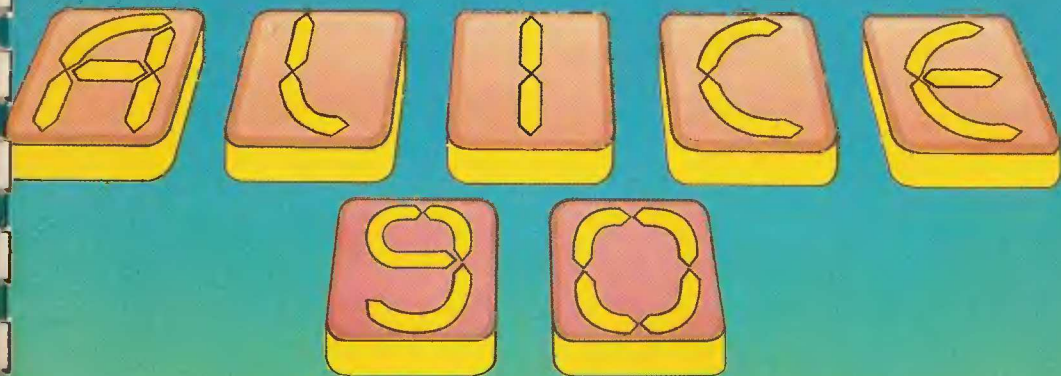
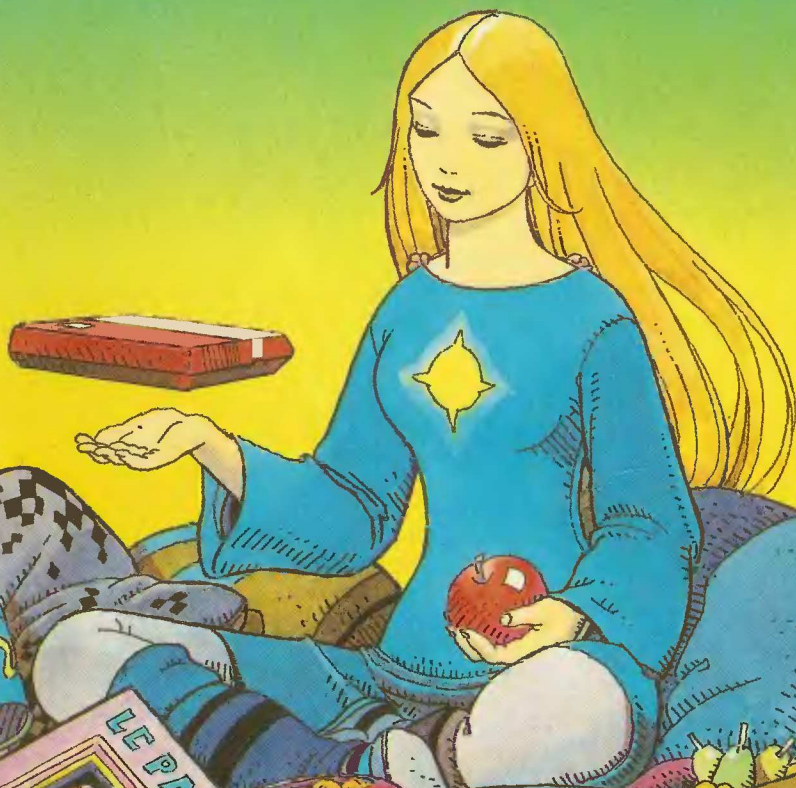


UNE MERVEILLE DE SIMPLICITÉ



DÉCOUVREZ LE BASIC



MATRA ET HACHETTE

ALICE

Ce guide est une nouvelle version du guide d'Alice,
écrit par Madame Anne-Marie Bourgeade
et remis à jour par la Société European Media Business.

Monsieur Olivier Zitoun
est l'auteur des programmes des pages 167 à 183 ;
Monsieur Guy Johnson
l'auteur des programmes des pages 183 à 190.

Dessins de Evelyne Drouhin.

L'illustration de couverture est l'œuvre de Moebius.

Le design d'ALICE 90 a été réalisé par DIÈDRE + DIMENSION 4.



DÉCOUVREZ LE BASIC

MATRA ET HACHETTE

Sommaire

Préface de Jean-Luc Lagardère	7
Avant-propos	9
Premier contact	
1. Mise en route	11
2. Premiers essais	23
Parlez BASIC	
Présentation	31
1. Votre premier programme	32
2. Alice vous interroge, répondez	46
3. Un peu de traitement	55
4. Comment sauvegarder vos programmes	73
5. L'art des répétitions	76
6. Et si... ..	88
7. Des données en file d'attente	100
8. La récréation en couleurs	109
9. Le boomerang et les aiguillages	124
10. Des données en libre service	130
11. Pour aller plus loin	140
Pour en savoir plus	
1. Corrigé des exercices	145
2. Qu'est-ce qu'un micro-ordinateur ?	158
3. Résumé des commandes, instructions et fonctions BASIC	162
4. Codes et tables	168
5. Messages d'erreur	173
6. Si ça ne marche pas	174
7. Spécifications techniques	176
8. Cahier de programmes	178
Lexique	200
Index	202

Préface

La micro-informatique arrive dans notre vie à grands pas, dans nos bureaux et nos usines, mais aussi dans nos écoles et nos maisons.

Nous avons été marqués par bien des développements technologiques, mais peu ont été porteurs d'un tel impact potentiel sur notre vie quotidienne. Nous sommes pris de vertige devant cet effort incessant pour repousser la frontière du possible, pour concevoir des micro-ordinateurs plus performants, plus petits, moins coûteux. Mais surtout, nous nous prenons à rêver devant le potentiel que représente la micro-informatique pour notre vie à tous :

- Pour notre travail débarrassé des tâches répétitives et fastidieuses.
- Pour la manière dont nous acquérons des connaissances nouvelles, à l'école ou chez nous.
- Pour nos loisirs.

En un mot pour le potentiel créatif qu'elle représente.

Pour la première fois peut-être, chacun de nous peut être acteur, intervenant et non pas sujet passif, chacun de nous peut créer, étendre les frontières de son esprit et non pas subir et se plier à des normes pré-définies.

Face à une pareille révolution, MATRA et HACHETTE ne pouvaient rester indifférents. MATRA est concerné du fait de ses compétences technologiques de pointe depuis les composants jusqu'aux satellites, de ses relations étroites avec les sociétés les plus avancées au monde dans ce domaine et de sa volonté d'apporter des solutions du futur aux besoins des hommes, en France comme à l'étranger.

HACHETTE ne serait pas fidèle à sa mission de mettre à la portée de tous les instruments de la connaissance, si elle n'offrait pas au côté des livres et de la presse, de nouveaux média, audiovisuel et micro-informatique.

C'est pourquoi, MATRA et HACHETTE se sont associés et ont mis en commun leurs ressources pour permettre aux Français d'avoir accès à une micro-informatique adaptée à leurs besoins.

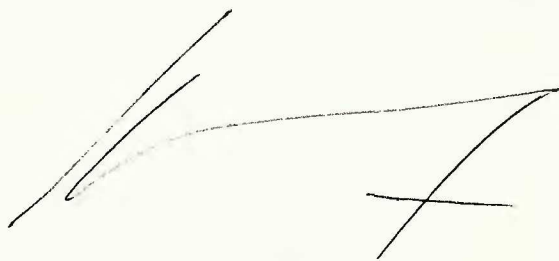
Certes, aujourd'hui, l'informatique, lourde de son passé, fait encore peur. Beaucoup d'entre nous, conscients de la nécessité, pour nous et pour nos enfants, de découvrir ce nouveau monde, sont d'abord rebutés par le vocabulaire technique, souvent anglo-saxon, l'aspect rébarbatif des machines et des manuels : ils hésitent, ils renoncent devant cet hermétisme. Quel dommage !

Ce que MATRA et HACHETTE réunis vous apportent aujourd'hui, c'est un moyen d'accès à l'informatique.

Pour vous, enfants, adolescents, adultes, qui êtes à la fois fascinés par cette révolution micro-informatique et rebutés par la difficulté de vous initier à ce nouveau monde, voici un outil de découverte, un moyen simple et facile de comprendre, de découvrir, de vous mettre à niveau :

un micro-ordinateur simple, facile à mettre en œuvre, agréable à utiliser et un manuel clair, détaillé, écrit dans un français aisément compréhensible. Avec eux, vous serez pris par la main, guidés, épaulés jusqu'à maîtriser parfaitement le fonctionnement de votre micro-ordinateur. Alors, vous pourrez participer sans complexes à cette révolution micro-informatique, et nous, nous aurons le sentiment d'avoir rempli notre mission.

Jean-Luc LAGARDÈRE



Avant-propos

C'est à vous, débutant, que s'adressent ce micro-ordinateur ALICE et cet ouvrage, qui vont vous faire découvrir et comprendre la micro-informatique et la programmation. Ils ont tous deux été conçus spécialement pour vous, ni trop simplistes, ni trop difficiles.

ALICE est un micro-ordinateur très facile à utiliser et à maîtriser, conçu pour la découverte, l'exploration de l'informatique. Dans cette exploration, ce manuel sera votre guide :

- il vous apprendra comment installer et utiliser votre micro-ordinateur ALICE ;
- il vous expliquera le langage qu'utilise ALICE, le BASIC ;
- il vous fera acquérir la technique nécessaire pour bien programmer ;
- il vous initiera à la réalisation d'effets graphiques.

A vous ensuite de vous perfectionner en pratiquant.

La *première partie* de cet ouvrage est destinée à vous familiariser avec le matériel. Vous y trouverez des explications sur la mise en route de votre micro-ordinateur ALICE et des exemples pratiques de manipulations. Quand vous vous sentirez à l'aise devant votre clavier et l'écran, désireux d'aller plus loin, il sera temps alors de vous initier à la programmation.

Dans la *deuxième partie*, vous apprendrez, en vous exerçant :

- les opérations les plus courantes dont est capable un ordinateur ;
- comment les lui demander : c'est l'apprentissage du langage ;
- et surtout comment prévoir et combiner ces opérations pour faire rendre à votre ordinateur des services de plus en plus satisfaisants. C'est là tout l'art de la programmation.

La *troisième partie* contient de multiples références, des tables, des codes, un lexique, un index, un cahier de programmes tout prêts, des corrigés d'exercices. En outre, une fiche de référence, séparée, est un aide-mémoire à conserver à portée de la main.

Commencez par étudier les paragraphes écrits en caractères romains – comme ceux que vous lisez en ce moment. Ils contiennent les connaissances élémentaires nécessaires à une progression rapide. Négligez dans un premier temps les textes

écrits en italique (*voici un échantillon d'italique*). En allant ainsi jusqu'à la fin du manuel et en effectuant les exercices proposés, vous aurez acquis "de bonnes bases". Vous pourrez alors reprendre depuis le début en vous attachant surtout aux textes en italique. Les notions qu'ils présentent ne sont pas nécessairement plus ardues que les premières, mais moins immédiatement indispensables.

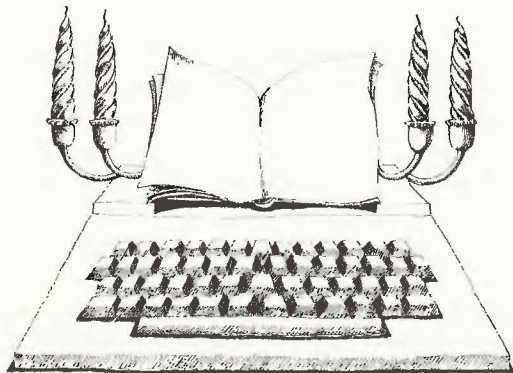
Cette découverte que vous allez faire doit être amusante. Si, en cours de route, vous étiez désespérés, essayez la démarche suivante :

– Peu avant l'endroit qui vous a moins intéressé, un mot ne fut probablement pas très clair pour vous. Un mot que vous "comprenez", mais que vous ne sauriez pas définir ni employer dans une phrase. Détectez-le (ou détectez-les, car un mot malentendu peut en cacher un autre !). Sautez alors sur le dictionnaire, trouvez la définition du mot qui s'accorde au contexte, comprenez-la, employez le mot dans une ou plusieurs phrases de votre cru. Vous pouvez alors relire la partie du texte qui le contenait : elle devrait vous apparaître beaucoup plus claire, et votre fatigue ou votre ennui s'être envolés. Si ce n'était pas le cas, c'est que vous auriez peut-être un autre mot malentendu, ou bien n'auriez-vous pas lu trop longtemps sans manipuler l'ordinateur ?

Des exercices et des manipulations vous sont constamment proposés dans ce manuel qui doit être lu en utilisant toujours le micro-ordinateur en même temps. Une main sur le livre, une main sur le clavier, sinon tout risque de devenir un peu irréel pour vous.

Installez-vous confortablement et découvrez ALICE. Vous verrez que c'est la façon la plus agréable et la plus facile de connaître et de comprendre l'informatique. Et puis, de temps en temps, n'oubliez pas qu'il y a d'autres plaisirs que l'informatique dans la vie et allez vous promener, lire ou retrouver des amis. Vous conserverez ainsi longtemps l'enthousiasme des premiers jours.

Bonne découverte...



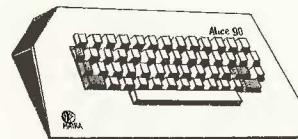
I Alice : Premier contact

Chapitre 1 Mise en route

Prenons notre micro-ordinateur ALICE, mettons le en marche et découvrons son clavier*.

Après avoir soigneusement déballé votre micro-ordinateur, vous avez devant vous :

- le micro-ordinateur ALICE ;
- un câble de raccordement à un téléviseur PÉRITEL ;
- un transformateur avec ses câbles (n'utilisez surtout pas d'autre transformateur) ;
- ce manuel et le GUIDE DE L'ÉDITEUR-ASSEMBLEUR ;
- la fiche de référence ;
- la carte de garantie.



Dessin N° 1

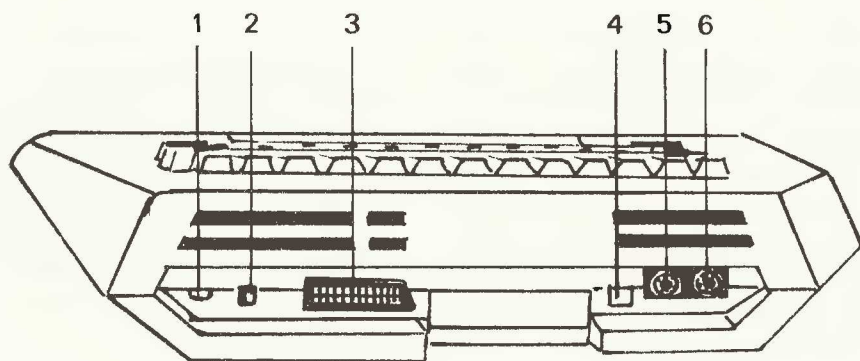


* Mais si vous souhaitez un peu de théorie avant de vous plonger dans la pratique, lisez page 148 « Qu'est-ce qu'un micro-ordinateur ? »

Conservez l'emballage pour toute éventualité et renvoyez la carte de garantie. Prenez votre micro-ordinateur ALICE. Nous nous familiariserons un peu plus tard avec le clavier, mais rien ne vous empêche déjà de pianoter et d'apprécier l'agrément de ses vraies touches.

A l'arrière, vous remarquez plusieurs prises et boutons (les raccordements vous seront expliqués un peu plus loin).

Dessin N° 2



1 BOUTON MARCHE-ARRÊT

2 ALIMENTATION Alimentation électrique : le transformateur inclus dans l'emballage vient se raccorder à cette prise.

3 PRISE TV C'est à cette prise que se raccorde le câble comportant deux fiches PÉRITEL. Suivant le sens du branchement, ce câble vous permet, soit d'utiliser simplement votre téléviseur comme écran de visualisation de votre ordinateur, soit, ce qui est beaucoup plus rare, de réaliser des **incrustations vidéo** (vous devrez toutefois vous procurer un logiciel spécial pour exploiter cette particularité d'Alice).

4 INIT Initialisation : bouton de remise à l'état initial, à utiliser si l'ordinateur se bloque.

5 E/S SÉRIE Entrée et sortie en série : cette prise sert à raccorder ALICE à une imprimante série.

6 CASSETTE Cette prise permet de raccorder ALICE à un magnétophone lecteur-enregistreur de cassettes.

Sous votre micro-ordinateur ALICE, vous remarquerez l'étiquette de fabrication (Fabriqué en France) et l'étiquette de garantie. Attention, **n'ouvrez jamais votre micro-ordinateur ALICE**, la garantie serait annulée.

I BRANCHEMENTS ET MISE EN MARCHÉ

Lisez attentivement cette partie, et plutôt deux fois qu'une. C'est très simple, mais mieux vaut ne pas faire de fausses manœuvres.

Placez votre micro-ordinateur sur une table ou un bureau, ni trop près, ni trop loin du téléviseur PÉRITEL que vous allez utiliser (le câble de raccordement mesure 1,75 m ; avec ALICE vous ne risquez pas d'ophtalmie...).

RACCORDEMENT A UNE PRISE DE COURANT

– Vérifiez que votre micro-ordinateur ALICE est en position **ARRÊT**.

– Prenez le transformateur inclus dans l'emballage. **Surtout n'utilisez jamais d'autre transformateur.**

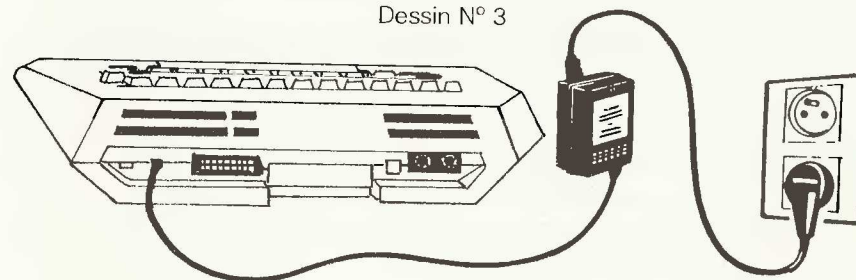
– Trouvez la prise marquée **ALIMENTATION** à l'arrière de votre micro-ordinateur (n° 2 dans le dessin n° 2 page 12) et enfoncez-y la fiche ronde à l'extrémité de l'un des câbles du transformateur.

– Enfoncez la fiche standard à l'extrémité de l'autre câble du transformateur dans une prise de courant murale (ou dans un câble de rallonge électrique ou dans une "nourrice" multiprises).

– Vérifiez que votre transformateur repose par terre ou sur une surface plane, qu'il n'est pas suspendu de tout son poids à l'un des câbles, et que la chaleur qu'il pourrait éventuellement dégager après une longue utilisation ne serait pas gênante.

– Voilà, c'est tout.

Dessin N° 3



QUELQUES RÈGLES ESSENTIELLES

Une fois les raccordements établis :

– Commencez par allumer les périphériques (téléviseur, magnétophone, imprimante, etc.).

– Allumez ensuite le micro-ordinateur.

Quand vous avez fini :

– Éteignez d'abord le micro-ordinateur.

– Éteignez ensuite les périphériques.

N'allumez ni n'éteignez jamais les périphériques quand le micro-ordinateur est en fonctionnement. Sinon, vous risquez de provoquer un mauvais fonctionnement de

l'ordinateur ou un arrêt anormal. Il vous faudra alors appuyer sur le bouton INIT à l'arrière de votre micro-ordinateur (n° 4 dans le dessin n° 2 page 12) ou éteindre puis rallumer le micro-ordinateur. Dans ce cas, toutes les données en mémoire seront effacées.

– Si vous éteignez l'ordinateur, attendez toujours au moins 15 secondes avant de le rallumer.

RACCORDEMENT A UN TÉLÉVISEUR

N'importe quel téléviseur SECAM aux normes PÉRITEL convient*. Pour savoir si votre téléviseur a une prise PÉRITEL, regardez s'il a, à l'arrière, une prise quasi rectangulaire à 20 broches (voir dessin n° 13 page 177).

– Vérifiez que votre micro-ordinateur ALICE et que le téléviseur sont tous deux raccordés à des prises de courant, mais en position ARRÊT.

– Prenez le câble de raccordement.



Dessin N° 4

– Trouvez la prise PÉRITEL à l'arrière de votre téléviseur et enfoncez-y la fiche PÉRITEL du câble de raccordement repérée par une BAGUE ROUGE (ne forcez pas, elle ne peut entrer que dans un sens).

– Repérez la PRISE TV à l'arrière de votre ordinateur, et enfoncez-y l'autre fiche PÉRITEL du câble de raccordement.

– Mettez le téléviseur en marche sur n'importe quelle chaîne.

– Réglez le volume.

– Mettez votre micro-ordinateur ALICE en marche (interrupteur sur le côté droit).

– Votre écran de téléviseur doit devenir vert et montrer le message suivant :

MICROCOLOR BASIC 1.0

COPYRIGHT 1982 MICROSOFT

OK

Si ce message n'apparaît pas (voir page 174) :

– Vérifiez qu'il n'y a pas de panne de courant.

– Vérifiez que le téléviseur est branché.

– Vérifiez que votre micro-ordinateur ALICE est branché.

– Revérifiez tous les raccordements.

– Essayez encore.

Si ça ne marche toujours pas, il peut y avoir diverses raisons :

– Votre téléviseur peut être endommagé.

– Votre téléviseur, bien qu'ayant une prise PÉRITEL, ne respecte peut-être pas les normes PÉRITEL (c'est hélas le cas de certaines marques).

– Le câble de raccordement du micro-ordinateur au téléviseur est peut-être endommagé.

* Si vous avez un téléviseur à prise antenne, vous pouvez acheter un adaptateur antenne Noir et Blanc pour ALICE, mais vous n'aurez pas d'image en couleur.

– Votre micro-ordinateur ALICE est peut-être endommagé.

Si l'image est de mauvaise qualité, trois raisons possibles :

– Votre téléviseur est endommagé ou ne respecte pas les normes PÉRITEL.

– Le câble de raccordement du micro-ordinateur au téléviseur est mal enfoncé ou endommagé.

– Il y a d'importantes fluctuations de courant (voir page 175).

En tout état de cause, sachez que les images produites par votre micro-ordinateur ALICE ne peuvent pas abîmer votre écran de téléviseur ; la qualité des images télévisées sur votre écran ne sera pas affectée par l'utilisation du micro-ordinateur ALICE avec votre téléviseur.

RACCORDEMENT A UN MAGNÉTOPHONE LECTEUR-ENREGISTREUR DE CASSETTES

Vous aurez besoin d'un magnétophone pour sauvegarder des programmes ou des données et pour utiliser des programmes achetés dans le commerce ou prêtés par des amis (voir page 73 les commandes de lecture et de sauvegarde).

Nous vous conseillons d'utiliser le lecteur-enregistreur ALICE de MATRA-HACHETTE, qui a été spécialement conçu pour votre micro-ordinateur, et qui vous garantira une plus grande facilité de fonctionnement et une meilleure fiabilité.

Sur la face arrière de votre ordinateur, repérez la prise marquée CASSETTE (n° 6 dans le dessin n° 3 page 12) : enfoncez-y la fiche DIN à 5 broches (regardez bien et ne forcez pas, elle ne peut entrer que dans un sens). Enfoncez ensuite le jack d'enregistrement dans la prise marquée E (pour Entrées) et le jack de lecture dans celle marquée S (pour Sorties).

A défaut, vous pouvez utiliser en principe n'importe quel magnétophone à cassette, à condition qu'il soit à la fois lecteur et enregistreur. Mais, avec certains appareils, vous aurez peut-être du mal à sauvegarder ou à charger vos programmes.

Vous devrez aussi vous procurer un câble de raccordement.

Celui-ci comportera à une extrémité une prise DIN 5 broches mâles que vous enfoncez dans la prise marquée CASSETTE à l'arrière de votre ordinateur (voir dessin n° 3 page 12), et, à l'autre extrémité, soit une autre fiche DIN 5 broches mâles, soit deux jacks mâles, suivant le type de votre magnétophone.

Si vous utilisez un magnétophone ordinaire, voici quelques conseils qui devraient vous éviter bien des déboires :

– les meilleurs résultats sont obtenus avec des magnétophones monophoniques ;

– un compteur est bien utile pour repérer les programmes enregistrés ;

– un magnétophone à prise jack permet d'enregistrer et de lire sans aucun bruit ;

– certains magnétophones fonctionnent avec le câble DIN/DIN pour l'écriture et le câble DIN/jacks pour la lecture, ce qui n'est guère pratique.

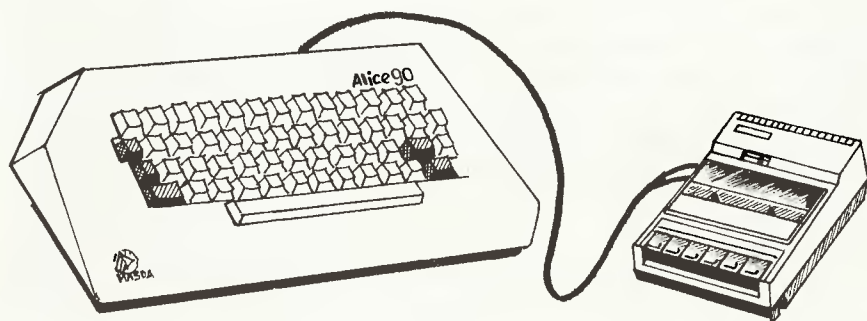
Un conseil : que vous utilisiez le magnétophone ALICE ou un appareil ordinaire, employez des cassettes "superferro" de type C60 ou des cassettes informatiques. N'utilisez pas de cassettes C90 ou CI20, vous éviterez ainsi de longues recherches de programme...

Suivez la procédure de branchement :

– Vérifiez que votre micro-ordinateur ALICE, votre magnétophone et tous les autres périphériques éventuellement raccordés sont branchés sur des prises de courant, mais en position ARRÊT.

– Repérez la prise marquée CASSETTE à l'arrière de votre micro-ordinateur (n° 6 dans le dessin n° 2 page 12) et enfoncez-y la fiche DIN 5 broches à l'extrémité du câble de raccordement (ne forcez pas, elle ne peut entrer que d'une seule manière).

– Connectez l'autre extrémité du câble au magnétophone selon les instructions du manuel d'utilisation de ce dernier. Dans le cas du câble DIN/jacks, le jack d'enregistrement doit être introduit dans la prise micro du magnétophone et le jack de lecture dans la prise écouteur (s'ils ne sont pas déjà différenciés, mettez un élastique au jack d'enregistrement pour ne pas vous tromper).



Dessin N° 5

– Insérez dans le magnétophone une cassette neuve ou préalablement effacée pour enregistrer, ou bien la cassette que vous voulez lire.

– Si une imprimante est raccordée à votre micro-ordinateur, débranchez-la ou mettez-la en position déconnectée (OFF LINE).

– Réglez le volume du magnétophone.

Le réglage du volume (et éventuellement de la tonalité) est identique pour l'enregistrement et la lecture. Le réglage du volume sera correct lorsque le chargement d'un programme (voir page 74) se terminera par le message OK. Si le message "IO ERROR" apparaît, il faut faire varier légèrement le volume et recommencer l'opération. Patience...

Nous vous conseillons de noter sur la page ci-contre les réglages propres à votre magnétophone et de faire une marque sur le bouton de réglage de votre magnétophone.

Modèle	Volume	Tonalité

- Mettez le magnétophone en marche.
- Mettez votre micro-ordinateur ALICE en marche (interrupteur sur le côté droit).
- Suivez les instructions données page 73 (CSAVE, CLOAD).

Notez que votre micro-ordinateur ALICE transfère les données au magnétophone à une vitesse de 190 caractères par seconde (soit environ six lignes d'écran), ou 11 000 caractères par minute (1 500 bauds)*.

Si vous avez des problèmes (voir page 174):

- Vérifiez qu'il y a du courant électrique.
- Vérifiez que le magnétophone est branché.
- Vérifiez que votre micro-ordinateur ALICE est branché.
- Révérifiez tous les raccordements.
- Attention à ne pas écrire sur la bande amorce de la cassette.
- Essayez encore.

Si ça ne marche toujours pas :

- Réglez le volume du magnétophone ; essayez d'autres réglages.
- La cassette a peut-être été abîmée par un passage dans un champ magnétique.

Dans ce cas :

- si vous voulez écrire sur cassette, essayez avec une autre cassette ;
- si vous voulez lire une cassette, essayez avec la copie de sauvegarde, si vous en avez une.

RACCORDEMENT A UNE IMPRIMANTE

Vous aurez besoin d'une imprimante pour garder une copie écrite (et parfaitement exacte...) de vos programmes. Nous vous conseillons de vous procurer l'imprimante thermique que MATRA-HACHETTE a spécialement mise au point pour ALICE : vous aurez ainsi un appareil parfaitement adapté aux caractéristiques techniques et à l'esthétique de votre ordinateur. Cette imprimante vous permettra de copier vos programmes à la vitesse de 30 caractères par seconde. Vous pourrez même reproduire les caractères semi-graphiques.

Toutefois, vous pouvez aussi utiliser un autre modèle, à condition qu'il s'agisse bien d'une imprimante série conforme aux spécifications logicielles de l'entrée-sortie d'ALICE (voir page 177). Consultez toujours très attentivement la notice d'utilisation et n'hésitez pas à vous renseigner auprès de votre revendeur ou du fabricant.

* Un baud est un bit par seconde (voir page 149).

Pour raccorder votre micro-ordinateur et l'imprimante :

– Vérifiez que votre micro-ordinateur ALICE, l'imprimante (que ce soit le modèle ALICE ou un autre) et tous les périphériques éventuellement raccordés sont branchés sur des prises de courant, mais en position ARRÊT.

– Repérez la prise marquée E/S SÉRIE à l'arrière de votre micro-ordinateur (n° 5 dans le dessin n° 3 page 12) et enfoncez-y la fiche DIN 4 broches à l'extrémité du câble de raccordement (ne forcez pas, elle ne peut entrer que d'une seule manière).

– Connectez l'autre extrémité du câble à l'imprimante, selon les instructions du manuel de l'imprimante.

– Mettez votre micro-ordinateur ALICE en marche (interrupteur sur le côté droit).

– Suivez les instructions données page 38 (LLIST) et page 60 (LPRINT).

Notez que la vitesse de transfert d'ALICE vers l'imprimante est de 75 caractères par seconde (un peu plus de deux lignes d'écran), ou 4500 caractères par minute (600 bauds).



Attention : si vous avez raccordé à la fois un magnétophone et une imprimante, il faut débrancher l'imprimante, ou la mettre en position déconnectée (OFF LINE) avant de sauvegarder des données sur cassette.

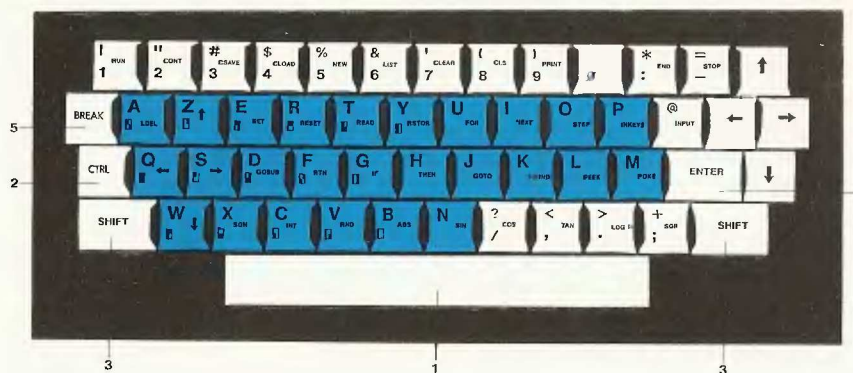
ALICE dispose de toutes les possibilités de communication de la norme RS232C. Le connecteur E/S SÉRIE peut être utilisé non seulement pour une imprimante, mais aussi pour un modem (afin de transmettre des informations par téléphone) ou pour d'autres périphériques compatibles avec cette norme (synthétiseur de voix, boîte à musique...).

Maintenant que votre micro-ordinateur ALICE est branché, raccordé à un téléviseur et éventuellement à un magnétophone et à une imprimante, partons à la découverte. D'abord le clavier...

II SE FAMILIARISER AVEC LE CLAVIER

Le clavier d'ALICE, reproduit ci-dessous, a l'avantage de disposer de touches à fonctions multiples, en particulier de touches correspondant aux mots-clés du BASIC. Elles seront décrites en détail plus tard, mais nous pouvons commencer à nous familiariser avec ce clavier.

Dessin N° 6



Regardez le dessin n° 6 (page précédente). Dans la rangée supérieure, remarquez que le zéro est noté "0" de façon à éviter la confusion avec le "O" ("o" majuscule) : cette distinction est indispensable pour l'ordinateur, mais les débutants, et les autres... font souvent la confusion. Les chiffres et les lettres de l'alphabet (touches bleues) sont directement accessibles : tapez par exemple les six touches **F R A I S E**. Vous lisez ceci à l'écran :

OK
FRAISE ■ et le curseur s'est déplacé de 6 positions.

Pour ajouter à la suite : "DES BOIS", il vous faut utiliser la *barre d'espace* qui "imprime" un espace. Faites-le une fois, appuyez sur la barre d'espace (Ⓢ sur le dessin n° 6), vous voyez le curseur se déplacer d'une case vers la droite. Tapez maintenant la suite prévue. Vous voyez à l'écran :

FRAISE DES BOIS ■

Si vous voyez autre chose, c'est que vous avez appuyé sur d'autres touches. Aucune importance, c'est plus facile à corriger que sur la plupart des machines à écrire.

La touche ←

Appuyez sur cette touche, placée à droite du clavier : le curseur recule d'une case et efface le caractère qui l'occupait. Pour corriger, il suffit de faire reculer le curseur de façon à ne garder que la partie à conserver, et de taper le reste de la ligne. (Nous verrons plus loin l'utilisation des autres flèches.)

La touche **CTRL** (ou **CONTROL**)

Cette touche (n° 2 sur le dessin n° 6) ne fonctionne qu'associée à une autre. Si vous appuyez dessus, il ne se passe rien.

Maintenez-la enfoncée, et appuyez sur la touche **Q** : le curseur recule et efface le caractère qui était placé à sa gauche ! En effet, ← et **CTRL** **Q** produisent le même effet.

Maintenant, appuyez simultanément sur **CTRL** et sur **A**. Regardez l'écran : plus rien ! Votre ligne a disparu. Remarquez sur la touche **A**, à mi-hauteur et à droite, l'inscription "LDEL", abréviation pour "Line Delete" ou "effacement de ligne" : la touche **CTRL** permet d'accéder aux mots et symboles figurant, en petits caractères, sur la partie droite des touches.

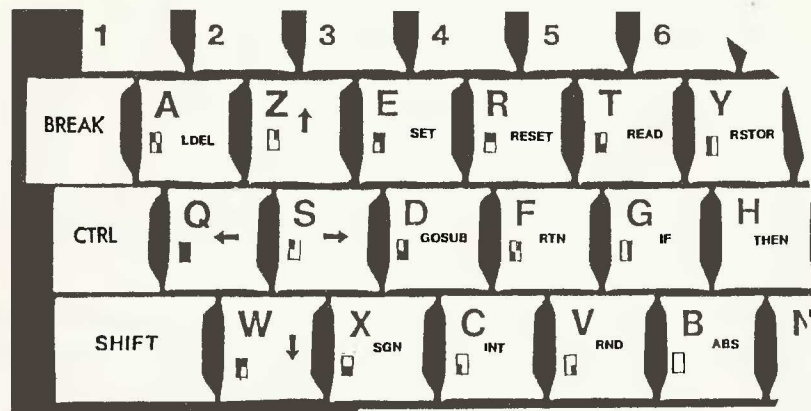
Appuyez par exemple sur **CTRL** et sur **J**. Vous obtenez à l'écran le mot "GOTO", qui est un terme utilisé en BASIC. Pour écrire vos programmes, au lieu de taper chaque mot lettre à lettre, vous n'aurez qu'à appuyer simultanément sur la touche **CTRL** (que nous appellerons **CONTROL**) et sur la touche portant le mot BASIC désiré.

Les touches **SHIFT**

Ces touches (3 sur le dessin n° 6) sont l'équivalent des "MAJUSCULES" des machines à écrire. Elles permettent d'obtenir le caractère supérieur des touches qui ne correspondent pas à une lettre de l'alphabet. Ainsi, **SHIFT** **I** affiche "!", **SHIFT** **/** affiche "?", etc.

Les touches **SHIFT** permettent aussi d'accéder aux caractères graphiques dessinés sur certaines touches (voir dessin n° 7). Faites **SHIFT** **F** ; remarquez que le caractère graphique qui apparaît à l'écran est inversé par rapport au dessin figurant sur la touche.

Dessin N° 7



SHIFT **0** appuyés une seule fois, puis relâchés, font passer le clavier en mode inversé :

- en inversion vidéo (à l'écran) ;
- en minuscules à l'imprimante si vous disposez d'une imprimante qui affiche les minuscules.

Tant que vous êtes sous ce mode, **SHIFT** en même temps qu'un caractère le fera échapper à l'inversion vidéo et l'affichera normalement en majuscules à l'imprimante. Nous verrons plus loin comment obtenir les minuscules à l'écran. Mais attention, toutes les commandes BASIC doivent être en majuscules.

On sort du mode inversé en appuyant à nouveau sur **SHIFT** **0**.

La touche **ENTER**

La touche **ENTER** (Ⓢ sur le dessin n° 6) s'appelle aussi *touche de validation*. Ce que vous avez tapé jusqu'à présent s'est inscrit à l'écran, mais n'a pas été pris en compte par l'ordinateur. C'est pourquoi, d'ailleurs, vous pouviez corriger une éventuelle faute de frappe aussi facilement.

La touche de validation **ENTER** signale à l'ordinateur d'enregistrer votre message.

Faites un essai ; tapez à nouveau le mot FRAISE.

L'écran se présente ainsi* :

OK
FRAISE
? SN ERROR
OK

Appuyez sur **ENTER**
Résultat. L'ordinateur a réagi, il a "entendu" votre message mais ne le comprend pas, et vous le fait savoir (SN ERROR signifie "erreur de syntaxe").
Mais il ne vous en veut pas et attend la suite !

Donnez-lui maintenant un message qu'il puisse comprendre. Appuyez simultanément sur **CONTROL** et **8** qui produisent à l'écran le mot CLS, puis **ENTER** ... Écran effacé !

Là encore, l'ordinateur a réagi. Il a entendu votre message et, cette fois, l'a compris, car CLS signifie "effacer l'écran" dans son langage (Clear Screen, en anglais).

La touche **BREAK**

La touche **BREAK** (Ⓢ sur le dessin n° 6) est une *touche d'interruption* qui ne nous est d'aucune utilité pour l'instant. Elle sert à interrompre une opération en cours, notamment l'exécution d'un programme.

Un dernier conseil : lorsque vous utilisez SHIFT ou CONTROL, appuyez d'abord sur cette touche et, tout en la maintenant enfoncée, appuyez sur l'autre ; vous éviterez ainsi bien des fautes de frappe.

Et maintenant, faites quelques essais. Tapez **CONTROL** **8** **ENTER** pour effacer l'écran, puis exercez-vous à taper :

"MESSAGE # 1 : ALICE, C'EST UN MICRO SUPER (& PAS CHER) !"

Effacez de nouveau l'écran, puis tapez :
0 = 1 + 2 - 3 < 4 * 5 > 6 / 7 < 99 %

Enfin, en effaçant à chaque fois, tapez l'alphabet dans l'ordre, puis toutes les commandes BASIC (avec la touche **CONTROL**), puis tous les caractères graphiques (avec la touche **SHIFT**).

Si vous faites des erreurs, utilisez **CONTROL** **Q** ou **CONTROL** **A**.

Refaites ensuite la même chose en mode inversé (avec **SHIFT** **0**). Corrigez-vous les erreurs de la même façon ? Peut-on faire du graphisme ? Continuez jusqu'à être tout à fait à l'aise avec le clavier d'ALICE.

* Désormais ce que vous tapez au clavier sera en noir et ce que l'ordinateur vous répond sera en bleu. Nos explications en seront plus claires.

Chapitre 2 Premiers essais

Voilà, votre micro-ordinateur ALICE est branché, vous savez vous servir du clavier, alors commençons maintenant nos premières manipulations : donner des ordres simples et les faire exécuter.

I COMMANDER, ÊTRE INSTANTANÉMENT OBÉI

Au lieu d'obtenir des affichages à l'écran par un simple effet d'écho, nous allons maintenant communiquer des ordres et voir l'ordinateur les recevoir et les accomplir immédiatement.

Nettoyons d'abord l'écran par **CONTROL** **8** **ENTER** ou, si vous préférez, les touches **C** **L** **S** et **ENTER**. On voit apparaître en haut de l'écran vide le message OK.

Maintenant tapez le mot PRINT, soit en tapant les 5 touches du mot au clavier, soit en appuyant en même temps sur **CONTROL** et **9**. Vous voyez à l'écran :

PRINT ■

PRINT est le mot que l'ordinateur reconnaît comme un ordre d'afficher à l'écran ce qui va lui être indiqué. Indiquons-lui ce qu'il doit afficher ; tapez "BIENVENUE A BORD", puis **ENTER**.

Voici l'aspect de l'écran :

OK
PRINT "BIENVENUE A BORD"
BIENVENUE A BORD
OK

Écho des touches frappées.
Affichage en réponse à l'ordre reçu.



L'ordre que nous avons donné comporte 2 parties distinctes :

PRINT	"BIENVENUE A BORD"
Soit : Afficher	Texte à afficher
Soit : <i>Instruction</i>	<i>Donnée</i>

Lorsque vous avez pressé la touche **ENTER**, c'est-à-dire **validé** votre message, l'ordinateur a analysé le mot PRINT. Il était écrit correctement, c'est bien une instruction du langage BASIC, il a donc été compris. Les guillemets sont interprétés comme l'emballage d'un paquet tout prêt, bien ficelé, que l'ordinateur n'analyse même pas ; de toutes façons cela ne le concerne pas.

Vérifions cela. Tapez :

PRIT (en oubliant le N) "BIENVENUE A BORD" **ENTER**

Vous voyez apparaître :

```
?SN ERROR
OK
```

SN ERROR signifie "erreur de syntaxe", donc instruction incompréhensible, non exécutable. Tapez maintenant :

PRINT "MERSI BOCOU" **ENTER**

A l'écran vous voyez :

```
MERSI BOCOU
```

Ne comptez pas sur l'ordinateur pour corriger spontanément ou critiquer vos textes !

Et si le message dépasse la longueur d'une ligne d'écran ?

Essayons. Tapez : **CONTROL** **9** "LE COMMANDANT ET SON EQUIPAGE VOUS SOUHAITENT LA BIENVENUE A BORD D'ALICE" **ENTER**

Cela se présente à l'écran de la manière suivante :

```
PRINT "LE COMMANDANT ET SON EQUI
```

Le curseur clignote sur la dernière case, la 32^e.

Continuez tranquillement, il passe tout seul à la ligne suivante.

```
PAGE VOUS SOUHAITENT LA BIENVENU
E A BORD D'ALICE"
```

A l'exécution, les coupures sont décalées de 7 caractères ; les 5 caractères de PRINT, le blanc et les guillemets.

Arrivés à ce point, vous avez peut-être déjà remarqué quelque chose :

1. Nous avons perdu la première ligne frappée et le curseur est sur la dernière ligne de l'écran. Le passage du curseur à la ligne après le dernier message "OK" a fait remonter vers le haut toutes les lignes imprimées, et la première a disparu. Si vous tapez une nouvelle ligne et la validez, la ligne actuellement en haut de l'écran disparaîtra à son tour. Tout se passe comme si nous faisons dérouler un parchemin que nous lisons à travers une fenêtre de 16 lignes de 32 caractères.

Si vous voulez à nouveau travailler sur un écran propre, vous savez comment faire, n'est-ce pas ?

Allons-y... CLS **ENTER**

2. Les instructions restent affichées et font double emploi avec le résultat de leur exécution ; les affichages sur plusieurs lignes sont bien peu esthétiques. Si nous laissons quelques instructions pour demander à l'ordinateur de faire le ménage ?

II LAISSER DES INSTRUCTIONS

Comment devrait se présenter l'affichage du dernier texte ? Définissons-le :

ligne	1	
	2	LE COMMANDANT
	3	ET SON EQUIPAGE
	4	VOUS SOUHAITENT LA BIENVENUE
	5	A BORD D'ALICE
	6	
	.	
	.	
	.	
	16	

Que faut-il faire pour parvenir à cet affichage ? Il faut :

1. effacer l'écran ;
 2. sauter une ligne ;
 3. afficher sur la ligne suivante "LE COMMANDANT" ;
 4. afficher sur la ligne suivante "ET SON EQUIPAGE" ;
- et ainsi de suite.

Ce sont point par point les instructions que vous allez laisser, en prenant soin de les numéroter afin que par la suite elles puissent être exécutées chacune à leur tour. Faites-le, tapez vos instructions ; en voici la traduction en BASIC :

1 CLS **ENTER**

Rien ne se passe, l'écran n'est pas effacé. Normal : le chiffre en début de ligne indique à l'ordinateur que l'instruction qui suit ne doit pas être exécutée immédiatement, mais mise en mémoire. Plus de message OK, seul le curseur clignote.

2 PRINT **ENTER**

L'instruction PRINT non suivie d'une donnée provoque le passage à la ligne.

3 PRINT "LE COMMANDANT" **ENTER**

4 PRINT "ET SON EQUIPAGE" **ENTER**

5 PRINT "VOUS SOUHAITENT LA BIENVENUE" **ENTER**

6 PRINT "A BORD D'ALICE" **ENTER**

En cas de malheur

La ligne 5 vous pose-t-elle quelques problèmes ? Elle dépasse la longueur de la ligne-écran, c'est vrai, mais pourquoi vous en soucier ? Laissez le curseur se déplacer tout seul à la ligne suivante ! Ne tapez surtout pas sur **ENTER** dans le seul but de faire aller le curseur à la ligne. Il irait, bien sûr, mais la touche **ENTER** a pour première fonction la validation : vous obtiendriez la mise en mémoire de votre ligne d'écran, soit :

```
5 PRINT "VOUS SOUHAITENT LA BIE ENTER
```

Si vous continuez vous taperez :

```
NVENUE ENTER
```

Vous obtenez :

```
?SN ERROR  
OK
```

Si vous avez été gratifié d'un message d'erreur pour une raison quelconque, ne tentez pas de l'effacer ni de corriger la ligne précédente à l'écran. Tapez simplement à la suite votre ligne correcte et complète.

Voici un autre exemple de ce qui pourrait se passer. Vous avez tapé :

```
CONTROL 2 PRINT ENTER
```

par erreur au lieu de 2 PRINT, etc.

Vous voyez à l'écran :

```
CONT PRINT  
?SN ERROR  
OK
```

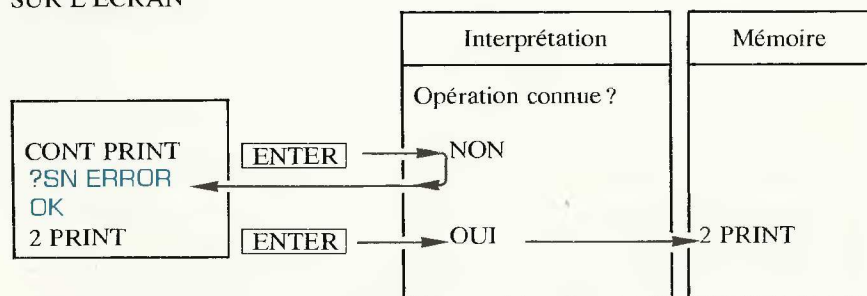
Tapez à la suite :
2 PRINT

Ce qui se trouve à l'écran depuis CONT PRINT jusqu'à OK n'est pas effacé, mais ne compte pas. La demande de validation **ENTER** que vous avez faite après CONT PRINT a été rejetée, la ligne fautive n'a pas été mise en mémoire.

Résumons-nous :

CE QUI EST VISIBLE
SUR L'ÉCRAN

CE QUI EST EN MÉMOIRE



III EXÉCUTION !

Tout est prêt ? Vous avez bien les 6 instructions prévues ?

Alors, pas d'hésitation, faites-les exécuter.

Taper **R U N ENTER** (ou **CONTROL 1** puis **ENTER**, ce qui revient au même).

Voici ce qui apparaît à l'écran :

```
LE COMMANDANT  
ET SON EQUIPAGE  
VOUS SOUHAITENT LA BIENVENUE  
A BORD D'ALICE  
OK
```

Exactement ce que nous voulions... Sauf le message OK que nous n'avions pas demandé. Il est là tout de même, car c'est le seul moyen dont dispose l'ordinateur pour nous dire : "Mission accomplie, quelle est la prochaine ?".

Avec quelle rapidité il s'est acquitté de sa tâche !

Voyez : effacez l'écran par une instruction à effet immédiat, c'est-à-dire sans numéro de ligne.

```
CLS ENTER
```

Puis, à nouveau, tapez :

```
RUN ENTER
```

Vous obtenez rigoureusement le même résultat que précédemment.

Recommenceriez-vous mille fois, vous obtiendriez mille fois la même chose, tant que vous ne changez pas les instructions en mémoire (et tant que vous ne coupez pas le courant, puisqu'elles sont stockées en mémoire vive).

Contrairement à une reproduction sur bande vidéo, il n'y a aucune usure de l'image au-delà d'un grand nombre d'exemplaires, car l'image est créée à chaque exécution.

IV SUR TROIS MODES : LES RELATIONS ENTRE VOTRE ORDINATEUR ET VOUS

Vous possédez maintenant assez d'éléments pour comprendre les trois manières de profiter de votre ordinateur :

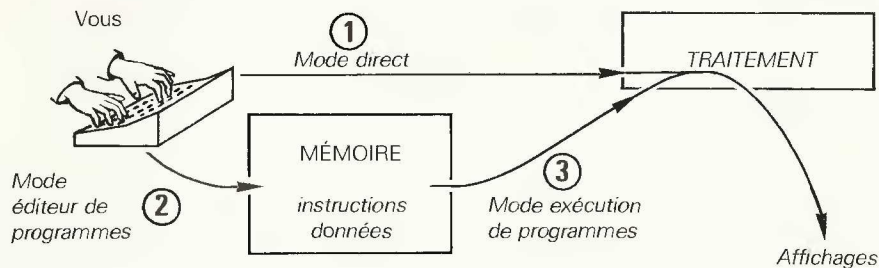
1. Les commandes comme CLS, RUN, immédiatement validées et exécutées, sont une utilisation en mode direct, aussi direct que lorsque vous appuyez sur le déclencheur d'un appareil photographique.

2. Les instructions numérotées que vous entrez en mémoire constituent un programme. (Voilà qui est dit ! Vous vous en doutez ? très bien !). Vous travaillez alors en mode éditeur de programmes.

En travaillant sous ces deux modes, vous "avez la main", c'est vous le maître à bord. Vous passez d'un mode à l'autre à votre gré.

3. Le mode exécution de programme (tout de suite après que vous avez tapé RUN **ENTER**) délègue entièrement votre autorité à l'ordinateur qui déroule alors les instructions du programme. Vous ne reprenez la main que lorsque l'exécution est terminée ou interrompue.

