instruction de dimension portant sur le même tableau a été rencontrée. Cette erreur apparaît souvent si le tableau a utilisé le dimensionnement implicite et si ensuite ce tableau figure dans une instruction DIM du programme. Ce message d'erreur est utile si vous souhaitez savoir le numéro de la ligne où le tableau a été dimensionné. Vous insérez une instruction dimension pour ce tableau à la première ligne et exécutez le programme et Applesoft vous signalera alors l'instruction contenant le dimensionnement original.

?RETURN WITHOUT GOSUB ERROR

Une instruction RETURN a été rencontrée sans correspondance avec une instruction GOSUB.

?STRING TOO LONG ERROR

Par l'intermédiaire de l'opérateur de concaténation (+), votre programme a tenté de créer une chaîne de longueur supérieure à 255 caractères. Cette erreur tend à apparaître quand une variable chaîne est utilisée plusieurs fois sans avoir été réinitialisée.

?SYNTAX ERROR

Parenthèses oubliées dans une expression, caractère illégal dans une ligne, ponctuation incorrecte, ou autre erreur de format. Cette erreur est souvent causée par une simple erreur de frappe.

?TYPE MISMATCH ERROR

La partie gauche d'une instruction d'affectation est de type numérique alors que la partie droite est de type chaîne, ou une fonction attendait un paramètre de type chaîne alors que vous lui avez donné un argument numérique ou vice et versa. Ceci apparaît souvent parce que le caractère ($)a été oublié.

?UNDEF'D STATEMENT ERROR

Tentative d'exécuter une instruction GOTO, GOSUB ou THEN devant se brancher à un numéro de ligne inexistant. Ceci peut se produire par destruction accidentelle d'une ligne, changement d'un numéro de ligne sans en reporter les conséquences sur les références à cette ligne ou simple erreur de frappe.

?UNDEF'D FUNCTION ERROR

Une référence a été faite à une fonction utilisateur qui n'a pas été définie.

Aide

Cette annexe vous donne quelques idées relatives à l'attitude à tenir lorsque tout ne se passe pas comme prévu, et où chercher l'information nécessaire.

Si votre programme (ou vous-même) êtes ''planté'' !

Un jour, pendant que vous travaillerez avec plaisir, vous découvrirez que plus rien ne se produit sur l'écrant. Si vous appuyez sur la touche [↵], rien ne se produit à l'écran. Il en est de même si vous essayez une autre touche. Le curseur peut même avoir disparu.

Certains professionnels disent dans le jargon informatique que le système est planté !

Il y a au moins 6 méthodes de revenir en mode normal, méthodes qui sont présentées par ordre de ''sévérité'' croissante. Il est toujours préférable d'essayer d'abord les méthodes 1 ou 2.

1. Appuyez sur la touche [Esc] située au coin supérieur gauche du clavier. Son nom complet est escape, (ou encore échappement !) Si vous avez de la chance, cela suffira pour refaire démarrer votre Apple *II*e.

2. Appuyez sur [Ctrl] - C. Beaucoup de programmes interprètent cette touche comme une demande d'annulation.

3. Appuyez sur [Ctrl] - C puis sur [↵].

4. Maintenez la touche [Ctrl] enfoncée et appuyez sur la touche [Reset] (loin sur le côté avant droit du clavier). [Ctrl] - [Reset] provoque le démarrage du programme résident.

5. Appuyez sur la touche [☝] et maintenez la enfoncée pendant que vous appuyez sur les touches [Ctrl] et [Reset]. Ce procédé est assez drastique en ce sens qu'il interrompt le programme en cours, efface la mémoire. Vous trouverez des détails relatifs à cette procédure dans le *Manuel de Référence Apple II*e.

6. Coupez l'alimentation électrique. Vous serez rarement obligé d'aller si loin, et il est plus facile d'utiliser la méthode 5 que de couper et de remettre l'alimentation.

Vous pouvez utiliser les mêmes méthodes pour sortir d'un programme.
Habituellement un programme (s'il est bien écrit) vous dit ce qu'il faut faire, mais il y a des cas où vous ne disposez pas de cette facilité.

Erreurs

Dans ce manuel, les messages d'erreur de l'Applesoft sont expliqués dans le texte du manuel et dans l'Annexe C. D'autres informations sont données à ce sujet dans le *Manuel de Référence Applesoft.*

Le système d'exploitation, comme DOS ou Pascal, donne également des messages d'erreur. Si vous trouvez donc un message qui ne vous est pas familier, regardez le manuel relatif au système d'exploitation.

Instructions et commandes

Les instructions de l'Applesoft sont décrites complètement dans le *Manuel de Référence Applesoft.*

Les commandes utilisées par le système d'exploitation disque sont décrites dans le *Manuel DOS.*

L'Apple *II*e comporte des sous-programmes d'entrées/sorties incorporées dans le firmware (mémoire morte). Voir à ce sujet le *Manuel de Référence Apple II*e pour des informations complémentaires.

Magnétophones à cassettes

Tous les programmes de ce manuel peuvent être sauvegardés sur des cassettes si vous ne disposez pas d'unité de disque. Le *Manuel de Référence Applesoft* explique comment utiliser un magnétophone à cassette avec l'Apple *II*e.

Encore des informations utiles

Les informations suivantes, bien qu'un peu disparates, sont loin d'être inutilisées.

Impression de programmes Applesoft

Si votre Apple *II*e dispose d'une imprimante connectée, ou si vous avez accès à un matériel, utilisez les instructions suivantes pour imprimer vos programmes.

1. Chargez le programme.
2. Frappez PR#1 si l'imprimante est connectée par le connecteur1. Si le circuit imprimé est connecté à un connecteur différent, frappez le numéro de connecteur après PR#.
3. Frappez RUN si vous voulez que les résultats de programme soient imprimés.
4. Frappez LIST si vous voulez que le programme proprement dit soit imprimé.

La mémoire de l'Apple IIe

La mémoire de l'Apple *II*e peut être utilisée à des fins diverses :

1. Pour stocker les instructions constituant votre programme.
2. Pour stocker les variables, chaînes et résultats intermédiaires et finaux.
3. Pour stocker les informations dont l'ordinateur a besoin : information pour le système, pour votre programme et des informations relatives à différentes choses contenues en mémoire.
4. Pour créer texte et graphique basse résolution qui normalement apparaît sur votre écran vidéo.
5. Pour créer du graphique haute résolution qui peut être affiché sur votre écran vidéo.

Chacune de ces utilisations occupe en général, une portion différente de la mémoire de l'ordinateur. L'information est placée dans différentes cases mémoire repérées par leur *adresse.* Un bloc de 1024 cases est appelé *1K* de mémoire. Chaque case est repérée par son adresse, nombre entier qui permet à l'Apple *IIe* de trouver la case pour ranger ou trouver l'information. Ces éléments d'information, que vous voyez rarement dans le détail sous forme de langage machine, sont appelés *octets.* Chaque octet occupe une case mémoire.

La portion de la mémoire de l'Apple *IIe*, occupée dans un but déterminé, peut être décrite en terme de cases mémoire utilisées, habituellement désignées par la plage d'adresse mémoire. Si une plage d'adresse mémoire est utilisée pour stocker votre programme, par exemple, ces adresses ne peuvent pas être utilisées pour créer du graphique haute résolution sinon votre programme sera perdu.

Dans le BASIC de l'Applesoft, les adresses mémoire et les autres nombres sont affichés sous la forme *décimale* habituelle. L'ordinateur utilise cependant une représentation *hexadécimale.* Pour vider les programmes expérimentés, les adresses mémoires sont quelquefois affichées avec les deux formes. Habituellement les nombres hexadécimaux sont précédés du caractère dollar ($).

Signification du caractère d'attente

L'une des fonctions du caractère d'attente est de vous indiquer que l'ordinateur attend une entrée et de vous indiquer le langage attendu. Voici les caractères d'attente que vous verrez fréquemment :

* pour un programme Moniteur

\> pour le BASIC Entier

] pour le BASIC de l'Applesoft

En regardant de caractère, vous pourrez facilement dire (si vous oubliez) quel langage est utilisé par l'ordinateur.

Encore des Programmes à essayer

Cette annexe contient des listes de programmes commentées. Elle présente des exemples pratiques d'utilisation du BASIC Applesoft et a plusieurs buts :

1. De construire des programmes astucieux et compréhensibles. Dans ces programmes, l'accent est mis sur l'entraînement à la bonne programmation et vous aide à apprendre l'utilisation de quelques techniques et instructions nouvelles.

2. De permettre un pont entre l'introduction à la programmation présentée dans ce manuel et les concepts de programmation avancés présentés dans le *Manuel de Référence du BASIC de l'Applesoft.*

3. D'offrir des sous-programmes que vous pourrez incorporer dans vos propres programmes.

4. de continuer à rendre la programmation amusante. Les programmes présentés ici sont destinés à être utilisés, modifiés, expérimentés puis abandonnés.

Ces programmes sont sur le disque APPLESOFT SAMPLER que vous avez utilisé au cours de la lecture et de votre travail avec ce manuel. Voici leur nom dans l'ordre de la présentation.

- SCRAMBLER : jeu de mélange de phrases qui montre clairement l'utilisation de certaines instructions et de la fonction RND. Il laisse en outre ''la bride sur le cou'' à l'écrivain imaginatif.

- MAGIC MENU : un programme surprise qui contient de nombreuses routines utiles pour l'affichage de texte, l'entrée de texte en utilisant le caractère souligné lié au curseur et la création de menus conviviaux, toutes avec un temps d'écriture minimum.

- DISK MENU : programme qui affiche un menu présentant les programmes contenus sur le disque SAMPLER. Il contient un exemple de menu et utilise les sous-programmes contenus dans le programme MAGIC MENU.

- CONVERTER : squelette de programme utilisé pour des conversions d'unité. Ce programme vous laisse beaucoup de place pour vous permettre d'ajouter des instructions et d'expérimenter vos propres conversions pendant que vous apprenez à tailler un programme pour satisfaire vos besoins. Ce programme vous guide durant ce processus et vous aide à construire vos propres menus.

Avant de bricoler avec les programmes du disque APPLESOFT SAMPLER, faites une copie de sauvegarde selon le procédé dans le *Guide de l'Utilisateur de l'Apple IIe.* Ensuite vous pourrez exécuter ces programmes, ''jouer avec eux'' et voir ce qu'ils font.

Conseil pour les programmeurs novices

Jusqu'à pésent vous avez appris de nombreuses instructions BASIC : le but de cette annexe est de vous montrer quelques techniques pour utiliser efficacement ces instructions. De même qu'en rédaction, peinture et autres arts vous pouvez apprendre beaucoup en étudiant le travail des autres.

Vous apprendrez à éviter le piège du programme inutilisable car écrit de telle manière que même le programmeur ne peut pas le comprendre. Vous apprendrez à éviter le programme ''plat de spaghetti'' : programme rempli d'instructions GOTO. Au lieu de cela vous apprendrez comment les professionnels écrivent des programmes ayant une structure de blocs, ce qui permet à eux mêmes et à d'autres de comprendre ces programmes et de les modifier.

C'est toujours un choc de se trouver soudainement en présence de programmes de 2,5 ou 10 pages alors que l'on a tout juste écrit des programmes n'excédant pas dix lignes. Cependant, ils peuvent rester simples, comme par exemple le programme SCRAMBLER. Vous avez maintenant assez d'habileté pour le comprendre. Les trois autres programmes sont plus complexes mais vous serez plus récompensé quand vous les aurez compris. Si vous commencez à vous sentir débordé, arrêtez. Travaillez un peu de votre côté et revenez ensuite à ces programmes. Au fur et à mesure que vous vous poserez davantage de questions, vous trouverez de plus en plus de réponses.

SCRAMBLER

Avant d'étudier ce programme, exécutez-le pour voir ce qu'il fait. Cette présentation, de même que les suivantes, suppose que le programme est dans la machine pendant que vous lisez.

SCRAMBLER est un programme structuré facile à comprendre. Les programmes structurés sont composés de sous-programmes, qui effectuent chacun une tache déterminée. Le programme principal est au début (lignes 1000-1160). Il appelle quatre blocs de second niveau (sous-programmes) qui ressemblent à des éléments d'une chaîne haute fidélité en ce sens que chacun fait une tache spécifique.

SCRAMBLER effectue cinq taches

1. Donner des informations.
2. Accepter les premières moitiés de phrases.
3. Accepter les deuxièmes moitiés de phrases.
4. Afficher les phrases.
5. Terminer le programme.

Chacune de ces tâches a été affectée à un sous-programme distinct (ou bloc) avec un point d'entrée défini clairement. Essayez en tapant

RUN 1180

pour voir le bloc d'instruction par lui-même.

Chaque sous-programme est appelé d'un niveau plus élevé par l'intermédiaire de l'instruction GOSUB et chaque sous-programme se termine par l'instruction RETURN. (La raison de l'affichage du message ?RETURN WITHOUT GOSUB ERROR est due au fait que vous avez fait débuter l'exécution directement dans un sous-programme sans qu'il ait été appelé par une instruction GOSUB). Tout cela doit vous paraître facile car maintenant vous avez appris à écrire des sous-programmes avec GOSUB et RETURN. Cela demande un peu de réflexion car au premier abord il paraît plus facile d'écrire en mode non structuré (avec de nombreuses instructions GOTO). Dès que vous écrivez un programme sérieux atteignant 80 ou 100 lignes, vous découvrirez soudainement que vous ne pouvez plus contrôler le monstre que vous avez créé.

En réfléchissant quelques minutes à l'ensemble du programme que vous êtes en train d'écrire, vous pouvez éviter le programme du type ''spaghetti''. Quand un bloc devient si complexe que vous ne puissiez plus en avoir une vue sérieuse, vous devez le couper en plusieurs blocs plus petits.

Analyse des lignes du Programme

Les trois premières lignes de chacun des programmes présentés sont pratiquement les mêmes. La première (de numéro 0) comporte le nom donné au programme lorsqu'il a été écrit, la date où il y a été écrit et les initiales du responsable. Ces informations vous permettent d'avoir rapidement une idée du programme.

La seconde ligne (de numéro 1) contient le branchement à la routine principale qui débute toujours dans ces exemples à la ligne numéro 1000, ce qui vous laisse de la place pour ajouter plus tard des instructions, si cela vous paraît nécessaire.

La troisième ligne (de numéro 2) est un mini-programme complet. Tapez

RUN 2

puis

LIST 1350

pour voir ce qu'il fait. En supprimant les lignes d'un côté de l'écran, le programme oblige APPLESOFT à abandonner les espaces supplémentaires situés à l'intérieur des instructions PRINT : vous pouvez éditer aisément les instructions PRINT et REM. (Les espaces situés sur le côté droit de l'écran ne sont pas copiés). Chaque fois que vous souhaitez éditer le programme, frappez RUN 2, puis TEXT. (Ceci est valable pour tous les programmes de l'Annexe E).

Les lignes 1000 à 1060 sont au niveau le plus élevé du programme. En les lisant, vous pouvez comprendre exactement, ce que fait le programme et le rôle de chaque sous-programme.

Les lignes 1070 à 1160 terminent le programme. Les programmes ne doivent comporter qu'une seule fin car vous pouvez souhaiter par la suite ajouter des instructions. Ici le programmeur décida deux mois après avoir terminé son programme, d'inclure la ligne 1155 qui renvoie l'utilisateur au Menu. Comme ce programme n'a qu'une seule sortie, il n'a pas été nécessaire de relire le programme en entier pour chercher d'autres instructions END avant de faire l'adjonction.

Les lignes 1180 à 1390 initialisent le programme et affichent les directions.

La ligne 1200 doit être incluse dans tous vos programmes qui utilisent un écran 40 colonnes. (Le programme suivant, MAGIC MENU, a une ligne similaire pour un écran de 80 colonnes). Vous ne pouvez jamais être sûr que l'utilisateur exécutera votre programme avec le mode écran nécessaire. Cette ligne a également pour effet d'annuler les conséquences de la commande RUN 2. Ainsi il n'est pas nécessaire de frapper TEXT avant d'exécuter le programme.

La ligne 1210 sert à dimensionner les deux tableaux de chaînes utilisés pour stocker les premières et secondes moitiés de phrases. De l'espace est alloué pour 1000 éléments de chaque chaîne, ce qui fait qu'un utilisateur aura besoin d'un temps de 6 à 8 heures pour les remplir tous et provoquer une erreur. Nous espérons que peu d'utilisateurs seront passionnés à ce point par ce jeu de mélange de phrases.

Des mots anglais (comme TITLE$ en ligne 1220) sont utilisés comme nom de variable ''chaîne de caractères'' afin de les rendre plus faciles à identifier. Utilisez des noms mnémoniques pour les variables, qui sont peu pratiques, mais n'oubliez pas qu'Applesoft ne considère que les deux premiers caractères de chaque nom de variable. (Le programme suivant contient davantage de noms de variables).

Les lignes 1400 à 1540, sous-programme de second niveau acceptent les premières demi-phrases. Ce sous-programme fonctionne de la façon suivante : l'utilisateur rempli le tableau FIRST$ avec des chaînes et mémorise dans la variable F le nombre d'éléments remplis.

Les variables compteurs gardent trace ; vous les avez déjà utilisées dans les boucles FOR/NEXT. Une bonne règle à suivre pour décider si vous devez utiliser les instructions IF...THEN ou FOR/NEXT est la suivante

- Quant l'utilisation d'une variable compteur est déterminée (quand vous savez, par exemple, que vous voulez compter de 1 à 10 puis vous arrêter), utiliser la boucle FOR/NEXT.

- Quand l'utilisation d'une variable compteur est indéterminée (quand vous ne connaissez pas le nombre exact), comme dans SCRAMBLER, utilisez l'instruction IF...THEN.

Les lignes 1400 a 1470 donnent des instructions à l'utilisateur. La ligne 1490 détermine une fenêtre avec défilement, afin que les entrées de l'utilisateur ne provoquent pas un défilement des instructions qui les feraient sortir de l'écran.

Les lignes 1500 à 1530 constituent une boucle : tant que l'utilisateur n'appuie pas sur les touches [Ctrl] - [↵], F est incrémenté d'une unité et l'utilisateur peut introduire une nouvelle chaîne.

La ligne 1510 teste rapidement pour savoir si l'utilisateur a appuyé sur la touche [Ctrl]. Pour cela, elle teste le ''switch'' immédiatement après que [↵] a été appuyé dans l'instruction INPUT de la ligne 1500.

La ligne 1520 s'assure que l'utilisateur n'a rien frappé avant que la variable F n'ait été incrémentée. Cela évite que les lignes blanches soient introduites dans les demi-phrases. Essayez en changeant la ligne 1520

```
1520 F = F + 1
```

pour voir si le programme fonctionne sans ce test d'erreur.

Les lignes 1600 à 1720 donnent la seconde moitié des phrases. Ce sous-programme fonctionne comme le précédent. Seule la ligne message diffère.

Les lignes 1800 à 1950 contiennent les sous-programmes de second niveau qui combinent les demi-phrases de façon aléatoire et affichent les phrases ainsi obtenues à l'écran.

La ligne 1820 s'assure qu'il y a des demi-phrases à combiner. Sinon, elle renvoie au programme principal. Habituellement l'absence de phrases signifie que l'utilisateur désire quitter le programme.

La ligne 1830 utilise l'instruction TEXT pour supprimer la fenêtre placée à la ligne 1660 de telle façon que l'instruction HOME efface l'écran tout entier.

Les lignes 1910 à 1930 choisissent et affichent une phrase aléatoire. La ligne 1910 choisit les variables de phrase FF et SS à partir du nombre total de demi-phrases F et S. La ligne 1930 affiche alors des demi-phrases séparées par un espace.

Réglage Fin

SCRAMBLER a deux défauts qui ont été laissés de côté pour que vous travaillez dessus :

1. Les instructions standard INPUT d'Applesoft n'acceptent pas des virgules ou des ''deux points'' : essayez d'introduire une phrase comportant l'un de ces caractères. Deux façons permettent de résoudre le problème de ce type : prévenir l'utilisateur ou résoudre le problème. Dans le premier cas, il vous suffit de prévenir l'utilisateur de ne pas frapper des virgules ou des ''deux points'' à l'intérieur des phrases.

   Le programme MAGIC MENU vous permettra d'apprendre à construire un sous-programme d'entrée qui résoud ce problèm et comment l'introduire dans des programmes du type SCRAMBLER. Cette amélioration est plus difficile mais donne un programme plus commode pour l'utilisateur.

2. Si l'utilisateur frappe des espaces au début ou à la fin des demi-phrases, les deux demi-phrases seront mises de côté (essayez !). Cela peut être évité en supprimant les espaces inutiles au début et en queue, tâche complexe qui utilise les fonctions chaînes LEFT$ et RIGHT$. Comme précédemment, un moyen simple, tant que vous ne disposez pas d'une telle fonctionnalité dans Applesoft, est de prévenir l'utilisateur au moyen d'un message.

Essayez maintenant d'ajouter ces messages, à l'intention de l'utilisateur, après la ligne 1360. Si les instructions ajoutées sont trop longues, remplacez l'instruction VTAB 4 de la ligne 1230 par un simple PRINT. Quand vous serez certains que le programme fonctionne correctement, sauvez le sur disque. Plus tard, vous souhaiterez revenir au programme SCRAMBLER afin d'essayer d'autres méthodes avancées pour résoudre ces problèmes.

Programme

```
0  REM SCRAMBLER - SEPT, 1982 BY BG & TOG
1  GOTO 1000: REM BEGINNING OF PROGRAM
2  TEXT : PRINT  CHR$ (21): POKE 33,33: HOME : END
1000  REM THE SCRAMBLER - A SENTENCE SCRAMBLING GAME - 1982
1010  GOSUB 1180: REM SET UP PROGRAM & DISPLAY DIRECTIONS
1020  GOSUB 1400: REM INPUT 1ST HALF OF SENTENCES
1030  GOSUB 1600: REM INPUT 2ND HALF OF SENTENCES
1040  GOSUB 1800: REM DISPLAY SENTENCES
1050  REM    O.K., WE'RE DONE NOW, SO...
1060  REM
1070  REM *** END OF PROGRAM ***
1080  REM
1090  REM NOTE: PROGRAMS SHOULD HAVE ONLY 1 EXIT
1100  TEXT : HOME : VTAB 12
1110  PRINT "Would you like to play again (Y or N)? ";: GET IN$
1120  IF IN$ = "Y" OR IN$ = "y" THEN  TEXT : HOME : GOTO 1020
1130  IF IN$ <  > "N" AND IN$ <  > "n" THEN 1100
1140  TEXT : HOME : VTAB 22
1150  PRINT "Thanks for playing...."
1155  IF  PEEK (6) = 99 AND  PEEK (7) = 99 THEN  PRINT  CHR$ (4);"RUN HEL
    LO": REM SEE NOTES FOLLOWING LINE 9060 IN THE DISK MENU PROGRAM
1160  END
1170  REM
1180  REM *** DIRECTIONS ***
1190  REM
1200  PRINT  CHR$ (21): TEXT : HOME : REM   TURN OFF APPLE'S 80-COLUMN TE
    XT CARD, SELECT TEXT, AND CLEAR THE SCREEN
1210  DIM FIRST$(999),LAST$(999):F = 0:S = 0: REM GIVE THE USER 1000 1ST
    AND 2ND STRINGS
1220 TITLE$ = "*** THE SCRAMBLER ***": HTAB 21 -  LEN (TITLE$) / 2: PRINT
    TITLE$
1230  VTAB 4
1240  PRINT "The scrambler is a word game.  You will"
1250  PRINT "be asked to type the first halves of"
1260  PRINT "sentences such as 'Edwin cooked'.": PRINT
1270  PRINT "You will then be asked for the second"
1280  PRINT "halves of sentences such as 'a banana"
1290  REM
1300  PRINT "squash.'": PRINT
1310  PRINT "Type as many as you like; then command"
1320  PRINT "the computer to combine random halves"
1330  PRINT "into humorous sentences.": PRINT
1340  PRINT "The hilarity rises with the number of"
1350  PRINT "participants: what seems dull to you": PRINT "alone will be
    found hysterically funny"
1360  PRINT "by a group of 10 otherwise normal people";: PRINT "-- a stra
    nge phenomenon."
1370  VTAB 23
1380  INPUT "Press the RETURN key to begin.";IN$
1390  RETURN
1399  REM
```

```
1400  REM *** GET 1ST HALVES ***
1410  REM
1420  HOME : HTAB 9
1430  PRINT "The first half..."
1440  VTAB 20
1450  PRINT "Type the first halves of sentences."
1460  PRINT "Press RETURN after each entry."
1470  PRINT "Hold down OPEN-APPLE and press the": PRINT "RETURN key after
      your last entry.";: REM ALWAYS USE A SEMI-COLON AFTER PRINTS ON BOT
      TOM LINE
1480 F = 0:OAK = 0: REM SET COUNTER TO 0: SET OPEN-APPLE KEY VARIABLE TO
      0
1490  POKE 34,2: POKE 35,18: HOME : REM SET SCROLLING WINDOW
1500  INPUT FIRST$(F)
1510  IF  PEEK ( - 16287) > 127 THEN OAK = 1: REM CHECK QUICKLY TO SEE IF
      USER IS PRESSING THE OPEN-APPLE KEY
1520  IF  LEN (FIRST$(F)) > 0 THEN F = F + 1: REM ADVANCE COUNTER IF A SU
      CCESSFUL ENTRY
1530  IF OAK = 0 THEN 1500: REM IF OPEN-APPLE-RETURN NOT PRESSED, LOOP BA
      CK FOR THE NEXT HALF-SENTENCE
1540  RETURN
1550  REM
1600  REM *** GET 2ND HALVES ***
1610  REM
1620  TEXT : VTAB 1: HTAB 9
1630  PRINT "... the second half"
1640  VTAB 20
1650  PRINT "Type the second halves of sentences."
1660 S = 0:OAK = 0: POKE 34,2: POKE 35,18: HOME : REM SET UP SECOND HALF
      SAME WAY AS FIRST
1670  INPUT LAST$(S)
1680  IF  PEEK ( - 16287) > 127 THEN OAK = 1: REM CHECK QUICKLY TO SEE IF
      USER IS HOLDING THE OPEN-APPLE KEY
1690  IF  LEN (LAST$(S)) > 0 THEN S = S + 1:
1700  IF OAK = 0 THEN 1670
1710  RETURN
1720  REM
1800  REM ** DISPLAY SENTENCES **
1810  REM
1820  IF F = 0 OR S = 0 THEN  RETURN : REM IF THERE IS NOT AT LEAST 1 COM
      PLETE SENTENCE, CANCEL
1830  TEXT : HOME : HTAB 9
1840  PRINT "Scrambled sentences:"
1850  VTAB 20
1860  PRINT "Press RETURN for a new sentence."
1870  PRINT "Press OPEN-APPLE-RETURN to end."
1880 OAK = 0: POKE 34,2: POKE 35,18: HOME
1890  VTAB 17: INPUT IN$
1900  IF  PEEK ( - 16287) > 127 THEN  RETURN : REM IS USER PRESSING OPEN-
      APPLE? YES, THEN RETURN
1910 FF =  RND (1) * F:SS =  RND (1) * S: REM SELECT RANDOM SENTENCE
1920  VTAB 17
1930  PRINT FIRST$(FF);" ";LAST$(SS): REM AND PRINT IT
1940  PRINT : PRINT
1950  GOTO 1890
```

MAGIC MENU

Une règle bien connue des informaticiens est que 10% d'un programme occupe 90% du temps machine. Supposons que vous écriviez un programme pour l'établissement de prévisions de marché pour certains produits oléagineux. Il ne vous faut pas plus de deux heures pour écrire le programme, sans inclure les instructions pour l'utilisateur. Vous savez quand il faut introduire le prix de l'huile de grain. Ainsi tout apparaît comme une série de ''curseurs clignotants''.
Malheureusement M. DUBOIS, votre supérieur hiérarchique souhaite utiliser votre travail et avoir une copie qu'il puisse utiliser sur son Apple II Plus. Il vous faut à nouveau 3 ou 4 jours pour rendre le programme utilisable par M. DUBOIS. Quoi faire ? Utiliser MAGIC MENU.