Résumons-nous



Votre ordinateur ne peut faire qu'une chose à la fois :

- ou il vous écoute, mémorise et se croise les bras (2) ;
- ou il travaille, mais n'entend pas les ordres que vous pourriez vouloir lui donner (3).

Quand vous avez l'illusion qu'il fait les deux, en mode direct (1) c'est qu'il enchaîne un peu de 2 et un peu de 3 en un temps très court, de l'ordre de quelques micro-secondes (une micro-seconde vaut un millionième de seconde).

V ÉTONNEZ DÉJÀ VOS AMIS !

Enfin... pas n'importe lesquels. Choisissez tout de même un "pied-tendre", ce sera plus sûr !

Vous allez entrer les instructions suivantes :

1 CLS	ENTER
2 PRINT	ENTER
3 PRINT "BONJOUR ANTONIN !"	ENTER (si l'ami s'appelle Antonin, bien sûr, sinon adaptez !)
4 PRINT "ALICE ET MOI"	ENTER
5 PRINT "TE SALUONS"	ENTER
6 PRINT " SIGNE : "	ENTER
↑ ↑	
12 espaces Ici, votre prénom.	

Testez tout de même votre petit programme pour voir si tout va bien en le faisant exécuter une fois (RUN **ENTER**). Au cas où une erreur se serait glissée, tapez à nouveau la ligne (inutile de taper à nouveau les autres.) Testez et corrigez si nécessaire jusqu'à ce que tout aille bien.

Maintenant, effacez l'écran par CLS **ENTER**.

Allez chercher votre ami et installez-le devant l'écran.

Faites exécuter le programme (commande RUN **ENTER**). Vous le savez, mais c'est un rappel au cas où l'émotion vous ferait perdre vos moyens !).

Ça y est ? L'effet produit a été favorable ?

Voulez-vous faire encore mieux et varier vos effets ?

Si oui, alors la deuxième partie du manuel s'ouvre à vous !





II Alice : Parlez Basic

Présentation

Chacun des chapitres suivants vous propose une activité progressive :

- d'abord apprendre en pratiquant, et si possible en s'amusant. Vous apprenez en même temps que vous manipulez le clavier.
- ensuite, mais pas systématiquement, un exercice expliqué constitue une brasse d'essai (avec bouée!).

- dernière phase : on enlève la bouée. Des exercices vous invitent à appliquer les acquisitions du ou des chapitres précédents. Il vous faudra parfois de l'astuce. Mais vous avez pied tout de même : les corrigés se trouvent à la fin du manuel, il suffit d'y jeter un coup d'œil. Nous ne vous promettons pas toujours le corrigé complet, mais au moins les "tuyaux" qui permettent de résoudre les difficultés.

Et permettez-nous de vous rappeler ceci :

- ne laissez pas passer un mot mal compris. Ayez souvent recours au dictionnaire français. Vous n'aurez probablement pas besoin d'un dictionnaire d'informatique pour l'instant, car nous avons eu le souci de définir les mots techniques utilisés, et vous les retrouverez d'autant plus facilement dans leur contexte qu'ils sont soulignés.

- la présentation typographique permet le choix entre 2 méthodes de progression : la meilleure pour un débutant total nous semble de prêter attention uniquement au texte en caractères romains et, arrivé à la fin de cette section, de reprendre depuis le début du manuel en lisant les caractères en italique. Toutefois, si vous possédez déjà quelques notions, vous trouverez des compléments utiles dans les deux présentations du texte, à lire sans les séparer.

La progression va être la suivante :

- d'abord quelques éléments de base : qu'est-ce qu'un programme, définition des variables, présentation d'un peu de traitement, sauvegarde des programmes ;
- puis des notions plus complexes : répétitions et itérations, branchements conditionnels, données en file d'attente, utilisation de la couleur ;
- enfin, la maîtrise du BASIC : sous-programmes et aiguillages, données en libre service. pour aller plus loin...

Chapitre 1

Votre premier programme

I PREMIERS PAS...

Dans la section précédente, nous avons commencé à programmer presque sans le savoir. Jetons un nouveau coup d'œil sur la partie intitulée : "laisser des instructions", page 25.

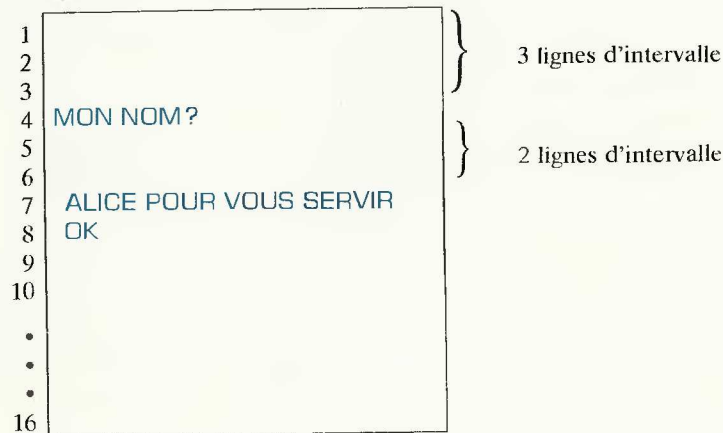
La démarche suivie consistait déjà à :

- définir le résultat souhaité ;
- analyser • les moyens d'y parvenir ;
 - dans quel ordre les combiner ;
- traduire en BASIC pour ALICE. Quand on en arrive à cette étape, le plus gros du programme est déjà fait.

A PROGRAMMER, C'EST PRÉVOIR

Prévoyons quelque chose et construisons un programme dans les règles de l'art.

1 Définition du résultat. Nous allons faire afficher à l'écran :



2 Comment y parvenir, par quelles étapes ?

1. Effacer l'écran (nous avons déjà vu comment) ;
2. aller 3 fois à la ligne sans rien afficher ;
3. afficher "MON NOM?" ;
4. aller 2 fois à la ligne sans rien afficher ;
5. afficher "ALICE POUR VOUS SERVIR".

3 Traduction en BASIC

Vous connaissez les mots, vous pouvez taper :

1 CLS	<input type="text" value="ENTER"/>	. Efface l'écran.
2 PRINT	<input type="text" value="ENTER"/>	
3 PRINT	<input type="text" value="ENTER"/>	
4 PRINT	<input type="text" value="ENTER"/>	
5 PRINT "MON NOM?"	<input type="text" value="ENTER"/>	
6 PRINT	<input type="text" value="ENTER"/>	
7 PRINT	<input type="text" value="ENTER"/>	
8 PRINT "ALICE POUR VOUS SERVIR"	<input type="text" value="ENTER"/>	

Exécution, , tout va bien.

Ce premier travail méthodique accompli, nous allons maintenant triturer notre programme, le "bidouiller" (bidouiller : terme de mépris désignant l'acte de programmer sans méthode, obligeant à retoucher maintes fois un programme).

B "BIDOUILLER", C'EST EXPÉRIMENTER

Jouons avec les numéros de lignes.

Tapez :

3	<input type="text" value="ENTER"/>
4	<input type="text" value="ENTER"/>
6	<input type="text" value="ENTER"/>
7	<input type="text" value="ENTER"/>
RUN <input type="text" value="ENTER"/>	

Vous obtenez à l'écran :



Que s'est-il passé ?

En tapant un numéro de ligne immédiatement suivi de , on efface la ligne qui porte ce numéro.

Il n'est pas nécessaire que les numéros de lignes se suivent ; il est même recommandé de numéroter les lignes de 10 en 10, de façon à pouvoir intercaler de nouvelles lignes en cas de modification.

La 1^{re} ligne du programme peut avoir n'importe quel numéro. On peut commencer un programme en ligne 100, par exemple.

Les lignes peuvent être numérotées de 0 à 63 999.

II CONSULTEZ LE PROGRAMME!

Depuis un moment, nous ne voyons plus notre programme. Nous voyons ce qu'il fait, mais les instructions ont disparu, – pas perdues pour tout le monde, et surtout pas pour l'ordinateur, – mais ensevelies dans les profondeurs de sa mémoire. Nous pouvons très bien lui donner l'ordre (commande – en mode direct) de nous le présenter.

Tapez la commande LIST en tapant les 4 touches du mot LIST ou .

La liste des lignes (ou "listing") apparaît :

```
1 CLS
2 PRINT
5 PRINT "MON NOM?"
8 PRINT "ALICE POUR VOUS SERVIR"
OK
```

Intéressant, non ?

Nous savions déjà que nous pouvions *exécuter* un programme ; on peut aussi le "lister", cela donne envie de le modifier.

Si vous tapez :

```
9 PRINT "BOF..."
3 PRINT "C'EST MOI!"
```

à la suite, à l'écran tout cela paraît bien désordonné... Et que se passe-t-il dans la mémoire ?

Effaçons d'abord l'écran : CLS

Puis LIST

Et voici l'écran :

```
OK
LIST
1 CLS
2 PRINT
3 PRINT "C'EST MOI!"
5 PRINT "MON NOM?"
8 PRINT "ALICE POUR VOUS SERVIR"
9 PRINT "BOF..."
```

Tout y est, toutes les lignes reclassées par numéro. S'exécuteront-elles ainsi ?

Faites RUN et voyez...

```
C'EST MOI!
MON NOM?
ALICE POUR VOUS SERVIR
BOF...
OK
```

UN PEU DE CORRECTION

Bien désinvolte, ce "C'EST MOI!". "ME VOICI" serait plus correct. Tapez :

```
3 PRINT "ME VOICI"
RUN 
```

On voit alors

```
ME VOICI
MON NOM?
ALICE POUR VOUS SERVIR
BOF...
OK
```

Pourquoi "C'EST MOI!" a-t-il disparu ? La nouvelle ligne 3 a remplacé l'ancienne sans qu'il ait été nécessaire de l'effacer. On dit que la nouvelle ligne "écrase" l'ancienne.

Par contraste avec "ME VOICI", très stylé, "BOF" est parfaitement révoltant ! Supprimons-le en tapant par exemple :

```
9 PRINT "MONSIEUR" 
```

Ou MADAME ou JEUNE MAÎTRE, comme dans les contes ! Vérifiez que la ligne a bien été "écrasée" en faisant :

```
RUN 
```

Mais il existe un procédé beaucoup plus souple et plus rapide pour corriger une ligne. Faites :

```
  5 
```

La ligne 5 s'affiche à l'écran.

```
 PRINT "MON NOM?"
```

Le curseur s'est placé sur le premier caractère de la ligne, le 5, et le curseur et le chiffre 5 clignotent en alternance : ceci vous indique que vous êtes en **mode ÉDITION**, c'est-à-dire que certaines touches jouent un rôle particulier pour vous permettre de corriger plus facilement votre programme.

Appuyez sur la touche située sur la partie droite de votre clavier : le curseur se déplace d'une case vers la droite. Répétez l'opération. Chaque fois que vous appuyez sur cette touche , le curseur se décale d'une case vers la droite, sans modifier le texte affiché sur l'écran. Faites la même chose avec la touche : le curseur se déplace vers la gauche, sans effacer les caractères qu'il rencontre.

Vous pouvez donc facilement ajouter ce que vous voulez, où vous le voulez. Vous souhaitez remplacer NOM par PRÉNOM ? Amenez le curseur sur le N de NOM, et tapez P :

5 PRINT "MON PNOM?"

La lettre P est venue s'insérer dans la ligne 5, à l'endroit où se trouvait le curseur, et toutes les lettres suivantes se sont automatiquement décalées vers la droite. Continuez et tapez, à la suite REN. Voici ce que devez obtenir sur votre écran :

5 PRINT "MON PRENNOM?"

Eh oui, emportés par notre élan, nous avons mis un N de trop ! Aucune importance, supprimons-le ! Assurez-vous que le curseur est bien sur le deuxième N et appuyez sur la touche \downarrow : le N disparaît et tous les autres caractères se décalent d'une case vers la gauche. Répétez l'opération : le O, puis le M disparaissent à leur tour et, à chaque fois, le reste de la ligne se décale pour combler le trou... Pour avoir à nouveau le mot PRENOM en entier, il suffit de relâcher la touche \downarrow et de taper à nouveau les lettres manquantes.

*Remarque : Vous pouvez obtenir le même résultat avec la touche **CONTROL** :*
 $\text{CTRL} \text{ O}$ et $\text{CTRL} \text{ S}$ déplacent le curseur dans le sens indiqué par les flèches,
 $\text{CTRL} \text{ W}$ supprime le caractère où se trouve le curseur.

Ça y est, vous êtes satisfait de votre texte ? Pour le faire savoir à ALICE, il suffit de valider la ligne en faisant ENTER . Cette fois, le curseur abandonne la ligne 5 : vous êtes revenu au mode normal.

Mais on peut aller encore plus vite pour modifier une ligne de programme. Remplaçons la ligne 5 par QUI ÊTES-VOUS ?

Faites $\text{SHIFT} * \text{ENTER}$

Puis faites $\text{SHIFT} \text{ ESPACE}$ (Nous appellerons ainsi la barre d'espacement : I sur le dessin n° 6 page 19).

Dorénavant, toutes les touches seront répétées automatiquement si vous les maintenez enfoncées. Déplacez le curseur avec les touches \leftarrow et \rightarrow . Amenez-le sur le premier M et appuyez sur \downarrow : regardez avec quelle rapidité les lettres disparaissent ! Pour abandonner la répétition automatique, faites $\text{CONTROL} \text{ ESPACE}$, et tapez la nouvelle version.

Dernier avantage du mode édition, la facilité avec laquelle on peut recopier une ligne. Si nous voulons qu'ALICE demande à son tour à son interlocuteur qui il est, il suffit de faire :

$\text{SHIFT} * 5 \text{ ENTER}$

$5 \text{ PRINT "QUI ETES-VOUS?"}$

Appuyons sur la touche \downarrow pour supprimer le 5 et tapons à la place le nouveau numéro, par exemple : 12. Appuyons sur ENTER : cela suffit pour valider notre nouvelle ligne.

Faisons LIST pour vérifier : nous avons bien deux lignes identiques, la 5 et la 12.

```
1 CLS
2 PRINT
3 PRINT "ME VOICI"
5 PRINT "QUI ETES-VOUS?"
8 PRINT "ALICE POUR VOUS SERVIR"
9 PRINT "MONSIEUR"
12 PRINT "QUI ETES-VOUS?"
```

RÉSUMONS

Pour **supprimer une ligne** : taper seulement son numéro et ENTER .
 Pour **modifier une ligne** :
 soit **retaper** le même numéro, la nouvelle version et ENTER ,
 soit faire $\text{SHIFT} * \text{N}^\circ \text{ LIGNE} \text{ ENTER}$, ce qui vous met en **mode ÉDITION**. Vous disposez alors des commandes suivantes :

\leftarrow et \rightarrow déplacent le curseur ;

\downarrow supprime le caractère où se trouve le curseur ;

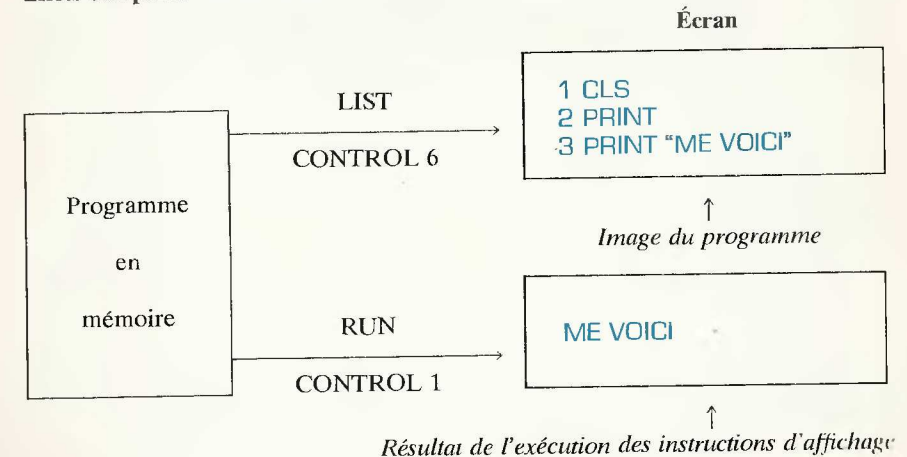
L'appui d'une touche seule insère le caractère correspondant à l'endroit où clignote le curseur ;

$\text{SHIFT} \text{ ESPACE}$ provoque la répétition automatique de toutes les touches utilisées par la suite ;

$\text{CONTROL} \text{ ESPACE}$ met fin à la répétition ;

ENTER valide la ligne et fait repasser en mode normal.

Effets comparés des commandes LIST et RUN sur un même programme



Résultat de l'exécution des instructions d'affichage

LIST 1 - 5 [ENTER] permet de ne lister qu'une partie du programme, tout ce qui est entre la ligne 1 et la ligne 5 comprises.

LIST 3 [ENTER] ne liste que la ligne 3. Si elle n'existe pas, le message OK apparaît seul.

LIST 3 - [ENTER] liste la ligne 3 et les suivantes jusqu'à la fin.

LIST - 7 [ENTER] liste de la 1^{re} ligne à la ligne 7.

LIST n° de ligne de départ - n° de ligne finale.

N° de ligne de départ : la 1^{re} ligne que vous voulez lister. Si vous l'omettez, le listing à l'écran commencera à la 1^{re} ligne du programme.

N° de ligne finale : la dernière ligne que vous voulez lister. Si vous l'omettez, le listing à l'écran se terminera sur la dernière ligne du programme.

LIST N° de ligne
liste une seule ligne à l'écran.

LIST
liste tout le programme à l'écran.

Si vous disposez d'une imprimante connectée à votre micro-ordinateur, et en état de marche (voir page 17), la commande LLIST [ENTER] listera sur papier le programme alors en mémoire (on peut aussi taper L, puis [CONTROL] [6]).

LLIST

suit les mêmes règles que la commande LIST, mais liste sur l'imprimante.

Si vous essayez LLIST alors que l'imprimante n'est pas connectée à votre micro-ordinateur, celui-ci va se bloquer sur cette commande inexécutable. Rendez-vous compte : il est incapable de désobéir, par nature, et vous le mettez dans l'impossibilité d'obéir ! Le mieux à faire est de vite connecter l'imprimante, il sera heureux d'obéir. Si c'est impossible, dites-lui d'oublier tout cela, en appuyant sur le bouton [INIT] placé sur sa face arrière. Bon prince, il garde quand même en mémoire votre programme.

RUN 8 [ENTER] dans le dernier programme provoquerait l'exécution à partir de la ligne 8. Essayez. Sans nettoyage d'écran, puisque la ligne 1 n'est pas exécutée ; l'affichage est celui-ci :

```
OK
RUN 8
ALICE POUR VOUS SERVIR
MONSIEUR
OK
```

RUN n° de ligne
le N° de ligne indique où l'exécution du programme doit commencer. S'il n'est pas indiqué, l'exécution commence à la 1^{re} ligne du programme.
RUN n'est utilisable qu'en mode direct.

III A L'AFFICHE : DES CHIFFRES, DES LETTRES ET DES COULEURS

Voyons maintenant comment améliorer la présentation des instructions et des programmes.

Ne trouvez-vous pas que notre programme est un peu trop "tassé" ? Aérons-le un peu, sautons par exemple 5 lignes entre "QUI ÊTES-VOUS" et "ALICE POUR VOUS SERVIR" : il n'y a qu'à ajouter 5 lignes de programme ne comportant chacune que l'instruction PRINT. Nous ne disposons que de deux lignes ? Ce n'est pas grave, essayez :

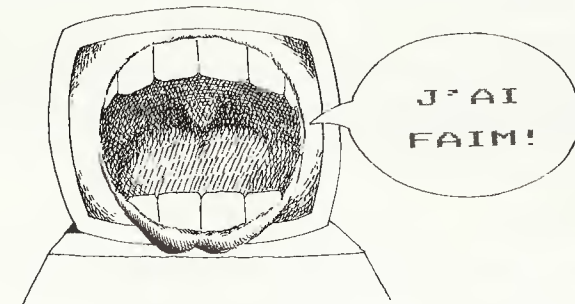
```
6 PRINT : PRINT : PRINT : PRINT : PRINT [ENTER]
RUN [ENTER]
```

Il y a maintenant 5 lignes vides entre "QUI ÊTES-VOUS" et "POUR VOUS SERVIR", autant que d'instructions PRINT à la ligne 6 du programme.

On peut placer plusieurs instructions sur une même ligne, à condition :

- de les séparer par le signe ":", dit "séparateur d'instructions" ;
- de ne pas dépasser au total 127 caractères, y compris les numéros de ligne et les espaces.

Cette quantité maximale correspond au volume de la plus grosse bouchée que l'ordinateur puisse avaler et stocker en mémoire pour une ligne de programme, ou ligne logique. Ce n'est pas le passage à la ligne d'écran, mais la frappe de la touche [ENTER] qui détermine pour l'ordinateur la fin d'une ligne de programme.



Améliorons encore la présentation en décalant l'affichage des lignes pour obtenir la présentation suivante :

```
ALICE POUR VOUS SERVIR
MONSIEUR
```

La ligne d'écran a une largeur de 32 caractères ; il faut donc prévoir 10 espaces avant "ALICE" et 24 espaces avant "MONSIEUR". Il faudrait par conséquent taper 10 ou 24 fois sur la barre d'espacement... Heureusement, il y a plus simple. Tapez :

```
8 PRINT TAB(10) "ALICE POUR VOUS SERVIR" [ENTER]
9 PRINT TAB(24) "MONSIEUR" [ENTER]
RUN [ENTER]
```

... exactement le résultat souhaité : le A de ALICE est sur la 11^e "colonne" et le M de MONSIEUR sur la 25^e "colonne" d'écran.

Au lieu de PRINT, vous pouvez taper [CONTROL] [9] ou bien [?]

Pour vous familiariser avec cette instruction, tapez par exemple :

```
3 PRINT TAB(1) "*" : PRINT TAB(32) "*"
RUN [ENTER]
```

Regardez bien l'écran : la première étoile est affichée sur la deuxième colonne et la seconde sur la première colonne de la troisième ligne. En effet, l'écran est bien divisé en 32 colonnes, mais celles-ci sont numérotées de 0 à 31. Les nombres 32, 64, 96, etc., correspondent donc à la première colonne des lignes suivantes. Pour bien vous familiariser avec cette instruction, amusez-vous à corriger cette ligne pour afficher ces étoiles un peu partout.

LES 4 TYPES D'ÉCRAN

Mais ALICE a encore bien d'autres ressources ! 16 lignes de 32 caractères ne vous suffisent plus ? Alors, partons à la découverte de la touche CLS (CLEAR SCREEN), que vous avez déjà utilisée pour nettoyer l'écran.

Faites :

```
CLS [0] [ENTER]
```

Comme au Pays des Merveilles, l'écran vert d'ALICE se met à grandir et prend possession de la quasi-totalité de l'écran de votre téléviseur. Cette fois, vous disposez de **25 lignes de 40 colonnes**. Pour vous en assurer, il suffit de faire :

```
RUN [ENTER]
```

Votre programme fonctionne parfaitement, mais, sur l'écran, il se présente différemment. Essayez de le modifier pour l'adapter aux nouvelles caractéristiques de l'écran (les 40 colonnes sont numérotées de 0 à 39).

Dans ce mode 40 colonnes, tout se passe exactement comme dans le mode 32 colonnes, sauf en ce qui concerne l'affichage.

Mais ce n'est pas tout !
Essayez maintenant :

```
CLS [80]
```

L'écran ne s'agrandit plus, mais le curseur semble maigrir et s'allonger démesurément. Ce n'est pas étonnant car nous disposons cette fois de 25 lignes de 80 colonnes (numérotées de 0 à 79). Vérifiez-le en adaptant votre programme à ces nouvelles dimensions.

Maintenant, vous savez jongler avec l'affichage du contenu des guillemets (on appelle ce contenu une *chaîne de caractères*).

PRINT donnée :
affiche une donnée à la ligne suivante.
S'il n'y a pas de donnée, le curseur passe directement à la ligne, c'est-à-dire qu'il saute une ligne.

PRINT TAB(n° de colonne) donnée :
affiche une donnée à la ligne suivante et en partant de la colonne qui suit le n° donné entre parenthèses. Ce numéro ne peut pas dépasser 255, quel que soit le mode utilisé ; par contre, le passage à la ligne se fait automatiquement à chaque multiple de 32, 40 ou 80 suivant le mode.

Ça y est ? Vous êtes bien habitué à ce nouvel écran ? Alors, nous allons encore modifier la ligne 9. Tapez :

```
9 PRINT "M" [SHIFT] [0]
```

Et complétez le mot MONSIEUR (n'oubliez pas les guillemets). A l'écran, la fin du mot apparaît en minuscules. Eh oui, en mode 80 colonnes, [SHIFT] [0] ne fait plus passer en mode vidéo inversée, mais donne accès aux minuscules. Pour repasser en majuscules, il suffit de faire à nouveau [SHIFT] [0].

ATTENTION : Les mots du langage BASIC doivent toujours être écrits en majuscules, sinon ALICE répond par un message d'erreur.

Faites maintenant :

```
CLS [81]
```

Cette fois, l'écran devient tout noir ; seuls le traditionnel OK et le curseur clignotant mettent une petite note verte en haut de l'écran. C'est très chic ! Utilisez [SHIFT] [0] : tout se passe comme en mode 80. Nous verrons plus loin (page 119) les véritables particularités du mode 81.

Pour revenir au mode 32 colonnes, il suffit de taper :

CLS 32 **ENTER**

Et les couleurs ?

Vous en verrez de toutes les couleurs dans le chapitre 8, mais dès maintenant nous pouvons nous amuser à jouer un peu avec la couleur du fond de l'écran.

En mode direct, tapez :

CLS 2 **ENTER**, et voici :
l'écran s'est effacé et est devenu d'un beau jaune vif,

CLS 3 **ENTER**, bleu roi !

CLS 4 **ENTER**, rouge !

Explorez ainsi la série des codes de couleurs :

Code	Couleur	Code	Couleur
0	Noir	5	Ivoire
1	Vert	6	Bleu pâle
2	Jaune	7	Mauve
3	Bleu roi	8	Orange
4	Rouge		

Mais vous avez remarqué que la bande verte sur laquelle s'inscrit OK demeure en haut de l'écran, et que les caractères tapés sont de toutes façons noirs sur fond vert. Les affichages de caractères ne se font que sur fond vert. (Rassurez-vous, les caractères graphiques, sous certaines conditions que nous verrons plus tard, s'affichent sur n'importe quelle couleur de fond.)

CLS

efface l'écran.

CLS n° de couleur (de 0 à 8)

efface l'écran et lui donne la couleur sélectionnée ;

en mode 80 : 0 donne un écran noir, les autres chiffres un écran vert ; en

mode 81, 0 donne un écran vert, les autres chiffres un écran noir.

CLS 32

donne accès au mode 32 colonnes.

CLS 40

donne accès au mode 40 colonnes.

CLS 80

donne accès au mode 80 colonnes (écran vert).

CLS 81

donne accès au mode 80 colonnes (écran noir).

Au lieu de CLS, on peut taper **CONTROL** 8.

Le n° de couleur peut être mis entre parenthèses, notamment quand on veut utiliser une expression calculée : CLS (A + 1) donnera un écran bleu roi si A vaut 2.

Amusez-vous maintenant à faire varier les couleurs en passant d'un mode à l'autre (32, 40, 80, 81). Que remarquez-vous en mode 80 et 81 ? Rassurez-vous, nous verrons plus loin (page 119) comment jouer avec les couleurs sur 80 colonnes.

Faisons un dernier essai avec fond de couleur sur le programme tant "bidouillé" jusqu'ici. Remplaçons la ligne 1 qui efface l'écran par la ligne suivante :

```
1 CLS 3 ENTER      (ou toute autre couleur à votre goût)
RUN ENTER          ... seul le bas de l'écran reste bleu. Un peu décevant,
                    mais... on ne peut pas tout faire à la fois.
```

L'affichage de la couleur et celui des caractères alphabétiques ne sont pas compatibles. L'ordinateur peut les juxtaposer, mais pas les mélanger. Il peut juxtaposer Y ■, mais pas écrire Y■, tout comme s'il y avait différence de nature entre ces deux affichages.

De même, en mémoire, il ne peut pas mélanger des valeurs arithmétiques et des caractères. Tout comme vous et moi, d'ailleurs. Auriez-vous l'illusion d'avoir multiplié la quantité figurée ainsi : III ou 3 en ajoutant un "S" à "3" ? Vous ne confondez pas les valeurs et les formes par lesquelles on les figure ; l'ordinateur non plus. Il est encore plus rigoureux que nous.

Un 4 que l'on tape au clavier, par exemple : il faut lui indiquer s'il doit le considérer comme :

– le caractère 4, combinable avec tous les autres caractères.

Dans ce cas, on le met entre guillemets. Ainsi : "4 sous".

– la valeur 4. Le seul fait de ne pas le placer entre guillemets signifie que l'ordinateur doit prendre en compte sa valeur. Ainsi : 4 + 2.

Essayons un peu cela.

Débarrassons-nous de notre vieux programme par un NEW **ENTER**

Au lieu de NEW, on peut taper **CONTROL** 5.

NEW

vide la mémoire,
mais ne nettoie pas l'écran.

(Si vous faites LIST **ENTER**, rien ne se passe. Vous vérifiez ainsi qu'il n'y a plus de programme en mémoire.)

Nettoyez aussi l'écran par CLS **ENTER**.

Tapez le nouveau programme :

```

10 PRINT « QUATRE ET DEUX FONT »
20 PRINT 4 + 2
RUN
QUATRE ET DEUX FONT
6
OK

```

ENTER chaîne
ENTER valeur

A l'exécution de la ligne 20, l'ordinateur a pris en compte les valeurs 4 et 2. sur lesquelles l'addition (codifiée par le signe "+") est possible. Il a effectué le traitement et affiché le résultat (6), en le faisant précéder automatiquement d'un espace (ce qu'il ne fait pas quand il affiche des chaînes de caractères).

Remplaçons maintenant la ligne 10 par :

```

10 PRINT "4 + 2 ="
RUN
4 + 2 =
6
OK

```

ENTER
ENTER Exécution

Détaillons ce qui s'est passé :

```

10 PRINT "4 + 2 ="
4 + 2 =

```

Pas d'analyse du contenu, l'affichage se fait comme s'il s'agissait d'une décalcomanie. Ce sont les guillemets qui déterminent qu'il s'agit d'une chaîne de caractères.

```

20 PRINT 4 + 2
6
C'est clair ?

```

Calcul et affichage d'une représentation de la valeur calculée, car pas de guillemets.

POUR VOUS MUSCLER

Quelques (petits) exercices, dont vous trouverez le corrigé en fin de manuel.

1. Analysez les étapes
2. Traduisez en BASIC
3. Entrez en mémoire
4. Exécutez

Avec un papier
 et un crayon

les programmes qui produisent les résultats suivants à l'écran :

```

BIZARRE, BIZARRE...
OK

```

puis celui-ci :

```

64 + 30 =
94
18 + 15 =
33
OK

```

En obtenant 94 et 33 par un calcul de l'ordinateur, comme ci-dessus.

et enfin celui-là :

```

Vert → JE CONNAIS LES COULEURS, LA PREUVE :
OK
Une couleur →
de votre choix.

```

N'oubliez pas de taper NEW **ENTER** avant d'entrer chaque programme, sinon il se mélangerait avec le précédent.



Chapitre 2

Alice vous interroge, répondez

Nous savons réaliser un programme très simple. Apprenons maintenant à manier des variables et à dialoguer avec Alice.

Jusqu'à présent, nous n'avons utilisé la mémoire de l'ordinateur que pour stocker des lignes de programme, du type :

30	PRINT	38
N° de ligne	Instruction	Constante

(On appelle constante une donnée fixée en même temps que l'instruction et qu'aucune autre instruction du programme ne fait changer.)

Cela donne des programmes très "figés". Cent fois exécuté, le programme d'addition du chapitre précédent nous donnera cent fois le même affichage. Il serait maintenant intéressant de pouvoir faire varier les nombres à additionner. Lors de l'exécution du programme, l'ordinateur nous les demanderait et nous donnerait instantanément le résultat. C'est cet **échange** que nous allons apprendre à programmer dans ce chapitre.

I DES "VARIABLES" POUR PLUS DE SOUPLESSE

A RENCONTRE D'UN IDENTIFICATEUR

Pour comprendre ce que sont un identificateur et une variable, voici un nouveau programme que vous allez taper.

Assurez-vous d'abord par NEW **ENTER** que vous videz la mémoire de son contenu précédent, sinon vous vous retrouveriez avec d'anciennes lignes de programme intercalées dans les nouvelles. (L'ordinateur considère toutes les instructions qu'il a en mémoire comme appartenant au même programme, car il ne peut avoir en mémoire plus d'un programme à la fois.)

Apparemment, il ne se passe rien, car le nettoyage de la mémoire est une opération interne, et l'écran reste tel qu'il était. Pour votre confort visuel, effacez aussi l'écran par CLS **ENTER**.

Tapez les lignes suivantes qui donnent à l'exécution :

10 CLS	* Effacement d'écran
20 LET A = 24	Mise en mémoire des valeurs de A et B.
30 LET B = 12 + 24	
40 PRINT : PRINT	Saut de 2 lignes.

* Toutes les lignes de programme sont validées par ENTER, vous le savez maintenant, nous ne le signalerons plus.

50 PRINT TAB (12) "ADDITION"	ADDITION
60 PRINT : PRINT TAB (11) A	24
70 PRINT TAB (10) "+"	+
80 PRINT TAB (11) B	36
90 PRINT TAB (10) "-----" : PRINT	-----
100 PRINT TAB (11) A + B	60

Examinons d'abord les lignes 20 et 30.

LET A = 24 signifie : "que A soit affecté de la valeur 24" ou "que A reçoive la valeur 24".

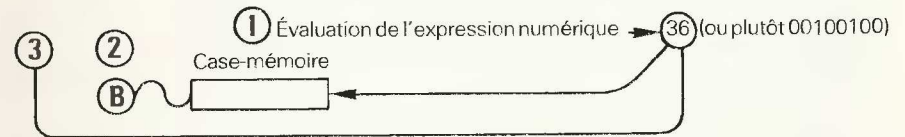
Qu'est-ce que A ? un **identificateur**.

Lorsque l'ordinateur rencontre l'identificateur A pour la première fois au cours de l'exécution, il lui attribue une "case" en mémoire et met dans cette case la donnée qui figure à droite du signe "=". Imaginez l'affectation d'une valeur à un identificateur comme le remplissage d'un casier désigné par une étiquette :



LET B = 12 + 24, à la ligne 30, est à peine moins simple. L'ordre des opérations est le même :

LET B = 12 + 24



Les identificateurs A et B correspondent à la même valeur tout au long du programme. Ce sont des **constantes**, pas encore des "variables", mais nous y venons.

B VIE ET MŒURS D'UNE VARIABLE

Remplaçons la ligne 30 par :

30 LET B = 12 + A

l'exécutez : aucun changement, A vaut 24.

Mais tapez maintenant :

33 LET A = A + 10

35 LET B = 12 + A (même ligne que la n° 30)

Devinez quel résultat vous allez obtenir...

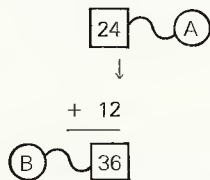
Exécutez. Aviez-vous deviné juste ?

Sinon, voici un schéma du déroulement des instructions :

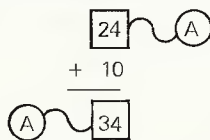
20 LET A = 24



30 LET B = 12 + A

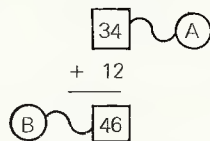


33 LET A = A + 10



La nouvelle valeur de A "écrase" l'ancienne. Place aux jeunes !

35 LET B = 12 + A



Dans la suite des instructions, A "vaudra" désormais 34 et B 46.

Tout est clair ?

Quelques remarques encore sur ces lignes :

1^{re} remarque

La ligne 30 ne sert plus à rien, on peut l'effacer sans changer le résultat. Essayez. Peut-on en dire autant de la ligne 20 ?

Regardez bien la ligne 33, où se fait la dernière affectation de A. Sa nouvelle valeur est calculée à partir de sa valeur précédente. La **variable** A avait été **initialisée** en ligne 20, c'est-à-dire qu'elle y avait reçu sa valeur initiale (ou de départ).

Si cette valeur n'est pas précisée au moment où le contenu de A doit être utilisé, comme en 33, l'ordinateur initialise systématiquement la variable à 0.

Pour vérifier ce fait, supprimez la ligne 20 en tapant 20 **ENTER**.

Exécutez :

$$\begin{array}{r} 10 \\ + 12 \\ \hline 22 \end{array}$$

Par rapport à l'exécution précédente, il manque 24. Les calculs sur A se sont faits à partir de la valeur 0 (initialisation automatique) au lieu de se faire à partir de la valeur 24.

2^e remarque

Dans l'instruction d'affectation, LET est facultatif. Il suffisait d'écrire :

20 A = 24. Essayez.

Mais la ligne 33, écrite ainsi, devient :

33 A = A + 10 ... Horreur mathématique !

Écrivons-la ainsi pour aller plus vite, mais avec une restriction mentale. Comprenons bien que ce signe "=" ne signifie pas "égale" mais "reçoit la valeur de". Ce n'est pas la benoîte constatation d'une égalité, mais bel et bien une instruction d'affectation sous ce seul signe.

3^e remarque

Comment choisir un identificateur ?

L'ordinateur reconnaît un identificateur sur 2 caractères au maximum, combinés sous la forme :

Lettre, lettre	ou Lettre, chiffre
AG, PV, NB	A9, X1, R5
A, P, V	

Choisissez de préférence des noms qui vous "disent quelque chose", qui vous rappellent la signification du contenu de la variable. Ainsi, PV est parfait pour contenir la valeur du montant d'une contravention !

Vous pouvez employer plus de deux caractères pour votre usage. Ils sont acceptés par l'ordinateur comme un nom de variable, mais seuls les 2 premiers sont reconnus ; Exemple : si vous avez une variable que vous appelez BOCAL, l'ordinateur la connaît comme BO. N'en appelez pas une autre BOUT, car elle aussi serait identifiée comme BO et considérée comme la même variable que la précédente.

N'utilisez pas non plus dans un nom de variable un mot du langage BASIC. La variable PRUNE provoquerait une "erreur de syntaxe" à l'exécution, car le mot RUN s'y trouve en entier, et l'ordinateur le reconnaît et ne sait que faire. Par contre, une variable RU serait acceptée.

Sur deux caractères, les seuls noms auxquels vos variables n'aient pas droit sont ON, IF, TO, car ce sont des mots du langage BASIC, qui seraient alors source d'ambiguïté lors de leur interprétation par l'ordinateur.

LET variable = expression

assigne à la variable la valeur de l'expression située à droite du signe "=". Le mot LET est facultatif, son absence ne change rien.

DES CHAÎNES ET DES NOMBRES

Dans le chapitre précédent, nous avons soigneusement distingué :

- les expressions **chaînes** (de caractères) comme "BONJOUR" ou "10 ANS" ;
- les expressions **numériques** comme 18, 21 + 43.

Lors d'une affectation, on préserve cette distinction.

L'identificateur simple correspond à une valeur numérique,

l'identificateur suivi du signe "\$" correspond à une chaîne de caractères.

Exemple : LET A\$ = "BONJOUR"
LET AD\$ = "15 + 30 ="

Écritures incorrectes : LET A\$ = B + 12
LET A = "LE CHAT"

Ces instructions incorrectes mélangent les types de données. A l'exécution, elles provoquent l'affichage d'un message d'erreur

?TM ERROR

qui signifie "types incompatibles" ("type mismatch" en anglais) et, bien entendu, l'exécution s'arrête là.

Tout va bien jusque-là ?

Nous ne sommes pas encore arrivés au but fixé en début de chapitre, qui était : faire changer les données à chaque exécution du programme et obtenir chaque fois un résultat différent. Mais nous avons parcouru la plus grande partie du chemin.

Avant d'aller plus loin, il serait bon de faire une halte et de vous servir de ce que vous avez appris.

Voici un exercice dont l'énoncé est suivi d'un corrigé détaillé. Il serait excellent que vous ne lisiez du corrigé que ce qui vous est indispensable pour continuer. Si vous n'avez besoin de rien pour arriver jusqu'à la solution, tant mieux.

Si votre solution "tourne" bien sur l'ordinateur, ne vous inquiétez nullement d'une éventuelle différence avec notre solution : le passage sur l'ordinateur est le seul test valable. Si ça marche, alors la solution est bonne. Et il y a toujours plusieurs démarches possibles pour arriver à la solution d'un problème.

Exercice guidé

Énoncé : Soit le prix de la nuit d'hôtel à plein tarif : 185 F par personne. Si vous voyagez en groupe, vous avez droit à une réduction de 10 %. Mais si vous occupez seul(e) votre chambre, le tarif est majoré, par rapport au précédent, de 15 %. Bâissez un programme qui affiche le plein tarif, le tarif de groupe et le tarif majoré pour une personne seule.

(Le signe de la division est "/"; celui de la multiplication "*".)

Définition du résultat

L'énoncé la contient. Il n'est pas indispensable ici de déterminer une présentation particulière de l'écran.

Analyse

1. Le *plein tarif* d'une nuit est connu. Disons tout de suite que NT = 185 (NT pour "nuit"). Il suffira de l'afficher, précédé de "plein tarif".

2. Le *tarif de groupe* pour une nuit, c'est le plein tarif moins la réduction qu'on peut calculer d'abord :

- - calcul de la réduction R : $R = NT * 10 / 100$ ou $R = NT / 10$
- calcul de la nouvelle valeur de NT : $NT = NT - R$
- affichage de NT.

3. Le *tarif pour une personne seule*, c'est le tarif de groupe, soit la dernière valeur de NT, plus une majoration (M) de 15 %.

- - calcul de la majoration M : $M = NT * 15 / 100$
- calcul de la nouvelle valeur de NT : $NT = NT + M$
- affichage de NT.

Traduction en BASIC

```
10 CLS
20 LET NT = 185
30 PRINT "PLEIN TARIF"
40 PRINT NT
50 LET R = NT * 10 / 100
60 LET NT = NT - R
70 PRINT
  "TARIF DE GROUPE"
80 PRINT NT
90 LET M = NT * 15 / 100
100 LET NT = NT + M
110 PRINT "MAJORE POUR 1 PERS."
120 PRINT NT
```

Description du traitement

Initialisation de la variable NT.

Calcul de la réduction, le résultat est affecté à R.

Calcul de la nuit au tarif réduit ; le résultat est affecté à NT qui change de valeur.

Calcul de la majoration, résultat affecté à M.
Calcul de la nuit au tarif majoré. NT change encore de valeur.

Les lignes 40, 80 et 120 sont identiques, mais l'instruction PRINT NT donne chaque fois l'affichage d'une valeur différente : NT est bien une variable.

Remarquons que nous pourrions supprimer la ligne 100 et écrire directement en ligne 120 PRINT NT + M sans rien changer au résultat.

Mais pourrions-nous effacer aussi la ligne 60 et écrire 80 PRINT NT-R ?

Quelle valeur aurait NT en arrivant sur la ligne 90 ?

Ce serait toujours 185, la réduction pour les groupes aurait été calculée et affichée en ligne 80, mais le résultat n'aurait été affecté nulle part à NT, il n'en resterait pas trace dans la mémoire.

Il n'y a pas de mise en mémoire d'une donnée sans affectation de cette donnée à un identificateur.

Nous avons fini cet exercice qui n'était pas particulièrement facile. Mais les perfectionnements que ce petit programme nécessite nous amèneront à notre but.

Pour mieux faire...

Il s'agit du programme, pas de vous !

Il est bien dommage qu'il ne fonctionne que sur un seul tarif. Les tarifs de base peuvent varier selon la catégorie de l'hôtel, selon l'époque de l'année et avec la hausse des prix.

Comment faire pour utiliser ce même programme ?

1^{re} solution :

changer la ligne 20 et écrire 20 LET NT = 200.

Bien sûr, c'est possible. Mais imaginez un employé d'agence de voyages, qui utilise

le programme ; il ne sait pas forcément modifier un programme en BASIC, et la manœuvre, à chaque exécution, est fastidieuse.

2^e solution :

pouvoir "entrer" à partir du clavier la valeur de NT au début de chaque exécution. Nous y voilà !

II ENTREZ, ENTREZ (DES DONNÉES) ET VOUS VERREZ



Dans notre programme, remplaçons la ligne 20 par

```
20 INPUT NT
RUN [ENTER]
```

?

C'est tout ce que vous voyez se passer à l'écran.

Le point d'interrogation est un message de l'ordinateur qui attend une réponse de l'utilisateur et l'attendra jusqu'à ce qu'elle lui soit donnée...

L'instruction *INPUT* suspend l'exécution du programme dans l'attente d'une entrée de donnée à partir du clavier. Vous tapez :

```
? 185 [ENTER]
```

L'exécution reprend aussitôt et vous voyez sur l'écran :

1^e exécution

```
? 185 [ENTER]
PLEIN TARIF
185
TARIF DE GROUPE
166.5
MAJORE POUR 1 PERS.
191.475
OK
```

Exécutez à nouveau :

2^e exécution

```
? 200 [ENTER]
PLEIN TARIF
200
TARIF DE GROUPE
180
MAJORE POUR 1 PERS.
207
OK
```

Essayez encore 2 à 3 fois en entrant chaque fois une valeur différente.

Qu'a fait l'instruction *INPUT NT* ? Elle a :

1. interrompu l'exécution du programme et vous a très momentanément rendu l'initiative ;
2. dès que vous avez tapé au clavier et validé votre réponse (200, par exemple) la case-mémoire attribuée à l'identificateur NT a été affectée de la valeur 200.

Remarque : puisque NT est de type numérique, l'ordinateur n'acceptera que des chiffres en entrée. Lancez à nouveau l'exécution et commettez l'erreur suivante :

```
? A2
```

```
? REDO
```

```
? 200
```

```
PLEIN TARIF
```

Message vous signalant d'avoir à recommencer.

Suite de l'exécution...

Raison de plus pour indiquer à l'utilisateur quel type de données on attend de lui. Pour utiliser notre programme plus commodément, il faudrait en réalité effacer la ligne 20 et placer *INPUT NT* en ligne 40.

```
10 CLS
```

```
30 PRINT "PLEIN TARIF" _____> PLEIN TARIF
```

```
40 INPUT NT _____> ?
```

En pratique, l'instruction *INPUT* est presque toujours précédée d'un message à afficher à l'écran. On demande une réponse de la part de l'utilisateur ; la moindre des choses est de lui poser une question ! C'est pourquoi l'écriture de l'instruction *INPUT* peut combiner les deux.

Effacez la ligne 30 et tapez :

```
40 INPUT "PLEIN TARIF" ; NT (N'omettez pas le " ; ".)
```

```
RUN [ENTER]
```

```
PLEIN TARIF? 200
```

Le point d'interrogation et le curseur restent sur la même ligne que le message.

INPUT "message"; variable

après avoir affiché le message, attend l'entrée d'une donnée à partir du clavier et l'attribue à la variable. La donnée et la variable doivent être de même type.

N.B. : le point virgule empêche le retour à la ligne.

```
40 INPUT "PLEIN TARIF"; NT est l'équivalent exact de :
30 PRINT "PLEIN TARIF"; :INPUT NT
```

Au lieu de INPUT, on peut taper **CONTROL** **@**

Vous avez maintenant pris connaissance de notions de base essentielles. Il serait bon que maintenant vous vous exerciez.

POUR VOUS MUSCLER

Une série d'exercices dont vous trouverez le corrigé à la fin. Sauf le premier, tous vous demandent de faire des programmes. Ils ont chacun leur personnalité, vous allez le voir.

1. Le studieux

Dans le programme d'addition de la page 46, que va-t-il se passer si on écrit en ligne 50 PRINT "SOUSTRACTION" ?

Quelle ligne faudrait-il transformer pour que l'opération soit bien une soustraction ?
Quels sont les numéros des lignes qui contiennent une instruction d'affichage ?
Quels sont les numéros des lignes qui contiennent une instruction d'opération ou de mise en mémoire ?

2. L'indiscret

Faites demander par l'ordinateur l'année de naissance d'une personne et faites-lui afficher l'âge de cette personne.

3. Le "m'as-tu-vu"

Il demande un nombre compris entre 1 et 8 et, selon la réponse, colore l'écran dans l'une des 8 teintes possibles. (Utilisez CLS variable).

4. L'économe

On lui donne le nombre de kilomètres parcourus en voiture, la consommation en litres et il calcule la consommation en litres aux cent kilomètres.

5. Le distrait

Vous lui donnez deux nombres. il en fait l'addition, affiche un résultat faux, puis se reprend en disant : « Oh pardon ! C'était... », et cette fois vous donne le résultat exact.

6. Le flatteur

Il vous demande de lui donner un adjectif flatteur (ex. : sympathique), puis votre prénom (ex. : Marcel). Il affiche au milieu d'un écran vide « BONJOUR. SYMPATHIQUE MARCEL ! ».

Chapitre 3 Un peu de traitement

Maintenant que nous savons communiquer avec ALICE, nous allons pouvoir développer des programmes de traitement pour calculer, modifier et présenter les données entrées en mémoire.

De l'entrée d'une donnée à sa sortie sur écran, bien des choses peuvent se passer. Et les traitements que notre micro-ordinateur ALICE est capable de faire subir à une donnée méritent un chapitre à eux seuls.

Ce chapitre n'apportera rien de nouveau sur la construction des programmes et nous continuerons à utiliser les entrées/sorties (nous enjoliverons même les affichages). Mais nous ferons un peu plus travailler l'ordinateur. Jusqu'à présent, il ne nous a fait que quelques opérations simples sur des nombres. Pour lui demander davantage, sachons quoi lui demander.

I UN CHAMPION DE CALCUL MENTAL

Les cinq signes d'opérations arithmétiques, appelés "opérateurs", s'écrivent ainsi :

+ addition
- soustraction
* multiplication
/ division
↑ exponentiation ou « élévation à une puissance » (obtenu par **CONTROL** **Z**)

Comment s'utilisent plusieurs opérateurs dans une même expression ?

Essayons rapidement :

```
10 CLS
20 INPUT "A ="; A : INPUT "B ="; B. (Entrer 2 nombres.)
30 PRINT "A + B / 2 ="; A + B / 2 (Calcul, affichage de l'opération et de son résultat)
40 PRINT "A * B ↑ 2 + A ="; A * B ↑ 2 + A
50 PRINT "A - 2 * B / A ="; A - 2 * B / A
60 PRINT "(A - 2) * B / A ="; (A - 2) * B / A
RUN ENTER
```

Et voici le résultat, quasi instantané :

```
A = ?4
B = ?7
A + B / 2 = 7.5
A * B ↑ 2 + A = 200
A - 2 * B / A = .5
(A - 2) * B / A = 3.5
OK
```

(.5 signifie 0,5. Comme sur une calculette, les virgules décimales sont remplacées par des points.)

L'ordinateur a respecté les **priorités** suivantes **entre les opérateurs** :

1. Évaluation de l'expression contenue à l'intérieur des parenthèses. Quand il y en a plusieurs, l'intérieur des parenthèses les plus internes a priorité.
2. S'il n'y a pas de parenthèses, le calcul se fait d'abord sur :
 - l'élevation à une puissance (\uparrow);
 - puis toutes les multiplications et divisions;
 - enfin les additions et soustractions.
3. Si deux opérateurs ont priorité nulle (comme deux $*$, ou un $*$ et un $/$, ou un $+$ et un $-$), les opérations s'effectuent de gauche à droite.