MAGIC MENU contient 5 blocs de sous-programmes qui ensemble font attention au temps consommé, et ce qui est le plus important, rendent votre programme utilisable par d'autres (''d'autres'' signifie également vous-même si vous reprenez le programme 2 mois sans l'avoir utilisé et que vous essayez à nouveau de le faire marcher alors que vous ne vous rappelez plus ce que signifie le curseur clignotant).

Les blocs sont intelligents. Par exemple, la subroutine Computer Identifier reconnaît la version de l'Apple II utilisée, ainsi les autres subroutines peuvent tirer parti des caractéristiques de la machine. Cela vous permet d'utiliser à la fois les minuscules et les majuscules dans votre programme même s'il doit être exécuté sur un Apple II ou un Apple II Plus (les versions anciennes de l'Apple II ne disposent pas de minuscules à partir du clavier).

Le code formé par chacun des blocs est très complexe mais cela ne doit pas vous empêcher de les utiliser. Ces blocs peuvent être considérés comme des ''boîtes noires'' : vous leur donnez l'entrée qu'elles désirent et elles font leur travail correctement. Il n'est pas nécessaire que vous compreniez comment la radio fonctionne pour trouver votre station favorite, de la même façon ces blocs de subroutines exécutent des taches très complexes dont vous n'aurez pas à vous encombrer.

Les paragraphes suivants sont concentrés exclusivement sur l'utilisation de ces blocs et pas sur leur fonctionnement. Vous trouverez à la fin de MAGIC MENU quelques conseils sur la façon de construire une bibliothèque pour vos boîtes noires favorites ainsi qu'une discussion sur les noms de variable et la façon de les choisir.

Note pour les programmeurs confirmés

MAGIC MENU, ainsi que les deux programmes suivants, utilisent une version Applesoft compatible avec l'interface standard Apple *II*e.

Les instructions REM disséminées dans le programme MAGIC MENU donnent de nombreuses informations relatives à la façon d'utiliser la carte 80 colonnes, les instructions HTAB et VTAB et de lire la position du curseur.

Fonctionnement des cinq sous-programmes : démonstration

Ce paragraphe utilise des exemples, que vous devez essayer sur votre ordinateur. Ils montrent le fonctionnement des cinq sous-programmes (boîtes noires) et comment ils sont liés dans MAGIC MENU. Les nouvelles lignes que vous entrez vous permettent de considérer chaque sous-programme comme un bloc isolé d'instructions.

Si vous n'avez pas encore exécuté MAGIC MENU, faites-le maintenant. Ensuite supprimez tout le programme en mémoire à l'exception de l'ensemble des sous-programmes et abstenez-vous d'exécuter Computer Identifier qui débute à la ligne 63000, frappez

```
DEL 1000, 62999
2000 END
```

Maintenant, pour vous assurer que tout est normal, frappez

```
RUN 2
TEXT
```

Cela doit mettre la machine dans le mode 40 colonnes et effacer l'écran. Les lignes 0 à 2 de MAGIC MENU ressemblent beaucoup aux lignes 0 à 2 des autres programmes de cette Annexe E. Leur rôle est expliqué en détail à propos du programme SCRAMBLER. La ligne 1 reste parce que tous les exemples donnés débutent à la ligne 1000. Elle vous permet de frapper RUN au lieu de RUN 1000.

Le sous-programme Input

Les lignes 100 à 299 constituent le sous-programme INPUT qui constitue une extension de l'instruction

INPUT '' ''; AN$

Tout message, tel que Voulez vous la réponse en dollars ? (O, N) doit au préalable avoir été donné. Vous pouvez indiquer le nombre maximum de caractères que l'utilisateur peut introduire en initialisant la variable FL (FL pour ''Field Length'') avec un nombe compris entre 1 et 250. (Même avec une réponse O/N comme ci-dessus, il est prudent de donner à l'utilisateur un peu d'espace pour s'exprimer. Si vous savez que l'utilisateur souhaitera probablement une réponse en dollars, vous pouvez mettre la réponse dans AN$; dans ce cas AN$ = ''O''. Alors il suffit que l'utilisateur appuie sur la touche RETURN pour dire ''Oui''. (Les utilisateurs d'Apple II et Apple II plus ne pourront pas donner cette réponse implicite.

Ce sous-programme INPUT, exécuté sur un Apple *IIe* prendra en compte l'appui sur l'une des touches suivantes : Esc ☌ et . Si Esc est appuyée ou si l'une des touches ☌ ou est appuyée en même temps qu'une autre touche le sous-programme donnera : Esc = 1 si la touche Esc est appuyée, OAKEY = 1 si ☌ est appuyée, SAKEY = 1 si la touche est appuyée. AN$ ne prendra pas en compte ce que l'utilisateur frappe avant d'appuyer sur l'une de ces touches ; et OAKEY$ ou SAKEY$ contiendront la touche appuyée en même temps que ☌ ou respectivement.

Vous pouvez ignorer toutes ces possibilités. (Voici un exemple, frappez le et essayez le !)

```
1000 AIIE = 1 : HOME
1010 PRINT ''Do you like margarine?'';
1020 AN$ = '' '' : FL = 0 : GOSUB 100 : IF ESCKEY <> 0
     OR OAKEY <> 0 OR SAKEY <> 0 THEN 1000
1030 PRINT AN$
RUN
```

Cet exemple n'utilise pas les possibilités particulières du sous-programme INPUT, c'est-à-dire longueur de champs et reconnaissance des touches spéciales. Mais même en ignorant ces possibilités, vous serez capable d'utiliser le curseur clignotant souligné et les utilisateurs pourront frapper les caractères virgule et ''deux points''.

La ligne 130 liste toutes les variables utilisées (et pouvant être modifiées par le sous-programme). Toutes les instructions REM des bonnes boîtes noires font cela.

Le sous-programme Get Return

Les lignes 300 à 399 constituent le sous-programme ''touche appuyée'' du programme. elles font effectuer à l'Apple *II*e une boucle jusqu'à ce que l'utilisateur appuie sur ↵. Il existe de nombreux endroits, dans un programme, où vous souhaitez dire quelque chose du type ''Press ↵ to see the Oleomargarine Futures'' et ensuite attendre jusqu'à ce que la touche ↵ soit appuyée. Ce sous-programme fait ce travail en fournissant le même curseur que le sous-programme INPUT.

Pour le voir fonctionner, frappez simplement

```
DEL 1000, 1030
1000  PRINT ''Please press RETURN.'' ;
1010  GOSUB 300
RUN
```

Dans cet exemple, vous ne donnez aucune information et aucune n'est renvoyée. Il changera les valeurs de I, J, K, L et P. (La ligne 1010 doit être supprimée car elle était utilisée dans l'exemple du sous-programme INPUT).

Le sous-programme Screen Formatter

Les lignes 400 à 499 constituent le sous-programme Screen Formatter. Cette boîte effectue un travail équivalent à

```
400 PRINT AN$ ; '' '' ;  : RETURN
```

Mais de façon très intelligente.

D'abord il ne coupe jamais un mot en deux parties éditées sur deux lignes consécutives : il cherche à partir de la fin de ligne, le premier espace et coupe la phrase à cet endroit, la suite étant éditée sur la ligne suivante. Cela vous évite de perdre un temps important à rédiger votre mode d'emploi de telle façon que les mots ne soient pas coupés en fin de ligne. Il vous suffit de mettre votre texte dans AN$ et d'appeler ce sous-programme.

Ensuite, ce sous-programme est actif pendant l'exécution du programme. Si l'ordinateur dispose d'une carte 80 colonnes, il formatera correctement le texte pour 80 colonnes.

Enfin, si le programme est exécuté sur un Apple II ou Apple II Plus, Screen Formatter convertira toutes les minuscules que ces ordinateurs ne peuvent pas afficher en majuscules. Ainsi votre programme tout entier tire parti des possibilités de votre nouvel Apple *IIe* sans qu'il soit limité seulement à l'Apple *IIe*.

Le sous-programme suppose que AN$ contient le texte que vous souhaitez afficher. Il sera affiché en majuscules et minuscules si le sous-programme Computer Identifier a placé AIIE à vrai (AIIE = 1). Il formatera en mode 80 colonnes si vous avez donné à COL80 la valeur vraie (COL80 = 1). Plus d'informations dans la ligne 1025. Essayez cet exemple :

```
1000  AIIE = 1
1010  AN$ = ''Margarine used to taste horrible
      and came with the coloring in a separate
      pouch. It has improved considerably. Is it good enough now
      to compete with the $4.98
      spread?''
1020  GOSUB 400
1030  GOSUB 100
```

Après l'exécution dans ce mode, modifiez la ligne 1000 par

```
1000 AIIE = 0
```

et exécutez-le à nouveau. De cette façon votre ordinateur ressemblera à un Apple II ou un Apple II Plus : lent mais utilisable. Dans ces modèles, le sous-programme doit tester si chaque caractère (après avoir trouvé le 40e caractère et recherché le premier espace pour couper le texte) est une minuscule. Si tel est le cas, une formule complexe est utilisée pour la transformer en majuscule afin de permettre un affichage correct (voir ligne 445). Notez aussi que lister de telles lignes sur un Apple II ou Apple II Plus provoque l'affichage des minuscules comme des ''ordures''. Le transcodage ne fonctionne que lors de l'exécution du programme.

Le sous-programme Menu Maker

Les lignes 500 à 899 contiennent le quatrième bloc appelé Menu Maker. Ce sous-programme crée un menu correct en quelques minutes au lieu de jours. Frappez

```
1000  AIIE = 1
```

et indiquez le nom du programme par :

```
1010 TITLE$ = ''Oleomargarine Futures Forecaster''
```

Le nom du menu est

```
1020 SUBTITLE$ = ''Type of Margarine''
```

Le choix proposé par le menu est le suivant

```
1030 MENU$(1) = ''Corn Oil''
1040 MENU$(2) = ''Sunflower Seed Oil''
1050 MENU$(3) = ''Cottonseed Oil''
1060 MENU$(4) = ''Crude Oil''.
1070 MENU$(5) = ''End the Program''
```

Pour indiquer à Menu Maker que vous avez terminé, faites :

```
1080 MENU$(6) = ''END''.
```

Il acceptera aussi ''End'' ou ''end''. En 1090, la chaîne FROM$ sera mise à zéro car il n'y a pas de menu secondaire dans cet exemple. Cependant, dans le cas d'un programme à plusieurs menus, la variable FROM$ contient le nom du menu d'où vient l'utilisateur. Quand FROM$ n'est pas nul, la touche Esc est testée et le message suivant apparaît : To return to the main menu, press the Esc key.

```
1090 FORM$ = '' ''
```

Vous pouvez aussi proposer une explicitation des différents choix du menu dans votre programme. MAGIC MENU peut le faire. Dans notre cas, il n'y a pas de tableaux de description, aussi OAKEY reçoit la valeur logique ''faux'' (OAKEY = 0).

```
1100 OAKEY = 0
```

Enfin ajoutez

```
1110 GOSUB 500

1120 PRINT AN, AN$
```

Vous pouvez constater que le numéro du choix sélectionné dans le menu est placé à la fois dans AN et AN$. Il vous a fallu seulement quelques minutes pour composer ce menu. Maintenant tapez RUN.

C'est Magique !

Bien sûr, toutes les options termineront le programme car vous affichez simplement AN, mais ne les utilisez pas. Votre programme pourrait aussi se brancher sur différents sous programmes traitant des différentes huiles. Référez-vous, pour plus de renseignements, à l'étude de la ligne 1670 dans la section ''Autres Remarques sur MAGIC MENU''.

Le sous-programme Computer Identifier

A présent vous pouvez essayer le cinquième et dernier bloc, Computer Identifier. La ligne 1000 est la suivante pour l'instant :

```
1000 AIIE = 1
```

Pour que Computer Identifier puisse trouver quelle machine vous utilisez, faites :

```
1000 GOSUB 63000
```

AIIE recevra la valeur 1 si le programme tourne sur Apple *//*e et 0 sur Apple II ou Apple II Plus. La variable RESULTS peut recevoir les valeurs suivantes :

- 0 pour Apple II ou II Plus
- 32 pour Apple *//*e
- 64 pour Apple *//*e avec une carte texte Apple *//*e 80 Colonnes
- 128 pour Apple *//*e avec une carte texte Extension 80 Colonnes

Afin d'utiliser les Cartes Texte Apple *//*e 80 Colonnes ou Extension 80 Colonnes et de permettre aux sous programmes Menu Maker et Screen Formatter de savoir s'il y a une telle carte, ajoutez :

```
1005 IF RESULTS >= 64 THEN COL80 = 1 : PRINT
     CHR$(4) ; ''PR#3''
```

Faites exécuter le programme. Si vous n'avez pas de carte, rien ne se passera. Si vous en avez une, votre écran affichera 80 colonnes. Pour revenir en 40 colonnes, tapez simplement

```
RUN 2
```

Autres remarques sur Magic menu

Pour recharger le programme, tapez

LOAD MAGIC MENU

Vous avez exploré toutes les ''boîtes noires'' ; maintenant regardons les lignes du programme construit dessus. Il y a très peu de difficultés dans ces lignes ; pour l'essentiel, vous pouvez regarder les listings et exécuter le programme pour comprendre ce qui se passe. Certaines lignes, par contre, qui demandent des commentaires, sont étudiées dans ce chapitre.

Ligne 1010 : Applesoft place chaque variable rencontrée pendant l'exécution du programme dans une table de variables contenant les valeurs actuelles de ces variables. Chaque fois qu'un programme consulte ou modifie la valeur d'une variable, Applesoft examine la table vérifiant chaque variable après l'autre. En déclarant en premier les variables les plus utilisées, vous pouvez augmenter la vitesse du programme.

Ligne 1360 : les utilisateurs ayant une carte texte 80 colonnes peuvent avoir sur leur écran davantage de texte que ceux ayant 40 colonnes. Le programme en profite pour offrir davantage d'informations pour les utilisateurs ayant 80 colonnes.

La ligne 1640 traite les demandes d'explications.

La ligne 1645 traite une demande de retour à DISK MENU.

La ligne 1670 envoie l'utilisateur à la sous section de programme demandée. ON AN GOSUB 2000, 3000, 4000 ira à un sous programme commençant à la ligne 2000 si AN = 1 ; 3000 si AN = 2, 4000 si AN = 3,... Remarquez, quand vous utilisez ON GOSUB, il n'y a pas de relation logique entre le contenu de la variable et le sous programme vers lequel elle envoie l'exécution. MENU MAKER n'autorise pas des nombres non prévus, aussi une vérification d'erreur dans AN est déjà faite.

Les lignes 1800 à 1890 sont très semblables aux lignes ci dessus. Elles traitent des demandes d'explication et branchent aux sous sections appropriées.

La ligne 2010 effectue le tri des vérifications d'erreurs dont vous avez besoin pour que votre programme soit universel. (Vous pouvez tester votre programme en mode Apple II Plus, pour cela il suffit d'affecter la valeur 0 à la variable AIIE au lieu d'utiliser le sous programme Computer Identifier.)

Vitesse du programme

Quand Applesoft exécute un programme, il lit chaque numéro de ligne en séquence. Plus un programme est long, plus sa durée d'exécution est grande ; un élément de tableau prend plus de place à trouver qu'une variable simple.

Applesoft exécute des sous programmes d'autant plus vite que le numéro des lignes est petit. Aussi, il vaut mieux réserver les petits numéros aux sous programmes qui doivent être exécutés souvent dans le minimum de temps. Les programmes principaux sont plutôt vers la fin car il sont appelés une seule fois.

Pour mieux voir la différence dans les temps d'exécution, comparez les deux petits programmes suivants :

```
10 I = I + 1 : IF I < 300 THEN 10
20 STOP
RUN 10.
```

et

```
50000 I = I + 1 : IF I < 300 THEN 50000
50010 STOP
RUN 50000
```

Quelques mots sur les noms de variables

Devenu familier avec les boîtes noires et leur fonctionnement, vous voudrez sans doute en faire vous-même. Les lignes ci-dessus sont justement là pour vous aider.

Il vaut mieux en général utiliser des noms de variables descriptifs pour rendre le programme plus lisible : AIIE = 1 vous dit quelque chose, pas L3 = 1.

Souvent, la vitesse d'exécution de vos programmes est un résultat de votre programmation. Si vous utilisez peu de variables, déclarées tôt dans le programme et utilisées dans tous les sous programmes rapides, votre programme ira beaucoup plus vite, jusqu'à 3 ou 4 fois.

Heureusement, il y a des repères, basés sur des conventions remontant à 1950 pour les variables de grande vitesse. 8 noms de variables standard et non descriptifs sont utilisés pour accélérer l'exécution. Bien entendu, elles échappent à la règle selon laquelle il faut utiliser des noms descriptifs. Ce sont :

I, J, K, L, M, N, O, P

I est la plus commune de toutes, puis J, etc... On utilise souvent P pour la fonction PEEK, pour tester le clavier et autres éléments hardware.

Il vaut mieux utiliser ces variables dans un sous programme et pas entre deux sous programmes. Dans ce cas là, prendre plutôt des noms qui signifient quelque chose.

Si vous voulez appeler un sous programme de niveau inférieur (ex : sous programme INPUT) tout en ayant une exécution ultra rapide dans une routine de niveau supérieur (exemple : Menu Maker), utilisez la lettre en double dans le sous programme de plus haut niveau :

II, JJ, KK, LL, MM, NN, OO, PP

Cela empêchera le sous programme de niveau inférieur de détruire des valeurs dans celui de niveau supérieur. Par exemple, Menu Maker n'utilise que des variables double lettre parce qu'il utilise à la fois les sous programmes INPUT et Screen Formatter.

Assurez-vous grâce à des instructions REM des variables que vous affectez à ces sous-programmes.

Finalement, n'utilisez pas ces variables à moins d'avoir vraiment besoin de la vitesse si vous ne voulez pas transformer vos sous programmes favoris en boîtes noires. Vous devez pouvoir lire et comprendre votre programme et, pour cela, utilisez au maximum des noms descriptifs.

En suivant ces repères dans la création des 5 blocs sous programmes dans ce programme, l'auteur a permis aux programmeurs l'utilisant la plus grande facilité de construction de leurs propres programmes.

Quelques remarques sur la logique

Le BASIC Applesoft utilise une logique appelée Booléenne, basée sur les valeurs faux, vrai, 0 et 1.

Quand vous dites

```
X = 0 : IF A = 23 THEN X = 14
```

vous demandez à Applesoft de voir si A = 23 est vrai,si c'est vrai il exécutera les instructions suivantes. On peut faire la même chose d'une autre manières :

```
X = 14 * (A = 23)
```

C'est un peu plus difficile à croire. Essayez en exécution immédiate en donnant d'abord à A la valeur 23, puis en rendant A égal à quelque chose d'autre que 23.

Tapez

```
A = 23
X = 14 * (A = 23)
PRINT X.

A = - 32.68
X = 14 * (A = 23)
PRINT X.
```

Quand Applesoft rencontre un argument Booléen (A = 23) il regarde si c'est vrai (1) ou faux (0). (14 * 0) vaut 0 et (14 * 1) vaut 14. (Vous pouvez étudier la logique Booléenne plus en détail dans la section sur IF... THEN du *Manuel de Référence Applesoft).*

MAGIC MENU utilise beaucoup de drapeaux vrai ou faux pour communiquer entre des programmes. Un drapeau est une variable dont la valeur (1 ou 0) indique si une conditions est vraie ou si un événement s'est produit. Cela permet de contrôler les effets des programmes à n'importe quel moment.

Par un drapeau, Computer Identifier communique au programme Screen Formatter que le programme tourne sur Appel *IIe* (AIIE = 1 Alors Screen Formatter peut prendre des décisions basées sur IF AIIE = 1 THEN ou IF AIIE THEN. Le programme INPUT vous avertit que la touche [Ô] a été enfoncée, en donnant à l'étiquette OAKEY la valeur vraie. Vous pouvez alors aller plus loin et voir ce qu'il y a en OA$ pour trouver quelle clé a été enfoncée.

Une bonne habitude est de déclarer les deux noms de variables TRUE et FALSE en tête du programme :

```
1000 FALSE = 0 : TRUE = 1
```

Ainsi, vous pouvez voir plus clairement ce que vous faites :

```
1010 AIIE = TRUE
1020 IF AIIE = TRUE THEN PRINT ''TRUE''
1030 IF AIIE = FALSE THEN PRINT ''FALSE''.
```

Beaucoup de jeunes programmeurs évitent les étiquettes et préfèrent d'autres types de test. Il y a quelques années, un programmeur Apple voulait stocker l'âge de l'utilisateur dans la variable YEARS. Son sous programme pour le faire demandait d'abord à l'utilisateur son âge en années et mois. Puis, il divisait les mois par 12, les ajoutait aux années et finalement prenait l'entier YEARS + .5 pour l'arrondir à l'année la plus proche :

```
YEARS = INT (YEARS + .5) : RETURN
```

Quand le sous-programme se terminait, le programme qui l'avait appelé devait savoir si l'utilisateur avait bien répondu ou simplement appuyé ↵. Le programmeur découvrit qu'il pouvait l'indiquer en testant simplement la variable YEARS :

1000 IF YEARS THEN (continuez le programme)

Cela marcha très bien jusqu'à ce qu'un collège tapat l'âge de son bébé : 0 an, 2 mois. Le programme arrondit la réponse à 0. Et il continua de demander sans arrêt quel âge avait Betty.

Vous n'épuiserez pas les noms de variables il y en a plus de 5000. Utilisez une étiquette et donnez lui un nom descriptif.