Détaillons notre exemple :

$$\begin{array}{l} (2) \quad (1) \\ A + B / 2 \\ \underbrace{\qquad\qquad\qquad}_{3.5} \\ 4 + 3.5 = 7.5 \end{array}$$

$$\begin{array}{l} (2) \quad (1) \quad (3) \\ A * B \uparrow 2 + A \\ \underbrace{\qquad\qquad\qquad}_{49} \\ \underbrace{\qquad\qquad\qquad}_{196} \\ 200 \end{array}$$

$$\begin{array}{l} (3) \quad (1) \quad (2) \\ A - 2 * B / A \\ \underbrace{\qquad\qquad\qquad}_{14} \\ \underbrace{\qquad\qquad\qquad}_{3.5} \\ 0.5 \text{ (noté .5)} \end{array}$$

$$\begin{array}{l} (1) \quad (2) \quad (3) \\ (A - 2) * B / A \\ \underbrace{\qquad\qquad\qquad}_{2} \\ \underbrace{\qquad\qquad\qquad}_{14} \\ \underbrace{\qquad\qquad\qquad}_{3.5} \end{array}$$

Essayez maintenant de donner à A et B des valeurs plus complexes et voyez comme le calcul se fait rapidement.

Essayez $A = 199.23$ et $B = 42.66$

Le calcul de $A * B \uparrow 2 + 4 = 362773.046$ pour ALICE;
362773.045788 pour vous si vous vérifiez sur le papier.

ALICE ne vous donnera jamais que 9 chiffres, utilisés en priorité par la partie entière. Exemple :

PRINT 888888887 / 2 ENTER
444444444

Au lieu de 444444443.5.

PRINT 888888889 / 2 ENTER
444444445

Au lieu de 444444444.5.

PRINT 1 / 3 ENTER
.333333334

Au lieu d'une infinité de 3 pour la partie décimale.

→ La précision des résultats figurés se limite à une représentation sur 9 chiffres. Quand d'autres chiffres devraient suivre celui qui se trouve le plus à droite, ce dernier est élevé à la valeur immédiatement supérieure.

Cela ne concerne que la représentation; le calcul se fait tout de même de façon beaucoup plus précise :

PRINT (1 / 3) * 2 ENTER
666666667

Et non .666666668 comme ce serait le cas si .333333334 était la valeur considérée comme résultat de l'opération (1 / 3).

Au-delà de 9 chiffres significatifs, ALICE, comme presque tous les ordinateurs, utilise la notation scientifique pour figurer les nombres.

Exemple : 6.17269877E+11

où E+11 signifie qu'il faut décaler le point décimal de 11 chiffres vers la droite pour obtenir la valeur approchée du résultat.

Ici, ce serait :

617269877000

E-8 signifierait qu'il faut décaler le point décimal de 8 chiffres vers la gauche.

Exemple : 5.00300002E-8 représente 0.0000000500300002

On peut ainsi figurer des nombres compris entre -10^{38} et $+10^{38}$. Au-delà, on dépasserait les capacités de l'ordinateur qui afficherait alors le message "OV ERROR" signifiant qu'il est submergé ("Overflow")

Les chaînes aussi ont leur opérateur !

Ou plus exactement elles empruntent le signe "+" aux opérations arithmétiques pour l'accueillir à leur usage et s'en servir de "maillon".

Deux chaînes reliées par le signe "+" sont mises bout à bout sans espace entre elles. Elles sont "concaténées".

Exemple : supposons que vous souhaitez voir ALICE s'exprimer comme certain personnage de bande dessinée et que, dans un programme où l'utilisateur est questionné, une réponse fautive soit toujours signalée par le message "Erreur funeste, estimable (ici, le prénom de l'utilisateur)".

Voici ce que vous écririez en début de programme :

```
10 INPUT "VOTRE PRENOM"; P$
20 M$ = "ERREUR FUNESTE, ESTIMABLE " + P$
```

La chaîne M\$ se terminera bien par le prénom de l'utilisateur. Remarquez qu'il faut ajouter un espace entre la fin du mot ESTIMABLE et ", sinon le prénom sera collé au mot qui le précède.

Chaque fois que vous voudrez afficher ce message, il suffira de placer une instruction PRINT M\$.

II OÙ L'ON SOIGNE LA PRÉSENTATION

Au fur et à mesure que nous savons faire faire plus de choses à ALICE, nous devrions mettre un peu mieux en valeur ses résultats.

En écrivant autrement les instructions INPUT et PRINT on peut changer la présentation.

Essayez en mode direct :

```
A$ = "DEPENSES"   [ENTER]
B$ = "RECETTES"   [ENTER]
PRINT A$; B$      [ENTER]
DEPENSESRECETTES
OK
```

Pour les décoller, il faut taper :

```
PRINT A$; " "; B$ [ENTER]
DEPENSES RECETTES
OK
```

Mais essayez ceci :
(virgule de séparation)

```
PRINT A$, B$      [ENTER]
DEPENSES          RECETTES
OK
```

La première lettre de B\$ est venue s'afficher sur la 17^e colonne de l'écran. La virgule de séparation crée une tabulation automatique qui partage l'écran en 2 blocs de 16 caractères.

PRINT donnée ; donnée ou **PRINT donnée ; PRINT donnée**
Le ";" fait afficher les données sans espace entre elles ni passage à la ligne.
(Toutefois si la 2^e donnée est numérique et de valeur positive, elle sera tout de même précédée d'un espace.)
Il n'y a pas de limite au nombre de données figurant derrière PRINT.

PRINT donnée, donnée ou **PRINT, donnée**
La "," fait afficher la 1^{re} lettre de la 2^e donnée sur la 1^{re} colonne du prochain bloc de 16 caractères.

Ces variations s'appliquent aussi à l'instruction LPRINT qui produit les mêmes effets, mais sur l'imprimante.

Application immédiate :

```
10 CLS
20 INPUT "DESTINATION"; D$
30 INPUT "DEPART DE"; DP$
40 INPUT "HEURE DE DEPART"; HD$
50 INPUT "HEURE D'ARRIVEE"; HA$
60 CLS : PRINT : PRINT : PRINT : PRINT
70 PRINT "DEPART", "ARRIVEE"
80 PRINT DP$, D$
90 PRINT HD$, HA$
```

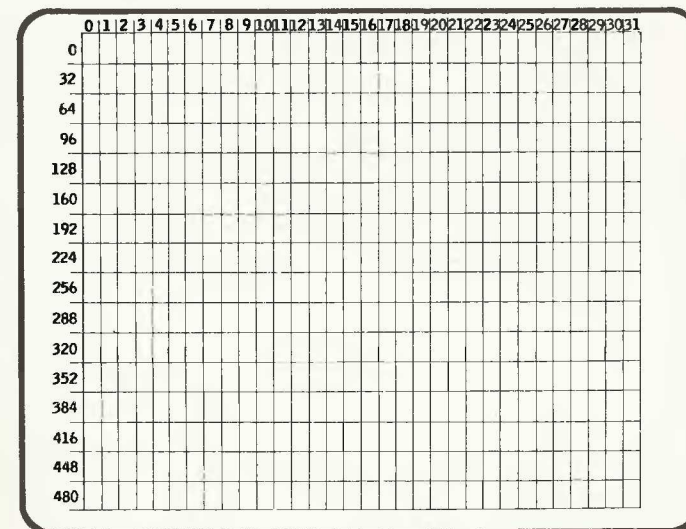
Résultat

```
DESTINATION? MILAN
DEPART DE? PARIS
HEURE DE DEPART? 12 H 30
HEURE D'ARRIVEE? 14 H 00
```

Effacement d'écran
Saut de quatre lignes

```
DEPART  ARRIVEE
PARIS   MILAN
12 H 30 14 H 00
```

PRINT@ C'est une variante de la commande PRINT qui permet de spécifier directement quelle "case" de l'écran on choisit pour l'affichage, en utilisant les numéros de la grille d'écran ci-dessous :



Dessin N° 8

Tapez CLS 32. L'écran est divisé en 512 positions d'impression (32*16) : le coin supérieur gauche occupe la position 0, celui de droite la position 31 et l'on continue en "sautant" de ligne en ligne jusqu'à la case 511 dans le coin inférieur droit, le milieu occupant la position 239.

Pour vérifier, essayez :

```
CLS   
PRINT@ 239,"A" 
```

Vous pouvez utiliser la même instruction en mode 40 ou 80 colonnes. Vous disposez alors de 1000 (40*25) ou 2000 (80*25) positions d'impression. Cherchez le milieu ! Cette instruction, qui permet de faire "remonter" le curseur, est très intéressante, notamment pour les caractères semi-graphiques, comme nous le verrons plus tard.

PRINT@ exp. numérique, donnée
affiche une donnée à l'écran, à l'emplacement désigné par l'expression numérique dont la valeur doit être comprise entre 0 et 511, 999 ou 1999 suivant le mode utilisé.

Et si vous voulez utiliser votre imprimante ?

Pour le détail des branchements, reportez-vous à la page 17. Supposons la connexion établie.

LPRINT

produit exactement les mêmes effets que la commande PRINT, mais à l'imprimante et non sur l'écran.
La seule exception est l'usage de PRINT@, interdit sur l'imprimante, car il permet de faire remonter le curseur, ce que l'imprimante est incapable de faire.

Si toutefois vous aviez tenté d'exécuter la commande ou l'instruction LPRINT alors que votre imprimante n'est pas connectée, l'ordinateur est bloqué (on dit couramment qu'il est "planté"), et le mieux que vous ayez à faire est d'appuyer sur le bouton d'initialisation situé sur la face arrière de l'ordinateur.

Tout comme PRINT, l'instruction INPUT peut être suivie de plusieurs données à entrer.

Mais le message à afficher doit être unique et les identificateurs séparés par des virgules.

INPUT "chaîne de caractères"; ident., ident., ident.

permet l'entrée sous une seule instruction et sur une seule ligne d'écran de plusieurs données. A l'entrée au clavier, les données doivent être séparées par des virgules.

INPUT s'obtient aussi par

En voici un exemple :

```
10 INPUT "3 NOMBRES"; A, B, C  
20 PRINT A; B; C
```

A l'exécution, cela donne :

```
3 NOMBRES ? 23, 67, 89   
23 67 89  
OK
```

Ce qui correspond au mode d'emploi normal d'un INPUT multiple. Mais on pourrait aussi l'exécuter de la manière suivante :

```
3 NOMBRES ? 23   
? ? 67   
? ? 89   
23 67 89  
OK
```

Et si l'on donne plus de nombres que prévu ? Essayez :

```
3 NOMBRES ? 23, 67, 89, 11  
? EXTRA IGNORED  
23 67 89  
OK
```

Les 3 premiers nombres ont été entrés ; un message vous fait savoir que le surnuméraire a été ignoré et l'exécution du programme continue normalement. Comme après un PRINT, on peut juxtaposer les données de type numérique et de type chaîne.

Exemple :

```
10 INPUT "NOM ET NOTE / 20"; N$, NO  
20 PRINT N$, NO  
RUN   
NOM ET NOTE / 20 ? MUZEAU, 14  
MUZEAU 14  
OK
```

MAIS VOUS N'AVEZ TOUJOURS PAS ENTENDU LA VOIX D'ALICE !

Pour lui faire produire un son, utilisez :

SOUND exp. numérique, exp. numérique.

La 1^{re} expression numérique doit être comprise entre 1 et 255, elle donne le TON.

La 2^e spécifie la durée pendant laquelle le ton est généré. Sa valeur doit aussi être comprise entre 1 et 255. Chaque unité de durée représente à peu près 7.5/100^e de seconde.

On peut aussi faire **CONTROL** **K**.

Avant d'essayer **SOUND**, vérifiez que le réglage du volume sonore de votre téléviseur permet d'obtenir un son.

Essayez en mode direct :

SOUND 1,30 **ENTER** → Le son le plus grave.

SOUND 255,30 **ENTER** → Le son le plus aigu.

Voir la correspondance avec les notes de musique page 171. A vous de composer de petits airs...

Et maintenant ALICE vous salue en musique !

Tapez :

```
10 PRINT "C'EST FINI, AU REVOIR"
```

```
20 D = 6
```

```
30 T = 100
```

```
40 SOUND T,D
```

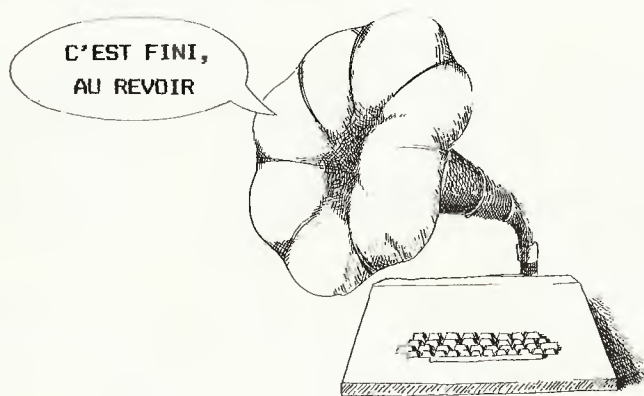
```
50 D = D/2
```

```
60 SOUND T,D : SOUND T,D : SOUND T,D
```

```
70 SOUND T,D
```

```
80 SOUND T * 1.4, D * 3
```

```
RUN ENTER → C'EST FINI, AU REVOIR  
et un petit air...
```



Pas très romantique, il est vrai, mais nous ferons mieux plus tard !

Avant de poursuivre plus avant l'exploration des traitements de données, exercez-vous et augmentez votre dextérité dans la pratique des affichages.

Exercice guidé

Énoncé : Faites un programme qui demande un nom de ville, un numéro de mois, un quantième (c'est-à-dire un numéro de jour dans le mois, comme "12" dans "12 octobre") et l'année, de façon à pouvoir afficher ensuite :

RDUEN, LE 12/10/83 comme en-tête d'une lettre.

Analyse

1. Entrer d'abord la série des 4 données (éventuellement sous 2 instructions **INPUT** seulement).

2. Effacer l'écran et déplacer le curseur en milieu de ligne par un **PRINT**,

3. Afficher la succession d'identificateurs et de chaînes de caractères.

"ROUEN", c'est le contenu de **V\$**;

" . LE ", chaîne de caractères;

12, c'est le contenu de l'identificateur **Q\$**. (Utiliser une chaîne évite d'avoir des espaces affichés avant une donnée numérique.)

Traduction en BASIC

```
10 INPUT "DANS QUELLE VILLE SOMMES-NOUS ET EN QUELLE ANNEE"; V$, AN$
```

```
20 INPUT "DONNEZ LE NO DU MOIS ET LE QUANTIEME"; NM$, Q$
```

```
30 CLS
```

```
40 PRINT, V$; ", LE "; Q$; "/" ; NM$; "/" ; AN$
```

Tout se passe comme prévu, si toutefois vous n'avez pas oublié la virgule après **PRINT**.

Ici, on aurait pu placer le signe de la concaténation ("+") à la place des ";" de la ligne 40 sans aucune différence visible.

III LES FONCTIONS OU L'ART D'ACCOMMODER LES DONNÉES

Une fonction, c'est la relation qui existe entre deux quantités ou données, telle que la variation de l'une entraîne la variation correspondante de l'autre.

Par exemple, la relation entre A et B lorsque A = 5 si B = 10, A = 7 si B = 14, A = 12 si B = 24 pourrait s'appeler la fonction MOITIÉ et se définir par l'écriture ($* 0.5$). Inversement, si c'est B qui varie par rapport à A (vaut 18 quand A vaut 9, etc.), cette relation pourrait s'appeler la fonction DOUBLE et s'écrire ($* 2$).

Dans cet exemple très simple, nous n'avons pas besoin d'un mot particulier du BASIC pour faire calculer cette fonction. Écrire $A = B * 0.5$ est aussi rapide qu'écrire $A = \text{MOITIÉ}(B)$.

Mais il est des fonctions moins simples, exigeant plusieurs calculs, que nous serons bien contents de trouver toutes prêtes.

A LES FONCTIONS NUMÉRIQUES

Les amateurs de calculs mathématiques trouveront plus loin la description des fonctions mathématiques et trigonométriques. Ils en verront facilement l'application.

Que les autres se rassurent : il existe des fonctions appliquées à des données numériques qui s'utilisent couramment en dehors des applications mathématiques sophistiquées. C'est le cas de la fonction INT.

La fonction INT

INT est l'abréviation de INTEGER ou ENTIER.

Essayez :

```
PRINT INT (183.45) [ENTER]      PRINT INT (100/3) [ENTER]
183                               33
PRINT INT (- 3.78) [ENTER]     PRINT INT (54) [ENTER]
- 4                               54
```

INT (exp. numérique)
arrondit la valeur de l'expression numérique à l'entier immédiatement inférieur.
Si elle est déjà une valeur entière, elle reste inchangée.

INT s'obtient aussi par [CONTROL] [C].

Notez qu'une fonction s'emploie comme s'emploierait son résultat. Elle s'emploie comme une donnée, pas comme une instruction.

Voici un petit *exercice d'application* de la fonction INT.

Énoncé : Vous voulez charger avec des caisses toutes de même poids un monte-charge. Compte tenu de la charge maximale que peut supporter le monte-charge, vous voulez savoir combien de caisses peuvent y être contenues à chaque voyage (en nombre entier, bien sûr).

Le petit programme suivant nous le calcule :

```
10 INPUT "CHARGE MAXI" ; CH
20 INPUT "POIDS D'UNE CAISSE" ; P
30 NC = INT (CH/P)
40 PRINT "ON PEUT CHARGER" ; NC ; "CAISSES"
RUN [ENTER]
```

```
CHARGE MAXI ? 430
POIDS D'UNE CAISSE ? 57.5
ON PEUT CHARGER 7 CAISSES
OK
```

La fonction RND : c'est la bénédiction des joueurs, la roulette de l'ordinateur.

RND est une abréviation de RANDOM ou HASARD.

On l'obtient aussi par [CONTROL] [V].

RND (exp. numérique)
renvoie une valeur numérique entière prise au hasard entre 1 et la limite supérieure définie par l'expression numérique.

PRINT RND (10) [ENTER] renvoie une valeur comprise entre 1 et 10.

Petit programme à l'usage des parieurs :

```
10 INPUT "TIRAGE ENTRE 1 ET" ; N
20 PRINT RND (N)
RUN [ENTER]
```

TIRAGE ENTRE 1 ET ? 5

```
5
DK
```

Si vous voulez jouer à pile ou face, c'est RND (2), bien sûr !

En réalité, la série générée est pseudo-aléatoire, c'est-à-dire que pour une même limite supérieure fixée par programme, deux lancements en début de séance (ou en cours, sans que RND ait fonctionné durant ces séances) donneront exactement la même série. Il y a un moyen d'éviter cet inconvénient, qui consiste à effectuer un certain nombre de tirages pour rien, nombre variable, différent à chaque exécution. Vous pouvez placer en début de programme ces quelques lignes, ou en faire un sous-programme :

```
10 INPUT "DONNEZ UN MOT DE PLUS DE 2 LETTRES" ; A$
20 UN = ASC (LEFT$ (A$,1))
30 DE = ASC (RIGHT$ (A$,1))
40 X = ABS (UN-DE)
50 FOR I = 1 TO X
60 N = RND (10)
70 NEXT I
```

Ce sous-programme emploie des commandes qui vous sont encore inconnues (sauf si vous en êtes à votre seconde lecture...); elles vous seront expliquées dans les chapitres suivants.

La fonction ABS

ABS (exp. numérique)
retourne la valeur absolue, c'est-à-dire la valeur indépendante du signe + ou - de l'expression numérique.

Elle s'obtient aussi par [CONTROL] [B].

Elle est utile chaque fois que l'on veut connaître la différence entre deux nombres sans se soucier de savoir lequel est le plus grand.

Essai :

```
10 INPUT A,B
20 PRINT "LA DIFFERENCE ENTRE LES 2 NOMBRES EST DE" ; ABS
(A - B)
RUN [ENTER]
```

```
? 18, 48
LA DIFFERENCE ENTRE LES 2 NOMBRES EST DE 30
OK
```

Voici les autres fonctions numériques disponibles dans le BASIC d'ALICE :

Fonction SQR (racine carrée) (ou **CONTROL** **;**)

*SQR (exp. numérique)
renvoie la racine carrée de la valeur de l'expression numérique qui doit être supérieure à zéro.*

PRINT SQR (100) **ENTER**

10
OK

Fonction SGN (Signe) (ou **CONTROL** **X**)

Elle indique si un nombre est positif, négatif, ou a la valeur zéro.

*SGN (exp. numérique)
renvoie 1 si la valeur numérique est positive ;
0 si elle est égale à 0 ;
- 1 si elle est négative.*

Essai :

10 INPUT A
20 PRINT, SGN (A)
RUN **ENTER**

? 12 OK	? - 5 OK	? 0 OK
------------	-------------	-----------

Les trois fonctions trigonométriques SIN, COS, TAN

*SIN (exp. numérique) renvoie le sinus d'un angle (ou **CONTROL** **N**) ;
COS (exp. numérique) renvoie le cosinus d'un angle (ou **CONTROL** **Z**) ;
TAN (exp. numérique) renvoie la tangente d'un angle (ou **CONTROL** **D**) .
L'expression numérique correspond à la valeur d'un angle mesuré en radians.*

Vous exprimez ordinairement les angles en degrés.

L'ordinateur fonde ses calculs sur des valeurs en radians.

La solution la plus simple consiste à attribuer la valeur 57.29577951 à une constante C (C pour "conversion") et à diviser toutes les valeurs d'angles par C, puisque la formule de la conversion est :

Radians = degrés / 57.29577951
Degrés = radians * 57.29577951

Si la valeur de l'angle exprimée en degrés est supérieure à 360, seule la quantité supérieure à 360 sera prise en compte. SIN (370) donne le même résultat que SIN (10) ou SIN (730). Autrement dit, les trois fonctions se calculent sur une valeur modulo de 360.

Testez cela avec ce programme d'essai :

```
10 INPUT "ANGLE EN DEGRES" ; A
20 C = 57.29577951
30 A = A/C
40 PRINT "SINUS", SIN (A)
50 PRINT "COSINUS", COS (A)
60 PRINT "TANGENTE", TAN (A)
```

ANGLE EN DEGRES ? 30

```
SINUS          .5
COSINUS        .866025404
TANGENTE       .577350268
OK
```

La fonction LOG (ou **CONTROL** **.**)

*LOG (exp. numérique)
donne le logarithme naturel en base e = 2.7182718228 de l'expression numérique qui doit être supérieure à zéro.*

Pour trouver le logarithme d'un nombre dans une base autre (B), utilisez la formule :

$$\text{Log}_B (X) = \frac{\text{Log}_e (X)}{\text{Log}_e (B)}$$

Par exemple, LOG (32768) / LOG (2)

donne le logarithme en base 2 de 32768 (c'est-à-dire la puissance à laquelle 2 est élevé pour donner 32768).

La fonction EXP

Cette fonction est l'inverse de la fonction LOG. Elle fournit l'exponentielle naturelle d'un nombre.

*EXP (exp. numérique)
renvoie l'exponentielle naturelle de la valeur que figure l'expression numérique. Cette valeur ne doit pas être supérieure à 88.*

Au-delà de 88, on dépasse la capacité de traitement de l'ordinateur qui le fait savoir par un message "OV ERROR" qui signifie "dépassement". ALICE ne sait pas traiter des valeurs supérieures à IOE+38 et on dépasse cette valeur limite lorsque la base e est élevée à la puissance 89.

EXP et LOG étant inverses l'une de l'autre, il s'ensuit que $X = \text{EXP} (\text{LOG} (X))$ et $X = \text{LOG} (\text{EXP} (X))$.

B EN CODE...

Lorsque nous avons parlé de la transmission d'un caractère frappé au clavier vers l'intérieur de l'ordinateur, nous avons évoqué ce qu'est un code.

A chaque configuration d'un octet correspond une valeur en code décimal à laquelle correspond elle-même un caractère. La table de correspondance entre les valeurs décimales et les caractères s'appelle le *CODE ASCII* (prononcez ASKI). Vous en trouverez un exemplaire en page 168 mais en voici quelques aperçus parmi les plus utiles.

65	A	(65 + 32)	97	A en inversion vidéo
66	B		98	B en inversion vidéo
⋮	⋮		⋮	⋮
90	Z		122	Z en inversion vidéo

13 : Validation et retour du curseur en début de ligne suivante.

C'est le code de la touche **ENTER**.

8 : Déplacement du curseur vers la gauche et effacement.

C'est l'équivalent de **CONTROL Q**.

32 : Déplacement du curseur vers la droite (= barre d'espacement).

Deux fonctions permettent de changer les valeurs décimales en caractères et inversement.

Prenons l'exemple du caractère "A" :

`CHR$(65)` → "A" (résultat chaîne)

`ASC("A")` → 65 (résultat numérique).

Pour vous rendre compte des possibilités de `CHR$`, tapez les lignes suivantes :

```
10 INPUT X
20 PRINT "ESSAI" + CHR$(X) + "■"
■ s'obtient par SHIFT B.
```

Aux valeurs que vous allez entrer pour X va correspondre un caractère concaténé entre le mot "ESSAI" et le pavé noir sur l'écran. Il vous sera facile de le repérer. Commencez par donner à X des valeurs du code ASCII correspondant à des mouvements du curseur et non à des caractères.

? 13	? 8	? 32
ESSAI	ESSA■	ESSAI ■
■	OK	DK
DK		

Essayez maintenant avec n'importe quelle valeur comprise entre 32 et 255, jusqu'à ce que vous réalisiez bien à quoi sert la fonction `CHR$`.

La fonction `ASC` est un peu moins souvent utilisée. Elle peut servir à vérifier la valeur décimale d'un caractère. Cherchons par exemple la valeur de **SHIFT B** :

```
PRINT ASC ("■") ENTER
128
```

A l'inverse, si vous tapez maintenant

```
PRINT CHR$(128) ENTER, vous obtenez :
```

`CHR$` (expression numérique)

renvoie le caractère correspondant à l'expression numérique dans le code ASCII. La valeur de l'expression est comprise entre 0 et 255.

`ASC` (expression chaîne)

renvoie la valeur décimale du code ASCII correspondant à l'expression chaîne.
Si l'expression chaîne contient plus d'un caractère, seul le premier est pris en compte.

Remarque : `CHR$` est la première fonction que nous rencontrons, qui renvoie un résultat de type chaîne ; son nom se termine par le signe \$, comme toutes les fonctions qui ont pour résultat une chaîne.

On peut combiner plusieurs fonctions. Exemple :

Pour transformer un caractère en sa propre inversion vidéo ou en minuscule à l'imprimante (dont le code vaut 32 de plus que celui du caractère normal en majuscules), on peut utiliser la combinaison suivante :

```
10 INPUT C$
20 PRINT CHR$(ASC(C$) + 32)
30 LPRINT CHR$(ASC(C$) + 32)
```

?G

G (en inversion à l'écran)

g (à l'imprimante)

C LES FONCTIONS DÉPENDANT D'UNE CHAÎNE

La fonction `LEN` (*LENGTH = LONGUEUR*)

Elle donne le nombre de caractères dont se compose une chaîne. Essayez :

```
10 INPUT C$
20 PRINT C$ ; "A" ; LEN(C$) ; "LETTRES"
```

`LEN` (expression chaîne)

renvoie une valeur exprimant le nombre de caractères dont se compose la chaîne.

Vous allez utiliser tout de suite cette fonction dans un **exercice guidé**.

Énoncé : écrivez les lignes de programme qui permettent d'afficher une chaîne de caractères quelconque, variable, en haut de l'écran sur la première ligne et bien au milieu.

(Nous supposons que la chaîne a moins de 32 caractères.)

Analyse : la seule partie délicate, c'est le calcul du nombre d'espaces à afficher avant et après la chaîne.

Supposons une chaîne (T\$) qui soit le mot RÉVEIL. Sa longueur est donnée par LEN (T\$) et vaut 6. L'écran a 32 caractères de large. Pour que RÉVEIL soit bien au milieu, il faut qu'il soit précédé et suivi de (32-6) / 2 espaces, ce qui, pour n'importe quel T\$, s'exprime : (32 - LEN (T\$)) / 2.

Traduction en BASIC

```
10 INPUT "TITRE"; T$
20 CLS
30 NE = (32 - LEN (T$)) / 2
40 PRINT TAB (NE) T$
RUN [ENTER]
```

TITRE ? REVEIL
Effacement d'écran

REVEIL

OK

Remarquez qu'en toute rigueur il aurait fallu prendre la partie entière de la division par 2. Si la longueur de T\$ est un nombre impair, NE comporte une partie décimale (.5).

Ce n'est pas grave; dans ce cas, PRINT TAB (6.5) arrondit automatiquement à TAB (6).

Les ciseaux

C'est à quoi l'on pourrait comparer les trois autres fonctions, les 3 sœurs. Chacune d'entre elles extrait une sous-chaîne d'une chaîne de caractères, mais chacune à sa manière.

LEFT\$ (son nom signifie GAUCHE) se sert à gauche.

RIGHT\$ (son nom signifie DROITE) se sert à droite.

MID\$ (son nom signifie MILIEU) se sert au milieu.

Voyons ce qu'elles font et comment elles découpent une chaîne.

```
5 CLS
10 INPUT A$
20 PRINT LEFT$ (A$,2)
30 PRINT RIGHT$ (A$,7)
40 PRINT MID$ (A$,4,6)
RUN [ENTER]
```

? LE JARDIN DES MERVEILLES

LE
VEILLES
JARDIN
OK

LEFT\$ (exp. chaîne, exp. numérique)

renvoie une sous-chaîne composée des (exp. numérique) premiers caractères de la chaîne.

RIGHT\$ (exp. chaîne, exp. numérique)

renvoie une sous-chaîne composée des (exp. numérique) derniers caractères de la chaîne.

MID\$ (exp. chaîne, exp. num. 1, exp. num. 2)

renvoie une sous-chaîne composée des caractères pris à la chaîne à partir du (exp. num. 1) ème sur une longueur de (exp. num. 2) caractères.

(Faut-il préciser que dans l'exemple que nous avons pris la chaîne A\$ restait intacte? La comparaison avec les ciseaux était excessive...)

Voilà : en combinant toutes ces fonctions, numériques et chaînes, vous avez déjà de quoi vous occuper.

Surtout quand vous saurez que rien ne vous empêche de transformer la valeur A (= 1983) en chaîne A\$ (= "1983"), par la baguette magique de la fonction STR\$: A\$ = STR\$ (A)

STR est l'abréviation de STRING (= CHAÎNE).

Vous pouvez alors pratiquer sur A\$ toutes les opérations possibles sur des chaînes.

Par exemple prendre les deux derniers caractères

D\$ = RIGHT\$ (A\$,2) (D\$ contient "83")

et si vous voulez à nouveau faire des calculs sur la valeur 83,

vous avez deux solutions :

X = A - 1900 (X vaut 83)

X = VAL (D\$) (Même résultat)

Un tour de passe-passe parfois utile !

STR\$ (exp. numérique)

renvoie une chaîne de caractères constituée des chiffres qui expriment la valeur de l'expression numérique.

VAL (exp. chaîne)

renvoie la valeur numérique correspondant aux chiffres de la chaîne, à condition que celle-ci soit composée de chiffres.

POUR VOUS MUSCLER

1. A qui la priorité?

Écrivez en BASIC l'expression :

$$\frac{23 + 19}{6} \times (8 + 3)^2$$

$$13 - 5$$

2. Une occasion...

Calculez la valeur d'une voiture d'occasion à partir de :

- sa cote à l'ARGUS ;
- son âge en années ;
- son kilométrage.

La cote est généralement établie pour des voitures ayant parcouru environ 15 000 km par an. Au-delà de cette moyenne, comptez une moins-value de 5 % par tranche de 15 000 km excédentaires. (On supposera que la voiture a parcouru plus de 15 000 km/an.)

3. En lettres géantes

Vos initiales. Dessinez-les au milieu de l'écran sur 5 ou 7 cases-écran de haut, en vous servant de l'instruction PRINT@et des caractères semigraphiques.

4. Vous la troublez...

Faites bégayer ALICE quand elle prononce votre prénom. Supposons que vous vous appelez FÉLIX. Après lui avoir donné votre nom, vous voyez ALICE nettoyer l'écran et afficher au milieu : BONJOUR F...FÉ...FÉLIX.

Attribuez cela à l'émotion.

5. Tentez votre chance

Vous entrez d'abord le nombre de chevaux partants dans une course. Avant d'appuyer sur **ENTER**, vous avez intérêt à prendre les paris, car ensuite tout va très vite, ALICE donne le tiercé gagnant.

Pour amuser de jeunes enfants, on peut aussi parier sur la couleur que va prendre l'écran et que l'on fait changer au hasard. (Utilisation de CLS.)



Chapitre 4

Comment sauvegarder vos programmes

Nous commençons à savoir écrire des programmes un peu longs. Voyons maintenant comment les sauvegarder. En effet, vous avez certainement à un moment ou un autre trouvé fastidieux d'être obligé de taper à nouveau un programme que vous vouliez réutiliser.

Vos programmes ne peuvent pas être conservés en mémoire vive, cela vous le savez. Mais vous pouvez utiliser une mémoire auxiliaire dont le support matériel est une simple cassette, si vous disposez d'un magnétophone à cassettes.

Pour le branchement reportez-vous page 15.

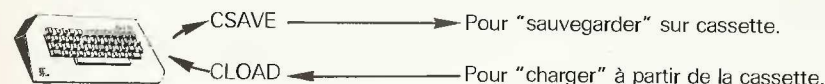
I BUT DE L'OPÉRATION

Conserver sur une bande magnétique les signaux correspondant aux positions binaires en mémoire vive lorsque le programme s'y trouve.

En sens inverse, recréer en mémoire vive les positions binaires correspondant au programme à partir des signaux conservés sur la bande magnétique.



Les deux opérations se commandent à partir de votre ordinateur.



II SAUVEGARDE D'UN PROGRAMME : CSAVE

L'ordinateur et le magnétophone doivent se trouver dans l'état décrit ci-dessous :

ORDINATEUR

Sous tension ainsi que le téléviseur.
Un programme en mémoire.

MAGNÉTOPHONE

Connecté.
Volume d'enregistrement réglé. Équipé d'une cassette.
La tête d'écriture est positionnée au début d'une plage non enregistrée du ruban magnétique (après la bande amorcée).

Vous choisissez un nom de programme et le tapez entre guillemets après CSAVE. L'enregistrement de ce nom précédera celui de votre programme sur la cassette, et c'est pratiquement le seul moyen de retrouver votre programme sur la cassette. Aussi mieux vaut le noter et noter aussi l'emplacement approximatif sur la bande, vous gagnerez du temps lors de la recherche du programme quand vous voudrez le charger à nouveau en mémoire.

Manœuvres

1 CSAVE "Nom du programme" (un mot de 1 à 8 caractères).

2 Enfoncez **PLAY** et **RECORD** jusqu'au verrouillage.

3 **ENTER**

4 OK apparaît à l'écran, signifiant que le programme est sauvegardé.

5 Appuyez sur **STOP**.

CSAVE s'obtient aussi par **CONTROL** 3.

III RÉCUPÉRATION OU "CHARGEMENT" D'UN PROGRAMME EN MÉMOIRE : CLOAD

ORDINATEUR

Mémoire vide
(soit en début de séance, soit nettoyée par **NEW** **ENTER**).

Manœuvres

1 Tapez CLOAD "nom du programme".
ENTER

Affichage de "S" dans le coin supérieur gauche de l'écran.

Affichage de "F" et du nom du programme en haut de l'écran.
Programme chargé
→ OK à l'écran.

CLOAD s'obtient aussi par **CONTROL** 4.

IV RECHERCHE D'UN PROGRAMME : SKIPF

Les manœuvres suivantes sont à utiliser quand vous ne savez plus où se trouve votre programme sur la cassette, ou bien lorsque vous ne savez plus quels programmes se trouvent sur la cassette.

ORDINATEUR

2 Tapez SKIPF "nom du programme".

ENTER

SKIPF signifie "saute jusqu'à ce que tu aies trouvé".

Le nom des programmes rencontrés apparaît à l'écran

Message OK

Il vous reste alors à faire la manœuvre de chargement CLOAD.

Quand vous souhaitez seulement consulter le nom des programmes, il suffit de donner après SKIPF un nom de programme impossible, par exemple "ZZZZZ". Vous verrez alors défiler tous les noms de programmes jusqu'au message d'erreur final.

V ERREURS DE CHARGEMENT

Elles sont variées, et des messages peuvent apparaître en haut de l'écran. Vous les éviterez pour la plupart avec les précautions suivantes :

- utilisez des cassettes neuves et de bonne qualité ;
- effacez complètement une cassette avant de la réutiliser. N'enregistrez pas sur un ruban non effacé ;
- prenez la précaution d'avoir toujours au moins une copie supplémentaire de chaque programme auquel vous tenez.

Pour plus de détail voir pages 15 et 174.

MAGNÉTOPHONE

1. Positionnez la tête de lecture au début du ruban.

3 **PLAY**

Recherche.

Les programmes qui portent un autre nom sont "sautés", mais

Programme trouvé.

Chapitre 5

L'art des répétitions

Jusqu' alors vous avez surtout appris à affecter et manipuler des variables (chapitre 2), à effectuer des traitements sur des variables numériques ou sur des chaînes au moyen d'opérations (+, - etc.) ou de fonctions (RND, LEN, MID\$ etc.).

Et cela vous a peut-être permis de mener à bien le programme de calcul de la valeur d'une voiture d'occasion qui vous était proposé à la fin du chapitre 3. Mais chaque exécution de ce programme ne permet de calculer que le prix d'une seule voiture. Si l'on en avait plusieurs dizaines à calculer, il faudrait plusieurs dizaines de fois taper RUN [ENTER]. Nous allons voir dans ce chapitre comment programmer des répétitions d'exécution automatiques.

Ce qu'ALICE sait faire une fois, elle sait le faire des milliers de fois, et les instructions à laisser pour obtenir cette répétition ne sont pas très compliquées.

I OÙ LE CHARME RÉSIDE DANS L'ACCUMULATION

Si vous vouliez faire la somme de 4 nombres, votre premier mouvement vous porterait probablement à prévoir les opérations suivantes :

- 1 entrer N1, N2, N3, N4 ;
- 2 en faire la somme ($S = N1 + N2 + N3 + N4$) ;
- 3 afficher S.

Si vous voulez additionner 50 nombres, le principe est le même. et vous utilisez 50 identificateurs... et c'est long à écrire.

Comment faites-vous, VOUS, quand vous additionnez ?

Prenez une feuille de papier, FAITES-LE.

Posez l'opération suivante :

8
+ 3
+ 4
+ 9

et observez-vous en train d'additionner. Chaque fois que vous posez la pointe de votre crayon, quelle opération mentale faites-vous ?

Cela pourrait se décrire ainsi :

8	
+ 3	addition
11	<u>mise en mémoire du résultat</u>
+ 4	addition
15	<u>mise en mémoire du résultat</u>
+ 9	addition
24	<u>mise en mémoire du résultat</u>

La suite d'opérations

addition
mise en mémoire du résultat

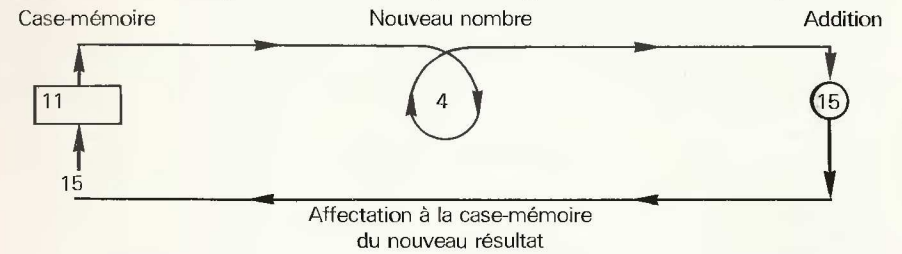
est une *séquence* d'instructions

qui se répète sans aucun changement.

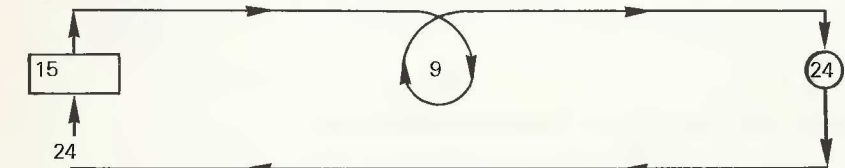
Seules changent les valeurs sur lesquelles s'effectuent les opérations.

On appelle la répétition de cette séquence une **itération**.

Détaillons cette séquence. Observons, par exemple, l'avant-dernière séquence :



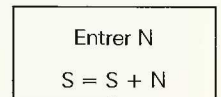
Séquence suivante :



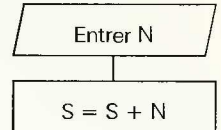
Appliqué à l'ordinateur et à notre problème énoncé au début, la séquence devient :

- Entrer le nouveau nombre.
- Affecter à une variable (S) le résultat de l'addition de l'ancien contenu de S et du nouveau nombre.

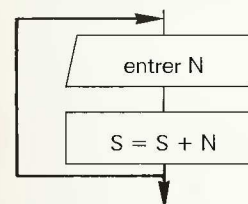
ou →



ou encore →



Ce qui permet de figurer ainsi l'itération :



On inscrit les entrées/sorties dans un parallélogramme.

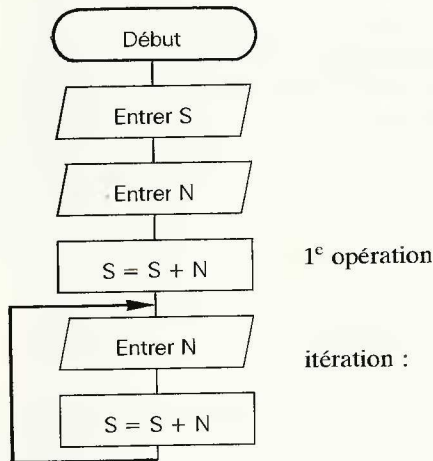
On inscrit les traitements dans un rectangle.

Ceci est un morceau d'**organigramme** (= représentation graphique de la logique des opérations telle qu'on l'établit après une analyse).

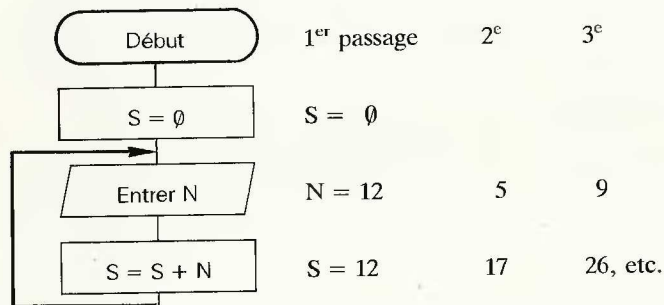
Mais notre organigramme n'a pour le moment ni queue ni tête, il représente une boucle sans fin.

Où commencer ?

Revenons à notre énoncé du début. On pourrait décrire ainsi le début des opérations :



Voilà en effet qui est mieux ; mais pourquoi pas ceci :



Notre organigramme commence bien, et ce que nous venons de faire ci-dessus s'appelle le "faire tourner à la main".

C'est ainsi que l'on peut vérifier si un organigramme est probablement correct. Notre lenteur par rapport à la vitesse de l'ordinateur nous permet de décortiquer l'évolution d'une variable et de voir où est le défaut s'il y en a un, alors que la vitesse d'exécution de l'ordinateur nous montrera le défaut éventuel en fournissant un résultat incorrect. A partir de là, notre recherche de la faute (on dit couramment un "bug") ressemblera à la remontée d'un torrent par un saumon, alors qu'il est plus facile de se laisser glisser au fil du courant en suivant un organigramme, AVANT.

Pour l'instant, notre organigramme a un défaut, nous savons lequel : il n'a pas de FIN. Un programme ainsi conçu "bouclera", c'est-à-dire itérera sans fin. D'ailleurs vous verrez, puisque vous allez l'essayer tel quel. Dans les itérations, le plus délicat est de s'arrêter juste à l'endroit voulu. Aussi, laissons cela pour le moment.

II L'ITÉRATION SANS FIN OU LE MOUVEMENT PERPÉTUEL

L'instruction qui nous permet, arrivés en fin de séquence, de retourner au début s'écrit en BASIC "GOTO" (prononcez "gotou") ce qui veut dire "aller à".

Organigramme	Traduction	Commentaires
	10 CLS 20 INPUT "DONNEZ UN NOMBRE"; N 30 S = S + N 40 PRINT TAB(20) S 50 GOTO 20	L'initialisation à 0 est automatique. Permet de suivre le déroulement. = "aller en ligne 20".

Exécution :

```
DONNEZ UN NOMBRE ? 56
                    56
DONNEZ UN NOMBRE ? 98
                    154
DONNEZ UN NOMBRE ? 34
                    188
DONNEZ UN NOMBRE ?
```

Dès que la valeur de S est affichée (exécution de la ligne 40) GOTO renvoie à la ligne 20.

GOTO (nombre en chiffres)
 (= n° de ligne)
 provoque une rupture de séquence dans l'exécution d'un programme.
 L'exécution s'interrompt et reprend immédiatement à la ligne indiquée.
 Le numéro doit renvoyer à une ligne existante du programme.
 Il peut être inférieur, supérieur ou égal au numéro de la ligne courante.

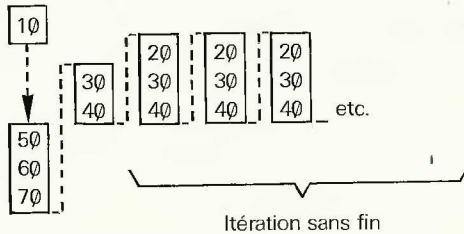
GOTO s'obtient aussi par **CONTROL J**.

Il peut y avoir plusieurs GOTO, c'est-à-dire plusieurs branchements au cours d'un même programme. Telle cette structure qui présenterait elle aussi une itération sans fin.

Exécution :

```

10 GOTO 50
20
30
40 GOTO 20
50
60
70 GOTO 30
    
```



Comment en sortir ?

Sur l'écran de votre téléviseur s'affiche toujours DONNEZ UN NOMBRE ? et le curseur clignote.

Essayez, pour lui changer les idées, de faire faire autre chose à votre ordinateur, et tapez alors la commande LIST. Voici ce que cela donne :

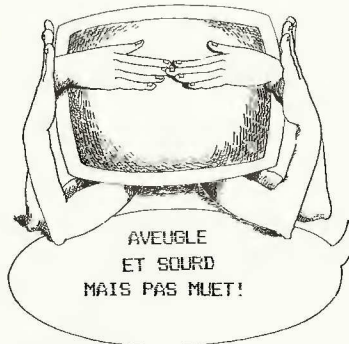
```

DONNEZ UN NOMBRE ? LIST
? REDO
DONNEZ UN NOMBRE ?
    
```

ALICE a pris la chaîne de caractères LIST sans l'interpréter, puisqu'elle attend une donnée. Malheureusement, cette donnée n'est pas du type qui convient, puisque c'est une donnée numérique qui est attendue. Alors vous êtes prié de recommencer (REDO).

Une logique aussi parfaite chez un être humain serait considérée selon la situation et le degré de bienveillance de l'observateur comme une manifestation de folie obsessionnelle ou le comble de l'humour. Imaginez ce dialogue :

- "Combien de sucres dans votre thé ?
- Là ! une vipère ! elle m'a mordu !
- Je n'ai pas entendu de nombre. Combien de sucres dans votre thé ?"



Vous ne supporteriez pas un tel être humain. Mais cette imperturbabilité caractérise votre ordinateur en train d'exécuter vos instructions. Il vous obéit, aveugle et sourd à toute autre sollicitation. La "voix de son maître" pour l'amener à s'interrompre, c'est la touche **BREAK**.

Si vous voulez interrompre l'itération sans fin, appuyez sur la touche **BREAK** (= "interruption"). A l'écran s'affiche :

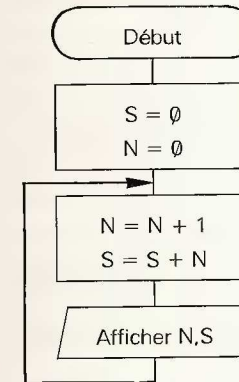
```

BREAK IN 20           = "interruption en ligne 20"
OK
    
```

Maintenant que vous connaissez la touche **BREAK**, amusons-nous à une autre itération sans fin, nettement plus vertigineuse. Dans le précédent exemple, tout se passait lentement, pondéré par l'attente de notre bon vouloir à chaque INPUT de la ligne 20.

Cette fois, au lieu d'introduire N, faisons-le varier automatiquement, ce qui sera beaucoup plus rapide à l'exécution.

Un coup d'œil au nouvel organigramme :



Changements au programme :

```

20 N = N + 1
40 PRINT N,S
    
```

RUN **ENTER**

Il est probable que vous n'avez pas attendu longtemps avant d'appuyer sur la touche **BREAK** : impossible de lire, les affichages défilent trop vite. Supposons que vous vous soyez arrêté à temps pour lire :

```

50      1275
51      1326
BREAK IN 40
OK
    
```

Les chiffres de la colonne de gauche signifient la suite des entiers positifs par ordre croissant (valeurs successives de N), et les chiffres de la colonne de droite représentent la somme des entiers positifs depuis 1 jusqu'à l'entier figurant sur la même ligne (valeurs successives de S), puisque chaque nouvelle valeur de N se cumule au total des précédentes.

Remarques à propos de $N = N + 1$

Combien de fois la séquence des lignes 20 à 40 a-t-elle été exécutée? 51 fois. Remarquons que N sert de *compteur* à l'itération en même temps que sa valeur est utilisée pour faire varier S.

Mais, au contraire de S, N est une variable dont la valeur ne dépend pas d'une autre variable. L'augmentation de sa valeur est régulière, constante. A chaque passage sur l'instruction $N = N + 1$, la valeur de N est augmentée de 1, quantité fixe. Pareille majoration s'appelle une *incrément*. On peut dire que le compteur d'une voiture s'incrémente de 1 à chaque kilomètre parcouru. L'évolution inverse s'appelle une *décrément*. (Ce serait $N = N - 1$.)

Voici un exercice que vous devriez essayer de résoudre seul(e). Cherchez au moins l'analyse et la description des opérations sous forme d'organigramme. C'est le plus difficile. Le traduire et l'écrire ensuite sont amusants, surtout à l'exécution.

Exercice guidé

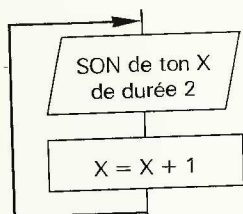
Énoncé : vous rappelez-vous l'instruction SOUND qui permet de générer un son en en fixant le ton et la durée?

Elle s'écrit SOUND nombre de 1 à 255 nombre de 1 à 255
 ton durée

Nous allons explorer toute l'échelle des tons, du plus grave (1) au plus aigu (255). ALICE s'arrêtera sur une erreur d'exécution quand on dépassera 255! La durée ne variera pas, fixons-la tout de suite à 2.

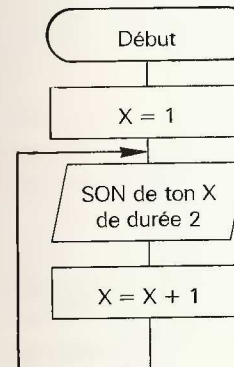
Analyse

1. Quelle est l'instruction qui constitue l'essentiel de la séquence? C'est celle qui génère un SON de ton variable (X) et de durée 2.
2. Supposons une note venant d'être émise. Que faut-il faire ensuite? Incrémenter X. Puisque nous explorons tous les sons, incrémentons-le de 1.
3. Ensuite? Émettre la note suivante avec la nouvelle valeur de X, c'est-à-dire revenir au point 1.
On peut déjà décrire le cœur de l'organigramme, on ajoutera le début ensuite.



On ne peut pas laisser l'ordinateur initialiser X à 0 automatiquement, car on aurait tout de suite une erreur d'exécution. En effet, la valeur la plus basse possible pour le ton est 1. Initialisons X à 1.

Organigramme



Exécutez!

Trouvez-vous que l'écart entre les tons est trop faible et préféreriez-vous mieux les distinguer? C'est que le pas d'incrément, 1, est insuffisant. Changez-le :

```
30 X = X + 3  
RUN [ENTER]
```

On dirait que le décollage est plus rapide! Changez encore une fois le pas d'incrément :

```
30 X = X + 10  
RUN [ENTER]
```

La durée d'exécution est de plus en plus brève, bien sûr, car dans ce dernier cas on n'itère plus 255 fois mais seulement 255/10 fois. Si l'on compare à une échelle les tons de 1 à 255, nous pouvons dire que la longueur de l'échelle est restée la même, mais que nous avons escaladé les barreaux d'abord 1 par 1, puis 3 par 3, puis 10 par 10.

Il est tout de même dommage de compter sur une erreur d'exécution pour mettre fin à l'itération. Cela reviendrait à compter sur la collision avec un autre skieur pour s'arrêter sur une pente! Ici, c'est moins dangereux, mais ça paraît tout aussi maladroit.

III PAS PLUS, PAS MOINS

Où l'art des arrêts précis après un nombre déterminé de passages dans une séquence d'instructions.

C'est possible chaque fois que l'on sait d'avance combien de passages on veut. Prenons un exemple on ne peut plus simple. On veut afficher 3 fois "Pas si vite". La "séquence" à itérer consiste seulement à afficher "Pas si vite".

Pour contrôler l'itération, il nous faudrait un compteur qui s'incrémenterait de 1 à chaque passage. L'itération cesserait dès que le compteur aurait dépassé le nombre prévu.

Une seule instruction en plusieurs mots fait tout cela.

Traduction en BASIC

```
faites NEW [ENTER]
```

```
10 X = 1
```

```
20 SOUND X,2
```

```
30 X = X + 1
```

```
40 GOTO 20
```

```

10 CLS
20 FOR C = 1 TO 3
30 PRINT "PAS SI VITE"
40 NEXT C

```

```

PAS SI VITE
PAS SI VITE
PAS SI VITE
OK

```

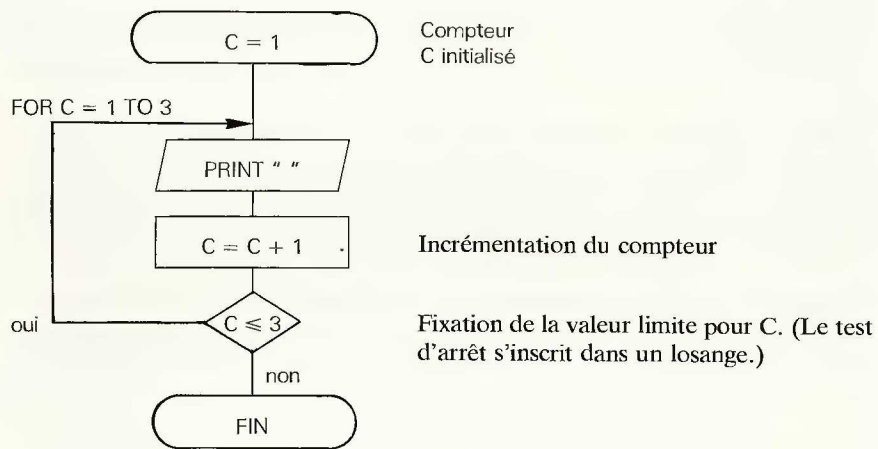
Où est le *compteur*? C'est C, bien sûr.

TO signifie jusqu'à.

FOR C = 1 TO 3 signifie "pour C variant de 1 jusqu'à 3". C'est la ligne où est défini le *test d'arrêt* (quand C > 3) et où C est initialisé à 1.

NEXT C marque la fin de la séquence contrôlée par le compteur C. Tout se passe comme s'il y avait sur cette ligne 40 une instruction implicite (non écrite, mais faisant partie de NEXT C) : C = C + 1, puis : la valeur de "C" est-elle inférieure ou égale à 3? (C ≤ 3). Si oui, la séquence comprise entre FOR... et NEXT est exécutée une fois de plus; si non, c'est la ligne suivant NEXT (s'il y en a une) qui est exécutée.

Tout se passe comme si...



Test :

Si C ≤ 3 alors aller au début de la séquence, sinon passer à l'instruction suivante. Vérifier que le dépassement du test d'arrêt entraîne bien l'exécution de l'instruction qui suit NEXT en ajoutant :

```
50 PRINT "JE REFUSE LE"; C; "EME PASSAGE"
```

ce qui vous permet de vérifier par la même occasion que C a dépassé la valeur du test d'arrêt.

RUN **ENTER** et vous voyez :

```

PAS SI VITE
PAS SI VITE
PAS SI VITE
JE REFUSE LE 4EME PASSAGE
OK

```

N.B. : Depuis longtemps l'usage s'est établi d'appeler I, J, K, L, les variables de contrôle (ici le compteur C). Systématiquement on utilise d'abord I, bien qu'il n'y ait aucune raison valable en BASIC et que cela provoque même des confusions entre I, J, et L.

Souvent, la variable de contrôle est utilisée comme compteur; dans ce cas, elle s'incrémente implicitement de 1.

Mais on peut faire varier le pas d'incrémentation (surtout quand on utilise la valeur de la variable).

Supposons que nous voulions afficher tous les entiers pairs depuis 0 jusqu'à 100.

Entrez les lignes :

```

10 CLS
20 FOR I = 0 TO 100 STEP 2
30 PRINT I;
40 NEXT I

```

Exécutez. Vous allez en effet obtenir la liste des nombres 0 2 4 6 8 10, etc.

Ajouter maintenant après NEXT I

```
50 PRINT "I = "; I et relancez l'exécution; vous constatez que I vaut maintenant 102.
```

A la sortie normale d'une itération, la variable de contrôle vaut le test d'arrêt + le pas.

Le pas peut avoir une valeur négative.

Si vous voulez produire des sons de ton décroissant, amusez-vous à ceci :

```

10 FOR I = 200 TO 1 STEP - 5
20 SOUND I, 2
30 NEXT I

```

Exécution : ? ! on regrette seulement de ne pas savoir accélérer la chute à la fin. Mais on le fera dans le prochain chapitre !

Remarques :

1. Des itérations peuvent être imbriquées comme des poupées russes. Dans ce cas, chaque itération est contrôlée par une variable différente.

Exemple de structure :

```

FOR I = 1 TO 5
  FOR J = 1 TO 10
    NEXT J
  NEXT I

```

Ici le nom de la variable est obligatoire après NEXT. Qu'est-ce que cela donnerait si on inversait NEXT I et NEXT J

```

FOR I = 1 TO ...
FOR J = 1 TO ...
NEXT I
NEXT J

```

Interdit !

Vous pouvez entrer et exécuter le petit programme suivant, uniquement destiné à vous montrer comment s'effectuent les itérations imbriquées :

```

10 FOR I = 1 TO 5
20 PRINT "I = "; I, "J = "
30 FOR J = 1 TO 5
40 PRINT J;
50 NEXT J
60 PRINT "....."
70 NEXT I

```

2. On peut aussi écrire :

```
60 FOR K = 1 TO 500 : NEXT K
```

On donne à ALICE l'ordre de faire 500 fois... rien. Mais comme cela lui prend tout de même un certain temps, on se sert couramment de cette astuce pour créer une "temporisation", c'est-à-dire une attente (pour faire défiler plus lentement un affichage, par exemple).

Vous pouvez essayer d'introduire une temporisation dans le programme des sons, page 82, en ajoutant une ligne :

```
25 FOR J = 1 TO 200 : NEXT J
```

vous constaterez alors que les sons émis sont séparés par un silence dont vous réglez la durée en faisant varier la valeur d'arrêt en ligne 25.

RAPPELONS LA SYNTAXE DE FOR... NEXT

FOR (id.) =	(e.n. 1)	TO (e.n. 2)	STEP (e.n.)
Variable	Valeur de départ	Valeur limite de	"Pas"
numérique "de	de la variable	la variable	d'incrémenta-
contrôle"			tion.
			Si STEP
			(e.n.) n'est pas
			indiqué, le pas est
			+ 1.

NEXT
(identificateur : variable de contrôle facultative si aucune confusion n'est possible).

FOR
| Délimitent une séquence à itérer et fixent le nombre de ses répétitions au moyen de la variable de contrôle.
NEXT

Pour taper plus vite :

FOR = **CONTROL** **U**

STEP = **CONTROL** **O**

NEXT = **CONTROL** **I**.

Rares sont les programmes où l'on n'utilise aucune itération. Aussi exercez-vous.

POUR VOUS MUSCLER

1. On tire un trait

Utilisez la structure d'itération pour tirer un trait vertical au milieu de l'écran avec un caractère semi-graphique.

Puis tirez un trait en diagonale depuis le coin supérieur à gauche de l'écran.

Puis tirez un trait horizontal au milieu de l'écran. (Chacun de ces 3 traits se place seul sur un écran vide.)

Variante : isolez chaque lettre d'un mot entré au clavier et affichez-les en diagonale. [Utilisez MID\$().]

2. Intéressé ?

Votre argent de poche mensuel est supposé augmenter chaque année selon un pourcentage constant.

Calculez quel en sera le montant dans 3 ans, puis dans N années.

(... Et si vous voulez en déduire les effets de l'inflation pour calculer ce que sera, en francs constants, votre pouvoir d'achat, la difficulté est plus dans la prévision du taux d'inflation que dans la programmation !)

3. Sur 5 notes ...

Faites jouer par votre ordinateur une petite musique aléatoire sur 5 notes (c'est-à-dire que la hauteur du ton et la durée sont tirées au sort chaque fois).

4. Le compte à rebours

Composez un programme qui affiche le compte à rebours de 10 à 1, arrivé à 0, il affiche FEU!! et l'on entend le bruit d'une fusée qui décolle (ou ce que vous pouvez produire d'approchant).

Chapitre 6

Et si...

Après le traitement des données et la répétition/itération, il nous faut maintenant maîtriser le branchement conditionnel.

Dans le chapitre précédent, nous avons utilisé un test dans un organigramme (page 84) : est-ce que le compteur était inférieur ou égal à la valeur limite? ($C \leq 3$) et selon que le résultat de la *comparaison* était vrai ou faux, ou bien on itérait, ou bien on passait à l'instruction suivante NEXT.

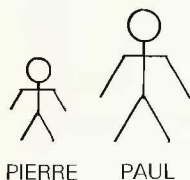
Nous n'avions pas appris à écrire ce test en BASIC, car il était implicite dans l'instruction NEXT, qui évalue toujours un compteur.

Mais nous avons très souvent besoin de comparer des données entre elles ou de tester une donnée dès qu'elle est entrée au clavier.

Les exemples sont tellement nombreux que vous vous demanderez en lisant ce chapitre comment vous avez pu programmer sans jamais jusqu'ici demander à ALICE de vérifier si...

I SI... QUOI? QUELLE EST LA CONDITION?

Une condition, au sens où nous l'utilisons, c'est l'état "vrai" ou "faux" du résultat d'une *comparaison*.



PAUL plus grand que PIERRE? VRAI
 PIERRE plus petit que PAUL? VRAI
 PIERRE et PAUL de même taille? FAUX

- Plus grand que
 - Plus petit que
 - De même taille
- } - mettent en relation la taille de PIERRE et celle de PAUL ;
 - définissent la comparaison à établir afin d'évaluer la condition VRAI ou FAUX.

Rendons les choses un peu plus abstraites, car ALICE n'est pas un robot et serait incapable d'aller réellement mesurer PIERRE ou PAUL!

Soit A la taille de PAUL : 182 cm.
 B la taille de PIERRE : 160 cm.

A > B ? VRAI
 B > A ? FAUX
 A = B ? FAUX
 B = 160 ? VRAI

Nous n'écrivons pas toutes les combinaisons, mais vous les imaginez. La relation entre A et B est établie par des *opérateurs de relation*.

Les opérateurs	signifient	s'écrivent en BASIC
>	supérieur à	>
<	inférieur à	<
=	égal à	=
≤	inférieur ou égal à	< =
≥	supérieur ou égal à	> =
≠	différent de	< >

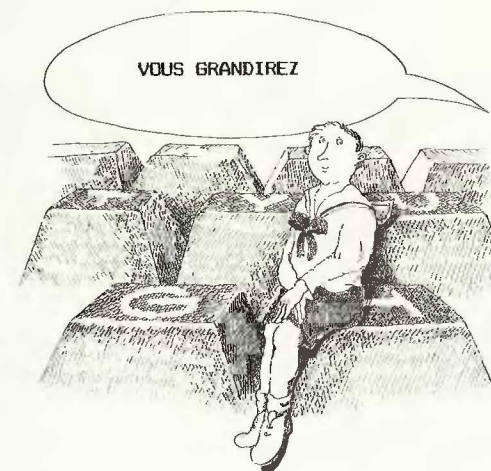
Et alors?

On peut imaginer qu'ALICE vous demande votre taille en cm. Si votre taille est inférieure à 150 cm, par exemple, *alors* elle vous répondra que vous êtes sûrement en pleine croissance.

Cela s'écrit :

```
10 CLS
20 INPUT "COMBIEN DE CM MESUREZ-VOUS"; T
30 IF T < 150 THEN PRINT "VOUS GRANDIREZ..."
40 PRINT "MOI, JE MESURE 22 CM"
```

(IF... THEN signifie SI... ALORS)



Exécution :

COMBIEN DE CM MESUREZ-VOUS ? 145
 VOUS GRANDIREZ
 MOI JE MESURE 22 CM
 OK

T < 150?
 Condition vraie

Autre exécution :

COMBIEN DE CM MESUREZ-VOUS ? 180
 MOI JE MESURE 22 CM
 OK

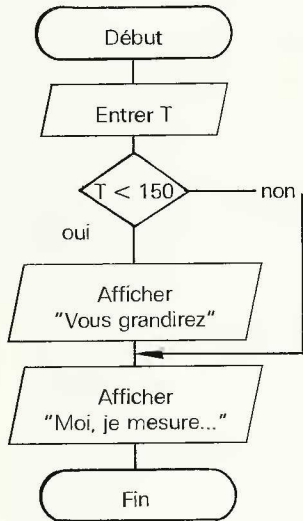
T < 150?
 Condition fausse

Remarques :

L'affichage prévu en ligne 40 s'effectue dans tous les cas.

L'instruction qui suit la condition n'est exécutée que si la condition est vraie. On dit que c'est une instruction conditionnelle.

L'organigramme de ce SI... ALORS (IF... THEN) ressemble à ceci :

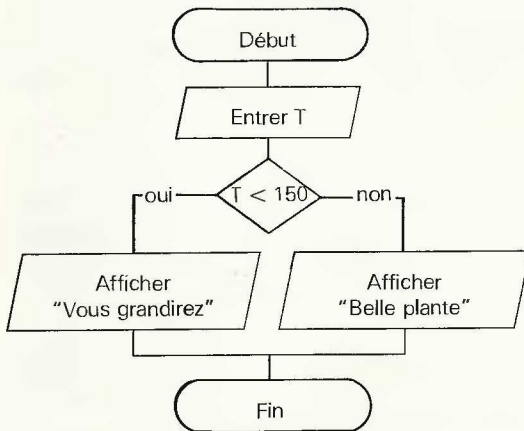


Il n'y a pas véritablement de "sinon", puisqu'il n'y a aucun traitement particulier de la condition fautive; l'affichage "Moi, je mesure..." est commun aux 2 cas.

Maintenant nous voulons qu'ALICE effectue le tri et les affichages suivants :

Si T < 150
 alors afficher "Vous grandirez..."
 sinon afficher "Quelle belle plante!"
 Fin Si (= fin des instructions conditionnelles)

L'organigramme devient :



2 instructions conditionnelles

Structure de SI... ALORS... SINON

Oui, mais... "SINON" n'existe pas dans le BASIC que comprend ALICE. Il faut se débrouiller avec des GOTO pour sauter certaines instructions.

Comme ceci :

- IF... THEN 1 Instruction(s) écrite(s) sur une même ligne à exécuter si la condition est vraie.
La ligne se termine par GOTO.
- 2 Instruction(s) (sur une ou plusieurs lignes) à exécuter si la condition est fautive.
- 3 Suite du programme.

Condition vraie : 1, puis 3 sont exécutés.

Condition fautive : 2, puis 3 sont exécutés.

Appliquons cela à notre petit programme :

Gardez les lignes 10, 20 et 40.

Tapez :

```

30 IF T < 150 THEN PRINT "VOUS GRANDIREZ" : GOTO 40
35 PRINT "QUELLE BELLE PLANTE !"
  
```

Exécution :

```

COMBIEN DE CM MESUREZ-VOUS ? 123
VOUS GRANDIREZ
MOI JE MESURE 22 CM
OK

COMBIEN DE CM MESUREZ-VOUS ? 180
QUELLE BELLE PLANTE !
MOI JE MESURE 22 CM
OK
  
```

Listez votre programme et comparez-le avec le cheminement expliqué à la page précédente.

Remarques :

1. Plusieurs instructions peuvent suivre THEN; elles sont toutes conditionnelles. En ligne 30, le GOTO 40 ne s'exécute que si T < 150. Vous n'êtes limité dans le nombre d'instructions à placer que par la dimension de la ligne logique (157 caractères).

Et si la ligne logique n'est pas assez longue pour la séquence à exécuter en cas de condition vraie? Alors on fait des "sous-programmes" (une seule instruction permet d'en exécuter tout un bloc). Nous verrons cela plus tard. En attendant, n'abusez pas des GOTO, qui rendent vite un programme illisible.

2. Il n'y a pas nécessairement de suite commune à exécuter après les instructions conditionnelles. Si nous voulons réaliser exactement l'organigramme de la page 86, il faut écrire :

```

30 IF T < 150 THEN PRINT "VOUS GRANDIREZ" : END
  
```

END signifie TERMINER : c'est évidemment la manière de couper court.

IF (condition) THEN instruction 1 (: instruction 2... facultatif)
fait exécuter les instructions conditionnelles si la condition est évaluée vraie.

END termine l'exécution d'un programme.

IF s'obtient aussi par **CONTROL** **G** et THEN par **CONTROL** **H**.
Voici un "truc" à propos de END, qui permet d'avoir plusieurs petits programmes en mémoire en même temps.

```

10      1er programme      RUN ENTER fait exécuter le 1er programme.
120    END
200    2e programme      RUN 200 ENTER fait exécuter le 2e programme.
260    END
500    3e programme      RUN 500 ENTER fait exécuter le 3e programme.
610    END

```

Exercice guidé (et récréatif)

Énoncé : Faire deviner en 5 coups maximum un nombre tiré aléatoirement par l'ordinateur (de 1 à 100). L'utilisateur est guidé quand il donne une réponse par un commentaire de l'ordinateur : « Trop petit », « Trop grand ».

Analyse

Les étapes sont :

1. Le tirage aléatoire (AL) ;
2. affichage de « Trouvez un nombre en 5 essais » ensuite... 5 coups, c'est une itération finie. Chaque passage représente un coup joué ;
3. à chaque coup joué, il faut :

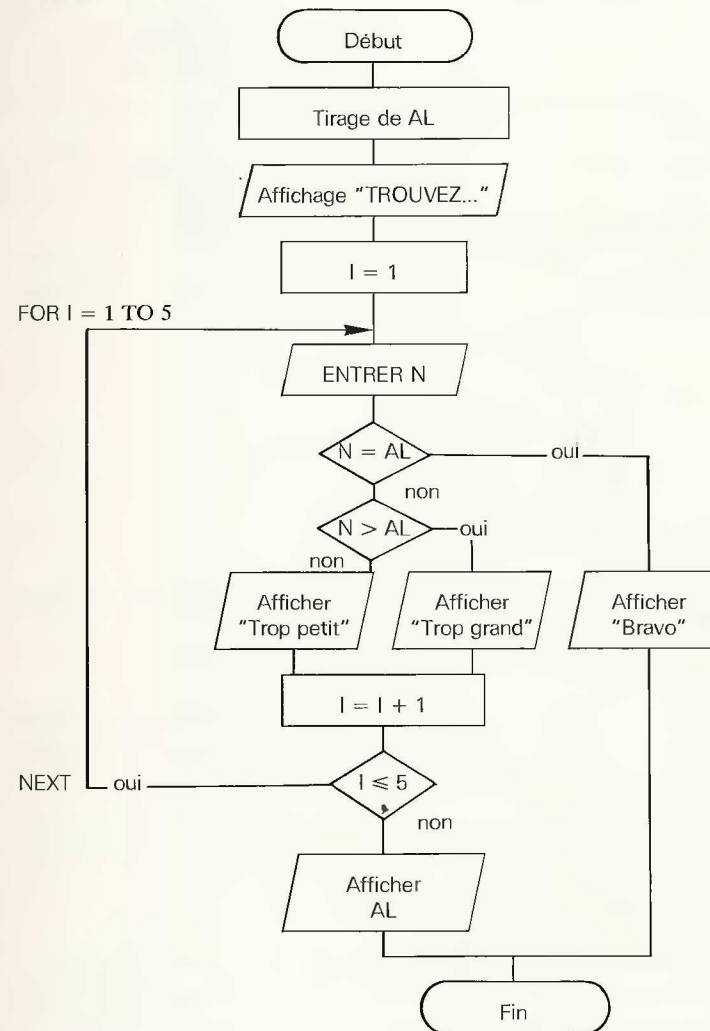
- afficher "proposez un nombre" ;
 - entrer N ;
 - Si N = AL ;
 - alors afficher « Bravo », terminer
 - sinon Si N > AL
 - alors afficher « Trop grand »
 - sinon afficher « Trop petit »
- FinSi FinSi

en face de l'accolade se trouve tout le déroulement d'une séquence dont l'itération sera contrôlée par un :

```
FOR I = 1 TO 5 ..... NEXT I
```

4. Si on sort de l'itération parce que I vaut 1 de plus que le test d'arrêt, c'est que le joueur n'a pas trouvé. Il faut donc lui afficher quel était le nombre à trouver.

Organigramme



L'organigramme commence à tenir beaucoup de place. On n'est pas obligé de faire toujours l'organigramme complet. Ici celui du "Si... alors sinon" aurait peut-être été suffisant, car c'était la seule partie délicate à écrire.

Autre remarque : l'organigramme est destiné à *montrer*, c'est une représentation graphique. Ne vous étonnez pas si vous ne faites pas un bel organigramme du 1^{er} coup, c'est normal ! Analysez d'abord les "si... alors... sinon" ; l'organigramme, par sa mise en forme, vous aidera seulement à clarifier ce que vous avez déjà conçu.

Traduction en BASIC

Elle suit pas à pas l'organigramme :

```

10 CLS
20 AL = RND (100)
30 PRINT "TROUVEZ UN NOMBRE", "COMPRIS ENTRE 1 ET 100",
   "EN 5 COUPS"
35 PRINT : PRINT : PRINT
40 FOR I = 1 TO 5
50 INPUT "VOUS PROPOSEZ" : N
60 IF N = AL THEN PRINT, "BRAVO !" : END
70 IF N > AL THEN PRINT, "TROP GRAND" : GOTO 90
80 PRINT "TROP PETIT"
90 PRINT
100 NEXT I
110 PRINT TAB (10) "C'ETAIT" ; AL
    
```

Pour tout dire sur les *opérateurs de relation* : ils peuvent aussi relier des chaînes de caractères.

Soit : A\$ = "HENRI"
 B\$ = "JOE"

Si vous écrivez : IF A\$ > B\$, le résultat est "FAUX" car "HENRI" est inférieur (vient avant) "JOE" par *ordre alphabétique*.

La comparaison se fait caractère par caractère sur le Code ASCII.

ASC ("H"), c'est 72
 ASC ("J"), c'est 74

- Dès le 1^{er} caractère, "HENRI" est inférieur à "JOE". Les autres caractères n'ont même pas à être examinés.

II POSEZ PLUSIEURS CONDITIONS

Exemple : "Si j'ai fini mon travail *et* s'il fait beau, je fais un tour à bicyclette". L'évaluation du résultat VRAI ou FAUX se fait après examen de plusieurs conditions, ici deux. Ces deux conditions doivent être vraies pour que le résultat soit vrai. Le *ET* qui les relie nous l'indique.

"Si j'ai fini mon travail *ou* s'il fait beau, je fais un tour...", les chances d'aller se promener sont accrues !

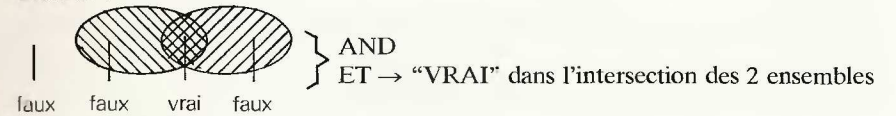
Les résultats peuvent être différents pour deux mêmes conditions selon qu'elles sont reliées par un mot ou un autre.

"ET" et "OU" sont des *opérateurs logiques*.

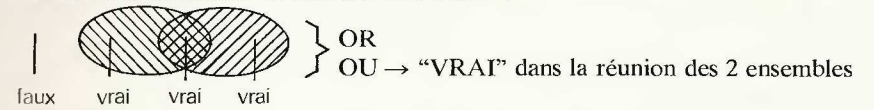
Les opérateurs logiques disponibles dans le BASIC d'ALICE sont :

- AND (= ET);
- OR (= OU inclusif = "l'un ou l'autre ou les deux");
- NOT (= NON. Exclusion).

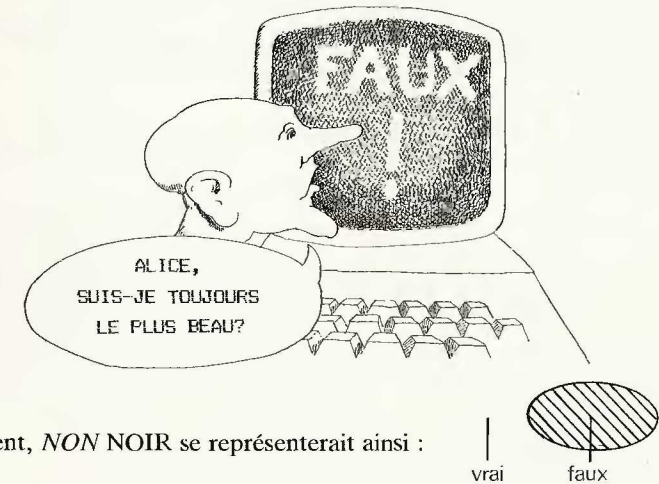
Évaluons la *vérité de la condition* "NOIR ET ROUGE" d'après la figure ci-dessous :



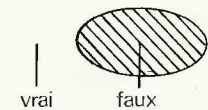
Vérité de la condition "NOIR OU ROUGE" :



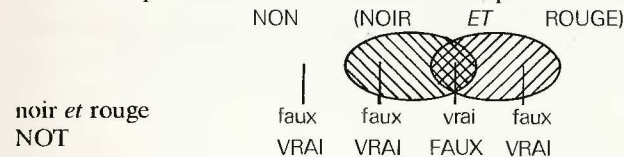
NOT signifie "absence de condition vraie". Il a l'esprit de contradiction.



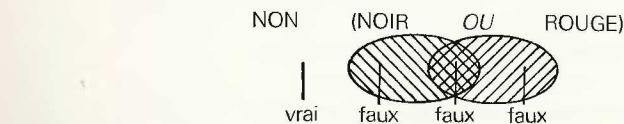
Évidemment, *NON NOIR* se représenterait ainsi :



Pour évaluer la vérité de la condition *NON (NOIR ET ROUGE)*, il faut commencer par évaluer *NOIR ET ROUGE*, puis inverser le résultat.



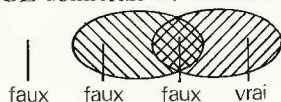
NOT... AND
 NON... ET → "VRAI" partout sauf dans l'intersection.



NOT... OR
 NON... OU → "VRAI" hors des deux ensembles (c'est "NI... NI").

Tous ces opérateurs peuvent se combiner. Les priorités sont établies au moyen des parenthèses.

(NON NOIR) ET ROUGE donnerait les résultats suivants :



Exerçons-nous tout de suite en tapant les lignes suivantes qui constituent un jeu amusant :

```

5 CLS
10 A = RND (10)
20 B = RND (10)
30 INPUT "DONNEZ 2 NOMBRES"; X, Y
35 REM LES DEUX SONT JUSTES
40 IF (X = A OR X = B) AND (Y = A OR Y = B) THEN PRINT
   "FANTASTIQUE!", "2 JUSTES" : END
45 REM 1 SEUL JUSTE
50 IF X = A OR X = B OR Y = A OR Y = B THEN PRINT "TOUCHE!",
   "1 JUSTE", "C'ETAIT" ; A ; B : END
60 PRINT "NI L'UN NI L'AUTRE" : GOTO 30

```

Exécution amusante, non ?

A remarquer : - la différence d'écriture entre les lignes 40 et 50 ;
 - l'instruction REM en ligne 35 et 45 permet d'inclure des remarques dans le programme.

REM message

Tout ce qui se trouve après REM, sur la même ligne, figure au listing mais est ignoré lors de l'exécution.

On l'utilise surtout lorsqu'on veut conserver des programmes sur cassettes. Les commentaires derrière REM facilitent la relecture du programme.

Pour que ce soit plus varié, décidons qu'en plus des nombres à trouver, un nombre est interdit. Appelons-le W. Si vous donnez le nombre interdit parmi les 2 nombres que vous proposez, vous avez perdu tout de suite.

Ajoutons :

```

25 W = RND (10)
32 REM NOMBRE INTERDIT
33 IF X = W OR Y = W THEN PRINT "A LA TRAPPE!" : END

```

Petit casse-tête : comment écrivez-vous la ligne 33 en utilisant NOT ?

Il faut :

1. écrire le contraire de $X = W$ ou $Y = W$
 c'est $X <> W$ et $Y <> W$;

2. le mettre entre parenthèses et y appliquer NOT

```

33 IF NOT (X <> W AND Y <> W) THEN PRINT "A LA TRAPPE!" : END.

```

Ça marche! (selon le principe "pourquoi faire simple quand on peut faire compliqué").

III "BRANCHÉ", PEUT-ÊTRE...

Quelquefois, la seule instruction après THEN est GOTO.

Par exemple, à la fin de l'exercice précédent, au lieu de faire terminer, dès que vous avez trouvé ou perdu, vous pourriez poser la question "On recommence?". Si la réponse est "Oui", on retourne en ligne 5. Le *branchement* sur la ligne 5 se fait à *condition* que la réponse soit "oui".

Cela s'écrit :

```

70 INPUT "ON RECOMMENCE"; R$
80 IF R$ = "OUI" THEN 5

```

GOTO est implicite.

Évidemment, il faudrait remplacer END en ligne 33, 40 et 50 par GOTO 70. Pour vous éviter de taper à nouveau, faites plus simple : il s'agit de proposer une lettre de l'alphabet, de comparer avec la "bonne" lettre (tirée aléatoirement par l'ordinateur). Cela est à *répéter jusqu'à* ce que la lettre ait été trouvée et s'écrit :

```

5 CLS
10 C$ = CHR$(RND (26) + 64)

```

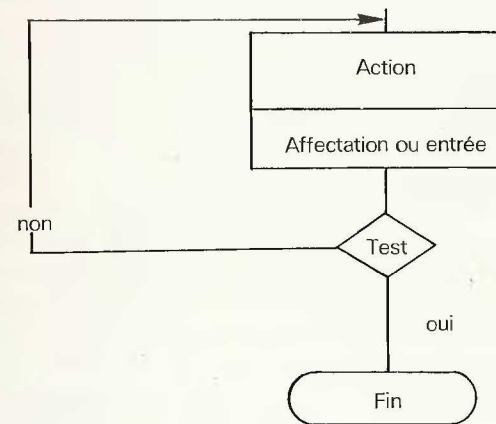
Tirage aléatoire d'un nombre entre 65 et 90, puis conversion en caractère.

```

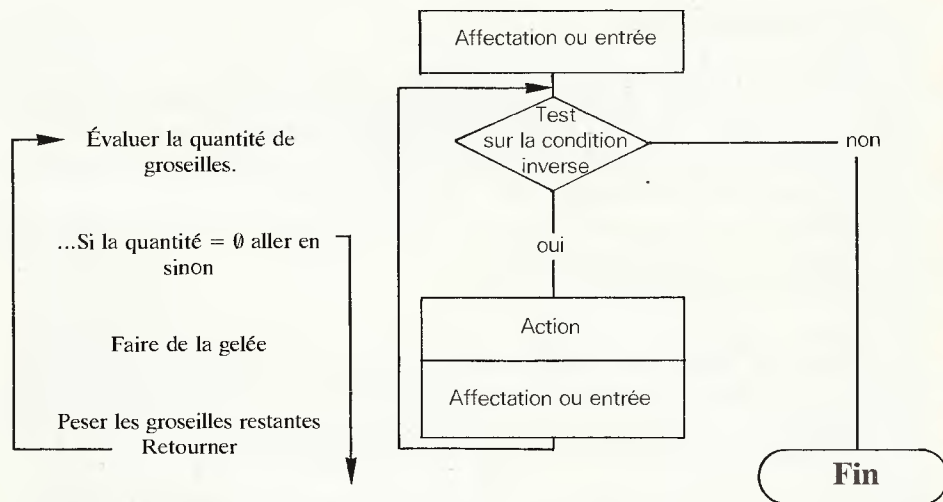
20 INPUT "LETTRE PROPOSEE"; L$
30 IF L$ <> C$ THEN PRINT, "NON" : GOTO 20
40 PRINT "BRAVO!"

```

Pour "*répéter jusqu'à*", il suffit d'appliquer l'organigramme suivant :



Une autre structure logique de branchement conditionnel s'exprime par "*Tant que*". A la différence de "*répéter jusqu'à*", où l'action a lieu au moins une fois, le test se fait ici avant la séquence d'action, de telle sorte qu'on n'y passe pas nécessairement. Par exemple "Tant qu'il y aura des groseilles, j'en ferai des gelées". Votre groseillier peut n'avoir pas un seul fruit, et "je ferai des gelées" n'avoir jamais lieu.



Structure TANT QUE

Avec ce schéma, on s'arrête quand il n'y en a plus. On ne commence même pas s'il n'y en a pas.

Voici un petit programme calqué sur ce schéma, avec un autre exemple.

```

5 CLS
10 INPUT "REPONDEZ-MOI : OUI" ; R$
20 IF R$ = "OUI" THEN 60
30 PRINT "VOULEZ-VOUS AVOIR LE DERNIER MOT ?"
40 INPUT R$
50 GOTO 20
60 PRINT "MERCI D'AVOIR REPONDU OUI!"
  
```

Tant que la réponse n'est pas "oui", poser la question de la ligne 30. Aussi faut-il tester en ligne 20 si la réponse est "oui" (condition inverse). Cette structure permet d'écrire autant d'instructions que l'on veut entre le test (IF) et le retour (GOTO).

Une variante du branchement conditionnel

Chaque fois que vous avez l'occasion d'écrire dans un programme, quelque chose comme :

```

100 INPUT C
110 IF C = 1 THEN 200
120 IF C = 2 THEN 230
130 IF C = 3 THEN 350
140 IF C = 4 THEN 60
  
```

il est plus rapide d'écrire :
 110 ON C GOTO 200, 230, 350, 60.
 On n'a alors plus besoin des lignes 120 à 140.

ON (variable) GOTO (numéro de ligne), (numéro de ligne),...
 la valeur de la variable indique à quelle place dans la file des numéros de ligne doit être pris celui auquel renverra l'instruction GOTO.

La variable doit avoir une valeur entre 1 et 255.

POUR VOUS MUSCLER

1. Saluer avec discernement

L'ordinateur demande le numéro de Sécurité sociale, ou du moins ses 3 premiers chiffres. Il affiche "Bonjour Monsieur" si le 1^{er} chiffre est 1, "Bonjour Madame" si c'est 2 (c'est le codage de la Sécurité sociale), et tout commentaire humoristique de votre choix si c'est un autre chiffre. Ensuite, il affiche l'âge de la personne. (Rappelons que l'année de naissance est figurée sur les 2^e et 3^e chiffres du n° de Sécurité sociale).

2. Un bruit de chute

En vous inspirant de l'exercice sur les sons de ton décroissant, page 85, chapitre 5, produisez une accélération vers la fin de l'itération qui évoque le bruit d'une chute accélérée.

3. La lettre interdite

Une lettre "interdite" est tirée aléatoirement et surtout pas affichée. On introduit un mot (au moins 3 lettres, sinon c'est trop facile). S'il contient la lettre interdite, le message en retour est : "Refusé". Sinon le message est "Accepté". On introduit un autre mot, etc. jusqu'à ce qu'on pense avoir trouvé la lettre interdite. On tape alors "STOP". L'ordinateur demande à quelle lettre on a pensé. Il affiche un message de félicitations si la réponse est exacte, et indique quelle était la lettre interdite si la réponse est fausse.

4. Classique : le calcul du PGCD (Plus Grand Commun Diviseur)

Le PGCD de 2 nombres est le plus grand entier qui divise entièrement ces 2 nombres. Il existe plusieurs démarches pour obtenir un PGCD. Prenez celle que vous connaissez ou choisissez-en une et mettez-la en œuvre dans un programme. (Les 2 nombres sont bien entendu entrés par l'utilisateur.)

Si vous avez trouvé cela facile, vous pouvez chercher comment transformer le programme pour lui faire calculer le PGCD de plusieurs nombres (une famille d'entiers).

Chapitre 7

Des données en file d'attente

Maintenant que nous maîtrisons bien la structure logique des programmes, examinons comment utiliser des données dans ces programmes.

Jusqu'à maintenant, nous avons vu deux manières d'attribuer une valeur à un identificateur.

Par affectation : `LET X = 1.`

En entrée à partir du clavier : `INPUT X.`

L'une et l'autre nous ont déjà permis bien des fantaisies, mais elles ne suffisent pas toujours.

Il arrive que l'on ait besoin de pouvoir disposer d'une série de données, qui sont autant de valeurs que l'on aimerait attribuer successivement à une variable.

Imaginez un morceau de programme où l'on ait besoin d'afficher tous les mois de l'année, ligne par ligne et dans l'ordre. Pas question de les entrer chaque fois, ni d'affecter 12 variables !

La solution serait la suivante :

```
10 CLS
50 DATA JANVIER, FEVRIER, MARS, AVRIL, MAI, JUIN, JUILLET
60 DATA AOUT, SEPTEMBRE, OCTOBRE, NOVEMBRE, DECEMBRE
70 FOR I = 1 TO 12
80 READ M$
90 PRINT M$
100 NEXT I
```

I EN RANG

DATA

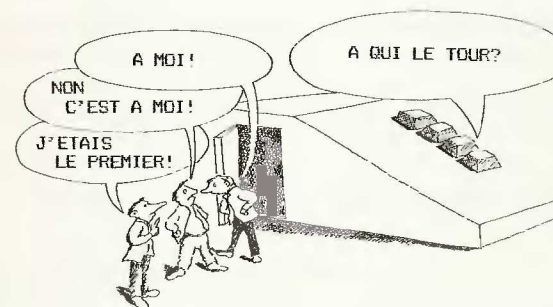
Le mot `DATA` signifie "données". C'est une instruction qui permet de stocker des données.

Les données sont rangées en file, dans l'ordre que vous leur avez attribué. Elles s'accumulent donnée après donnée, ligne après ligne sans limitation de quantité autre que celle de la place en mémoire. Les `DATA` peuvent se situer à n'importe quel endroit du programme, être séparées les unes des autres. Dès que l'ordinateur a "traduit" le programme, sa préparation à l'exécution consiste à constituer une seule file de `DATA` et à placer un pointeur en 1^{re} position. Comme ceci :

↓ pointeur
JANVIER, FEVRIER, MARS, AVRIL, MAI...

sens de déplacement du pointeur →

II A QUI LE TOUR ?



READ

Les données sont prêtes à être lues. C'est le but de l'instruction `READ` en ligne 80.

Au 1^{er} passage dans l'itération :

- `READ M$` attribue à `M$` "JANVIER"
- et le pointeur se déplace d'un cran

```
1 ↓          3 ↓
JANVIER, FEVRIER, MARS, AVRIL
2 ↓
M$ JANVIER
```

Au passage suivant, quelle valeur va prendre `M$`? Celle qui est désignée par le pointeur : "FÉVRIER"

A l'exécution, les douze passages dans l'itération affichent chacun sur une ligne le nom d'un mois de l'année, dans l'ordre que vous leur avez donné.

ATTENTION : Vous ne pouvez pas lire plus de variables qu'il n'y en a. Dans le cas ci-dessus, vous pouvez ne lire que les douze mois, aucun inconvénient. Mais si l'itération de `READ M$` se poursuivait au-delà de douze, dès la treizième vous auriez le message d'erreur "?OD ERROR" (Out of Data : "plus de données").

Une nouvelle exécution remet le pointeur sur la 1^{re} position.

Les données : elles sont exprimées directement en valeurs (chaînes ou numériques), et ne peuvent être des identificateurs de contenus.

(Si vous écrivez `DATA A$`, vous pouvez considérer que `READ M$` produirait le même effet que d'écrire `M$ = "A$"`. C'est pourquoi vous n'avez pas besoin de mettre les chaînes de caractères entre guillemets; elles ne risquent pas d'être confondues avec des noms de variables, puisqu'il est exclu qu'elles en soient.)

Il y a toutefois des exceptions : lorsqu'une donnée contient elle-même une virgule (interprétée comme séparateur de données) ou deux points (:) (interprétés comme la fin de l'instruction `DATA` et le début d'une autre) ou des blancs que vous voulez garder en début de chaîne, vous devez placer la chaîne entre guillemets.

Exemple :

```
DATA "VOICI :", " UN", "DEUX, TROIS"
```

III EXERCICE GUIDÉ

Énoncé : L'utilisateur donne son numéro de Sécurité sociale, et l'ordinateur lui dit en quel mois il est né, sous la forme : "Vous êtes né en (mois en toutes lettres)".

Analyse : 1. Comment savoir le mois de naissance? Le numéro de Sécurité sociale suivant serait celui d'un jeune homme né en mars 1957 :

1570313387103

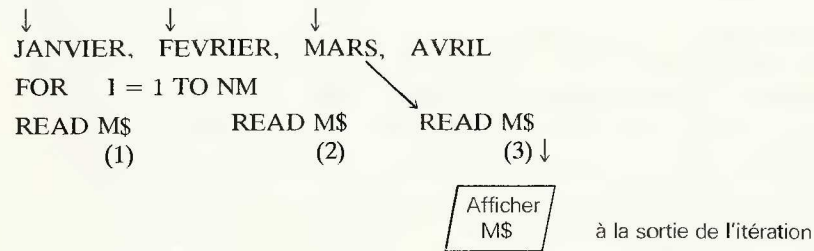
Les chiffres n^{os} 4 et 5 indiquent le n^o du mois.

Il faut :

- entrer le n^o de S.S. dans une chaîne de caractères (identificateur : S\$);
- extraire les caractères n^{os} 4 et 5 au moyen de MID\$ () pour obtenir "03" : NM\$ = MID\$(S\$,4,2);
- mais "03" pour l'ordinateur, ce sont des caractères et cela ne correspond pas du tout à un numéro d'ordre, à une valeur quelconque. Si l'on veut pouvoir utiliser cette valeur 3, il faut l'obtenir au moyen de la fonction VAL, qui transforme une chaîne de chiffres en leur valeur numérique : NM = VAL(NM\$). NM contiendra alors la valeur 3.

2. Il suffira alors de positionner le pointeur de DATA sur la position désignée par la valeur de NM (3) et, après avoir lu la donnée correspondante(MARS), de l'afficher.

Comment positionner ainsi le pointeur ?



Traduction en BASIC

Gardez les lignes 10, 50 et 60 du programme précédent.

```

70 INPUT "NO DE S.S." : S$
80 NM$ = MID$(S$,4,2)
90 NM = VAL(NM$)
100 FOR I = 1 TO NM
110 READ M$
120 NEXT I
130 PRINT "VOUS ETES NE EN" ; M$
    
```

Exécution

```

NO DE S.S. ? 26410182051
VOUS ETES NE EN OCTOBRE
OK
    
```

IV POUR BIEN UTILISER DATA ET READ

On peut juxtaposer des données numériques et des chaînes dans une même file, à condition de bien accorder à chaque type de variable un contenu qui lui convienne.

A titre d'exemple, voici le petit programme suivant :

```

10 CLS
30 READ C$, X
40 FOR I = 1 TO X
50 PRINT C$ :
60 NEXT I
70 PRINT
80 GOTO 30
100 DATA R, 12, T, 7, S, 6, X, 17, Y, 13
    
```

Lecture des données par 2
Itération qui affiche X caractères (C\$) sur la même ligne.
Passage à la ligne.
(Les DATA peuvent être placées n'importe où.)

A l'exécution, cela donne :

```

AAAAAAAAAAAA
TTTTTT
SSSSSS
XXXXXXXXXXXXXXXXXX
YYYYYYYYYYYYYY
?OD ERROR IN 30
OK
    
```

Bien sûr, puisque nous n'avons mis aucun test d'arrêt.

Pour le *test d'arrêt*, deux solutions.

On utilise FOR... NEXT. Ici on rajouterait alors, puisque nous avons 10 données lues 2 par 2, soit (10/2) passages :

```

20 FOR J = 1 TO 5
80 NEXT J
    
```

Exécutez-le : voyez, l'arrêt est normal.

(Remarquez au passage que vous utilisez des boucles imbriquées, et que nous les avons fabriquées de la façon classique : la boucle, ou itération, interne est construite la première ; ensuite on établit combien de fois il faudra avoir recours à cette itération.)

Mais c'est une solution qui ne convient pas toujours, car elle fixe le nombre de DATA. Si l'on voulait ajouter des données en DATA ?

On écrirait par exemple :

```

110 DATA B, 25, E, 3, Q, 6
    
```

mais cela obligerait à changer la ligne 20.

Le mieux, quand on veut garder souple le nombre d'éléments en DATA, est de construire ainsi son programme :

- mettre après les DATA une ligne de DATA un peu éloignée qui contienne le nombre d'éléments qui sont lus ensemble, et l'un de ces deux éléments sert de test d'arrêt dans le programme. On lui attribue une valeur dont on est sûr de n'avoir jamais besoin.

Le dernier programme deviendrait :

```
10 CLS
30 READ C$, X
35 IF X = 999 THEN END
40 FOR I = 1 TO X
50 PRINT C$;
60 NEXT I
70 PRINT
80 GOTO 30
100 DATA R, 12, T, 7, S, 6,
    X, 17, Y, 13
110 DATA B, 25, Z, 3, Q, 6
150 DATA ?, 999
```

Valeur "invraisemblable"
C'est une structure
"tant que"
(tant que X ≠ 999).

On ne se sert pas de ces dernières données, la première est là uniquement, pour que la dernière instruction READ C\$ soit possible et ne provoque pas de message d'erreur. La valeur 999 a le même rôle puis, à la ligne 35, sert de test d'arrêt. Vous avez de la place pour mettre d'autres DATA entre 110 et 150. Remarquez qu'il vaut mieux écrire plusieurs lignes courtes qu'une longue. C'est plus vite modifié.

V NOUVEL EXERCICE GUIDÉ

Exerçons-nous avec les caractères semi-graphiques, que nous avons un peu oubliés... Mettons-nous d'abord en mode 32 ou 40 (le programme est prévu pour 32 colonnes, mais vous l'adapterez facilement).

Rappel : [A] en même temps que [SHIFT] affiche [■] à l'écran en noir sur fond vert.

Si vous appuyez 1 fois sur [CONTROL] [0] en même temps, observez que le curseur clignote en noir et jaune.

Tapez à nouveau [A] [SHIFT].

Vous avez [■], mais cette fois sur fond jaune.

Appuyez encore une fois :

[CONTROL] [0], le curseur devient bleu...

[A] [SHIFT] donne alors [■] sur fond bleu.

Vous pouvez passer ainsi toute la série des couleurs. Juste après l'orange, vous revenez au "normal" : noir et vert.

On peut ainsi écrire des caractères graphiques sur le fond de son choix et les taper tels quels dans une chaîne de caractères.

Faites un essai :

Quelle que soit la couleur de votre curseur, tapez en mode direct
PRINT TAB (10) " [■] [■] [■] [■] " [ENTER]

soit [Y], [T], [R], [E] avec [SHIFT]

et vous voyez s'afficher la chaîne de caractères graphiques.

(Remarquez que les caractères ordinaires sont de toutes façons sur fond vert.)

Exercice guidé

Énoncé : Obtenir à l'écran l'affichage en noir sur fond vert d'un train dont les fenêtres sont éclairées en lumière jaune.

Résultat à obtenir et analyse :

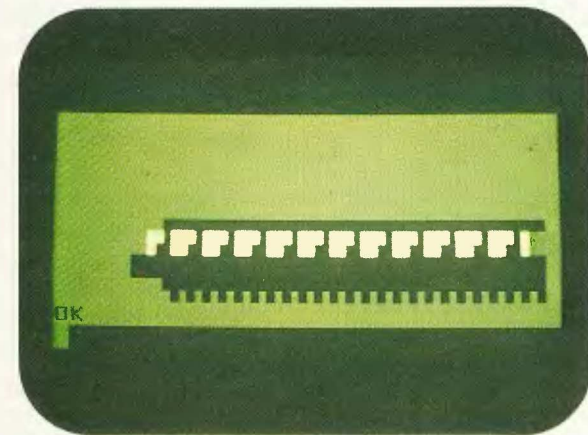


Photo N° 1.

On obtiendra :

la 1^{re} ligne horizontale au moyen de la répétition de [SHIFT] [R]
"normal" ;

La 2^e au moyen de [SHIFT] [W] [G] sur fond jaune ;

La 3^e au moyen de [SHIFT] [B] sur fond normal ;

La 4^e au moyen de [SHIFT] [C] sur fond normal.

Comment obtenir une ligne d'affichage ?

Prenons l'exemple de la dernière ligne, celle où les caractères sont les plus faciles à compter.

Supposons 2 caractères déjà affichés. L'affichage du 3^e serait seulement :

PRINT G\$;

du 4^e :

PRINT G\$; etc.

Positionner le 1^{er} devrait consister à positionner le curseur au début d'une nouvelle ligne, avec tabulation, car l'affichage ne commence pas sur la 1^{re} position.