Programme

```
0  REM MAGIC MENU - SEPT, 1982 BY TOG
1  GOTO 1000
2  TEXT : PRINT  CHR$ (21): HOME : POKE 33,33: END
100  REM *** INPUT ROUTINE ***
102  REM SEPT, 1982 BY B. TOGNAZZINI
105  REM USES FLASHING UNDERLINE CURSOR
110  REM YOU MAY SET THE NUMBER OF CHARACTERS THE USER MAY TYPE (THE FIEL
     D LENGTH) IN THE VARIABLE FL
115  REM  A FIELD LENGTH OF 0 (FL=0) ALLOWS THE MAXIMUM INPUT
120  REM IF YOU WANT A "DEFAULT" (SUPPLIED-BY-YOU) ANSWER, PUT IT IN AN$.
      FOR EXAMPLE: AN$= "CAT"
124  REM UPON RETURN, AN$ (ANSWER$) WILL CONTAIN THE INPUT FROM THE USER
125  REM ESCKEY WILL BE SET TO TRUE (WILL EQUAL 1) IF THE USER PRESSED IT
      TO EXIT THE INPUT
126  REM SAKEY OR OAKEY WILL BE SET TO TRUE (WILL EQUAL 1) IF THE USER PR
     ESSED ANY CHARACTER KEY WHILE HOLDING THE SOLID-APPLE OR OPEN-APPLE
     KEY, RESPECTIVELY
127  REM ABOVE 3 KEYS ARE ONLY READ IN AN APPLE IIE: PROVIDE SOME OTHER M
     ETHOD OF SIGNALLING FOR AII AND AII+ OWNERS
130  REM THE ROUTINE USES (AND MAY CHANGE THE VALUES IN) I,J,K,L,M,P,FL,E
     S (FOR ESCAPE KEY),OA (FOR OPEN-APPLE KEY),SA (FOR SOLID-APPLE KEY),
     FL,I$,OA$,SA$,AN$
135  REM * TAKE INPUT *
140 I = 5:J = 0:K = 0:L = 0:M = 0:I$ = "":ESC = 0:OA = 0:OA$ = "":SA = 0:
     SA$ = "": IF FL = 0 THEN FL = 245
142  IF  NOT AIIE THEN  INPUT "";AN$: RETURN
145  PRINT AN$;:J =  LEN (AN$)
149 M =  PEEK (37)
153 L =  PEEK (36): IF COL80 THEN  IF L =  PEEK (1147) THEN L =  PEEK (14
     03): REM FIND CURRENT HORIZ POSITION WITH 80 COLUMN CARD TURNED ON
155  PRINT " ";I$;" ";
160 N = 1: IF L +  LEN (I$) >  =  PEEK (33) - 3 AND  PEEK (37) =  PEEK (3
     5) - 1 THEN N = 0
165  POKE 36,L: POKE 1403,L: VTAB M + 1
170 I = I - 1: IF I < 0 THEN K = 1 - K:I = 5: PRINT  CHR$ (32 + 63 * K);
175  POKE 36,L: POKE 1403,L: VTAB M + 1
180 P =  PEEK ( - 16384): IF P < 128 THEN 170
185  IF  PEEK ( - 16287) > 127 THEN OAKEY = 1
190  IF  PEEK ( - 16286) > 127 THEN SAKEY = 1
195  POKE  - 16368,0:K = 0:I = 0
```

```
200  IF OAKEY THEN OA$ =  CHR$ (P - 128):AN$ = AN$ + I$: PRINT I$;" ": RETU
     : REM * OPEN-APPLE KEY
205  IF SAKEY THEN SA$ =  CHR$ (P - 128):AN$ = AN$ + I$: PRINT I$;" ": RET
     : REM * SOLID-APPLE KEY
210  IF P > 159 AND P <  > 255 THEN  IF J +  LEN (I$) < FL THEN  IF N THEN
     AN$ = AN$ +  CHR$ (P - 128):J = J + 1: PRINT  CHR$ (P);: GOTO 149
215  IF P <  > 255 THEN 240: REM DELETE KEY
220  IF J THEN  PRINT " ";: POKE 36,L: VTAB M + 1: PRINT  CHR$ (136);:J =
     J - 1
225  IF J = 0 THEN AN$ = ""
230  IF J THEN AN$ =  LEFT$ (AN$,J)
235  GOTO 149
240  IF P <  > 136 THEN 265: REM * BACK ARROW KEY
245  IF J THEN  PRINT " ";: POKE 36,L: VTAB M + 1: PRINT  CHR$ (136);:I$ =
      RIGHT$ (AN$,1) + I$:J = J - 1
250  IF J = 0 THEN AN$ = ""
255  IF J THEN AN$ =  LEFT$ (AN$,J)
260  GOTO 149
265  IF P = 141 THEN AN$ = AN$ + I$: PRINT I$;" ": RETURN : REM * RETURN
     KEY
270  IF P <  > 149 THEN 294: REM * FORWARD ARROW KEY
275  IF  NOT  LEN (I$) THEN 149
280 AN$ = AN$ +  LEFT$ (I$,1):J = J + 1: PRINT  LEFT$ (I$,1);
285  IF  LEN (I$) = 1 THEN I$ = ""
290  IF  LEN (I$) THEN I$ =  RIGHT$ (I$, LEN (I$) - 1)
292  GOTO 149
294  IF P = 155 THEN ESCKEY = 1: PRINT : RETURN : REM ESCAPE KEY PRESSED
296  GOTO 149
298  REM


300  REM  ** GET RETURN ROUTINE **
305  REM USES I,J,K,L,P
310  IF AIIE THEN 325
315  GET AN$: IF  ASC (AN$) <  > 13 THEN 315
320  PRINT : RETURN
325 I = 0:J = 0:K = 0
330 I = I - 1: IF I < 0 THEN K = 1 - K:I = 5: PRINT  CHR$ (32 + 63 * K);
335  IF I <  > 5 THEN 355
340 L =  PEEK (36): IF COL80 THEN  IF L =  PEEK (1147) THEN L =  PEEK (14
     03): REM FIND CURRENT HORIZ POSITION WITH 80 COLUMN CARD TURNED ON
345  IF L = 0 THEN  POKE 36, PEEK (33): VTAB  PEEK (37)
350  IF L <  > 0 THEN  POKE 36,L - 1
355 P =  PEEK ( - 16384): IF P <  > 141 THEN 330: REM NOT A RETURN
360  PRINT " ";
365  IF  PEEK (37) = 23 THEN  VTAB 23: REM PEEK(37) CONTAINS CURRENT VTAB
      POSITION -1.  IF ON BOTTOM LINE (VERY COMMON WITH WAIT-FOR-RETURN-K
     EYS) MOVE UP ONE TO PREVENT SCREEN FROM SCROLLING
370  PRINT
375  POKE  - 16368,0: RETURN
399  REM


400  REM  *** SCREEN FORMATTER ROUTINE ***
401  REM STRING TO BE PRINTED IN AN$
402  REM IF 80 COLUMN BOARD IS TURNED ON, MAKE SURE COL80 = 1.  IF BOARD
     IS NOT TO BE USED, MAKE SURE COL80 = 0.
403  REM USES I,J,I$
404  REM ROUGH EQUIVALENT OF  PRINT AN$;" ";
405  REM  USES AIIE SET BY COMPUTER IDENTIFIER ROUTINE
406  REM USUALLY LEAVES 1 EXTRA BLANK AT END OF LINE
407  REM PERFORMS WORD-WRAP AND WILL CONVERT LOWER- TO UPPER-CASE IF USED
      INSIDE AN APPLE II OR II+
410 I =  LEN (AN$): IF  NOT I THEN  RETURN
411 P =  PEEK (36): IF COL80 THEN  IF P =  PEEK (1147) THEN P =  PEEK (14
     03): REM FIND CURRENT HORIZ POSITION WITH 80 COLUMN CARD TURNED ON
412  IF  NOT P THEN  IF I > 1 THEN  IF  ASC (AN$) = 32 THEN AN$ =  RIGHT$
     (AN$,I - 1)
413  IF P + 2 + I <  PEEK (33) AND AIIE THEN  PRINT AN$;" ";:AN$ = "": RE
     : REM EXPRESS CHECK-OUT
```

```
414  IF I > 1 THEN  IF  RIGHT$ (AN$,1) = " " THEN AN$ =  LEFT$ (AN$,I - 1
     )
417  IF P + I <  PEEK (33) THEN I$ = AN$:AN$ = "": GOTO 440
420 J =  PEEK (33) - P + 2:I = J
425 I = I - 1: IF I THEN  IF  MID$ (AN$,I,1) <  > " " THEN 425
430  IF I = 1 THEN I = J
431  IF I = 0 THEN  PRINT : GOTO 410
435 I$ =  LEFT$ (AN$,I - 1): IF  LEN (AN$) > I THEN AN$ =  RIGHT$ (AN$, LEN
     (AN$) - I): REM ISOLATE 1 LINE IN I$
440  IF AIIE THEN  PRINT I$;
445  IF  NOT AIIE THEN K =  LEN (I$) + 1: FOR I = 1 TO  LEN (I$):J =  ASC
     ( RIGHT$ (I$,K - I)): PRINT  CHR$ (J - 32 * (J > 96 AND J < 123));: NEXT
     I
447 P =  PEEK (36): IF COL80 THEN  IF P =  PEEK (1147) THEN P =  PEEK (14
     03)
450  IF  LEN (AN$) THEN  IF P <  > 0 THEN  PRINT
455  IF  LEN (AN$) THEN 410
460  IF P <  > 0 THEN  IF  MID$ (I$, LEN (I$),1) <  > " " THEN  PRINT " "
     ;
465  RETURN
499  REM



500  REM  *** MENU MAKER ROUTINE ***
505  REM SUPPLY PROGRAM TITLE IN TITLE$, MENU TITLE IN SUBTITLE$, CALLING
      MENU TITLE (WHERE USER CAME FROM) IN FROM$, OA=1 IF HELP IS AVAILAB
     LE


510  REM PUT MENU ITEMS IN MENU$(1) THROUGH MENU$(12).  REMEMBER TO DIM M
     ENU$(12) IF MORE THAN 10 ITEMS

515  REM PROGRAM USES INPUT ROUTINE AND SCREEN FORMATTER.  ALL VARIABLES
     AFFECTED BY THEM AS WELL AS II, JJ, KK, LL, MM, NN, OO,PP, AN, II$ A
     RE AFFECTED
520  REM USER'S CHOICE IS RETURNED IN BOTH AN$ AND AN

525  REM  IF FROM$ HAS A TITLE IN IT, ESCKEY WILL BE 1 IF USER WANTS TO G
     O BACK TO CALLING MENU.  MAIN MENUS SHOULD MAKE FROM$="" (NOTHING)

530 J = 0:K = 0:JJ = 0:KK = 0:LL = 0:MM = 0:NN = 0:OO = 0:AN = 0
535 OO = OA: REM STORE WHETHER DESCRIPTIONS ARE AVAILABLE
540 AN$ = ""
545 II = 1
550 KK = KK +  INT ( LEN (MENU$(II)) / (27 + 35 * (COL80 = 1))): REM APPR
     OXIMATE NUMBER OF EXTRA LINES MENU ITEM WILL TAKE
555  IF II < 12 THEN  IF MENU$(II + 1) <  > "End" AND MENU$(II + 1) <  >
     "end" AND MENU$(II + 1) <  > "END" THEN II = II + 1: GOTO 550
560 LL = II: REM NUMBER OF MENU ITEMS
565 NN = 0: IF LL * 2 + KK < 14 THEN NN = 1: REM DETERMINE IF MENU CAN BE
      DOUBLE-SPACED
570 MM = 3: IF COL80 THEN MM = 9: REM SELECT LEFT MARGIN OFFSET: 3 IF 40
     COLUMN, 9 IF 80 COLUMN
575 JJ = 3 + MM: IF LL > 9 THEN JJ = 4 + MM: REM IF MORE THAN 9 ITEMS, IN
     DENT NAME OF EACH BY 1 MORE
580  REM


585  REM   * DISPLAY MENU ROUTINE *
590  TEXT : HOME
595 AN$ = TITLE$: GOSUB 400: POKE 36, PEEK (33) -  LEN (SUBTITLE$) - 1:AN
     $ = SUBTITLE$
600  GOSUB 400
605  FOR II = 1 TO  PEEK (33): PRINT "_";: NEXT : PRINT
610  REM PRINT SELECTIONS
615  FOR II = 1 TO LL
620  HTAB MM: PRINT II;". ";
625  VTAB  PEEK (37): IF COL80 THEN  VTAB  PEEK (1531): REM BELOW PRINT M
     OVES DOWN 1 LINE: THIS COMMAND IS A TRICK TO MOVE UP EXACTLY 1 LINE
     FIRST
```

```
630  POKE 32,JJ: POKE 33, PEEK (33) - JJ: PRINT : REM SET "WINDOW" SO THA
     T TEXT OF MENU ITEM IS INDENTED
635 AN$ = MENU$(II): GOSUB 400: REM PRINT MENU ITEM
640  POKE 32,0: POKE 33, PEEK (33) + JJ: PRINT : REM "RESTORE" FULL WINDO
     W
645  IF NN THEN  PRINT
650  NEXT II
655  IF  PEEK (37) > 16 THEN  PRINT "TOO MANY MENU ITEMS OR TOO LONG LINE
     S.": STOP
660  TEXT : VTAB 17:AN$ = "Select option >": GOSUB 400: PRINT
665  FOR II = 1 TO  PEEK (33): PRINT " ";: NEXT
670  IF  NOT 00 OR  NOT AIIE THEN  PRINT : REM  IF ROOM, SPACE DOWN 1
675  IF  LEN (FROM$) THEN AN$ = "For " + FROM$ + ": press ESC": GOSUB 400

680  PRINT
685  IF AIIE THEN  PRINT "To erase: use the DELETE key"
690 AN$ = "To select: type a number from 1 to " +  STR$ (LL)
700  GOSUB 400: PRINT
705  IF  NOT 00 THEN 720
710  IF AIIE THEN  PRINT "For descriptions: press OPEN-APPLE-?"
715  IF  NOT AIIE THEN  PRINT "FOR DESCRIPTIONS: FOLLOW ANSWER WITH ?"
720 AN$ = "To go to selected item: press RETURN": GOSUB 400: PRINT
723  IF  NOT 00 THEN AN$ = "(There are no descriptions available)": GOSUB
     400
725 FL = 3:AN$ = "": REM SET-UP VALUES FOR INPUT ROUTINE BEFORE CALLING I
     T
730  REM



735  REM ** GET INPUT FROM USER **
740  VTAB 17: HTAB 17: CALL  - 868: HTAB 17
745  IF AIIE OR  NOT  LEN (FROM$) THEN  GOSUB 100: GOTO 795: REM THE LINE
     S THAT FOLLOW ARE TO PICK UP AN ESC PRESS ON AN APPLE II OR II+
750 I = 0:J = 0:M =  PEEK (37):L =  PEEK (36):OAKEY = 0
755 I = I - 1: IF I < 0 THEN K = 1 - K:I = 5:J = 1 - J: NORMAL : IF J THEN
     INVERSE
760  PRINT " ";: POKE 36,L: VTAB M + 1
765 P =  PEEK ( - 16384): IF P < 128 THEN 755
770  NORMAL : REM ABOVE PRINTS A BLINKING CURSOR THE HARD WAY
775  IF P = 155 THEN  POKE  - 16368,0:ESCKEY = 1: GOTO 795: REM ESC PRESS
     ED FIRST
780 L =  PEEK (36): VTAB 20: HTAB 1: CALL  - 868: VTAB M + 1: HTAB L + 1
785  INPUT "";AN$
790  REM
         END OF AII AND AII+ ROUTINE

795  VTAB 19
800  IF  NOT OAKEY THEN 825
805  IF  NOT 00 OR OA$ <  > "?" AND OA$ <  > "/" THEN 735: REM HELP NOT A
     VAILABLE (00 WAS SET TO VALUE OF OAKEY AT BEGINNING OF ROUTINE) OR W
     RONG KEY PRESSED
810  REM  JJ WILL NOW BE USED TO HOLD LENGTH OF AN$:
815 JJ =  VAL (AN$): IF JJ > 0 AND JJ <  = LL THEN II = JJ: GOTO 880: REM
     EXIT MENU FOR HELP
820  IF JJ = 0 THEN  PRINT "--> PLEASE SELECT A NUMBER FIRST <--";: CALL
      - 868: PRINT :AN$ = "": GOTO 735
825  IF SAKEY THEN 735: REM SOLID-APPLE KEY NOT USED FOR MENU
830  IF ESCKEY THEN  IF  LEN (FROM$) THEN 880: REM IF ESCAPE KEY IS PRESS
     ED, MENU IS EXITED
835 JJ =  VAL (AN$): IF JJ > 0 AND JJ <  = LL THEN II = JJ: REM ISOLATE N
     UMBER FROM AN$ AND MAKE II EQUAL IT
840  IF  LEN (AN$) = 0 THEN 735
845  IF 00 THEN  IF  NOT AIIE AND ( RIGHT$ (AN$,1) = "?" OR  RIGHT$ (AN$,
     1) = "/") THEN  IF II > 0 AND II <  = LL THEN OAKEY = 1: GOTO 880
850  IF  LEN (AN$) THEN K = 0: FOR I = 1 TO  LEN (AN$):J =  ASC ( RIGHT$
     (AN$,I)):K = K + (J <  > 32 AND (J < 48 OR J > 57)): NEXT I: IF K THE
      PRINT "       --> PLEASE USE DIGITS <--";: CALL  - 868: PRINT :AN$ =
     "": GOTO 735
855  REM ABOVE LINE CHECKS FOR PRESENCE OF NON-NUMERIC CHARACTERS OTHER T
     HAN SPACE IN THE LINE; IF FOUND, THE LINE IS REJECTED
```

```
860  IF  NOT JJ THEN  IF  LEN (AN$) THEN K = 0: FOR I = 1 TO  LEN (AN$):J
      =  ASC ( RIGHT$ (AN$,I)):K = K + (J = 48): NEXT I: IF  NOT K THEN  PRINT
     "      --> PLEASE USE DIGITS <--";: CALL  - 868: PRINT :AN$ = "": GOTO
     735
865  REM ABOVE LINE CHECKS FOR A SINGLE SPACE AS ANSWER (OTHERWISE, A SPA
     CE IS A ZERO)
870  IF JJ > LL THEN  PRINT "      --> ";JJ;" IS TOO LARGE <--";: CALL  -
     868: PRINT :AN$ = "": GOTO 735
875  IF JJ < 1 THEN  PRINT "      --> 0 IS TOO SMALL <--";: CALL  - 868:A
     N$ = "": GOTO 735
880 AN = II: TEXT : HOME : RETURN : REM EXIT POINT FOR MENU PROGRAM
885  REM



1000  REM *** MAIN PROGRAM ***
1010 I = 0:AN$ = "":J = 0:I$ = "":K = 0: REM DECLARE MOST OFTEN USED VARI
     ABLES FIRST FOR SPEED.
1020  GOSUB 63000: REM FIND OUT  IF AIIE OR NOT
1021  DIM ME$(20)
1025  IF RESULTS >  = 64 THEN COL80 = 1: PRINT  CHR$ (4);"PR#3": REM IF A
     N 80-COLUMN CARD IS PRESENT, USE IT.  IF YOU DON'T WANT IT, CHANGE T
     HE LINE TO:
                1025 PRINT CHR$(21)

1030  TEXT : PRINT : HOME : REM TEXT CLEARS ANY OLD WINDOWS: PRINT CLEARS
      OUT ANY OLD HTAB AND VTAB INFORMATION: HOME CLEARS THE SCREEN
1040  PRINT : HOME
1050 AN$ = "*** Magic Menu ***": POKE 36,( PEEK (33) -  LEN (AN$)) / 2: GOSUB
     400: PRINT : REM CENTER TITLE
1060  PRINT :AN$ = "Magic Menu has five basic subroutines upon which you
     can build your programs:"
1070  GOSUB 400: PRINT : IF COL80 THEN  PRINT : REM BY CHANCE, THE SECOND
      LINE IS JUST LONG ENOUGH TO CAUSE AN EXTRA CARRIAGE RETURN IN 40-CO
     LUMN MODE
1080 AN$ = "1. COMPUTER IDENTIFIER: says you are now using an Apple": GOSUB
     400
1090  IF RESULT = 0 THEN AN$ = "II OR II+."
1100  IF RESULT = 32 THEN AN$ = "IIe."
1110  IF RESULT = 64 THEN AN$ = "IIe with an 80-Column Card."
1120  IF RESULT = 128 THEN AN$ = "IIe with a Memory-Expansion Card."
1130  IF COL80 THEN  GOSUB 400:AN$ = "The other four routines use the inf
     ormation from COMPUTER IDENTIFIER to let your software take full adv
     antage of whatever Apple computer it is run on."
1140  GOSUB 400: PRINT : PRINT
1150 AN$ = "2. SCREEN FORMATTER: ": GOSUB 400
1160 AN$ = "controls text display so that lines are ended between words i
     nstead of in the middle of them."
1170  IF COL80 THEN  GOSUB 400:AN$ = "It also automatically converts all
     lower-case letters to capitals when the program is run on an Apple I
     I or an Apple II+."
1180  GOSUB 400: PRINT : IF COL80 THEN  PRINT
1190 AN$ = "3. MENU MAKER: lets you create uniform, friendly menus in min
     utes, instead of days.  (A sample menu follows.)"
1200  GOSUB 400: PRINT : PRINT
1210 AN$ = "4. INPUT: is the flashing-underline cursor routine you learne
     d on Apple Presents...APPLE.": GOSUB 400
1220 AN$ = "": IF AIIE THEN AN$ = "(Pick 1 from menu.)"
1230  IF COL80 THEN AN$ = " (To reacquaint yourself with this input routi
     ne, select option 1 from the menu that follows.)
1240  GOSUB 400: PRINT : PRINT
1250 AN$ = "5. GET RETURN: waits for you to press RETURN.  It is waiting
     now..."
1260  GOSUB 400: PRINT : PRINT
1270 AN$ = "Press RETURN to continue.": GOSUB 400: GOSUB 300: REM PRINT P
     ROMPT: WAIT FOR A RETURN
1299  REM
```

```
1300  REM *** "help" SCREEN ***
1310  HOME
1315  IF COL80 THEN  VTAB 4
1320 AN$ = "A few words about the sample menu:"
1330  GOSUB 400: PRINT : PRINT : PRINT
1340 AN$ = "The following menu has six selections, only two of which act
     ally do something.  The first selection shows you a familiar exampl
      of the flashing-cursor input routine; the last selection exits the
     program."
1350  GOSUB 400
1360  IF COL80 THEN AN$ = " The other four selections are there to take
     p space on the menu and in the program, so you can see how a more e
     aborate program would be structured.": GOSUB 400
1370  PRINT : PRINT
1375  IF COL80 THEN  PRINT
1380 AN$ = "The Menu Maker enables you to give a description of each men
      item, so the user needn't first wade through pages of " +  CHR$ (3
     ) + "directions" +  CHR$ (34) + ":"
1390  GOSUB 400
1400 AN$ = "to see the description of option 1, type"
1410  GOSUB 400
1420 AN$ = "1? RETURN": IF AIIE THEN AN$ = "1, then press OPEN-APPLE-?"
1430  GOSUB 400: PRINT : PRINT
1435  IF COL80 THEN  PRINT
1440 AN$ = "Explore the menu, then review the listings in Appendix E of
     he Applesoft Tutorial manual for details on how to use these routin
     s in your own programs."
1450  GOSUB 400: PRINT
1460  VTAB 24:AN$ = "Press RETURN to go to the menu.": GOSUB 400: GOSUB
     00
1499  REM


1500  REM ** SAMPLE MAIN MENU **
1510  REM ** MAIN MENU LOOP **
1520 TITLE$ = "Magic Menu": REM TITLE OF MENU
1530 SUBTITLE$ = "Main Menu": REM TITLE OF MENU OR SUBSECTION OF PROGRAM
1540 FROM$ = "": IF  PEEK (6) = 99 AND  PEEK (7) = 99 THEN FROM$ = "the
     isk Menu": REM SEE NOTES FOLLOWING LINE 9060 OF DISK MENU PROGRAM
1550 MENU$(1) = "A sample of the input routine"
1560 MENU$(2) = ""
1570 MENU$(3) = ""
1580 MENU$(4) = ""
1590 MENU$(5) = ""
1600 MENU$(6) = "End the program"
1605  IF  PEEK (6) = 99 AND  PEEK (7) = 99 THEN MENU$(6) = "End the prog
     am and return to the Disk Menu": REM SEE NOTES FOLLOWING LINE 9060
     F DISK MENU
1610 MENU$(7) = "END": REM MENU PROGRAM KEEPS DISPLAYING ITEMS UNTIL IT
     INDS THE WORD "END"
1620 OAKEY = 1: REM HELP AVAILABLE; OPEN-APPLE KEY WILL BE READ
1630  GOSUB 500
1640  IF OAKEY THEN  GOSUB 1810: GOTO 1510: REM "HELP" ASKED FOR
1645  IF ESKEY THEN  GOTO 7000: REM RETURN TO DISK MENU
1650 AN$ = ""
1660  VTAB 8
1670  ON AN GOSUB 2000,3000,4000,5000,6000,7000: REM RETURN PRESSED, SO
     O TO THE SELECTED SECTION
1680  VTAB 24
1685  IF AN <  > 1 THEN  POKE 36,0: IF COL80 THEN  POKE 1147,255: REM TH
     SE 2 POKES ARE EQUIVALENT TO HTAB 1, BUT WORK IN EITHER 40 OR 80 CO
     UMN MODE
1690 AN$ = "Press RETURN for the menu.": GOSUB 400
1700  GOSUB 300: REM WAIT FOR RETURN
1705  IF COL80 THEN  VTAB 22: PRINT : PRINT  CHR$ (4);"PR#3": REM OPTION
     1 TURNS OFF 80-COL CARD (IF ANY).  THIS TURNS IT BACK ON
1710  GOTO 1510
1720  REM
```

```
1800  REM *** GO TO HELP ***
1810  TEXT : HOME : VTAB 8
1820 HOLD$ = AN$
1830  ON AN GOSUB 9000,9200,9300,9400,9500,9600,9725,9825,9925
1850 AN$ = "Press RETURN to go to the menu.": GOSUB 400
1860  GOSUB 300: REM WAIT FOR RETURN
1870 AN$ = HOLD$
1880  RETURN
1890  REM BLANK LINES ARE DONE BY PRESSING THE DOWN-ARROW KEY


2000  REM *** INPUT SAMPLE ***
2010  IF  NOT AIIE THEN AN$ = "SORRY, THIS SAMPLE WORKS ONLY ON AN APPLE
     IIE COMPUTER.": VTAB 10: GOSUB 400: PRINT : RETURN
2015  PRINT  CHR$ (21): REM TURN OFF 80 COLUMN MODE IF IT IS ON.  TRY REM
     OVING THIS LINE TO SEE EFFECT ON PROGRAM
2020  HOME :HT = 1: IF  PEEK (33) > 40 THEN HT = 21: HTAB HT
2030  PRINT "Correct the answer in the box to read": GOSUB 400: PRINT
2040 AN$ = "A herd of cattle"
2050 II = ( PEEK (33) -  LEN (AN$)) / 2: POKE 36,II: REM POKE 36,X IS THE
      SAME AS HTAB X-1, EXCEPT IT WORKS WITH EITHER 40 OR 80-COLUMN MODES

2055  GOSUB 400: PRINT
2060  POKE 36,II: PRINT "  ----"
2070  HTAB HT:AN$ = "by doing the following:": GOSUB 400: PRINT : PRINT
2075  HTAB HT + 1
2080 AN$ = "1. Press left arrow to back up to the": GOSUB 400: PRINT
2085  HTAB HT + 4
2090 AN$ = "right of the " +  CHR$ (34) + "t" +  CHR$ (34) + " in the wor
     d " +  CHR$ (34) + "lot" +  CHR$ (34): GOSUB 400: PRINT
2092  PRINT
2095  HTAB HT + 1
2100 AN$ = "2. Press the DELETE key several times": GOSUB 400: PRINT
2105  HTAB HT + 4
2110 AN$ = "to erase the words " +  CHR$ (34) + "whole lot" +  CHR$ (34):
      GOSUB 400: PRINT : PRINT
2115  HTAB HT + 1
2120 AN$ = "3. Type " +  CHR$ (34) + "herd" +  CHR$ (34): GOSUB 400: PRINT
     : PRINT
2125  HTAB HT + 1
2130 AN$ = "4. Now press the RETURN key to": GOSUB 400: PRINT
2135  HTAB HT + 4
2140 AN$ = "accept the entire answer": GOSUB 400: PRINT
2200  REM BOX
2220  HTAB HT + 2
2230  PRINT "_____"
2240  FOR I = 0 TO 4: HTAB HT + 1: PRINT "|
        |": NEXT : VTAB  PEEK (37)
2250  HTAB HT + 2: PRINT "_____"
2260  VTAB 19: HTAB HT + 4
2270 AN$ = "What is a " +  CHR$ (34) + "drift" +  CHR$ (34) + "?": GOSUB
     400: PRINT : PRINT
2280  HTAB HT + 5
2290  PRINT "> ";
2300  REM GET INPUT
2310 FL = 30: REM  LENGTH OF FIELD
2320 AN$ = "A whole lot of cattle": REM  THE "default" ANSWER (SUPPLIED B
     Y THE PROGRAM - THE USER CAN CHANGE IT)
2330  GOSUB 100: REM INPUT
2335  VTAB 24
2340  IF AN$ = "A herd of cattle" THEN AN$ = "Perfect! ": GOSUB 400
2350  IF AN$ = "A HERD of cattle" THEN AN$ = "Very good!": GOSUB 400
2360  RETURN
3000 AN$ = "Sample routine number 2": GOSUB 400
3999  RETURN
4000 AN$ = "Sample routine number 3": GOSUB 400
4999  RETURN
5000 AN$ = "Sample routine number 4": GOSUB 400
5999  RETURN
```

```
6000 AN$ = "Sample routine number 5": GOSUB 400
6999  RETURN
7000  REM END
7005  PRINT : REM DISK COMMANDS (SEE NEXT LINE) WILL NOT WORK IF THERE IS
      AN "OPEN" PRINT COMMAND SUCH AS A "PRINT;" OR "PRINT,".  THEREFORE,
      ALWAYS DO A "PRINT" BEFORE ISSUING A DOS COMMAND
7010  IF  PEEK (6) = 99 AND  PEEK (7) = 99 THEN  PRINT  CHR$ (4);"RUN DIS
K MENU": END
7020  REM ABOVE LINE RUNS DISK MENU IF DISK MENU ASKED IT TO.  SEE NOTES
FOLLOWING LINE 9060 OF DISK MENU.
7050  TEXT : HOME : TEXT : END : REM CLEAN UP & GO AWAY
9000  REM *HELP SCREENS*
9100  REM
9105 AN$ = "Option 1: the Apple IIe Input Routine"
9107  VTAB 1
9110  HTAB ( PEEK (33) -  LEN (AN$)) / 2
9115  GOSUB 400: PRINT : PRINT
9117  IF COL80 THEN  VTAB 7: REM A DIFFERENT SPACING LOOKS MORE PLEASING
IN 80-COLUMN MODE
9120 AN$ = "The Apple IIe is equipped with a full ASCII keyboard, includi
ng the DELETE key, a new key for the Apple II series.": GOSUB 400
9122  PRINT : IF COL80 THEN  PRINT : REM BY COINCIDENCE, THE ABOVE SENTEN
CE, IN 40-COLUMN MODE, IS JUST LONG ENOUGH TO "force" AN EXTRA CARRI
AGE RETURN.  SO, TO GET THE EXTRA SPACE IN 80-COLUMN MODE, WE MUST P
RINT ONCE MORE
9125 AN$ = "Using the DELETE key and the flashing-cursor input routine in
cluded within this program,": GOSUB 400
9130 AN$ = "you can scan forward and backward through your answers, inser
ting and deleting characters at will.": GOSUB 400
9135 AN$ = "  The input routine automatically gives the standard Applesof
t BASIC blinking cursor to users of the Apple II or II+.": GOSUB 400

9145  PRINT : PRINT
9150 AN$ = "The input routine, along with the other routines in MAGIC MEN
U, will enable you to easily create humanized programs.": GOSUB 400
9155  GOSUB 400: PRINT : PRINT
9160 AN$ = "Since you are using an Apple II or II+, you cannot take advan
tage of this cursor directly, but using it in your programs will mak
e it easier for others."
9165  IF AIIE THEN AN$ = "Select option 1 to reacquaint yourself with the
 underline cursor."
9170  GOSUB 400
9180  IF COL80 THEN  PRINT : PRINT
9199  RETURN
9200 AN$ = "There is no description available for this option.": GOSUB 40
0
9250  PRINT : VTAB 24
9299  RETURN
9300 AN$ = "There is no option available for this description.": GOSUB 40
0
9350  PRINT : VTAB 24
9399  RETURN
9400 AN$ = "Sample help screen for option 4": GOSUB 400
9450  PRINT : VTAB 24
9499  RETURN
9500 AN$ = "Sample help screen for option 5": GOSUB 400
9550  PRINT : VTAB 24
9599  RETURN
9600 AN$ = "This option lets you gracefully exit the program.": GOSUB 400

9650  PRINT : VTAB 24
9699  RETURN
10000  RETURN
63000  REM  *** COMPUTER ID ROUTINE ***
63010  REM *** AIIE OR NOT? ***
63020  REM  USES I,J,K,RE  -- SETS AIIE TO 1 IF IT IS AN AIIE
63030  REM SETS RESULT DEPENDENT ON AVAILABLE HARDWARE
63040  REM  RESULTS OF 0 MEANS NOT AIIE; 32 MEANS AIIE BUT NO 80 COLUMNS;
 64 MEANS AIIE WITH 80 COLUMNS BUT NO AUX MEM; 128 MEANS AIIE WITH A
UX MEM
```

```
63050  DATA  8, 120, 173, 0, 224, 141, 208, 2, 173, 0, 208, 141, 209, 2,
     173, 0, 212, 141, 210, 2, 173, 0, 216, 141, 211, 2, 173, 129, 192, 1
     73, 129, 192, 173, 179, 251, 201, 6, 208, 73, 173
63060  DATA  23, 192, 48, 60, 173, 19, 192, 48, 39, 173, 22, 192, 48, 34,
      160, 42, 190, 162, 3, 185, 0, 0, 150, 0, 153, 162, 3, 136, 208, 242
     , 76, 1, 0, 8, 160, 42, 185, 162, 3, 153
63070  DATA  0, 0, 136, 208, 247, 104, 176, 8, 169, 128, 141, 207, 3, 76,
      73, 3, 169, 64, 141, 207, 3, 76, 73, 3, 169, 32, 141, 207, 3, 76, 7
     3, 3, 169, 0, 141, 207, 3, 173, 0, 224
63080  DATA  205, 208, 2, 208, 24, 173, 0, 208, 205, 209, 2, 208, 16, 173
     , 0, 212, 205, 210, 2, 208, 8, 173, 0, 216, 205, 211, 2, 240, 56, 17
     3, 136, 192, 173, 0, 224, 205, 208, 2, 240, 6
63090  DATA  173, 128, 192, 76, 161, 3, 173, 0, 208, 205, 209, 2, 240, 6,
      173, 128, 192, 76, 161, 3, 173, 0, 212, 205, 210, 2, 240, 6, 173, 1
     28, 192, 76, 161, 3, 173, 0, 216, 205, 211, 2
63100  DATA  240, 3, 173, 128, 192, 40, 96, 169, 238, 141, 5, 192, 141, 3
     , 192, 141, 0, 8, 173, 0, 12, 201, 238, 208, 14, 14, 0, 12, 173, 0,
     8, 205, 0, 12, 208, 3, 56, 176, 1, 24
63110  DATA  141, 4, 192, 141, 2, 192, 76, 29, 3, 234
63120 J = 975:K = 724
63130  FOR I = 0 TO 249
63140  READ L
63150  POKE K + I,L
63160  NEXT
63170  CALL K
63180 RESULTS =  PEEK (J)
63190  IF RESULTS <  > 0 THEN AIIE = 1
63200  RETURN
```