Structure de l'itération pour une ligne d'affichage

PRINT TAB (T) ;

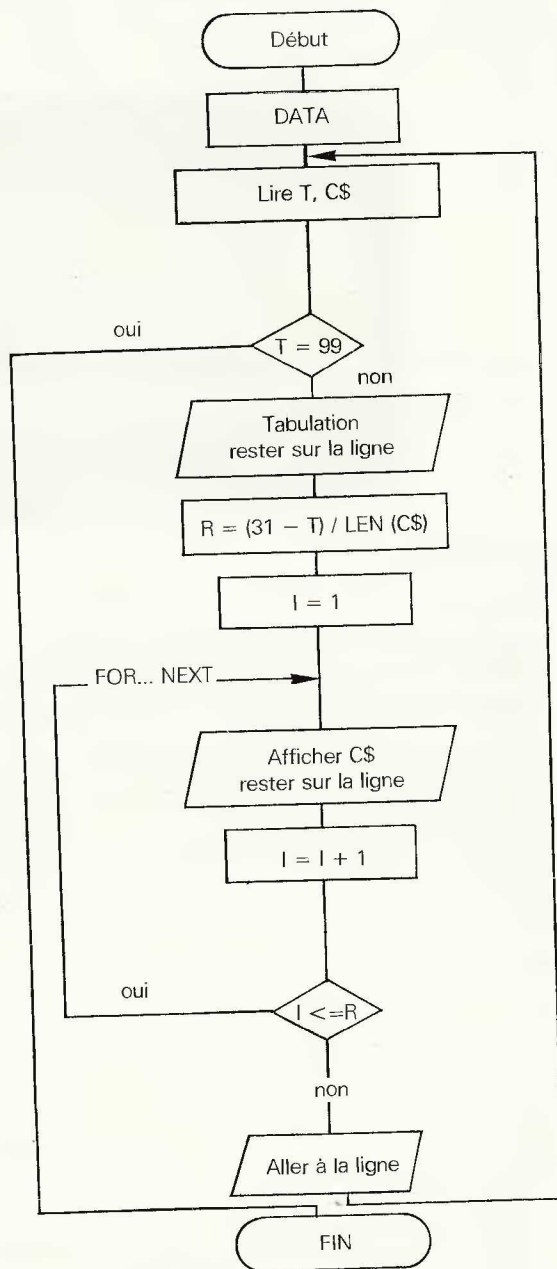
PRINT G\$;

un nombre de fois égal à (30 - T) si l'on veut que le train soit dessiné jusqu'au bord droit de l'écran.

Cas particulier de la 2^e ligne : l'élément G\$ est en fait composé de 2 caractères, la répétition devra être de (31 - T)/2 fois.

Cela nous oblige soit à calculer nous-mêmes le facteur de répétition R et à le mettre en DATA, soit à en prévoir le calcul pour chaque ligne. L'ordinateur calculera pour nous, ce qui est bien son rôle.

L'organigramme pour une ligne devrait se présenter comme ci-dessous.



99 est la donnée à mettre en DATA comme valeur d'arrêt de T.

Les données à mettre en DATA pour chaque ligne seront :

- 1) la tabulation,
- 2) le (ou les) caractère(s) à répéter.

Décrivons la manière d'obtenir les fenêtres lumineuses en ligne 2 et, par la même occasion, décrivons comment taper la ligne de DATA :

DATA 7, [SHIFT] R, 6, [CONTROL] @ [SHIFT] W [SHIFT] G puis appuyez sur [CONTROL] @ et relâchez jusqu'à ce que le curseur clignote à nouveau en vert et noir (juste après orange), et continuez à taper, 5, [SHIFT] B, 7, [SHIFT] C.

Traduction en BASIC

```

10 CLS
20 DATA 7, █, 6, █, 5, █, 7, █
30 DATA 99, A
50 PRINT : PRINT : PRINT : PRINT : PRINT
60 REM DEBUT DU DESSIN
70 READ T, C$
80 IF T = 99 THEN END
85 PRINT TAB (T);
90 R = (31 - T) / LEN (C$)
95 FOR I = 1 TO R
100 PRINT C$;
110 NEXT I
115 PRINT
120 GOTO 70
  
```

La difficulté de cet exercice résidait surtout dans l'analyse de chaque ligne. Une fois l'analyse faite, la détermination des données à mettre en DATA devenait évidente. Il en est toujours ainsi, l'instruction DATA n'est pas difficile à utiliser, et son écriture doit venir après l'analyse.

VI ON PREND LES MÊMES ET ON RECOMMENCE

Comment empiler les trains

Si nous voulions recommencer le même dessin juste au-dessous, puis encore au-dessous, etc. Si l'on remplaçait END par GOTO 70 en ligne 80, on pourrait peut-être recommencer. Essayez...

Vous obtenez tout de suite à l'exécution :

? OD ERROR IN 70

car le stock de données est épuisé. Pour pouvoir le réutiliser, il faut *replacer le pointeur au début de la file*.

C'est ce que fait l'instruction *RESTORE* qui signifie "replacer à la position d'origine".

Essayez ceci :

```
80 IF T = 99 THEN RESTORE : GOTO 70.
```

Succès total ! On n'arrête le défilé que par la touche BREAK.

RESTORE s'obtient aussi par [CONTROL] Y.

Pour mieux contrôler le déroulement, il existe un truc : la fonction *INKEY\$*, la seule dont nous n'avons pas encore parlé et qui renvoie le caractère tapé au clavier.

On l'utilise pour dire "Tant qu'on n'a pas tapé une touche du clavier", autrement dit :

"tant que INKEY\$ renvoie une chaîne vide, redemander la valeur de INKEY\$."

Ainsi, l'écriture `130 IF INKEY$ = "" THEN 130` ferait stationner l'exécution du programme, jusqu'à ce que INKEY\$ vaille n'importe quoi d'autre que... rien, c'est-à-dire jusqu'à ce qu'une touche du clavier soit tapée.

INKEY\$ s'obtient aussi par `[CONTROL][P]`.

Essayez INKEY\$ dans ce programme en tapant ceci :

```
80 IF T = 99 THEN RESTORE : GOTO 130
130 IF INKEY$ = "" THEN 130
140 GOTO 70
```

Cette temporisation et cet arrêt par la touche BREAK sont les meilleurs moyens de ne pas voir apparaître le message OK et un saut de ligne sur un dessin. Nous l'utiliserons avec tous les programmes graphiques à répétition.

Résumé

DATA valeur, valeur, valeur (= "données")
stocke des données exprimées en chaîne de caractères ou en chiffres ;
les données sont séparées par une virgule.

Si une donnée contient elle-même une virgule, deux points (:) ou commence par des espaces, elle doit être placée entre guillemets.

READ variable (ou `[CONTROL][T]`)

affecte à la variable la valeur occupant la position pointée à ce moment-là sur la file de données.

Il ne doit pas y avoir plus de variables "lues" que de données.

RESTORE ou `[CONTROL][Y]`

replaces le pointeur en 1^{re} position sur la file de données.

POUR VOUS MUSCLER

1. Faites fonctionner le programme de la page 102 pour plusieurs utilisateurs.

2. *Les reconnaissez-vous ?*

Faites afficher un renseignement permettant d'identifier facilement une personne, et l'ordinateur demande de qui il s'agit.

Selon la réponse, on comptabilise un bon point et on affiche "trouvé" ou bien on affiche "non, c'était (nom de la personne)". On recommence de même pour plusieurs personnes. A la fin, on donne le nombre de points acquis.

3. *Folk, classique ou rock ?*

Composer une petite musique en rangeant en DATA la hauteur du ton et la durée de chaque note.

Chapitre 8

La récréation en couleurs

Vous l'avez bien méritée.

Et vous avez maintenant suffisamment de connaissances pour être à même d'en profiter.

Les programmes qui permettent de manipuler la couleur et les points lumineux au moyen d'instructions graphiques nécessitent la plupart du temps un stockage des données sous forme de DATA (comme nous en avons déjà eu un aperçu au dernier chapitre), ou l'usage des itérations (chapitre 5 et 6), notamment quand il s'agit de colorer une surface ou de créer une impression de mouvement.

I GRAPHIQUE, SEMI-GRAPHIQUE, DE QUOI PARLONS-NOUS ?

Les caractères

Ils s'affichent sur les 512, 1000 ou 2000 cases-écran, selon le mode d'affichage utilisé : 32, 40 ou 80 colonnes (voir page 60). Le fond est toujours vert en mode 32 et 40 ; par contre, nous verrons page 121 comment en fixer la couleur en mode 80.

Les caractères semi-graphiques.

Ils ne peuvent être visualisés qu'en mode 32 ou 40 colonnes, soit sur 512 ou 1000 cases-écran (Vous pouvez cependant les utiliser dans n'importe quel programme, mais ils n'apparaîtront que si vous faites tourner le programme en mode 32 ou 40 colonnes.). Dans ces deux modes, on peut les afficher sur n'importe quelle case-écran. Ils sont toujours noirs, sur fond de couleur variable, du vert (code 1) à l'orange (code 8) ; le fond noir (code 0) est exclu.

Les caractères et les caractères semi-graphiques s'affichent au moyen de l'instruction PRINT et de ses variantes.

Les points graphiques.

Il faut faire une distinction entre les points graphiques des modes 32 et 40 colonnes d'une part et ceux des modes 80 colonnes de l'autre.

Pour le graphisme en mode 32 et 40 colonnes, chaque case-écran est divisée en 4 petits rectangles que l'on appelle des points graphiques. Nous disposons ainsi de 2048 points en mode 32 colonnes (soit 64 positions dans le sens de la largeur, 32 dans le sens de la hauteur) et de 4000 points en mode 40 colonnes (soit 80 positions dans le sens de la largeur, 50 dans le sens de la hauteur).

On peut allumer chaque point séparément, et lui donner n'importe quelle couleur. Le fond doit toujours être noir.

Pour le graphisme en mode 80 colonnes, chaque case-écran est divisée en 10 petits rectangles. Nous disposons ainsi de 20000 points répartis sur 160 positions dans le sens de la largeur et sur 125 positions dans le sens de la hauteur, ce qui permet d'obtenir un graphisme beaucoup plus fin que dans les autres modes.

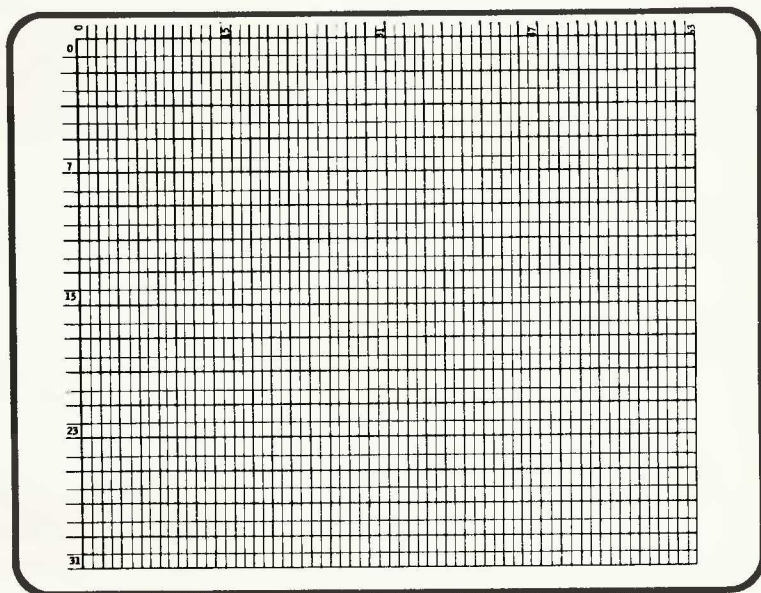
Chaque point peut être allumé séparément ou être associé aux neuf autres points appartenant à la même case-écran que lui. Vous pouvez également donner au fond la couleur de votre choix, pour créer de véritables harmonies de couleur.

Que vous soyez en mode 32, 40 ou 80 colonnes, les points graphiques ne s'allument pas par l'instruction PRINT, mais par des instructions particulières que nous allons découvrir maintenant.

II LE GRAPHISME EN MODE CLS 32 ou CLS 40

Remarque : Tous les programmes sont écrits pour le mode CLS 32. Mais vous pourrez facilement les adapter au mode CLS 40 en modifiant les valeurs de x et y.

Dessin N° 9



I ALLUMER UN POINT, CRÉER UN DESSIN : SET

SET (x,y,c) ou **CONTROL** **E**

x,y,c, sont des expressions numériques;

x : position horizontale du point, varie de 0 à 63 (ou 79);

y : position verticale du point, varie de 0 à 31 (ou 49);

c : code de la couleur, varie de 1 à 8.

Un préalable à tout allumage de point est la mise au noir de l'écran CLS 0 **ENTER**. Après cela, faites les essais suivants :

SET (40, 25, 3) **ENTER** : un petit point bleu apparaît en bas de l'écran. (Remarquez que les caractères que vous avez tapés et OK s'impriment sur fond vert et font passer toute la ligne qu'ils occupent à la couleur verte.)

Dans les dessins que vous ferez, il faudra veiller à éliminer ces parasites, notamment OK, en empêchant le programme de se terminer !

Essayez maintenant :

SET(50, 25, 4) **ENTER** : un petit point rouge apparaît sur la même ligne (on a gardé la même valeur - 25 - pour le n° de ligne), mais un peu plus à droite.

→ Si l'on veut tracer une ligne verticale, on garde la même valeur pour X et l'on fait varier Y comme ci-dessous.

```
10 CLS 0
20 X = 30
25 C = 2
30 FOR Y = 0 TO 30
40 SET (X, Y, C)
50 NEXT Y
60 GOTO 60
```

← Provoque une boucle sans fin que l'on arrête par la touche BREAK.

Et voilà une ligne jaune sur votre écran ! Voulez-vous la dessiner en pointillés ? Il suffit de remplacer la ligne 30 par :

```
30 FOR Y = 0 TO 30 STEP 2
```

Voilà ! avec STEP 3, les pointillés seraient plus espacés. Et pour tracer une diagonale ?

X et Y sont toujours égaux :

```
35 X = Y
```

et rétablissez la ligne 30 :

```
30 FOR Y = 0 TO 30
```

Exécutez. Voyez le résultat.

Si vous voulez que la diagonale aille jusqu'au coin de l'écran, X doit valoir le double de Y :

```
35 X = Y * 2
```

Voyez le résultat.

Vous pouvez même écrire :

```
35 X = Y * 0.5
```

Voyez le résultat. L'instruction SET *accepte des valeurs non entières*, elle les arrondit.

Ce qui permettra de calculer et de figurer, par exemple des sinusôides.

Il suffit d'écrire :

```
10 CLS 0
15 C = 3
20 X = 1
25 H = 10
30 FOR I = 180 TO - 720 STEP - 15
40 R = 1157.29577951
50 Y = SIN (R) * H + 15
60 SET (X, Y, C)
70 X = X + 1
80 IF X = 64 THEN 100
90 NEXT I
100 GOTO 100
```

et l'on obtient une sinusôide. En voici un exemple sous forme de biorythmes.

Photo N° 2



2 L'ANIMATION : RESET

RESET éteint un point allumé.

RESET (x,y) ou **CONTROL** **R**

x et y sont les coordonnées du point à éteindre, x est compris entre 0 et 63 (ou 79), et y entre 0 et 31 (ou 49).

Essai : CLS 0 **ENTER**

SET (40, 20, 6) allume un point bleu clair.

RESET (40, 20) l'éteint.

Comment créer du mouvement grâce à RESET ?

Un point qui s'allume, s'éteint, se rallume à côté, s'éteint, se rallume à côté, etc., donne l'illusion du mouvement.

Essayez avec le dernier programme de diagonales : après la ligne

```
40 SET (X, Y, C), ajoutez :
```

```
45 RESET (X, Y)
```

Exécutez. Si vous trouvez que cela va trop vite, vous pouvez mettre une temporisation régulière :

```
42 FOR Z = 1 TO 200 : NEXT Z
```

Vous pouvez aussi déplacer un point sur l'écran à l'aide de **CONTROL** **Q**.

```
10 CLS 0
```

```
20 H = 63
```

```
30 SET (H, 16, 7)
```

```
40 IF INKEY$ = CHR$(8) THEN 70
```

```
50 GOTO 40
```

```
70 H = H - 1
```

```
80 IF H < 0 THEN END
```

```
100 GOTO 30
```

Position horizontale.

Allume un point rose.

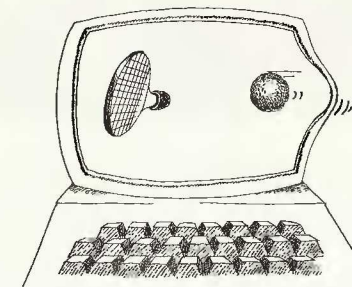
Ce programme vous permet de faire progresser une ligne en déplaçant un point vers la gauche.

Si vous voulez faire progresser le point seul, ajoutez :

```
90 RESET (H + 1, 16).
```

Essayez... C'est clair ?

3 LE "TRUC" POUR REBONDIR SUR LE BORD DE L'ÉCRAN



Jusqu'à présent, nous avons prévu un test d'arrêt pour que l'exécution cesse quand le point atteint le bord de l'écran.

Nous déplaçons X en disant que X = X + 1 (ou dans une itération FOR... NEXT). De même pour Y. Continuons à les faire avancer pas à pas de 1.

Écrivons :

```
10 CLS 0
20 Y = 1
30 X = 30
40 PY = 1
50 PX = 1
55 C = 3
60 REM FIN DES INITIALISATIONS
70 Y = Y + PY
80 X = X + PX
100 SET (X, Y, C)
120 RESET (X - PX, Y - PY)
130 IF Y < 1 OR Y > 30 THEN PY = - PY
140 IF X < 1 OR X > 62 THEN PX = - PX
150 GOTO 70
```

Point de départ en haut, au milieu de l'écran

pas d'incrément de Y
pas d'incrément de X

Vous voyez un point bleu parcourir ainsi l'écran. (C'est sur ce principe, en partant du milieu de l'écran et sans éteindre le point qu'a été conçu le programme des papillons dont vous pouvez voir le résultat ci-dessous.)

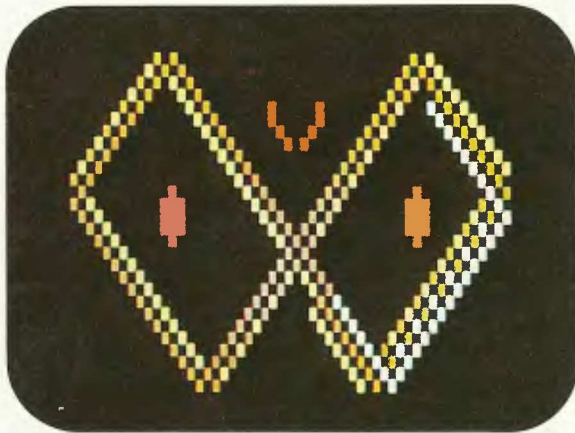


Photo N° 3

Le truc consiste à renverser le signe du pas d'incrément qui est positif jusqu'à un bord, négatif jusqu'à l'autre où il redevient positif, etc. C'est ce que font les lignes 130 et 140.

Remarques :

1. Vous pouvez rapprocher les bords de l'écran en fixant des valeurs maximales plus basses pour Y ou X.
2. Vous pouvez changer le pas d'incrément. Choisissez, par exemple $50 \text{ PX} = 1.5$, et supprimez la ligne 120, vous obtenez un dessin écossais du plus bel effet (voir page 115).

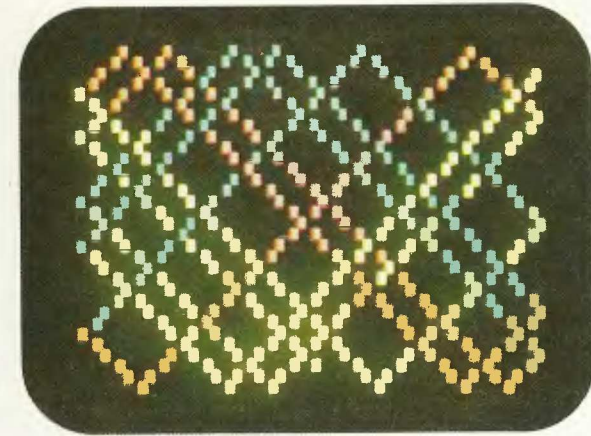


Photo N° 4

Il est aussi intéressant de faire changer la couleur à chaque "rebondissement" de X. Il suffit d'écrire :

```
55 C = 2
140 IF X < 1 OR X > 62 THEN PX = - PX : C = C + 2 : IF C > 8 THEN C = 2
```

Essayez, voyez le résultat.

Pour éviter la tache verte en haut de l'écran, il faudrait ajouter un compteur et prévoir une temporisation pour une valeur du compteur.

L'écriture des deux lignes 55 et 140 visait simplement à vous montrer les possibilités de ce petit truc.

3. Rétablissez la ligne 120 :

```
120 RESET (X - PX, Y - PY).
```

Si l'on écrivait `RESET (X, Y)`, le point s'éteindrait trop vite, on aurait à peine le temps de le voir. Il vaut mieux, lorsqu'on ne contrôle pas le déplacement par une touche au clavier, éteindre l'avant-dernier point. Ainsi il y a toujours sur l'écran un point allumé.

4 COMMENT SAVOIR SI UN PROJECTILE A "TOUCHÉ" ?

La fonction *POINT* vous renseigne sur l'état d'un point.

POINT (X,Y)

X est la position sur l'axe horizontal ;

Y est la position sur l'axe vertical ;

POINT retourne une valeur numérique

- 1 si le point est occupé par un caractère

0 s'il est éteint

1 → 8 : le code couleur s'il est allumé.

Le petit programme suivant fait allumer un point rouge au hasard sur la fenêtre supérieure gauche de l'écran, et on le voit sautiller tandis qu'un projectile part régulièrement de la gauche de l'écran.

Si le point qui est juste en avant du projectile est rouge (IF POINT (...) = 4), alors le projectile "touche" et tout l'écran devient rouge.

```

10 CLS 0
20 X = RND (10) + 10
30 Y = RND (10)
40 SET (X, Y, 4)
50 FOR I = 1 TO 30
60 SET (I, 5, 2)
70 IF POINT (I + 2, 5) = 4 THEN CLS 4 : END
75 RESET (I - 1, 5)
80 NEXT I
90 GOTO 10

```

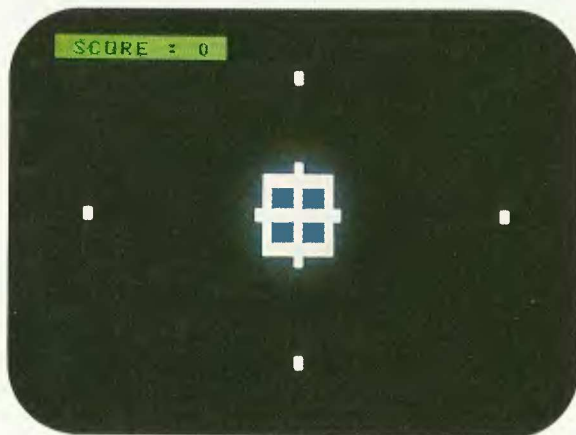
Allumage du point rouge

Test | Lancement
du
projectile

Pour finir : si vous cherchez à allumer un point sur un fond de couleur, le point occupera dans sa couleur la place de 4 points, soit la même chose que le **SHIFT** **Q** en semi-graphique.

Voici maintenant quelques exemples de graphismes en couleur qu'Alice peut réaliser pour vous.

Photo N° 5
TITANS



Pour réaliser ces dessins, voyez page 178.

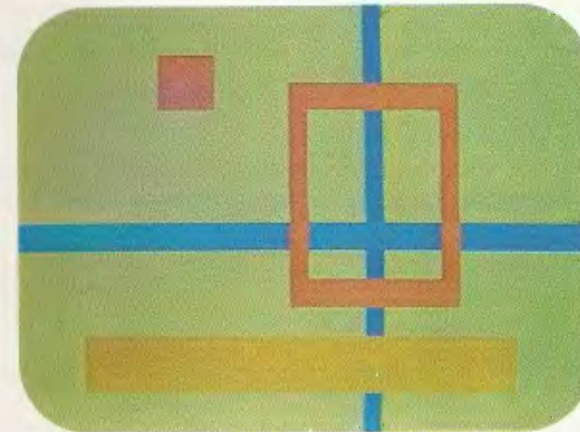


Photo N° 6
PICTOR



Photo N° 7
PENDU

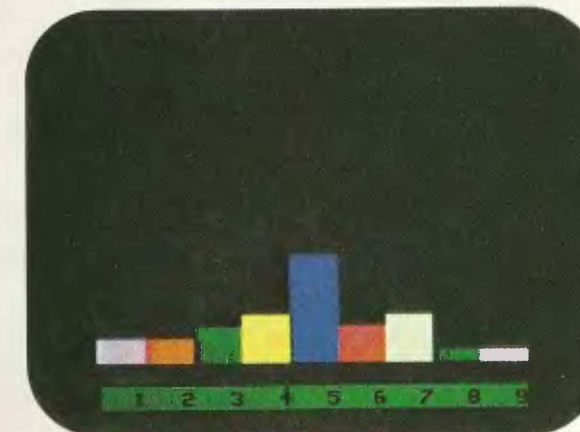


Photo N° 8
HISTOGRAMMES

POUR VOUS MUSCLER

Nous vous proposons cette fois deux exercices progressifs. Les idées ne vous manquent certes pas pour des essais assez courts, mais nous avons pensé que vous auriez peut-être besoin d'être guidés pour des réalisations plus complexes.

1) Les rectangles superposés

1. Commencez par dessiner un rectangle dont vous choisissez la longueur, la largeur, la couleur et la position du coin supérieur gauche (qui sert de point de départ au coloriage du rectangle).

2. Puis tirez au sort toutes ces données (attention à ce que la longueur plus le point de départ sur l'axe horizontal ne dépassent pas les limites de l'écran). Même précaution pour l'axe des Y.

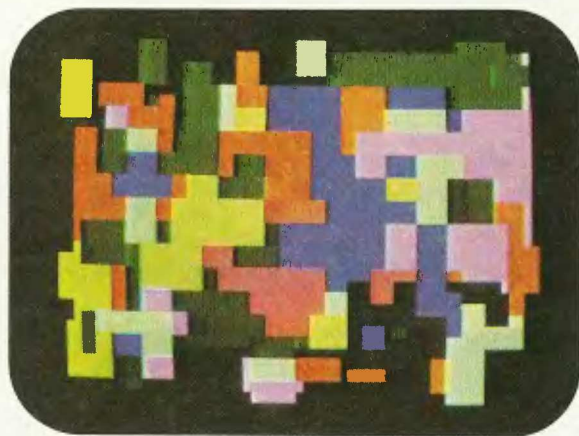
Vous aurez à chaque exécution la surprise du résultat.

3. Supprimez le message OK et la bande verte qui s'affichent à chaque fin d'exécution.

4. Faites recommencer automatiquement l'exécution et mettez un compteur, de façon à ce qu'après 8 exécutions, la taille possible des rectangles soit réduite (on peut faire de même après 15 exécutions).

Vous aurez alors le programme qui a permis de réaliser la photo ci-dessous.

Photo N° 9



2) Capturez la goutte

1. Faites partir un point lumineux du bord gauche de l'écran et avancer assez lentement sur une longueur fixée aléatoirement. Lorsqu'il arrive à la fin de sa course horizontale, le point fait alors une chute verticale rapide. Il donne l'impression d'une goutte d'eau qui tombe.

Faites recommencer cette séquence un nombre limité de fois (entre 5 et 10).

2. Positionnez au hasard en début de programme une "soucoupe", c'est-à-dire une barre horizontale de 5 ou 6 caractères de large, sur la dernière ligne de l'écran.

Si la goutte tombe sur cette soucoupe, affichez "touché" et arrêtez.

3) Dessinez

1. D'abord un dessin de votre choix, en traits de couleur sur fond noir, après l'avoir dessiné sur une grille. Repérez ses coordonnées, mettez-les en DATA. Faites exécuter le dessin.

2. Vous pouvez essayer le dessin du bateau ci-dessous.

Photo N° 10



III LE GRAPHISME EN MODE CLS 80 et CLS 81

On utilise les mêmes instructions qu'en mode CLS 32 et CLS 40 (SET, RESET, POINT). Mais attention, nous disposons maintenant de 20000 cases-écran : x varie donc de 0 à 159 pour la localisation horizontale, et y de 0 à 124 pour la localisation verticale (voir page 110).

Mais ce ne sont pas, et de loin, les seules particularités de ces deux modes que nous allons explorer ensemble.

1 ALLUMER UN POINT

1. Faites d'abord CLS 80

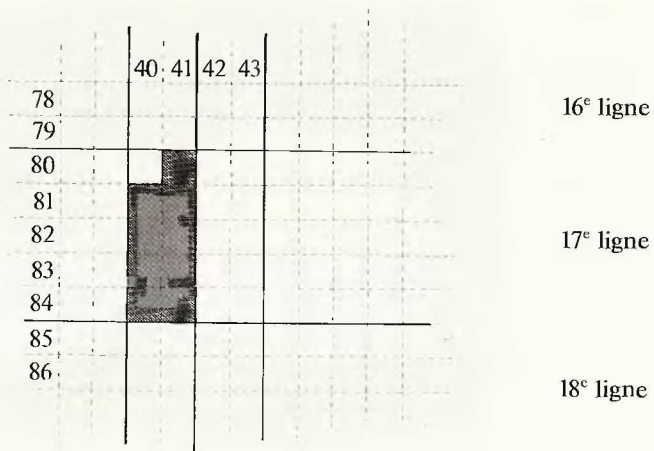
Pas besoin ici de faire CLS 0 et de mettre l'écran au noir pour faire du graphisme. Pour allumer un point, il faut faire SET, n'est-ce pas ?

Alors tapez, par exemple :

SET(40,80,2)

A la place du point jaune attendu, vous voyez apparaître un rectangle noir, dont le coin supérieur gauche est blanc ! Cela mérite quelques explications !

Cette petite tache blanche, c'est le point graphique correspondant à l'adresse 40,80 : c'est bien celui que vous avez voulu allumer. Mais rappelez-vous ce que nous disions page 110 : pour le graphisme en mode 80, chaque rectangle délimité par l'intersection d'une colonne et d'une ligne comporte 10 points graphiques, comme le montre le dessin ci-dessous.



Or, dans le graphisme en 80 colonnes, un point graphique n'est jamais totalement dissocié des autres points du rectangle dont il fait partie. Pour vous familiariser avec cette caractéristique et avec la manipulation des points à l'intérieur d'un rectangle, allumez donc les points situés en 40,82 ; 40,84 ; 41,81 ; 41,83.

Essayez ensuite sur d'autres rectangles.

On peut donc modifier à volonté l'aspect de ce rectangle et lui donner toutes sortes de formes : nous voyons tout de suite que ce mode 80 colonnes va nous donner un graphisme beaucoup plus précis et plus fin.

Changeons maintenant le code de la couleur et tapons par exemple :

SET(77,77,1)

Cette fois, notre rectangle a un "trou" ! Faisons comme tout à l'heure et modifions son aspect en "repassant" dessus pour "gommer" certains points.

Remplacez ensuite le chiffre de couleur, 1, par 2, comme précédemment. Que se passe-t-il ?

Rappelez-vous que tous les points graphiques allumés à l'intérieur d'un même rectangle, prennent la couleur du dernier point allumé.

2. Pour revenir à quelque chose de plus familier (du moins en apparence), faites donc :

CLS 81

Et recommencez les mêmes instructions : cette fois, vous allumez bien des points isolés, qui vont vous permettre de faire des dessins précis et des courbes qui mériteront pleinement leur nom...

Pour vous familiariser avec ces instructions, faites un programme qui dessine deux rectangles, l'un vert, l'autre blanc. Puis repassez en mode CLS 80. Que se passe-t-il ? Adaptez le programme en vous aidant des grilles.

RÉSUMONS :

SET(x,y,1) : en CLS 80 : affiche un rectangle auquel il manque le point graphique de coordonnées x et y ;

en CLS 81 : affiche un point graphique ;

SET(x,y,2) : en CLS 80 : affiche un rectangle dont le point situé en x,y a une autre couleur ;

en CLS 81 : affiche un point graphique.

Précisons tout de suite que ce résumé est provisoire...

2 JOUER AVEC LES COULEURS

Tâchons maintenant d'expliquer pourquoi nos points graphiques sont blancs, alors que nous les attendions jaunes, et pourquoi nous avons eu, en prime, un rectangle noir.

En mode 80 colonnes, les couleurs n'ont pas de code fixe, car vous pouvez choisir non seulement la couleur des points (et des rectangles...) que vous allumez, mais aussi celle de l'écran.

Sur votre écran, actuellement, vous avez trois couleurs : le noir, le vert, et le blanc. Faites :

SET*2,3,4

Toutes les couleurs changent : le jaune remplace le noir, le vert devient bleu, et vos petits points blancs deviennent rouges. Avec cette instruction, vous pouvez modifier instantanément les trois couleurs de votre écran.

Cela mérite quelques explications. Tout d'abord, les chiffres qui ont entraîné ces changements de couleur sont ceux que vous avez utilisés précédemment en mode 32 et 40. Mais ici, au lieu d'allumer un point, ils sont affectés à une certaine zone d'écran ou à certaines fonctions.

Observez votre écran : autour du grand rectangle bleu, il y a un espace coloré en jaune ; c'est en quelque sorte la marge de votre écran. Nous dirons donc que, actuellement, notre *couleur de marge*, c'est le jaune. Remarquez qu'il sert aussi à l'affichage des caractères, et colore également le fameux petit rectangle dans lequel s'inscrit le point que nous avons allumé.

Le rectangle bleu constitue en quelque sorte le fond sur lequel nous écrivons. *Retenez bien cette différence entre la marge et le fond*, elle est essentielle. En mode 80, notre fond a la *couleur d'intensité*.

Quant au rouge, qui n'est affecté qu'aux petits points que nous avons allumés, on l'appelle couleur de *demi-intensité*.

Dans l'instruction SET utilisée pour allumer des points, la couleur de marge a toujours la valeur 0, celle d'intensité 1, celle de demi-intensité 2.

Passez en mode 81 : retrouvez-vous la distinction entre la marge et le fond ? Quelle est la couleur de l'écran, celle de marge ou celle d'intensité ?

Or une instruction SET peut parfaitement s'intégrer à un programme : vous pouvez donc modifier toutes les couleurs, y compris celle de l'écran, au cours de l'exécution du programme. Vous pouvez ainsi faire de vrais feux d'artifice et créer vos propres harmonies de couleur.

RÉSUMONS :

SET*M,I,D permet de modifier les trois couleurs disponibles simultanément en mode 80.

M,I,D sont des chiffres variant de 0 à 7

M désigne la couleur de marge ;

I désigne la couleur d'intensité ;

D désigne la couleur de demi-intensité ;

Les valeurs 0,1,2 utilisées pour fixer la couleur des points graphiques correspondent respectivement à M, I et D.

3 ANIMER LES PROGRAMMES

Pour animer vos programmes, vous disposez bien entendu de l'instruction RESET que vous connaissez déjà. Mais attention, alors qu'en mode 32 ou 40 elle éteignait purement et simplement un point, en mode 80 elle donne à ce point la couleur de marge, c'est-à-dire que, par exemple, en mode CLS 80, le point prend la couleur du fameux petit rectangle, et non celle du fond. En mode CLS 81, son comportement est plus classique. Faites donc quelques essais, sur des points isolés d'abord, sur des dessins ensuite.

Vous retrouvez aussi l'instruction POINT(x,y).

Mais les valeurs numériques fournies par cette instruction sont différentes puisqu'elle renvoie :

- 1 si ce n'est pas du graphisme,
- 0 si le point a la couleur de marge,
- 1 s'il a la couleur d'intensité,
- 2 s'il a la couleur de demi-intensité.

Enfin, vous pouvez aussi utiliser tous les caractères de votre choix pour ajouter des petites étoiles, des commentaires en tous genres, à condition de ne pas déborder sur les dessins bien sûr ! Les caractères ayant la couleur de marge en mode CLS 80 et la couleur de fond en mode CLS 81 seront parfaitement en harmonie avec vos dessins. Pour les afficher à l'endroit voulu, employez tout simplement PRINT .

Chapitre 9

Le boomerang et les aiguillages

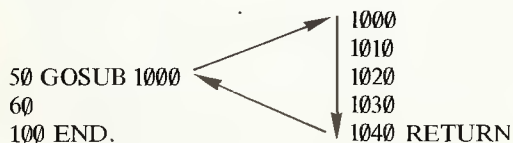
Où il est à nouveau question des branchements et de leur bon usage. Vous êtes déjà familiarisés avec l'instruction GOTO qui permet d'effectuer des branchements et nous l'avons utilisée dans des branchements conditionnels. Mais le chapitre 6 sur ce sujet contenait quelques restrictions, des invitations à ne pas en abuser. Nous allons éclaircir ce point.

Il s'agit moins, dans ce chapitre, de vous apprendre de nouvelles instructions que d'affiner votre pratique de la programmation en vous accoutumant à décomposer un problème en éléments relativement simples dont chacun peut être résolu au moyen d'un petit programme interne ou *sous-programme*.

I LE BOOMERANG OU LES SOUS-PROGRAMMES

Le boomerang est cet outil australien qui, bien lancé, va frapper le gibier et revient vers le chasseur après avoir travaillé pour lui. Nous pouvons fabriquer des boomerangs dans nos programmes.

Ce sont des sous-programmes que l'on lance (c'est le branchement sur une ligne du programme), qui exécutent un certain nombre d'instructions et reviennent ensuite à leur point de départ. Comme ceci :



GOSUB n° de ligne ou **CONTROL D**
appelle l'exécution du sous-programme commençant au n° de ligne.

RETURN ou **CONTROL F**
interrompt l'exécution du sous-programme et envoie à l'instruction qui suit immédiatement le GOSUB qui lui correspond.

L'utilisation d'un sous-programme se justifie *chaque fois que l'on a besoin d'utiliser plusieurs fois une séquence au cours du même programme*.

EXEMPLE : on veut obtenir l'enchaînement suivant.

1. Encadrer au milieu de l'écran :
"coucher de soleil";
2. afficher au bas de l'écran : "pressez sur ENTER pour continuer" et dès que l'on a appuyé, effacer l'écran ;
3. colorer l'écran en rouge. Le laisser un moment, puis :

4. encadrer au milieu de l'écran : "c'était beau...";
5. temporisation comme en 2 ;
6. encadrer au milieu de l'écran "mais c'est fini".

Les opérations 1, 4, 6 sont similaires ; c'est un *encadrement* de chaîne de caractères. Les opérations 2 et 5 sont identiques ; c'est une *temporisation*. Leur répétition à différents endroits du programme détermine notre décision d'en faire des sous-programmes.

Sous-programme de temporisation

Il faut :

1. positionner le curseur au début de la dernière ligne. Nous utiliserons l'instruction PRINT @ (voyez page 59) ;
2. attendre l'entrée d'un caractère quelconque au clavier après avoir affiché le message.
C'est un INPUT A\$ qui provoquera l'attente ; le contenu de A\$ nous est indifférent (c'est un moyen de temporisation que nous ne pouvons pas utiliser avec les graphiques, car il affiche au moins un "?" et une bande verte. Nous l'avions remplacé par INKEY\$) ;
3. effacer l'écran.

Traduction en BASIC, pour le mode 32 colonnes.

```
500 REM      TEMPORISATION
510 PRINT @ 480, "PRESSEZ <ENTER> POUR CONTINUER";
520 INPUT A$
530 CLS
540 RETURN
```

Ne l'exécutez surtout pas ainsi, ou plutôt faites-le, juste pour voir ce qui se passe.

Vous avez bien :

PRESSEZ <ENTER> POUR CONTINUER

Vous le faites,

et voici

```
? RG ERROR IN 540
OK
```

qui signifie : RETURN sans GOSUB. Bien sûr : l'appel du sous-programme par GOSUB fait mémoriser par l'ordinateur l'emplacement de l'instruction qui suit GOSUB. RETURN va chercher cet emplacement pour reprendre à partir de l'instruction qui s'y trouve l'exécution du programme principal. S'il n'a pas été *appelé* par un GOSUB, le sous-programme s'exécute mais bloque sur le RETURN, car il n'a pas d'adresse de retour. C'est pourquoi on *place toujours le sous-programme après l'instruction END* ou son équivalent.

Aussi devons-nous ajouter le petit programme d'essai :

```
10 CLS
20 GOSUB 500
30 PRINT "ÇA MARCHE"
100 END
... et c'est vrai, ça marche très bien.
```

Remarquez qu'il vaut mieux essayer tout de suite chaque sous-programme dès sa fabrication. On est sûr de ses outils et l'on progresse l'esprit tranquille.

Sous-programme d'encadrement

L'encadrement doit avoir une largeur variable selon la longueur de la chaîne à afficher.

Nous écrivons le sous-programme sans savoir exactement ce que sera la chaîne. Pour nous, ce n'est qu'un nom de variable.

Il faudra bien entendu qu'elle soit initialisée avant d'appeler le sous-programme. Cette "inconnue" (quand on écrit le sous-programme), "connue" (quand on exécute le sous-programme) s'appelle un paramètre.

Vous pouvez chercher vous-même l'analyse de l'encadrement, mais pour une fois nous vous proposons le résultat tout prêt, car le but n'est pas de s'exercer aux affichages mais de manier les GOSUB... RETURN.

Voici :

```
600 REM ENCADREMENT
610 CLS
620 PRINT @ 160, " "           Positionnement du curseur
630 L = LEN (M$)
640 IF L > 29 THEN M$ = LEFT$  Calibrage de la chaîne
    (M$, 29) : L = 29
650 T = (31 - L) / 2           Calcul de tabulation
660 E$ = " "                   Création de la chaîne
670 FOR I = 1 TO L + 2         d'encadrement qui
680 E$ = E$ + "*"              sera affichée 2 fois
690 NEXT I
700 PRINT TAB (T) ; E$         Affichage de M$ et
710 PRINT TAB (T) ; "*" ; M$ ; "*" de son cadre
720 PRINT TAB (T) ; E$
730 PRINT
740 RETURN
et le programme d'essai
10 INPUT "TITRE" : M$
20 GOSUB 600
100 END
```

Essayez :

```
TITRE? ALICE
```

ENTER

Effacement d'écran

```
*****
*ALICE*
*****
OK
```

Essayez avec des chaînes de différentes longueurs, en particulier supérieures à 29.

Quand vous êtes sûr de vos sous-programmes, à ce moment-là seulement écrivez le programme principal.

Programme principal

Il a déjà été analysé dans l'énoncé (p. 124).

```
10 M$ = "COUCHER DE SOLEIL"
20 GOSUB 600
30 GOSUB 500
40 CLS 8
50 FOR I = 1 TO 1000 : NEXT I
60 M$ = "C'ETAIT BEAU..."
70 GOSUB 600 : GOSUB 500.
90 M$ = "MAIS C'EST FINI"
100 GOSUB 600
200 END
```

Encadrement
Temporisation
L'écran reste rouge
quelques secondes
Encadrement et temporisation
Encadrement

Exécution. Voyez-vous comme les sous-programmes sont de bons serviteurs, et faciles à mettre en action ?

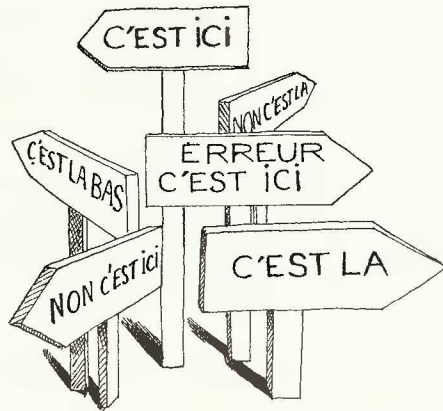
II LES AIGUILLAGES

Vous avez parfois dans un programme plusieurs séquences possibles, et vous devez vous aiguiller par un GOTO ou un GOSUB sur une séquence ou une autre.

Dans le cas où l'utilisateur a le choix entre plusieurs options, et choisit l'option 1, 2 ou 3, ou bien lorsqu'un compteur d'erreurs aiguille sur un commentaire différent à afficher selon le nombre d'erreurs déjà commises, le choix de la séquence dépend de la valeur d'une variable.

On peut dans ce cas utiliser :

ON variable num. GOTO n° ligne, n° ligne, n° ligne.
envoie l'exécution à un numéro de ligne dont la position dans la file de numéros correspond à la valeur de la variable.



Exemple à taper et utiliser pour comprendre comment cela fonctionne :
(Il est inutile d'effacer le programme précédent dont nous utilisons un sous-programme.)

```

10 M$ = "CHOIX"
20 GOSUB 600
30 PRINT "VOULEZ-VOUS ?"
40 PRINT "1 - UNE LIGNE JAUNE"
50 PRINT "2 - UN CARRE BLEU ET UNE LIGNE JAUNE"
60 PRINT "3 - LES MEMES SUR FOND ROUGE"
70 PRINT "4 - RIEN, DU TOUT"
75 INPUT "VOTRE REPONSE"; C      Affectation de C
80 CLS 0
90 ON C GOTO 170, 130,          Branchement : c'est comme si on écri-
    100, 200                    vait
100 REM ECRAN ROUGE           IF C = 1 GOTO 170
110 CLS 4                     IF C = 2 GOTO 130
130 REM CARRE BLEU           IF C = 3 GOTO 100
135 FOR Y = 5 TO 14          IF C = 4 GOTO 200
140 FOR X = 24 TO 38
145 SET (X, Y, 3)
150 NEXT X, Y
170 REM LIGNE JAUNE
180 Y = 20
190 FOR X = 1 TO 30
195 SET (X, Y, 2)
197 NEXT X
200 END

```

Selon la valeur de C, on n'exécute aucune séquence (si C = 4), ou une : seulement la ligne jaune (ligne 170 si C = 1), ou deux (ligne 130 jusqu'à la fin si C = 2), etc. (Si vous tapez un chiffre > 4, l'exécution continue sans tenir compte de l'instruction GOTO.)

De la même manière, on peut utiliser :

ON variable GOSUB n° de ligne, n° de ligne
RETURN
fonctionne comme ON... GOTO, mais le RETURN renvoie à l'instruction qui suit ON... GOSUB dans le programme.

Essayez ceci :

```

NEW [ENTER]
10 INPUT "TAPEZ 1, 2 OU 3"; C
20 ON C GOSUB 100, 50, 140
30 END
50 CLS                               Si C = 2 alors
60 PRINT "CIRCULEZ"...
70 RETURN
100 PRINT "C'EST A VOUS LA          Si C = 1 alors
    MOTO ?"
110 RETURN
140 FOR I = 1 TO 3                   Si C = 3
150 SOUND 70,8
160 SOUND 120,8
170 NEXT I
180 RETURN

```

Voilà une autre façon d'écrire des si... alors entre la ligne 20 et la ligne 30 !
On peut programmer sans savoir très bien manier cela, mais on se fatigue davantage. Tandis que si l'on pratique la programmation structurée par "pavés", on ne pense qu'à la séquence ou au sous-programme que l'on est en train d'écrire ; on ne s'occupe qu'ensuite de fabriquer le programme principal dont les outils sont tout prêts.

POUR VOUS MUSCLER

1. Un trait pour souligner

Créez un sous-programme qui vous permet d'afficher une chaîne générée par la répétition N fois d'un caractère et utilisez-le dans le programme ci-dessus.

2. Réponse en musique

Mettez en sous-programme une petite musique de 3 ou 4 notes. Utilisez-la dans le programme de la "lettre interdite" (page 99) avec chaque affichage de "Accepté".

3. Calcul à la demande

L'utilisateur fournit un nombre N compris entre 1 et 20. Il choisit l'une des trois options suivantes :

- obtenir la suite des carrés des N premiers nombres ;
- obtenir la table de multiplication de N par 9 ;
- obtenir une série de N nombres aléatoires.

Chapitre 10

Des données en libre-service

Nous avons vu avec les instructions DATA... READ comment stocker des données en file indienne et les réutiliser de même.

Mais si l'on veut accéder à la 2^e donnée après avoir lu la 4^e d'une file de 20, c'est-à-dire accéder directement à une donnée sans lire toute la file ?

Mais si l'on veut stocker des données entrées par l'utilisateur et non des données fixées par le programmeur ?

On dispose pour cela de *variables indicées*.

I UNE FAMILLE BIEN RANGÉE

Un même identificateur sert de nom commun à toute une famille de variables. Chaque variable se distingue de celles de sa "famille" par un numéro.

A (1) désigne la variable n° 1 de la famille A, de type numérique. A\$ (8) désigne la variable n° 8 de la famille A\$, de type chaîne. Toutes les variables d'une même famille sont de même type.

On appelle parfois une famille de variables un *tableau*, car on peut se représenter ces variables comme rangées en colonne dans un tableau. Exemple :

Nom	N° indice	Exemples d'instructions d'affectation	
N\$	1	LE CHAT	← N\$ (1) = "LE CHAT"
	2	LE LAPIN	← INPUT N\$ (2)
	3	LE LIEVRE	← READ N\$ (3)
	4	LE LOIR	← INPUT N\$ (4)

Utilisons les variables indicées dans un *exemple* déjà familier.

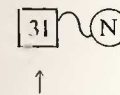
Nous voulons additionner 5 nombres. On peut écrire :

```
SOIT
10 FOR I = 1 TO 5
20 INPUT N
30 S = S + N
40 NEXT I
50 PRINT "SOMME =" ; S
```

```
SOIT
10 FOR I = 1 TO 5
20 INPUT N (I)
30 S = S + N (I)
40 NEXT I
50 PRINT "SOMME =" ; S
```

Apparemment, les deux programmes semblent faire la même chose. Mais l'état de la mémoire est différent dans les deux cas.

Cas de la variable N



Dernière valeur de N

Cas des variables indicées
N (I)

N (1)	12
N (2)	27
N (3)	8
N (4)	18
N (5)	31

(A chaque passage dans l'itération, la valeur de I a désigné une variable différente quoique de même nom.)

Et si maintenant nous avons besoin d'évaluer la différence entre le premier nombre et le dernier ? Seul le programme avec variable indicée nous permet de retrouver le premier nombre entré et d'écrire :

```
60 D = N (5) - N (1)
```

```
70 PRINT "ECART ENTRE LE PREMIER ET LE DERNIER" ; D
```

Comparaison avec DATA... READ

Quels sont les avantages par rapport au stockage de données en DATA ?

D'abord c'est l'utilisateur qui en choisit le contenu. Elles ne sont pas fixées par programme. Ensuite, elles fonctionnent en *accès direct* (en "libre-service").

Il est très facile d'afficher le 4^e nombre par :

```
80 PRINT "L'AVANT-DERNIER EST" ; N (4)
```

Par contre, les données en DATA sont très avantageuses quand on veut fixer des données à l'avance par programme. (Il est fastidieux d'écrire, par exemple, M\$ (1) = "JANVIER"... etc.)

Comment bénéficier des avantages des deux structures ?

Reprenons l'exercice guidé qui nous a servi à étudier DATA page 102 : on fait entrer un n° de Sécurité sociale, on extrait le 4^e et le 5^e chiffres dont la valeur correspond au mois.

Nous avons déjà renoncé à écrire M\$ (1) = "JANVIER", etc., cela 12 fois !

On écrit bien :

```
10 DATA JANVIER, FEVRIER..., etc., en début de programme.
```

A la fin, on voudrait pouvoir écrire directement après avoir établi la valeur du mois (NM) :

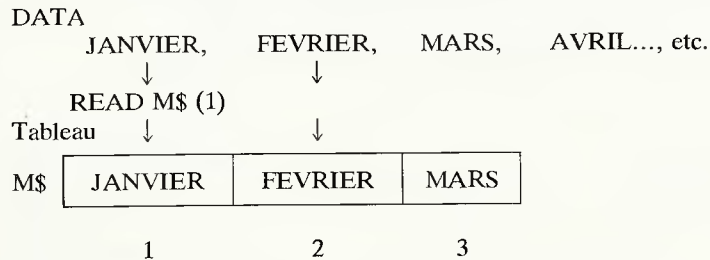
```
130 PRINT "VOUS ETES NE EN" ; M$ (NM)
```

Que faut-il entre les deux ?

ceci : M\$ (1)	JANVIER
M\$ (2)	FEVRIER
M\$ (3)	MARS

..., etc.

Autrement dit, faire venir les DATA en tableau. On peut le faire :



Il suffit d'une itération exécutée 12 fois :

```
FOR I = 1 TO 12  
READ M$(I)  
NEXT I
```

pour faire passer toutes les données dans un tableau où leur accès est direct.

Mais en réalité, pour remplir 12 variables indicées nous avons une difficulté : le nombre de variables.

II LA CONQUÊTE DE L'ESPACE VITAL

Sans autre spécification, si vous écrivez dans un programme :

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

tout va bien. Mais c'est la limite car, implicitement, une famille de variables indicées est prévue pour compter 10 "individus".

Mais nous avons 12 noms de mois à caser, dans une case pour chacun. Il suffit de demander un livret de famille à 12 pages !

DIM identificateur (expression numérique)

l'expression numérique indique le nombre maximal de variables qui appartiendront au tableau nommé par l'identificateur.

On peut écrire :

DIM ident. (e. num), ident. (e. num), ident. (e. num)..., etc., si l'on a plusieurs tableaux à déclarer.

Le programme des mois deviendrait alors :

```
10 CLS  
20 DATA JANVIER..., etc.  
30 DATA JUILLET..., etc.  
40 DIM M$(12)  
50 FOR I = 1 TO 12  
55 READ M$(I)
```

```
60 NEXT I  
70 INPUT "N° DE SECURITE SOCIALE" ; S$  
80 NM$ = MID$(S$, 4, 2)  
90 NM = VAL(NM$)  
100 PRINT "VOUS ETES NE EN" ; M$(NM)
```

Ce qui, pour le moment n'offre pas d'avantage évident sur le précédent, sauf si vous voulez vous amuser à écrire :

```
110 IF NM > 9 THEN NM = NM - 12  
120 PRINT "VOUS AVEZ ETE PROGRAMME EN" ; M$(NM + 3)
```

Remarquez au passage les avantages de l'accès direct et la possibilité d'une expression calculée pour l'indice de la variable (M\$(NM + 3)).

Précisons que vous n'êtes pas limité dans le nombre de variables indicées. La seule limite est celle de la *place en mémoire*. Que vous pouvez tester par :

MEM

retourne la quantité d'octets disponibles en mémoire.

Utilisez MEM en mode direct :

```
PRINT MEM [ENTER]  
31814
```

ou dans un programme

```
230 IF MEM < 50 THEN 300, par exemple, pour éviter de continuer à remplir un tableau quand vous n'avez plus de place.
```

Puisque nous évoquons les questions d'espace, profitons-en pour apporter quelques précisions sur l'espace réservé pour les chaînes (que l'on peut considérer comme des tableaux de caractères).

Implicitement, une chaîne de caractères peut avoir :

- 127 caractères maximum si elle est entrée par l'instruction INPUT ;
- 100 si elle est fixée par programme (instruction LET =).

On ne peut pas changer la taille maximale d'une chaîne entrée par INPUT ; elle ne dépassera jamais 127 caractères. Mais on peut réserver de la place pour un maximum de 255 caractères grâce à l'instruction CLEAR dans les autres cas.

CLEAR 300 réserve de la place pour 150 caractères (sauf dans un INPUT).

CLEAR 100 limite la place de toutes les chaînes à 50 caractères (y compris les chaînes entrées par INPUT).

CLEAR 510 réserve de la place jusqu'à 255 caractères.

Au-delà de 510, les valeurs sont acceptées, mais la limite pour les chaînes est toujours de 255.

CLEAR 1000 aura le même effet que CLEAR 510.

CLEAR s'obtient aussi par [CONTROL] [7].

III EXERCICE GUIDÉ : RECHERCHE DU MAXIMUM

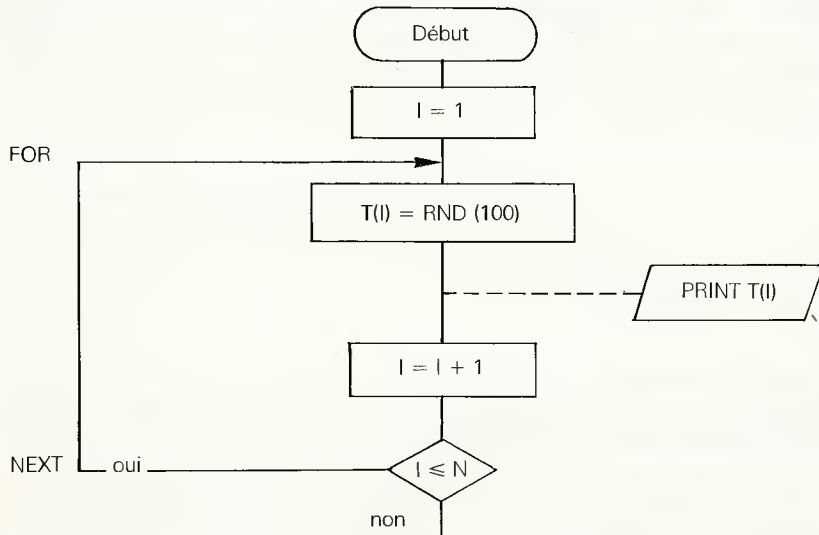
Énoncé : Rechercher le maximum dans un tableau numérique à N éléments, constitué par tirage aléatoire.

Analyse. Il faut :

1. constituer le tableau et l'afficher ;
2. rechercher le maximum ;
3. l'afficher.

1. CRÉATION ET AFFICHAGE DU TABLEAU

1. Déclaration du tableau N = 14 : DIM T (N) ;
2. affectation de chaque variable indiquée dans une itération ;



3. affichage du tableau dans une itération de même type qu'en 2. Plutôt que de le recommencer, on pourrait simplement ajouter PRINT T(I) dans la séquence.

2. RECHERCHE DU MAXIMUM

Supposons le tableau constitué des nombres suivants et supposons-nous en cours de recherche :

50 32 14 66 45 34 76 30 11 83 84 12 60 62

I = 5
T(I) = 45

↑
Quel est le maximum jusqu'alors ? C'est 66. Puisque 45 n'est pas plus grand que 66, alors nous passons au suivant. C'est-à-dire que le maximum reste l'ancien maximum et que I = I + 1.

I = 6
T(I) = 34

Même démarche. Rien ne change, sauf I qui s'incrémente de 1.

I = 7
T(I) = 76.

Mais 76 est supérieur à 66 !

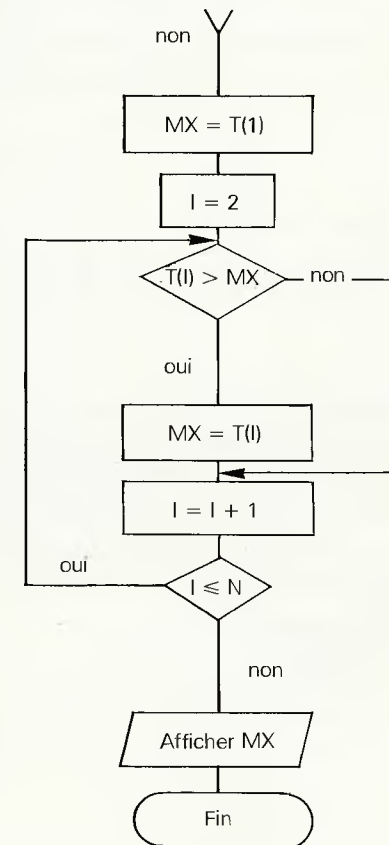
Il prend sa place en tant que maximum, ce qui pourrait s'écrire :

Si T(I) > MX
... alors MX = T(I)
FinSi

Quand I vaudra 11 et T(I) 84, MX prendra la valeur 84. La recherche se poursuivra jusqu'à I = N compris, mais la condition T(I) > MX ne sera plus jamais vraie. En fin de parcours, MX contiendra bien le maximum : 84.

Et le premier élément ? est-il bien utile de le comparer à quoi que ce soit ? Le plus simple est de considérer qu'il représente alors le maximum et initialiser MX à T(1).

L'organigramme s'écrit à la suite du précédent :



La comparaison commence au 2^e élément.

La traduction en BASIC, nous vous laissons le soin de l'écrire. Suivez pas à pas l'organigramme, n'oubliez pas de déclarer le tableau au début : tout se passera bien.

Savoir rechercher le maximum est indispensable dans la plupart des programmes de construction de diagrammes ou histogrammes.

Imaginez de représenter par des barres horizontales proportionnelles les valeurs 180, 42 et 75, sur une largeur de 30 caractères. 30 caractères figureront donc le maximum : 180.

Et les autres valeurs ? Pour que les valeurs restent proportionnelles, il suffit de diviser 180 par 30 pour obtenir la valeur qui divisera les autres quantités. Les barres horizontales auront : 180/6, 75/6, 42/6 caractères.

IV UNE OPÉRATION CLASSIQUE : LE TRI

Nous vous proposons une méthode de tri par ordre croissant, une parmi les nombreuses méthodes qui existent. Toutes exigent une certaine dextérité dans le maniement des indices. Celle-ci est une des plus simples à programmer.

Soit un tableau numérique à N éléments ($N = 5$). Il s'agit de le trier par ordre croissant.

Supposons-le ainsi constitué :

$I =$		
1	18	15
2	33	18
3	15	21
4	87	33
5	21	87

on veut obtenir

$N = 5$

Le principe consiste à comparer les nombres par couple.

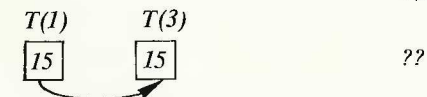
Ex : $18 > 33 = \left. \begin{array}{l} T(1) > T(2) \end{array} \right\} \text{faux}$ Ils sont en ordre croissant et ne nécessitent aucun traitement.
 $18 > 15 = \left. \begin{array}{l} T(1) > T(3) \end{array} \right\} \text{vrai}$ Les deux éléments sont à permuter.

COMMENT SE FAIT LA PERMUTATION ?

Nous ne pouvons pas écrire simplement :

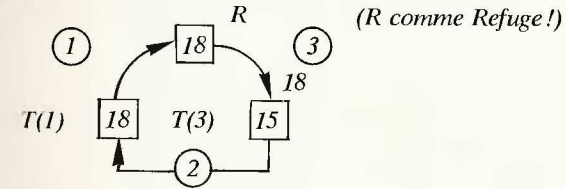
Si $T(1) > T(3)$ alors $T(1) = T(3)$
 $T(3) = T(1)$

car nous obtiendrions :



La valeur 18 doit être sauvegardée dans une variable intermédiaire avant d'échanger $T(1)$ et $T(3)$.

Comme ceci :



COMMENT DÉFINIR LES COUPLES A COMPARER

On peut suivre la méthode décrite ci-dessous, où il nous faut 2 indices I et J :

```

I = 1
Pour J = I + 1 à N
  Si T(I) > T(J)
    alors R = T(I)
         T(I) = T(J)
         T(J) = R
  FinSi
FinPour
  
```

Permutation

(On pourrait en faire l'organigramme, dont la présentation est plus évidente ; mais cette écriture, dite **algorithmique**, est plus rapide et correspond mieux au raisonnement spontané et à la structure FOR... NEXT en BASIC.)

Après ce premier passage, le tableau n'est pas tout trié ; mais on est sûr que la plus petite valeur (15) correspondant à l'indice $T(1)$ est en place.

Reste à faire la même opération pour $T(2)$, en le comparant avec chacun des suivants.

I vaut 2, J varie de 3 ($I + 1$) à N , comme précédemment. La seule différence avec l'algorithme précédent, c'est qu'entre temps I s'est incrémenté de 1.

Après le 2^e passage, les deux premiers éléments sont classés.

Quel sera le dernier passage ? Lorsqu'il ne restera plus à comparer que $T(4)$ à $T(5)$, I vaudra 4 soit $(N - 1)$, J vaudra 5, à la fois $I + 1$ et N .

L'écriture complète de la structure serait :

```

Pour I = 1 à N - 1
  Pour J = I + 1 à N
    Si T(I) > T(J)
      alors permutation
    FinPour
  FinPour
  
```

TRADUCTION EN BASIC

Nous conservons le tirage aléatoire du programme précédent pour remplir le tableau.

```
100 REM TRI
110 FOR I = 1 TO N - 1
120 FOR J = I + 1 TO N
130 IF T(I) > T(J) THEN R = T(I) : T(I) = T(J) : T(J) = R
140 NEXT J
150 NEXT I
```

On peut écrire :
140 NEXT J, I et pas de ligne 150.

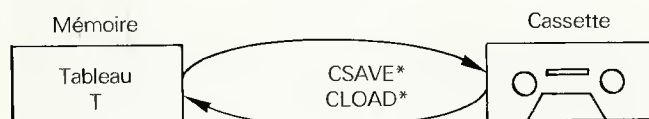
Reste ensuite à afficher le tableau trié.

```
160 REM AFFICHAGE
170 FOR I = 1 TO N
180 PRINT T(I);
190 NEXT I
```

V SAUVEGARDE D'UN TABLEAU ENTIER SUR CASSETTE

Cela peut permettre d'utiliser les mêmes valeurs dans plusieurs programmes différents.

Par exemple, vous avez un programme pour constituer et mettre à jour un tableau de noms et un tableau de numéros de téléphone, un autre programme pour la recherche d'un numéro à partir d'un nom. Il est alors utile de ne pas avoir à reconstituer chaque fois les tableaux !



Après avoir rempli un tableau T, vous voulez le ranger.

CSAVE* nom du tableau, "nom du fichier"
range sous un nom de fichier qui sera enregistré sur la cassette, un tableau dont le nom est celui qui était utilisé dans le programme.

On pourrait ranger ainsi le tableau du programme précédent :

```
200 CSAVE* T, "ALEATRI"
```

Le nom du fichier peut avoir jusqu'à 8 caractères. C'est le nom qui sera "connu" sur la cassette pour votre tableau.

Quand vous voulez le récupérer dans un autre programme, le programme commence par :

```
10 DIM T (14)
20 CLOAD* T, "ALEATRI"
```

La place pour votre tableau doit avoir été réservée au préalable s'il contient plus de 10 éléments.

CLOAD* nom du tableau, "nom du fichier"
met en mémoire, à partir d'une cassette, un tableau rangé précédemment avec un nom de fichier par la commande **CSAVE***.

N.B. Si vous avez en mémoire un tableau T rempli avec 20 éléments et que vous chargez en mémoire par **CLOAD*** un tableau T qui n'en a que 10, vous aurez dans les 10 premiers les valeurs qui étaient sur la cassette et dans les 10 derniers les anciennes valeurs.

POUR VOUS MUSCLER

1. Un mot à l'envers

Mettez en tableau les lettres d'un mot et affichez-les à l'envers.

2. L'élément manquant

Remplissez un tableau numérique avec des valeurs calculées de façon à constituer une série dont la raison est décelable.

Par exemple :

3 5 9 17 33

sont les puissances de 2 auxquelles on ajoute 1. Vous affichez tous les éléments, sauf l'avant-dernier que vous faites deviner.

3. Des phrases au hasard

Remplissez 3 tableaux-chaîne :

un tableau de sujets ;

un tableau de verbes ;

un tableau de compléments.

Constituez ensuite des phrases au hasard en effectuant un tirage aléatoire de l'indice de chacun des tableaux.

Chapitre 11

Pour aller plus loin

Ce manuel est destiné à vous apprendre à programmer avec ALICE. Après les données exposées dans les chapitres précédents, il serait bon de vous exercer, de créer des programmes personnels. Et ce n'est qu'après avoir acquis un maniement confiant de la logique de programmation, du langage et de l'ordinateur qu'il est souhaitable d'aller au-delà.

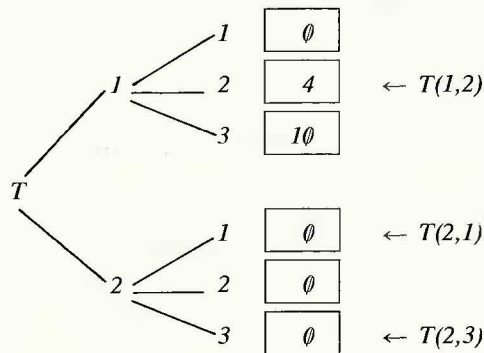
ALICE offre quelques possibilités, que nous allons vous présenter succinctement. Mais rappelez-vous que ce ne sont là que des échantillons de ce que vous découvrirez plus tard sur des ordinateurs aux capacités plus vastes, au langage plus complet.

I DES VARIABLES A PLUSIEURS INDICES

Comment comprenez-vous au premier abord les écritures suivantes ?

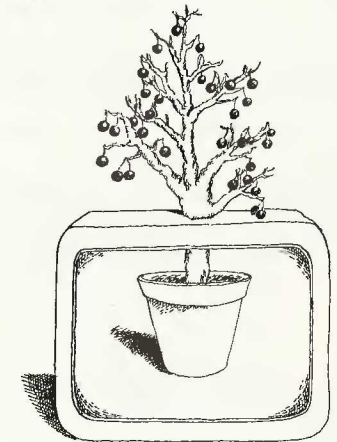
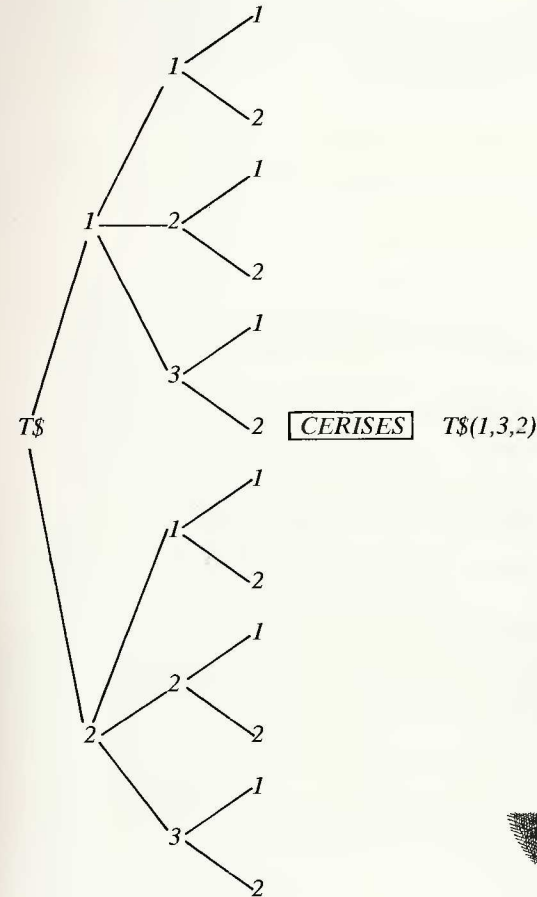
$DIM T(2,3) : T(1,2) = 4$
 $DIM T\$(2,3,2) : T\$(1,3,2) = "CERISES"$

$T(1,2)$ désigne une variable numérique T caractérisée par 2 indices dans le tableau T qui a 2×3 éléments :



Structure du tableau T

$T\$(1,3,2)$ désigne une variable chaîne $T\$\$$ caractérisée par 3 indices dans le tableau $T\$\$$ à $2 \times 3 \times 2$ éléments, soit 12 éléments organisés ainsi :



C'est une structure arborescente.

Il y a une hiérarchie des indices. Chaque niveau de hiérarchie s'appelle une "dimension".

Le tableau T est un tableau à 2 dimensions, le tableau $T\$\$$ est à 3 dimensions. Les tableaux du chapitre précédent étaient à une dimension.

Si vous deviez ranger les températures de chaque heure de chaque journée de chaque mois de chaque année pendant 5 ans, il vous faudrait un tableau déclaré ainsi :

$TM(5 , 12 , 31 , 24)$
 ans mois jours heures
 lan lmois ljour

soit un tableau à 4 dimensions, et TM(2,10,4,12) contiendrait la température à midi le 4 octobre de la 2^e année.

De tels tableaux se manient par des boucles imbriquées, autant de boucles qu'il y a de dimensions.

ALICE peut utiliser des tableaux à 3, 4, n... dimensions, ce qui est remarquable pour un si petit ordinateur (mais bien sûr, on reste limité par la taille de la mémoire). En première approche, inspirez-vous du petit programme suivant qui affiche la table de multiplication par 9 des nombres de 5 à 9, après avoir mis en tableau la table par 9 des 9 premiers nombres.

```
10 FOR I = 1 TO 9
20 FOR J = 1 TO 9           Remplissage du tableau
30 T(I,J) = I * J
40 NEXT J,I
50 CLS
60 FOR I = 5 TO 9
70 FOR J = 1 TO 9
80 PRINT T (I,J);         Affichage sur la même ligne des multiples de I
90 NEXT J
100 PRINT                 Passage à la ligne avant l'incrément de I
110 NEXT I
```

II POUR "TRAQUER L'OCTET" : POKE ET PEEK

Une fonction permet d'aller chercher le contenu (traduit en décimal) d'une adresse en mémoire.

PEEK (adresse) ou **CONTROL** **L**
donne le contenu d'une adresse,
comprise entre 12 288 et 45 055.

Amusez-vous à explorer le contenu de la mémoire avec ce programme qui vous affiche, pour chaque adresse, la valeur logée à cette adresse et quel caractère elle représente :

```
10 FOR I = 12288 TO 45055
20 P = PEEK (I)
30 PRINT I,P; " " + CHR$(P)
40 IF INT(I/15) = INT((I+15)/15) THEN IF INKEY$ = " " THEN 40
50 NEXT I
```

À l'exécution, c'est très instructif : vous voyez défiler le contenu de la mémoire, le programme s'y trouve, mais dans quel état ! Interprété par ALICE à sa façon. En repérant certaines parties aisément reconnaissables (les chiffres), vous pouvez en tirer des conclusions.

Essayez, par exemple, de repérer comment ALICE traduit les opérateurs "I" et "+" ou quel est le caractère qui traduit INT.

Vous pouvez aussi aller écrire dans les adresses-mémoire.

POKE adresse, valeur ou **CONTROL** **M**
insère une valeur entière (de 0 à 255)
à l'adresse spécifiée (de 12 288 à 45 055).

Si vous avez déjà des idées d'application, vous n'avez presque plus besoin de nous : félicitations et bonne route !

III DU BASIC A L'ASSEMBLEUR

Le BASIC est un langage très pratique, mais parfois un peu lent, car toutes les instructions doivent être traduites (on dit interprétées) en langage machine avant d'être exécutées. Pour obtenir des effets graphiques plus saisissants ou créer des jeux d'animation, vous aurez peut-être intérêt à abandonner provisoirement le BASIC. Ceci vous sera d'autant plus facile avec ALICE que vous disposez de trois instructions vous permettant de passer facilement du BASIC au code machine grâce au langage qui vous est décrit dans le GUIDE DE L'ÉDITEUR-ASSEMBLEUR.

Ces fonctions vous permettent, à partir d'un programme BASIC, soit de réserver de la place en mémoire pour un programme écrit en Assembleur, soit de faire exécuter ou de charger un programme écrit en Assembleur.

CLEAR longueur, adresse

Outre la définition qui en est donnée page 133 du Guide d'Alice, la fonction CLEAR peut aussi permettre de réserver en mémoire un certain espace pour y stocker des données ou des programmes écrits en langage machine, par exemple en Assembleur ; cet espace sera en quelque sorte "interdit" au BASIC. La première variable définit toujours le nombre d'octets réservés aux chaînes de caractères. La seconde l'adresse (voir page 142) à partir de laquelle est réservé cet espace. Ainsi CLEAR 100, 19500 réserve 100 octets pour des chaînes de caractères, et "interdit" au BASIC l'espace compris entre 19 500 et la fin de la mémoire.

EXEC adresse

Cette fonction permet, à partir d'un programme BASIC, d'appeler et d'exécuter un programme écrit en langage machine (ou en Assembleur). La variable correspond à l'adresse de lancement du programme ; celle-ci doit être comprise entre 13 250 et 45 055.

Cette fonction permet de charger un programme que vous avez écrit en Assembleur et stocké sur cassette. Un tel programme contient deux informations spécifiques :

- l'adresse de début de stockage (c'est-à-dire celle du premier octet du programme) ;
- l'adresse de lancement du programme, c'est-à-dire celle où commence l'exécution (en général la même que celle du début du stockage).

Pour exécuter le programme, il suffit de taper EXEC sans nom de variable, puisque l'adresse de lancement est déjà connue.

Pour utiliser ces instructions ou pour sauvegarder sur cassette des programmes en Assembleur, reportez-vous au Guide de l'Assembleur-Éditeur.

III

Alice :

Pour en savoir plus

Chapitre 1

Corrigé des exercices

QUELQUES TRUCS PRATIQUES POUR METTRE AU POINT VOS PROGRAMMES

Il est rare qu'un programme marche dès le premier essai. Ne vous arrachez pas les cheveux de désespoir.



1. TENEZ COMPTE DES MESSAGES D'ERREUR

Vous en avez une traduction page 173.

Quand le programme s'est arrêté sur une erreur d'exécution :

- listez la ligne où l'exécution s'est arrêtée ;
- si le message était SN ERROR, ce n'est pas grave ; cherchez si vous n'avez pas mis ; au lieu de :, oublié une parenthèse, tapé la lettre O à la place du chiffre 0, ou l à la place de I... ;
- pour tout autre message, testez systématiquement la valeur des variables.

Supposons une ligne fautive :

230 T(I) = K * 3

Tapez :

? K

10

(? est l'abréviation de PRINT, acceptée par l'ordinateur)

Tout va bien

? I

21

et le tableau T n'a que 20 éléments.

Il ne peut pas accueillir le 21^e.

L'erreur est là, aucun doute, bien visible, mais elle a certainement commencé en amont. Remontez dans votre programme ; au besoin, révisez votre analyse (ne jetez pas vos brouillons, et si vous avez fait des organigrammes, mettez-les sous verre !).

2. APRÈS UNE ERREUR, PRÉVOYEZ LE TEST

Avant la ligne 230, systématiquement ajoutez l'instruction qui arrêtera l'exécution du programme pour que vous puissiez tester les variables. Elle s'écrit simplement 229 STOP.

A la différence de END, vous poursuivez l'exécution par la commande CONT.

Quand vous avez du mal à mettre le doigt sur une erreur, placez des STOP à chaque fin de séquence et là, testez les variables. Puis continuez jusqu'au prochain STOP.

3. NE SOYEZ PAS AVARES DE REM

Mettez des titres à chaque séquence d'un programme. Vous ne le ferez pas au début, mais comme tout un chacun, vous y viendrez lorsque vous aurez dû entièrement refaire des programmes plutôt que d'essayer de comprendre ce que vous aviez bien pu écrire une semaine plus tôt.

ALICE vous semblera parfois impitoyable. Mais rassurez-vous ; aucune erreur ne pourra la déprimer ou la déranger. Au pire, elle se bloquera. Appuyez alors sur INIT, listez le programme... et cherchez l'erreur.

Tous ces programmes sont conçus pour le mode 32 colonnes. Essayez de les adapter aux autres modes.

EXERCICES DU CHAPITRE 1 (PAGE 44)

I ANALYSE DES ÉTAPES

Traduction en BASIC

1. Effacer l'écran ;

10 CLS

2. sauter 6 lignes ;

20 PRINT : PRINT : PRINT : PRINT
: PRINT : PRINT

3. déplacer le curseur de 7 caractères, et afficher "BIZARRE, BIZARRE" ; sauter encore 6 lignes (pour afficher OK).

30 PRINT TAB(7) "BIZARRE,
BIZARRE.."
40 PRINT : PRINT : PRINT : PRINT
: PRINT : PRINT

II ANALYSE

1. Effacer l'écran ;

2. sauter 4 lignes ;

3. à la ligne, afficher la chaîne de caractères précédée d'un espace, car le résultat numérique sera affiché précédé d'un espace sur la ligne au-dessous ;

4. à la ligne, afficher le résultat ;

5. à la ligne, déplacer le curseur de 19 cases et afficher "18 + 15 =" ;

6. à la ligne, déplacer le curseur de 18 cases et afficher le résultat.

Traduction en BASIC

10 CLS

20 PRINT : PRINT : PRINT : PRINT

30 PRINT " 64 + 30 ="

40 PRINT 64 + 30

50 PRINT TAB(19) "18 + 15 ="

60 PRINT TAB(18) 18 + 15

III ANALYSE

La coloration de l'écran se fait en même temps que son effacement.

Il faut donc colorer l'écran avant d'afficher les caractères.

Traduction en BASIC

10 CLS 3

20 PRINT "JE CONNAIS LES
COULEURS"

30 PRINT TAB(20) "LA PREUVE :"

EXERCICES DU CHAPITRE 2 (PAGE 54)

I LE STUDIEUX

L'ordinateur n'analyse pas le contenu d'une chaîne entre guillemets ; il ne se passera rien d'autre que l'affichage de "soustraction". Seul un changement de signe en ligne 100 (– au lieu de +) provoquerait une soustraction.

Instructions d'affichage : lignes 10, 40, 50, 60, 70, 90.

Instruction de traitement : ligne 100.

II L'INDISCRET

Analyse des étapes

1. Demander l'âge (afficher la question, entrer la réponse (A)) ;

2. calculer (1983 - A) et l'afficher.

Traduction en BASIC

10 CLS

20 INPUT "VOTRE ANNEE DE
NAISSANCE" ; A

30 PRINT "VOUS AVEZ" ; 1983-A ;
"ANS"

III LE M'AS-TU VU ?

10 CLS

20 INPUT "DONNEZ UN NOMBRE COMPRIS ENTRE 2 ET 8" ; C

30 CLS(C)

IV L'ÉCONOME

Analyse

1. Demander et entrer le nombre de km (K) ;

2. demander et entrer le nombre de litres consommés (L) ;

Traduction en BASIC

10 CLS

20 INPUT "N. DE KM" : K

30 INPUT "N. DE LITRES
CONSOMMES" ; L

3. calculer et afficher $L/K*100$.

V LE DISTRAIT

Analyse

Vous pouvez mettre le résultat exact dans une variable et l'afficher augmenté de 5 la 1^{re} fois et seul la 2^e fois.

```
40 PRINT "VOTRE VOITURE A
CONSOMME"
50 PRINT L/K*100; "LITRES/100
KM"
```

Traduction en BASIC

```
10 CLS
20 INPUT "DONNEZ LE 1er
NOMBRE"; A
30 INPUT "DONNEZ LE 2e
NOMBRE"; B
40 S = A + B
50 PRINT A; "+" ; B; "=" ; S + 5
60 PRINT "J'ETAIS DISTRAIT,
C'EST"; S
```

VI LE FLATTEUR

Analyse

Il s'agit, là encore, d'une entrée suivie d'un affichage. La seule difficulté est de savoir comment placer les espaces. Il n'y en aura aucun en début et en fin de la chaîne entrée par INPUT, donc l'espace entre l'adjectif et le prénom doit être ajoutée au moment de l'affichage.

Traduction en BASIC

```
10 CLS
20 INPUT "DONNEZ-MOI UN
ADJECTIF FLATTEUR"; F$
30 INPUT "DONNEZ-MOI VOTRE
PRENOM"; P$
40 CLS
50 PRINT : PRINT : PRINT : PRINT
: PRINT : PRINT
60 PRINT "BONJOUR,"; F$; " ";
P$; "!"
```

EXERCICES DU CHAPITRE 3 (PAGE 71)

I A QUI LA PRIORITÉ ?

$(23 + 19) / (6 / (13 - 5)) * (8 + 3) \uparrow 2$

II UNE OCCASION

Analyse des étapes

1. Entrer la cote (CA), l'âge (A) le kilométrage (K);

2. calculer les km excédentaires (E); c'est la différence entre $(A*15000)$ et K;

3. calculer la moins-value MV. Il faut d'abord connaître le pourcentage à appliquer : c'est la partie entière de $E/15000$ à laquelle on ajoute 1 pour avoir le nombre de tranches et qu'ensuite on multiplie par 5. Pour avoir

Traduction en BASIC

```
10 CLS
20 INPUT "AGE, KILOMETRAGE"
; A, K
30 INPUT "COTE"; CA
40 E = K - A*15000
50 P = (INT(E/15000) + 1)*5
60 MV = CA*P/100
```

MV, on multiplie CA (cote) par P et on divise par 100;

4. calculer et afficher la cote (CA-MV). 70 PRINT "VALEUR PROBABLE :"; CA-MV; "FRANCS"

III EN LETTRES GÉANTES

Il faut :

1. dessiner le résultat souhaité sur une grille d'écran (voir page 59);
2. déterminer pour chaque case le caractère à utiliser;
3. écrire une instruction PRINT @ pour chaque caractère.

IV VOUS LA TROUBLEZ...

L'affichage de F... FE... FÉLIX s'obtient en utilisant la fonction LEFT\$ (). Supposons que P\$ contienne le prénom. La ligne d'affichage s'écrit :
50 PRINT "BONJOUR"; LEFT\$(P\$, 1); "..."; LEFT\$(P\$, 2); "..."; P\$

V TENTEZ VOTRE CHANCE

Vous avez besoin de connaître au début le nombre de chevaux partants pour savoir les limites du tirage aléatoire.

```
10 CLS
20 INPUT "COMBIEN DE PARTANTS"; N
30 A = RND (N) : B = RND (N) : C = RND (N)
40 PRINT "TIERCE GAGNANT"
50 PRINT A, B, C.
```

On pourrait écrire directement en ligne 50 : PRINT RND (N); RND (N); RND (N).

Mais nous risquons d'avoir deux fois le même nombre.

Après le chapitre 6, vous saurez comment remédier à cela, et vous aurez besoin d'avoir pensé à loger en mémoire les résultats du tirage.

EXERCICES DU CHAPITRE 5 (PAGE 87)

I TRAIT VERTICAL

La tabulation ne change pas, il suffit d'aller à la ligne à chaque passage dans l'itération.

```
10 CLS
20 FOR I = 1 TO 15
30 PRINT TAB(15) "■"
40 NEXT I
```

Trait horizontal

Il faut sauter 6 lignes et afficher 32 fois le caractère sans aller à la ligne.

```
10 CLS
20 FOR I = 1 TO 6
30 PRINT
40 NEXT I
50 FOR I = 1 TO 32
60 PRINT "■";
70 NEXT I
```

Diagonale

Il faut à la fois aller à la ligne et incrémenter la valeur de la tabulation. Voici le listing du programme avec les lettres d'un mot.

```
10 CLS
20 INPUT "DONNEZ UN MOT"; M$
30 L = LEN (M$)
40 FOR I = 1 TO L
50 PRINT TAB(I) MID$(M$,I,1)
60 NEXT I
```

II INTERESSÉ ?

Si vous voulez calculer l'augmentation sur N années, il faut entrer en début de programme le montant mensuel (M), le pourcentage (P) que vous transformez tout de suite en $P = P/100$ et le nombre d'années N. L'itération s'écrit alors :

```
50 FOR I = 1 TO N
60 M = M + M*P
70 NEXT I
80 PRINT "VOUS AUREZ"; M; "FRANCS PAR MOIS"
90 PRINT "LA"; N + 1; "EME ANNEE"
```

III SUR 5 NOTES

```
10 FOR I = 1 TO 5
20 SOUND (RND(150), RND(10))
30 NEXT I
```

Il est préférable de ne pas étendre le tirage sur la valeur du ton jusqu'à 250, car la mélodie est trop "décousue". On peut même préférer un tirage sur 75 tons.

IV LE COMPTE À REBOURS

Le compte à rebours, c'est une itération avec un pas négatif. Pour que cela n'aille pas trop vite, il faut placer une ligne de temporisation. Le bruit de la fusée, c'est l'instruction SOUND dans une itération où I (variable de contrôle) donne le ton, au sens propre !

```
10 CLS
20 FOR I= 10 TO 1 STEP-1
30 PRINT TAB(15) I; "...": SOUND 100, 1
40 FOR Z = 1 TO 500: NEXT Z
50 NEXT I
60 CLS: PRINT @ 236, "FEU!!"
70 FOR I = 1 TO 220
80 SOUND I, 1
90 NEXT I
```

EXERCICES DU CHAPITRE 6 (PAGE 99)

I SALUER AVEC DISCERNEMENT

L'extraction du chiffre et le calcul de l'âge, vous savez faire. Le seul point à analyser ce sont les conditions.

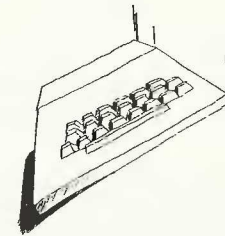
Soit S1\$ le 1^{er} chiffre du N° (S\$)

Ce peut être "1", ou "2", ou... autre chose.

Si c'est "1" on affiche "BONJOUR MONSIEUR", sinon... Ce n'est pas nécessairement "BONJOUR MADAME"; il faut d'abord vérifier que S1\$ est égal à "2".

```
50 IF S1$ = "1" THEN B$ = "MONSIEUR": GOTO 80
60 IF S1$ = "2" THEN B$ = "MADAME": GOTO 80
70 B$ = "CAMARADE"
80 PRINT "BONJOUR"; B$; "!"
```

II UN BRUIT DE CHUTE



Chose promise, chose due. Voici, au cas où vous ne l'auriez pas trouvé, le truc pour accélérer la chute.

```
10 S = - 5
20 FOR I = 200 TO 1 STEP S
30 SOUND I, 2
40 IF I = 120 THEN S = S*2
50 NEXT I
```

III LA LETTRE INTERDITE

Comment tirer la "lettre interdite"? On ne peut tirer avec RND qu'une valeur numérique. Entre 65 et 90, une valeur numérique peut être le code ASCII d'un caractère.

LI\$ = CHR\$(RND(26) + 64) (LI\$ est la lettre interdite)

Comment vérifier si elle se trouve ou non dans un mot ?

En comparant chaque lettre du mot (isolée par MID\$()) avec LI\$ dans une itération.

Si MID\$(M\$, I, 1) = LI\$ alors inutile de continuer, on affiche "Refusé" et on sort de l'itération.

```

10 LI$ = CHR$(RND(26) + 64)
20 CLS
30 INPUT "TAPEZ UN MOT"; M$
35 IF LEN(M$) < 3 THEN 30
35 IF M$ = "STOP" THEN 90
40 FOR I = 1 TO LEN(M$)
50 IF MID$(M$,I,1) = LI$ THEN PRINT "REFUSE" : GOTO 80
60 NEXT I
70 PRINT "ACCEPTÉ"
80 GOTO 30
90 INPUT "QUELLE EST LA LETTRE INTERDITE"; L$
100 IF L$ = LI$ THEN PRINT TAB(20) "BRAVO!" : END
110 PRINT "NON C'ETAIT"; LI$

```

IV LE CALCUL DU PGCD

La méthode la plus simple consiste à travailler sur le reste de la différence des 2 nombres :

```

10 CLS
15 PRINT "RECHERCHE DU PGCD DE 2 NOMBRES"
20 INPUT A, B : X = A : Y = B
30 IF A = B THEN PRINT "FARCEUR!" : END
40 IF A = 0 OR B = 0 THEN PRINT "PAS DE PGCD" : END
50 REM RECHERCHE
60 IF A > B THEN A = A - B : GOTO 80
70 IF B > A THEN B = B - A
80 IF A = B THEN PRINT "LE PGCD DE"; X; "ET"; Y; "EST :"; A : END
90 GOTO 60

```

Le calcul du PGCD de N nombres, c'est le calcul du PGCD de A et B. Le résultat est conservé dans A, on entre une nouvelle valeur pour B et l'on recommence autant de fois que nécessaire.

EXERCICES DU CHAPITRE 7 (PAGE 108)

I SÉCURITÉ SOCIALE

Il faut penser à replacer le pointeur en 1^{re} position pour chaque utilisateur. Plutôt qu'une itération finie, il vaut mieux prévoir que l'itération se fait tant que le n° de Sécurité sociale est différent de 99 (ce qui permet de taper 99 pour arrêter). Il suffit d'ajouter :

```

75 IF S$ = "99" THEN END
140 RESTORE : GOTO 70

```

II LES RECONNAISSEZ-VOUS ?

Pour chaque identité à découvrir, il faut ranger en DATA et lire un nom, un renseignement.

Étant donné la longueur de chaque donnée, il vaut sans doute mieux mettre un nom, un renseignement par numéro de ligne. N'oubliez pas le test d'arrêt de DATA (ou bien prévoyez juste si vous utilisez une structure FOR... NEXT).

III FOLK, CLASSIQUE OU ROCK ?

Comme l'intervalle des tons n'est pas régulier de 1 à 255 (il va croissant), il est difficile d'établir une relation simple avec la gamme connue.

Le plus simple est de fixer une note de départ, par exemple 60, puis évaluer l'intervalle souhaitable entre 2 tons, 12 peut-être, et d'affecter un identificateur.

Cela permet d'appeler les notes 1, 2, 3 plutôt que 60, 72, 84.

Et 2.5 représente évidemment un demi-ton.

```

10 DATA 1, 6, 1, 3, 4, 6, 4, 3, 6, 6, 6, 3
12 DATA 4, 3, 2.5, 3, 4, 9
20 DATA 99, 6
30 N = 60
40 IN = 12
100 READ X, D
105 IF X = 99 THEN END
110 J = N + INT(IN*X)
120 SOUND J, D
130 GOTO 100

```

Variante

(THEN 140)

140 RESTORE : GOTO 100 permet de faire une ritournelle.

EXERCICES DU CHAPITRE 8 (PAGE 118)

I LES RECTANGLES SUPERPOSÉS

```

10 CLS 0
15 C = RND(8)
20 F = 2
25 IF K > 8 THEN F = 1.5
30 IF K > 15 THEN F = 1
35 IF K > 25 THEN F = 0.5
40 L = RND(15)*F
45 H = RND(8)*F
50 DL = RND(60)
55 IF DL + L > 63 THEN DL = 62 - L
60 DH = RND(30)
65 IF DH + H > 31 THEN DH = 30 - H
100 FOR X = DL TO DL + L
110 FOR Y = DH TO DH + H
120 SET (X,Y,C)
130 NEXT Y, X
150 C = C + 1 : IF C = 9 THEN C = 1
160 IF INKEY$ = " " THEN 160
170 K = K + 1
180 GOTO 25

```

Réduction progressive de la taille du rectangle

Largeur
Hauteur
Point de départ

Attend qu'on presse une touche.

Pour dessiner le rectangle suivant qui, éventuellement, se superposera partiellement au précédent, il suffit d'appuyer sur une touche quelconque du clavier. L'exécution s'arrête quand on enfonce la touche BREAK.

II RATTRAPEREZ-VOUS LA GOUTTE ?

```

5 CLS
10 PRINT "RATTRAPEREZ-VOUS LA GOUTTE ?"
15 PRINT "POSITION DE LA SOUCOUBE"
20 INPUT "(DE 5 A 55)"; PS
25 IF PS > 55 OR PS < 5 THEN 20
30 PRINT "LARGEUR DE LA SOUCOUBE"
35 INPUT "(DE 2 A 10)"; D
37 D = D/2
40 IF D < 1 OR D > 5 THEN 35
50 CLS 0
60 FOR I = PS - D TO PS + D
65 SET (I,31,8)
70 NEXT I
90 FOR I = 1 TO 5
100 L = RND(61)

105 C = RND(7)
110 H = RND(20)
120 FOR X = 1 TO L
130 SET (X,H,C)
140 FOR J = 1 TO 80 : NEXT J
150 RESET (X - 1, H)
160 NEXT X
170 FOR Y = H + 1 TO 31
175 SET (L,Y,C)
180 RESET (L,Y - 1)
190 NEXT Y
195 IF L >= PS - D AND L <= PS + D THEN PRINT "GAGNE !" :
    GOTO 240
200 NEXT I
220 REM ON PEUT ECRIRE ICI GOTO 15
240 INPUT "ON RECOMMENCE ?" ; R$
250 IF R$ = "OUI" THEN 15

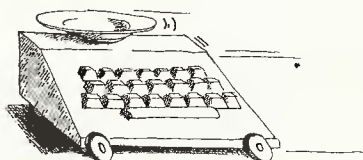
```

Dessin de la soucoupe

Longueur du déplacement
horizontal
Couleur
Hauteur du point de départ

Tracé horizontal
Temporisation

Tracé vertical



III LE BATEAU

```

10 CLS 0
30 READ Y,D,F,C
40 FOR X = D TO F
50 SET (X,Y,C)
60 NEXT X
65 IF Y = 21 THEN 100
70 GOTO 30
100 C = 3
110 Y1 = 20 : Y2 = 31
120 GOSUB 300
150 GOTO 150
200 DATA 10, 28, 29, 5, 11, 29, 30, 5
202 DATA 12, 29, 30, 5, 13, 15, 32, 5
204 DATA 14, 14, 33, 5, 15, 13, 33, 5
206 DATA 15, 37, 48, 4, 16, 12, 32, 5, 16, 33, 47, 4
207 DATA 17, 7, 42, 4, 17, 43, 46, 5
208 DATA 18, 7, 33, 4, 18, 34, 45, 5
209 DATA 19, 8, 44, 5, 20, 8, 44, 5
210 DATA 21, 9, 43, 5
220 GOTO 220
300 FOR Y = Y1 TO Y2
310 FOR X = 1 TO 63
330 IF POINT (X,Y) < 1 THEN SET (X,Y,C)
340 NEXT X,Y
350 RETURN

```

Dessin du bateau

Remplissage en bleu des points
non allumés

Ce programme s'interrompt en appuyant sur la touche **BREAK**.

EXERCICES DU CHAPITRE 9 (PAGE 129)

I UN TRAIT POUR SOULIGNER

Pensez à employer dans le sous-programme des noms de variables que vous utilisez rarement dans les programmes, afin d'éviter les risques d'interférence. (Pour les mêmes raisons, évitez les DATA dans un sous-programme.)

```

500 FOR WZ = 1 TO WN
510 W$ = W$ + C1$
520 NEXT WZ
530 RETURN
A essayer avec :
5 INPUT N
10 WN = N : C1$ = "%"
20 GOSUB 500
30 PRINT W$
40 END

```

II RÉPONSE EN MUSIQUE

Évitez les DATA.

Calculez les valeurs :

600 T1 = 20 : T2 = 50

610 FOR W = 1 TO 2

620 SOUND T1, 3

630 SOUND T2, 4

640 T1 = T1*2 : T2 = T2*2

650 NEXT W

660 RETURN

III CALCUL A LA DEMANDE

Ici on a le choix entre construire 3 sous-programmes appelés par GOSUB ou de simples séquences de programmes où l'on se branche par GOTO.

*Structure en cas
de sous-programmes*

```
ON... GOSUB
END
1er sous-programme
RETURN
2e sous-programme
RETURN
3e sous-programme
RETURN
```

*Structure en cas
de GOTO*

```
ON... GOTO
1re séquence
END
2e séquence
END
3e séquence
END
```

EXERCICES DU CHAPITRE 10 (PAGE 139)

I UN MOT A L'ENVERS

Utilisez MID\$() pour l'extraction des lettres.

Pensez à calculer d'abord la longueur du mot et, le cas échéant, déclarez le tableau au nombre d'éléments nécessaires (ou bien dimensionnez-le d'avance largement).

Affichez à l'envers par une itération à pas négatif (vous pouvez même essayer STEP - 2).

II L'ÉLÉMENT MANQUANT

L'itération aurait la structure suivante :

X = 2

Pour I = 1 à N

T(I) = X ↑ I + 1

Si I = N - 1

alors afficher "??"

sinon afficher T(I)

FinSi

FinPour

L'exponentiation calcule parfois des résultats un peu faux.

Il est prudent d'écrire que T(I) = INT (X ↑ I) + 1.

III DES PHRASES AU HASARD

Jouez sur tous les tableaux avec un seul indice.

Remplissage des tableaux :

Pour I = 1 à N

entrer SU\$(I)

entrer V\$(I)

entrer CO\$(I)

FinPour

	Sujets SU\$	Verbes V\$	Compléments CO\$
I = 1			
2			
N			

Tirage aléatoire et composition de la phrase se font ensemble :

PH\$ = SU\$ (RND(N)) + " " + V\$ (RND(N)) + " " + CO\$ (RND(N))

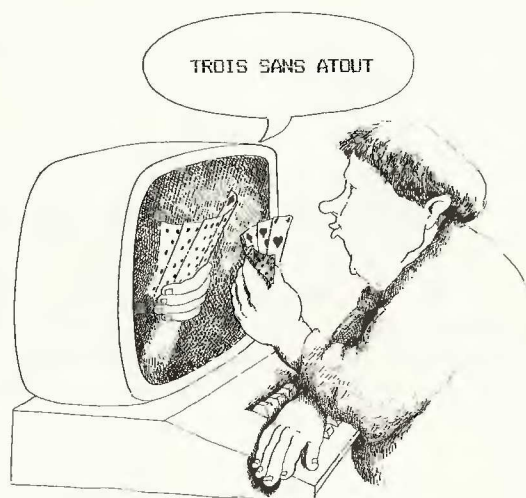
Chapitre 2

Qu'est-ce qu'un micro-ordinateur ?

Dans ce chapitre, découvrons de manière simple ce qu'est un micro-ordinateur et comment il fonctionne.

Les micro-ordinateurs sont parfois appelés "ordinateurs individuels". Leur format réduit, la baisse constante de leur prix d'achat contribuent à les répandre.

Dans les bureaux, ils servent à tenir la comptabilité, établir des factures, imprimer des étiquettes d'adresses à partir d'un fichier, etc. A la maison, ils peuvent aussi aider à faire les comptes, à tenir en ordre une collection de timbres, à réviser son baccalauréat ; mais, surtout, on joue avec eux : jeux graphiques, jeux logiques, petits et grands se passionnent.



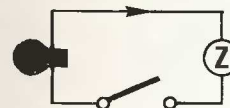
L'ordinateur est une machine à trier automatiquement de l'information.

Le mot information est un mot à double sens :

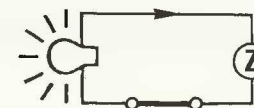
- c'est un renseignement à propos de quelque chose. Quand on dit : "Avez-vous une information sur le déroulement des courses ?", on attend un renseignement ;
 - c'est aussi le signal ou l'ensemble des signaux qui transmettent un renseignement. Par exemple les — . du code Morse ou les lettres de l'alphabet dans un texte.
- L'informatique, c'est l'art (ou la science) de collecter, traiter et transmettre de l'information-signal au moyen d'un ordinateur, afin que nous, les utilisateurs, puissions la comprendre et l'utiliser comme renseignement.

Ainsi, si on frappe sur la touche A du clavier d'un ordinateur, un A s'affiche à l'écran. Que s'est-il passé dans l'ordinateur ? Des impulsions électriques ont circulé, et l'ordinateur a "reconnu" la forme A selon un système que nous allons expliquer sans trop entrer dans le détail.