Disk Menu

DISK MENU a été écrit en deux heures à peine. En fait, il n'a pas exactement été écrit : l'auteur a utilisé MAGIC MENU pour créer DISK MENU (bientôt vous pourrez faire de même).

La méthode est très simple. Après avoir chargé MAGIC MENU, l'auteur a enlevé tout ce qui n'était plus utile, a changé les noms des MENU$S, et éliminé les sous-menus.

Explorez ce programme pour qu'il vous devienne familier. Quand vous écrivez vos propres programmes en utilisant le MAGIC MENU, employez DISK MENU comme base. Vous pouvez aussi vous reporter au listing de MAGIC MENU pour les détails, et vous obtiendrez la vitesse accrue et la taille réduite de la version DISK MENU compacte.

Si vous comparez les lignes 100 à 900 de DISK MENU et les mêmes dans MAGIC MENU, vous verrez une différence. Les sous-programmes sont plus compacts et les instructions REM ont été en grande partie supprimées. Tous les noms de variables plus long que 2 caractères ont été réduits au minimum.

La ligne 1030 fait sauter au programme les premières instructions si DISK MENU a assigné la valeur 99 aux cases mémoires 6 et 7, ce qui signifie que l'utilisateur n'en est pas à sa première exécution. Référez vous à l'étude de la ligne 9060 qui suit.

Les lignes 1047 à 1075 constituent le texte de DISK MENU. Comme le Screen Formatter de MAGIC MENU répartit automatiquement le texte non formaté, l'auteur n'avait pas à faire attention aux fins de ligne en rajoutant ces lignes. En dehors de cela, il n'y a rien de particulièrement nouveau ou intéressant dans DISK MENU. C'est une illustration exacte de la manière d'utiliser MAGIC MENU : un programme pratique fonctionnel, utilisant des entrées standard, des instructions claires, et des menus agréables à lire, a été créé en un court laps de temps. Il a fallu plus d'un mois pour écrire le bloc sous-programme sur lequel le programme est basé ; cela ne prend que quelques heures pour construire quelque chose dessus.

La ligne 3060 a été mise précisément parce que l'auteur voulait savoir si Menu Maker traiterait convenablement une ligne aussi longue.

Les lignes 6000 à 6999 sont le sous-programme qui simule la frappe de la commande CATALOG. La commande est produite de manière normale par la ligne 6080. Les commandes DOS peuvent être exécutées à partir d'un programme Applesoft en tapant une chaîne composée de Ctrl – D suivi de la commande DOS ; comme CHR$(4) est le code ASCII pour Ctrl – D et que TITLE$ a la valeur "CATALOG" à la ligne 6030, la ligne 6080 produit le listing CATALOG sur l'écran.

La ligne 9060 est la clé par laquelle les programmes annexes reviennent à DISK MENU après exécution. Le problème est de communiquer entre les programmes avec un drapeau : le drapeau est mis (si DISK MENU a été exécuté), revenez ; sinon ne revenez pas. On ne peut pas utiliser une variable standard comme drapeau entre deux programmes parce que toutes les variables sont mises à zéro quand un programme démarre. Mais beaucoup de cases mémoire ne sont pas affectées par Applesoft. Dans ce cas, l'auteur a choisi les cases 6 et 7, plaçant la valeur arbitraire 99 dans chacune. Les chances que les deux cases se voient affecter par hasard la valeur 99 sont seulement de 1 sur 65536.

Les sous-programmes de DISK MENU ont été réduits à la moitié de leur leur taille dans MAGIC MENU et considérablement accélérés. L'outillage du programmeur DOS offre des programmes qui compactent votre software et lui permettent en même temps d'exécuter à une vitesse maximum. L'usage de ces "utilitaires" vous donne la possibilité d'écrire des programmes que vous pouvez lire et comprendre, tout en faisant une version comprimée exécutable à grande vitesse.

Il existe cinq types d'utilitaires particulièrement utiles aux programmeurs Applesoft :

1. Ceux qui **compressent** les programmes en enlevant les instructions REM, en raccourcissant les noms de variables à 2 caractères, en combinant des instructions qui peuvent être mises ensemble.

2. Ceux qui **compilent** ou transforment les instructions en langage machine, ce qui accélère de 100 fois l'exécution.

3. Ceux qui **renumérotent** les numéros de ligne aussi bien que les lignes de référence de GOSUB et GOTO. (Cf la partie suivante pour explication sur cette techique).

4. Ceux qui vous permettent d'**éditer** des lignes de programme plus facilement que le mode escape.

5. Ceux qui fournissent des **références croisées**, vous disent aussi bien tous les noms de variables et les lignes où ils sont utilisés que les numéros de ligne appelés par des instructions GOSUB et GOTO.

Renumérotation et fusion de portions de programmes

Les applications dans cette partie utilisent un programme sur le disque DOS 3.3 SYSTEM MASTER appelé RENUMBER. Ce programme a été utilisé par l'auteur de DISK MENU pour copier les menus de MAGIC MENU de telle manière que seuls les contenus des chaînes aient à être réécrits. Avec l'habitude, vous apprendrez à moins taper que ce que vous faites actuellement et RENUMBER est là pour ça.

Cet exemple et le suivant dans la partie d'après vous permettent d'essayer la méthode sur votre machine. Insérez le disque DOS 3.3 SYSTEM MASTER dans votre lecteur si ce n'est déjà fait. Puis tapez

```
RUN RENUMBER
```

Si vous utilisez un seul lecteur, enlevez la disquette SYSTEM MASTER et mettez la disquette APPLESOFT SAMPLER dans le lecteur. Si vous avez deux lecteurs, APPLESOFT SAMPLER peut être placé dans le second. (Les instructions qui suivent sont pour le premier cas. Si vous avez deux lecteurs, n'oubliez pas d'ajouter, D2 à la suite de la commande suivante). Tapez

```
LOAD DISK MENU
```

et, quand DISK MENU est chargé, tapez

```
DEL 0,1999
DEL 3000,63999
LIST
```

Vous verrez qu'il ne reste pas grand chose. A présent pour numéroter la portion de DISK MENU commençant à la ligne 7000, tapez

```
& 7000
LIST
```

Le menu est marqué sur de nouveaux numéros de ligne. C'est RENUMBER qui fait cela. En utilisant ''&'' tout seul, vous renumérotez tout le programme en partant de la ligne 10 et de 10 en 10. & 7000 renumérote le programme à partir de la ligne 7000 et de 10 en 10. A présent, pour le stocker, tapez.

```
&H
```

Maintenant, si vous listez le programme, vous verrez qu'il n'y a rien de visible : il est en mémoire. &H dit au DOS de stocker quelque chose dans la mémoire principale de l'ordinateur de manière à ce que vous puissiez charger un autre programme. Cet outil est limité, bien sûr, pour la taille des programmes sur lesquels vous travaillez et la taille de la mémoire de l'ordinateur.

Avant de fusionner un programme dans un autre, assurez-vous que vous ne détruisez pas d'anciennes lignes. Tapez

```
LOAD DISK MENU
LIST 7000, 7999
```

Vous ne verrez pas de listing car ces numéros de ligne n'ont pas été utilisés. Vous pouvez procéder à l'opération appelée fusionnage (''merge'' en anglais). Frappez

```
&M
```

pour fusionner les nouvelles lignes dans l'ancien programme. En changeant les noms de programme en de nouveaux noms, y compris le nom du menu secondaire dans le menu principal, et en donnant l'adresse du nouveau menu à l'instruction ON...GOSUB de la ligne 1230, vous pouvez encore indexer 12 autres programmes sur le disque. L'auteur a utilisé cette méthode (RENUMBER, &H, &M) 3 fois pour créer trois menus secondaires dans le programme.

RENUMBER est un programme très utile pour programmer. Il donne beaucoup de souplesse. Quand vous avez des programmes si longs que vous ne pouvez plus les développer, vous pouvez créer des sous-programmes et les déplacer à d'autres numéros de ligne. RENUMBER résoud le problème d'avoir à placer 11 lignes supplémentaires entre les lignes 40 et 50 : déplacez simplement la ligne 50 et ce qu'il y a au-dessus vers 500 ou 1000. Cela vous donnera toute la place voulue (voir le *Manuel DOS* pour des instructions complètes de RENUMBER).

Programme

```
0  REM DISK MENU - SEPT 1982 - BY TOG
1  GOTO 1000
2  TEXT : PRINT  CHR$ (21): HOME : POKE 33,33: END
100  REM

          *** INPUT ROUTINE ***


140 I = 5:J = 0:K = 0:L = 0:M = 0:I$ = "":ESC = 0:0A = 0:0A$ = "":SA = 0:
     SA$ = "": IF FL = 0 THEN FL = 245
142  IF  NOT AIIE THEN  INPUT "";AN$: RETURN
145  PRINT AN$;:J =  LEN (AN$)
149 M =  PEEK (37)
153 L =  PEEK (36): IF COL80 THEN  IF L =  PEEK (1147) THEN L =  PEEK (14
     03)
155  PRINT " ";I$;" ";
160 N = 1: IF L +  LEN (I$) >  =  PEEK (33) - 3 AND  PEEK (37) =  PEEK (3
     5) - 1 THEN N = 0
165  POKE 36,L: POKE 1403,L: VTAB M + 1
170 I = I - 1: IF I < 0 THEN K = 1 - K:I = 5: PRINT  CHR$ (32 + 63 * K);
175  POKE 36,L: POKE 1403,L: VTAB M + 1
180 P =  PEEK ( - 16384): IF P < 128 THEN 170
185  IF  PEEK ( - 16287) > 127 THEN OAKEY = 1
190  IF  PEEK ( - 16286) > 127 THEN SAKEY = 1
195  POKE  - 16368,0:K = 0:I = 0
200  IF OAKEY THEN OA$ =  CHR$ (P - 128):AN$ = AN$ + I$: PRINT I$;" ": RETURN

205  IF SAKEY THEN SA$ =  CHR$ (P - 128):AN$ = AN$ + I$: PRINT I$;" ": RETURN

210  IF P > 159 AND P <  > 255 THEN  IF J +  LEN (I$) < FL THEN  IF N THEN
     AN$ = AN$ +  CHR$ (P - 128):J = J + 1: PRINT  CHR$ (P);: GOTO 149
215  IF P <  > 255 THEN 240
220  IF J THEN  PRINT " ";: POKE 36,L: VTAB M + 1: PRINT  CHR$ (136);:J =
     J - 1
225  IF J = 0 THEN AN$ = ""
230  IF J THEN AN$ =  LEFT$ (AN$,J)
235  GOTO 149
240  IF P <  > 136 THEN 265
245  IF J THEN  PRINT " ";: POKE 36,L: VTAB M + 1: PRINT  CHR$ (136);:I$ =
      RIGHT$ (AN$,1) + I$:J = J - 1
250  IF J = 0 THEN AN$ = ""
255  IF J THEN AN$ =  LEFT$ (AN$,J)
260  GOTO 149
265  IF P = 141 THEN AN$ = AN$ + I$: PRINT I$;" ": RETURN
270  IF P <  > 149 THEN 294
275  IF  NOT  LEN (I$) THEN 149
280 AN$ = AN$ +  LEFT$ (I$,1):J = J + 1: PRINT  LEFT$ (I$,1);
285  IF  LEN (I$) = 1 THEN I$ = ""
290  IF  LEN (I$) THEN I$ =  RIGHT$ (I$, LEN (I$) - 1)
292  GOTO 149
294  IF P = 155 THEN ESCKEY = 1: PRINT : RETURN
296  GOTO 149
300  REM

          ** GET RETURN **


310  IF AIIE THEN 325
315  GET AN$: IF  ASC (AN$) <  > 13 THEN 315
320  PRINT : RETURN
325 I = 0:J = 0:K = 0
330 I = I - 1: IF I < 0 THEN K = 1 - K:I = 5: PRINT  CHR$ (32 + 63 * K);
335  IF I <  > 5 THEN 355
340 L =  PEEK (36): IF COL80 THEN  IF L =  PEEK (1147) THEN L =  PEEK (14
     03)
345  IF L = 0 THEN  POKE 36, PEEK (33): VTAB  PEEK (37)
```

```
350  IF L <  > 0 THEN  POKE 36,L - 1
355 P =  PEEK ( - 16384): IF P <  > 141 THEN 330
360  PRINT " ";
365  IF  PEEK (37) = 23 THEN  VTAB 23
370  PRINT
375  POKE  - 16368,0: RETURN
400  REM

        *** SCREEN FORMATTER ***


410 I =  LEN (AN$): IF  NOT I THEN  RETURN
411 P =  PEEK (36): IF COL80 THEN  IF P =  PEEK (1147) THEN P =  PEEK (14
     03)
412  IF  NOT P THEN  IF I > 1 THEN  IF  ASC (AN$) = 32 THEN AN$ =  RIGHT$
     (AN$,I - 1)
413  IF P + 2 + I <  PEEK (33) AND AIIE THEN  PRINT AN$;" ";:AN$ = "": RETURN

414  IF I > 1 THEN  IF  RIGHT$ (AN$,1) = " " THEN AN$ =  LEFT$ (AN$,I - 1
     )
417  IF P + I <  PEEK (33) THEN I$ = AN$:AN$ = "": GOTO 440
420 J =  PEEK (33) - P + 2:I = J
425 I = I - 1: IF I THEN  IF  MID$ (AN$,I,1) <  > " " THEN 425
430  IF I = 1 THEN I = J
431  IF I = 0 THEN  PRINT : GOTO 410
435 I$ =  LEFT$ (AN$,I - 1): IF  LEN (AN$) > I THEN AN$ =  RIGHT$ (AN$, LEN
     (AN$) - I)
440  IF AIIE THEN  PRINT I$;
445  IF  NOT AIIE THEN K =  LEN (I$) + 1: FOR I = 1 TO  LEN (I$):J =  ASC
     ( RIGHT$ (I$,K - I)): PRINT  CHR$ (J - 32 * (J > 96 AND J < 123));: NEXT
     I
447 P =  PEEK (36): IF COL80 THEN  IF P =  PEEK (1147) THEN P =  PEEK (14
     03)
450  IF  LEN (AN$) THEN  IF P <  > 0 THEN  PRINT
455  IF  LEN (AN$) THEN 410
460  IF P <  > 0 THEN  IF  MID$ (I$, LEN (I$),1) <  > " " THEN  PRINT " "
     ;
465  RETURN
500  REM

         *** MENU MAKER ***


530 J = 0:K = 0:JJ = 0:KK = 0:LL = 0:MM = 0:NN = 0:00 = 0:AN = 0
535 00 = 0A
540 AN$ = ""
545 II = 1
550 KK = KK +  INT ( LEN (MENU$(II)) / (27 + 35 * (COL80 = 1)))
555  IF II < 12 THEN  IF MENU$(II + 1) <  > "End" AND MENU$(II + 1) <  >
     "end" AND MENU$(II + 1) <  > "END" THEN II = II + 1: GOTO 550
560 LL = II
565 NN = 0: IF LL * 2 + KK < 14 THEN NN = 1
570 MM = 3: IF COL80 THEN MM = 9
575 JJ = 3 + MM: IF LL > 9 THEN JJ = 4 + MM
590  TEXT : HOME
595 AN$ = TITLE$: GOSUB 400: POKE 36, PEEK (33) -  LEN (SUBTITLE$) - 1:AN
     $ = SUBTITLE$
600  GOSUB 400
605  FOR II = 1 TO  PEEK (33): PRINT "_";: NEXT : PRINT
615  FOR II = 1 TO LL
620  HTAB MM: PRINT II;". ";
625  VTAB  PEEK (37): IF COL80 THEN  VTAB  PEEK (1531)
630  POKE 32,JJ: POKE 33, PEEK (33) - JJ: PRINT
635 AN$ = MENU$(II): GOSUB 400
640  POKE 32,0: POKE 33, PEEK (33) + JJ: PRINT
645  IF NN THEN  PRINT
650  NEXT II
655  IF  PEEK (37) > 16 THEN  PRINT "TOO MANY MENU ITEMS OR TOO LONG LINE
     S.": STOP
```

```
660  TEXT : VTAB 17:AN$ = "Select option >": GOSUB 400: PRINT
665  FOR II = 1 TO  PEEK (33): PRINT "_";: NEXT
670  IF  NOT OO OR  NOT AIIE THEN  PRINT
675  IF  LEN (FROM$) THEN AN$ = "For " + FROM$ + ": press ESC": GOSUB 400

680  PRINT
685  IF AIIE THEN  PRINT "To erase: use the DELETE key"
690 AN$ = "To select: type a number from 1 to " +  STR$ (LL)
700  GOSUB 400: PRINT
705  IF  NOT OO THEN 720
710  IF AIIE THEN  PRINT "For descriptions: press OPEN-APPLE-?"
715  IF  NOT AIIE THEN  PRINT "FOR DESCRIPTIONS: FOLLOW ANSWER WITH ?"
720 AN$ = "To go to selected item: press RETURN": GOSUB 400: PRINT
723  IF  NOT OO THEN AN$ = "(There are no descriptions available)": GOSUB
     400
725 FL = 3:AN$ = ""
735  REM
740  VTAB 17: HTAB 17: CALL  - 868: HTAB 17
745  IF AIIE OR  NOT  LEN (FROM$) THEN  GOSUB 100: GOTO 795
750 I = 0:J = 0:M =  PEEK (37):L =  PEEK (36):OAKEY = 0
755 I = I - 1: IF I < 0 THEN K = 1 - K:I = 5:J = 1 - J: NORMAL : IF J THEN
     INVERSE
760  PRINT " ";: POKE 36,L: VTAB M + 1
765 P =  PEEK ( - 16384): IF P < 128 THEN 755
770  NORMAL
775  IF P = 155 THEN  POKE  - 16368,0:ESCKEY = 1: GOTO 795
780 L =  PEEK (36): VTAB 20: HTAB 1: CALL  - 868: VTAB M + 1: HTAB L + 1
785  INPUT "";AN$
795  VTAB 19
800  IF  NOT OAKEY THEN 825
805  IF  NOT OO OR OA$ <  > "?" AND OA$ <  > "/" THEN 735
815 JJ =  VAL (AN$): IF JJ > 0 AND JJ <  = LL THEN II = JJ: GOTO 880
820  IF JJ = 0 THEN  PRINT "--> PLEASE SELECT A NUMBER FIRST <--";: CALL
      - 868: PRINT :AN$ = "": GOTO 735
825  IF SAKEY THEN 735
830  IF ESCKEY THEN  IF  LEN (FROM$) THEN 880
835 JJ =  VAL (AN$): IF JJ > 0 AND JJ <  = LL THEN II = JJ
840  IF  LEN (AN$) = 0 THEN 735
845  IF OO THEN  IF  NOT AIIE AND ( RIGHT$ (AN$,1) = "?" OR  RIGHT$ (AN$,
     1) = "/") THEN  IF II > 0 AND II <  = LL THEN OAKEY = 1: GOTO 880
850  IF  LEN (AN$) THEN K = 0: FOR I = 1 TO  LEN (AN$):J =  ASC ( RIGHT$
     (AN$,I)):K = K + (J <  > 32 AND (J < 48 OR J > 57)): NEXT I: IF K THEN
      PRINT "       --> PLEASE USE DIGITS <--";: CALL  - 868: PRINT :AN$ =
     "": GOTO 735
860  IF  NOT JJ THEN  IF  LEN (AN$) THEN K = 0: FOR I = 1 TO  LEN (AN$):J
      =  ASC ( RIGHT$ (AN$,I)):K = K + (J = 48): NEXT I: IF  NOT K THEN  PRINT
     "       --> PLEASE USE DIGITS <--";: CALL  - 868: PRINT :AN$ = "": GOTO
     735
870  IF JJ > LL THEN  PRINT "       --> ";JJ;" IS TOO LARGE <--";: CALL  -
     868: PRINT :AN$ = "": GOTO 735
875  IF JJ < 1 THEN  PRINT "       --> 0 IS TOO SMALL <--";: CALL  - 868:A
     N$ = "": GOTO 735
880 AN = II: TEXT : HOME : RETURN

1000  REM *** MAIN PROGRAM ***
1010 I = 0:AN$ = "":J = 0:I$ = "":K = 0: REM DECLARE MOST OFTEN USED VARI
     ABLES FIRST FOR SPEED.
1020  GOSUB 63000: REM FIND OUT  IF AIIE OR NOT
1021  DIM ME$(20)
1025  IF RESULTS >  = 64 THEN COL80 = 1: PRINT  CHR$ (4);"PR#3": REM IF A
     N 80-COLUMN CARD IS PRESENT, USE IT.  IF YOU DON'T WANT IT, CHANGE T
     HE LINE TO:
                1025 PRINT CHR$(21)

1030  IF  PEEK (6) = 99 AND  PEEK (7) = 99 THEN 1100: REM  SEE NOTES FOLL
     OWING 9060
1035  TEXT : HOME : VTAB 2: PRINT
1040 AN$ = "*** THE APPLESOFT SAMPLER DISK ***": POKE 36,( PEEK (33) -  LEN
     (AN$)) / 2: GOSUB 400: PRINT
1045  VTAB 6 + 3 * (COL80 = 1): REM START ON LINE 6 UNLESS COL80=1, IN WH
     ICH CASE, START ON LINE 9
1047 AN$ = "Featuring": GOSUB 400
```

```
1050 AN$ = "the collection of programs to be studied in conjuction with t
     he Applesoft Tutorial, with a special appearance by ": GOSUB 400
1060 AN$ = "Postage Rates, the example program from the Applesoft Referen
     ce Manual.": GOSUB 400: PRINT : PRINT
1065 AN$ = "The following " +  CHR$ (34) + "main menu" +  CHR$ (34) + " l
     ets you select one of several " +  CHR$ (34) + "sub-menus" +  CHR$ (
     34) + " with the names of the programs on this disk.": GOSUB 400
1070 AN$ = "You can choose to pick a program from a sub-menu, or you can
     elect to end the": GOSUB 400
1075 AN$ = "program. ": GOSUB 400
1076  GOSUB 400
1080  PRINT : VTAB 24
1089 AN$ = "Press RETURN to go to the Main Menu.": GOSUB 400: GOSUB 300
1090  REM  ** MAIN MENU **
1100  REM ** MAIN MENU LOOP **
1110 TITLE$ = "The Applesoft Sampler Disk"
1120 SUBTITLE$ = "Main Menu": REM TITLE OF MENU OR SUBSECTION OF PROGRAM
1130 MENU$(1) = "Example Programs from The Applesoft Tutorial"
1140 MENU$(2) = "Tutorial Appendix E: More Programs To Play With"
1150 MENU$(3) = "Postage Rates -- From The Applesoft Reference Manual"
1180 MENU$(4) = "End the program (This option will type CATALOG for you a
     nd will leave you in Applesoft.)"
1190 MENU$(5) = "END": REM MENU PROGRAM KEEPS DISPLAYING ITEMS UNTIL IT F
     INDS THE WORD "end"
1195 OAKEY = 0: REM HELP NOT AVAILABLE; OPEN-APPLE KEY WILL NOT BE READ
1197 FROM$ = ""
1200  GOSUB 500
1215 AN$ = ""
1225  VTAB 8
1230  ON AN GOSUB 2000,3000,4000,6000: REM RETURN PRESSED, SO GO TO THE S
     ELECTED SECTION
1240  GOTO 1100
1250  REM


1999  REM BLANK LINES ARE DONE BY PRESSING THE DOWN-ARROW KEY



2000  REM *** EXAMPLES ***
2010  REM


2020 TITLE$ = "Applesoft Tutorial"
2030 SUBTITLE$ = "Examples"
2040 MENU$(1) = "COLORLOOP"
2045 MENU$(2) = "HUE"
2046 MENU$(3) = "QUILT"
2047 MENU$(4) = "SPACES"
2048 MENU$(5) = "COLORBOUNCE"
2049 MENU$(6) = "RANDOM"
2050 MENU$(7) = "HORSES"
2051 MENU$(8) = "MOIRE"
2100 MENU$(9) = "ALPHABET"
2110 MENU$(10) = "DECIMAL"
2120 MENU$(11) = "COLORBOUNCESOUND"
2140 MENU$(12) = "Return to Main Menu"
2160 FROM$ = "Main Menu":AN$ = "":OAKEY = 0: GOSUB 500
2170  IF ESCKEY THEN  RETURN
2180  IF AN = 12 THEN  RETURN
2190 TITLE$ = MENU$(AN): REM STORE NAME OF SELECTION FOR USE BY EACH ROUT
     INE
2195  PRINT  CHR$ (21)
2200  GOTO 9000
2210  GOTO 2000
2220  REM


3000  REM TUTOR EXAMPLES ***
3010  REM
```

```
3020 TITLE$ = "Applesoft Tutorial"
3030 SUBTITLE$ = "Samples"
3040 MENU$(1) = "SCRAMBLER"
3050 MENU$(2) = "MAGIC MENU"
3060 MENU$(3) = "DISK MENU (Which is this very program -- selecting it wi
     ll only result in a long wait followed by the program starting over.
       It is included here because it is included in Appendix E.)"
3070  IF  NOT COL80 THEN MENU$(2) = MENU$(2) +  CHR$ (10):MENU$(3) = MENU
     $(3) +  CHR$ (10): REM CHR$(10) IS DOWN-ARROW: ADDING DOWN-ARROWS WI
     LL ADD SPACES ABOVE AND BELOW MENU OPTION 3
3080  REM IN 80 COLUMN MODE, THERE IS ENOUGH ROOM THAT THE MENU MAKER ROU
     TINE WILL ADD SPACES AUTOMATICALLY, SO WE DO IT ONLY IF THE COMPUTER
      IS NOT IN 80-COLUMN MODE
3090 MENU$(4) = "CONVERTER"
3100 MENU$(5) = "Return to Main Menu"
3110 MENU$(6) = "end"
3120 FROM$ = "Main Menu":AN$ = "":OAKEY = 0: GOSUB 500
3130  IF ESCKEY THEN  RETURN
3140  IF AN = 5 THEN  RETURN
3150  IF AN = 1 AND  NOT AIIE THEN  HOME : VTAB 10: PRINT "SORRY, SCRAMBL
     ER CAN ONLY BE": PRINT "PLAYED ON AN APPLE IIE.": VTAB 24: PRINT "PR
     ESS RETURN TO GO BACK TO THE MENU.";: GOSUB 300: GOTO 3000: REM  "pa
     tch"
3160 TITLE$ = MENU$(AN): REM STORE NAME OF SELECTION FOR USE BY EACH ROUT
     INE
3165  IF AN = 2 THEN TITLE$ = "MAGIC MENU": REM A DOWN-ARROW MAY HAVE BEE
     N APPENDED IN LINE 3070
3170  IF AN = 3 THEN TITLE$ = "DISK MENU": REM A "PATCH" TO THE PROGRAM:
     MENU$(3) HAS AN EXTRA DESCRIPTION
3180  GOTO 9000
3190  REM


4000  REM *** REFERENCE MAN ***
4010  REM

4020 TITLE$ = "Applesoft Tutorial"
4030 SUBTITLE$ = "Example"
4040 MENU$(1) = "POSTAGE RATES"
4060 MENU$(2) = "Return to Main Menu"
4070 MENU$(3) = "end"
4080 FROM$ = "Main Menu":AN$ = "":OAKEY = 0: GOSUB 500
4090  IF ESCKEY THEN  RETURN
4100  IF AN = 2 THEN  RETURN
4110 TITLE$ = MENU$(AN): REM STORE NAME OF SELECTION FOR USE BY EACH ROUT
     INE
4120  GOTO 9000
4140  REM


5999  RETURN
6000  REM *** CATALOG ***
6010  HOME
6020  VTAB 10
6025  PRINT "]";
6030 TITLE$ = "CATALOG"
6040  FOR I = 1 TO  LEN (TITLE$)
6050  PRINT  MID$ (TITLE$,I,1);
6055 K =  PEEK ( - 16336) +  PEEK ( - 16336)
6060  FOR J = 1 TO 400 *  RND (1): NEXT
6070  NEXT I
6072  POKE 6,0: POKE 7,0: REM SEE NOTES FOLLOWING 9060
6075  PRINT
6080  PRINT  CHR$ (4);TITLE$
6999  END
```

```
 9000  REM *** GO TO DISK ***
 9060  POKE 6,99: POKE 7,99: REM LET THE PROGRAMS FROM APPENDIX E KNOW THA
     T THEY CAN RETURN TO THIS PROGRAM.  USUALLY, YOU USE A VARIABLE TO "
     TURN ON" OR "TURN OFF" AN OPTION
 9061  REM BUT VARIABLES ARE ALL CLEARED BETWEEN PROGRAMS.  LOCATIONS 6 &
     7 ARE NOT AFFECTED BY APPLESOFT, SO THE AUTHOR CHOSE THEM AS A WAY T
     O COMMUNICATE
 9075  PRINT
 9080  PRINT  CHR$ (4);"RUN ";TITLE$
 9090  END
63000  REM *** COMPUTER ID ***
63010  REM *** AIIE OR NOT? ***
63020  REM  USES I,J,K,RE  -- SETS AIIE TO 1 IF IT IS AN AIIE
63030  REM SETS RESULT DEPENDENT ON AVAILABLE HARDWARE
63040  REM RESULTS OF 0 MEANS NOT A //E; 32 MEANS A//E BUT NO 80 COLUMNS;
     64 MEANS A//E WITH 80 COLUMNS BUT NO AUX MEM; 128 MEANS A//E WITH A
     UX MEM
63050  DATA  8, 120, 173, 0, 224, 141, 208, 2, 173, 0, 208, 141, 209, 2,
     173, 0, 212, 141, 210, 2, 173, 0, 216, 141, 211, 2, 173, 129, 192, 1
     73, 129, 192, 173, 179, 251, 201, 6, 208, 73, 173
63060  DATA  23, 192, 48, 60, 173, 19, 192, 48, 39, 173, 22, 192, 48, 34,
     160, 42, 190, 162, 3, 185, 0, 0, 150, 0, 153, 162, 3, 136, 208, 242
     , 76, 1, 0, 8, 160, 42, 185, 162, 3, 153
63070  DATA  0, 0, 136, 208, 247, 104, 176, 8, 169, 128, 141, 207, 3, 76,
     73, 3, 169, 64, 141, 207, 3, 76, 73, 3, 169, 32, 141, 207, 3, 76, 7
     3, 3, 169, 0, 141, 207, 3, 173, 0, 224
63080  DATA  205, 208, 2, 208, 24, 173, 0, 208, 205, 209, 2, 208, 16, 173
     , 0, 212, 205, 210, 2, 208, 8, 173, 0, 216, 205, 211, 2, 240, 56, 17
     3, 136, 192, 173, 0, 224, 205, 208, 2, 240, 6
63090  DATA  173, 128, 192, 76, 161, 3, 173, 0, 208, 205, 209, 2, 240, 6,
     173, 128, 192, 76, 161, 3, 173, 0, 212, 205, 210, 2, 240, 6, 173, 1
     28, 192, 76, 161, 3, 173, 0, 216, 205, 211, 2
63100  DATA  240, 3, 173, 128, 192, 40, 96, 169, 238, 141, 5, 192, 141, 3
     , 192, 141, 0, 8, 173, 0, 12, 201, 238, 208, 14, 14, 0, 12, 173, 0,
     8, 205, 0, 12, 208, 3, 56, 176, 1, 24
63110  DATA  141, 4, 192, 141, 2, 192, 76, 29, 3, 234
63120 J = 975:K = 724
63130  FOR I = 0 TO 249
63140  READ L
63150  POKE K + I,L
63160  NEXT
63170  CALL K
63180 RESULTS =  PEEK (J)
63190  IF RESULTS <  > 0 THEN AIIE = 1
63200  RETURN
```

CONVERTER

CONVERTER est un programme en lui même qui est basé sur l'ensemble des sous programmes maintenant familiers. Il est dérivé de MAGIC MENU, et permet l'adjonction de différentes conversions qui peuvent vous être utiles. Il peut vous fournir l'expérience de programmation la plus valable : fonctionnement intime avec un programme écrit par des experts.

Quand vous faites des adjonctions à CONVERTER, utilisez le programme RENUMBER. De nombreuses conversions sont faites de la même manière : seul le texte change. Recopiez les lignes et changer les mots.

La ligne 11145 est particulière : c'est un exemple de la sorte de précaution qu'il faut prendre en programmation. La création de programmes conviviaux qui ne font jamais de fautes ou n'apparaissent pas insensés aux utilisateurs est un challenge difficile mais existant.