C'est oui ou c'est non ?



NON
Le courant ne passe pas, la lampe n'est pas allumée



OUI
Le courant passe et allume la lampe

C'est sur ce principe que fonctionne un ordinateur où chaque composant électronique ne peut avoir que 2 états : c'est un fonctionnement *binnaire*.

Sur la réalité de ce fonctionnement, on a bâti un code *binnaire* à 2 chiffres, 0 ou 1. Un chiffre binaire se dit en anglais Binary Digit, en abrégé *bit*.

L'état d'un composant est représenté sur le papier par un bit, selon la convention suivante :

0 pour "Non, le courant ne passe pas"

1 pour "Oui, le courant passe".

Sur 1 bit, on ne représente que 2 "combinaisons". Mais avec deux composants, c'est-à-dire sur 2 bits, on en représente 4 (c'est-à-dire 2^2)

00, 01, 10, 11

Sur 3 bits, on en représente 8 (c'est-à-dire 2^3)

000, 001, 010, 011, 100, 101, 110, 111

Et ainsi de suite.

Dans la plupart des ordinateurs, les impulsions circulent par paquets de 8, figurés sur 8 bits. On appelle *octet* un groupe de 8 bits. Exemple : 01100101.

Sur un octet, on peut représenter 256 (soit 2^8) combinaisons.

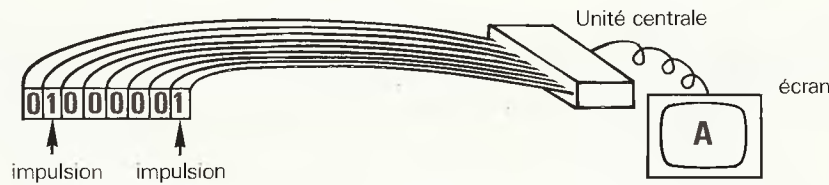
... Des combinaisons pas très faciles à lire.

On a donc établi des systèmes de correspondance, ou codes, entre la figuration binaire indispensable pour représenter la réalité physique de l'ordinateur et les notations courantes dans notre langage (chiffres et lettres). En voici un exemple :

Binaire	Chiffres en code décimal	Lettres en code
01000001	65	A
01011010	90	Z

Quand on frappe au clavier la touche A, 2 impulsions sont transmises, et leur position détermine les circuits qui afficheront les points lumineux disposés en forme de A sur l'écran.

Sur ce même principe, l'ordinateur peut effectuer les calculs les plus complexes.



I AU CŒUR DE L'ORDINATEUR, L'UNITÉ CENTRALE

C'est la partie la moins volumineuse, la moins visible. Dans ALICE, elle est située sous le clavier. Tous les ordinateurs effectuent à peu de choses près les mêmes opérations de traitement. Mais les unités centrales diffèrent surtout les unes des autres par leur capacité de mémoire. Cette capacité se compte en kilo-octets, ou *k-octets*. "Kilo" devrait signifier 1 000. Mais non... le kilo des informaticiens fait 1 024 unités : 2^{10} , voyons !

II LES MÉMOIRES D'UN ORDINATEUR

Une mémoire *permanente*, livrée "pleine" quand vous achetez l'ordinateur, contient tout son système de communication, en particulier son système de traduction entre le binaire et le langage que nous utilisons.

Du fait qu'elle n'est pas modifiable par l'utilisateur, on la nomme Read Only Memory : mémoire où l'on peut lire mais pas écrire (en abrégé, *ROM*). On dit aussi mémoire morte.

Une mémoire *temporaire* est à la disposition de l'utilisateur. Des informations ou *données* entrées par l'utilisateur y sont rangées, et l'ordinateur doit pouvoir aller les chercher à la demande, dans n'importe quel ordre, un peu comme un magasinier parcourt ses rayons pour satisfaire à la demande imprévisible du client. C'est pourquoi on a appelé cette mémoire Random Access Memory (en abrégé, *RAM*), ce qui signifie : mémoire à accès aléatoire. On dit aussi mémoire à accès direct.

Plus couramment, dans la pratique, on l'appelle mémoire vive... Si vive qu'elle s'évapore chaque fois qu'on coupe le courant.

Aussi la sauvegarde du contenu de cette mémoire est-elle chose souhaitable. Elle peut éventuellement être assurée par une mémoire auxiliaire sur bande magnétique ou disquette située à l'extérieur de l'unité centrale.

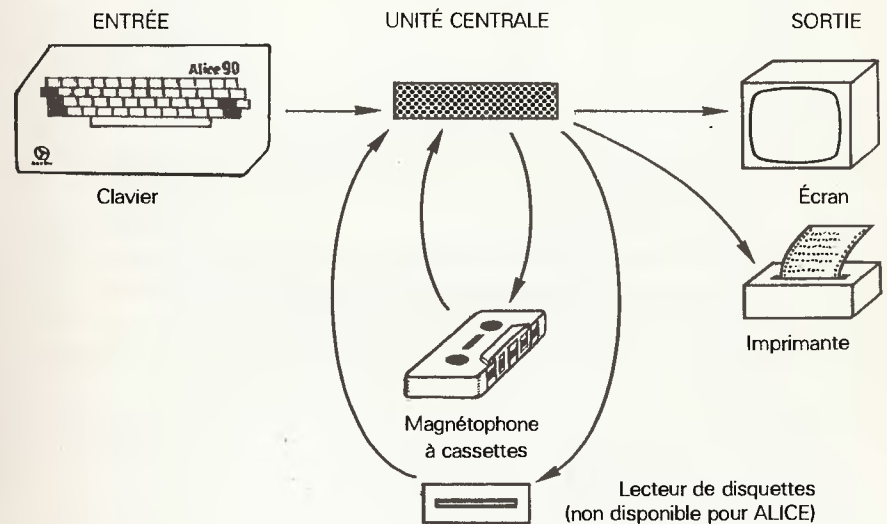
III LE LANGAGE

Pour dialoguer de manière intelligible avec l'ordinateur, il nous faut un langage commun. Le langage le plus répandu est le BASIC, utilisé dans tous les pays, sur pratiquement toutes les machines. Il a été conçu pour que des débutants puissent en maîtriser les premiers éléments très rapidement. ALICE utilise le BASIC, langage que ce manuel va vous faire découvrir. Ne vous étonnez pas de trouver beaucoup de mots anglais dans le BASIC, car il a été développé aux États-Unis par la société Microsoft. C'est ainsi presque une norme mondiale aujourd'hui. Un lexique anglais-français des instructions BASIC figure en page 200.

IV LES ORGANES EXTÉRIEURS OU PÉRIPHÉRIQUES

Ils assurent les communications entre l'univers binaire ultra-rapide de l'unité centrale et notre monde à nous, lent mais complexe et subtil.

En voici quelques exemples, pris parmi les plus courants.



Chapitre 3

Résumé des commandes, instructions et fonctions BASIC

MOT BASIC	Ce qu'il fait	Comment il s'utilise
<i>ABS</i> (n)	Calcule la valeur absolue, c'est-à-dire indépendante du signe.	? ABS (- 5) 5
<i>ASC</i> (c)	Renvoie le code ASCII du premier caractère de la chaîne c.	T\$ = "ABC" : ? ASC(T\$) 65
<i>CHR\$</i> (n)	Renvoie le caractère ou la commande correspondant à n dans le code ASCII.	? CHR\$(66) B
<i>CLEAR</i> <i>CLEAR</i> nombre	Réserve la moitié de l'espace indiqué par le nombre. <i>CLEAR</i> seul réserve 100 caractères pour les chaînes.	<i>CLEAR</i> 510 est le maximum utile
<i>CLEAR</i> n1,n	Réserve n1 octets, à partir de l'adresse n2 pour y stocker des programmes en ASSEMBLEUR.	<i>CLEAR</i> 50,19500
<i>CLOAD</i> nom de fichier	Charge le fichier à partir d'une cassette (le premier venu, s'il n'y a pas de nom après <i>CLOAD</i>). Le nom a au maximum 8 caractères.	<i>CLOAD</i> "FANNY"
<i>CLOAD*</i> nom du tableau, nom du fichier	Charge des données numériques dans un tableau à partir d'un fichier sur cassette constitué par <i>CSAVE*</i> .	<i>CLOAD*</i> T, COMPTE\$
<i>CLOADM</i> nom	Charge en mémoire centrale un programme en Assembleur stocké sur cassette.	<i>CLOADM</i> "TOTO"
<i>CLS</i> <i>CLS</i> n	Efface l'écran dans la couleur spécifiée par n. <i>CLS</i> seul produit un écran vert. n doit valoir entre 0 et 8 pour sélectionner une couleur. <i>CLS</i> 32, 40, 80, 81 définissent le mode d'affichage.	<i>CLS</i> 7
<i>CONT</i>	Commande qui permet de poursuivre l'exécution d'un programme après une interruption volontaire.	<i>CONT</i> ENTER

MOT BASIC	Ce qu'il fait	Comment il s'utilise
<i>COS</i> (n)	Fournit le cosinus d'un angle n mesuré en radians.	? COS(1.5) .0707372015
<i>CSAVE</i> "nom de programme"	Sauvegarde un programme en fichier sur cassette. Le nom doit avoir au maximum 8 caractères.	<i>CSAVE</i> "DESSIN"
<i>CSAVE*</i> nom de tableau, "nom de fichier"	Sauvegarde le tableau dont le nom est indiqué dans le fichier sur cassette.	<i>CSAVE*</i> T, "MOTOS"
<i>DATA</i> donnée, donnée, donnée...	Enregistre des données séquentiellement dans un programme. S'utilise avec <i>READ</i> .	<i>DATA</i> 8, 9, 34
<i>DIM</i> nom de tableau (n), nom de tableau (n)...	Déclare un ou plusieurs tableaux à 1 dimension et n éléments.	<i>DIM</i> T(25)
<i>DIM</i> nom de tableau (n, n), nom de tableau (n, n)	Déclare un ou plusieurs tableaux à 2 dimensions et n*n éléments.	<i>DIM</i> T(3,25)
<i>END</i>	Arrête l'exécution d'un programme.	<i>END</i>
<i>EXEC</i> adresse	Exécute un programme écrit en Assembleur; l'adresse est celle de lancement du programme.	<i>EXEC</i> 18432
<i>EXP</i> (n) n1 ↑ n2	Fournit l'exponentielle naturelle de n. Élévation à une puissance. Élève n1 à la puissance n2. ↑ s'obtient au clavier par CONTROL Z .	? EXP(5) 148.413159 ? 2 ↑ 10 1024
<i>FOR</i> variable =n TO n1 <i>STEP</i> n <i>NEXT</i> variable	Détermine les bornes d'une itération qui se fera à partir d'une valeur n de la variable jusqu'à ce qu'elle ait atteint une valeur n1 par "pas" spécifié après <i>STEP</i> . Si l'on n'utilise pas <i>STEP</i> , le pas est de 1.	<i>FOR</i> X = 1 TO N <i>NEXT</i> X → N passages <i>FOR</i> I = 8 TO 12 <i>STEP</i> 2 <i>NEXT</i> I → 3 passages
<i>GOSUB</i> n° de ligne <i>GOTO</i> n° de ligne	Envoie l'exécution à un sous-programme qui commence au n° de ligne spécifié. Provoque une rupture dans la séquence d'exécution qui reprend au n° de ligne indiqué.	<i>GOSUB</i> 2000 <i>GOTO</i> 145

MOT BASIC	Ce qu'il fait	Comment il s'utilise
<i>IF</i> condition <i>THEN</i> instruction	L'instruction est exécutée si la condition est vraie. Toutes les instructions qui suivent <i>THEN</i> sont soumises à la condition.	IF X = A THEN PRINT "GAGNE" : K = K + 1
<i>INKEY\$</i>	Sert à lire un caractère provenant du clavier. S'utilise généralement dans une itération de façon à permettre un "balayage" du clavier dans l'attente d'une touche frappée.	50 IF INKEY\$ = " " THEN 50 60 IF INKEY\$ = "N" THEN END
<i>INPUT</i> "message" ; identificateur	Après l'affichage du message (facultatif) et d'un point d'interrogation, attend les données fournies à partir du clavier et les affecte à l'identificateur.	INPUT "SUITE" ; B
<i>INT</i> (n)	Convertit n en l'entier immédiatement inférieur ou égal à n.	? INT(7/2) 3
<i>LEFT\$</i> (c, n)	Renvoie les n premiers caractères de la chaîne c.	? LEFT\$ ("ROSE", 2) RO
<i>LEN</i> (c)	Renvoie le nombre de caractères de la chaîne c.	? LEN ("ROSE") 4
<i>LET</i> variable = expression de même type	Attribue la valeur de l'expression à la variable. Le mot <i>LET</i> est facultatif.	LET X = 12 + 3 ? X 15
<i>LIST</i> n° de ligne	Liste une ligne. Liste toutes les lignes si aucun numéro n'est précisé.	LIST 123
<i>LIST</i> n° de ligne- n° de ligne	Liste les lignes comprises entre les 2 numéros.	LIST 50 - 100
<i>LIST</i> - n° de ligne	Liste du début du programme jusqu'à la ligne dont le n° est indiqué.	LIST - 35
<i>LIST</i> n° de ligne -	Liste depuis le n° de ligne indiqué jusqu'à la fin du programme.	LIST 65 -
<i>LLIST</i>	Fonctionne comme <i>LIST</i> , mais sort la liste sur l'imprimante.	
<i>LOG</i> (n)	Calcule le logarithme naturel de n.	? LOG(4) 1.38629436
<i>LPRINT</i>	Fonctionne comme <i>PRINT</i> , mais affiche sur l'imprimante.	LPRINT A ; B

MOT BASIC	Ce qu'il fait	Comment il s'utilise
<i>MEM</i>	Fait apparaître le nombre d'octets inemployés en mémoire.	? MEM 2987
<i>MID\$</i> (c, n1, n2)	Extrait de la chaîne c, à partir du caractère n1, n2 caractères.	? MID\$ ("RO- SIER", 2,2) SI
<i>NEW</i>	Efface le contenu de la mémoire. Laisse la place nette pour entrer un nouveau programme.	
<i>ON</i> variable <i>GOSUB</i> n° de ligne, n° de ligne, etc.	Branche vers un sous-programme qui commence à un n° de ligne désigné dans la file qui suit <i>GOSUB</i> par la valeur de la variable.	ON X GOSUB 50, 75 branche sur la ligne 75 si X vaut 2
<i>ON</i> variable <i>GOTO</i> n° de ligne, n° de ligne, etc.	Fonctionne comme <i>ON... GOSUB</i> , mais ne mémorise pas de point de retour.	ON X GOTO 50, 75
<i>PEEK</i> (n)	Fait apparaître le contenu codé en décimal (de 0 à 255) d'une adresse en mémoire n.	? PEEK (32176) 0
<i>POINT</i> (n1, n2)	Vérifie si un point graphique placé en position horizontale n1 et verticale n2 est allumé ou éteint. Renvoie 0 s'il est éteint ; Renvoie 1 à 8 selon la couleur s'il est allumé ; Renvoie - 1 s'il est occupé par un caractère ; En mode 80 ou 81 : Renvoie - 1 s'il est occupé par un caractère ; Renvoie 0 si le point a la couleur de marge ; Renvoie 1 s'il a la couleur d'intensité ; Renvoie 2 s'il a la couleur de demi-intensité.	? POINT (12,1) - 1
<i>POKE</i> n1, n2	Insère une valeur n1 comprise entre 0 et 255 (code ASCII) à l'adresse n2 ; n2 doit être compris entre 12288 et 45055.	POKE 17100, 66 plante le code du caract- ère B à l'adres- se 17100
<i>PRINT</i>	Le curseur passe à la ligne.	
<i>PRINT</i> n ou c (ou ?)	Passé à la ligne et affiche le contenu de l'expression. Si n et c sont des expressions calculées, leur résultat est affiché.	? "JOJO" JOJO ? 12 + 67 79

MOT BASIC	Ce qu'il fait	Comment il s'utilise
<i>PRINT</i> n ; c ; c	Accole en les affichant le contenu des expressions.	? "JOJO" ; "ET" JOJOET
<i>PRINT</i> n, c	Affiche le contenu de n et c par colonnes de 16 caractères.	? "JOJO", "ET" JOJO ET
<i>PRINT</i> <i>TAB</i> (n) expression	Passe à la ligne et positionne le curseur à la position horizontale qui suit n (n commence à 0; il y a passage à la ligne à chaque multiple de 32, 40 ou 80; n ne peut excéder 255).	?TAB(3) "5" 5
<i>PRINT</i> @ n, n ou c	Affiche n ou c en commençant à la position d'écran désignée par n, qui peut valoir de 0 à 511, 999 ou 1999 suivant le mode utilisé.	?@ 66, "X" X sur la 3 ^e ligne
<i>READ</i> variable, variable, etc.	Lit la valeur suivante dans une file constituée par DATA et l'affecte à la variable qui le suit, fait de même pour la variable suivante, etc.	READ A, B
<i>REM</i> commentaire	Tout ce qui suit REM est ignoré. On s'en sert pour placer des commentaires ou des titres dans un programme.	10 X = X + 1 : REM INCREMENTATION
<i>RESET</i> (n1, n2)	Efface le point situé en n1 et n2 ; En mode 80, lui donne la couleur de marge.	RESET (14,15)
<i>RESTORE</i>	Remplace le pointeur au début de la file constituée par DATA.	
<i>RETURN</i>	A la fin d'un sous-programme, ramène l'exécution à l'instruction qui suit l'appel par GOSUB.	
<i>RIGHT</i> \$ (c, n)	Extrait les n derniers caractères de la chaîne c.	? RIGHT\$ ("PANIER", 5) ANIER
<i>RND</i> (n)	Renvoie un entier positif pseudo-aléatoire dont la valeur est comprise entre 1 et n.	? RND (20) 15
<i>RUN</i>	Commande qui fait exécuter un programme.	
<i>RUN</i> n° de ligne	Fait commencer l'exécution à partir de la ligne indiquée.	
<i>SET</i> (n1,n2,n3)	Allume un point graphique dans la couleur n3, à la position horizontale n1 (entre 0 et 63,79 ou 159) et verticale n2 (entre 0 et 31, 49 ou 124). En mode 80, n3 varie entre 0 et 2.	SET (X,Y,C)

MOT BASIC	Ce qu'il fait	Comment il s'utilise
<i>SET</i> *M.I.D	En CLS 80, et CLS 81, fixe la couleur de Marge, d'Intensité et de Demi-intensité.	SET*3,4,5
<i>SGN</i> (n)	Renvoie le signe de n - 1 s'il est négatif ; 0 s'il est égal à 0 ; 1 si n est positif.	? SGN (50-60) - 1
<i>SIN</i> (n)	Calcule le sinus d'un angle n mesuré en radians.	? SIN (1) 841470985
<i>SKIPF</i>	Recherche sur une cassette le programme demandé en ignorant les autres.	SKIPF "ADDIT"
<i>SOUND</i> n1, n2	Émet un son de hauteur n1 (de 0 à 255) pendant une durée n2 dont chaque unité représente 7.5/100 ^e de seconde. Calcule la racine carrée de n.	SOUND 50, 55
<i>SQR</i> (n)		
<i>STOP</i>	Arrête l'exécution du programme qui peut ensuite reprendre par CONT.	IF I = 100 THEN STOP X = 1983
<i>STR</i> \$ (n)	Change le type de l'expression numérique et en fait une chaîne.	? RIGHT\$ (X\$, 2) 83
<i>TAN</i> (n)	Calcule la tangente d'un angle n mesuré en radians.	? TAN (2) -2.18503986
<i>VAL</i> (c)	Change le type de l'expression chaîne c et en fait un nombre. Cette fonction est l'inverse de STR\$.	N\$ = "83" ? 100 - VAL (N\$) 17

Dans la liste ci-dessus :

n est mis pour "expression numérique". Ce peut être :

- un nombre exprimé en chiffres ; ex : 123
- un identificateur numérique ; ex : B
- une expression calculée ; ex : (B*C)/2
- le résultat numérique d'une fonction ; ex : RND(10)

c est mis pour "expression chaîne". Ce peut être :

- des caractères entre guillemets ; ex : "21 janvier"
- un identificateur de type chaîne ; ex : B\$
- une expression concaténée ; ex : B + "S"
- le résultat chaîne d'une fonction ; ex : MID\$(B\$)

Chapitre 4

Codes et tables

CODES CARACTÈRES ASCII (en décimal)

Touche	Mode majuscules		Mode inversé		CONTROL **
	Sans SHIFT	Avec SHIFT	Sans SHIFT	Avec SHIFT	
1	49	33	49	33	142
2	50	34	50	34	147
3	51	35	51	35	152
4	52	36	52	36	151
5	53	37	53	37	150
6	54	38	54	38	148
7	55	39	55	39	149
8	56	40	56	40	157
9	57	41	57	41	134
0	48	-	48	-	-
A	65	233*	97	65	21
B	66	224	98	66	179
C	67	226	99	67	178
D	68	231	100	68	130
E	69	237	101	69	155
F	70	230	102	70	144
G	71	229	103	71	132
H	72	72	104	72	163
I	73	73	105	73	138
J	74	74	106	74	129
K	75	75	107	75	158
L	76	76	108	76	188
M	77	77	109	77	146
N	78	78	110	78	185
O	79	79	111	79	165
P	80	80	112	80	199
Q	81	239	113	81	8
R	82	236	114	82	156
S	83	232	115	83	9
T	84	235	116	84	140
U	85	85	117	85	128
V	86	225	118	86	181
W	87	238	119	87	10
X	88	227	120	88	177
Y	89	234	121	89	143
Z	90	228	122	90	94

* selon couleur

** non accessible directement

Minuscules : code 97 à 122

















Touche	Mode majuscules		Mode inversé		CONTROL
	Sans SHIFT	Avec SHIFT	Sans SHIFT	Avec SHIFT	
:	58	42	58	42	137
-	45	61	45	61	145
@	64	-	64	-	136
;	59	43	59	43	182
,	44	60	44	60	187
.	46	62	46	62	183
/	47	63	47	63	186
!	-	33	-	33	142
~	-	34	-	34	147
#	-	35	-	35	152
\$	-	36	-	36	151
%	-	37	-	37	150
&	-	38	-	38	148
'	-	39	-	39	149
(-	40	-	40	157
)	-	41	-	41	134
*	-	42	-	42	137
=	-	61	-	61	145
+	-	43	-	43	182
<	-	60	-	60	187
>	-	62	-	62	183
?	-	63	-	63	186
BREAK	3	3	3	3	-
ENTER	13	13	13	13	-
Barre d'espace	32	32	32	32	-

CARACTÈRES CLAVIER	
CARACTÈRE	ACTION
ENTER	Indique à l'ordinateur que vous avez atteint la fin de votre ligne de programme ou de votre ligne de commande.
BREAK	Arrête l'exécution de votre programme.
SHIFT	Interrompt l'exécution de votre programme. Enfoncez n'importe quelle touche pour continuer. <i>sauf bracket unit</i>
SHIFT0	En mode 32 et 40, opère le passage de et vers le mode inversé. Permet d'accéder aux minuscules en mode 80.

SYMBOLES BASIC	
SYMBOLE	EXPLICATION
" "	Indique que la donnée entre guillemets est une constante.
:	Sépare les "instructions" de programme sur une même ligne.
()	Demande à l'ordinateur d'effectuer d'abord l'opération entre parenthèses.
;	Force l'impression à la droite les unes des autres des variables et constantes.

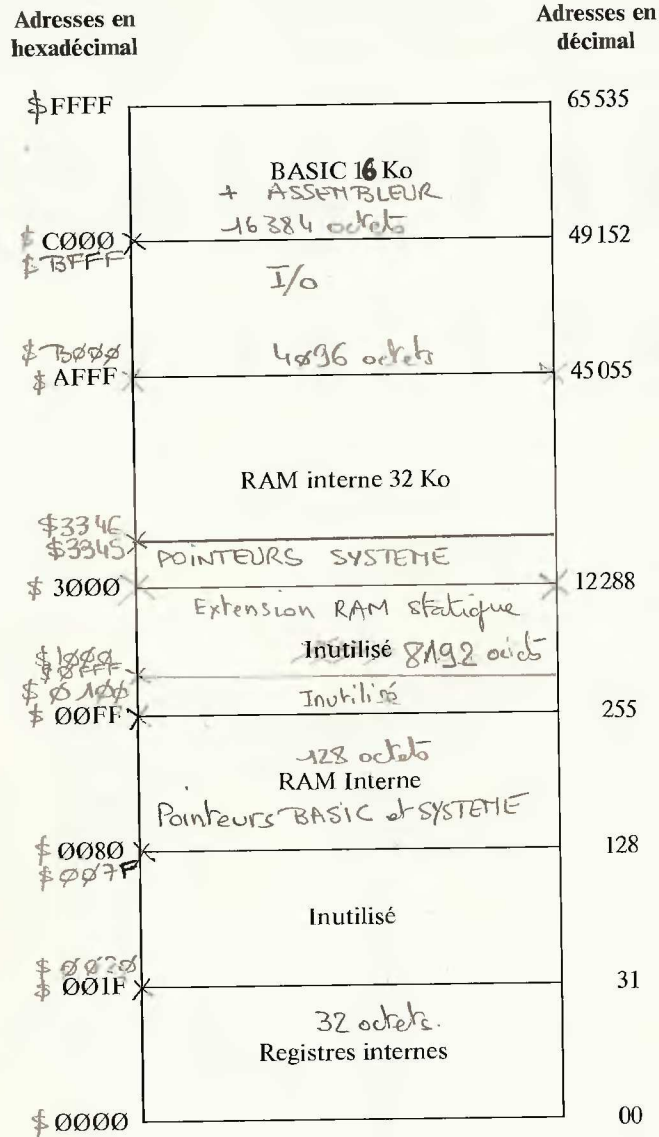
OPÉRATEURS BASIC	
OPÉRATEUR	FONCTION
+	Combine des chaînes
+	Addition
-	Soustraction
*	Multiplication
/	Division
↑	Exponentiation
=	Égale
>	Plus grand que
> = ou = >	Plus grand ou égal à
< = ou = <	Plus petit ou égal à
<	Plus petit que
<> ou > <	Non égal à
AND	ET logique
OR	OU logique
NOT	NON logique

CODE	COULEUR
0	noir
1	vert
2	jaune
3	bleu roi
4	rouge
5	ivoire
6	bleu pâle
7	mauve
8	orange

CARACTÈRES GRAPHIQUES							
Voici les codes des caractères graphiques du micro-ordinateur ALICE. Pour les afficher, mettez-vous en mode 32 ou 40 et utilisez CHR\$(code) suivi du code du caractère. Par exemple, PRINT CHR\$(129) produit le caractère 129.							
Dessin N° 10							
							
128	129	130	131	132	133	134	135
							
136	137	138	139	140	141	142	143
Pour créer ces caractères dans l'une des couleurs ci-dessous, ajoutez-en le numéro au code du caractère. Par exemple, PRINT CHR\$(129 + 16) produit le caractère 129, mais la zone verte est maintenant jaune.							
+ 16 - jaune	+ 64 - ivoire	+ 96 - mauve					
+ 32 - bleu roi	+ 80 - bleu pâle	+ 112 - orange					
+ 48 - rouge							

NOTES DE MUSIQUE				
NOTE	OCTAVE			
	Première	Deuxième	Troisième	Quatrième
DO	-	102	180	219
DO #	-	111	185	221
RÉ	-	119	189	223
RÉ #	-	127	193	227
MI	11	135	196	229
FA	25	141	200	-
FA #	37	148	203	-
SOL	50	154	206	-
SOL #	62	160	209	-
LA	72	165	212	-
LA #	83	171	214	-
SI	93	175	217	-

Exemple: SOUND 102, 10 produit un DO deuxième octave.



L'adressage du clavier et du son se fait en BFFF (soit 49151 en décimal), celui du générateur vidéo allant de BF20 (soit 48928) à BF2F (soit 48943).

Chapitre 5

Messages d'erreur

Message	Signification
BS	Indice incorrect dans un tableau.
CN	Impossible de continuer, en général après une commande CONT.
DD	Erreur avec un tableau.
FC	Écriture incorrecte d'une fonction.
FM	Erreur avec un fichier (cassette).
ID	Instruction interdite en mode direct.
IO	Erreur d'entrée/sortie (avec les périphériques, magnétophone, imprimante).
LS	Chaîne trop longue (plus de 255 caractères).
NF	NEXT sans FOR préalable.
OD	Le pointeur a dépassé la fin des données dans les lignes DATA.
OM	Plus de place disponible en mémoire.
OS	Dépassement de la longueur réservée aux chaînes (normalement 100, mais on peut utiliser CLEAR).
OV	Dépassement de capacité, généralement dans un calcul.
RG	RETURN sans GOSUB préalable ; oubli probable d'un END dans le programme principal.
SN	Erreur de syntaxe, écriture incompréhensible.
ST	Expression de chaîne trop compliquée.
TM	Types incompatibles ; vous avez probablement oublié \$ ou " ".
UL	Numéro de ligne non défini ; après un GOTO ou un GOSUB, il n'y a pas de ligne au numéro demandé.
/0	Division par zéro.

Chapitre 6

Si ça ne marche pas

Si ça ne marche pas... **Que faire ?**

Symptôme	Remède
Le signal OK n'apparaît pas quand vous allumez l'ordinateur.	<ol style="list-style-type: none"> 1. Vérifiez qu'il y a du courant. 2. Vérifiez que l'ordinateur et le téléviseur sont branchés correctement. 3. Allumez le téléviseur avant l'ordinateur puis mettez l'interrupteur d'ALICE en position MARCHE. 4. Vérifiez les raccordements, en particulier le câble allant du micro-ordinateur au téléviseur. 5. Vérifiez que votre téléviseur fonctionne bien. 6. Vérifiez auprès d'un revendeur que votre téléviseur est bien aux normes PÉRITEL. 7. Vérifiez auprès d'un revendeur que le câble de raccordement n'est pas endommagé. 8. Vérifiez auprès d'un revendeur que votre micro-ordinateur n'est pas endommagé.
L'image est de mauvaise qualité.	<ol style="list-style-type: none"> 1. Vérifiez tous les raccordements. 2. Lisez le paragraphe sur l'alimentation électrique ci-dessous. 3. Vérifiez auprès d'un revendeur si votre téléviseur n'est pas endommagé. 4. Vérifiez auprès d'un revendeur si votre téléviseur est bien aux normes PÉRITEL. 5. Vérifiez auprès d'un revendeur si le câble de raccordement n'est pas endommagé.

Si ça ne marche pas... **Que faire ?**

Symptôme	Remède
Un programme sur cassette ne se charge pas.	<ol style="list-style-type: none"> 1. Vérifiez qu'il y a du courant. 2. Vérifiez que l'ordinateur et le magnétophone sont correctement branchés. 3. Allumez le magnétophone avant l'ordinateur. 4. Vérifiez le raccordement du micro-ordinateur au magnétophone. 5. Vérifiez que votre magnétophone fonctionne bien. 6. Modifiez le réglage du volume du magnétophone ; essayez vos propres réglages. 7. Essayez la cassette de sauvegarde ou une autre cassette.
Votre micro-ordinateur ALICE se bloque en cours de fonctionnement normal.	<ol style="list-style-type: none"> 1. Attention aux fluctuations électriques ; lisez ci-dessous. 2. Vérifiez tous les raccordements. 3. Vérifiez qu'on n'a pas appuyé par inadvertance sur le bouton INIT ou sur l'interrupteur. 4. Vérifiez votre programme.

Alimentation électrique

Les ordinateurs sont sensibles aux fluctuations du courant électrique. Mais, sauf cas très particulier, vous n'aurez aucun problème de ce type si vous choisissez soigneusement l'emplacement de votre micro-ordinateur ALICE. En effet, les fluctuations électriques affectant votre micro-ordinateur sont causées soit par la proximité de puissantes machines électriques, soit par celle d'appareils électriques dont l'interrupteur est défectueux. Normalement, il suffit de vous éloigner de ces appareils et de réparer ou remplacer les interrupteurs défectueux. Si cela ne suffisait pas, vous devriez :

- installer un système d'isolation dans les appareils à l'origine de la perturbation, ou
- installer un filtre sur la ligne électrique alimentant l'ordinateur, ou
- installer une ligne électrique propre à l'ordinateur.

Mais rassurez-vous, il est très improbable que vous deviez en arriver là.

Entretien

Votre micro-ordinateur demande très peu d'entretien. Veillez à le conserver propre et à éviter la poussière, en particulier sur le clavier. Nettoyez-le avec un chiffon non pelucheux et très légèrement humide. Les périphériques (magnétophone, imprimante...) demandent en général plus d'entretien. Reportez-vous à leur manuel d'utilisation.

Chapitre 7

Spécifications techniques

Transformateur

Entrée	220 V – 50 Hz – 13 VA
Sortie	10 V – 1,3 A

Composants

Microprocesseur	6803
ROM	16 K
RAM	32 K dont 31 K utilisateur

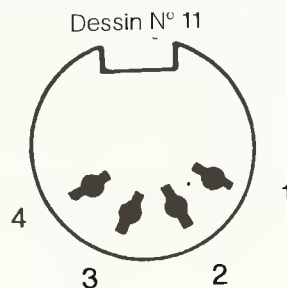
Températures

De fonctionnement	5 à 40 °C
De stockage	- 20 à 70 °C

Taux d'humidité

De fonctionnement	40 % à 80 %
De stockage	20 % à 90 %

Description de la fiche RS-232-C (E/S SÉRIE)



Signal RS-232-C	Broche #
CD Détection de porteuse	1
RD Réception de données	2
GND Terre	3
TD Transmission de données	4

Caractéristiques logicielles d'impression

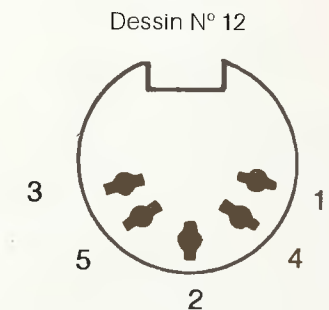
- 600 bauds
- 1 bit de départ
- 8 bits de données
- 2 bits d'arrêt
- Pas de parité
- Largeur d'impression 132 colonnes
- Retour chariot automatique en fin de ligne

Interface magnétophone

- Niveau d'entrée recommandé (lecture) 1 à 5 V crête-à-crête à une impédance minimale de 220 ohms.
- Niveau de sortie typique (écriture) 800 mV crête-à-crête à 1 000 ohms.

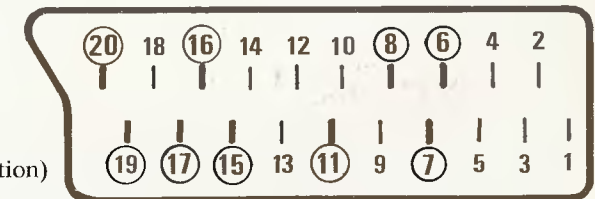
Description de la fiche de magnétophone

- Sortie vers la prise AUX ou MIC du magnétophone.
- Masse du signal
- Entrée de la prise écouteur (EAR) du magnétophone
- Non utilisé
- Non utilisé



Description de l'embase PÉRITEL (sortie ALICE 90)

- 6 Son
- 7 Bleu
- 8 Commutation lente
- 11 Vert
- 15 Rouge
- 16 Commutation rapide
- 17 Masse
- 19 Entrée vidéo (incrustation)
- 20 Sortie vidéo



Dessin N° 13

Chapitre 8

Cahier de programmes

Entrez ces programmes, sans faute de frappe; vous pouvez aussi les améliorer.

Pour découvrir les possibilités d'ALICE :

MUSIC-ALICE
DESSIN → PICTOR
MAGICUBES

Pour jouer :

PENDU
TRONIX
TITANS
SOUS-MARIN
LA GUERRE DES ÉTOILES

Un peu plus sérieux :

BIORYTHMES
REPRÉSENTATION GRAPHIQUE DE FONCTIONS

Vraiment sérieux :

MINI-BASE DE DONNÉES
AMORTISSEMENT LINÉAIRE
GRAPHIQUE HISTOGRAMME

De la Haute Définition :

BATAILLE ORGUE

MUSIC-ALICE

```
1 CLEAR 1000
5 MAX = 100
7 DIM T$(MAX)
10 CLS1 : PRINT@103, "MUSIC-ALICE"
20 PRINT : PRINT : PRINT "A/Z/E/R/T/Y/U.... DO/RE/MI/FA/SOL/LA/SI"
21 PRINT "Q.... MOINS LONG, S.... PLUS LONG"
22 PRINT " ' ' ' ' ' PLUS AIGU"
23 PRINT " ' ' ' ' ' PLUS GRAVE"
25 PRINT : PRINT "POUR ECOUTER : "
26 PRINT "BREAK PUIS TAPÉZ GOTO 300"
33 PRINT : INPUT "TAPÉZ 'ENTER' POUR DÉBUTER"; A$
35 CLS0
40 D = 10
44 GOSUB 50 : I = I + 1
45 IF MAX-I < 1 THEN GOTO 300
46 T$(I) = Z$
```

47 GOTO 44

50 Z\$ = INKEY\$

100 IF Z\$ = "A" THEN N\$ = "DO" : N = 102

110 IF Z\$ = "Z" THEN N\$ = "RE" : N = 119

120 IF Z\$ = "E" THEN N\$ = "MI" : N = 135

130 IF Z\$ = "R" THEN N\$ = "FA" : N = 141

140 IF Z\$ = "T" THEN N\$ = "SOL" : N = 154

145 IF Z\$ = "." THEN N = N + 10

147 IF Z\$ = "," THEN N = N - 10

150 IF Z\$ = "Y" THEN N\$ = "LA" : N = 165

160 IF Z\$ = "U" THEN N\$ = "SI" : N = 175

165 IF Z\$ = "Q" THEN D = D - 1

167 IF Z\$ = "S" THEN D = D + 1

172 IF Z\$ = " " THEN GOTO 50

185 IF N < 1 THEN N = 1

187 IF N > 255 THEN N = 255

190 IF D < 1 THEN D = 1

192 IF D > 255 THEN D = 255

195 PRINT@0, N\$; " DUREE "; D; "NOTE "; N : PRINT@32, "MEMOIRE : "
; MAX - I

200 SOUND N, D

205 SET (RND(63), RND(28) + 2, RND(8))

210 RETURN

300 REM ECOUTE

305 CLS0

310 FOR J = 1 TO I

320 Z\$ = T\$(J) : GOSUB 100

330 NEXTJ

340 CLS1 : END

DESSIN PICTOR.BAS

10 CLS1 : PRINT@106, "DESSIN"

20 PRINT : PRINT "LES FLECHES CONTROLÉNT LE" : PRINT
"CURSEUR"

25 PRINT "0...8 COULEUR DU CRAYON" : PRINT " 'C' COULEUR DU
FOND" : PRINT : INPUT "POUR DÉBUTER TAPÉZ 'ENTER' "; P\$

50 X = 31 : Y = 15

100 CLS0

110 Z\$ = INKEY\$

120 IF Z\$ = "Z" THEN D = - 2

125 IF Z\$ = " " THEN D = 0

130 IF Z\$ = "W" THEN D = 2

140 IF Z\$ = "Q" THEN D = - 1

150 IF Z\$ = "S" THEN D = 1

155 IF Z\$ > "0" AND Z\$ < "9" THEN C = VAL(Z\$)

157 IF Z\$ = "C" THEN GOTO 600

```

160 IF D = 2 THEN Y = Y + 1 : GOTO 197
170 IF D = - 2 THEN Y = Y - 1 : GOTO 197
190 X = X + D
195 IF X < 0 OR X > 63 THEN X = ABS(63 - ABS(X))
197 IF Y < 0 OR Y > 31 THEN Y = ABS(31 - ABS(Y))
200 SET (X,Y,C)
500 GOTO 110
600 REM COULEUR FOND
610 CLS1 : INPUT "COULEUR DU FOND"; BC
620 IF BC < 0 OR BC > 8 THEN 610
630 CLS BC
635 FOR T = 1 TO 1000 : NEXT T
637 C = 1
640 GOTO 200

```

MAGICUBES

```

1 CLS1 : PRINT@104, "M A G I C U B E S"
2 PRINT : PRINT : PRINT : INPUT "TAPEZ 'ENTER' POUR DEBUTER"; AS
5 BC = 0
7 N = 160
10 CLS BC
20 FOR I = 1 TO N : X = RND(62) + 1 : Y = RND(28) : C = RND(8) :
  GOSUB 400 : NEXT
30 Y = 0 : X = RND(50) + 9
31 IF POINT(X,0) <> BC THEN 30
32 C = RND(8)
35 HX = X : HY = Y
40 IF POINT(X,Y + 1) = BC THEN 100
41 SOUND C * 10,1 : SOUND C * 10 - 5,1
50 IF X = 2 OR X = 62 THEN 30
60 IF POINT(X - 1,Y) <> BC AND POINT(X + 1,Y) <> BC THEN 30
70 DX = 1 : IF RND(3) = 1 THEN DX = - 1
75 IF POINT(X + DX,Y) <> BC THEN 70
80 GOTO 300
100 IF Y = 28 THEN 41
105 Y = Y + 1
110 GOSUB 500 : HX = X : HY = Y : GOSUB 400 : GOTO 40
300 X = X + DX : GOSUB 500 : HX = X : HY = Y : GOSUB 400
311 IF Y = 28 THEN 320
315 IF POINT(X,Y + 1) = BC THEN 40
320 IF X = 2 OR X = 62 THEN 30
330 IF POINT(X + DX,Y) <> BC THEN 30
350 GOTO 300
400 SET(X,Y,C) : RETURN
500 RESET(HX,HY) : RETURN

```

PENDU

```

5 CLS0 : CLEAR
X 10 PRINT@0, "P E N D U";
20 FOR I = 1 TO RND(10) : READ AS : NEXT I
B0 LM = LEN(AS) : MS = ""
90 FOR K = 1 TO LM : MS = MS + "-" : NEXT K
100 K = 1 : GOSUB 5000
110 K = LM : GOSUB 5000
X 115 PRINT@15, MS
X 125 PRINT@416, " "; : INPUT "TAPEZ UNE LETTRE"; RS : IF RS = ""
  THEN 125 : IF ASC(RS) < 65 OR ASC(RS) > 90 THEN 125
126 LS = LS + RS
X 127 PRINT@44B, "DEJA TAPEES ."; LS;
128 V = 0
130 FOR K = 2 TO LM - 1
155 CS = MID$(AS, K, 1)
160 IF CS = RS THEN IF MID$(MS, K, 1) = "-" THEN 500
161 REM
180 NEXT K
181 IF V = 1 THEN GOTO 125
185 X = X + 1
190 SOUND 50 + X*15,1
195 ON X GOSUB 220, 230, 240, 250, 260, 270, 280, 290, 300,
  310, 320
X 200 IF X = 10 THEN FOR SS = 1 TO 10 : SOUND SS,2 : NEXT SS :
  PRINT : PRINT : PRINT "P E R D U!!! LE MOT ETAIT .": PRINT :
  PRINT " "; AS : END
210 GOTO 125
215 CLS0
220 SET(10,24,3) : SET(11,23,3) : SET(12,22,3) : SET(13,21,3)
225 RETURN
230 FOR V = 24 TO 10 STEP - 1 : SET(14,V,3) : NEXT V
235 RETURN
240 FOR H = 15 TO 28 : SET(H,10,3) : NEXT H : RETURN
250 FOR V = 10 TO 13 : SET(28,V,6) : NEXT V : RETURN
260 FOR H = 27 TO 29 : SET(H,13,7) : SET(H,14,7) : NEXT H
262 SET (28,15,7)
265 RETURN
270 FOR H = 25 TO 31 : SET(H,16,8) : NEXT H
275 RETURN
280 FOR V = 17 TO 21 : SET(28,V,8) : NEXT V
285 RETURN
290 FOR V = 17 TO 19 : SET(25,V,8) : SET(31,V,8) : NEXT V : RETURN
300 FOR H = 27 TO 29 : SET(H,21,4) : NEXT H : RETURN
310 SET(26,22,4) : SET(30,22,4) : SET(25,23,7) : SET(31,23,7)
312 SOUND 10,2 : SOUND 8,2 : SOUND 6,2 : SOUND 4,2 : SOUND 1,1
315 RETURN

```

```

320 END
390 END
500 SOUND 200,1 : SOUND 210,3
501 V = 1
502 GOSUB 5000
X 505 PRINT@15, M$
510 G = G + 1
X 515 IF G = LM - 2 THEN SOUND 220,2 : SOUND 218,2 : SOUND
240,2 : SOUND 245,1 : PRINT : PRINT : PRINT "VOUS AVEZ
G A G N E!!!!!" : END
520 GOTO 161
5000 M$ = MID$(M$,1,K - 1) + MID$(A$,K,1) + MID$(M$,K + 1,LM
- 1)
5010 RETURN
X 6000 DATA ORDINATEUR, PANTOUFLE, LOGICIEL, TELEVISION, ESPA-
CE, CACTUS, IMPRIMANTE, INITIATION, COULEUR, ALICE

```

TRONIX

```

5 S1 = 0 : S2 = 0
10 CLS1 : PRINT@106, "TRONIX" : PRINT : PRINT : PRINT
11 PRINT "          JAUNE ROUGE"
12 PRINT "DEPLACEMENT VERS : "
13 PRINT "          HAUT      Z      P"
14 PRINT "          BAS        W      ."
15 PRINT "          GAUCHE     Q      L"
16 PRINT "          DROITE     S      M"
19 PRINT : PRINT
20 INPUT "          MUSIQUE (OUI/NON)"; R$
25 IF LEFT$(R$,1) = "O" THEN M = 1
30 XA = 1 : YA = 14 : XB = 1 : YB = 16
40 DA = 1 : DB = DA
55 CLS 0
60 IF S2 = 15 THEN CLS 1 : PRINT@100, "JAUNE A GAGNE" : END
65 IF S1 = 15 THEN CLS 1 : PRINT@100, "ROUGE A GAGNE" : END
90 FOR H = 0 TO 63 : SET(H,0,3) : SET(H,27,3) : NEXT H
100 FOR V = 0 TO 27 : SET(0,V,3) : SET(63,V,3) : NEXT V
110 PRINT@448, "          JAUNE"; S2; "          ROUGE"; S1
150 Z$ = INKEY$
155 D1 = DA
200 IF Z$ = "S" THEN DA = 1
210 IF Z$ = "Q" THEN DA = - 1
220 IF Z$ = "W" THEN DA = 2
230 IF Z$ = "Z" THEN DA = - 2
250 IF DA = - D1 THEN DA = D1
255 IF DA = 2 THEN YA = YA + 1 : GOTO 270
257 IF DA = - 2 THEN YA = YA - 1 : GOTO 270

```

```

260 XA = XA + DA
270 IF POINT(XA,YA) < > 0 THEN S2 = S2 + 1 : SOUND 10,1 : SOUND
5,1 : GOTO 30
275 SET(XA,YA,4)
283 D1 = DB
300 IF Z$ = "M" THEN DB = 1
325 IF Z$ = "L" THEN DB = - 1
330 IF Z$ = "." THEN DB = 2
340 IF Z$ = "P" THEN DB = - 2
343 IF D1 = - DB THEN DB = D1
350 IF DB = 2 THEN YB = YB + 1 : GOTO 370
355 IF DB = - 2 THEN YB = YB - 1 : GOTO 370
360 XB = XB + DB
370 IF POINT(XB,YB) < > 0 THEN S1 = S1 + 1 : SOUND 100,1 :
SOUND 90,1 : GOTO 30
375 SET(XB,YB,2)
390 IF M = 1 THEN SOUND 20,1
400 GOTO 150

```

TITANS

```

1 V = 3
2 CLS1 : PRINT@106, "T I T A N S" : PRINT@136, "=====
====" : PRINT : PRINT : PRINT "UN DUEL ENTRE VOUS ET UN
TITAN" : PRINT "DE L'ESPACE....."
3 PRINT : PRINT "< --- --> ...DEPLACEMENT" : PRINT "L          ..IMMO
BILISATION" : PRINT "ESPACE          ...TIRER"
4 PRINT : PRINT : INPUT "TAPEZ 'ENTER' POUR JOUER"; A$
5 GOTO 50
10 REM RESET ENNEMI
12 RESET(XX,YY) : RESET(XX + 1,YY)
13 RESET(XX + 2,YY)
15 RETURN
50 CLS 0
51 FOR H = 0 TO 63 STEP 5 : SET(H,22,7) : SET(H + 1,22,7) : NEXT H
52 XM = RND(40) + 10 : YM = 8
55 X = 30 : Y = 25
57 X0 = X
100 Z$ = INKEY$
110 IF Z$ = "Q" THEN D = - 1
120 IF Z$ = "S" THEN D = 1
130 IF Z$ = " " THEN GOSUB 280
140 IF Z$ = "L" THEN D = 0
150 X = X + D
152 IF X < 1 THEN X = 1
153 IF X > 60 THEN X = 60
155 RESET(X0,Y) : RESET(X0 + 1,Y) : RESET(X0 + 2,Y) : RESET(X0 +
1,Y - 1)

```

```

160 SET(X,Y,8) : SET(X + 1,Y,8) : SET(X + 2,Y,8) : SET(X + 1,Y -
1,8)
190 X0 = X
200 R = RND(5) - 3 : XM = XM + R
202 XM = XM + 1
205 IF XM < 1 THEN YM = YM - 1 : XM = 1 : XX = XM : YY = YM + 1 :
GOSUB 10
206 IF RND(3) = 1 THEN GOTO 210
207 IF XM > 60 THEN XM = XM - RND(40)
210 IF XM > 60 THEN XM = 60 : YM = YM + 1 : XX = XM : YY = YM -
1 : GOSUB 10
215 IF YM < 8 THEN YM = 8
220 IF YM > 22 THEN GOSUB 500
230 SET(XM,YM,2)
232 SET(XM,YM,2) : SET(XM + 1,YM,2)
233 SET(XM + 2,YM,2)
240 XX = MX : YY = MY : GOSUB 10
260 MX = XM : MY = YM
264 IF X < XM + 8 AND X > XM - 8 AND RND(3) = 1 THEN GOSUB 400
265 PRINT@416, "    SCORE :": SC; "    VAISSEAUX :": V;
267 PRINT "    ";
270 GOTO 100
280 REM TIR
282 SOUND 240,1
285 FOR YT = Y - 2 TO 7 STEP - 1
290 SET(X + 1,YT,4)
295 RESET(X + 1,YT)
300 NEXT YT
310 IF X + 1 = XM OR X + 1 = XM + 1 OR X + 1 = XM + 2 THEN SOUND,
200,2 : SOUND 190,1 : SOUND 205,1 : SOUND 210,2 : SC = SC
+ 10 : YM = YM - 1
320 GOTO 265
400 REM ENNEMI ATTAQUE
405 SOUND 50,1
410 FOR MT = YM + 1 TO 25
415 IF POINT(XM,MT) < >0 THEN GOTO 100
420 SET(XM + 1,MT,6)
425 RESET(XM + 1,MT) : NEXT MT
430 IF XM + 1 = X OR XM + 1 = X + 1 OR XM + 1 = X + 2 THEN GOSUB
500
460 RETURN
500 REM VAISSEAU TOUCHE
510 V = V - 1
515 SOUND 50,1
520 SOUND 30,2 : SOUND 20,3 : SOUND 10,1 : SOUND 1,1
525 IF V = 0 THEN CLS1 : PRINT@100, "T E R M I N E...": SC : PRINT :
PRINT : END
540 GOTO 50

```

SOUS-MARIN

```

5 DIM F(3), Y(3)
10 CLS1 : PRINT@102, "S O U S - M A R I N" : PRINT : PRINT : PRINT
"    SCORE :": SC
12 PRINT : PRINT "5 SOUS-MARINS ENNEMIS SONT A COULER"
13 PRINT "VOUS DISPOSEZ DE 3 BOMBES POUR CHACUN"
14 PRINT "POUR TIRER, APPUYER SUR ESPACE" : PRINT : PRINT
15 INPUT "TAPEZ 'ENTER' POUR JOUER": A$
16 IC = 1
17 CLS0
20 Y(0) = 0 : Y(1) = 0 : Y(2) = 0 : Y(3) = 0
25 F(0) = 0 : F(1) = 0 : F(2) = 0 : F(3) = 0
30 N = 1
50 P = RND(25) + 2
70 FOR I = 1 TO 60
80 SET(I,P,3) : SET(I + 1,P,3)
85 SET(I + 2,P,3) : SET(I + 2,P - 1,3)
87 SET(I + 3,P,3)
90 RESET(I - 1,P) : RESET(I + 1,P - 1)
100 BS = INKEY$
110 IF BS = " " OR N > 3 THEN 140
115 SOUND 250,2
120 F(N) = 1
130 N = N + 1
140 FOR J = 1 TO 3
150 IF F(J) = 0 THEN GOTO 190
160 RESET(J * 16,Y(J))
170 IF Y(J) = 31 THEN GOTO 210
180 Y(J) = Y(J) + 1
190 SET(J * 16,Y(J),2)
200 IF P = Y(J) AND (J * 16 = I OR J * 16 = I + 1 OR J * 16 = I + 2 OR
J * 16 = I + 3) THEN 260
210 NEXT J
220 NEXT I
230 SOUND 10,2 : SOUND 4,2 : SOUND 2,1 : GOTO 266
260 SOUND 200,1 : SOUND 210,2 : SOUND 220,2
265 SC = SC + 40 - J * 10
266 IC = IC + 1
267 IF IC = 6 THEN GOTO 10
268 GOTO 17

```

LA GUERRE DES ÉTOILES

```

5 CLS1 : PRINT@100, "LA GUERRE DES ETOILES" : PRINT : PRINT :
PRINT "VOUS DEVEZ DETRUIRE LES" : PRINT "VAISSEAUX DE L'EM
PIRE !": PRINT : PRINT

```

```

6 PRINT "VOUS CONTROLEZ VOTRE VAISSEAU"
7 PRINT "AVEC LES FLECHES Q Z W S"
9 PRINT : PRINT
10 PRINT "B... TIR LASER"
11 PRINT : PRINT : INPUT "TAPEZ 'ENTER' POUR JOUER"; AS$
25 XM = RND(60) + 2 : YM = RND(28) + 2
27 MX = XM : MY = YM
50 CLS0
60 FOR I = 27 TO 35 : FOR J = 12 TO 18 STEP3
62 SET(I,J,7) : NEXT J : NEXT I
63 FOR J = 12 TO 18 : FOR I = 27 TO 35 STEP4 : SET(I,J,7) : NEXT I :
NEXT J
64 RESET(31,15)
65 SET(26,15,7) : SET(31,11,7) : SET(31,19,7) : SET(36,15,7)
70 SET(4,15,7) : SET(58,15,7)
75 SET(31,3,7) : SET(31,28,7)
100 Z$ = INKEY$
105 IF Z$ = "B" THEN GOSUB 300
110 IF Z$ = "W" THEN D = - 2
120 IF Z$ = "S" THEN D = - 1
130 IF Z$ = "Q" THEN D = 1
140 IF Z$ = "Z" THEN D = 2
150 IF D = - 2 THEN YM = YM + 1 : GOTO 170
160 IF D = 2 THEN YM = YM - 1 : GOTO 170
165 XM = XM - D
170 IF RND(2) = 1 THEN XM = XM + RND(3) - 2 : YM = YM + RND(3) - 2
172 IF XM < 3 OR XM > 59 THEN GOTO 500
173 IF YM < 3 OR YM > 28 GOTO 600
175 M = 1
180 IF M = 1 THEN SET(XM,YM,2) : RESET(MX,MY)
190 IF XM < - 20 THEN XM = - 20
195 IF XM > 85 THEN XM = 85
200 IF YM < - 20 THEN YM = - 20
205 IF YM > 55 THEN YM = 55
210 MX = XM : MY = YM
215 PRINT@0, "SCORE :"; SC;
220 GOTO 60
300 REM TIR
302 J = 31
305 SDUND 200, 1 : SOUND 220,1
310 FOR I = 0 TO 31
312 IF INT(I/2) = I/2 THEN J = J - 1
315 SET(I,J,4) : SET(63 - I,J,4)
320 RESET(I,J) : RESET(63 - I,J)
330 NEXT I
340 IF (XM = 31 AND YM = 15) OR (YM = 16 AND (XM = 30 OR
XM = 32)) THEN SOUND 250,2 : SOUND 240,2 : SOUND 230,2 :
SC = SC + 10 : GOTO 10

```

```

350 RETURN
500 FOR V = 1 TO 31 : RESET(3,V) : RESET(4,V) : RESET(58,V)
501 IF XM < 3 THEN XM = 50
502 RESET(59,V) : NEXT V
503 IF XM > 59 THEN XM = 5
504 GOTO 175
600 FOR H = 0 TO 63 : RESET(H,3) : RESET(H,4) : RESET(H,27) :
RESET(H,28)
602 NEXT H
604 IF YM < 3 THEN YM = 25
605 IF YM > 28 THEN YM = 6
607 GOTO 175
650 PRINT @0, "S C O R E :"; SC;
660 GOTO 25

```

BIORYTHMES

```

5 CLS1
10 PRINT @3, "BIORYTHMES"
11 PRINT : PRINT
20 DATA 1 31,2 28,3 31,4 30,5 31,6 30,7 31,8 31
25 DATA 9 30, 10 31,11 30,12 31
40 DIM M$(24)
45 PI = 3.1415926
46 U = 12
47 V = 19
50 FOR I = 1 TO 12 : READ M$(I) : NEXT
53 FOR I = 13 TO 24 : M$(I) = M$(I - 12) : NEXT
72 B = A/4 : A = A * 365
73 INPUT "VOTRE PRENOM"; PS
74 PRINT "VOTRE DATE D'ANNIVERSAIRE"
75 INPUT "MOIS (EN CHIFFRES)"; AS$
76 INPUT "ET JOUR (1 à 31)"; J
80 PRINT "TAPEZ LE MOIS A ETUDIER" : INPUT "(EN CHIFFRES) :"; ES
110 I = 1
120 IF AS$ = LEFT$(M$(I),LEN(M$(I)) - 3) THEN 180
130 I = I + 1
140 IF I > 24 THEN 170
145 GOTO 120
170 PRINT "RETAPEZ LES DONNEES..." : FOR I = 1 TO 500 : NEXT
171 RUN
180 N = VAL(RIGHT$(M$(I),2)) : C = 0
190 I = I + 1
200 IF I > 24 THEN 170
220 E = VAL(RIGHT$(M$(I),2))
233 C = C + E
240 K = LEN(M$(I)) - 3

```

```

250 IF E$ = LEFT$(M$(I),K) THEN 280
260 GOTO 190
280 C = C + N - J + 2 - E
290 T = A + B + C
300 P0 = T - INT(T/23) * 23
305 A0 = T - INT(T/28) * 28
310 I0 = T - INT(T/33) * 33
450 PRINT
470 PRINT "TAPEZ UNE TOUCHE POUR CONTINUER"
480 X$ = INKEY$: IF X$ = " " THEN 480
490 CLS0
495 FOR I = 1 TO 63 : SET (I,V,5) : NEXT
496 GOTO 600
500 FOR I = 1 TO 2
505 FOR J = 2 TO 4
510 SET (J,V - U/2 + U * (I - 1),4)
515 NEXT J,I
520 SET (3,V - U/2 - 1,4) : SET (3,V - U/2 + 1,4)
530 X$ = INKEY$: IF X$ = " " THEN 530
540 RUN
600 FOR X = 1 TO 31
610 W = U * SIN(2 * PI * (X - 24 + P0)/22)
630 SET (X * 2,V - W,2) : SET (X * 2 + 1, V - W,2)
635 W = U * SIN(2 * PI * (X - 29 + A0)/27)
640 SET (X * 2,V - W,7) : SET (X * 2 + 1, V - W,7)
650 W = U * SIN(2 * PI * (X - 34 + I0)/33)
660 SET (X * 2,V - W,3) : SET(X * 2 + 1, V - W,3)
670 NEXT X
671 PRINT@0, "JAUNE = PHYSIQUE ROSE = AFFECTIF" : PRINT@32,
" BLEU = INTELLECTUEL"
680 PRINT@64, "1..5..9..13..17..21..25..29..31"
700 GOTO 700

```

REPRÉSENTATION GRAPHIQUE DE FONCTIONS

```

5 SOUND 10,1
10 CLS1 : PRINT@64, " REPRESENTATION GRAPHIQUE
DE FONCTIONS F(X)" : PRINT
20 PRINT : PRINT "LA FONCTION F DOIT ETRE RENTREE EN LIGNE 200
F = A*X + B"
22 PRINT : PRINT "ENSUITE TAPEZ 'RUN 30' POUR COMMENCER..."
25 STOP
30 CLS0
35 G = - 1
40 CLS0
60 FOR H = 0 TO 63 : SET(H,15,3) : NEXT H
70 FOR V = 0 TO 31 : SET(31,V,3) : NEXT V

```

```

150 FOR X = - 31 TO 31
200 F = X*X
210 XX = X + 31
220 Y = 15 + F * G
225 IF Y < 0 OR Y > 31 THEN GOTO 240
230 SET(XX,Y,2)
240 NEXT
250 GOTO 250

```

MINI-BASE DE DONNÉES

```

1 SOUND 1,1
2 MF = 30
5 DIM T$(MF,5)
7 N = 1
10 CLS1 : PRINT@105, "M I N I - B A S E"
100 PRINT@160, "1.....CREATION D'UNE FICHE" : PRINT "2.....RECHER
CHE" : PRINT "3.....LISTE COMPLETE" : PRINT "4.....FIN"
110 PRINT : PRINT : INPUT " QUEL CHOIX"; CS
120 IF VAL(C$) < 1 OR VAL(C$) > 4 THEN SOUND 20,1 : GOTO 10
130 ON VAL(C$) GOTO 300,400,500,550
300 REM CREATION FICHE
305 RESTORE
310 CLS1
315 PRINT : PRINT
320 FOR I = 1 TO 5 : READ Z$
330 PRINT Z$; " "; INPUT T$(N,I)
340 NEXT I
345 PRINT : PRINT
350 INPUT "EST-CE CORRECT"; R$: IF LEFT$(R$,1) = "N" THEN 300
355 N = N + 1
360 GOTO 10
400 REM RECHERCHE
410 CLS1 : PRINT " RECHERCHE PAR..."
420 PRINT : PRINT
425 PRINT " 1.....CODE"; PRINT
" 2.....NOM" : PRINT : INPUT " QUEL CHOIX";
R$
430 ON VAL(R$) GOTO 440,470
440 PRINT : PRINT : INPUT " CODE"; CS
445 FOR I = 1 TO N
450 IF T$(I,1) = CS THEN GOSUB 600
455 NEXT I : GOTO 10
470 PRINT : PRINT : INPUT " NOM "; N$
475 FOR I = 1 TO N
480 IF T$(I,2) = N$ THEN GOSUB 600
485 NEXT I : GOTO 10

```



```

499 REM FIN
500 CLS1 : PRINT " LISTE COMPLETE....."
510 FOR I = 1 TO N - 1
515 GOSUB 600
520 NEXT I
540 GOTO 10
550 CLS1 : END
600 REM AFFICHAGE FICHE
605 PRINT : RESTORE
610 SOUND 220,2
620 FOR J = 1 TO 5
630 READ A$
640 PRINT A$ ; " " ; T$(I,J)
650 NEXT J
655 PRINT : INPUT "TAPEZ 'ENTER' " ; Z$
660 RETURN
1000 DATA CODE, NOM, ADRESSE, TEL, DIVERS

```

AMORTISSEMENT LINÉAIRE

```

5 SOUND 100,1
10 CLS1 : PRINT@100, "AMORTISSEMENT LINEAIRE"
20 PRINT : PRINT : PRINT : INPUT "IMMOBILISATIONS " ; P
30 INPUT "DUREE " ; N
40 PRINT@288, " " ; : INPUT "DATE JJ.MM.AA " ; A$
50 IF LEN(A$) < > 8 THEN 40
60 IF VAL(MID$(A$,1,2)) > 31 OR VAL(MID$(A$,4,2)) > 12 THEN 40
70 X = 1
100 CLS1
300 D = 0
305 A = VAL(MID$(A$,7,2))
310 PRINT "ANNEE AMORT. V.N.C"
400 B = 1 - ((VAL(MID$(A$,4,2)) - 1) * 30 - 1 + VAL(MID$(A$,1,2)) / 360
410 FOR I = 1 TO N
430 C = INT(P*B/N)
440 D = D + C
450 PRINT " " ; A ; TAB(16 - LEN(STR$(C))) ; C ; TAB(27 - LEN(STR$(P - D))) ; P - D
470 A = A + 1 : B = 1
480 NEXT
500 PRINT " " ; A ; TAB(16 - LEN(STR$(P - D))) ; P - D ; TAB(26) ; "0"
510 SOUND 200,1

```

GRAPHIQUE HISTOGRAMME

```

2 DIM T(300) : DIM M(10)
3 CLS1 : C = 1
4 PRINT@105, "HISTOGRAMME" : PRINT : PRINT
5 PRINT "VOUS POUVEZ ENTRER AUTANT"
6 PRINT "DE NOMBRES QUE VOUS VOULEZ"
7 PRINT "MAIS PAS PLUS DE"
8 PRINT "10 VALEURS DISTINCTES"
9 PRINT : PRINT : PRINT
10 INPUT "NOMBRE D'ELEMENTS " ; N
11 IF N < 2 THEN PRINT "DOIT ETRE > 1"
12 IF N < 2 THEN GOTO 10
13 U = 480 : NN = N : C = RND(8)
20 FOR I = 1 TO N : PRINT "T( " ; I ; " ) " ; : INPUT T(I)
30 NEXT I
35 FOR I = 1 TO N - 1
40 FOR J = I + 1 TO N
50 IF T(I) < T(J) THEN 70
60 A = T(I) : T(I) = T(J) : T(J) = A
70 NEXT J : NEXT I
105 X = T(1)
106 J = 1
107 M(J) = 1
110 FOR I = 2 TO N
115 IF T(I) = X THEN M(J) = M(J) + 1 : GOTO 119
117 X = T(I) : IF J = 10 GOTO 250
118 J = J + 1 : M(J) = 1
119 NEXT
120 N = J
121 FOR I = 1 TO N
122 M = M + M(I)
123 NEXT
125 X = 0
140 CLS0
145 U1 = INT(31/N)
150 FOR I = 1 TO N
160 FOR J = 0 TO 2*U1 - 1
170 FOR V = 27 TO 27 - INT (24*M(I)/M) STEP - 1
180 SET(X + J,V,C)
190 NEXT V
200 NEXT J
205 GOSUB 300
207 X = X + 2*U1 : UU = U
210 NEXT I
215 B$ = " " ; FOR I = 1 TO U1 : B$ = B$ + " " : NEXT I
216 X = - 1E6
217 FOR I = 1 TO NN

```

```

218 IF T(I) = X THEN GOTO 230
220 X = T(I) : A$ = RIGHT$ (B$ + STR$(X), U1)
225 PRINT@UU,A$ : UU = UU + U1
230 NEXT I
241 X$ = INKEY$ : IF X$ = " " THEN 241
242 M = 0 : GOTO 3
250 PRINT "PLUS DE 10"
251 PRINT "VALEURS DISTINCTES"
252 PRINT "RETAPEZ LES DONNEES"
253 GOTO 20
300 C = C + 1 : IF C > 8 THEN C = 1
305 RETURN

```

BATAILLE

Pour exécuter ce programme, tapez CLS 81, puis RUN.

```

1000 DIM L(10,10),G(10),R(10)
1002 RESTORE:FOR I=1 TO 10:X1=RND(10):Y1=RND(10):
L(X1,Y1)=1:NEXT I
1005 CLS:SET X=0,6,6
1006 PRINT@26,"B A T A I L L E   N A V A L E"
1010 FORX=29 TO 129 STEP 10:FORY=11 TO 120:SET(X,Y,2):NEXTY,X
1020 FORY=20 TO 120 STEP 10:FORX=20 TO 129:SET(X,Y,2):NEXTX,Y
1029 REM 'A'
1030 X=22:FORY=23 TO 28:SET(X,Y,2):SET(X+4,Y,2):NEXTY:Y=22:
FORX=23 TO 25=SET(X,Y,2):SET(X,Y+4,2):NEXTX
1039 REM 'B'
1040 X=22:FORY=32 TO 38:SET(X,Y,2):NEXTY:Y=32:FORX=23 TO 25:
SET(X,Y,2):SET(X,Y+3,2):SET(X,Y+6,2):NEXTX:X=26:
FORY=33 TO 34:SET(X,Y,2)
1045 SET(X,Y+3,2):NEXTY
1049 REM 'C'
1050 X=22:FORY=43 TO 47:SET(X,Y,2):NEXTY:Y=42:FORX=23 TO 25:
SET(X,Y,2):SET(X,Y+6,2):NEXTX:Y=43:X=26:SET(X,Y,2):
SET(X,Y+4,2)
1059 REM 'D'
1060 X=22:FORY=52 TO 58:SET(X,Y,2):NEXTY:Y=52:FORX=23 TO 25:
SET(X,Y,2):SET(X,Y+6,2):NEXTX:X=26:FORY=53 TO 57:SET(X,Y,2):
NEXTY
1069 REM 'E'
1070 X=22:FORY=62 TO 68:SET(X,Y,2):NEXTY:Y=62:FORX=23 TO 26:
SET(X,Y,2):SET(X,Y+6,2):NEXTX:Y=65:FORX=23 TO 24:SET(X,Y,2):
NEXTX

```

```

1079 REM 'F'
1080 X=22:FORY=72 TO 78:SET(X,Y,2):NEXTY:Y=72:FORX=23 TO 26:
SET(X,Y,2):NEXTX:Y=75:FORX=23 TO 24:SET(X,Y,2):NEXTX
1089 REM 'G'
1090 X=22:FORY=83 TO 87:SET(X,Y,2):NEXTY:Y=82:FORX=23 TO 25:
SET(X,Y,2):SET(X,Y+6,2):NEXTX:X=26:FORY=86 TO 88:SET(X,Y,2):
NEXTY
1095 SET(26,83,2):SET(25,86,2)
1099 REM 'H'
1100 X=22:FORY=92 TO 98:SET(X,Y,2):SET(X+4,Y,2):NEXTY:Y=95:
FORX=23 TO 25:SET(X,Y,2):NEXTX
1109 REM 'I'
1110 Y=102:FORX=23 TO 25:SET(X,Y,2):SET(X,Y+6,2):NEXTX:X=24:
FORY=103 TO 107:SET(X,Y,2):NEXTY
1119 REM 'J'
1120 Y=112:FORX=24 TO 26:SET(X,Y,2):NEXTX:X=25:
FORY=113 TO 117:SET(X,Y,2):NEXTY:SET(24,118,2):
SET(23,118,2):SET(22,117,2)
1129 REM '1'
1130 X=34:FORY=12 TO 18:SET(X,Y,2):NEXTY:SET(33,13,2)
1139 REM '2'
1140 Y=18:FORX=42 TO 46:SET(X,Y,2):NEXTX:Y=12:FORX=43 TO 45:
SET(X,Y,2):NEXTX:X=46:FORY=13 TO 14:SET(X,Y,2):NEXTY:Y=15:
FORX=44 TO 45
1145 SET(X,Y,2):NEXTX:SET(42,13,2):SET(43,16,2):SET(42,17,2)
1149 REM '3'
1150 Y=12:FORX=52 TO 55:SET(X,Y,2):NEXTX:X=56:FORY=12 TO 13:
SET(X,Y,2):SET(X-1,Y+2,2):SET(X,Y+4,2):NEXTY:Y=18:
FORX=53 TO 55
1155 SET(X,Y,2):NEXTX:SET(54,15,2):SET(52,17,2)
1159 REM '4'
1160 X=65:FORY=12 TO 18:SET(X,Y,2):NEXTY:Y=16:FORX=62 TO 66:
SET(X,Y,2):NEXTX:SET(64,13,2):SET(63,14,2):SET(62,15,2)
1169 REM '5'
1170 Y=12:FORX=73 TO 76:SET(X,Y,2):SET(X-1,Y+2,2):NEXTX:X=72:
FORY=12 TO 13:SET(X,Y,2):NEXTY:X=76:FORY=15 TO 17:
SET(X,Y,2):NEXTY
1175 Y=18:FORX=73 TO 75:SET(X,Y,2):NEXTX:SET(72,17,2)
1179 REM '6'
1180 Y=15:FORX=83 TO 85:SET(X,Y,2):SET(X,Y+3,2):NEXTX:X=82:
FORY=14 TO 17:SET(X,Y,2):NEXTY:X=86:FORY=16 TO 17:
SET(X,Y,2):NEXTY
1185 SET(85,12,2):SET(84,12,2):SET(83,13,2):SET(82,14,2)
1189 REM '7'
1190 Y=12:FORX=92 TO 96:SET(X,Y,2):NEXTX:X=93:FORY=16 TO 18:
SET(X,Y,2):NEXTY:SET(96,13,2):SET(95,14,2):SET(94,15,2)
1199 REM '8'

```

```

1200 Y=12:FORX=103TO105:SET(X,Y,2):SET(X,Y+3,2):SET(X,Y+6,2):
NEXTX:X=102:FORY=13TO14:SET(X,Y,2):SET(X+4,Y,2):
SET(X,Y+3,2)
1205 SET(X+4,Y+3,2):NEXTY
1209 REM '9'
1210 Y=12:FORX=113TO115:SET(X,Y,2):SET(X,Y+3,2):SET(X,Y+6,2):
NEXTX:X=112:FORY=13TO14:SET(X,Y,2):SET(X+4,Y,2):
SET(X+4,Y+3,2)
1215 NEXTY
1219 REM '0'
1220 X=122:FORY=14TO16:SET(X,Y,2):SET(X+4,Y,2):NEXTY:
Y=12:X=124:SET(X,Y,2):SET(X-1,Y+1,2):SET(X+1,Y+1,2):
SET(X,Y+6,2)
1221 SET(X-1,Y+5,2):SET(X+1,Y+5,2)
1222 DATA1,31,41,51,61,71,81,91,101,111
1223 FORI=1TO10:READA:Q(I)=A:NEXTI
1224 DATA30,40,50,60,70,80,90,100,110,120
1225 FORI=1TO10:READA:R(I)=A:NEXTI
1228 REM programme principal
1229 PRINT@ 706,"VAISS.COULES";Z;
1230 PRINT@ 388,":INPUT"LETTRE ";A$
1240 PRINT@ 468,":INPUT"NOMBRE ";N
1250 H=ASC(A$)-64:IFN=0THEN1255
1253 M=M+1:IFM=90THEN10050
1254 GOTO1300
1255 N=10
1300 IFL(H,N)=1THEN10000
1301 X=RN():Y=Q(H):SET(X,Y,2):SET(X+8,Y,2):SET(X+1,Y+1,2):
SET(X+7,Y+1,2):SET(X+2,Y+2,2):SET(X+6,Y+2,2):
SET(X+3,Y+3,2)
1305 SET(X+5,Y+3,2):SET(X+4,Y+4,2):SET(X+3,Y+5,2):
SET(X+5,Y+5,2):SET(X+2,Y+6,2):SET(X+6,Y+6,2):
SET(X+1,Y+7,2):SET(X+7,Y+7,2)
1306 SET(X,Y+8,2):SET(X+8,Y+8,2)
1310 GOTO1230
10000 L(H,N)=0:FORX=1TO20:SET(RND(8)+R(N),RND(8)+Q(H),2):
NEXTX
10010 SOUND102,4:SOUND141,4:SOUND165,4:SOUND180,6:
SOUND165,4:SOUND180,10
10030 Z=Z+1:IFZ=10THEN10080
10040 GOTO1229
10050 CLS:PRINT"Vous n'avez pas gagne.";
10060 IF INKEY$="" THEN 10060
10070 GOTO1002
10080 CLS:PRINT"Felicitations!!! Vous avez gagne!!!";
10100 IF INKEY$="" THEN 10100
10110 GOTO1002

```

ORGUE

Pour exécuter ce programme, tapez CLS 81, puis RUN.

```

40 SET#0,1,2
50 PRINT@ 26,"A L I C E P R E S E N T E . . . ";
51 REM Page d'introduction
52 REM 'O'
60 Y=16:FORX=35TO46:SET(X,Y,1):SET(X,Y+1,1):SET(X,Y+33,1):
SET(X,Y+34,1):NEXTX
70 X=29:FORY=22TO44:SET(X,Y,1):SET(X+1,Y,1):SET(X+22,Y,1):
SET(X+23,Y,1):NEXTY
80 X=33:FORY=18TO19:SET(X,Y,1):SET(X+1,Y,1):SET(X+14,Y,1):
SET(X+15,Y,1)
85 SET(X,Y+29,1):SET(X+1,Y+29,1):SET(X+14,Y+29,1):
SET(X+15,Y+29,1):NEXTY
90 X=31:FORY=20TO21:SET(X,Y,1):SET(X+1,Y,1):SET(X+18,Y,1):
SET(X+19,Y,1)
95 SET(X,Y+25,1):SET(X+1,Y+25,1):SET(X+18,Y+25,1):
SET(X+19,Y+25,1):NEXTY
99 REM 'r'
100 X=55:FORY=31TO50:SET(X,Y,1):SET(X+1,Y,1):NEXTY
110 X=57:FORY=33TO34:SET(X,Y,1):SET(X+1,Y,1):SET(X+9,Y,1):
SET(X+10,Y,1):NEXTY
120 Y=31:FORX=59TO65:SET(X,Y,1):SET(X,Y+1,1):NEXTX
121 REM 'g'
130 X=70:FORY=35TO46:SET(X,Y,1):SET(X+1,Y,1):NEXTY
140 X=84:FORY=31TO60:SET(X,Y,1):SET(X+1,Y,1):NEXTY
150 Y=31:FORX=74TO81:SET(X,Y,1):SET(X,Y+1,1):SET(X,Y+18,1):
SET(X,Y+19,1):NEXTX
160 X=72:FORY=33TO34:SET(X,Y,1):SET(X+1,Y,1):SET(X+10,Y,1):
SET(X+11,Y,1)
165 SET(X,Y+14,1):SET(X+1,Y+14,1):SET(X+10,Y+14,1):
SET(X+11,Y+14,1):NEXTY
170 X=70:FORY=61TO62:SET(X,Y,1):SET(X+1,Y,1):SET(X+12,Y,1):
SET(X+13,Y,1):NEXTY
180 Y=63:FORX=72TO81:SET(X,Y,1):SET(X,Y+1,1):NEXTX
181 REM 'i'
190 X=88:FORY=31TO46:SET(X,Y,1):SET(X+1,Y,1):NEXTY
200 X=105:FORY=31TO50:SET(X,Y,1):SET(X+1,Y,1):NEXTY
210 X=90:FORY=47TO48:SET(X,Y,1):SET(X+1,Y,1):SET(X+13,Y,1):
SET(X+14,Y,1):NEXTY
220 Y=49:FORX=92TO102:SET(X,Y,1):SET(X,Y+1,1):NEXTX
221 REM 'e'
230 X=109:FORY=35TO46:SET(X,Y,1):SET(X+1,Y,1):NEXTY
240 Y=31:FORX=113TO122:SET(X,Y,1):SET(X,Y+1,1):NEXTX

```

```

250 X=125:FORY=35TO40:SET(X,Y,1):SET(X+1,Y,1):NEXTY
260 Y=39:FORX=111TO124:SET(X,Y,1):SET(X,Y+1,1):NEXTX
270 Y=49:FORX=113TO126:SET(X,Y,1):SET(X,Y+1,1):NEXTX
280 X=111:FORY=33TO34:SET(X,Y,1):SET(X+1,Y,1):SET(X+2,Y,1):
  SET(X+3,Y,1):SET(X,Y+14,1):SET(X+1,Y+14,1):NEXTY
281 REM note
290 X=130:FORY=9TO22:SET(X,Y,2):NEXTY
300 X=139:FORY=4TO19:SET(X,Y,2):NEXTY
310 FORX=125TO129:FORY=21TO23:SET(X,Y,2):SET(X+9,Y-3,2):
  NEXTY,X
320 X=127:Y=20:SET(X,Y,2):SET(X+1,Y,2):SET(X+9,Y-3,2):
  SET(X+10,Y-3,2)
330 X=124:Y=22:SET(X,Y,2):SET(X,Y+1,2):SET(X+9,Y-3,2):
  SET(X+9,Y-2,2)
340 Y=24:FORX=125TO128:SET(X,Y,2):SET(X+9,Y-3,2):NEXTX
350 Y=5:FORX=137TO138:SET(X,Y,2):SET(X-2,Y+1,2):
  SET(X-4,Y+2,2):SET(X-6,Y+3,2)
360 SET(X,Y+2,2):SET(X-2,Y+3,2):SET(X-4,Y+4,2):SET(X-6,Y+5,2):
  NEXTX
365 PRINT@ 1233,"Electronique";
366 PRINT@ 1978,"Ecrit par Guy Johnson";
367 REM Musique
368 X=0
369 IFX=2THENSOUND102,14:GOTO500
370 FORI=1TO33:READN,L:SOUNDN,L:NEXTI
380 DATA 102,12,119,4,135,8,135,8,119,4,102,4,119,4,135,4,102,8,
  50,8,102,12,119,4,135,8,135,8,119,4,102,4,119,4,135,4,102,
  14,102
385 DATA4,102,4,102,4,135,4,154,4,154,4,154,8,154,4,165,4,154,4,
  141,4,135,4,119,4,102,8
390 X=X+1:FORI=1TOX:SOUND154,4:SOUND154,4:SOUND154,8:NEXTI
  SOUND50,4:SOUND50,4:SOUND135,12:SOUND119,12:
400 SOUND154,12:SOUND141,12:SOUND135,12:SOUND119,12:
  RESTORE:GOTO369
500 PRINT@ 1920,"Tapez ENTER pour continuer";
510 A$=INKEY$:IFA$=CHR$(13)THEN519
515 GOTO510
518 REM clavier
519 CLS:SET*0,5,4
520 X=8:FORY=0TO1:SET(X,Y,2):SET(X+1,Y,2):SET(X-2,Y+2,2):
  SET(X-1,Y+2,2):SET(X+2,Y+2,2):SET(X+3,Y+2,2):SET(X-4,Y+4,2)
  SET(X-1,Y+2,2):SET(X+2,Y+4,2):SET(X+5,Y+4,2):SET(X-6,Y+6,2):
525 SET(X-3,Y+4,2):SET(X+4,Y+4,2):SET(X+5,Y+4,2):SET(X-6,Y+6,2):
  SET(X-5,Y+6,2):SET(X+6,Y+6,2):SET(X+7,Y+6,2):NEXTY
530 X=0:FORY=8TO21:SET(X,Y,2):SET(X+1,Y,2):SET(X+16,Y,2):
  SET(X+17,Y,2):NEXTY
540 Y=14:FORX=2TO15:SET(X,Y,2):SET(X,Y+1,2):NEXTX
550 X=0:FORY=25TO44:SET(X,Y,2):SET(X+1,Y,2):NEXTY
560 Y=45:FORX=0TO17:SET(X,Y,2):SET(X,Y+1,2):NEXTX

```

```

570 Y=50:FORX=0TO17:SET(X,Y,2):SET(X,Y+1,2):SET(X,Y+20,2):
  SET(X,Y+21,2):NEXTX
580 X=8:FORY=52TO70:SET(X,Y,2):SET(X+1,Y,2):NEXTY
590 Y=75:FORX=0TO17:SET(X,Y,2):SET(X,Y+1,2):SET(X,Y+20,2):
  SET(X,Y+21,2):NEXTX
600 X=0:FORY=77TO95:SET(X,Y,2):SET(X+1,Y,2):NEXTY
610 Y=100:FORX=0TO17:SET(X,Y,2):SET(X,Y+1,2):SET(X,Y+20,2):
  SET(X,Y+21,2):NEXTX
620 X=0:FORY=102TO121:SET(X,Y,2):SET(X+1,Y,2):NEXTY
630 Y=110:FORX=2TO13:SET(X,Y,2):SET(X,Y+1,2):NEXTX
635 PRINT@ 1930,"MATRA ET HACHETTE";
640 FORY=0TO29:FORX=34TO144:SET(X,Y,1):NEXTX,Y
650 FORX=34TO144STEP10:FORY=0TO29:RESET(X,Y):NEXTY,X
660 FORS=0TO20STEP10:FORX=41TO47:FORY=0TO19:
  SET(X+S,Y,0):NEXTY,X,S
670 FORS=0TO10STEP10:FORX=81TO87:FORY=0TO19:
  SET(X+S,Y,0):NEXTY,X,S
680 FORS=0TO20STEP10:FORX=111TO117:FORY=0TO19:
  SET(X+S,Y,0):NEXTY,X,S
690 FORX=146TO149:FORY=0TO19:SET(X,Y,0):NEXTY,X
700 FORY=50TO79:FORX=39TO139:SET(X,Y,1):NEXTX,Y
710 FORX=39TO139STEP10:FORY=50TO79:RESET(X,Y):NEXTY,X
730 FORS=0TO10STEP10:FORX=46TO52:FORY=50TO69:
  SET(X+S,Y,0):NEXTY,X,S
740 FORS=0TO20STEP10:FORX=76TO82:FORY=50TO69:
  SET(X+S,Y,0):NEXTY,X,S
750 FORS=0TO10STEP10:FORX=116TO122:FORY=50TO69:
  SET(X+S,Y,0):NEXTY,X,S
760 S=338:PRINT@ S," A ";;PRINT@ S+5," Z ";;PRINT@ S+10," E ";;
  PRINT@ S+15," R ";;PRINT@ S+20," T ";;PRINT@ S+25," Y ";;
765 PRINT@ S+30," U ";;PRINT@ S+35," I ";;PRINT@ S+40," O ";;
  PRINT@ S+45," P ";;PRINT@ S+50," ";
770 S=181:PRINT@ S," 2 ";;PRINT@ S+5," 3 ";;PRINT@ S+10," 4 ";;
  PRINT@ S+20," 6 ";;PRINT@ S+25," 7 ";;PRINT@ S+35," 9 ";;
775 PRINT@ S+40," 0 ";;PRINT@ S+45," : ";;
780 S=1141:PRINT@ S," W ";;PRINT@ S+5," X ";;PRINT@ S+10," C ";;
  PRINT@ S+15," V ";;PRINT@ S+20," B ";;PRINT@ S+25," N ";;
785 PRINT@ S+30," / ";;PRINT@ S+35," . ";;PRINT@ S+40," . ";;
  PRINT@ S+45," ; ";;
790 S=984:PRINT@ S,"S";;PRINT@ S+5,"D";;PRINT@ S+15,"G";;
  PRINT@ S+20,"H";;PRINT@ S+25,"J";;PRINT@ S+35,"L";;
  PRINT@ S+40,"M";
800 A$=INKEY$
810 IFA$="W"THEN1000
820 IFA$="S"THEN1010
830 IFA$="X"THEN1020
840 IFA$="D"THEN1030
850 IFA$="C"THEN1040

```

860 IFA\$="V"THEN1050
870 IFA\$="G"THEN1060
880 IFA\$="B"THEN1070
890 IFA\$="H"THEN1080
900 IFA\$="N"THEN1090
905 IFA\$="J"THEN1100
910 IFA\$="/"THEN1110
915 IFA\$=";"THEN1120
920 IFA\$="L"THEN1130
925 IFA\$=","THEN1140
930 IFA\$="M"THEN1150
935 IFA\$=":"THEN1160
940 IFA\$="A"THEN1170
945 IFA\$="2"THEN1180
950 IFA\$="Z"THEN1190
955 IFA\$="3"THEN1200
960 IFA\$="E"THEN1210
965 IFA\$="4"THEN1220
970 IFA\$="R"THEN1230
975 IFA\$="T"THEN1240
980 IFA\$="6"THEN1250
981 IFA\$="Y"THEN1260
982 IFA\$="7"THEN1270
983 IFA\$="U"THEN1280
984 IFA\$="I"THEN1290
985 IFA\$="9"THEN1300
986 IFA\$="O"THEN1310
987 IFA\$="0"THEN1320
988 IFA\$="P"THEN1330
989 IFA\$="."THEN1340
990 IFA\$="@ "THEN1350
991 IFA\$="1"THEN10
995 GOTO800
1000 SOUND1,4.5:GOTO800
1010 SOUND6,4.5:GOTO800
1020 SOUND7,4.5:GOTO800
1030 SOUND9,4.5:GOTO800
1040 SOUND11,4.5:GOTO800
1050 SOUND25,4.5:GOTO800
1060 SOUND37,4.5:GOTO800
1070 SOUND50,4.5:GOTO800
1080 SOUND62,4.5:GOTO800
1090 SOUND72,4.5:GOTO800
1100 SOUND83,4.5:GOTO800
1110 SOUND93,4.5:GOTO800
1120 SOUND102,4.5:GOTO800
1130 SOUND111,4.5:GOTO800
1140 SOUND119,4.5:GOTO800

1150 SOUND127,4.5:GOTO800
1160 SOUND135,4.5:GOTO800
1170 SOUND141,4.5:GOTO800
1180 SOUND148,4.5:GOTO800
1190 SOUND154,4.5:GOTO800
1200 SOUND160,4.5:GOTO800
1210 SOUND165,4.5:GOTO800
1220 SOUND171,4.5:GOTO800
1230 SOUND175,4.5:GOTO800
1240 SOUND180,4.5:GOTO800
1250 SOUND185,4.5:GOTO800
1260 SOUND189,4.5:GOTO800
1270 SOUND193,4.5:GOTO800
1280 SOUND196,4.5:GOTO800
1290 SOUND200,4.5:GOTO800
1300 SOUND203,4.5:GOTO800
1310 SOUND206,4.5:GOTO800
1320 SOUND209,4.5:GOTO800
1330 SOUND212,4.5:GOTO800
1340 SOUND214,4.5:GOTO800
1350 SOUND217,4.5:GOTO800

Petit lexique anglais-français

ABS (absolute)	Valeur absolue
Access	Accès
AND	Et
BREAK	Casser, interrompre
CHR (Character)	Caractère
CLEAR	Nettoyer
CLOAD (cassette load)	Charger depuis la cassette
CLS (Clear screen)	Nettoyer l'écran
CONT (continue)	Continuer
CONTROL	Contrôler
COS (Cosine)	Cosinus
CSAVE (cassette save)	Sauver sur cassette
DATA	Données
DIM (dimension)	Dimension
END	Fin
ENTER	Entrer
EXP (exponentiation)	Exponentiation
FOR	Pour
GOSUB	Aller (dans la sous-routine)
GOTO	Aller à
IF	Si
INIT (initialize)	Initialiser
INKEY	Entrée à partir des touches du clavier (KEYS)
INPUT	Entrée
INT (integer)	Entier
LEFT	Gauche
LEN (length)	Longueur
LET	Faire, donner
LIST	Lister
MID (middle)	Milieu
Memory	Mémoire
NEXT	Prochain, suivant
NEW	Nouveau
NOT	Non

OK	D'accord
ON	En fonction de (dans ce contexte)
ONLY	Seulement
OR	Ou
PEEK	Regarder
POINT	Point
POKE	Mettre (quelque chose quelque part)
PRINT	Imprimer
READ	Lire
REDO	Refaire, recommencer
REM (remark)	Remarque
RESET	Remettre
RESTORE	Remettre en état
RETURN	Revenir
RIGHT	Droite
RND (random)	Hasard
RUN	Courir, exécuter
SET	Mettre (dans un certain état)
SGN (sign)	Signe
SHIFT	Déplacer
SIN (sine)	Sinus
SOUND	Son
SQR (square root)	Racine carrée
STEP	Pas
STOP	Arrêt
TAB (tabulation)	Tabulation
TAN (tangent)	Tangente
THEN	Alors
VAL (value)	Valeur

Index

A

ABS (fonction) 65
Affectation (définition et exemple) 47, 51
Aléatoire (tirage) v. RND
Algorithme 137
Allumer (un point) v. SET
AND (opérateur logique) 94
Animation (graphique) 112, 122
ASC (fonction) 69
ASCII (code) 68, 168

B

BASIC 161
Binaire 159
Bit 159
Boucles imbriquées 85
Branchements conditionnels 97
Branchements (matériel, TV, imprimante) 11
BREAK (touche) 22

C

Chaîne
définition 44, 49
fonctions sur... 69
espace réservé pour les... 133
CHR\$ (fonction) 69
Clavier (description) 19
CLEAR (instruction) 133, 143
CLOAD (instruction) 74
CLOADM (instruction) 144
CLOAD* (instruction) 139
CLS (commande) 42
Code ASCII 68, 168
Concaténation 57
Conditions 88
Constante (définition) 47
CONT (commande) 146
CONTROL (touche) 20
Conversion
d'une chaîne en numérique : v. VAL
d'une valeur en chaîne : v. STR\$
d'angles en radians 66
Corriger une ligne 26, 37
COS (fonction) 66
Couleur (graphique) 111, 119, 121
(semi-graphique) 109
(code) 42, 170

CSAVE (commande) 74
CSAVE* (commande) 138

D

DATA (instruction) 100
Déclarer (un tableau) 132
Décrémenter 85
Degré (conversion en radians) 66
DIM (instruction) 132
Dimension (d'un tableau) 141
Dimensionner (un tableau) v. déclarer
Donnée
définition 160
stockage v. DATA et tableau
traitement 63

E

Ecran (types) 40, 110, 119
Effacer
l'écran 22, 42
une ligne 37
le contenu de la mémoire v. NEW
END (instruction) 92
ENTER (touche de validation) 22
Entière (partie) v. INT
ET (opérateur logique) v. AND
Eteindre (un point graphique) v. RESET
EXEC 143
EXP (fonction) 67
E + } v. notation scientifique
E - }

F

Fonctions (définitions) 63
FOR... NEXT 86

G

GOSUB 124
GOTO 79
Graphiques (points et instructions) 109

H

Hasard v. RND

I

Identificateur (définition) 46

IF... THEN (instruction) 92
Imprimante v. LPRINT et LLIST
Incrémenter 82
Indice 130
Informatique (définition) 158
INIT (bouton) 38, 60, 146
Initialisation 48
INKEY\$ (fonction) 108
INPUT (instruction) 54, 60
INT (fonction) 64
Interruption volontaire
v. BREAK,
v. INIT
Inversion vidéo 21, 68
Itération 77

K

K-octets ou kilo-octets 160

L

LEFT\$ (fonction) 71
LEN (fonction) 69
LET (instruction) 49
Ligne logique (définition) 39
LIST (commande) 38
LLIST (commande) 38
LOG (fonction) 67
LPRINT (commande) 60

M

MEM (fonction) 133
Mémoire 160
Message d'erreur 26, 173
Micro-ordinateur (définition) 158
MID\$ (fonction) 71
Minuscules 21, 41
Minuscules (comment les obtenir à l'imprimante) 21, 69
Mise au point de programmes 145

N

NEXT v. FOR... NEXT
NEW 43
Nom
d'identificateur 49
de programme 74
Nombres (représentation) 57
NOT (opérateur logique) 94
Notation scientifique 57

O

Octet 159
OK (suppression du message) 108
ON... GOSUB (instruction) 129
ON... GOTO (instruction) 99, 127
Opérateurs
arithmétiques 55
de relation 89
logiques 94
OR (opérateur logique) 94
Organigramme (définition) 77
Ou (opérateur logique) v. OR

P

Paramètre (définition) 126
Partie entière v. INT
Pas (d'incrémenter) v. STEP
PEEK (instruction) 142
Périphériques 161
Place en mémoire v. MEM
Plantage 38, 60
POINT (instruction) 115, 123
POKE (instruction) 143
PRINT (instruction) 41, 58, 60
Programme (définition) 27

R

Racine carrée v. SQR
Radians 66
RAM (Random Access Memory) 160
READ (instruction) 101
Recherche d'un programme enregistré 74
REDO (message) 53
REM (instruction) 96
Répéter jusqu'à (structure) 97
Répétition v. itération
RESET (instruction) 112, 122
RESTORE (instruction) 107
RETURN (instruction) 124
RIGHT\$ (fonction) 71
RND (fonction)
description 64
usage 92
ROM (Read Only Memory) 160
RUN (commande) 27, 38

S

Sauvegarde sur cassette
d'un programme 73
d'un tableau 138

Semi-graphique 109
 Séparateur
 d'instructions 39
 de données 58
 Séquence (définition) 77
 SET (instruction) 111, 119
 SET* (instruction) 121
 SGN (fonction) 66
 SHIFT (touche) 21
 SIN (fonction)
 description 66
 utilisation 112
 Sinon (traduction en BASIC) 91
 SOUND (instruction)
 description 61
 utilisation 82
 Sous-programme 124
 SQR (fonction) 66
 STEP (instruction) 86
 STOP (instruction) 146
 STR\$ (fonction) 71
 Supprimer une ligne 37


T

TAB (instruction avec PRINT) 41
 Tableau
 définition 130
 à plusieurs dimensions 141
 Tabulation 58
 TAN (fonction) 66
 Tant que... (structure conditionnelle
 itérative) 97
 Temporisation 86, 125
 THEN v. IF
 Tri 136

V

VAL (fonction) 71
 Variable (définition) 47
 Variable indiquée 130

NOTES

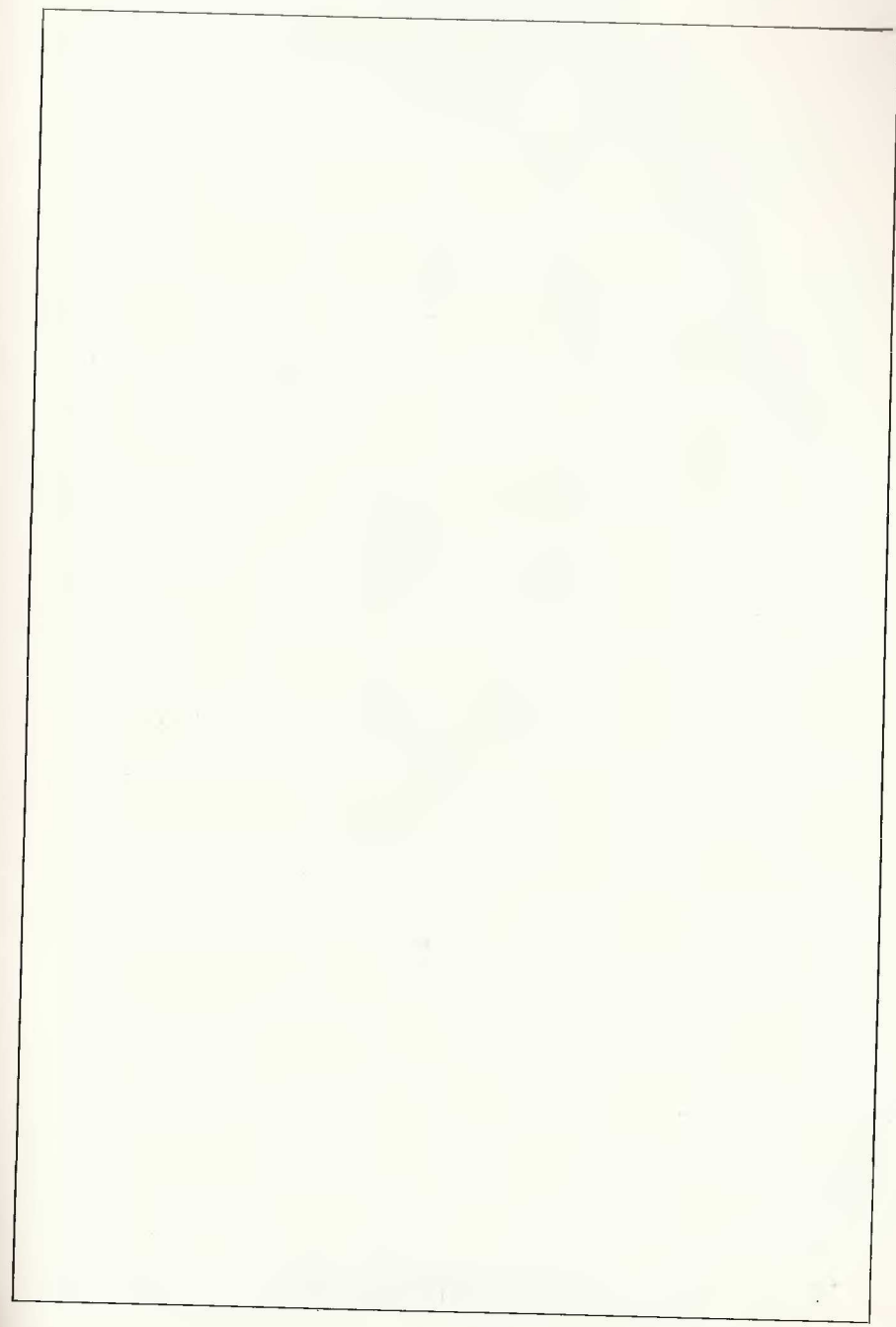
342C
 - Poke 12314, 1: EXEC 54316 initialisation 40 colonnes
 - Poke 12314, 0: EXEC 54316: initialisation 80 colonnes.
 Set et Reset contrôle par boucle = vitesse pour
 set de Rset avec chaque valeur ^{vitesse}
 For U = X to Y: set (N1, N2, N3) \neq Next U
 identique à
 set (N1, N2, N3) \rightarrow set (N1, N2, N3)
 Reminitialisation à froid EXEC 63306 (\$F74A)
 " " " " à chaud EXEC 63278 (\$F72E)
 flèches clavier
 \uparrow $\phi B = 11$
 \rightarrow $\phi A = 08$
 \leftarrow $\phi 8 = 8$
 \downarrow $\phi 9 = 09$
 en mode 40 colonnes - EXEC 54879 (\$D65F) (112)
 \rightarrow Poke 251, A avec A = 0a 255 pour tous les colonnes
 (m. digitolement) plus CLS ϕ à 8
 permet d'initialiser en 40 colonnes.
 Mode CLS32 - CLS40 - CLS80 - CLS81
 10 Poke 12314, A
 20 EXEC 63278 \rightarrow EXEC 53000
 A = 2 \rightarrow CLS32, 1 \rightarrow CLS40, 2 CLS80 et CLS81
 A = 85, 

NOTES

Dans un prog Poke 234, \emptyset \rightarrow action de init = reset
d'abord normalement Poke 234, 127 ou 85

Poke 48929, 248 : Poke 48936, 130 = double hauteur
de caractères.

NOTES



Imprimé en France par
ALSACIENNE D'IMPRESSION Colmar-Paris
Dépôt légal N° 5518 - Septembre 84