Programme

```
0  REM CONVERTER - SEPT, 1982 - BG & TOG
1  GOTO 2000
2  TEXT : PRINT  CHR$ (21): HOME : POKE 33,33: END
100  REM

          *** INPUT ROUTINE ***


140 I = 5:J = 0:K = 0:L = 0:M = 0:I$ = "":ESC = 0:0A = 0:0A$ = "":SA = 0:
     SA$ = "": IF FL = 0 THEN FL = 245
142  IF  NOT AIIE THEN  INPUT "";AN$: RETURN
145  PRINT AN$;:J =  LEN (AN$)
149 M =  PEEK (37)
153 L =  PEEK (36): IF COL80 THEN  IF L =  PEEK (1147) THEN L =  PEEK (14
     03)
155  PRINT " ";I$;" ";
160 N = 1: IF L +  LEN (I$) >  =  PEEK (33) - 3 AND  PEEK (37) =  PEEK (3
     5) - 1 THEN N = 0
165  POKE 36,L: POKE 1403,L: VTAB M + 1
170 I = I - 1: IF I < 0 THEN K = 1 - K:I = 5:*PRINT  CHR$ (32 + 63 * K);
175  POKE 36,L: POKE 1403,L: VTAB M + 1
180 P =  PEEK ( - 16384): IF P < 128 THEN 170
185  IF  PEEK ( - 16287) > 127 THEN OAKEY = 1
190  IF  PEEK ( - 16286) > 127 THEN SAKEY = 1
195  POKE  - 16368,0:K = 0:I = 0
200  IF OAKEY THEN OA$ =  CHR$ (P - 128):AN$ = AN$ + I$: PRINT I$;" ": RETURN

205  IF SAKEY THEN SA$ =  CHR$ (P - 128):AN$ = AN$ + I$: PRINT I$;" ": RETURN

210  IF P > 159 AND P <  > 255 THEN  IF J +  LEN (I$) < FL THEN  IF N THEN
     AN$ = AN$ +  CHR$ (P - 128):J = J + 1: PRINT  CHR$ (P);: GOTO 149
215  IF P <  > 255 THEN 240
220  IF J THEN  PRINT " ";: POKE 36,L: VTAB M + 1: PRINT  CHR$ (136);:J =
     J - 1
225  IF J = 0 THEN AN$ = ""
230  IF J THEN AN$ =  LEFT$ (AN$,J)
235  GOTO 149
```

```
240  IF P <  > 136 THEN 265
245  IF J THEN  PRINT " ";: POKE 36,L: VTAB M + 1: PRINT  CHR$ (136);:I$ =
     RIGHT$ (AN$,1) + I$:J = J - 1
250  IF J = 0 THEN AN$ = ""
255  IF J THEN AN$ =  LEFT$ (AN$,J)
260  GOTO 149
265  IF P = 141 THEN AN$ = AN$ + I$: PRINT I$;" ": RETURN
270  IF P <  > 149 THEN 294
275  IF  NOT  LEN (I$) THEN 149
280 AN$ = AN$ +  LEFT$ (I$,1):J = J + 1: PRINT  LEFT$ (I$,1);
285  IF  LEN (I$) = 1 THEN I$ = ""
290  IF  LEN (I$) THEN I$ =  RIGHT$ (I$, LEN (I$) - 1)
292  GOTO 149
294  IF P = 155 THEN ESCKEY = 1: PRINT : RETURN
296  GOTO 149
300  REM

         ** GET RETURN **


310  IF AIIE THEN 325
315  GET AN$: IF  ASC (AN$) <  > 13 THEN 315
320  PRINT : RETURN
325 I = 0:J = 0:K = 0
330 I = I - 1: IF I < 0 THEN K = 1 - K:I = 5: PRINT  CHR$ (32 + 63 * K);
335  IF I <  > 5 THEN 355
340 L =  PEEK (36): IF COL80 THEN  IF L =  PEEK (1147) THEN L =  PEEK (14
     03)
345  IF L = 0 THEN  POKE 36, PEEK (33): VTAB  PEEK (37)
350  IF L <  > 0 THEN  POKE 36,L - 1
355 P =  PEEK ( - 16384): IF P <  > 141 THEN 330
360  PRINT " ";
365  IF  PEEK (37) = 23 THEN  VTAB 23
370  PRINT
375  POKE  - 16368,0: RETURN
400  REM

         *** SCREEN FORMATTER ***


410 I =  LEN (AN$): IF  NOT I THEN  RETURN
411 P =  PEEK (36): IF COL80 THEN  IF P =  PEEK (1147) THEN P =  PEEK (14
     03)
412  IF  NOT P THEN  IF I > 1 THEN  IF  ASC (AN$) = 32 THEN AN$ =  RIGHT$
     (AN$,I - 1)
413  IF P + 2 + I <  PEEK (33) AND AIIE THEN  PRINT AN$;" ";:AN$ = "": RETURN

414  IF I > 1 THEN  IF  RIGHT$ (AN$,1) = " " THEN AN$ =  LEFT$ (AN$,I - 1
     )
417  IF P + I <  PEEK (33) THEN I$ = AN$:AN$ = "": GOTO 440
420 J =  PEEK (33) - P + 2:I = J
425 I = I - 1: IF I THEN  IF  MID$ (AN$,I,1) <  > " " THEN 425
430  IF I = 1 THEN I = J
431  IF I = 0 THEN  PRINT : GOTO 410
435 I$ =  LEFT$ (AN$,I - 1): IF  LEN (AN$) > I THEN AN$ =  RIGHT$ (AN$, LEN
     (AN$) - I)
440  IF AIIE THEN  PRINT I$;
445  IF  NOT AIIE THEN K =  LEN (I$) + 1: FOR I = 1 TO  LEN (I$):J =  ASC
     ( RIGHT$ (I$,K - I)): PRINT  CHR$ (J - 32 * (J > 96 AND J < 123));: NEXT
     I
447 P =  PEEK (36): IF COL80 THEN  IF P =  PEEK (1147) THEN P =  PEEK (14
     03)
450  IF  LEN (AN$) THEN  IF P <  > 0 THEN  PRINT
455  IF  LEN (AN$) THEN 410
460  IF P <  > 0 THEN  IF  MID$ (I$, LEN (I$),1) <  > " " THEN  PRINT " "
     ;
465  RETURN
500  REM

         *** MENU MAKER ***
```

```
530 J = 0:K = 0:JJ = 0:KK = 0:LL = 0:MM = 0:NN = 0:00 = 0:AN = 0.
535 00 = 0A
540 AN$ = ""
545 II = 1
550 KK = KK +  INT ( LEN (MENU$(II)) / (27 + 35 * (COL80 = 1)))
555  IF II < 12 THEN  IF MENU$(II + 1) <  > "End" AND MENU$(II + 1) <  >
     "end" AND MENU$(II + 1) <  > "END" THEN II = II + 1: GOTO 550
560 LL = II
565 NN = 0: IF LL * 2 + KK < 14 THEN NN = 1
570 MM = 3: IF COL80 THEN MM = 9
575 JJ = 3 + MM: IF LL > 9 THEN JJ = 4 + MM
590  TEXT : HOME
595 AN$ = TITLE$: GOSUB 400: POKE 36, PEEK (33) -  LEN (SUBTITLE$) - 1:AN
    $ = SUBTITLE$
600  GOSUB 400
605  FOR II = 1 TO  PEEK (33): PRINT "_";: NEXT : PRINT
615  FOR II = 1 TO LL
620  HTAB MM: PRINT II;". ";
625  VTAB  PEEK (37): IF COL80 THEN  VTAB  PEEK (1531)
630  POKE 32,JJ: POKE 33, PEEK (33) - JJ: PRINT
635 AN$ = MENU$(II): GOSUB 400
640  POKE 32,0: POKE 33, PEEK (33) + JJ: PRINT
645  IF NN THEN  PRINT
650  NEXT II
655  IF  PEEK (37) > 16 THEN  PRINT "TOO MANY MENU ITEMS OR TOO LONG LINE
    S.": STOP
660  TEXT : VTAB 17:AN$ = "Select option >": GOSUB 400: PRINT
665  FOR II = 1 TO  PEEK (33): PRINT "_";: NEXT
670  IF  NOT 00 OR  NOT AIIE THEN  PRINT
675  IF  LEN (FROM$) THEN AN$ = "For " + FROM$ + ": press ESC": GOSUB 400

680  PRINT
685  IF AIIE THEN  PRINT "To erase: use the DELETE key"
690 AN$ = "To select: type a number from 1 to " +  STR$ (LL)
700  GOSUB 400: PRINT
705  IF  NOT 00 THEN 720
710  IF AIIE THEN  PRINT "For descriptions: press OPEN-APPLE-?"
715  IF  NOT AIIE THEN  PRINT "FOR DESCRIPTIONS: FOLLOW ANSWER WITH ?"
720 AN$ = "To go to selected item: press RETURN": GOSUB 400: PRINT
723  IF  NOT 00 THEN AN$ = "(There are no descriptions available)": GOSUB
    400
725 FL = 3:AN$ = ""
735  REM
740  VTAB 17: HTAB 17: CALL  - 868: HTAB 17
745  IF AIIE OR  NOT  LEN (FROM$) THEN  GOSUB 100: GOTO 795
750 I = 0:J = 0:M =  PEEK (37):L =  PEEK (36):0AKEY = 0
755 I = I - 1: IF I < 0 THEN K = 1 - K:I = 5:J = 1 - J: NORMAL : IF J THEN
     INVERSE
760  PRINT " ";: POKE 36,L: VTAB M + 1
765 P =  PEEK ( - 16384): IF P < 128 THEN 755
770  NORMAL
775  IF P = 155 THEN  POKE  - 16368,0:ESCKEY = 1: GOTO 795
780 L =  PEEK (36): VTAB 20: HTAB 1: CALL  - 868: VTAB M + 1: HTAB L + 1
785  INPUT "";AN$
795  VTAB 19
800  IF  NOT 0AKEY THEN 825
805  IF  NOT 00 OR 0A$ <  > "?" AND 0A$ <  > "/" THEN 735
815 JJ =  VAL (AN$): IF JJ > 0 AND JJ <  = LL THEN II = JJ: GOTO 880
820  IF JJ = 0 THEN  PRINT "--> PLEASE SELECT A NUMBER FIRST <--";: CALL
     - 868: PRINT :AN$ = "": GOTO 735
825  IF SAKEY THEN 735
830  IF ESCKEY THEN  IF  LEN (FROM$) THEN 880
835 JJ =  VAL (AN$): IF JJ > 0 AND JJ <  = LL THEN II = JJ
840  IF  LEN (AN$) = 0 THEN 735
845  IF 00 THEN  IF  NOT AIIE AND ( RIGHT$ (AN$,1) = "?" OR  RIGHT$ (AN$,
    1) = "/") THEN  IF II > 0 AND II <  = LL THEN 0AKEY = 1: GOTO 880
850  IF  LEN (AN$) THEN K = 0: FOR I = 1 TO  LEN (AN$):J =  ASC ( RIGHT$
    (AN$,I)):K = K + (J <  > 32 AND (J < 48 OR J > 57)): NEXT I: IF K THEN
     PRINT "      --> PLEASE USE DIGITS <--";: CALL  - 868: PRINT :AN$ =
    "": GOTO 735
```

```
860  IF  NOT JJ THEN  IF  LEN (AN$) THEN K = 0: FOR I = 1 TO  LEN (AN$):J
     =  ASC ( RIGHT$ (AN$,I)):K = K + (J = 48): NEXT I: IF  NOT K THEN  PRINT
     "      --> PLEASE USE DIGITS <--";: CALL  - 868: PRINT :AN$ = "": GOTO
     735
870  IF JJ > LL THEN  PRINT "      --> ";JJ;" IS TOO LARGE <--";: CALL  -
     868: PRINT :AN$ = "": GOTO 735
875  IF JJ < 1 THEN  PRINT "      --> 0 IS TOO SMALL <--";: CALL  - 868:A
     N$ = "": GOTO 735
880 AN = II: TEXT : HOME : RETURN
1000  REM *** SUBROUTINES ***
1010  REM SOME SPECIAL ROUTINES FOR THIS PROGRAM
1100  REM Y OR N?
1110  REM  PUT NAME OF CALLING ROUTINE IN AN$
1120  REM   UPON RETURN, AN WILL BE 1 IF YES, 0 IF NO
1130  REM USES AN, AN$, I, II$
1135  PRINT
1140 II$ = "Do you want to do another " + AN$ + " conversion? (Y OR N) "
1145  HTAB 1: VTAB 22: CALL  - 958:AN$ = II$: GOSUB 400:FL = 2: GOSUB 100

1150 I = 0: IF  LEN (AN$) THEN I =  ASC (AN$): IF I <  >  ASC ("Y") AND I
     <  >  ASC ("y") AND I <  >  ASC ("N") AND I <  >  ASC ("n") THEN 11
     45
1155  IF  NOT I THEN 1145
1160 AN = 0: IF I =  ASC ("y") OR I =  ASC ("Y") THEN AN = 1
1170  RETURN
2000  REM *** BEGIN PROGRAM ***
2010  REM THE CONVERTER - A PROGRAM SKELETON USED FOR MEASUREMENT CONVERS
     IONS - 1982
2020  GOSUB 63000: REM SET AIIE TO TRUE (1) OR FALSE (0)
2025  IF RESULTS >  = 64 THEN COL80 = 1: PRINT  CHR$ (4);"PR#3": REM IF A
     N 80-COLUMN CARD IS PRESENT, USE IT.  IF YOU DON'T WANT IT, CHANGE T
     HE LINE TO:
                1025 PRINT CHR$(21)

2030  TEXT : PRINT : HOME
2040 I = 0:AN$ = "":J = 0:I$ = "":K = 0
2050  DIM MENU$(12)
2070 AN$ = "*** Converter ***": POKE 36,( PEEK (33) -  LEN (AN$)) / 2: GOSUB
     400: PRINT : REM CENTER TITLE
2080  PRINT : PRINT : IF COL80 THEN  VTAB 6
2090 AN$ = "Converter is a program to convert"
2095  GOSUB 400
2100 AN$ = "measurements from one form to another,": GOSUB 400
2110 AN$ = "such as kilometers to miles.": GOSUB 400: PRINT : PRINT
2120 AN$ = "The program is structured to provide hundreds of conversions,
      but it is now in skeleton form": GOSUB 400
2130 AN$ = "-- that is, only a few sample conversions are provided.  The
     intent": GOSUB 400
2140 AN$ = "is for you to learn, through studying the program listings an
     d the given samples, how to alter the program": GOSUB 400
2150 AN$ = "to provide you the conversions you will find useful.": GOSUB
     400: PRINT : PRINT
2160 AN$ = "Through this learning process, you will gain insight into alt
     ering other programs to suit your needs.": GOSUB 400
2220  PRINT
2230  VTAB 23
2240 AN$ = "Press RETURN to continue.": GOSUB 400: GOSUB 300
2250  HOME
2260  REM


2270  REM *** MAIN MENU ***
2280 TITLE$ = "Converter":SUBTITLE$ = "Main Menu"
2285 FROM$ = "": IF  PEEK (6) = 99 AND  PEEK (7) = 99 THEN FROM$ = "the D
     isk Menu": REM SEE NOTES FOLLOWING LINE 9060 IN THE DISK MENU PROGRAM
2290 MENU$(1) = "LINEAR MEASURES"
2300 MENU$(2) = "TEMPERATURE MEASURES"
2310 MENU$(3) = "SPEED MEASURES"
2320 MENU$(4) = ""
2330 MENU$(5) = ""
2340 MENU$(6) = ""
```

```
2350 MENU$(7) = ""
2360 MENU$(8) = ""
2370 MENU$(9) = ""
2380 MENU$(10) = ""
2390 MENU$(11) = ""
2400 MENU$(12) = "End the program"
2405  IF  LEN (FROM$) THEN MENU$(12) = "End program and go to Disk Menu"
2410  REM

2430 AN$ = ""
2440 OAKEY = 0: REM NO HELP AVAILABLE
2450  GOSUB 500: REM DO MENU
2455  IF ESCKEY THEN  GOTO 22000: REM SEE NOTES FOLLOWING LINE 9060 IN TH
     E DISK MENU PROGRAM
2460  ON AN GOSUB 11000,12000,13020,14000,15000,16000,17000,18000,19000,2
     0000,21000,22000
2470  GOTO 2270
10999  REM

11000  REM *** LINEAR MEASURE
11001  REM

11002 TITLE$ = "Conversions"
11005 SUBTITLE$ = "Change linear measure"
11010 MENU$(1) = "Kilometers to Miles"
11020 MENU$(2) = "Miles to Kilometers"
11030 MENU$(3) = "Return to Main Menu"
11035 MENU$(4) = "end"
11040 FROM$ = "Main Menu":AN$ = "":OAKEY = 0: GOSUB 500
11050  IF ESCKEY THEN  RETURN
11060  IF AN = 3 THEN  RETURN
11065 TITLE$ = MENU$(AN): REM STORE NAME OF SELECTION FOR USE BY EACH ROU
     TINE
11070  ON AN GOSUB 11100,11200
11080  GOTO 11000
11099  REM

11100  REM ** KILOMETERS --> MILES
11110  TEXT : HOME : PRINT :AN$ = "Convert Kilometers to Miles": GOSUB 40
     0: PRINT : REM FUNCTION TITLE
11120  VTAB 10:AN$ = "Enter number of kilometers ": GOSUB 400:AN$ = "":FL
      = 10: GOSUB 100
11130 AN =  VAL (AN$): IF AN = 0 AND AN$ <  > "0" THEN  PRINT  CHR$ (7): GOTO
     11110: REM ONLY NUMBERS ALLOWED
11140 MILE = AN * .62137
11145 I$ = " mile": IF  INT (MILE * 10000) / 10000 <  > 1 THEN I$ = " mil
     es"
11146 J$ = " kilometer": IF AN <  > 1 THEN J$ = " kilometers
11150  PRINT :AN$ =  STR$ (AN) + J$ + " = " +  STR$ (MILE) + I$: GOSUB 40
     0
11160 AN$ = TITLE$: GOSUB 1100
11170  IF AN THEN 11100
11190  RETURN
11199  REM


11200  REM ** MILES --> KILOMETERS
11210  TEXT : HOME : PRINT :AN$ = "Convert Miles to Kilometers": GOSUB 40
     0: PRINT : REM FUNCTION TITLE
11220  VTAB 10:AN$ = "Enter number of miles ": GOSUB 400:AN$ = "":FL = 10
     : GOSUB 100
11230 AN =  VAL (AN$): IF AN = 0 AND AN$ <  > "0" THEN  PRINT  CHR$ (7): GOTO
     11210: REM ONLY NUMBERS ALLOWED
11240 KILO = AN / .62137
11245 I$ = " mile": IF AN <  > 1 THEN I$ = " miles"
11246 J$ = " kilometer": IF  INT (KILO * 10000) / 10000 <  > 1 THEN J$ =
     " kilometers"
11250  PRINT :AN$ =  STR$ (AN) + I$ + " = " +  STR$ (KILO) + J$: GOSUB 40
     0
11260 AN$ = TITLE$: GOSUB 1100
```

```
11270  IF AN THEN 11200
11280  RETURN
11290  REM


11300  RETURN : REM PROGRAMMER TO CREATE CONVERSION ROUTINE HERE
12000  REM *** TEMPERATURE ***
12010  REM

12020 TITLE$ = "Conversions"
12030 SUBTITLE$ = "Change temperature measure"
12040 MENU$(1) = "Fahrenheit to Celsius"
12050 MENU$(2) = "Celsius to Fahrenheit"
12060 MENU$(3) = "Return to Main Menu"
12070 MENU$(4) = "end"
12080 FROM$ = "Main Menu":AN$ = "":OAKEY = 0: GOSUB 500
12090  IF ESCKEY THEN  RETURN
12100  IF AN = 3 THEN  RETURN
12110 TITLE$ = MENU$(AN): REM STORE NAME OF SELECTION FOR USE BY EACH ROU
     TINE
12120  ON AN GOSUB 12200,12400
12130  GOTO 12000
12140  REM


12200  REM ** FAHRENHEIT --> CELSIUS
12210  TEXT : HOME : PRINT :AN$ = "Convert Fahrenheit to Celsius": GOSUB
     400: PRINT : REM FUNCTION TITLE
12220  VTAB 10:AN$ = "Enter degrees Fahrenheit ": GOSUB 400:AN$ = "":FL =
     10: GOSUB 100
12230 AN =  VAL (AN$): IF AN = 0 AND AN$ <  > "0" THEN  PRINT  CHR$ (7): GOTO
     12210: REM ONLY NUMBERS ALLOWED
12235  IF AN <  - 459.6 THEN  PRINT :AN$ = "One cannot have a temperature
     below Absolute 0 (-459.6 degrees Fahrenheit)": GOSUB 400: GOTO 1248
     0
12240 CEL = (AN - 32) / 1.8
12250 I$ = " degree": IF  INT (CEL * 10000) / 10000 <  > 1 THEN I$ = " de
     grees"
12260 J$ = " degree": IF AN <  > 1 THEN J$ = " degrees"
12270  PRINT :AN$ =  STR$ (AN) + J$ + " Fahrenheit = " +  STR$ (CEL) + I$
     + " Celsius": GOSUB 400
12280 AN$ = TITLE$: GOSUB 1100
12290  IF AN THEN 12200
12300  RETURN
12310  REM


12400  REM ** CELSIUS --> FAHRENHEIT
12410  TEXT : HOME : PRINT :AN$ = "Convert Celsius to Fahrenheit": GOSUB
     400: PRINT : REM FUNCTION TITLE
12420  VTAB 10:AN$ = "Enter degrees Celsius": GOSUB 400:AN$ = "":FL = 10:
     GOSUB 100
12430 AN =  VAL (AN$): IF AN = 0 AND AN$ <  > "0" THEN  PRINT  CHR$ (7): GOTO
     12410: REM ONLY NUMBERS ALLOWED
12435  IF AN <  - 273.1 THEN  PRINT :AN$ = "One cannot have a temperature
     below Absolute 0 (-273.1 degrees Celsius)": GOSUB 400: GOTO 12480
12440 FAHR = AN * 1.8 + 32
12450 I$ = " degree": IF AN <  > 1 THEN I$ = " degrees"
12460 J$ = " degree": IF  INT (FAHR * 10000) / 10000 <  > 1 THEN J$ = " d
     egrees"
12470  PRINT :AN$ =  STR$ (AN) + I$ + " Celsius = " +  STR$ (FAHR) + J$ +
     " Fahrenheit": GOSUB 400
12480 AN$ = TITLE$: GOSUB 1100
12490  IF AN THEN 12400
12500  RETURN
12510  REM


12600  RETURN : REM PROGRAMMER TO CREATE KELVIN CONVERSION ROUTINE HERE
13000  REM *** SPEED ***
13010  REM
```

```
13020 TITLE$ = "Conversions"
13030 SUBTITLE$ = "Change speed measure"
13040 MENU$(1) = "Kilometers/hour to miles/hour"
13050 MENU$(2) = "Miles/hour to kilometers/hour
13060 MENU$(3) = "Return to Main Menu"
13070 MENU$(4) = "end"
13080 FROM$ = "Main Menu":AN$ = "":OAKEY = 0: GOSUB 500
13090  IF ESCKEY THEN  RETURN
13100  IF AN = 3 THEN  RETURN
13110 TITLE$ = MENU$(AN): REM STORE NAME OF SELECTION FOR USE BY EACH ROU
     TINE
13120  ON AN GOSUB 13200,13400
13130  GOTO 13000
13140  REM


13200  REM ** KILO/HR --> MILES/HR
13210  TEXT : HOME : PRINT :AN$ = "Convert Kilometers per hour to Miles p
     er hour": GOSUB 400: PRINT : REM FUNCTION TITLE
13220  VTAB 10:AN$ = "Enter kilometers per hour ": GOSUB 400:AN$ = "":FL =
     10: GOSUB 100
13230 AN =  VAL (AN$): IF AN = 0 AND AN$ <  > "0" THEN  PRINT  CHR$ (7): GOTO
     13210: REM ONLY NUMBERS ALLOWED
13240 MILE = AN * .62137
13250 I$ = " mile": IF  INT (MILE * 10000) / 10000 <  > 1 THEN I$ = " mil
     es"
13260 J$ = " kilometer": IF AN <  > 1 THEN J$ = " kilometers
13270  PRINT :AN$ =  STR$ (AN) + J$ + " per hour = " +  STR$ (MILE) + I$ +
     " per hour": GOSUB 400
13280 AN$ = TITLE$: GOSUB 1100
13290  IF AN THEN 13200
13300  RETURN
13310  REM


13400  REM  ** MILES /HR --> KILO/HR
13410  TEXT : HOME : PRINT :AN$ = "Convert Miles per hour to kilometers p
     er hour": GOSUB 400: PRINT : REM FUNCTION TITLE
13420  VTAB 10:AN$ = "Enter number of mph ": GOSUB 400:AN$ = "":FL = 10: GOSUB
     100
13430 AN =  VAL (AN$): IF AN = 0 AND AN$ <  > "0" THEN  PRINT  CHR$ (7): GOTO
     13410: REM ONLY NUMBERS ALLOWED
13440 KILO = AN / .62137
13450 I$ = " mile": IF AN <  > 1 THEN I$ = " miles"
13460 J$ = " kilometer": IF  INT (KILO * 10000) / 10000 <  > 1 THEN J$ =
     " kilometers"
13470  PRINT :AN$ =  STR$ (AN) + I$ + " per hour = " +  STR$ (KILO) + J$ +
     " per hour": GOSUB 400
13480 AN$ = TITLE$: GOSUB 1100
13490  IF AN THEN 13400
13500  RETURN
13600  REM


13601  RETURN : REM PROGRAMMER TO CREATE CONVERSION ROUTINE HERE
14000  REM *** SELECTION 4 ***
14999  RETURN
15000  REM *** SELECTION 5 ***
15999  RETURN
16000  REM *** SELECTION 6 ***
16999  RETURN
17000  REM *** SELECTION 7 ***
17999  RETURN
18000  REM *** SELECTION 8 ***
18999  RETURN
19000  REM *** SELECTION 9 ***
19999  RETURN
20000  REM *** SELECTION 10 ***
20999  RETURN
21000  REM *** SELECTION 11 ***
```

```
21999  RETURN
22000  REM *** SELECTION 12 ***
22010  REM *** END THE PROGRAM ***

22020  IF  PEEK (6) = 99 AND  PEEK (7) = 99 THEN  PRINT : PRINT  CHR$ (4)
     ;"RUN HELLO": END
22050  TEXT : HOME : TEXT : END
63000  REM *** COMPUTER ID ***
63010  REM *** AIIE OR NOT? ***
63020  REM  USES I,J,K,RE  -- SETS AIIE TO 1 IF IT IS AN AIIE
63030  REM SETS RESULT DEPENDENT ON AVAILABLE HARDWARE
63040  REM RESULTS OF 0 MEANS NOT A //E; 32 MEANS A//E BUT NO 80 COLUMNS;
      64 MEANS A//E WITH 80 COLUMNS BUT NO AUX MEM; 128 MEANS A//E WITH A
     UX MEM
63050  DATA  8, 120, 173, 0, 224, 141, 208, 2, 173, 0, 208, 141, 209, 2,
      173, 0, 212, 141, 210, 2, 173, 0, 216, 141, 211, 2, 173, 129, 192, 1
      73, 129, 192, 173, 179, 251, 201, 6, 208, 73, 173
63060  DATA  23, 192, 48, 60, 173, 19, 192, 48, 39, 173, 22, 192, 48, 34,
      160, 42, 190, 162, 3, 185, 0, 0, 150, 0, 153, 162, 3, 136, 208, 242
      , 76, 1, 0, 8, 160, 42, 185, 162, 3, 153
63070  DATA  0, 0, 136, 208, 247, 104, 176, 8, 169, 128, 141, 207, 3, 76,
      73, 3, 169, 64, 141, 207, 3, 76, 73, 3, 169, 32, 141, 207, 3, 76, 7
      3, 3, 169, 0, 141, 207, 3, 173, 0, 224
63080  DATA  205, 208, 2, 208, 24, 173, 0, 208, 205, 209, 2, 208, 16, 173
      , 0, 212, 205, 210, 2, 208, 8, 173, 0, 216, 205, 211, 2, 240, 56, 17
      3, 136, 192, 173, 0, 224, 205, 208, 2; 240, 6
63090  DATA  173, 128, 192, 76, 161, 3, 173, 0, 208, 205, 209, 2, 240, 6,
      173, 128, 192, 76, 161, 3, 173, 0, 212, 205, 210, 2, 240, 6, 173, 1
      28, 192, 76, 161, 3, 173, 0, 216, 205, 211, 2
63100  DATA  240, 3, 173, 128, 192, 40, 96, 169, 238, 141, 5, 192, 141, 3
      , 192, 141, 0, 8, 173, 0, 12, 201, 238, 208, 14, 14, 0, 12, 173, 0,
      8, 205, 0, 12, 208, 3, 56, 176, 1, 24
63110  DATA  141, 4, 192, 141, 2, 192, 76, 29, 3, 234
63120 J = 975:K = 724
63130  FOR I = 0 TO 249
63140  READ L
63150  POKE K + I,L
63160  NEXT
63170  CALL K
63180 RESULTS =  PEEK (J)
63190  IF RESULTS <  > 0 THEN AIIE = 1
63200  RETURN
```

Quelques Idées

Nous espérons que vous avez pris plaisir à explorer la ''profondeur'' de grands programmes. Si vous vous sentez un peu débordé, reposez-vous. Vous avez acquis une énorme quantité d'informations nouvelles dans l'espace de quelques heures. Examinez ces programmes à nouveau dans un délai de quelques jours ou d'une semaine. Regardez les programmes écrits par d'autres auteurs. Chaque fois que vous le ferez, vous découvrirez une nouvelle astuce ou technique que vous pouvez incorporer dans vos propres programmes.

Glossaire

adresse : Nombre utilisé pour identifier quelque chose, par exemple un emplacement dans la mémoire de l'ordinateur.

affichage : (1) Information accessible visuellement, spécialement sur un écran ou un appareillage d'affichage. (2) Le fait de rendre accessible l'information visuellement. (3) Un appareil d'affichage.

algorithme : Procédure séquentielle pou résoudre un problème ou l'accomplissement d'une tâche.

amorce (boot) : Sert à initialiser un ordinateur en chargeant un programme dans sa mémoire à partir d'une mémoire externe, un disque par exemple. Est souvent réalisé en deux temps, d'abord en chargeant un petit programme dont l'utilité est de charger un programme beaucoup plus étendu. On dit alors que ce programme est « auto-chargeable » car il se charge lui-même en mémoire à partir de sa propre amorce.

annuaire : Liste ou répertoire des fichiers stockés dans un disque, souvent appelé catalogue.

appel : Sert à demander l'exécution d'un sous-programme ou d'une fonction.

Apple *II*e : Ordinateur Personnel de la famille Apple II, fabriqué et vendu par Apple Computer, Inc.

Applesoft : Version étendue du langage de programmation BASIC utilisé avec l'ordinateur Apple *II*e et capable de traiter des nombres en virgule flottante. Un interpréteur qui permet de créer et exécuter des programmes en Applesoft est résident en mémoire morte dans l'Apple *II*e. Voir **BASIC entier.**

argument : Valeur sur laquelle opère une fonction. Un argument peut être une constante, une variable ou une expression et est contenu entre les parenthèses qui suivent la fonction. Paramètre est un synonyme.

ASCII : Abréviation d'American Standard Code for Information Interchange ; c'est un code dans lequel les nombres de 0 à 127 représentent des caractères de texte comprenant les chiffres de 0 à 9, les lettres de l'alphabet, les signes de ponctuation, des caractères spéciaux et des caractères de contrôle. Ce code est utilisé pour la représentation de texte à l'intérieur du micro-ordinateur et pour transmettre du texte entre micro-ordinateurs ou entre un micro-ordinateur et un périphérique.

autocollant de protection d'écriture : Petit autocollant, habituellement argenté, utilisé pour protéger le disque à l'écriture lorsqu'il recouvre l'encoche de permission d'écriture.

avis : Sert à rappeler ou à signaler à l'utilisateur qu'une action donnée est attendue, généralement en affichant un symbole distinctif, un message de rappel ou un menu avec des choix possibles sur l'écran.

BASIC : Abréviation de Beginner's All-purpose Symbolic Instruction Code ; c'est un langage de haut niveau conçu pour être facile à apprendre et à utiliser. Deux versions de BASIC sont disponibles chez Apple Computer pour être utilisées sur Apple *II*e : Applesoft (sous forme de mémoire morte résidente) et l'Integer BASIC (fourni sur le DOS 3.3 SYSTEM MASTER disque).

BASIC entier : Version du langage de programmation BASIC utilisée dans la famille des ordinateurs Apple II ; plus ancien qu'Applesoft, il n'est capable que de traiter des nombres entiers (virgule fixe). Un interpréteur pour créer et exécuter des programmes en BASIC entier est inclus dans le disque système d'exploitation disque 3.3 SYSTEM MASTER et il est automatiquement chargé dans la mémoire de l'ordinateur quand l'ordinateur est initialisé avec ce disque. Voir **Applesoft**.

binaire : Représentation des nombres sous forme de puissance de deux en utilisant les chiffres 0 et 1. Communément utilisée dans les ordinateurs car les valeurs 0 et 1 peuvent facilement être représentées pour des grandeurs physiques sous différentes formes, comme la présence ou l'absence de courant, une tension positive ou négative, ou un point blanc ou noir sur l'écran.

bit : C'est un des deux chiffres binaires (0 ou 1) ; la plus petite unité d'information qu'il est possible de traiter avec un ordinateur, consistant en un simple choix, comme oui ou non, marche ou arrêt, positif ou négatif, quelque chose ou rien. Voir aussi **hexadécimal**.

blanc : Caractère de texte dont la représentation imprimée est un espace ou blanc ; on le tape sur le clavier en enfonçant la touche Espace .

boucle : Portion de programme exécutée de façon répétitive jusqu'à ce que la limite soit atteinte.

boucle emboîtée : C'est une boucle contenue dans le corps d'une autre boucle et qui est exécutée de façon répétée à chaque passage dans la boucle qui la contient.

boucle infinie : Portion de programme qui répète la même séquence d'action indéfiniment.

branchement : Sert à renvoyer l'exécution d'un programme à une ligne autre que celle qui suit en séquence.

branchement conditionnel : Branchement qui a lieu si une condition est satisfaite. Voir **branchement inconditionnel.**

branchement inconditionnel : Branchement qui ne dépend pas d'une condition.

bug ; Erreur dans un programme, erreur qui entraîne un fonctionnement incorrect du programme. (Dans le langage courant : bug = punaise).

caractère : Lettre, chiffre, signe de ponctuation ou tout autre symbole, utilisés en impression ou affichage d'information sous forme lisible.

caractère avis : Caractère de texte affiché à l'écran pour aviser l'utilisateur d'une action donnée. Il sert souvent aussi à identifier le programme ou la partie du système qui occasionne l'avis ; par exemple le caractère d'avis] est utilisé par l'interpréteur BASIC Applesoft, > par le BASIC entier, et * par le programme moniteur.

caractères de contrôle : Ils permettent de contrôler ou de modifier la façon dont l'information est imprimée ou affichée. Les caractères de contrôle ont des codes ASCII entre 0 et 31 et sont tapés sur le clavier de l'Appel *II*e en maintenant enfoncée la touche Ctrl pendant que l'on appuie sur une autre touche de caractère. Par exemple le caractère Ctrl - M (Code ASCII 13) veut dire ''retour en début de ligne'' et est équivalent de la touche ↵ .

Carte Texte 80 Colonnes Apple *II*e : Une carte périphérique fabriquée et vendue par Apple Computer, Inc. Elle se branche dans le connecteur auxiliaire de l'Apple *II*e et convertit de 40 à 80 colonnes la largeur de l'affichage du texte de l'ordinateur.

Carte Texte 80 Colonnes Etendue Apple *II*e : Une carte périphérique faite et vendue par Apple Computer, Inc. Elle se branche dans le connecteur auxiliaire de l'Apple *II*e et convertit de 40 à 80 colonnes la largeur de l'affichage du texte de l'ordinateur, étendant sa capacité de mémoire de 64K octets.

catalogue (catalog) : Liste de tous les fichiers mémorisés sur un disque. Est aussi appelé parfois répertoire.

chaîne de caractères : Quantité d'informations formée d'une suite de caractères de texte. Les deux exemples suivants sont des chaînes de caractères : ''flipping frogs endanger widow's mite'' et ''(A + B)/973 = 14''.

chaîne vide : Chaîne de caractères ne contenant pas de caractères.

charger : Transférer des informations d'un périphérique de stockage (comme une disquette) dans la mémoire centrale pour les utiliser ; par exemple transférer un programme en mémoire pour exécution.

chiffre : (1) Un des caractères 0 à 9 utilisés pour exprimer des nombres sous forme décimale. (2) Un des caractères utilisés pour exprimer un nombre d'une autre façon, par exemple 0 et 1 en binaire ou 0 à 9 et A à F en hexadécimal.

clavier : Ensemble des touches intégrées à l'ordinateur Apple *IIe*, semblable à un clavier de machine à écrire, pour entrer les informations vers l'ordinateur.

code : (1) Chiffre ou symbole utilisé pour représenter un fragment d'information sous une forme compacte facile à traiter. (2) Les instructions constituant un programme.

colonne : Arrangement vertical de points graphiques ou de caractères sur l'écran.

commande : Communication de l'utilisateur vers l'ordinateur (habituellement tapée à partir du clavier) entraînant une action immédiate.

concaténer : Littéralement « chaîner ensemble ». Sert à transformer deux chaînes de caractères (ou plus) en une, plus longue, contenant tous les caractères des chaînes d'origine.

Contrôle : Ordre dans lequel les instructions d'un programme sont exécutées.

crash : Arrêt inattendu du travail, avec possibilité d'endommagement ou de destruction des informations en traitement.

CRT : Voir **tube cathodique.**

curseur : Marqueur ou symbole affiché sur l'écran qui signale la position où la prochaine action prendra effet, ou bien, l'endroit où le prochain caractère tapé à partir du clavier s'affichera. Les curseurs sont habituellement représentés par un carré blanc, un tiret sous le caractère ou un petit damier clignotant.

debug : Opération permettant de localiser et de corriger sur un système une erreur ou la cause d'un problème ou d'un mauvais fonctionnement. Typiquement utilisé pour les problèmes de logiciel. Voir **mauvais fonctionnement.**

décimal : Représentation courante des nombres exprimés en puissance de 10 en utilisant les chiffres de 0 à 9.

défaut : Valeur, action, activation utilisée automatiquement par un ordinateur quand aucune autre information explicite ne lui a été fournie. Par exemple, si une commande de déroulement de programme à partir d'un disque n'identifie pas le lecteur de disque à utiliser, le système d'exploitation disque choisira automatiquement le lecteur qui était utilisé dans l'opération précédente.

défilement : Sert à changer les contenus de tout ou partie de l'affichage de l'écran en décalant l'information vers une extrémité (le plus souvent vers le haut) de façon à faire de la place pour les nouvelles informations apparaissant à l'autre bout (le plus souvent vers le bas). Ceci produit un effet semblable au déplacement d'un rouleau de papier devant une fenêtre de visualisation fixe. Voir **fenêtre.**

délimiteur : Caractère utilisé comme ponctuation pour marquer le début ou la fin d'une séquence de caractère et qui, par conséquent n'est pas considéré comme faisant partie de la séquence. Par exemple, les guillemets (") sont utilisés comme délimiteurs par Applesoft pour la chaîne de caractères D, O et G : "DOG" n'inclut pas les guillemets. En Français écrit, l'espacement est utilisé comme délimiteur entre les mots.

démarrage à chaud : Acte de réinitialiser l'Apple *II*e bien qu'il soit déjà sous tension, sans recharger le système d'exploitation dans la mémoire centrale et souvent sans perdre le programme ou l'information déjà en mémoire. Voir **démarrage à froid.**

démarrage à froid : C'est la procédure de démarrage de l'Apple *II*e quand on vient de le mettre sous tension la première fois (ou les suivantes) en chargeant le système d'exploitation dans la mémoire principale, puis en chargeant le programme et en l'exécutant. Voir **démarrage à chaud.**

dimension : Taille maximum de l'un des indices d'un tableau.

disque : Support de stockage d'information consistant en une surface plate, circulaire et magnétique, sur laquelle l'information peut être enregistrée sous forme de petits domaines magnétisés de la même façon que les sons sont enregistrés sur une bande.

disque amorce : Voir **disque d'initialisation.**

disque de démarrage : Disque contenant le logiciel sous la forme requise pour être chargé dans la mémoire de l'Apple *II*e pour rendre le système opérationnel. On dit parfois aussi disque amorce. Voir **amorce.**

disque flexible : Disque en matière plastique flexible ; souvent appelés disques « floppy ».

disque « floppy » : Voir **disque flexible.**

disquette : Terme utilisé parfois pour les petits disques flexibles (5 pouces 1/4) utilisés sur le lecteur Apple Disk II.

données : Informations spécialement utilisées ou traitées par un programme.

DOS : Voir **système d'exploitation disque.**

écran : Voir **écran afficheur.** Porte de visualisation.

écran afficheur : Le panneau en verrre ou en matière plastique à l'avant d'un appareil d'affichage sur lequel les images sont affichées.

écrire : Transférer des informations du micro-ordinateur vers une destination externe (comme un lecteur de disque, une imprimante, un modem) ou à partir du processeur vers une destination qui lui est externe (comme la mémoire centrale).

édite : Sert à changer ou à modifier. Par exemple, pour insérer, déplacer, replacer, ou changer un texte dans un document.

élément : Un objet appartenant à un ensemble ou une collection ; plus précisément, une des variables individuelles constituant un tableau. Voir aussi **indice.**

emplacement : Voir **emplacement mémoire.**

emplacement mémoire : Unité de la mémoire principale qui est identique par une adresse et peut retenir un seul côté d'information d'une taille déterminée. Dans l'Apple *IIe* un emplacement mémoire contient un octet ou huit bits d'information.

encoche de permission d'écriture : Découpe rectangulaire sur le côté du disque qui donne la permission d'écrire sur ce dernier. S'il n'y a pas d'encoche de permission d'écriture, ou si elle est recouverte par un autocollant de protection contre l'écriture, l'information peut être lue sur le disque mais ne peut pas être écrite.

entier : C'est un nombre sans partie fractionnaire ; il est représenté à l'intérieur de l'ordinateur en virgule fixe. Voir **nombre réel.**

entrée : (1) Information transférée à l'intérieur d'un ordinateur à partir d'une source externe quelconque comme un clavier, un lecteur de disque ou un modem. (2) L'acte ou l'accomplissement du transfert d'une telle information.

E/S : Entrée/Sortie. Transfert d'information vers l'intérieur ou vers l'extérieur d'un ordinateur. Voir **entrée, sortie.**

exécution : (1) Mener à bien, réussir une action spécifique ou une suite d'actions comme celles décrites par un programme. (2) Chargement d'un programme dans la mémoire centrale à partir d'un périphérique de stokage, comme un disque pour l'exécuter.

exécution différée : C'est la sauvegarde d'une ligne de programme Applesoft, qui sera exécutée plus tard, faisant partie d'un programme complet ; cela arrive chaque fois qu'une ligne est tapée avec un numéro de ligne. Voir **exécution immédiate.**

expression : C'est une formule dans un programme décrivant un calcul qui devra être exécuté.

expression arithmétique : Combinaison de nombres et d'opérateurs arithmétiques (par exemple 3 + 5) qui indiquent qu'une opération doit être effectuée.

fenêtre : Partie ou ensemble d'informations (document, image, feuille de travail) qui est visible sur une porte de visualisation sur l'écran. Voir **porte de visualisation.**

fenêtre de texte : Zone de l'écran de l'Apple *II*e à l'intérieur de laquelle du texte est affiché et où il peut défiler.

fichier : Ensemble d'informations stockées comme une unité identifiée sur un support périphérique de stockage, tel un disque.

fonction : Calcul préprogrammé qui peut être exécuté sur demande à n'importe quel endroit du programme.

format : (1) Forme dans laquelle l'information est organisée ou présentée. (2) Sert à spécifier ou à contrôler le format de l'information. (3) Préparation d'un disque vierge à la réception des informations en divisant sa surface en pistes et secteurs ; on dit aussi initialiser.

graphiques : (1) Information présentée sous forme de dessins ou d'images. (2) Affichage de dessins ou d'images sur l'écran d'affichage d'un ordinateur. Voir **texte.**

graphique basse résolution : Affichage de graphiques sur l'écran de l'Apple *II*e sous forme d'un tableau de 16 couleurs, large de 40 colonnes et haut de 48 lignes. Quand la fenêtre texte est en service, le graphique basse résolution visible est de 40 par 40 points.

graphique haute résolution : L'affichage de graphiques sur l'écran d'affichage de l'Apple *II*e en un tableau de points de 6 couleurs, large de 280 colonnes et de 192 lignes. Lorsque l'on utilise la fenêtre texte, l'affichage graphique haute résolution visible est de 280 par 160 points.

hexadécimal : Représentation de nombres en terme de puissance de seize, en utilisant seize chiffres 0 à 9 et A à F. Les nombres hexadécimaux sont plus faciles à lire et à comprendre que les nombres binaires mais peuvent facilement et directement être convertis sous forme binaire : chaque chiffre hexadécimal correspond à une séquence de quatre chiffres binaires ou bits.

indice : Nombre servant d'index et utilisé pour identifier un élément particulier dans un tableau.

information : Fait, concept ou renseignement représenté sous forme organisée.

initialisation : (1) Positionner dans un état initial ou à une valeur de départ pour préparer un traitement. (2) Préparer un disque vierge à recevoir des informations en divisant sa surface en pistes et secteurs ; on dit aussi formater.

instruction : Elément de programme dans un langage de haut niveau spécifiant une action devant être exécutée par l'ordinateur, cela correspond généralement à plusieurs instructions en langage machine.

Integer BASIC : Voir **BASIC entier.**

interface : On nomme ainsi le matériel, les règles ou les conventions au moyen desquels une partie du système communique avec une autre vidéo inverse. C'est lorsque l'affichage de texte sur l'écran se fait sous forme de points noirs sur fond blanc (ou tout autre couleur brillante) à la place de points blancs sur fond noir.

K : Deux puissance dix, ou 1024 (vient de la racine grecque kilo voulant dire mille). Par exemple 64K veut dire 64 FOIS 1024, soit 65536.

kilo octet : Unité d'information de 1K (1024) octetc ou 8 K (8192) bits, voir K.

langage : Voir **langage de programmation.**

langage assembleur : Langage de programmation de bas niveau dans lequel les instructions spécifiques en langage machine sont écrites sous forme symbolique plus facilement compréhensible par un programmeur que le langage machine lui-même.

langage de bas niveau : Langage de programmation relativement proche de la forme sous laquelle le processeur de l'ordinateur peut directement travailler. Le langage assembleur en est un exemple.

langage de haut niveau : Langage de programmation relativement facile à comprendre. FORTRAN, BASIC et Pascal sont tous des exemples de langages de haut niveau. Une instruction dans un langage de haut niveau correspond généralement à plusieurs instructions en langage machine. Synonyme : langage évolué.

langage de programmation : Ensemble de règles ou de conventions pour l'écriture des programmes.

langage de programme : Unité de base d'un programme en Applesoft, elle contient une ou plusieurs instructions séparées par des doubles points (:).

langage d'ordinateur : Voir **langage de programmation.**

langage machine : Forme sous laquelle les instructions pour un ordinateur sont stockées pour être directement exécutées par le processeur de l'ordinateur. Chaque type de microprocesseurs (comme le 6502 utilisé dans l'Apple *II*e) ont leur propre forme de langage machine.

lecteur de disque : Appareil périphérique qui écrit et lit l'informatio sur la surface d'un disque magnétique.

lecteur DISK II : Modèle de lecteur de disque fabriqué et vendu par Apple Computer, Inc., pour être utilisé par l'ordinateur Apple *II*e ; i utilise des disques souples « floppy » de 5 pouces 1/4.

liaison : Suite automatique du texte de la fin d'une ligne au début de la suivante comme sur l'écran ou l'imprimante.

lieu de démarrage : Voir **démarreur.**

ligne : Arrangement horizontal de caractères, d'espaces, ou de points graphiques sur l'écran.

limite en caractères : Le nombre maximum de caractères autorisés pour une instruction Applesoft est 255.

lire : Transférer de l'information dans la mémoire de l'ordinateur à partir d'une source extérieure à l'ordinateur (par exemple un lecteu de disque ou un modem) ou à l'intérieur du processeur à partir d'une source extérieure ou processeur (comme le clavier ou la mémoire centrale).

logiciel : Partie d'un système informatique consistant en programme, ce qui détermine ou contrôle le comportement de l'ordinateur. Voir **matériel** et **logiciel intégré.**

logiciel intégré : C'est une partie du système de l'ordinateur constituée de programmes figés en mémoire morte. De tels programmes (par exemple l'interpréteur Applesoft et le programme moniteur Apple *II*e) sont intégrés, en usine, dans l'ordinateur ; ils peuvent être exécutés à tout moment mais ne peuvent être ni modifiés ni effacés à partir de la mémoire centrale. Voir **matériel** et **logiciel.**

matériel : Tous les composants physiques d'un système (électroniques et mécaniques). Voir **logiciel** et **logiciel intégré.**

mémoire : Composant matériel d'un système qui peut stocker l'information pour la restituer plus tard. Voir **mémoire principale** et **mémoire à accès aléatoire.** Mémoire uniquement lisible, mémoire lisible et inscriptible.

mémoire à accès aléatoire : Mémoire dans laquelle le contenu des emplacements individuels peut être repéré dans un ordre arbitraire ou aléatoire. Ce terme est souvent utilisé incorrectement pour parler d'une mémoire vive mais, à strictement parler, on peut avoir accès de façon aléatoire à la fois à une mémoire morte et à une mémoire vive. L'accès aléatoire signifie que chaque cellule de stockage n'a qu'une seule adresse et qu'il n'y a qu'une seule façon pour chacune d'elles d'être lue ou écrite.

mémoire morte : Mémoire dont le contenu peut être lu mais non écrit. On l'utilise pour stocker du logiciel intégré. L'information est écrite en mémoire morte une fois, en usine ; elle reste ensuite de façon permanente même si l'ordinateur n'est pas alimenté et elle ne peut plus être ni effacée, ni changée. Voir **mémoire vive** et **mémoire à accès aléatoire.**

mémoire principale : Partie mémoire d'un système qui est à l'intérieur de l'ordinateur et dont le contenu est directement accessible par le processeur.

mémoire vive : Mémoire dont le contenu peut être à la fois lu et écrit ; on l'appelle parfois de façon abusive mémoire à accès aléatoire ou RAM. L'information contenue dans une mémoire à lecture et écriture est effacée quand on coupe l'alimentation de l'ordinateur, elle est complètement perdue sauf si le contenu a été sauvé auparavant sur un support plus durable comme un disque. Voir **mémoire morte** et **mémoire à accès aléatoire.**

menu : Liste de choix présentés par un menu, habituellement sur un écran, à partir desquels l'utilisateur peut choisir.

message d'avis : Message affiché sur l'écran pour aviser l'utilisateur d'une action donnée. Appelé aussi message d'accueil (demande de saisie d'une information)

message d'erreur : Message affiché ou imprimé pour aviser l'utilisateur d'une erreur ou d'un problème dans l'exécution d'un programme.

mode : Etat d'un ordinateur ou d'un système qui détermine son comportement.

mode escape : Etat de l'ordinateur Apple *IIe*, entré en enfonçant la touche Esc, dans lequel certaines touches du clavier prennent des significations spéciales pour positionner le curseur et contrôler l'affichage du texte à l'écran. Ce mode est parfois appelé « évasion » sur certains documents Apple.

Mode exécution immédiate : Exécution d'une ligne de programme en Applesoft dès qu'elle a été tapée ; cela se produit quand une ligne est tapée sans numéro de ligne. C'est un avantage particulier d'Applesoft, qui n'existe pas dans de nombreux autres langages de programmation. Cela veut dire que vous pouvez essayer l'une après l'autre chaque instruction pour voir sur le champ son résultat. Voir **exécution différée.**

mode immédiat : Mode de fonctionnement dans lequel, chaque f de ligne (caractérisée par le caractère retour de chariot) provoque l'exécution immédiate de l'instruction ou des instructions contenues dans cette ligne. Ce mode s'obtient avec des lignes n comportant pas de numéro de ligne.

mode programme : (encore appelé exécution différée). Mode de fonctionnement dans lequel les lignes de programmes comporten toutes un numéro de ligne. La fin de ligne (retour chariot) ne provoque pas l'exécution des instructions contenues dans cette ligne. Ce mode permet de construire un programme dont l'exécution sera lancée au moyen de la commande RUN.

modulateur radio fréquence : Appareil pour convertir les signaux vidéo produits par un ordinateur dans une forme acceptable par u téléviseur.

modulateur RF : Voir **modulateur radio fréquence.**

moniteur : Voir **moniteur vidéo.**

moniteur vidéo : Appareil d'affichage capable de recevoir des signaux vidéo, en direct seulement, et qui ne peut recevoir de signaux diffusés comme la télévision commerciale. Il peut être directement connecté sur l'Apple *II*e comme terminal d'affichage. Voir **téléviseur.**

mot clé : Mot spécial ou séquence de caractères qui identifie une instruction ou une commande particulière, comme RUN ou PRINT

mot réservé : Mot ou suite de caractères réservés par un langage de programmation pour une utilisation particulière et, par conséquent, pas disponible comme nom de variable dans un programme.

nombre réel : Nombre avec une partie fractionnaire représentée à l'intérieur de l'ordinateur en virgule flottante. Voir **entier.**

nom de fichier : Le nom sous lequel un fichier est stocké.

normal : Se dit de l'affichage sur l'écran lorsqu'il est constitué de points blancs (ou d'une seule couleur) sur fond noir. Voir **inverse.**

notation scientifique : Méthode de représentation des nombres sous forme de puissance de dix, très utilisée pour exprimer des nombres qui peuvent varier dans de larges proportions, de très petits à très grands. Par exemple le nombre d'atomes dans un gramme d'hydrogène est approximativement 6,02E23 ce qui signifie 6,02 fois dix à la puissance 23. (La lettre E signifie « exposant »). Le nombre est ainsi plus facile à comprendre que sous la forme 602000000000000000000000. Applesoft utilise cette méthode pour afficher avec précision les nombres avec plus de neuf chiffres.

numéro de ligne : Nombre identifiant une ligne programme dans un programme Applesoft. Les numéros de lignes sont nécessaires pour les instructions différées.

Octet : Unité d'information consistant en un nombre fixe de bits. Sur Apple *II*e, un mot contient huit bits et peut contenir n'importe quelle valeur entre 0 et 255. Chaque caractère en code ASCII peut être représenté par un mot avec un bit supplémentaire à gauche.

Opérateur : Symbole ou suite de caractères comme +, OR, AND, spécifiant une opération devant être accomplie sur une ou plusieurs valeurs pour donner un résultat. Voir **opérateur arithmétique, opérateur relationnel, opérateur logique, opérateur monadique** et **opérateur diadique.**

opération arithmétique : C'est un opérateur comme le +, qui combine des valeurs numériques pour donner un résultat numérique. Voir **opérateur relationnel** et **opérateur logique.**

opérateur binaire : C'est un opérateur qui combine deux opérandes pour donner un résultat ; par exemple + est un opérateur arithmétique binaire, < est un opérateur binaire relationnel, et OR est un opérateur binaire logique. Voir **opérateur unaire.**

opérateur logique : Opérateur comme AND qui combine des valeurs logiques pour donner un résultat logique. Voir **opérateur arithmétique** et **opérateur relationnel.**

opérateur relationnel : Opérateur comme > qui compare les valeurs numériques pour donner un résultat logique. Voir **opérateur arithmétique, opérateur logique.**

opérateur unaire ou monadique : Opérateur dont l'action s'exerce sur une seule opérande, le signe moins (−) dans un nombre négatif comme −6 est un opérateur arithmétique unaire. Voir **opérateur binaire** ou diadique.

opérations arithmétiques : Ce sont les cinq actions que Applesoft peut effectuer avec des nombres : addition, soustraction, multiplication, division et élévation à la puissance.

ordinateur : Appareil électronique pour effectuer des séquences prédéfinies (programmées) à haute vitesse et avec une grande précision. C'est une machine utilisée pour stocker, transférer et transformer l'information.

page : (1) Le contenu d'un écran rempli d'informations, ce qui consiste sur l'Apple *II*e en 24 lignes de 40 ou 80 caractères chacunes. (2) Une zone de la mémoire centrale contenant du texte ou des informations graphiques devant être affichée sur l'écran.

pas : Quantité de laquelle une variable est incrémentée ou décrémentée à chacun de ses passages dans une boucle.

périphérique : Dispositif extérieur d'une unité de traitement, il peut s'agir d'un élément physique (comme un organe périphérique) ou d'un élément logique (comme une carte périphérique).

porte de visualisation : Tout ou partie de l'écran de visualisation, utilisé par un programme d'application pour afficher une portion de l'information (comme un document, une image, une feuille de travail) sur laquelle le programme travaille. Voir **fenêtre.**

précédence : Ordre dans lequel les opérateurs sont appliqués dans l'évaluation d'une expression.

processeur : Composant matériel d'un ordinateur qui effectue le travail en exécutant directement les instructions qui se présentent sous forme de langage machine et stockées dans la mémoire centrale.

processeur unité centrale : Voir **processeur.**

programmation : Action d'écrire des programmes.

programme : (1) Ensemble d'instructions, conformes aux règles et aux conventions d'un langage de programmation particulier, décrivant les actions à accomplir par l'ordinateur pour effectuer une tache donnée. En Applesoft, c'est une suite de lignes de programmes avec chacune un numéro de ligne différent.

programme d'application : C'est un programme qui met les ressources et les possibilités de l'ordinateur au service d'une tâche spécifique, comme le traitement de texte, la gestion d'une base de donnée, du traitement graphique ou des télécommunications. Voir **programme système.**

programme moniteur : Programme intégré dans l'Apple *II*e en mémoire morte, utilisé pour vérifier directement ou changer les contenus de la mémoire centrale et pour travailler au niveau langage machine avec le processeur.

programme qui boucle (pendu) : On dit d'un programme qu'il boucle lorsqu'il tourne sur lui-même indéfiniment, sans accomplir de tâche utile.

programmer : Ecrire un programme.

programmeur : Auteur d'un programme ; quelqu'un qui écrit des programmes.

protection d'écriture : Fait de protéger les informations contenues dans un disque en couvrant l'encoche de permission d'écriture avec un autocollant de protection, empêchant d'écrire toute nouvelle information sur le disque.

puce : Petit morceau de matériau semi-conducteur (habituellement du silicium) sur lequel est fabriqué le circuit intégré. Le mot puce désigne, à proprement parler, seulement le morceau de silicium, mais il est souvent utilisé pour désigner le circuit intégré dans son boîtier. Voir **circuit intégré.**

punaise : Voir **bug.**

RAM : Voir **mémoire à accès aléatoire.**

répertoire : Voir **catalogue.**

ROM : Voir **mémoire morte.**

sauver : Transférer l'information de la mémoire centale vers un support périphérique de stockage pour une utilisation ultérieure.

séquence escape : C'est une suite de touches enfoncées, commençant pour la touche Esc, utilisée pour positionner le curseur et contrôler l'affichage du texte à l'écran. Voir **mode escape.**

sortie : (1) Information transférée d'un ordinateur vers une destination externe quelconque comme l'écran, un lecteur de disque, une imprimante ou un modem. (2) Le fait ou le traitement d'une telle information.

sous-programme : Participation d'un programme qui peut être exécutée sur demande en n'importe quel endroit du programme principal. Dès la fin de son exécution, il redonne le contrôle au programme principal au point qui suit l'endroit où il a été appelé.

syntaxe : Règle régissant la structure des instructions dans un langage de programmation.

système d'exploitation : Système logiciel qui organise les ressources et les possibilités d'un ordinateur et les rend disponibles à l'utilisateur ou à des programmes d'application se déroulant sur l'ordinateur.

système d'exploitation de disque : Logiciel optionnel pour l'Apple *II*e qui permet à l'ordinateur de contrôler et de communiquer avec un ou plusieurs lecteurs de disques, Disk II.

système informatique : Un ordinateur et le matériel associé sous forme de logiciel.

tableau : C'est une collection de variables référencées par le même nom et distinguées entre elles au moyen d'indices numériques. Chaque variable dans le tableau peut être adressée de façon indépendante en utilisant son indice propre et unique.

tampon (buffer) : Espace de la mémoire de l'ordinateur affectée à une tâche spécifique, comme le stockage d'informations graphiques, l'affichage sur l'écran ou des informations de type caractères de texte à partir d'un périphérique. Cet espace est souvent utilisé comme zone de stockage intermédiaire pour transférer des informations entre les constituants travaillant à des vitesses différentes, comme l'unité centrale de l'ordinateur et une imprimante ou un lecteur de disque. Les informations peuvent être stockées dans le tampon par un des sous-ensembles et être lues par un autre à une vitesse différente.

téléviseur : Appareil d'affichage capable de recevoir des signaux vidéo diffusés (comme la télévision commerciale) au moyen d'une antenne. Il peut être utilisé avec un modulateur radio fréquence comme écran pour l'Apple *II*e. Voir **moniteur vidéo.**

texte : (1) Information présentée sous forme de caractères lisibles. (2) Affichage de caractères sur l'écran de l'Apple *II*e. Voir **Graphique.**

utilisateur : Personne utilisant ou contrôlant un ordinateur.

valeur : Quantité d'informations qui peut être stockée dans une variable comme un nombre ou une chaîne de caractères.

variable : (1) Emplacement dans la mémoire de l'ordinateur où une valeur peut être stockée. (2) Symbole utilisé dans un programme pour représenter un tel emplacement. Voir **constante.**

vidéo : (1) Forme utilisée pour transmettre de l'information sous forme d'image pour être affichée sur l'écran d'un tube cathodique. (2) Information organisée ou transmise sous forme vidéo.

virgule fixe : Méthode de représentation des nombres à l'intérieur de l'ordinateur dans laquelle la virgule (plus exactement la représentation binaire de la virgule) occupe toujours la même place à l'intérieur du nombre. Généralement, on considère que la virgule est positionnée à l'extrémité droite du nombre, ainsi le nombre est considéré comme étant entier. Les nombres en virgule fixe d'une longueur donnée couvrent une plus petite étendue que ceux en virgule flottante, mais ils sont plus précis. Voir **virgule flottante.**

virgule flottante : Méthode de représentation des nombres à l'intérieur de l'ordinateur dans laquelle la virgule (plus exactement la représentation binaire de la virgule) peut flotter à des positions différentes à l'intérieur du nombre. Un certain nombre de bits dans le nombre lui-même est utilisé pour repérer la position de la virgule. Les nombres en virgule flottante d'une longueur donnée couvrent une plus grande étendue que ceux en virgule fixe de la même longueur, mais ils sont moins précis. Voir **virgule fixe.**

Index

A

Accent circonflexe 10
Addition 10
Adresse 17
Adresse d'identification 170
Affichage
  d'une suite de zéros 11
  40 colonnes 175
  80 colonnes 175
Afficheur
  différence entre graphique et texte 66
  horizontalement, utilisation de VTAB 66
  verticalement, utilisation de VTAB 66
AGE programme 48
Aide 167
Aide (écrans) 185
Alternance des couleurs en mode graphique 114, 118
Animation en programmation 89-122
Antislash (\) 6
Appellation des programmes 47-48
Apple *IIe*
  Carte 80 colonnes pour texte 181, 183
  Haut parleur 97
  Interface standard 181
  Guide de l'utilisateur *
Applesoft introduction 3
Arguments dans des chaînes de caractères 126
Arithmétique
  Expressions 31
  Opérations 10
    à des vitesses différentes 99
    en combinaison 11
Arrondis de nombre 12
Assertion 53
Astérisque (*) 10
ATN 161

B

?BAD SUBSCRIPT ERROR 139, 164
BASIC X
  Applesoft, caractère d'avis (]) 170.
  Entier 146
  Entier Apple, caractère d'avis (>) 146
Binaire 146
  système numérique 50
Blancs dans les lignes de programme, comment les éviter 80
Blocs de construction 172
Boîte grise *
Boîtes noires 180, 181, 183
Boucles 52
  utilisation pour balayer une chaîne de caractères 127
  utilisation avec l'instruction GOTO 42-44
Boucle déterminée (IF...THEN) 52
Boucle emboîtée 62, 62
Boucle enchevétrée 62, 62
Boucle imbriquée 62
Boucle indéterminée (GOTO) 42-45
Boucle infinie 58
BREAK IN 43

C

Calculs 9-12, 27, 33
CALL 133, 145
?CAN'T CONTINUE ERROR 164
Caractères
  limites 6, 7
  chaînes 123
Carte texte 80 colonnes 183, 187
  inactivation

Chaînes de caractères 124, 131
avec les tableaux 123, 140
caractères les composant 124
duplication 128
entraînement à la programmation 126
et utilisation de boucles 127
ponctuation et espaces à l'intérieur 126
réunions
variables
comment les mettre à zéro 124, 129
manipulation 124
noms 123
vides 124, 129

Changement de travail (voir édition)
Changements (voir édition)
Chargement d'un programme à partir du disque 54
Chasse aux erreurs 110, 134, 155, 168, 188
CHR$ (instruction) 146, 202
CLEAR (instruction) 129, 46
Clés de déplacement du curseur 85
Code ASCII 145, 146
Code IX
Colonnes 14
Colonne verticale 14
COLORBOUNCE (programme) 89-100, 134-135
ajoutons du bruit 100
listage 93
comment il travaille 94
COLORBOUNCESOUND (programme) 135
COLOR= (instruction) 14-18, 17
comment créer une impression visuelle 89
graphique basse résolution 14-18, 20-23
groupes 17, 146
table 146
Combinaisons d'instruction 56
Commande CATALOG 47, 145, 202
Commande échappement 76
Comment corriger les erreurs de frappe 73
Comment ne pas produire des zéros 103
Comment se débarrasser de lignes de programme 83, 147
Commutation mode graphique mode texte 18-23
Compilation 202
Compression de programmes 201
Compteurs 60, 175
Concaténation 130
Conclusion 140
Conditions : comment déterminer le ''vrai'' 49
Construction de jeux 89
Dés 103
CONT 44, 147
CONTROL 43
Contrôle des espacements dans votre programme 64-69
CONVERTER (programme) 172, 211
Coordonnées cartésiennes 16
Coordonnées horizontales 113
Coordonnées verticales 113
Copie avec la touche ← 74
Correction de lignes (voir édition)
Couleur 17
Ctrl - C 167
Ctrl - C 43
Ctrl - Reset 167
Ctrl - S 83
Ctrl - X 82
Curseur 3
déplacement vers la droite sans marques de blancs 80
en graphique haute résolution 114
Curseur clignotant 182
Curseur, utilisation pour les entrées 80

D

DATA (instruction) 147
DECIMAL (programme) 132
Défaut 48
Défilement 44, 138, 155
DEL (ordre) 83, 147
Del touche 83
DELETE (commande) 83, 145, 148

Dépannage 5
Dessin 118, 150, 151
  chevaux 104
  diagonales 104
  graphique haute résolution 116
  lignes 20-23-104
  points 104
Deux points 90, 177
  utilisation des instructions combinées à REM 90
Dimensionnement de
  tableaux 175
  concepts 137
DIM (instruction) 136
DISK MENU 201
Disque initialisé 47
Distinguons les variables 26
Division 10
?DIVISION BY ZERO ERROR 164
Drapeaux fanions 191, 202
Duplication de chaînes de caractères 128

E

Ecran pour affichage 8, 157
Edition 4-6, 73-86, 85
  changement de texte 74-76
  comment se débarrasser de lignes 82-83
  histoire 84-85
  insertion de texte 77-81
  sommaire des possibilités 86
  travaux pratiques 74-81
Effacement, avec la touche ← 6,78
Eléments 136
Emplacement en mémoire 170
END 109, 148
Entier
  fonction 101
  impression 42
Epelons à l'envers 128
Esc - @ 76
Esc - E 76
Esc - F 76
Esc - touche 73, 167, 182
  alternance avec les touches A,B,C,D, 84
  en même temps que les touches I,J,K,M, 85
Espaces mémoires 24
Espacements en trop dans les lignes programmes
  comment les éviter 83
ET commercial (&) 160, 204
Exécution immédiate 37
Exécution (programme) 38-44
Exécution retardée 37-41
  avantages 41
  exécution d'instruction (erreurs dans l') 163
Exemples en Applesoft
  programme ALPHABET 127
  programme COLORBOUNCE 89
  programme COLORBOUNCESOUND 135
  programme COLOR LOOP 54
  programme DECIMAL 132
  programme HORSES 112
  programme MOIRE 118
Exponentiation 10

F

Faux 190
  condition 50
Fenêtre de défilement 176
Fenêtre de texte 14
  en mode graphique 57
Fichier caractère 146
Fonctions
  arithmétiques, intégrées 98
  INT 152
  LEFT$ 125, 153
  LEN 124, 153
  MID$ 125, 154
  PEEK 155
  RIGHT$ 125, 156
  RND 156, 125, 156
  STR$ 132, 157
  TAB 157
  VAL 131, 158
Fonction ASC 145
Fonctions de chaîne de caractères 124, 131
FOR instruction 148
  nécessité de lui coupler NEXT 62
Formatage 132, 201
?FORMULA TOO COMPLEX ERROR 164
FOR/NEXT
  boucle 59
  définition de l'étendue des variables 59
  instruction 59-63, 60

G

GET RETURN programme 183
GOSUB instruction 149, 173
  dans un programme principal 111, 113
GOTO instruction 42, 45, 149
Graphique 13-18, 89-122
  basse résolution 13-18
  comment dessiner des lignes 20, 23, 111, 118
  fenêtre texte 14, 57
  haute résolution 113-122
  grilles 14
  sous-programmes additionnels 111
Graphique haute résolution 114, 118
Grilles basse résolution coordonnées 23
GR instruction 14, 129
Guillemets (") 3, 11, 27
  dans les variables chaîne 124
  utilisation de 5

H

&H 204
Haut-parleur Apple *IIe* 97
HCOLOR= instruction 115
HGR instruction 114-119, 150
  utilisation de couleurs avec 118
  haute résolution (graphique) 113-119, 115, 150
HIMEM : 161
HLIN instruction 21
  syntaxe 20
HOME instruction 9-12, 150
HORSE programme 112
HPLOT instruction 116-119, 150
  combinaison de lignes 117
HTAB instruction 67-69, 151
  utilisation avec VTAB dans l'APPLESOFT SAMPLER 67
HUE programme 62
Humanisation des programmes 211

I

Identification par le système 184-186
IF (instruction) 53, 56
IF...THEN (instruction) 52, 151
?ILLEGAL DIRECT ERROR 164
?ILLEGAL QUANTITY ERROR 19, 68, 96, 105, 139, 164
Imbriquement à deux niveaux 62
Imbriquement sur trois niveaux 62
Impression fantaisie 8
Indices 137
Information additionnelle X
INPUT (instruction) 45-47, 94-96, 152, 177
  écriture 96
  exécution 46
  utilisation 46
INPUT (programme) 182
Insertion de texte
  dans une ligne existante 77
  instructions pas à pas 77
Instructions 4-7
  et commande 168
  pour construire des blocs dans les programmes 89
  pour contrôler 91
  pour exécuter 37
  pour mémorisation 40
  des raisons pour les combiner 91
  utilisation entr'elles 56
  pour parer les erreurs 6
  problèmes 5
  multiples par lignes 21
INT (fonction) 101, 152
INVERSE (instruction) 8, 152

L

Langage machine 50, 170
Lecteur de disque 48
Lecteur de disque n° 2, suggestion d'utilisation 48
  voir aussi second lecteur de disque
Lecteur de disque pris par défaut 48
LEFT ARROW touche 6,78
  en mode échappement 74
LEFT$ fonction 125, 153
LEN (fonction) 124, 153
LET instruction 24-29, 153
  définition de variable 24
  syntaxe 25, 27
  valeur 27

Lignes
différentes entre le listage et l'exécution 38
horizontales 15
Limitation des variables 52
Limites d'écran 92
Listage d'arrêt 153
Listage de programme d'arrêt 84
Listage programme
comment reprendre 153
comment voir le listage complet 57
LIST instruction 38-44, 153
numéro de lignes dans le listage 44
LOAD commande 49, 53, 145, 154
Logique booléenne 190
LOMEM 161
Longueur (fonction) 124
Longs programmes, comment en lister des portions 84

M

&M 204
MAGIC/MENU programme 171
Magnétophone à cassette (utilisation du) 47, 48, 154, 157, 168, X
Majuscules, utilisation pour les instructions 3
Manuel de références du programmeur en BASIC Applesoft X
Mémoire 38, 169
emplacement 170
étendue de l'adresse 98
Mémoire centrale 24
Mémorisation temporaire des programmes 38
Menus conviviaux 171
Menus (programme pour faire des) 184
Messages d'erreur (format pour) 163-166
MID$ (fonction) 125, 154
Mise hors tension 8
Mises en garde
⇩ touche enfoncée 3
démarrage 8
instruction FOR 62
mise hors tension 8
?SYNTAX ERROR 3
touche ↵ 14
Mode échappement 111
limites 76-77
règles d'utilisation 73-74
sommaire 86
travaux pratiques 74-81
Mode graphique basse résolution 14
Mode texte 23
MOIRE (programme) 118
Moniteur et caractère d'avis 170
Moniteur vidéo X
Mots clés 5
Mots réservés 26
table 160
Multiples instructions par lignes 90
Multiplication 10

N

NEW (commande) 37-38, 47, 57, 154
NEXT (instruction) 154
?NEXT WITHOUT FOR ERROR 165
Noir sur blanc 9
Nombres, affichage 42
Nombres aléatoires 100-104, 176
NORMAL (instruction) 8, 154
Notation scientifique 12
NOTRACE (instruction) 109, 154
Numéro de ligne 38, 59
espace à laisser entre 40

O

Octets 70
ON...GOSUB 204
ON...GOTO 155
OPEN APPLE (touche) 167, 176, 182
Ordinateur
langage IX
mise hors tension 8
programme IX
Ordre conditionnel 50, 52
règle 51
symboles 51
Ordre d'exécution des opérations arithmétiques 32
?OUT OF DATA ERROR 165
?OUT OF MEMORY ERROR 165

?OVERFLOW ERROR 165

P

Parenthèses 135
règles Applesoft 33
utilisation pour modifier l'ordre 32
Passage continu entre lignes 68, 77, 81
PEEK (fonction) 98, 155
avec variables 98
différents types de PEEK 99
Perluète (&) 160, 204
Petits nombres
arrondis 11
traitement 11
Placement de points, bonne résolution 116
PLOT 14-22, 155
coordonnées 15, 67-66
messages d'erreur 19
utilisé avec l'instruction HLIN 2
Plus grand ou égal à < > = > 50, 51
Plus grand que zéro 100

Q

Questions sans réponses 8

R

Radio fréquence
modulateur 18
RANDOM (programme) 102
READ (instruction) 147, 156
?REDIM'D ARRAY ERROR 139, 165
Références croisées avec les programmes utilitaires 203
Remarques (REM) 58
REM (instruction) 58, 59, 113
Remise en place des lignes 39
RENUMBER (programme) 203
Renumérotation des numéros de lignes 202
Répétition sans fin 108
RETURN (instruction) 156, 173
↵ (touche) 4, 13, 43
?RETURN WITHOUT GOSUB ERROR 107, 166, 173
RF modulateur utilisé avec un téléviseur 18
RIGHT ARROW (touche) → 6, 78
dans le mode d'échappement 74
RIGHT$ (fonction) 125-156
RND (fonction) 100, 156, 171
combinée avec des instructions graphiques 103
ROT= 160
RUN2 174
RUN (instruction) 38-44, 157
en face d'un numéro de ligne 61
pour démarrer à n'importe quelle ligne 44

S

SAVE (commande) 46-49, 145, 157
Sauvegarde de programmes en utilisant le DOS (Système d'exploitation disque) 47-3, 48
SCALE= 160
SCRAMBLER (programme) 171
SCRN 160
Second lecteur de disque 48, 49, 55
SHIFT ⇧ (touche) 3
Signe dollar
dans les variables chaînes 124
utilisation avec les variables 68
Signe égal (=)
Signe moins (−) 10
Signe moins en opérateur unaire (monadique) 31
Signe pause XI
Signe plus (+) 10, 130
Simulation d'une paire de dés 103
Si vous ou votre programme êtes à sec 167
Slash (/) trait oblique 10
SOLID APPLE (touche) 182
Sommaire de chapitre
chapitre 1 34
chapitre 2 70
chapitre 3 86
chapitre 4 120
chapitre 5 141

Sortie
en colonne 64
sans espacement entre mots 64
Sous-programmes 104, 106
blocs 173, 174, 180
pour afficher instructions 171
Soustraction 10
SPACE barre d'espacement 73
SPACE (programme), APPLESOFT SAMPLER 67
SPC 161
SPEED= 160
STEP (commande) 61
Stopper le listage du programme 84
STR$ (fonction) 132, 157
?STRING TOO LONG ERROR 124, 166
Suite d'instructions mémorisées 40
Symbole de mise en garde XI
Symboles utilisés dans ce manuel XI
Syntaxe 46
?SYNTAX ERROR 19, 26, 97, 124, 166
Système bouclé 167
Système de numérotation 16, 16, 68
en mode graphique 68
en mode texte 68
Système d'exploitation disque DOS 147
commandes DOS 145, 156
CATALOG 47, 202
DELETE 83, 145, 148
PFR$ 145, 156, 169
SAVE 48, 157

T

TAB (fonction) 66, 157, 161
suivie par un argument 66
Tableaux 135-139, 137, 148, 175
éléments de 136
à deux dimensions 137-138, 137
Table des symboles 51
Téléviseur connecté à l'Apple *//e* 18
Texte non formaté 201
TEXT (instruction) 158
TO 161
Touches à flèches 6
direction 86
Touche CAPS LOCK ⇩ 3
Tracé 17
TRACE (instruction) 108-110, 108, 158
Traitement 9-12, 27-33
ordre de 30
Transferts de programmes 53
?TYPE MISMATCH ERROR 131, 166

U

?UNDEF'D FUNCTION ERROR 166
?UNDEF'D STATEMENT ERROR 166
UP-ARROW (touche) ↑ 84
Utilisateurs 94
recherche d'erreurs 110

V

Valeur 15,27
incrémentation de variables 28
Valeurs numériques
virgule et point virgule dans leurs utilisations 65
VAL (fonction) 131, 158
Variable 24-30, 45, 68, 105, 110, 182
Appellations 24, 123, 175, 189, 153
Chaîne de caractères 123
définition 24, 153
définition avec l'instruction LET 24
incrémention de valeur 28
listage dans les programmes 182
mise en mémoire 24, 187-188
numérique 24, 123, 153
récapitulation des règles 29
table 187-188
utilisation avec

FOR/NEXT 59, 60
utilisation de l'instruction CLEAR 146
utilisation en indice 136
valeur non figée 92
Variable compteur déterminée 175
Variable indéterminée compteur 175
Variable numérique 123
Virgule 64, 65, 177
VLIN (instruction) 22, 57, 158
VLIN programme boucle 57
Vrai 190
VTAB (instruction) 66-69, 158

W

WELCOME (programme) 69

X

XPLOT 160

Avenue de l'Océanie
Z.A. de Courtabœuf - B.P. 131
91944 LES ULIS CEDEX
Tél. : 33 (6) 928.01.39 - Télex 692719

F030-0358-B